Lab-8
PIC microcontroller Timer Programming using assembly language

Objective:

In this lab students will learn

- How many timers are there in PIC18f4550
- How timers are used for introducing delays.

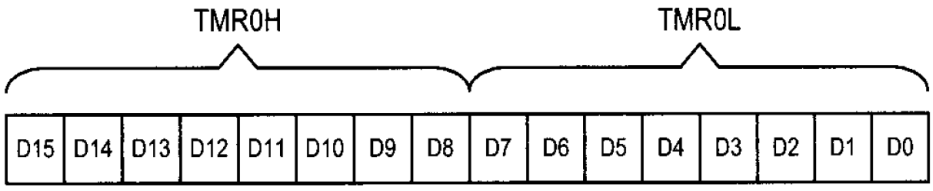
Theory

The PIC18 has two to five timers depending on the family member. They are referred to as Timers 0, 1, 2, 3, and 4. They can be used either as timers to generate a time delay or as counters to count events happening outside the microcontroller.

Basic registers of the timer

Many of the PIC18 timers are 16 bits wide. Because the PIC18 has an 8-bit architecture, each 16-bit timer is accessed as two separate registers of low byte (TMRxL) and high byte (TMRxH). Each timer also has the TCON (timer control) register for setting modes of operation. Next, we discuss each timer separately.

Timer0 register and programming



Each timer has a control register, called TCON, to set the various timer operation modes. T0CON is an 8-bit register used for control of Timer0. The bits for T0CON are shown in Figure 9-2.

T0CS (Timer0 clock source)

This bit in the T0CON register is used to decide whether the clock source is internal (Fosc/4) or external. If T0CS = 0, then the Fosc/4 is used as clock source. In this case, the timers are often used for time delay generation. See Example 9-1. If T0CS = 1, the clock source is external and comes from the RA4/T0CKI, which is pin 6 on the DIP package of PIC1818F4580/4520. When the clock source comes from an external source, the timer is used as an event counter. We will discuss that option in the next section. See Example 9-2.

TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0

TMR0ON D7 Timer0 ON and OFF control bit
1 = Enable (start) Timer0
0 = Stop Timer0

T08BIT D6 Timer0 8-bit/16-bit selector bit
1 = Timer0 is configured as an 8-bit timer/counter.
0 = Timer0 is configured as a 16-bit timer/counter.

T0CS D5 Timer0 clock source select bit
1 = External clock from RA4/T0CKI pin
0 = Internal clock (Fosc/4 from XTAL oscillator)

T0SE D4 Timer0 source edge select bit
1 = Increment on H-to-L transition on T0CKI pin
0 = Increment on L-to-H transition on T0CKI pin

PSA D3 Timer0 prescaler assignment bit
1 = Timer0 clock input bypasses prescaler.
0 = Timer0 clock input comes from prescaler output.

T0PS2:T0PS0 D2D1D0 Timer0 prescaler selector

0 0 0 = 1:2	Prescale value (Fosc / 4 / 2)
0 0 1 = 1:4	Prescale value (Fosc / 4 / 4)
0 1 0 = 1:8	Prescale value (Fosc / 4 / 8)
0 1 1 = 1:16	Prescale value (Fosc / 4 / 16)
1 0 0 = 1:32	Prescale value (Fosc / 4 / 32)
1 0 1 = 1:64	Prescale value (Fosc / 4 / 64)
1 1 0 = 1:128	Prescale value (Fosc / 4 / 128)
1 1 1 = 1:256	Prescale value (Fosc / 4 / 256)

16-bit timer programming

The following are the characteristics and operations of 16-bit mode:

1. It is a 16-bit timer; therefore, it allows values of 0000 to FFFFH to be loaded into the registers TMR0H and TMR0L.
2. After TMR0H and TMR0L are loaded with a 16-bit initial value, the timer must be started. This is done by "`BSF T0CON,TMR0ON`" for Timer0.
3. After the timer is started, it starts to count up. It counts up until it reaches its limit of FFFFH. When it rolls over from FFFFH to 0000, it sets HIGH a flag bit called TMR0IF (timer interrupt flag, which is part of the INTCON register). This timer flag can be monitored. When this timer flag is raised, one option would be to stop the timer.
4. After the timer reaches its limit and rolls over, in order to repeat the process, the registers TMR0H and TMR0L must be reloaded with the original value, and the TMR0IF flag must be reset to 0 for the next round.

Lab Exercise

In the following program, we are creating a square wave of 50% duty cycle (with equal portions high and low) on the PORTB.5 bit. Timer0 is used to generate the time delay. Analyze the program.

```
        BCF    TRISB,5          ;PB5 as an output
        MOVLW  0x08             ;Timer0,16-bit,int clk,no prescale
        MOVWF  T0CON            ;load T0CON reg.
HERE    MOVLW  0xFF             ;TMR0H = FFH, the high byte
        MOVWF  TMR0H            ;load Timer0 high byte
        MOVLW  0xF2             ;TMR0L = F2H, the low byte
        MOVWF  TMR0L            ;load Timer0 low byte
        BCF    INTCON, TMR0IF   ;clear timer interrupt flag bit
        BTG    PORTB,5          ;toggle PB5
        BSF    T0CON, TMR0ON    ;start Timer0
AGAIN   BTFSS  INTCON, TMR0IF   ;monitor Timer0 flag until
        BRA    AGAIN            ;it rolls over
        BCF    T0CON, TMR0ON    ;stop Timer0
        BRA    HERE             ;load TH, TL again
```

Also calculate the delay introduced. **[5]**

Write down the steps for programming the timer in 16 bit mode. **[5]**

Calculate the frequency of square waveform generated at pin PORTB.5. Show Simulations on Proteus. **[5]**

```
                                                    Cycles
            BCF   TRISB,5
            MOVLW 0x08
            MOVWF T0CON
            BCF   INTCON, TMR0IF

HERE        MOVLW 0xFF                                1
            MOVWF TMR0H                               1
            MOVLW -D'48'                              1
            MOVWF TMR0L                               1
            CALL  DELAY                               1
            BTG   PORTB,5                             1
            BRA   HERE                                1
;-----------delay using Timer0
DELAY       BSF   T0CON, TMR0ON                       1
AGAIN       BTFSS INTCON, TMR0IF                      1
            BRA   AGAIN                               1
            BCF   T0CON, TMR0ON                       1
            BCF   INTCON, TMR0IF                      1
            RETURN                                    1
                                                     13
```

Introduction to Embedded Systems **Marks: _____/33_____**

GCU, LAHORE **ROLL NO. _____**

Write a program to generate a square waveform with a period of 20msec on pin PORTB.3. Show Simulations on Proteus and verify your calculations. [5]

Generate a square waveform of 3KHz frequency on pin PORTB.3. Show Simulations on Proteus and verify your calculations. [5]

Compare the 8-bit and 16-bit timer mode of PIC microcontroller [2]

What is the minimum frequency that you can generate using 16-bit timer mode? [3]

Conclusion [3]