ECE 375 LAB 6

External Interrupts

Lab Time: Wednesday 10-12

Tu Lam

INTRODUCTION

The purpose of the lab is to introduce the external interrupt and ask to set it up and learn how external interrupt work with the BumpBot. The lab also make you display the LCD Driver to convert binary number into ASCII value. The key point of the lab is to get familiar with the interrupt and know how to set other interrupt to not be queue up.

PROGRAM OVERVIEW

The program overall will run the BumpBot to move forward continuously in a loop in the MAIN function. When a trigger is hit, the program stop what it is doing and jump into the interrupt address that the trigger hit and perform it. Once it done, it leave the interrupt and go back to MAIN to resume what it was doing before.

INITIALIZATION ROUTINE

The initialization routine provides a one-time initialization of the stack pointer to set up where to point in the program and it initialize the interrupt mask register and set up PORTB and PORTD as input and output. Beside that, the INIT will initialize the screen of the LCD for it to display.

MAIN ROUTINE

The Main routine executes a simple statement of moving the BumpBot forward continuously forever and ever until a trigger is hit.

HITRIGHT ROUTINE

The HitRight routine will move the BumpBot backward and turn left and resume moving forward. There is also counter that increment the number of times the right trigger is hit and keep track of it.

HITLEFT ROUTINE

The HitLeft routine will move the BumpBot backward and turn right and resume moving forward. There is also counter that increment the number of times the left trigger is hit and keep track of it.

CLRRIGHT ROUTINE

The ClrRight routine will reset a counter to 0 and move it into the Bin2ASCII to convert and send it to the LCD. This is for the right trigger counter.

CLRLEFT ROUTINE

The ClrLeft routine will reset a counter to 0 and move it into the Bin2ASCII to convert and send it to the LCD. This is for the left trigger counter.

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.

Answer: While one of them is a low-language and the other is a high-language, both still do achieve the same result when writing the code for the BumpBot. One level of using assembler is you know the content going into which register while the other one is doing it behind the scene work and you can't control which value and data going into which content. For understandability, learning from assembly would be beneficial as you as the coder understand what going on in your code when you are debugging it while C will optimize it so it hard to understand your code doing the work behind it.

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.

Answer: Using timer/counter for the wait function might be tricky for the BumpBot. As interrupt can handle one interrupt at a time before moving on to the next one. If the wait function was an interrupt and we call the Hit Right interrupt first, the HitRight will perform first then the timer/counter will be next in the queue. That means that base on the priority, the program can't jump to timer/counter for wait while it is performing the other interrupt. So in this case, the answer is no.

CONCLUSION

For Lab #6, the lab helps us to determine how to use and set up the interrupt and how the whole behind the scenes is work. The lab looks at the side from falling edge cases as it is an active low before triggering the input to make it as falling edge. The problem that came up during the program is knowing how to clear the queue interrupt, but the problem is solve when the EIFR is load in when the button is pushed to help clear out queue. But overall, the program performs well and handle the interrupt as expected.

SOURCE CODE

Provide a copy of the source code. Here you should use a mono-spaced font and can go down to 8-pt in order to make it fit. Sometimes the conversion from standard ASCII to a word document may mess up the formatting. Make sure to reformate the code so it looks nice and is readable.

```
;***********************************************************
;*
;*      Tu_Lam_Lab6_SourceCode.asm
;*
;*      Description: This program run on controlling the BumpBot
;*                              To turn left or right using external
;*                              interrupt.
;***********************************************************
;*
;*       Author: Tu Lam
;*         Date: Nov 8th, 2020
;*
;***********************************************************

.include "m128def.inc"                                          ; Include definition
file

;***********************************************************
;*      Internal Register Definitions and Constants
;***********************************************************
.def    mpr = r16                                                       ; Multipurpose
register
.def    waitcnt = r23                                           ; R23-25 is a counter
for loop
.def    olcnt = r24
.def    ilcnt = r25
.def    rcnt = r14                                                      ; Hold the
HitRight counter
.def    lcnt = r13                                                      ; Hold the
HitLeft counter

.equ    Time = 100                                                      ; Wait time for
1 second
.equ    WskrR = 0                                                       ; Right Whisker
Input Bit
.equ    WskrL = 1                                                       ; Left Whisker
Input Bit

;***********************************************************
;*      Start of Code Segment
;***********************************************************
.cseg                                                                   ;
Beginning of code segment

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

                ; Set up interrupt vectors for any interrupts being used
.org    $0002
                rcall   HitRight                                        ; Call HitRight
interrupt
                reti                                                            ; Return
from interrupt

.org    $0004
                rcall   HitLeft                                         ; Call HitLeft
interrupt
                reti                                                            ; Return
from interrupt

.org    $0006
```

```
                rcall   ClrRight                                                ; Call ClrRight
interrupt
                reti                                                                    ; Return
from interrupt

.org    $0008
                rcall   ClrLeft                                                 ; Call ClrLeft
interrupt
                reti                                                                    ; Return
from interrupt

                ; This is just an example:
;.org   $002E                                   ; Analog Comparator IV
;               rcall   HandleAC        ; Call function to handle interrupt
;               reti                                    ; Return from interrupt

.org    $0046                                                                   ; End of
Interrupt Vectors

;***********************************************************
;*      Program Initialization
;***********************************************************
INIT:                                                                           ; The
initialization routine
                ldi             mpr, low(RAMEND)                        ; Initialize Stack
Pointer
                out             SPL, mpr
                ldi             mpr, high(RAMEND)
                out             SPH, mpr

                rcall   LCDInit                                                 ; Initialize the
LCD Screen

                ldi             mpr, (1<<7)|(1<<6)|(1<<5)|(1<<4) ; Initialize Port B for output
                out             DDRB, mpr

                ldi             mpr, (0<<3)|(0<<2)|(0<<1)|(0<<0) ; Initialize Port D for input
                out             DDRD, mpr

                ; Initialize screen to display the First line
                ldi             XL, low(LCDLn1Addr)                             ; Load in the
data memory into X
                ldi             XH, high(LCDLn1Addr)

                ldi             mpr, 0                                                  ; Set
mpr to hold value of 0
                rcall   Bin2ASCII

                ; Initialize screen to display the Second line
                ldi             XL, low(LCDLn2Addr)                             ; Load in the
data memory into X
                ldi             XH, high(LCDLn2Addr)

                ldi             mpr, 0                                                  ; Set
mpr to hold value of 0
                rcall   Bin2ASCII

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

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

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

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

                ; TODO: ???
                rcall   LCDWrite
                ldi             mpr, 0b01100000                         ; Set value to
move forward
                out             PORTB, mpr

                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
;-----------------------------------------------------------
; Below is the code for LEFT, RIGHT, and WAIT FUNCTIONS
;         ------------------------------------

;-----------------------------------------------------------
; Func: Waiting
; Desc: This function help the BumpBot to wait going through
;               a loop of 16 + 159975*waitcnt cycles.
;-----------------------------------------------------------
Waiting:
                push    waitcnt                                         ; Push registers
on the stack

OLOOP:
                ldi             olcnt, 224                                      ; Load
middle-loop with 224

MLOOP:
                ldi             ilcnt, 237                                      ; Load
the inner-loop with 237

ILOOP:
                dec             ilcnt                                           ;
Decrement the inner-loop
                brne    ILOOP                                           ; Continue to
loop inside inner-loop if not reach
                dec             olcnt                                           ;
Decrement the middle-loop
                brne    MLOOP                                           ; Continue to
loop inside middle-loop if not reach
                dec             waitcnt                                         ;
Decrement outer-loop
                brne    OLOOP                                           ; Continue to
loop inside outer-loop if not reach

                pop             waitcnt                                         ; Pop &
restore registers off of stac
                ret
        ; Return the subroutine


;-----------------------------------------------------------
; Func: HitRight
```

```
; Desc: This function help the BumpBot to turn left if the
;               Right whisker is hit and pull up the interrupt.
;-----------------------------------------------------------
HitRight:

COUNTR:
                ldi             XL, low(LCDLn1Addr)                             ; Load in the
data memory into X
                ldi             XH, high(LCDLn1Addr)

                ldi             mpr, 0

                inc             rcnt                                                    ;
Increment rcnt
                add             mpr, rcnt                                               ; Load
the value from rcnt into mpr
                rcall   Bin2ASCII                                               ; Convert number
to ASCII
                rcall   LCDWrite                                        ; Write the number onto
the screen


MOVERA:
                ldi             mpr, 0b00000000                                 ; Move BumpBot
backward
                out             PORTB, mpr
                ldi             waitcnt, Time                                   ; Load the value
of wait time for 1 second
                rcall   Waiting

                ldi             mpr, 0b00100000                                 ; The BumpBot
now turn left
                out             PORTB, mpr
                ldi             waitcnt, Time                                   ; Load 1 second
                rcall   Waiting

                ldi             mpr, 0b00001111                                 ; Clear any
queue interrupt
                out             EIFR, mpr                                               ; Store
clear values into EIFR

                ret
        ; Return the subroutine


;-----------------------------------------------------------
; Func: HitLeft
; Desc: This function help the BumpBot to turn right if the
;               Left whisker is hit and pull up the interrupt.
;-----------------------------------------------------------
HitLeft:

COUNTL:
                ldi             XL, low(LCDLn2Addr)                             ; Load in the
data memory into X
                ldi             XH, high(LCDLn2Addr)

                ldi             mpr, 0

                inc             lcnt                                                    ;
Increment lcnt
                add             mpr, lcnt                                               ; Load
the value from lcnt into mpr
                rcall   Bin2ASCII                                               ; Convert number
to ASCII
                rcall   LCDWrite                                        ; Write the number onto
the screen
```

```
MOVERB:
                ldi             mpr, 0b00000000                                 ; Move BumpBot
backward
                out             PORTB, mpr
                ldi             waitcnt, Time                                   ; Load the value
of wait time for 1 second
                rcall   Waiting

                ldi             mpr, 0b01000000                                 ; The BumpBot
now turn right
                out             PORTB, mpr
                ldi             waitcnt, Time                                   ; Load 1 second
                rcall   Waiting

                ldi             mpr, 0b00001111                                 ; Clear any
queue interrupt
                out             EIFR, mpr                                               ; Store
clear values into EIFR

                ret
        ; Return the subroutine


;-----------------------------------------------------------
; Func: ClrRight
; Desc: This function help to clear the LCD screen and
;               reset the counter.
;-----------------------------------------------------------
ClrRight:

                ldi             XL, low(LCDLn1Addr)                             ; Load in the
data memory into X
                ldi             XH, high(LCDLn1Addr)

                ldi             mpr, 0

                clr             rcnt                                                    ; Clear
register to set it to 0
                add             mpr, rcnt                                               ; Load
the value from lcnt into mpr
                rcall   Bin2ASCII                                               ; Convert number
to ASCII
                rcall   LCDWrite                                        ; Write the number onto
the screen

                ldi             mpr, 0b00001111                                 ; Clear any
queue interrupt
                out             EIFR, mpr                                               ; Store
clear values into EIFR

                ret
        ; Return the subroutine


;-----------------------------------------------------------
; Func: ClrLeft
; Desc: This function help to clear the LCD screen and
;               reset the counter.
;-----------------------------------------------------------
ClrLeft:

                ldi             XL, low(LCDLn2Addr)                             ; Load in the
data memory into X
                ldi             XH, high(LCDLn2Addr)

                ldi             mpr, 0

                clr             lcnt                                                    ; Clear
register to set it to 0
```

```
                add             mpr, lcnt                                                       ; Load
the value from lcnt into mpr
                rcall   Bin2ASCII                                                       ; Convert number
to ASCII
                rcall   LCDWrite                                                ; Write the number onto
the screen

                ldi             mpr, 0b00001111                                         ; Clear any
queue interrupt
                out             EIFR, mpr                                                       ; Store
clear values into EIFR

                ret
        ; Return the subroutine

;-----------------------------------------------------------
; Func: Template function header
; Desc: Cut and paste this and fill in the info at the
;               beginning of your functions
;-----------------------------------------------------------
FUNC:                                                   ; Begin a function with a label

                ; Save variable by pushing them to the stack

                ; Execute the function here

                ; Restore variable by popping them from the stack in reverse order

                ret                                                     ; End a function with RET

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


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