วัตถุประสงค์ของการทดลองปฏิบัติการ

1. ทำความเข้าเกี่ยวกับความสัมพันธ์ระหว่างวัฏจักรเครื่องจักรกล (Machine cycle) และความถี่ของตัวกำเนิดสัญญาณนาฬิกาเช่น Crystal oscillator
2. เรียนรู้และทำความเข้าใจการหน่วงเวลาโดยใช้กระบวนการทางซอฟแวร์ และนำความรู้ที่ได้ไปประยุกต์ใช้ในการเขียนโปรแกรมย่อยสำหรับการหน่วงเวลา
3. ปฏิบัติการทดลองประยุกต์ใช้โปรแกรมย่อยหน่วงเวลาในการควบคุมไฟวิ่ง LED

---

1 ทฤษฎีที่เกี่ยวข้อง

จากที่ได้เรียนรู้มาแล้วกันในชั้นเรียนว่า PIC microcontroller ในตระกูล 14-bit core เช่น PIC16F628A จะมีความสามารถในการ execute คำสั่งหนึ่งคำสั่งเสร็จสิ้นภายในเวลา 4 ลูกคลื่นของสัญญาณนาฬิกา (หรือคิดเป็น 1 machine cycle) เช่นถ้า PIC16F628A ถูกต่ออยู่กับ X'tal (crystal) ที่มีความถี่ 4 MHz คำสั่งหนึ่งคำสั่งจะใช้เวลาเพียง 1 ไมโครวินาที เราสามารถนำความรู้นี้มาประยุกต์ใช้ในการสร้างโปรแกรมย่อยสำหรับหน่วงเวลาให้ไมโครคอนโทรลเลอร์ช้าลงได้ เหตุผลหลักในการหน่วงเวลาไมโครคอนโทรลเลอร์ก็คือ ไมโครคอนโทรลเลอร์มีความสามารถในการประมวลผลที่รวดเร็วมาก (เช่นสามารถทำงาน execute คำสั่งได้เป็นจำนวน 1 ล้านคำสั่ง ภายในเวลา 1 วินาทีที่ความถี่ของสัญญาณนาฬิกา 4 MHz เป็นต้น) ที่ความเร็วสูงนี้มนุษย์จะไม่สามารถติดตามหรือสังเกตผลของการประมวลผลได้ทัน จึงมีความจำเป็นที่จะต้องหน่วงเวลาให้ไมโครคอนโทรลทำงานช้าลง

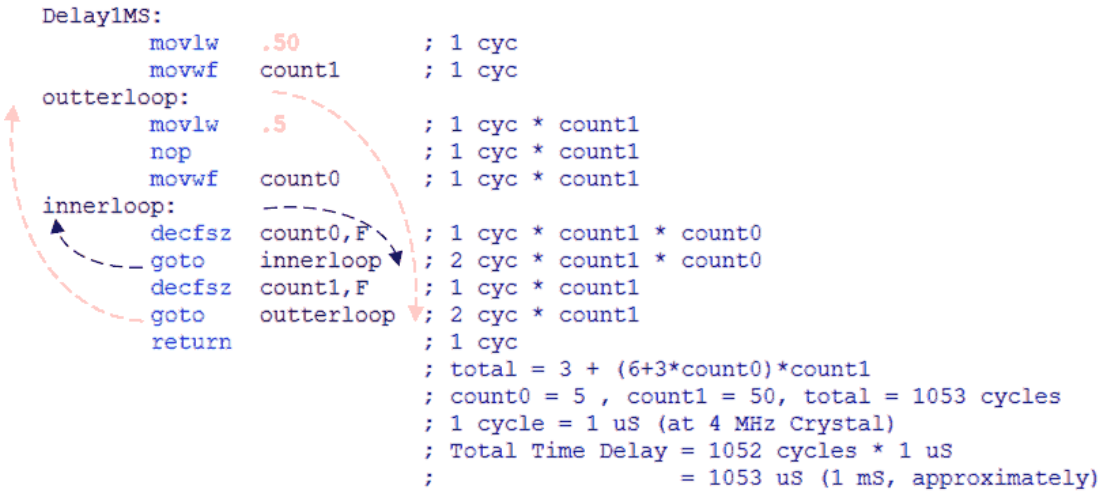
การหน่วงเวลาทางซอฟแวร์สามารถทำได้โดยการสร้างส่วนของโปรแกรมที่วนซ้ำๆ เป็นลูป (Loop) โดยในลูปนี้จะต้องมีการนับจำนวนของ machine cycles ภายในลูปและจำนวนครั้งที่ลูปนี้วนซ้ำ ดังแสดงในตัวอย่างในรูปที่ 1 จากตัวอย่างนี้จะเห็นได้ว่าโปรแกรมจะเสียเวลาในการวนซ้ำเป็นจำนวน count=10 รอบ โดยอาศัยคำสั่ง decfsz f,d ซึ่งจะลดค่าของ file register (count) ลงหนึ่งแล้วตรวจสอบว่ามีค่าเป็นศูนย์หรือไม่ ถ้า count ยังไม่มีค่าเป็นศูนย์ก็จะกระโดดกลับไปทำซ้ำ เนื่องจาก count ถูกกำหนดให้มีค่าเริ่มต้นเท่ากับ 10 จึงทำให้มีการวนลูปเป็นจำนวน 10 ครั้ง จากตารางสรุปคำสั่งของ PIC microcontroller จะพบว่าคำสั่ง movlw, movwf, nop, return จะกินเวลาในการ execute เท่ากับ 1 machine cycle ในขณะที่คำสั่ง goto จะกินเวลา 2 machine cycles เมื่อนำจำนวน machine cycles ของคำสั่งทั้งหมดในโปรแกรมย่อยและจำนวนครั้งที่มีการวนลูปมาใช้คำนวณ จะพบว่าในการเรียกโปรแกรมย่อยนี้ 1 ครั้ง (โดยใช้คำสั่ง call Delay) จะทำให้ไมโครคอนโทรลเลอร์เสียเวลาไปประมาณ 53 ไมโครวินาที (ที่ความถี่สัญญาณนาฬิกา 4 MHz)

```
Delay:
        movlw   .10         ; 1 cyc
        movwf   count       ; 1 cyc
Loop:
        nop                 ; 1 cyc * 10
        nop                 ; 1 cyc * 10
        decfsz  count,F     ; 1 cyc * 10
        goto    Loop        ; 2 cyc * 10
        return              ; 1 cyc
                            ; Total cycles = 1+1+10+10+10+20+1 = 53 cycles
                            ; If 1 cycle = 1 uS
                            ; Total time delay = 53*1 uS = 53 uS
```

Figure 1: โปรแกรมย่อยการหน่วงเวลาแบบวนลูปเดี่ยว Single loop time delay subroutine

ช่วงเวลาการหน่วงในตัวอย่างในรูปที่ 1 สามารถถูกยืดออกได้โดยการแทรกคำสั่งให้เสียเวลาเช่น nop (ย่อมาจาก no operation) เพิ่มเข้าไปในลูป แต่การกระทำแบบนี้จะไม่มีประสิทธิภาพมากนัก เรามีทางเลือกที่ดีกว่าโดยการเขียนโปรแกรมให้มีลักษณะเป็นลูปซ้อนอยู่ในลูปหรือ Nested loop ดังแสดงในรูปที่ 2 ในกรณีนี้โปรแกรมย่อยนี้จะมีลูปในหรือ innerloop ซ้อนอยู่ภายในลูปนอก outterloop โดยที่ทุกๆ ครั้งที่มีการวนซ้ำของลูปนอกลูปในก็จะถูกวนซ้ำทุกครั้งไปด้วย ดังนั้นเราจึงสามารถเขียนสมการของจำนวน machine cycles ในเทอมของตัวแปร (file

registers จำนวนสองตัวคือ count0 และ count1) ในตัวอย่างนี้ ถ้ากำหนดให้ count0 = 5 และ count1 = 50 จะสามารถคำนวณช่วงเวลาการหน่วงของโปรแกรมย่อยนี้ได้เป็น 1053 ไมโครวินาทีหรือ 1 มิลลิวินาทีโดยประมาณ



```
Delay1MS:
        movlw   .50         ; 1 cyc
        movwf   count1      ; 1 cyc
outterloop:
        movlw   .5          ; 1 cyc * count1
        nop                 ; 1 cyc * count1
        movwf   count0      ; 1 cyc * count1
innerloop:
        decfsz  count0,F    ; 1 cyc * count1 * count0
        goto    innerloop   ; 2 cyc * count1 * count0
        decfsz  count1,F    ; 1 cyc * count1
        goto    outterloop  ; 2 cyc * count1
        return              ; 1 cyc
                            ; total = 3 + (6+3*count0)*count1
                            ; count0 = 5 , count1 = 50, total = 1053 cycles
                            ; 1 cycle = 1 uS (at 4 MHz Crystal)
                            ; Total Time Delay = 1052 cycles * 1 uS
                            ;                  = 1053 uS (1 mS, approximately)
```

Figure 2: โปรแกรมย่อยการหน่วงเวลาแบบวนลูปซ้อน Nested loop time delay subroutine

ตัวอย่างในรูปที่ 3 แสดงการคำนวณหาค่าจำนวนครั้งที่โปรแกรมย่อยต้องวนซ้ำทั้งในลูปในและลูปนอก (count0 และ count1) หรือการคำนวณหาค่าคงที่ (literal values) X_var และ Y_var นั่นเอง ในตัวอย่างนี้แสดงการคำนวณหาค่าของ X_var และ Y_var ที่จะสามารถหน่วงเวลาได้ 100 มิลลิวินาที โดยจำเป็นต้องกำหนดค่าให้กับ X_var หรือ Y_var ตัวใดตัวหนึ่งเสียก่อน จึงจะสามารถแก้สมการคำนวณหาค่าที่เหลือได้ ในกรณีนี้กำหนดให้ X_var = 250 (ข้อสังเกต: เนื่องจาก file registers ทั้งสองตัว count0, count1 เป็น file registers ขนาด 8-bit ทำให้ file registers ทั้งสองนี้สามารถเก็บจำนวนเลขที่เป็นบวกที่มีค่าระหว่าง 0-255 เท่านั้น) ซึ่งจะสามารถแก้สมการได้ค่า Y_var = 131 (โดยประมาณ)

```
Delay:
        movlw   X_var       ; 1 cyc
        movwf   count0      ; 1 cyc
outterloop:
        nop                 ; 1 cyc * X_var
        nop                 ; 1 cyc * X_var
        nop                 ; 1 cyc * X_var
        movlw   Y_var       ; 1 cyc * X_var
        movwf   count1      ; 1 cyc * X_var
innerloop:
        decfsz  count1,F    ; 1 cyc * X_var * Y_var
        goto    innerloop   ; 2 cyc * X_var * Y_var
        decfsz  count0,F    ; 1 cyc * X_var
        goto    outterloop  ; 2 cyc * X_var
        return              ; 1 cyc
                            ; Total cycles = 3 + (8+3*Y_var)*X_var
                            ; 1 cycle = 1 uS (at 4 MHz Crystal)
                            ; Total Time Delay = (13 + (8+3*Y_var)*X_var) * 1 uS
                            ; If we want a delay time of 100 mS
                            ; 100 mS = (3 + (8+3*Y_var)*X_var) * 1 uS
                            ; assume that X_var = 250
                            ; 100 mS = (3 + (8+3*Y_var)*250) * 1 uS
                            ; solve for Y_var
                            ; Y_var =  (100000-3-(8*250))/(3*250)
                            ; Y_var = 130.66 or (131 approximately)
```

Figure 3: ตัวอย่างแสดงการคำนวณหาจำนวนรอบการวนซ้ำในลูปในและลูปนอกของโปรแกรมย่อยการหน่วงเวลาแบบวนลูปซ้อน Nested loop time delay subroutine

2 การทดลองปฏิบัติการ

พิจารณาวงจรไฟวิ่ง LED 8 ดวงในรูปที่ 4 และโปรแกรมควบคุมต่อไปนี้ จะเห็นว่าโปรแกรมควบคุมนี้ยังไม่เสร็จสิ้นสมบูรณ์ ทำให้โปรแกรมนี้เสร็จสมบูรณ์โดยการเขียนโปรแกรมย่อย Delay1_5S เพื่อหน่วงเวลาเป็นเวลา 1.5 วินาที เพื่อให้ได้คะแนนการทดลองปฏิบัติการนี้เต็มนักศึกษาจะต้อง

1. แสดงการคำนวณจำนวนครั้งการวนลูปของโปรแกรมย่อยหน่วงเวลาในพื้นที่ว่างต่อไปนี้
2. เขียนเฉพาะโปรแกรมย่อย Delay1_5S ในพื้นที่ว่างต่อไปนี้
3. ทำการจำลองการทำงานของโปรแกรมบน Proteus ISIS
4. ทำการดาวน์โหลดโปรแกรมที่ได้ลงสู่บอร์ด PCK-1000 และต่อวงจรจาก PortB เข้ากับ LED ทั้ง 8 ดวงและสาธิตการทำงานของวงจรจริงให้กับอาจารย์ผู้สอน

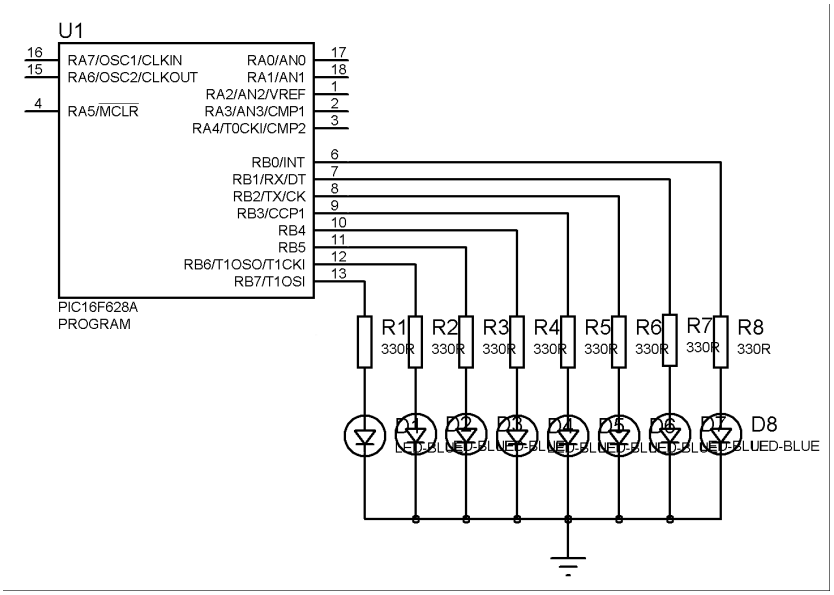


Figure 4: วงจรไฟวิ่งแบบ 8 ดวง

```
;*****************************************
;**** Program title:  Running LEDS  ********
;**** Programmer: Mr. Chatchai   ***********
;*****************************************

                PROCESSOR PIC16F628
                #include <P16F628.INC>
                __CONFIG        _CP_OFF & _MCLRE_OFF & _INTRC_OSC_NOCLKOUT & _LVP_OFF &
                    _WDT_OFF

;****** Define general purpose registers for temporary variables
                cblock  0x20
                        temp
                        count
                        count0
                        count1
                        count2
                endc
;*********************************************************
                ORG     0x00            ; Reset Vector

```

```
                banksel TRISB   ; Switch to bank1
                clrw
                movwf   TRISB   ; Make PortB an output port
                banksel PORTB   ; Switch back to bank0
                movwf   PORTB   ; Turn-off all LEDS

Main_loop:                      ; Main loop begins here
                movlw   0x00    ; clear file register 'temp'
                movwf   temp
Again:                                  ; repeat this loop 8 times
                movf    temp,w          ; use 'temp' to get a LED pattern from
                call    LED_PATTERN     ; LED_PATTERN look-up table
                movwf   PORTB           ; move the obtained LED pattern to PORTB
                call    Delay1_5S       ; Delay for 1.5 second

                incf    temp,f          ; increment 'temp' by one

                movf    temp,w
                sublw   .8              ; check if [temp] == 8 ?
                btfss   STATUS,Z
                goto    Again           ; if 'no' repeat this loop again
                goto    Main_loop       ; if 'yes' clear 'temp' back to zero


;******************** Subroutines *****************************

;=========================================================
;* Running LED patterns using a look-up table
;=========================================================
LED_PATTERN:
                addwf   PCL,F
                retlw   B'00000000'             ; Pattern 0
                retlw   B'10000001'             ; Pattern 1
                retlw   B'01000010'
                retlw   B'00100100'
                retlw   B'00011000'
                retlw   B'00100100'
                retlw   B'01000010'
                retlw   B'10000001'             ; Pattern 7

;=========================================================
; Delay subroutine for 1.5 second
;=========================================================
Delay1_5S:
                :
                :
                :
                :
                :
                return

                END
```

(แสดงการคำนวณและเขียนโปรแกรมในพื้นที่ว่างต่อไปนี้)

Laboratory #3

Software Delay Time Generation

Name _____ ID _____

(แสดงการคำนวณและเขียนโปรแกรมในพื้นที่ว่างต่อไปนี้)