ECE 375 LAB 7

Timer/Counters

Lab Time: Wednesday 5-7

Zhenggan Zheng

Abhishek Raol

Introduction

The purpose of Lab 7 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: speed up by one level, slow down by one level, set speed to maximum, and set speed to minimum. 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 speedincrement to 17 and set the TekBot to move forwards.

Main Routine

The main routine of this program is simply left empty.

Speedup Routine

The speedup 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 255 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 add 17 to the current speed and output it to OCR0 and OCR2. It then increments the speedmode 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.

Slowdown Routine

The slowdown routine is identical to the speedup routine except that instead of checking for the number 255 in the speed register, it will check for 0. Also instead of adding 17 to speed it will subtract 17 from speed.

Maxspeed Routine

The maxspeed routine simply loads the number 255 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.

Minspeed Routine

The minspeed routine is identical to the maxspeed routine except that instead of loading 255 into OCR0 and OCR2, it loads 0. Then it ORs the contents of PORTB with 0 instead of 15.

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: minspeed, maxspeed, speedup, and slowdown. 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

```
;***********************************************************
;*
;*     AssemblerApplication7.asm
;*
;*     Controls the speed of TekBot motors
;*
;*     This is the skeleton file for Lab 7 of ECE 375
;*
;***********************************************************
;*
;*      Author: Zhenggan Zheng
;*                      Abhishek Raol
;*         Date: 2/24/2016
;*
;***********************************************************

.include "m128def.inc"               ; Include definition file

;***********************************************************
;*      Internal Register Definitions and Constants
;***********************************************************
.def    mpr = r16                           ; Multipurpose register
.def    speed = r17
.def    speedinc = r18
.def    speedmode = r19
.equ    button0 = 0
.equ    button1 = 1
.equ    button2 = 2
.equ    button3 = 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    MovFwd = (1<<EngDirR|1<<EngDirL) ;Move forward command
;***********************************************************
;*      Start of Code Segment
;***********************************************************
.cseg                                               ; beginning of code segment

;***********************************************************
;*      Interrupt Vectors
;***********************************************************
.org    $0000
                rjmp    INIT                ; reset interrupt
.org    $0002
                rcall   speedup
                reti
.org    $0004
                rcall   slowdown
                reti
.org    $0006
                rcall   maxspeed
                reti
.org    $0008
                rcall   minspeed
                reti
                ; place instructions in interrupt vectors here, if needed

.org    $0046                               ; end of interrupt vectors

;***********************************************************
;*      Program Initialization
;***********************************************************
INIT:
                ; Initialize the Stack Pointer
                ldi r16, low(RAMEND)
                out SPL, r16
                ldi r16, high(RAMEND)
                out SPH, r16
                ; Configure I/O ports
                ldi mpr, $ff
                out DDRB, mpr
                ldi mpr, $00
                out PORTB, mpr

                ldi mpr, $00
                out DDRD, mpr
                ldi mpr, $ff
                out PORTD, mpr

                ; Configure External Interrupts, if needed
                ldi mpr, 0b11111111
```

```
		sts EICRA, mpr


		;configures masking
		ldi mpr, 0b00001111
		out EIMSK, mpr


		; Configure 8-bit Timer/Counters for fast PWM, inverted mode. It is inverted since
TekBot is active low
		ldi mpr, 0b01111001
		out TCCR0, mpr
		ldi mpr, 0b01111001
		out TCCR2, mpr

		ldi mpr, 0
		out OCR0, mpr
		out OCR2, mpr

		ldi speedinc, 17	;This is the speed that it will increment or decrement by
each level

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

		ldi mpr, MovFwd			; Command to make TekBot move forwards indefinitely
		out PORTB, mpr

		; Enable global interrupts (if any are used)
		sei
;***********************************************************
;*	Main Program
;***********************************************************
MAIN:

		rjmp	MAIN			; return to top of MAIN

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

;-----------------------------------------------------------
; Func:	Template function header
; Desc:	Cut and paste this and fill in the info at the
;		beginning of your functions
;-----------------------------------------------------------
maxspeed:
		ldi speed, 255 ;Loads 255 into speed
		out OCR0, speed		;Outputs speed into OCR0
		out OCR2, speed ;Outputs speed into OCR2
		in mpr, PINB	;Reads input from PORTB
		andi mpr, 0b11110000	;Masks the first 4 bits of PORTB
		ori mpr, 15				;OR it with 15 to display max speed
		out PORTB, mpr			;Output that OR-ed number to PORTB
		reti					;Return from interrupt
minspeed:
		ldi speed, 0	;Loads 0 into speed
		out OCR0, speed		;Outputs speed into OCR0
		out OCR2, speed ;Outputs speed into OCR2
		in mpr, PINB	;Reads input from PORTB
		andi mpr, 0b11110000	;Mask out the first 4 bits of PORTB
		ori mpr, 0				; OR it with 0 to display min speed
		out PORTB, mpr			;Output that OR-ed number to PORTB
		reti					;Return from interrupt
speedup:
		ldi mpr, 255	;Loads 255 into mpr
		cp speed, mpr	;compare that to speed
		breq return1	;If speed is maxed, return from interrupt
		add speed, speedinc	;Add 17 to speed
		out OCR0, speed			;Output speed to OCR0
```

```
               out OCR2, speed               ;Output speed to OCR2
               inc speedmode         ;Increment speed mode
               in mpr, PINB          ;Read input from PORTB
               andi mpr, 0b11110000  ;AND to mask out first 4 bits of PORTB
               or mpr, speedmode             ;OR it with speed mode
               out PORTB, mpr                ; Output it to PORTB
return1:
               reti    ;Return from interrupt
slowdown:
               ldi mpr, 0            ;Loads 0 into mpr
               cp speed, mpr  ;Compare that to speed
               breq return2   ;If already min speed, return from interrupt
               sub speed, speedinc   ;Subtract 17 from speed
               out OCR0, speed               ;Output speed to OCR0
               out OCR2, speed               ;Output speed to OCR2
               dec speedmode         ;Decrement speedmode
               in mpr, PINB          ;Read input from PORTB
               andi mpr, 0b11110000  ;AND to mask out 4 bits
               or mpr, speedmode             ;OR it with speed mode
               out PORTB, mpr                ;Output it to PORTB
return2:
               reti                          ;Return from interrupt


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

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