ECE 375 Lab 7

Timer/Counters

Lab Time: Wednesday 5-7pm

Rhea Mae Edwards

Jack Neff

INTRODUCTION

The purpose of this lab is to learn proper use of the Timer/Counter registers. For this lab, Timer/Counter0 and Timer/Counter2 is used to control the two motors on the TekBot. The Timer/Counter registers are used to control the speed of the TekBot. By completing this lab, students will learn how to control the Timer/Counter registers to properly control the frequency of PWM output.

PROGRAM OVERVIEW

This program allows the TekBot to change speed levels. There are 16 speed levels that it can access. It works by setting the interrupts to the first four buttons of the Atmega128 board. This way each button corresponds to a function: set speed to maximum, set speed to minimum, speed down by one level, and slow up by one level. It also shows the binary depiction of the speed that it is currently at.

INITIALIZATION ROUTINE

The initialization routine of this program initializes the stack pointer, then sets up PORTB for output and PORTD for input. Then it initializes external interrupts to trigger on a rising edge by setting up EICRA. Then sets up global interrupts by calling SEI. It then sets up TCCR0, and TCCR2 to be fast PWM, inverted. Then loads 0 into OCR0 and OCR2 so that it starts out running. Then configure the step to 17 and set the TekBot to move forwards.

MAIN ROUTINE

The main routine of this program is simply left empty.

MAXSPD ROUTINE

The maxspd routine simply loads the number 0 into OCR0 and OCR2. Then it loads the number from PORTB, ANDs it with 0b11110000 so that it masks out whatever is already there. Then it ORs with the number 15 to display both the maximum speed and the speed mode it is in and outputs that to PORTB.

MINSPD ROUTINE

The minspd routine is identical to the maxspd routine except that instead of loading 0 into OCR0 and OCR2, it loads 255. Then it ORs the contents of PORTB with 0 instead of 15 to wipe out the lower bits and display no LEDs.

SPDINC ROUTINE

The spdinc routine is used to increase the speed of the TekBot by 1 level. It works by first checking to see if the speed is already maxed by loading in 0 and comparing it to the speed register. If it is already maxed then it will RETI so that it does not loop around. If the speed is not maxed then it will subtract 17 to the current speed and output it to OCR0 and OCR2. It then increments the speedlevel register to let the user know which speed it is currently in. It does this by incrementing and then OR-ing it with the number in PINB, this way it does not turn the motors off when it displays the current speed. It also ANDs with the binary number 0b11110000 so that it masks out whatever number is already in PORTB.

Spddec Routine

The spddec routine is identical to the spdinc routine except that instead of checking for the number 0 in the speed register, it will check for 255. Also instead of adding 17 to speed it will subtract 17 from speed. In testing, this was the only function that did not work.

Overflow Routine

Overflow simply executes reti, to return from interrupts when a user tries to exceed the min and max speed limits.

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. Imagine instead that you used just one of the 8-bit Timer/Counters in Normal mode, and every time it overflowed, it generated an interrupt. In the overflow ISR, you manually toggled both Motor Enable pins of the TekBot, and wrote a new value into the 8-bit Timer/Counter's register. (If you used the correct values, you would essentially be manually implementing a fixed-frequency, variable duty-cycle output signal.) Provide your best assessment (in 1-2 paragraphs) of the advantages and disadvantages of this new approach, in comparison to the original PWM approach used in this lab.*

The disadvantages of doing it this way is that it is a lot more difficult, since CTC mode normally only varies the frequency of the waveform and not the duty cycle, the extra code it takes to work around it may be hard to implement. The advantage of doing it this way is that you are only using one of the 8-bit Timer/Counter and leaves the other one open to do anything else you want with it.

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).*

In normal mode, an interrupt happens when the Timer/Counter register overflows. In CTC mode, a waveform is generated every time TCNT0 and OCR0 matches. So to use this:

Pseudocode: Set OCR0 to a number and wait for TCNT0 to match it. To increase speed, decrease OCR0 and to decrease speed increase OCR0.

Conclusion

The purpose of this lab was to learn how to use the fast PWM mode of the timer/counter registers to control the intensity of an output. In this case it is the speed of two motors. This is done by setting up four interrupts that each lead to a subroutine. The subroutines are: maxspd, minspd, spddec, and spdinc. Each subroutine manipulates the value in a register that is outputted to OCR0 and OCR2 which controls the intensity of the output at PORTB and makes the motors speed.

SOURCE CODE

```
;***********************************************************
;*
;* main.asm
;*
;* Speed-adjustable fast PWM lab
;*
;***********************************************************
;*
;* Author: Jack Neff, Rhea Mae Edwards
;* Date: February 28th, 2017
;*
;***********************************************************

.include "m128def.inc"        ; Include definition file

;***********************************************************
;* Internal Register Definitions and Constants
;***********************************************************

.def    mpr = r16       ; Multipurpose register
.def    speedr = r17    ; Holds the value to be put into OCR registers
.def    step = r18      ; Hold the value by which we increment speed (17)
.def    speedlevel = r19       ; Hold the binary value of the speed (0-15)

.equ    Wskr0 = 0
.equ    Wskr1 = 1
.equ    Wskr2 = 2
.equ    Wskr3 = 3
.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    GoFwd = (1<<EngDirR|1<<EngDirL)

;***********************************************************
;* Start of Code Segment
;***********************************************************

.cseg   ; Beginning of code segment

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

.org $0000
        rjmp INIT       ; Reset interrupt

.org $0002
        rcall spdinc
        reti

.org $0004
        rcall spddec
        reti

.org $0006
        rcall maxspd
        reti

.org $0008
        rcall minspd
        reti

; Place instructions in interrupt vectors here, if needed

.org $0046

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

```
;* Program Initialization
;***********************************************************

INIT:
	; Configure I/O ports
	LDI mpr, low(RAMEND)
	OUT SPL, mpr
	LDI mpr, high(RAMEND)
	OUT SPH, mpr

	LDI mpr, $FF
	OUT DDRB, mpr
	LDI mpr, $00
	OUT PORTB, mpr

	LDI mpr, $00
	OUT DDRD, mpr
	LDI mpr, $FF
	OUT PORTD, mpr

	LDI mpr, 0b11111111
	STS EICRA, mpr

	LDI mpr, 0b00001111
	OUT EIMSK, mpr

	LDI mpr, 0b01111001
	OUT TCCR0, mpr
	OUT TCCR2, mpr

	LDI mpr, 255
	OUT OCR0, mpr
	OUT OCR2, mpr

	LDI step, 17

	LDI speedr, 255
	LDI speedlevel, 0

	LDI mpr, GoFwd
	OUT PORTB, mpr

	SEI

	; Initialize the Stack Pointer
	; Configure I/O ports
	; Configure External Interrupts, if needed
	; Configure 8-bit Timer/Counters
	; No prescaling
	; Set TekBot to Move Forward (1<<EngDirR|1<<EngDirL)
	; Set initial speed, display on Port B pins 3:0
	; Enable global interrupts (if any are used)

;***********************************************************
;* Main Program
;***********************************************************

MAIN:
	; Poll Port D pushbuttons (if needed)

	rjmp	MAIN ; Continue through main

	; If pressed, adjust speed
	; Also, adjust speed indication

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

;-----------------------------------------------------------
; Func: maxspd
; Desc:
;-----------------------------------------------------------

maxspd:
        LDI speedr, 0
        OUT OCR0, speedr
        OUT OCR2, speedr
        LDI speedlevel, 15
        in mpr, PINB
        andi mpr, 0b11110000
        ori mpr, 15
        out PORTB, mpr
        reti

;-----------------------------------------------------------
; Func: minspd
; Desc:
;-----------------------------------------------------------

minspd:
        LDI speedr, 255
        OUT OCR0, speedr
        OUT OCR2, speedr
        LDI speedlevel, 0
        in mpr, PINB
        andi mpr, 0b11110000
        ori mpr, 0
        out PORTB, mpr
        reti

;-----------------------------------------------------------
; Func: spddec
; Desc:
;-----------------------------------------------------------

spddec:
        ldi mpr, 255
        cp speedr, mpr
        breq overflow
        add speedr, step
        out OCR0, speedr
        out OCR2, speedr
        ; Dec speedlevel
        in mpr, PINB
        andi mpr, 0b11110000
        or mpr, speedlevel
        dec mpr
        dec speedlevel
        out PORTB, mpr

;-----------------------------------------------------------
; Func: spdinc
; Desc:
;-----------------------------------------------------------

spdinc:
        ldi mpr, 0
        cp speedr, mpr
        breq overflow
        sub speedr, step
        out OCR0, speedr
        out OCR2, speedr
        ;inc speedlevel
        in mpr, PINB
        andi mpr, 0b11110000
        or mpr, speedlevel
        inc mpr
        inc speedlevel
        out PORTB, mpr

;-----------------------------------------------------------
```

```
; Func: overflow
; Desc:
;-----------------------------------------------------------

overflow:

        reti

        ; If needed, save variables by pushing to the stack

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

        ; Enter any stored data you might need here

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

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