ENGR 270 Lab #5 Online – Timers & Pulse Width Modulation

Objectives
Apply Timers to scheduling tasks and implement Pulse Width Modulation (PWM)

Preparation
Complete the following steps before starting to work on the experiments in this lab:

1) Complete Lab4 and associated report.
2) Complete lecture, homework and videos in Chapter 4 “Program Flow, Event Handling and Control”

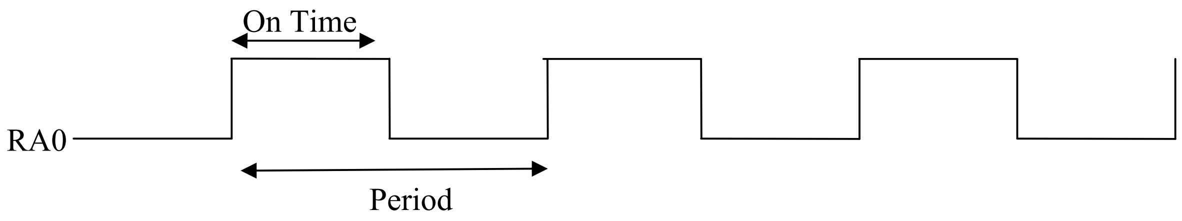
Following example code demonstrates the use of Timer 0 in order to generate an interrupt every one millisecond (mSec).

```
;--------------------------------------------------------------------------------
; File: main.asm
; Desc: Demonstrate Timer0 usage
;  * Use Timer 0 to count from 0 to 65535 (16-bit up counter).
;  * Duration for each count is 1 msec.
;  * Count is stored in  stateLSB and stateMSB
;  * Toggel RA0 every 256 uSec.
; Last Update: June, 2020
; Auth: Class
;--------------------------------------------------------------------------------

    list    p=18f1220   ;processor type
    radix   hex         ;default radix for data
    ; Disable Watchdog timer, Low V. Prog, and RA6 as I/O
    config  WDT=OFF,LVP=OFF,OSC=INTIO2

    #include    p18f1220.inc      ;header file

    #define     stateLSB    0x80   ;State variable is 16 bit with 1msec/increments
    #define     stateMSB    0x81


    org    0x000         ; Executes after rest
    GOTO    main

    org    0x008         ; Executes after high priority interrupt
    GOTO    HPRIO

    org    0x20      ; Code start here
main:
    ; initialize all I/O ports
    CLRF    stateLSB   ; clear LSB of current state variable
    CLRF    stateMSB   ; clear MSB of current state variable
    CLRF    PORTA     ; Initialize PORTA
    CLRF    PORTB      ; Initialize PORTB
    MOVLW   0x7F
    MOVWF   ADCON1    ; Configure all I/O as Digital
    MOVLW   0xFE
    MOVWF   TRISA     ; Set Port A<0> as output and rest of the bits as input
    MOVLW   0x60
    IORWF   OSCCON    ; Set internal Oscillator to frequency of 4 MHz

    ; Enable Timer0 Interrupt as High Priority
    BSF    INTCON, PEIE      ; enable all peripheral interrupts
    BSF    INTCON, TMR0IE    ; enable Timer 0 Interrupt
    BSF    INTCON2, TMR0IP   ; Set Timer 0 Interrupt to High priority
    BSF    RCON, IPEN        ; enable priority levels on interrupts
    BCF    INTCON, TMR0IF    ; clear Timer 0 Interrupt flag
    MOVLW  0x41              ; 8-bit, internal clock, 1:4 scale
    MOVWF  T0CON             ; Timer 0 tick is 4 usec
    CLRF   TMR0L             ; 256 count or 1 msec to Timer 0 Interrupt
    BSF    T0CON,TMR0ON      ; enable TMR0
    BSF    INTCON, GIE       ; enable interrupts globally

MainL:     ; Main loop - waiting
    NOP
    BRA    MainL

; Interrupt Handling Section
HPRIO:    ; High priority interrupts including Timer 0 Int.
    BTFSC   INTCON, TMR0IF  ; Check for Timer 0 high priority Interrupt
    BRA     TMR0int
    BRA     HPRIOdone    ; return from interrupt
TMR0int:    ; handel Timer 0 Interrupt
    ; Increment 16-bit variable state every 1 mSec.
    INCF    stateLSB
    TSTFSZ  stateLSB
    BRA     HPRIOdone
    INCF    stateMSB
    BTG     PORTA,0      ; toggle RA0

HPRIOdone:   ; Clean up and return from interrupt
    BCF     INTCON, TMR0IF   ; Clear Timer 0 interrupt Flag
    RETFIE

 end       ; end program
```

Experiment 1. Generating PWM using TIMER0

Pulse Width Modulation (PWM) is used to control power delivered to devices such as lighting systems, motors, etc. The power delivered is proportional to duty cycle. For example, if a signal is set to 50% duty cycle then the power delivered will also be 50% of maximum power. Duty cycle is calculated using the following equation:

$$\text{Percent of Maximum Power} = \mathit{Duty\ Cyle} = \frac{100 * (\text{On Time})}{\mathit{Period}}\%$$



For this experiment, write an assembly code that toggles RB0 every 0.256 seconds. Additionally, the code generates a PWM signal output at RA0 with the frequency of 250 Hz (Period of 4 mSec). PWM power level should change every 0.512 seconds according to the following steps:

1) 100% of maximum power
2) 75% of maximum power
3) 50% of maximum power
4) 25% of maximum power
5) 0% of maximum power
6) 25% of maximum power
7) 50% of maximum power
8) 75% of maximum power
9) 100% of maximum power
10) Go to step 1

Note: MPLAB Logic Analyzer generates graphical representation of the simulation results. You have the option of selecting the graph's horizontal axis (domain) as Real Time or Instruction Cycle. More information is available if you search for "Using the Simulator Logic Analyzer" in Help section of MPLAB.

Report Requirements

This lab and associated report must be completed individually. All reports must be computer printed (formulas and diagrams may be hand drawn) and at minimum:

For each experiment include:

- Clear problem statement in your words.
- Answer to any specific experiment questions (if any)
- Pseudo code which may be written in C-like syntax
- Disassembled code available after successful assembling
- Test plan which describes the input values and expect output/memory values for a successful design.
- Simulator output which shows the stimuli, relevant memory locations values showing validation based on test plan.

For the whole report include:

- A Cover sheet with your name, class, lab and completion date.
- A Lessons Learned section which summarizes your learning from this lab.
- A New Experiment section that has description of a new experiment and the experiment's results. Experiment should be related to material covered in class but not similar to one of the experiments in this lab.