ECE 375 LAB 7

Timer/Counter

Lab Time: Wednesday 12-2

Bradley Martin

INTRODUCTION

The Purpose of this lab is to use the timer/counters on the board to make PWM signals that control how fast or slow the tekbot is moving. The tekbot should have 16 different speeds that will be able to change via the buttons on the board. We will use interrupt vectors to signal when a button has been pushed.

PROGRAM OVERVIEW

This program allows the tekbot to have a variable speed with 16 steps. It ranges from step 1 being a not moving to step 16 being fully on. Each step increases the tekbots speed by 6.66%. There are also buttons for complete stop and full power.

Besides the standard INIT and MAIN routines there are also routines for Speed_Up, Speed_Down, Speed_Max, Speed_Min, and Wait. Speed up/down increase or decrease a step in speed and update those values on the led array. Speed max/min set the speed to full speed or stop and update those values on the led array. The wait routine allows us to create an accurate busy wait to allow for debouncing of the buttons.

INITIALIZATION ROUTINE

The initialization routine provides a one-time initialization of key registers that allow the program to execute correctly. First the Stack Pointer is initialized, allowing the proper use of function and subroutine calls. Port B was initialized to all outputs and will be used to direct the motors. Port D was initialized to inputs. Next, we set the interrupt controls for falling edge and initialize the interrupt mask for the first 4 interrupts. Next is setting up the timers with the settings said in the lab procedure and store those setting into the timers. We also need to initialize the compare values to 0. Last we set the tekbot to move forward initially and set up the led array.

MAIN ROUTINE

The Main routine is very simple that only has a jump back to main routine because are input is done through interrupts.

SPEED_UP ROUTINE

The Speed_ up routine first checks to see if the speed is already at the max value. If we are already at the max value then we will skip trying to increase the speed, but if we are not then we need to increase the speed by 17 and 1 step. Then update the led array.

SPEED_DOWN ROUTINE

The Speed_down routine works the same as the Speed_down routine except for minimum values. Checking if we are at minimum speed and if not decreasing the speed.

SPEED_MAX ROUTINE

The Speed Max routine loads the max speed values and puts those into the speed variable. It also sets what speed step we are on to the last one. It then updates the led array with the maxed out values.

SPEED_MIN ROUTINE

The Speed Min routine loads the minimum speed value and loads it into the speed variable. It also sets what speed step we are on to the last one. It then updates the led array with the minimum values.

WAIT ROUTINE

The Wait routine requires a single argument provided in the *waitcnt* register. A triple-nested loop will provide busy cycles as such that $16 + 159975 \cdot waitcnt$ cycles will be executed, or roughly $waitcnt \cdot 10\text{ms}$. In order to use this routine, first the *waitcnt* register must be loaded with the number of 10ms intervals, i.e. for one second, the *waitcnt* must contain a value of 100. Then a call to the routine will perform the precision wait cycle.

ADDITIONAL QUESTIONS

1) In this lab, you used the Fast PWM mode of both 8-bit Timer/Counters, which is only one of many possible ways to implement variable speed on a TekBot. Suppose instead that you used just one of the 8-bit Timer/Counters in 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.

An advantage to this way is that we only need one time/counter. This frees up how many processes the board is doing which could make it slightly more efficient. However, the main disadvantage is that now It is harder to run the motors separately if we wanted them both to be at different speeds, say for turning. We might use up all our efficiency we saved by just trying to get them to run separately.

2) The previous question outlined a way of using a single 8-bit Timer/Counter in Normal mode to implement variable speed. How would you accomplish the same task (variable TekBot speed) using one or both of the 8- bit Timer/Counters 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).

-Set up 2 CTC timers/counters

-wait till overflow for interrupt

-check tcnt for what step we are on

-check both compare values with each other (decide if they are different, they should not be for moving forward)

-increase speed, and update leds

CONCLUSION

In this we used the timer/counters to implement a program that takes in button input to control the speed of the tekbot. This lab was a good introduction to accessing the variables associated with the timer/coutners and how we can use them for counting. The hardest part of this lab for me was not even with the counters, it was how to access certain parts of our speed value and only update those bits.

in this lab, we were simply required to set up an AVRStudio4 project with an example program, compile this project and then download it onto our TekBot bases. The result of this program allowed the TekBot to behave in a

SOURCE CODE

```
;***********************************************************
;*
;*      Bradley_Martin_sourcecode
;*
;*      Program to take button input to change the speed of the tekbot.
;*
;***********************************************************
;*
;*       Author: Bradley Martin
;*         Date: 11/16/2020
;*
;***********************************************************

.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
.def    Speed = r20                             ; holds the value for what step we are on for lower bits
.def    SpeedStep = r21                 ; holds the value of what speed percent we are on for the higher
bits
.def    temp = r22

.equ    WTime = 10                              ; time to wait in loop

.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)      ; enable tekbot to move forward

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

;***********************************************************
;*      Interrupt Vectors
;***********************************************************
.org    $0000
                rjmp    INIT                    ; reset interrupt

.org    $0002
                rcall Speed_Up                  ; button for speeding up the tekbot
                reti

.org    $0004
                rcall Speed_Down        ; button for slowing down the tekbot
                reti

.org    $0006
                rcall Speed_Max                 ; button for max speed
                reti

.org    $0008
                rcall Speed_Min                 ; button for min speed
```

```
                reti

.org    $0046                                   ; end of interrupt vectors

;***********************************************************
;*      Program Initialization
;***********************************************************
INIT:
                ; Initialize the Stack Pointer

                ldi             mpr, low(RAMEND)
                out             SPL, mpr
                ldi             mpr, high(RAMEND)
                out             SPH, mpr

                ; 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

                ; Set the Interrupt Sense Control to falling edge
                ldi             mpr, (1<<ISC01) | (0<<ISC00) | (1<<ISC11)| (0<<ISC10)
                sts             EICRA, mpr

                ; Set the External Interrupt Mask
                ldi             mpr, (1<<INT0) | (1<<INT1) | (1<<INT2) | (1<<INT3)
                out             EIMSK, mpr

                ; Configure 8-bit Timer/Counters

                ldi             mpr, 0b01101001         ; initial settings
                out             TCCR0, mpr                      ; set timer0 to settings
                out             TCCR2, mpr                      ; set timer2 to settings

                ldi             mpr, $00                ; load 0 into mpr for compare value
                sts             OCR0, mpr                       ; set compare to 0
                sts             OCR2, mpr                       ; set compare to 0

                ; Set TekBot to Move Forward (1<<EngDirR|1<<EngDirL)

                ldi             mpr, MovFwd             ; set mpr to movefwd command
                out             PORTB, mpr              ; update the leds

                ; Set initial speed, display on Port B pins 3:0
                clr             Speed                           ; Starting at speed 0
                clr             SpeedStep

                out             OCR0, Speed                     ; update compare values
                out             OCR2, Speed

                in              mpr, PORTB                      ; store the current state of the leds

                cbr             mpr, 0b00001111         ; clear the lower half leds
                or              mpr, SpeedStep          ; perform or on upper half so only leds 5/8
change

                out             PORTB, mpr                      ; update leds
```

```
                ; Enable global interrupts (if any are used)
                sei
;***********************************************************
;*      Main Program
;***********************************************************
MAIN:
                rjmp    MAIN                    ; return to top of MAIN

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

;-----------------------------------------------------------
; Func: Speed_up
; Desc: Increses the speed by 17 and increments the speed step.
;-----------------------------------------------------------
Speed_Up:       ; Begin a function with a label
                mov             mpr, Speed      ; get the current speed
                cpi             mpr, $FF; check to see if we are at max speed
                breq    MaxSpeed; If we are then skip

                ldi             temp, $11       ; if we are not then load 17 into mpr
                add             Speed, temp     ; add a step to speed
                inc             r21                     ; increade what step we are on

                out             OCR0, Speed     ; update compare values
                out             OCR2, Speed

                in              mpr, PORTB      ; store the current state of the leds

                cbr             mpr, 0b00001111         ; clear the lower half leds
                or              mpr, SpeedStep          ; perform or on upper half so only leds 5/8
change

                out             PORTB, mpr                      ; update leds

MaxSpeed:
                rcall   Wait                            ; wait for debounce
                ldi             mpr, 0b11111111         ; Set mpr to all high
                out             EIFR, mpr                       ; Reset EIFR with all high to clear
interrupts

                sei
                ret                                             ; End a function with RET

;-----------------------------------------------------------
; Func: Speed_Down
; Desc: Decreses the speed by 17 and decrements the speed step.
;-----------------------------------------------------------
Speed_Down:     ; Begin a function with a label

                mov             mpr, Speed      ; get the current speed
                cpi             mpr, $00; check to see if we are at min speed
                breq    MinSpeed; If we are then skip

                subi    Speed, $11      ; Subtract 17 from speed
                dec             SpeedStep       ; decrement what step we are on

                out             OCR0, Speed     ; update compare values
                out             OCR2, Speed

                in              mpr, PORTB      ; store the current state of the leds

                cbr             mpr, 0b00001111 ; clear the lower half leds
                or              mpr, SpeedStep  ; perform or on upper half so only leds 5/8 change

                out             PORTB, mpr              ; update leds

MinSpeed:
```

```
                rcall   Wait                            ; wait for debounce

                ldi             mpr, 0b11111111         ; Set mpr to all high
                out             EIFR, mpr                       ; Reset EIFR with all high to clear
interrupts

                sei
                ret                                             ; End a function with RET

;-----------------------------------------------------------
; Func: Speed_Max
; Desc: Sets speed and speed step to max.
;-----------------------------------------------------------
Speed_Max:      ; Begin a function with a label

                ldi             mpr, $FF                        ; load max speed into mpr
                mov             Speed, mpr                              ; move that value into speed
                ldi             SpeedStep, 0b00001111   ; set the speed step to max

                out             OCR0, Speed                             ; upadate compare values
                out             OCR2, Speed

                in              mpr, PORTB                              ; store the current state of the
leds

                cbr             mpr, 0b00001111                 ; clear the lower half leds
                or              mpr, SpeedStep                  ; perform or on upper half so only leds
5/8 change

                out             PORTB, mpr                              ; update leds

                ldi             mpr, 0b11111111         ; Set mpr to all high
                out             EIFR, mpr                       ; Reset EIFR with all high to clear
interrupts


                ret                                             ; End a function with RET

;-----------------------------------------------------------
; Func: Speed_Min
; Desc: Sets speed and speed step to 0.
;-----------------------------------------------------------
Speed_Min:      ; Begin a function with a label

                ldi             mpr, $00                        ; load min speed into mpr
                mov             Speed, mpr                              ; move that value into speed
                ldi             SpeedStep, 0b00000000   ; set the speed step to min

                out             OCR0, Speed                             ;update compare values
                out             OCR2, Speed

                in              mpr, PORTB                              ; store the current state of the
leds

                cbr             mpr, 0b00001111                 ; clear the lower half leds
                or              mpr, SpeedStep                  ; perform or on upper half so only leds
5/8 change

                out             PORTB, mpr                              ; update leds

                ldi             mpr, 0b11111111                 ; Set mpr to all high
                out             EIFR, mpr                               ; Reset EIFR with all high to
clear interrupts

                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

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
```