Chapter 6 - Timer – "take-aways"

- Examples of timer application: scheduling an event, measuring elapsed time.

- Difference: Timer (period fixed & known) vs. Counter (period not fixed or known).

- **Timer 0**: 8 / 16 bit, pre-scalar, selectable clock source, interrupt flag set on overflow…

- Code below switch on LED sometime after switch is pressed.
    - while(PORTBbits.RB0 == 1);   // active low switch at RB0
    - Delay… // 0.1 s delay
    - PORTDbits.RD0 = 1;   // active high LED at RD0

Without using timer

- Timer 0 registers: $\underbrace{\text{TMR0H TMR0L}}_{\text{16 bit timer/counter}}$ $\underbrace{\text{TMR0IF}}_{\text{bit 2 of INTCON register}}$ T0CON

  set to 1 whenever timer overflows from 65535 to 0

TMR0ON 1=ON	T08BIT 0=16bit	T0CS 0=internal	T0SE 0=rising	PSA 0=use pre-scalar	T0PS2 T0PS1 T0PS0 100 = use 1:32 etc

- Code below uses Timer0 to switch on LED sometime after switch is pressed:
    - **T0CON = 0b10000100;**   // Timer on, 16-bit, Fosc/4, pre-scalar 32
    - while (PORTBbits.RB0 == 1);   // wait for button to be pressed
    - **TMR0H = 0x6D;**   // always write to TMR0H before TMR0L
    - **TMR0L = 0x84;**
    - **while (INTCONbits.TMR0IF == 0);**   // wait here for Timer0 overflow

      // the previous 3 lines gives a delay of 0.1 s
    - INTCONbits.TMR0IF = 0;   // clear the flag – **why?**
    - PORTDbits.RD0 = 1;   // turn on LED

Using timer

Summarized

- Timer0 delay (outline) [focus on 4 lines in bold above]
    - Use T0CON to configure and turn on the timer.
    - Write an appropriate "starting count" value to TMR0H:TMR0L.
    - Wait for the timer to overflow i.e. TMR01F to become 1.

- To lengthen the delay, use 16 bit (instead of 8 bit), use bigger pre-scalar, use lower frequency clock, start counting from a smaller number.

- Code below measures elapsed time:
    - **T0CON = 0b10000100;**   // Timer on, 16-bit, Fosc/4, pre-scalar 32
    - while (PORTBbits.RB0 == 0);   // wait for signal at RB0 to go high
    - **TMR0H = 0x00; TMR0L = 0x00;**   // reset timer -- always write to TMR0H first
    - while (PORTBbits.RB0 == 1);   // wait for signal at RB0 to go low
    - T0CONbits.TMR0ON = 0;   // stop timer
    - **TempLow = TMR0L;**   // read TMR0L first
    - **TempHigh = TMR0H;**
    - **Time_elapsed = TempHigh * 256 + TempLow;** // Time_elapsed x 2.667 us gives the "real time" elapsed

- **Timer 2**: 8 bit, pre-scalar & post-scalar, internal clock source, interrupt flag set on Timer & Period match…

- Timer 2 registers: TMR2 PR2 TMR2IF T2CON

  TMR2 → 8 bit timer

  PR2 → 8 bit period

  TMR2IF → bit 1 of PIR1 register

  set to 1 whenever TMR2=PR2

  T2CON ↓

D7	TOUTPS3 TOUTPS2 TOUTPS1 TOUTPS0	TMR2ON	T2CKPS1 T2CKPS0
not used	0000 = use 1:1, 0001 use 1:2 etc	1 = ON	00 = use 1:1 etc

- "Period" set to fixed value, "Timer" counts up. When they match, TMR2IF flag set to 1. (Post-scalar can be used so that a few matches are needed, before flag is set.)

- Code below turns on LED at RD0 when TMR2 reaches 100:
  - **T2CON = 0x00;** // Timer2, no pre-/post- scalar
  - **TMR2 = 0x00;** // TMR2 = 0
  - **PR2 = 100;** // load period register 2
  - **T2CONbits.TMR2ON = 1;** // turn ON Timer2
  - **while (PIR1bits.TMR2IF == 0);** // wait for TMR2IF to be raised
  - PORTDbits.RD0 = 1; // turn ON LED at RD0

- **PWM** = Pulse Width Modulation -- can be used for motor speed control.
- 25% duty cycle => high 25% of the time, low 75% of the time.
- 5V output, but 50% duty cycle, is equivalent to 2.5V output.
- Both period & high time must be known, to draw a PWM waveform.

- PIC18F4550 can produce PWM wave at RC2. So, make RC2 an output: **TRISCbits.TRISC2 = 0;**
- Last 4 bits of CCP1CON register must be 1100. So: **CCP1CON = 0b00 00 1100;**

- Example: **5 kHz, 25% duty cycle wave** is required… Period = 0.2 m.
  - Formula: PWM period = (PR2 + 1) x 4 x N x Tosc

    ⇨ where N = pre-scalar value of 1, 4 or 16 and Tosc = 1 / Fosc = 1 / 48M
  - Let's use N = 16, 0.2 m = (PR2 + 1) x 4 x 16 x (1 / 48M) $\Rightarrow$ **PR2 = 149;**
  - High Time = 25% of Period = 25% x 149 = 37.25 = 37 (truncating decimal portion) $\Rightarrow$ **CCPR1L = 37;**

- Code below produces a 5 kHz, 25% duty cycle wave at RC2:
  - **TRISCbits.TRISC2 = 0;** // RC2 as output
  - **CCP1CON = 0b00 00 1100;** // PWM mode
  - **PR2 = 149;** // Period
  - **CCPR1L = 37;** // High Time
  - TMR2 = 0; // start from 0
  - **T2CON = 0b0 0000 1 10;** // no post-scale, Timer2 on, pre-scale 16
  - while (1) {
  - // PWM wave generated continuously at RC2, without further code
  - // PIC free to do other things……
  - }