# ECE 375 LAB 7

Introduction to AVR Development Tools

**Lab Time: Friday 2-4**

*Hao Truong*

## INTRODUCTION

Timer/Counters are one of the most used components of a microcontroller which can be used to measure time or toggle an I/O pin after certain amount of time has passed. The purpose of this lab is to provide an understanding on the 8-bit Timer/Counters. We will learn how to configure and use the 8-bit Timer/Counters on the ATmega128 to generate pulse-width modulation (PWM) signals. We will also learn how to modify PWM duty cycle after initial configuration.

## PROGRAM OVERVIEW

The Timer/Counters program allows user to modify the TekBot speed based on input via port B. There are 16 levels of speed, with speed level 0 being completely stopped, speed level 15 being full speed, speed levels 1 through 14 are in between. The TekBot initially moves forward with full speed (speed level 15). When INT0 is triggered, the speed level of the TekBot decreases by 1. Similarly, when INT1 is triggered, the speed level of the TekBot increases by 1. When INT2 is triggered, the TekBot immediately decreases its speed to the minimum speed (speed level 0). When INT3 is triggered, the TekBot immediately increases its speed to the maximum level (speed level 15).

### INITIALIZATION ROUTINE

The initialization routine provides a one-time initialization of key registers that allow the program to execute correctly. The Stack Pointer is initialized, allowing the proper use of function and subroutine calls. Port B was configured as output and Port D was configured as input. The external interrupts and 8-bit Timer/Counters were also initialized. The TekBot was set to move forward with a maximum speed.

### MAIN ROUTINE

The Main routine executes a simple loop to loop around it infinitely.

### SUBROUTINES

1. SPEED_DOWN Routine

The SPEED_DOWN routine decreases the speed of the TekBot by just one level. The speed will not wrap around. This means that when the speed is at MIN, the speed will stay at this level when SPEED_DOWN is called.

2. SPEED_UP Routine

The SPEED_UP routine increases the speed of the TekBot by just one level. when the speed is at MAX, the speed will stay at this level when SPEED_UP is called.

3. SPEED_MIN Routine

The SPEED_MIN routine immediately decreases the speed of the TekBot to speed level 0 (stopped).

4. SPEED_MAX routine

The SPEED_MAX routine immediately increases the speed of the TekBot to speed kevel 15 (full speed)

## 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.*

The disadvantage of doing this in normal mode is that we are required to write a new correct value into the Timer/Counter register every time an overflow occurs. The advantage is that it is faster since it counts from loaded value to MAX in normal mode. With PWM, it counts from Bottom to Max.

*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).*

- Initialize TNCT0 and activate CTC mode with no prescale (TCCR0 = 00001001)
- Other routines would remain the same.

## CONCLUSION

In this lab, we learned how to configure and use the 8-bit Timer/Counters on the ATmega128 to generate pulse-width modulation (PWM) signals. We will also learned how to modify PWM duty cycle after initial configuration. The lab allowed us to modify the speed of the TekBok based on input via port B.

## 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.

```
;***********************************************************
;*
;*	 Hao_Truong_Lab7_sourcecode.asm
;*
;*	 Timer/Counters
;*
;*	 This is the skeleton file for Lab 7 of ECE 375
;*
;***********************************************************
;*
;*	  Author: Hao Truong
;*	    Date: 11/19/2021
;*
;***********************************************************

.include "m128def.inc"			; Include definition file

;***********************************************************
```

```
;*      Internal Register Definitions and Constants
;***********************************************************
.def    mpr = r16                           ; Multipurpose register
.def    speed_level = r17            ; speed level
.def    tmp = r18

.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

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

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

                ; place instructions in interrupt vectors here, if needed
.org    $0002                               ; INT0
                rcall   SPEED_DOWN          ; service INT0
                reti                                ; return from INT0

.org    $0004                               ; INT1
                rcall   SPEED_UP            ; service INT1
                reti                                ; return from INT1

.org    $0006                               ; INT2
                rcall   SPEED_MIN           ; service INT2
                reti                                ; return from INT2

.org    $0008                               ; INT3
                rcall   SPEED_MAX           ; service INT3
                reti                                ; return from INT3

.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
                ldi             mpr, 0b11111111             ; initialize Port B for output
                out             DDRB, mpr                   ; set Port B directional register for
output

                ldi             mpr, 0b00000000             ; initialize Port D for input
                out             DDRD, mpr                   ; set Port D directional register for
input
                ldi             mpr, 0b00001111
                out             PORTD, mpr                  ; activate pull-up registers

                ; Configure External Interrupts, if needed
                ; Set the Interrupt Sense Control to falling edge
                ldi             mpr, 0b10101010
                sts             EICRA, mpr
                ; Set up the external interrupt mask
                ldi             mpr, 0b00001111
                out             EIMSK, mpr
```

```
               ; Configure 8-bit Timer/Counters
               ldi             mpr, 0b01111001               ; activate Fast PWM mode with toggle
               out             TCCR0, mpr                    ; no prescaling, inverting
               out             TCCR2, mpr

               ; Set TekBot to Move Forward (1<<EngDirR|1<<EngDirL), with max speed
               ldi             speed_level, 0b01101111
               out             PORTB, speed_level
               ldi             mpr, 255
               out             OCR0, mpr                                              ;         write
speed_level to OCR0
               out             OCR2, mpr                                              ;         write
speed_level to OCR2


               ; Set initial speed, display on Port B pins 3:0

               ; Enable global interrupts (if any are used)
               sei                                           ; turn on interrupt

;***********************************************************
;*      Main Program
;***********************************************************
MAIN:
               ; poll Port D pushbuttons (if needed)

                                                             ; if pressed, adjust speed
                                                             ; also, adjust speed indication

               rjmp    MAIN                  ; return to top of MAIN

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

;-----------------------------------------------------------
; Sub:  SPEED_DOWN
; Desc: decrease speed by just one level
;
;-----------------------------------------------------------
SPEED_DOWN:
               in              mpr, OCR0                     ; read OCR0
               cpi             mpr, 0                        ; checks if speed level is 0
               breq    SKIP                          ; speed level is 0, skip
               subi    mpr, 17                       ; subtract 17 from mpr (levels are 17 apart)
               out             OCR0, mpr                     ; write speed level to OCR0
               out             OCR2, mpr                     ; write speed level to OCR2
               dec             speed_level                   ; decrement speed_level
               out             PORTB, speed_level    ; display speed_level on PORT B

SKIP:
               ; Avoid queued interrupts by writing 1 to EIFR
               ldi             mpr, 0b00001111
               out             EIFR, mpr
               ret

;-----------------------------------------------------------
; Sub:  SPEED_UP
; Desc: increase speed by just one level
;-----------------------------------------------------------
SPEED_UP:
               in              mpr, OCR2                     ; read OCR2
               cpi             mpr, 255                      ; checks if speed level is 15 (MAX)
               breq    SKIP1                         ; speed level is 0, skip
               ldi             tmp, 17
               add             mpr, tmp                      ; subtract 17 from mpr (levels are 17
apart)
               out             OCR0, mpr                     ; write speed level to OCR0
               out             OCR2, mpr                     ; write speed level to OCR2
               inc             speed_level                   ; decrement speed_level
               out             PORTB, speed_level    ; display speed_level on PORT B
```

```
SKIP1:
               ; Avoid queued interrupts by writing 1 to EIFR
               ldi            mpr, 0b00001111
               out            EIFR, mpr
               ret

;-----------------------------------------------------------
; Sub:  SPEED_MIN
; Desc: immediately decrease speed to lowest level (stopped)
;-----------------------------------------------------------
SPEED_MIN:
               in             mpr, OCR0                    ; read OCR0
               mov            tmp, mpr                     ; copy mpr to tmp
               sub            mpr, tmp                     ; subtract tmp from mpr (speed is now
0% -> speed level is 0)
               out            OCR0, mpr                    ; write speed level to OCR0
               out            OCR2, mpr                    ; write speed level to OCR2
               ldi            mpr, 0b11110000
               out            PORTB, mpr                   ; Tekbot halts

               ; Avoid queued interrupts by writing 1 to EIFR
               ldi            mpr, 0b00001111
               out            EIFR, mpr
               ret

;-----------------------------------------------------------
; Sub:  SPEED_MAX
; Desc: immediately increase speed to hoghest level (max)
;-----------------------------------------------------------
SPEED_MAX:


;              ldi            tmp, 255                     ; load 255 to mpr      (255   is   full
speed)
;              sub            tmp, mpr
;              add            mpr, tmp
               ldi            mpr, 255
               out            OCR0, mpr                    ; write speed level to OCR0
               out            OCR2, tmp                    ; write speed level to OCR2
               ldi            speed_level, 0b01101111
               out            PORTB, speed_level                   ; Tekbot moves forward at max
speed

               ; Avoid queued interrupts by writing 1 to EIFR
               ldi            mpr, 0b00001111
               out            EIFR, mpr
               ret

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

               ; If needed, save variables by pushing to the stack

               ; Execute the function here

               ; Restore any saved variables by popping from stack

               ret                                         ; End a function with RET

;***********************************************************
;*     Stored Program Data
;***********************************************************
               ; Enter any stored data you might need here

;***********************************************************
;*     Additional Program Includes
```

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