

ENGR 270 LAB #5 – Timer & Pulse width Modulation

Objective

Application of Timers to schedule tasks and use of Pulse Width Modulation (PWM) to control average power delivered.

Related Principles

- ❖ Computer Organization and Design
- ❖ Microprocessors
- ❖ Hardware and Software Interface
- ❖ Digital Design
- ❖ Assembly language

Equipment

- ❖ Windows-based PC with MPLAB Simulation Solutions Software
- ❖ USB hard disk or other removable drives
- ❖ Microchip PICKit programmer
- ❖ EDbot V7.0 Platform

Preparation/Background

Prior to start of this lab, you are expected to have completed all prior labs successfully and have reviewed Chapters 2, 4 and 5 of *Computer Organization and Microprocessor* textbook.

Following example code demonstrates the use of Timer1 and Pulse Width Modulation (PWM). This code uses Timer1 to generate an interrupt every two seconds and after each Timer1 interrupt, the power delivered to the right motor toggles between 10% and 30% power.

Note: After reset, switch 4 (RA3) must be changed (toggled) before the code is executed.

```
-----  
; Demonstrate use of Timer1 to change power delivered to right motor using  
; PWM functionality of the PICmicro.  
; LAST UPDATE: 8/14/2018  
; AUTH: Class  
; DEVICE: PICmicro (PIC18F1220)  
-----  
list           p=18F1220           ; processor type  
radix          hex                 ; default radix for data  
config        WDT=OFF, LVP=OFF, OSC = INTIO2      ; Disable Watchdog timer, Low V. Prog, and RA6 as a clock  
  
#include       p18f1220.inc         ; This header file includes address and bit definitions for all SFRs  
#define       LastPortA            0x80      ; register to hold last value of port A  
  
org           0x000                 ;  
GOTO         StartL                 ; Executes after reset  
  
org           0x008                 ;  
GOTO         HPRIO                  ; Executes after high priority interrupt  
  
org           0x20                   ; Start of the code  
  
HPRIO:       ; high priority service code  
BTFSC       PIR1, TMR1IF  
BRA         TIMERL                   ; If Timer1 is interrupting then go to Timer1 Service code  
RETFIE      ; Return from interrupt  
  
TIMERL:     ;  
BCF         T1CON, TMR1ON           ; Disable Timer 1  
  
; Start of code to be executed during Timer1 interrupts  
MOVLW      .30  
CPFSEQ     CCPR1L  
BRA        Percent30
```

```

Percent10:    ; set PWM to 10%
MOV LW      .10
MOV W F     CCPR1L
BR A        Tldone
Percent30:    ; set PWM to 30%
MOV LW      .30
MOV W F     CCPR1L

Tldone:       ; get ready to return from interrupt
; Reset Timer 1 so next timer interrupt is in approximately 2 seconds
MOV LW      0x7D
MOV W F     TMR1L
MOV LW      0xE1
MOV W F     TMR1H

BCF         PIR1, TMR1IF      ; Clear Timer 1 Interrupt Flag
BSF         T1CON, TMR1ON     ; Enable Timer 1;
RET F IE     ; Return from interrupt

StartL:       ; entry point from reset
; Initialize all I/O ports
CLRF        PORTA            ; Initialize PORTA
CLRF        PORTB            ; Initialize PORTB
MOV LW      0x7F              ; Set all A/D Converter Pins as
MOV W F     ADCON1           ; digital I/O pins
MOV LW      0x0D              ; Value used to initialize data direction
MOV W F     TRISA            ; Set Port A direction
MOV LW      0xC7              ; Value used to initialize data direction
MOV W F     TRISB            ; Set Port B direction
MOV LW      0x00              ; clear Wreg

; wait until Switch 4 (RA3) has been changed before continuing
MOV F       PORTA,0
AND LW      0x08
MOV W F     LastPortA

WaitS:
MOV F       PORTA,0
AND LW      0x08
XOR W F     LastPortA,0
BZ          WaitS

; Timer 1 Initialization + interrupt enable/disable
BSF         INTCON, GIE      ; enable interrupts
BSF         INTCON, PEIE     ; enable all interrupts
BSF         PIE1, TMR1IE     ; enable Timer1 Interrupt
BSF         IPR1, TMR1IP     ; Set Timer 1 Interrupt to High priority
MOV LW      0x58              ; Timer 1: "8&8-bit, osc. clock, 1:2 pre-scale, enabled, internal clk"
MOV W F     T1CON            ; "0 1 0 1 0 0 0"
; Set Timer 1 so next timer interrupt is in approximately 2 seconds
; 2 sec x (106 usec/sec) x (sysClk/32 usec) x (instClk/4sysClk) x (Tick/2 instClk) = 7,812 Ticks
; set (TRM1H & TMRL) to { (216) ? 11,719 = 57725} or (E17D)H
MOV LW      0x7D
MOV W F     TMR1L
MOV LW      0xE1
MOV W F     TMR1H

BSF         T1CON, TMR1ON    ; Enable Timer 1

; Following 6 steps configure PWM power level based on the PICmicro Data Sheet starting at page 131
; 1) PWM will be delivered on P1A (pin 18) which controls Left Motor; for this code, use TOSC = 32 usec.
MOV LW      0x00C            ; "0000 1100
MOV W F     CCP1CON          ; PWM output on P1A (Pin 18)
; 2) PWM Requires Timer 2 and must be enabled for (PWM requires Timer 2)
CLRF        TMR2            ; Timer 2 Register
MOV LW      0x05            ; Enable timer and set pre-scale to 4
MOV W F     T2CON
BCF         PIR1, TMR2IF     ; Clear Timer 2 flag
; 3) Initialize PWM Period to PWM Period = (PR2 + 1) * 4 * TOSC * (TMR2 Pre-scale) = (99 + 1) * 4 * 32 usec * 4 = 51 msec
MOV LW      .99
MOV W F     PR2
; 4) Set PWM On-time to (CCPR1L:CCP1CON<5:4>)*TOSC*(TMR2 Pre-scale) = (CCPR1L:00)* 32 * 4 usec
; With this configuration, value stored in CCPR1L defines the duty cycle and therefore the % power level
MOV LW      .10
MOV W F     CCPR1L          ; Set the power level to 10%
; 5) Need to wait until timer2 has overflowed once and set PWM Pin 18 to output

```

```
WAITL:
    BTFSS    PIR1, TMR2IF
    BRA      WAITL
    BCF      TRISB,3      ; Set P1A/RB3/CCP1 as an output pin
    BSF      PORTB, 5     ; turn on LED just to indicate EDbot is on
```

```
MainL:    ; waiting in a loop
    ; Add main (non-interrupt) code that should be executed here.
```

```
    BRA     MainL
```

```
end      ; end of code
```

Experiment #1

Write an assembly code that controls the power delivered to EDbot's left motor using PWM functionality of PICmicro. The system is expected to perform the following steps:

1. Drive the motor at minimum power level (0% duty cycle)
2. Increase the percent of power delivered to the motor by 10% every three seconds until 40% of maximum power is achieved.
3. Decrease the percent of power delivered to the motor by 10% every three seconds until minimum power is achieved.
4. Reverse the motor direction
5. Go to step 1

It is recommended that you experiment with provided sample code to gain an understanding of PWM and Timer application prior to starting work on this experiment..

This experiment requires that you review your high level design (flow chart or pseudo code) and demonstrate your system to the instructor upon completion. Include the approval signature in your report.

Experiment #2

Write an assembly code that drives EDbot forward in circles. Initially, Edbot circles clockwise at 50% power level for 5 seconds and then Edbot circles counter clockwise at 20% for 5 seconds before stopping. *you may not use the built-in hardware PWM, therefore, you have to write a program that modulates (PWM) left and right motor drive pins.*

This experiment requires that you review your high level design (flow chart or pseudo code) and demonstrate your system to the instructor upon completion. Include the approval signature in your report.

Report Requirements

All reports must be computer printed (formulas and diagrams may be hand drawn) and at minimum include:

For each experiment:

- a) Clear problem statement; specify items given and to be found.
- b) Specific responses to each question asked in the experiment.
- c) Documentation of resulting high level design, disassembled code, system diagram, schematics and any other supporting material.

For the report as a whole

- a) Cover sheet with your name, course, lab title, date of completion and your teammates' name.
- b) Lessons learned from this lab.
- c) A new experiment and expected results which provide additional opportunity to practice the concepts in this lab.