ECE 375 LAB 7

Timer/Counters

Lab Time: Tuesday 6-7:50

Adit Kotharii

Mac

INTRODUCTION

The Goal of this lab was to familiarize ourselves with Timers/Counters on the ATmega128. We used the Timers/Counters to generate PWM signals that were used with external interrupts to control the speed of the Tekbot.

PROGRAM OVERVIEW

The program allows us to modify the speed of the TekBot, for the input provided by the Port D. Two 8-bit timers are used in Fast PWM mode to drive the two motors. Changing the duty cycle allows us to modify the speed of the motors, thereby increasing or decreasing its speed.

The program contains 5 other subroutines apart from the main and initialization, that are used to control speed of the TekBot: addPress, SubPress, Maximum, Minimum and wait.

INITIALIZATION ROUTINE

First the Stack Pointer is initialized. Next, we configure the I/O ports, i.e Port B for output and Port D for input. Then we initialize the External Interrupts and configure the 8 bit timer/counters. The External Interrupts 3:0 are set to detect falling edge and the Timer/Counters are set to Fast PWM mode.

MAIN ROUTINE

Infinite loop that gets the Tekbot to move forward, until one of the buttons is pressed getting it into ISR.

ADDPRESS

This subroutine first checks the current level of speed if it is at max or not. If the current level of speed is not max 17 is added to the current speed. We also call the wait function to avoid multiple increase in speed level on button press The current level of speed can be known by looking at the first four LED’s.

SUBPRESS

This subroutine is pretty similar to the AddPress subroutine. It first checks the current level of speed if it is at min or not. If the current level of speed is not min 17 is subtracted from the current speed. We also call the wait function to avoid multiple decrease in speed level on button press. The current level of speed can be known by looking at the first four LED’s.

MAXIMUM

This subroutine sets speed to max and turns on the first four LED’S.

MINIMUM

This subroutine sets speed to minimum and turns off the first four LED’S.

WAIT

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 equation for the number of clock cycles in the wait loop:
((3 * ilcnt + 3) * olcnt + 3) * waitcnt + 13 + call

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 new approach requires us to use normal mode to generate PWM signal manually. Operating Timer/Counter in the Normal mode allows us to manipulate the frequency allowing us to better manipulate the period. However this new approach, seems to be more of a tedious and inefficient approach as it would rely on polling to check for input from the port D.

On the other hand using fast PWM mode, seems to be more advantageous for implementation of its lab as it automatically generates PWM signal and gives us full control over the duty cycle to be manipulated.

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 TCCRn to CTC mode
- Enable Interrupts
- Set desired value for OCRn
- Let TCNTn count up to OCRn
- When TCNTn value equals OCRn value, generate Interrupt and go into ISR
- ISR: change value in OCRn (changing value of OCRn changes the frequency of OCn, changing the output of LED)

CONCLUSION

In this lab, I learned about timers/counters and how to use them to generate PWM signal that allows us to control the speed of the TekBot. In addition I also learnt about advantages of using the timers/counter in fast PWM mode over other modes.

SOURCE CODE

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

;***********************************************************
;*
;*      Enter Name of file here
;*
;*      Enter the description of the program here
;*
;*      This is the skeleton file for Lab 7 of ECE 375
;*
;***********************************************************
;*
;*       Author: Enter your name
;*         Date: Enter Date
;*
;***********************************************************

.include "m128def.inc"               ; Include definition file

;***********************************************************
;*      Internal Register Definitions and Constants
;***********************************************************
.def    mpr = r16                             ; Multipurpose register
.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
.org    $0002
               rcall   addPress
               reti
.org    $0004
               rcall   subPress
               reti
.org    $0006
               rcall   minimum
               reti
.org    $0008
               rcall   maximum
               reti


.org    $0046                                               ; end of interrupt vectors

;***********************************************************
;*      Program Initialization
;***********************************************************
INIT:

               LDI R16, LOW(RAMEND)                  ; Initialize Stack Pointer
               OUT SPL, R16
               LDI R16, HIGH(RAMEND)
               OUT SPH, R16
               ; Configure I/O ports

               ;PORTB USAGE-LED's all of them
               ldi r19, $9F                                 ;R19 will hold the current state of
the LED's
               ldi mpr, $9F                                 ;All LED's on except for the 6th and
7th one to represnt forward
               out DDRB, MPR                                ;The two LED's not on are set for
input as they have 0s written to them
               ldi mpr, $ff
               out PORTB, mpr                               ;This writes 1's to all LED's.
However only 6 of them have been set up for output by writing 1's

               ; Initialize PortD for input
               ldi mpr, $00                                 ;write 0's to all buttons
               out DDRD, mpr                                ;all buttons are set for input
               ldi mpr, $FF                                 ;writes 1's to the first four
buttons. Buttons are active low so they go to 0 on being pushed down
               out PORTD, mpr                               ;Right 4 most buttons are used

               ; Initialize external interrupts
               ldi mpr, $AA                                 ;Falling edge
               sts EICRA, mpr                               ;Using internal interrupts 0, 1, 2 ,
3
               ldi mpr, $0F                                 ;need to mask out the other 4
external interrupts
               out EIMSK, mpr                               ;by writing 0's to the ones we are
not using

               ; Configure 8-bit Timer/Counters
               LDI mpr, 0b01101001                ;First 3 bits represent no prescalar
               out TCCR2, mpr                               ;bits 6 and 3 in conjunction set mode
which is PWM
               out TCCR0, mpr                               ;bits 4 and 5 set behavior of OC0 & 2
on match which is clear OC0 & 2

               ldi mpr, $0F                                 ;writes 255 to OCR0 & 2 inorder for
them to start at full speed
               out OCR0, mpr
               out OCR2, mpr

               ; Enable global interrupts
               sei
;************************************************************
;*     Main Program
;************************************************************
MAIN:
                                                            ; poll Port D pushbuttons (if needed)


               rjmp    MAIN                  ; return to top of MAIN

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

;-----------------------------------------------------------
; Func: addPress(ISR $0002)
; Desc: Checks to see if the current level of speed is at the max
;              if not it adds 17 to the current speed and handles displaying
;              the knew level of speed on the first four LED's
;-----------------------------------------------------------
addPress:
        in mpr, SREG
        push mpr

        in mpr, OCR0                         ;Check to see if speed is max
        cpi mpr, 255                         ;by comparing OCR0 with max speed
        breq noAdd                                   ;branch if they are equal
        inc r19                                      ;if not equal handle dispaly LED's by moving
speed level up one
```

```
        out PORTB, r19                          ;r19 should never get to 16 which means that its
initial value set in INIT should maintain the high nible while editing the low nible
        SUBI mpr, -17                           ;To add immidiate we subtract a negative number.
Each level increase is +17
        out OCR2, mpr                           ;write new values to OCR0 & 2
        out OCR0, mpr
noAdd:
        rcall wait                                      ;to stop multiple increase in level on one
button press add a wait

        pop mpr

        ldi r21, $0F                            ;unlock the external interrupts so they don't que
        OUT EIFR, r21

        ret                                                     ;this is the wrong place for this but
it works. Should be up at the interrupt vectors
;-------------------------------------------------------------
; Func: subPress(ISR $0004)
; Desc: Checks to see if the current level of speed is at the min
;               if not it subtracts 17 from the current speed and handles displaying
;               the knew level of speed on the first four LED's
;-------------------------------------------------------------
subPress:
        in mpr, SREG
        push mpr

        in mpr, OCR0                            ;same concept at add but we subtract 17. Also checks
to see that the level isn't 0 already before subtracting
        cpi mpr, 0
        breq noSub
        dec r19
        out PORTB, r19
        SUBI mpr, 17
        out OCR2, mpr
        out OCR0, mpr
noSub:
        rcall wait

        pop mpr

        ldi r21, $0F
        OUT EIFR, r21

        ret
;-------------------------------------------------------------
; Func: maximum(ISR $0008)
; Desc: Sets speed to max and turns on all 4 speed display LED's
;-------------------------------------------------------------
maximum:                                                ;BROKEN. Falls thru to minimum for some
reason
        in mpr, SREG
        push mpr

        ldi mpr, 255                            ;Write the max speed to OCR0 & 2
        out OCR2, mpr
        out OCR0, mpr
        ldi r19, $9F                            ;Write corresponding LED values to PortB. Same as
the inital value of PortB set in INIT
        out PORTB, r19                          ;level 15
        rcall wait

        pop mpr

        ldi r21, $0F
        OUT EIFR, r21

        ret
;-------------------------------------------------------------
; Func: maximum(ISR $0006)
; Desc: Sets speed to min and turns off all 4 speed display LED's
```

```
;-----------------------------------------------------------
minimum:
        in mpr, SREG
        push mpr

        ldi mpr, 0                                      ;write minimum value to OCR0 & 2
        out OCR2, mpr
        out OCR0, mpr
        ldi r19, $90                            ;write 0's to the display LED's to signify level 0
        out PORTB, r19
        rcall wait

        pop mpr

        ldi r21, $0F
        OUT EIFR, r21

        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:

                ldi             mpr, 30
Loop:   ldi             r22, 224                ; load olcnt register
OLoop:  ldi             r23, 237                ; load ilcnt register
ILoop:  dec             r23                     ; decrement ilcnt
                brne    ILoop                   ; Continue Inner Loop
                dec             r22             ; decrement olcnt
                brne    OLoop                   ; Continue Outer Loop
                dec             mpr             ; Decrement wait
                brne    Loop                    ; Continue Wait loop


                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
```