ECE 375 LAB 7

Timer/Counters

Lab Time: Tuesday 4-6

Owen Markley

INTRODUCTION

The purpose of this lab is to learn how to use the 8-bit timer and counters on the ATmega 128 board with the PWM signals. There are two halves of the bytes that we have to use which are the TCCR0 and the TCCR2. This will allow us to get used to using the timer to control behavior inside the program rather than creating obtuse functions to have things wait for a while. We can set a time for the timer to reach and have an action occur once it has triggered which is much more efficient. We also practice using this timer to toggle various I/O pins, and adjust the frequency of this timer in our program.

PROGRAM OVERVIEW

The program has four basic functions. These are the Increase and Decrease Speed, and the Max and Min speed. The Min Speed is essentially Halting the robot. The Max speed will allow the motors to run at maximum speed, and the increase speed and decrease speed will move between 16 different variable speeds. The lowest possible and highest possible being the same as the min and max speed respectively. Only one increment / decrement of speed can be made at a time. There should be no ability to overflow / underflow using the increment / decrement functions.

The results of these functions or the state of the robot will be represented by LED lights. The zero'th pin is the left motor speed, and the third pin is the right motor speed. These should always be the same throughout execution of the program. The first and second pins represent the Direction in which the bot is moving in. Since this bot always moves forwards, it would stand that they are always on. The next four LED's represent the increment of speed that the robot is on. They will count upwards in binary from 0 to 15 with the most significant bit being towards the left side of the array of LEDs. When incrementing and decrementing speed the "brightness" of the motor representing LEDs should be congruent with the binary value.

INITIALIZATION ROUTINE

The initialization routine provides a one-time initialization of key registers that allow the Robot's program to execute correctly. First the Stack Pointer is initialized, allowing the proper use of function and subroutine calls. Port B was initialized to all outputs and will be used to direct the motors. Port D was initialized to inputs and will receive the whisker input. Next the external interrupts are configured to activate on the falling edge of the signal. Then the mask will have the values for our used interrupts set to 1. After this the 8-bit timer/counters are configured. Next, the Move Forward command was sent to Port B to get the TekBot moving forward. Lastly the speed counter is set to 0 and the global interrupts are enabled.

MAIN ROUTINE

The main routine in this function is empty save for a jump statement back to main. This is because we want the robot to continue indefinitely, and all functions are accessed using interrupts.

INCSPEED ROUTINE

The IncSpeed will increase the motor speed by one interval. It first saves the program state, and then fills the flag register for the external mask so that no other signals interrupt it. Next the value 15 is loaded into the mpr, so that it can be compared to the current speed. Should it be less, then the speed counter will increase by one. Next the value is loaded into the mpr, so that it can be written out to PORTB aka the LED display. Next, the counter speed is loaded into the OCR0 and OCR2, which are used in the timer. The output from the timer is compared to this so that it knows if it needs to reset itself. Next the EIFR is restored so that interrupts can go through once more. And the program state is restored and a return is made to main.

DecSpeed Routine

This is the same as the above function, however it is decrementing the value of the speed counter, and the value compared is 0 rather than 15.

Rest Routine

This routine is unused, and does not need to be considered for this program.

TopSpeed Routine

First, the program state is stored, next the EIFR is set to prevent any interrupt requests. Next, the speed is set to 15 and output to PORTB. After this we repeat the same process as above to send this command out to the port, and set the first four motor LED's to be on. We then perform the same operation on the OCR0 and OCR2 registers as we did above in Inc/Dec Speed. We then restore the EIFR, and restore the state of the program

MinSpeed Routine

This is very similar to the Top Speed with a few differences. First is that the speed is set to 0 rather than 15. Next, we simply output 11110000 to the LED's since there isn't any other arithmetic to be done, since all LEDs are either fully on or fully off in this state. All following code is the same as the TopSpeed routine

Additional Questions

Lost my laptop earlier today, so these have not been completed.

Difficulties

Something went wrong with my hardware where the fifth button from the left is somewhat unresponsive and takes multiple presses to activate sometimes. I was unable to figure out how to fix this or what is causing it.

Conclusion

After completion, this program allows us to have a robot which always moves forwards and has a variable speed control. We are also given an option to jump straight to the fastest or slowest speed. The timer is also integrated into this program to allows us to control the brightness of the LED's which indicate the power going to each motor. Completing this lab has provided us with a basic understanding of the Timer on the ATmega128 board.

SOURCE CODE

```
;************************************************************
;*
;*     Owen_Markley_Lab7_sourcecode
;*
;*     Enter the description of the program here
;*
;*     This is the skeleton file for Lab 7 of ECE 375
;*
;************************************************************
;*
;*      Author: Owen Markley
;*        Date: 2/18/20
;*
;************************************************************

.include "m128def.inc"                    ; Include definition file

;************************************************************
;*     Internal Register Definitions and Constants
;************************************************************
.def   mpr = r16                          ; Multipurpose register
.def   cSpeed = r17

.def   waitcnt = r23               ; Rest Loop Counter
.def   ilcnt = r24                        ; Inner Loop Counter
.def   olcnt = r25                        ; Outer Loop Counter


.equ   WTime = 100                        ; Time to wait in wait loop

.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
.equ   Halt = (1<<EngEnR|1<<EngEnL)              ; Halt Command

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

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

.org   $0002
              rcall IncSpeed
              reti

.org   $0004
              rcall DecSpeed
              reti
```

```
.org   $0006
              rcall TopSpeed
              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           mpr, low(RAMEND) ;low end of stack pointer initialized
              out           SPL, mpr             ; Load SPL with low byte of RAMEND
              ldi           mpr, high(RAMEND) ;high end of stack pointer initialized
              out           SPH, mpr             ; Load SPH with high byte of RAMEND

       ; Configure I/O ports
              ; Initialize Port B for output
              ldi           mpr, 0b11111111      ; Set Port B Data Direction Register
              out           DDRB, mpr            ; for output
              ldi           mpr, $00             ; Initialize Port B Data Register
              out           PORTB, mpr           ; so all Port B outputs are low

              ; Initialize Port D for input
              ldi           mpr, $00      ; Set Port D Data Direction Register
              out           DDRD, mpr            ; for input
              ldi           mpr, $FF             ; Initialize Port D Data Register
              out           PORTD, mpr           ; so all Port D inputs are Tri-State

       ; Configure External Interrupts, if needed

              ldi mpr,
(1<<ISC01)|(0<<ISC00)|(1<<ISC11)|(0<<ISC10)|(1<<ISC21)|(0<<ISC20)|(1<<ISC31)|(0<<ISC30) ;
setting these values allows for the falling edge to trigger
              sts EICRA, mpr ;binary value is loaded into external interrupt control
register

              ;Set the External Interrupt Mask
              ldi mpr, (1<<INT0)|(1<<INT1)|(1<<INT2)|(1<<INT3) ; last four digits in
value are set to 1
              out EIMSK, mpr ; setting the external interrupt mask register allows for
signal to go through on these interrupts

              ; Configure 8-bit Timer/Counters
              ldi mpr, 0b01111001
              out TCCR0, mpr

              out TCCR2, mpr

                                                         ; no prescaling
              ; Set TekBot to Move Forward (1<<EngDirR|1<<EngDirL)
              ldi           mpr, MovFwd   ; Load Move Backward command
              out           PORTB, mpr    ; Send command to port


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

              ; Enable global interrupts (if any are used)
              sei

;************************************************************
;*     Main Program
;************************************************************
MAIN:
              ; poll Port D pushbuttons (if needed)
              ;rcall TopSpeed
              ;rcall MinSpeed
                                                        ; if pressed, adjust speed
                                                        ; also, adjust speed indication

              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
;-----------------------------------------------------------
IncSpeed:

              push   mpr                  ; Save mpr register
              in            mpr, SREG     ; Save program state
              push   mpr                  ;

              ldi mpr, 0b00000000 ; value of zero is loaded into mpr
              out EIFR, mpr ; filling the flag register with zeroes will clear any
requests for interrupts

              ldi           mpr, 15
              cpse   cSpeed, mpr
              inc           cSpeed

              ldi           mpr, 0b11110000
              or            mpr, cSpeed
              out           PORTB, mpr    ; Send command to port
              ldi mpr, 17
              mul cSpeed, mpr

              out OCR0, r0
              out OCR2, r0

              ldi mpr, 0b11111111 ; ones are loaded into the mpr
              out EIFR, mpr ; flags are then set to logical high
              pop           mpr           ; Restore program state
              out           SREG, mpr     ;
              pop           mpr           ; Restore mpr

              ret                         ; Return from subroutine

DecSpeed:

              push   mpr                  ; Save mpr register
              in            mpr, SREG     ; Save program state
              push   mpr                  ;

              ldi mpr, 0b00000000 ; value of zero is loaded into mpr
              out EIFR, mpr ; filling the flag register with zeroes will clear any
requests for interrupts

              ldi           mpr, 0
              cpse   cSpeed, mpr
              inc           cSpeed

              ldi           mpr, 0b11110000
              or            mpr, cSpeed
              out           PORTB, mpr    ; Send command to port
              ldi mpr, 17
              mul cSpeed, mpr

              out OCR0, r0
              out OCR2, r0

              ldi mpr, 0b11111111 ; ones are loaded into the mpr
              out EIFR, mpr ; flags are then set to logical high

              pop           mpr           ; Restore program state
              out           SREG, mpr     ;
              pop           mpr           ; Restore mpr

              ret                         ; Return from subroutine

TopSpeed:
cli
              push   mpr                  ; Save mpr register
              in            mpr, SREG     ; Save program state
              push   mpr                  ;

              ldi mpr, 0b00000000 ; value of zero is loaded into mpr
              out EIFR, mpr ; filling the flag register with zeroes will clear any
requests for interrupts

              ldi cSpeed, 15
              out PORTB, cSpeed

              ldi           mpr, 0b11110000
              or            mpr, cSpeed
              out           PORTB, mpr    ; Send command to port
              ldi mpr, 17
              mul cSpeed, mpr

              out OCR0, r0
              out OCR2, r0

              ldi mpr, 0b11111111 ; ones are loaded into the mpr
              out EIFR, mpr ; flags are then set to logical high

              pop           mpr           ; Restore program state
              out           SREG, mpr     ;
              pop           mpr           ; Restore mpr
              ret                         ; Return from subroutine

MinSpeed:
cli
              push   mpr                  ; Save mpr register
              in            mpr, SREG     ; Save program state
              push   mpr                  ;

              ldi mpr, 0b00000000 ; value of zero is loaded into mpr
              out EIFR, mpr ; filling the flag register with zeroes will clear any
requests for interrupts

              ldi cSpeed, 0
              out PORTB, cSpeed

              ldi mpr, 0b11110000
              or mpr, cSpeed
              out PORTB, mpr

              out OCR0, cSpeed
              out OCR2, cSpeed

              ldi mpr, 0b11111111 ; ones are loaded into the mpr
              out EIFR, mpr ; flags are then set to logical high

              pop           mpr           ; Restore program state
              out           SREG, mpr     ;
              pop           mpr           ; Restore mpr
              ret                         ; Return from subroutine

;-----------------------------------------------------------------
; Sub: Rest
; 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
;-----------------------------------------------------------------
Rest:
              push   waitcnt                     ; Save wait register
              push   ilcnt                ; Save ilcnt register
              push   olcnt                ; Save olcnt register

Loop:  ldi           olcnt, 224           ; load olcnt register
OLoop: ldi           ilcnt, 237           ; load ilcnt register
ILoop: dec           ilcnt                ; decrement ilcnt
              brne   ILoop                ; Continue Inner Loop
              dec           olcnt         ; decrement olcnt
              brne   OLoop                ; Continue Outer Loop
              dec           waitcnt              ; Decrement wait
              brne   Loop                 ; Continue Rest loop

              pop           olcnt         ; Restore olcnt register
              pop           ilcnt         ; Restore ilcnt register
              pop           waitcnt              ; Restore wait register
              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
```