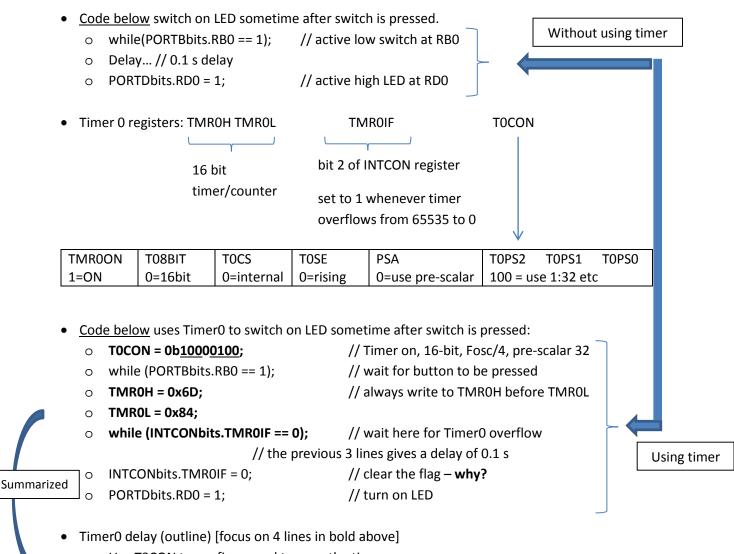
Chapter 6 - Timer - "take-aways"

- 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...

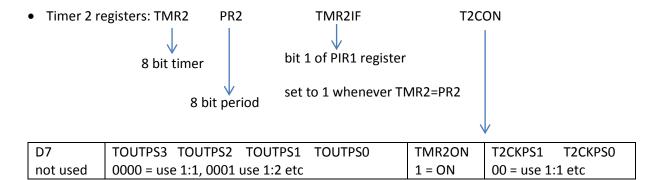


- o Use TOCON to configure and turn on the timer.
- Write an appropriate "starting count" value to TMR0H:TMR0L.
- Wait for the timer to overflow i.e. TMR01F to become 1.
- To lengthen the delay, use 16 bit (instead of 8 bit), use bigger pre-scalar, use lower frequency clock, start counting from a smaller number.
- <u>Code below</u> measures elapsed time:

```
O TOCON = 0b10000100;
                                 // Timer on, 16-bit, Fosc/4, pre-scalar 32
o while (PORTBbits.RB0 == 0);
                                 // wait for signal at RBO to go high
o TMR0H = 0x00; TMR0L = 0x00; // reset timer -- always write to TMR0H first
o while (PORTBbits.RB0 == 1);
                                 // wait for signal at RBO to go low
O TOCONbits.TMR0ON = 0;
                                 // stop timer
o 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...



- "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.

```
o Formula: PWM period = (PR2 + 1) \times 4 \times N \times Tosc

\Rightarrow where N = pre-scalar value of 1, 4 or 16 and Tosc = 1 / Fosc = 1 / 48M

o Let's use N = 16, 0.2 m = (PR2 + 1) \times 4 \times 16 \times (1 / 48M) \Rightarrow PR2 = 149;

o 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......
}
```