ECE 375 Lab 6

Timers/Counters

Lab Time: Friday 2 PM - 3:50 PM

Student 1: Winnie Woo
Student 2: Joseph Borisch

TA Signature

1 Introduction

The purpose of this lab was to learn how to configure and use 16-bit timers/counters on the ATMEGA 32u4 Tekbot to generate pulse width modulation (PWM) signals. As well as use the upper and lower byte of an I/O port for different tasks. Additionally, the Tekbot will be configured so the speed levels can be modified.

2 Program Overview

This program provides the behavior that allows the user to control the speed of the Tekbot, with sixteen equidistant speed levels. Speed 0 being completely stopped and Speed 15 being maximum speed. One button press will result in a single action, i.e. SPEED_UP will only increase the speed by one level.
Besides the INIT and MAIN routines within the program, three additional routines were created and used. The SPEED_UP routine increases the speed by one level. The SLOW_DOWN routine decreases the speed by one level. The SPEED_MAX routine immediately increases the speed to the highest level (Speed 15).

3 Initialization Routine

The INIT routine initializes the stack pointer, PORT B and PORT D for output, as well as the interrupt sense control (EICRA) trigger state to falling edge and configuring the external interrupt mask (EIMSK) to enable INT0, INT1, INT3. The 16-bit timer/counter (TCCR1A and TCCR1B) is also configured. Lastly the global interrupt is set.

4 Main Routine

The Main routine executes an infinite move forward command on PORT B.

5 SPEED_UP

This function increases the speed by one level. First we are saving mpr and the program state. Then the function is checking to see if we are the maximum speed level (speed 15) and if at the maximum speed then it will skip the increase of speed level and break. If not, then add 17 to the pulse width modulation speed to increase by one speed level, and increment the upper nibble of PORT B. Lastly restore the program state.

6 SLOW_DOWN

This function decreases the speed by one level. It checks for if it is at the lowest speed (speed 0) and if so then skip the decrease of speed level and break. If not, then subtract 17 to decrease the speed by one speed level and decrement the last nibble of PORT B. Lastly restore the program state.

7 SPEED_MAX

This function increases the speed to the max level. We are saving mpr and the program state. Then set the speed level to the maximum speed level (speed 15) and save the upper nibble of Port B. Lastly, clear the queued interrupts and restore the program state.

8 Additional Questions

1. In this lab, you used the Fast PWM mode of 16-bit Timer/Counter, which is only one of many possible ways to implement variable speed on a TekBot. Suppose instead that you used Normal mode, and had it generate an interrupt for every overflow. In the overflow ISR, you manually toggled both Motor Enable pins of the TekBot, and wrote a new value into the Timer/Counter's register. (If you used the correct sequence of values, you would be manually performing PWM.) Give a detailed assessment (in 1-2 paragraphs) of the advantages and disadvantages of this new approach, in comparison to the PWM approach used in this lab.

   Using the normal mode of the timer/ counter would have several advantages and disadvantages compared to using the PWM mode. One advantage of using normal mode is the fact that the compare register interrupt is immediate, where for PWM it is only set at the top of the count. This allows for faster interrupt control. There are several disadvantages that come with using normal mode in place of the PWM mode. To start, normal mode requires more thought out implementation and planning to set the PWM. In addition, using normal mode does not have the ability of adjusting the duty cycle of the waveform, only the frequency. Finally, normal mode requires the use of the OC0 pin in for the waveform, but OC0 cannot retain it's value while counting. Therefore it must be loaded into a separate register and reloaded.

2. The previous question outlined a way of using a single 16-bit Timer/Counter in Normal mode to implement variable speed. How would you accomplish the same task (variable TekBot speed) using in CTC mode? Provide a rough-draft sketch of the Timer/Counter-related parts of your design, using either a flow chart or some pseudocode (but not actual assembly code).

   The implementation of PWM with the CTC mode of operation would be very similar to that of the normal mode implementation. The two work in almost identical ways, however CTC mode has the advantage of not having to reset the OC0 pin when counting. Example of pseudocode for CTC based PWM:
   1) Normal operation (move forward)
   2) Interrupt triggered at overflow
   3) Check if button has been pressed
   4) Handle button functionality (speed up, slow down, max speed)
   5) return to program, begin next pulse

9 Difficulties

The biggest difficulty was the time constraint since we had a veterans day weekend and did not have the lab checkoff till Monday.

10 Conclusion

In conclusion, the lab overall was not too difficult. The resources provided were helpful and it was a good lab to learn about how to implement timers/counters.

11 Source Code

```
;***********************************************************
;*
;* This is the skeleton file for Lab 6 of ECE 375
;*
;*  Author: Winnie Woo and Joseph Borisch
;*  Date: 11/16/2022
;*
;***********************************************************

.include "m32U4def.inc"    ; Include definition file

;***********************************************************
;* Internal Register Definitions and Constants
;***********************************************************
.def  mpr = r16          ; Multipurpose register
.def  PWMLevel = r17     ; register for tracking duty cycle
.def  SpeedDisplay = r18    ; register for tracking/ displaying current speed
.def  SpeedInc = r19     ; value to increase the speed by
.def  waitcnt = r20
.def  ilcnt = r21
.def  olcnt = r22

.equ  WTime = 25
.equ  EngEnR = 5         ; right Engine Enable Bit
.equ  EngEnL = 6         ; left Engine Enable Bit
.equ  EngDirR = 4        ; right Engine Direction Bit
.equ  EngDirL = 7        ; left Engine Direction Bit
.equ  MovFwd = (1<<EngDirR|1<<EngDirL) ; instruction to make TekBot move forward
;***********************************************************
;* Start of Code Segment
;***********************************************************
.cseg                  ; beginning of code segment

;***********************************************************
;* Interrupt Vectors
```

```
;***********************************************************
.org $0000
     rjmp INIT       ; reset interrupt

.org $0002
     rcall SPEED_UP     ; interrupt to speed up tekbot one level
     reti

.org $0004
     rcall SLOW_DOWN       ; interrupt to slow tekbot down one level
     reti

.org $0008
     rcall SPEED_MAX       ; interrupt to set tekbot to max speed
     reti

.org $0056           ; end of interrupt vectors

;***********************************************************
;* Program Initialization
;***********************************************************
INIT:
     ; Initialize the Stack Pointer
     ldi mpr, low(RAMEND) ; Low byte of END SRAM addr
     out SPL, mpr       ; Write byte to SPL
     ldi mpr, high(RAMEND) ; high byte of END sram addr
     out SPH, mpr       ; Write byte to SPH

     ; Configure I/O ports
     ; Initialize Port B for output
     ldi mpr, $FF       ; Set Port B Data Direction Register
     out DDRB, mpr      ; for output
     ldi mpr, $00       ; Initialize Port B Data Register
     out PORTB, mpr     ; so all Port B outputs are low

     ; Initialize Port D for input
     ldi mpr, $00       ; Set Port D Data Direction Register
     out DDRD, mpr      ; for input
     ldi mpr, $FF       ; Initialize Port D Data Register
     out PORTD, mpr     ; so all Port D inputs are Tri-State

     ; Configure External Interrupts, if needed
     ldi     mpr, 0b10001010 ;
        (1<<ISC01)|(0<<ISC00)|(1<<ISC11)|(0<<ISC10)|(1<<ISC31)|(0<<ISC30) => sets
        trigger state to falling
     sts     EICRA, mpr    ; use sts, EICRA is in extended I\O space

     ; Configure the External Interrupt Mask
```

```
      ldi     mpr, 0b00001011 ; (1<<INT0)|(1<<INT1)|(1<<INT3) => enables INT0, INT1,
          INT3
      out     EIMSK, mpr
      ; Configure 16-bit Timer/Counter 1A and 1B
      ldi     mpr, 0b11110001       ; Fast PWM, 8-bit mode (WGM11:10)
      sts     TCCR1A, mpr        ; / inverting mode (COM1A1:COM1A0 and COM1B1:COM1B0)
      ldi     mpr, 0b00001001       ; Fast PWM, 8-bit mode (WGM13:WGM12)
      sts     TCCR1B, mpr        ; / no prescale (CS12:CS10)
      ; Set TekBot to Move Forward (1<<EngDirR|1<<EngDirL) on Port B
      ldi     mpr, MovFwd
      out     PORTB, mpr
      ; Set value for speed to be increased by (255/15 =17)
      ldi     SpeedInc, $11
      ; Set initial speed, display on Port B pins 3:0
      clr     SpeedDisplay
      or    mpr, SpeedDisplay  ; load speed level into lower mpr / save MovFwd command
      out     PORTB, mpr
      clr     PWMLevel        ; initially 0
      clr     mpr
      sts     OCR1AH, mpr        ;
      sts     OCR1AL, PWMLevel   ; Set compare vlaue for new pulse speed
      sts     OCR1BH, mpr        ;
      sts     OCR1BL, PWMLevel   ; Set compare vlaue for new pulse speed
      sbi     DDRB, PB5          ; turn on portb pin 5 for changing brightness
      sbi     DDRB, PB6          ; turn on portb pin 6 for changing brightness
      ; Enable global interrupts (if any are used)
      sei
;***********************************************************
;* Main Program
;***********************************************************
MAIN:
      ldi     mpr, MovFwd        ; load move forward command
      or    mpr, SpeedDisplay  ; load speed level without destroying MovFwd
      out     PORTB, mpr         ; Update PORTB
      rjmp  MAIN            ; return to top of MAIN

;***********************************************************
;* Functions and Subroutines
;***********************************************************
;-----------------------------------------------------------
; Func: SPEED_UP
; Desc: Increase speed by one level
;-----------------------------------------------------------
SPEED_UP:  ; Begin a function with a label

      ; If needed, save variables by pushing to the stack
      push  mpr               ; save mpr
      in    mpr, SREG         ; save program state
      push  mpr
```

```
     push  SpeedInc

     rcall wait            ; make sure the speed increases by only 1

     cpi     SpeedDisplay, $0F  ; Check if current speed = max (15)
     breq  INCSKIP            ; Skip increment if max speed
     add     PWMLevel, SpeedInc    ; increase Tekbot speed 255/15 = 17
     inc     SpeedDisplay       ; increase speed by one level

     clr     mpr
     sts     OCR1AH, mpr        ;
     sts     OCR1AL, PWMLevel   ; Set compare vlaue for new pulse speed
     sts     OCR1BH, mpr        ;
     sts     OCR1BL, PWMLevel   ; Set compare vlaue for new pulse speed

     clr     mpr                ; mpr all zeros
     in    mpr, PORTB         ; Save PORTB data
     or    mpr, SpeedDisplay  ; set new speed level
     out     PORTB, mpr         ; display new speed level
INCSKIP:
     ; Clear queued interrupts
     ldi     mpr, $0F        ; Cleared by writing a 1 to it
     out     EIFR, mpr

     ; Restore any saved variables by popping from stack
     pop     SpeedInc
     pop     mpr
     out     SREG, mpr
     pop     mpr

     ret                     ; End a function with RET
;-----------------------------------------------------------
; Func: SLOW_DOWN
; Desc: Decrease speed by one level
;-----------------------------------------------------------
SLOW_DOWN: ; Begin a function with a label

     ; If needed, save variables by pushing to the stack
     push  mpr               ; save mpr
     in    mpr, SREG         ; save program state
     push  mpr
     push  SpeedInc

     rcall wait            ; make sure the speed increases by only 1

     cpi     SpeedDisplay, $00  ; Check if current speed = min (0)
     breq  DECSKIP            ; Skip decrement if min speed
     sub     PWMLevel, SpeedInc    ; decrease tekbot speed by 17
     dec     SpeedDisplay       ; decrease speed one level
```

```
	clr	mpr
	sts	OCR1AH, mpr	;
	sts	OCR1AL, PWMLevel	; Set compare vlaue for new pulse speed
	sts	OCR1BH, mpr	;
	sts	OCR1BL, PWMLevel	; Set compare vlaue for new pulse speed

	clr	mpr		; mpr all zeros
	in	mpr, PORTB	; Save PORTB data
	eor	mpr, SpeedDisplay	; set new speed level
	out	PORTB, mpr	; display new speed level
DECSKIP:
	; Clear queued interrupts
	ldi	mpr, $0F ; Cleared by writing a 1 to it
	out	EIFR, mpr
	; Restore any saved variables by popping from stack
	pop	SpeedInc
	pop	mpr
	out	SREG, mpr
	pop	mpr

	ret			; End a function with RET
;-----------------------------------------------------------
; Func: SPEED_MAX
; Desc: Increase speed to max level
;-----------------------------------------------------------
SPEED_MAX: ; Begin a function with a label

	; If needed, save variables by pushing to the stack
	push mpr		; save mpr
	in	mpr, SREG	; save program state
	push mpr
	push SpeedInc

	ldi	SpeedDisplay, $0F	; load max speed value (15) into display reg
	ldi	PWMLevel, $FF	; 0% duty cycle for max speed
	clr	mpr
	sts	OCR1AH, mpr	;
	sts	OCR1AL, PWMLevel	; Set compare value for new pulse speed
	sts	OCR1BH, mpr	;
	sts	OCR1BL, PWMLevel	; Set compare value for new pulse speed
	clr	mpr		; mpr all zeros
	in	mpr, PORTB	; Save PORTB data
	or	mpr, SpeedDisplay	; set new speed level
	out	PORTB, mpr	; display new speed level
	; Clear the queued interrupts
	ldi	mpr, $0F
	out	EIFR, mpr
	; Restore any saved variables by popping from stack
```

```
      pop     SpeedInc
      pop     mpr
      out     SREG, mpr
      pop     mpr

      ret               ; End a function with RET

;-----------------------------------------------------------------
; 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

      ldi     waitcnt, WTime   ; Load time to delay


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
```