ECE 375 LAB 7

Timer/Counters

Lab Time: Wednesday 10-12

Tu Lam

INTRODUCTION

The purpose of the lab is to introduce the timer/counter and let us explore how it is a register whose contents are regularly incremented (or decremented) at a specified, configurable frequency. Exploring both the width and the frequency, the lab will help us determine how the resolution and the range can be seen through the eye of the BumpBot. Beyond simply measuring an interval of time, Timer/Counters can also be configured to take some action based on an observed interval, such as toggling an I/O pin after a certain amount of time has passed, generating a waveform. Overall, using the BumpBot, we can display how using the timer/counter to display the speed of the BumpBot.

PROGRAM OVERVIEW

The program overall will run the BumpBot to move forward continuously in a loop in the MAIN function. When a trigger is hit, the program will either increment the speed, decrement it, speed it up to the max, or halt it to zero. If the user hit the increment speed, it will increment by 1 and there are 0-15 different speeds that can be increment. If the user hit the decrement speed, it will decrement by 1 and there are 0-15 different speeds that can be decrement. The other two options will raise the speed to the MAX or MIN base on the user input.

INITIALIZATION ROUTINE

The initialization routine provides a one-time initialization of the stack pointer to set up where to point in the program and it initialize the interrupt mask register and set up PORTB and PORTD as input and output. Beside that, the INIT will initialize the timer/counter and set the set the BumpBot to move forward.

MAIN ROUTINE

The Main routine executes a simple statement of moving the BumpBot forward continuously forever and ever until a trigger is hit.

SPEED_UP ROUTINE

The SPEED_UP routine will increase the BumpBot speed by the increment of 1. Before that, there is a wait time for the button bounce just to make sure the button has been pushed for it to continue. Then the program run with comparing if the MAX value has reach in the counter, if not increment by 1 and increment the display by 1.

SPEED_DOWN ROUTINE

The SPEED_DOWN routine will decrease the BumpBot speed by the decrement of 1. Before that, there is a wait time for the button bounce just to make sure the button has been pushed for it to continue. Then the program run with comparing if the MIN value has reach in the counter, if not decrement by 1 and decrement the display by 1.

SPEED_MAX ROUTINE

The SPEED_MAX routine will set the counter to the highest speed that the BumpBot can run. This will compare if the program has reach is MAX counter and display the maximum speed out as the output.

SPEED_MIN ROUTINE

The SPEED_MIN routine is the exact same thing as the SPEED_MAX routine is performing. In one way, the only thing is different about it is that it looks at the MIN and display the halt (stop) of the BumpBot out onto the display.

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.

Answer: While having normal mode as the new approach for the timer/counter, there is the advantage and disadvantage. With having the PWM mode, it is easy to determine the bottom and the max, but with normal mode, it is harder as the bottom is the value loaded in and the max is just the maximum value of 255. This is a disadvantage as it is harder to know the bottom of the counter if it is requiring the value loaded in as the bottom of the counter. The advantage of doing with normal mode is that there is less amount of speed to increase/decrease as the interval for delay is shorter as it does not start at the bottom but at a specific location.

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).

Answer: Pseudo-Code

Set up the TRCCR0 to be the inverted, no prescale, and at CTC.

Set up the PORTB, PORTD, and interrupt for the function

For the decrease speed, if OCR0 has reach 0 (bottom) then do nothing, but if not, then decrement the speed
For the increase speed, if OCR0 has been reach (top) then do nothing, but if not, then increment the speed

For the max speed, if the speed is at OCR0, then display the max speed

For the min speed, if the speed is at 0, then display the min speed

CONCLUSION

For Lab #7, the challenge overall let the user understand how timer/counter using the range to find the speed of the BumpBot. Using the I/O as a way to run the timer/counter and use it to know where the start and max value is to turn on the speed of the BumpBot is a great way to implemented into the program. There was not any hard part when it comes to this program except understanding how to initialize the timer/counter. The overall lab was easy to understand but learning the concept of how to integrate the timer/counter can help in many future usage of the embedded system.

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.

```
;***********************************************************
;*
;*      Tu_Lam_Lab7_sourcecode.asm
;*
;*      Description: This file include moving the BumpBot but also
;*                               display the the speed changes in the BumpBot.
;*                               Also, you will learn how to use the Timer.
;*
;***********************************************************
;*
;*       Author: Tu Lam
;*         Date: November 14th, 2020
;*
;***********************************************************

.include "m128def.inc"                  ; Include definition file

;***********************************************************
;*      Internal Register Definitions and Constants
;***********************************************************
.def    mpr = r16                               ; Multipurpose register
.def    waitcnt = r17                           ; A wait counter from R17-R19
.def    olcnt = r18
.def    ilcnt = r19
.def    temp = r20                              ; Temporary register to use
.def    led = r21                               ; LED display

.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    Speed = 17                              ; Speed of counter increment

;***********************************************************
;*      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
                rcall   SPEED_UP                ; Call SPEED_UP interrupt
                reti                            ; Return from interrupt

.org    $0004
                rcall   SPEED_DOWN              ; Call SPEED_DOWN interrupt
                reti                            ; Return from interrupt

.org    $0006
                rcall   SPEED_MAX               ; Call SPEED_MAX interrupt
                reti                            ; Return from interrupt

.org    $0008
                rcall   SPEED_MIN               ; Call SPEED_MIN interrupt
                reti                            ; Return from interrupt
```

```
.org    $0046                           ; end of interrupt vectors

;***********************************************************
;*      Program Initialization
;***********************************************************
INIT:
                ldi             mpr, low(RAMEND)        ; Initialize Stack Pointer
                out             SPL, mpr
                ldi             mpr, high(RAMEND)
                out             SPH, mpr

                ldi             mpr, 0b11111111         ; Configure I/O ports
                out             DDRB, mpr               ; Initialize output for PORT B
                ldi             mpr, 0b11110000         ; Initialize input for PORT D
                out             DDRD, mpr

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

                ; Configure 8-bit Timer/Counters
                ldi             mpr, 0b01111001         ; Setting up the Fast PWM mode
                out             TCCR0, mpr              ; No prescaling (And inverting)
                out             TCCR2, mpr


                ; Set TekBot to Move Forward (1<<EngDirR|1<<EngDirL)
                ldi             led, (1<<EngDirR)|(1<<EngDirL)
                out             PORTB, led

                ; Set initial speed, display on Port B pins 3:0
                ldi             mpr, 0b00000000         ; Initialize to be 0 at first
                out             OCR0, mpr
                out             OCR2, mpr

                ; 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: This subroutine will increment the speed by 1 of the
;               timer/counter.
;-----------------------------------------------------------
SPEED_UP:

                ldi             waitcnt, 30             ; Account for wait if button bounce
                rcall   WAITING                         ; Wait for 30ms

                in              temp, OCR0              ; Get the speed level
                cpi             temp, 255               ; Compare if the speed is at MAX
                breq    CHECK_DONE                      ; If so, move to the label CHECK_DONE
                ldi             mpr, Speed              ; Load value 17 into mpr
                add             temp, mpr               ; If not, increment the speed by 17
                out             OCR0, temp              ; Write out the value to OCR0 & OCR2
                out             OCR2, temp
                inc             led                     ; Increment the display to display the next speed
                out             PORTB, led              ; Send it to PORTB as output
```

```
CHECK_DONE:
                ldi             mpr, 0b00001111         ; Clear any queue interrupt
                out             EIFR, mpr               ; Store clear values into EIFR
                ret                                     ; Return the subroutine


;-----------------------------------------------------------
; Func: SPEED_DOWN
; Desc: This subroutine will decrement the speed by 1 of the
;               timer/counter.
;-----------------------------------------------------------
SPEED_DOWN:

                ldi             waitcnt, 30             ; Account for wait if button bounce
                rcall   WAITING                         ; Wait for 30ms

                in              temp, OCR0              ; Get the speed level
                cpi             temp, 0                 ; Compare if the speed is at MIN
                breq    CHECK_DONE1                     ; If so, move to the label CHECK_DONE1
                ldi             mpr, Speed              ; Load value 17 into mpr
                sub             temp, mpr               ; If not, decrement the speed by 17
                out             OCR0, temp              ; Write out the value to OCR0 & OCR2
                out             OCR2, temp
                dec             led                     ; Decrement the display to display the next speed
                out             PORTB, led              ; Send it to PORTB as output

CHECK_DONE1:
                ldi             mpr, 0b00001111         ; Clear any queue interrupt
                out             EIFR, mpr               ; Store clear values into EIFR
                ret                                     ; Return the subroutine


;-----------------------------------------------------------
; Func: SPEED_MAX
; Desc: This subroutine will increment toward the MAX speed in
;               timer/counter.
;-----------------------------------------------------------
SPEED_MAX:

                ldi             waitcnt, 30             ; Account for wait if button bounce
                rcall   WAITING                         ; Wait for 30ms

                in              temp, OCR0              ; Get the speed level
                cpi             temp, 255               ; Compare if the speed is at MAX
                ldi             mpr, 255                ; Load into mpr with 255
                add             temp, mpr               ; Add them together
                out             OCR0, temp              ; Write out the value to OCR0 & OCR2
                out             OCR2, temp
                ldi             led, 0b01101111         ; Display out the full speed
                out             PORTB, led              ; Send it to PORTB as output

                ldi             mpr, 0b00001111         ; Clear any queue interrupt
                out             EIFR, mpr               ; Store clear values into EIFR
                ret                                     ; Return the subroutine


;-----------------------------------------------------------
; Func: SPEED_MIN
; Desc: This subroutine will decrement toward the MIN speed in
;               timer/counter.
;-----------------------------------------------------------
SPEED_MIN:

                ldi             waitcnt, 30             ; Account for wait if button bounce
                rcall   WAITING                         ; Wait for 30ms

                in              temp, OCR0              ; Get the speed level
                cpi             temp, 0                 ; Compare if the speed is at MIN
```

```
                ldi             mpr, 255                ; Load into mpr with 255
                sub             temp, mpr               ; Subtract them together
                out             OCR0, temp              ; Write out the value to OCR0 & OCR2
                out             OCR2, temp
                ldi             led, 0b11110000         ; Display out the full speed
                out             PORTB, led              ; Send it to PORTB as output

                ldi             mpr, 0b00001111         ; Clear any queue interrupt
                out             EIFR, mpr               ; Store clear values into EIFR
                ret                                     ; Return the subroutine

;-----------------------------------------------------------
; Func: WAITING
; Desc: This function help the BumpBot to wait going through
;               a loop of 16 + 159975*waitcnt cycles.
;-----------------------------------------------------------
WAITING:
                push    waitcnt                         ; Push registers on the stack

OLOOP:
                ldi             olcnt, 224              ; Load middle-loop with 224

MLOOP:
                ldi             ilcnt, 237              ; Load the inner-loop with 237

ILOOP:
                dec             ilcnt                   ; Decrement the inner-loop
                brne    ILOOP                           ; Continue to loop inside inner-loop if not reach
                dec             olcnt                   ; Decrement the middle-loop
                brne    MLOOP                           ; Continue to loop inside middle-loop if not
reach
                dec             waitcnt                 ; Decrement outer-loop
                brne    OLOOP                           ; Continue to loop inside outer-loop if not reach

                pop             waitcnt                 ; Pop & restore registers off of stac
                ret                                     ; Return the subroutine


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

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