ECE 375 LAB 6

External Interrupts

Lab Time: Tuesday 6-7:50

Aditya Kothari

Mac Santa Cruz

INTRODUCTION

The goal of this lab was to familiarize ourselves with external interrupts and their interrupt vectors. In this lab we re-created the bump bot behavior using vectors.

PROGRAM OVERVIEW

This program enables the TekBot to react to whisker inputs. The TekBot moves forward until one of its whiskers is triggered, causing it to back up a little, turn(change direction) and continue to move forward.

The program contains 5 sub routines: INIT, MAIN, HITRight, HITLeft and wait. These routines combined allow the TekBot to behave like a bump bot.

INITIALIZATION ROUTINE

We first initialize the stack pointer, which allows us to make appropriate functions and subroutine calls. Then we initialize Ports. Port B is initialized to output and Port D is initialized to input. Next, we initialize external interrupts and setup the Interrupt sense control to the falling edge and mask the unwanted buttons. Finally, we turn on the interrupts.

MAIN ROUTINE

The main routine is just an infinite loop that makes the TekBot move forward. This is accomplished by writing Move Forward command on to the PORTB.

HITRIGHT ROUTINE

The HitRight routine causes the TekBot to move backwards for a 1 second by writing Move Backwards command to PORTB, followed by call to the Wait routine. The Turn Left command is written to PORTB followed by another call to the Wait routine. Finally, EIFR is cleared so it doesn't lock the interrupt signal and therefore doesn't create a que of interrupts.

HITLEFT ROUTINE

The HitLeft routine causes the TekBot to move backwards for a 1 second by writing Move Backwards command to PORTB, followed by call to the Wait routine. The Turn right command is written to PORTB followed by another call to the Wait routine. Finally, EIFR is cleared so it doesn't lock the interrupt signal and therefore doesn't create a que of interrupts.

WAIT ROUTINE

The Wait routine is a triply-nested loop that provides a busy cycle of $(16 + 159975 \cdot waitcnt)$ cycles. The waitcnt is set to 100 so the total wait time is 1 sec.

ADDITIONAL QUESTION

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.

It is easier to interpret and write code in C, while it's hard to do so in assembly. The bump bot code written in C was way shorter than the assembly counterpart. In spite of the shorter size of the C program, its code density was way higher than that of assembly program (almost twice as high).

The C program relied on polling, while the assembly program relied on using interrupts. Polling is less efficient way of accomplishing the task, as it makes the processor do more work by making it check the status of the I/O device for interrupts constantly. On the other hand, using interrupts in the Assembly program makes it more efficient as the I/O device automatically notifies the processor if an interrupt is encountered.

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, it is possible to allow the use of timer/counter to perform the 1 second delays.

CONCLUSION

In this lab we learned how to use external interrupts and re-created the bump bot behavior for the TekBot. In addition to that we also learned some key concepts about the external interrupts like: Masking and setting/clearing the interrupt flags.

SOURCE CODE

```
;***********************************************************
;*
;*      Enter Name of file here
;*
;*      Enter the description of the program here
;*
;*      This is the skeleton file for Lab 6 of ECE 375
;*
;***********************************************************
;*
;*       Author: Enter your name
;*         Date: Enter Date
;*
;***********************************************************

.include "m128def.inc"              ; Include definition file

;***********************************************************
;*      Internal Register Definitions and Constants
;***********************************************************
.def    mpr = r16                           ; Multipurpose register
```

```
.def    waitcnt = r17                ; Wait Loop Counter
.def    ilcnt = r18                          ; Inner Loop Counter
.def    olcnt = r19                          ; Outer Loop Counter
.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 forwardward
.equ    MovBck = $00                 ;      Move Backward
.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
                rcall HitRight
                reti

.org    $0004
                rcall HitLeft
                reti
                ; Set up interrupt vectors for any interrupts being used

                ; 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
                ; Initialize Stack Pointer
                ldi mpr, high(RAMEND)
                out SPH, mpr
                ldi mpr, low(RAMEND)
                out SPL, mpr

                ; Initialize Port B for output (ALL LDE's)
                ldi mpr, (1<<EngEnR)|(1<<EngEnL)|(1<<EngDirR)|(1<<EngDirL)
                out DDRB, mpr

                ; Initialize Port D for input
                ldi mpr, (0<<WskrR)|(0<<WskrL)
                out DDRD, mpr
                ldi mpr, (1<<WskrR)|(1<<WskrL)
                out PORTD, mpr                                  ;       Enables pull up register
                ; Initialize external interrupts
                ldi  mpr,  (1<<ISC01)|(0<<ISC00)|(1<<ISC11)|(0<<ISC10);  Set  the  Interrupt  Sense
Control to falling edge
                sts EICRA, mpr
                ; Configure the External Interrupt Mask
                ldi mpr, (1<<INT0)|(1<<INT1)
                out EIMSK, mpr
                ; Turn on interrupts
```

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

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

                ; TODO: ???
                ldi mpr, MovFwd
                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
;-----------------------------------------------------------

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

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

                push    mpr                     ; Save mpr register
                push    waitcnt                 ; Save wait register
                in              mpr, SREG       ; Save program state
                push    mpr                     ;

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

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

                ; Move Forward again
                ldi             mpr, MovFwd     ; Load Move Forward command
                out             PORTB, mpr      ; Send command to port

                pop             mpr             ; Restore program state
                out             SREG, mpr       ;
                pop             waitcnt         ; Restore wait register
                pop             mpr             ; Restore mpr
                LDI             R20, 0x03       ; writing 1 to
                OUT             EIFR, R20       ; clear EIFR

                ret                             ; Return from subroutine

;-----------------------------------------------------------------
; Sub:  HitLeft
```

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

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

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

               ; Move Forward again
               ldi             mpr, MovFwd    ; Load Move Forward command
               out             PORTB, mpr     ; Send command to port

               pop             mpr            ; Restore program state
               out             SREG, mpr      ;
               pop             waitcnt        ; Restore wait register
               pop             mpr            ; Restore mpr
               LDI             R20, 0x03      ; writing 1
               OUT             EIFR, R20      ; clear EIFR
               ret                            ; Return from subroutine

;-----------------------------------------------------------
; Sub:  Wait
; 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
;-----------------------------------------------------------
Wait:
               push    waitcnt               ; 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             waitcnt        ; Decrement wait
               brne    Loop                   ; Continue Wait loop

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

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

; Enter any stored data you might need here

;***********************************************************
;*      Additional Program Includes
;***********************************************************
; There are no additional file includes for this program
```