Chapter 5
PIC18 Timers

Timers

- Counter is a register that can be loaded with a binary number (count) which can be incremented per clock/instruction cycle.
- We can use this counter to measure time.
- Time is calculated as follows:
  - Find difference between beginning count and last count
  - Multiply the count difference by the clock/instruction period

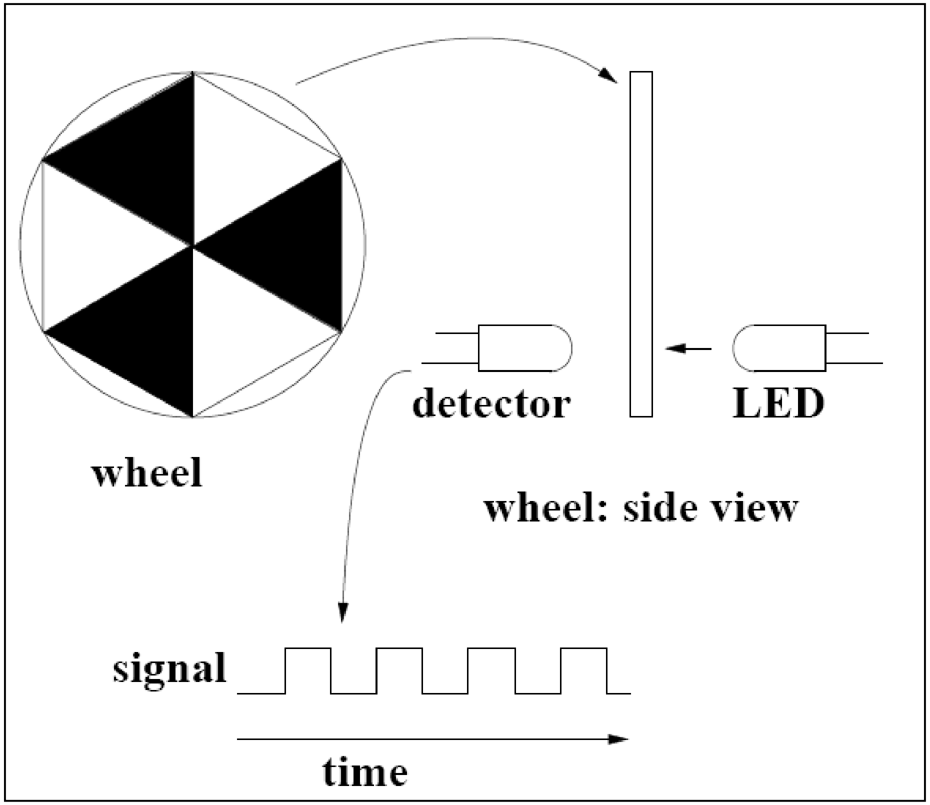
Counters for External Events

- The register can also be used as a counter for external events by replacing the clock with a signal from an event.
- When a signal from an event arrives, the count in the register is incremented (or decremented); thus, the total number of events can be counted.

Timer Applications

- Time delay
- Pulse wave generation
- Timer as an event counter
- Frequency measurement

Application: How long you have travelled by a bicycle?



Distance

= No. of times the wheel rotates × Wheel circumference

$= \frac{No. of\ square\ signal\ periods}{3} \times$ Wheel circumference

http://www.cs.ou.edu/~fagg/classes/es_general/timers.pdf

PIC18 Timers

- The PIC18 microcontroller has multiple timers
- Timers are divided into two groups: 8-bit and 16-bit
- Labeled as Timer 0 to Timer 2
  - Timer 0 can be set up as an 8-bit or 16-bit timer.
  - Timer 1 is a 16-bit timer.
  - Timer 2 is an 8-bit timer.
- Each timer associated with its Special Function Register (SFR): T0CON-T2CON

Timer 0

- Can be set up as an 8-bit or 16-bit timer
  - Parameters are set up by bits in T0CON register
  - Has eight options of pre-scale values (Bit2-Bit0)
  - Can run on internal clock source (instruction cycle) or external clock connected to pin RA4/T0CK1
  - Generates an interrupt or sets a flag when it overflows from FF_H to 00_H in the 8-bit mode and from $FFFF_H$ to 0000_H in the 16-bit mode
  - Can be set up on either rising edge or falling edge (Bit4) when an external clock is used

T0CON (Timer 0 Control) Register

REGISTER 10-1: T0CON: TIMER0 CONTROL REGISTER

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
bit 7							bit 0

bit 7 **TMR0ON:** Timer0 On/Off Control bit

1 = Enables Timer0
0 = Stops Timer0

bit 6 **T08BIT**: Timer0 8-bit/16-bit Control bit

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

bit 5 **T0CS**: Timer0 Clock Source Select bit

1 = Transition on T0CKI pin
0 = Internal instruction cycle clock (CLKO)

bit 4 **T0SE**: Timer0 Source Edge Select bit

1 = Increment on high-to-low transition on T0CKI pin
0 = Increment on low-to-high transition on T0CKI pin

bit 3 **PSA:** Timer0 Prescaler Assignment bit

1 = TImer0 prescaler is NOT assigned. Timer0 clock input bypasses prescaler.
0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

bit 2-0 **T0PS2:T0PS0**: Timer0 Prescaler Select bits

111 = 1:256 prescale value
110 = 1:128 prescale value
101 = 1:64 prescale value
100 = 1:32 prescale value
011 = 1:16 prescale value
010 = 1:8 prescale value
001 = 1:4 prescale value
000 = 1:2 prescale value

T0CON (Timer 0 Control) Register

- TMR0ON = T0CON<7>: On (1) or Off (0)
- T08BIT = T0CON<6>: 8-bit (1) or 16-bit (0)
- T0CS = T0CON<5>: Use External (1) or Internal (0) clock
- T0SE = T0CON<4>: Increment on H-L (1) or L-H (0) transition
- PSA = T0CON<3>: Use (0) or Do not use (1) prescaling
- T0PS2:T0PS0 = T0CON<2:0>: Specify prescaler factor

Internal Clock Mode (T0CS = 0)

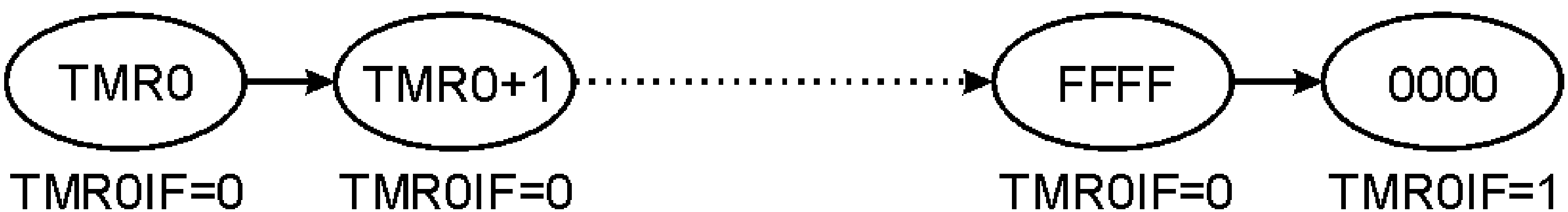
- Clock source is the instruction cycle clock
- The Timer 0 Interrupt Flag (TMR0IF = INTCON<2>) is raised when counter rollover from max value (FFFF in 16-bit mode, FF in 8-bit mode) to min value (0000 in 16-bit mode, 00 in 8-bit mode)
- Used as a timer to create delay

Create delay using Timer 0 16-bit Mode

- Characteristics of 16-bit mode
  - 16-bit timer – allows values of 0000 to FFFF to be loaded into registers TMR0H and TMR0L
  - The timer must be started by setting the TMR0ON (i.e., `bsf T0CON, TMR0ON`)
  - The timer counts up after being started. When it rolls over from FFFF to 0000, it sets, the interrupt flag, TMR0IF.

Steps to create delay

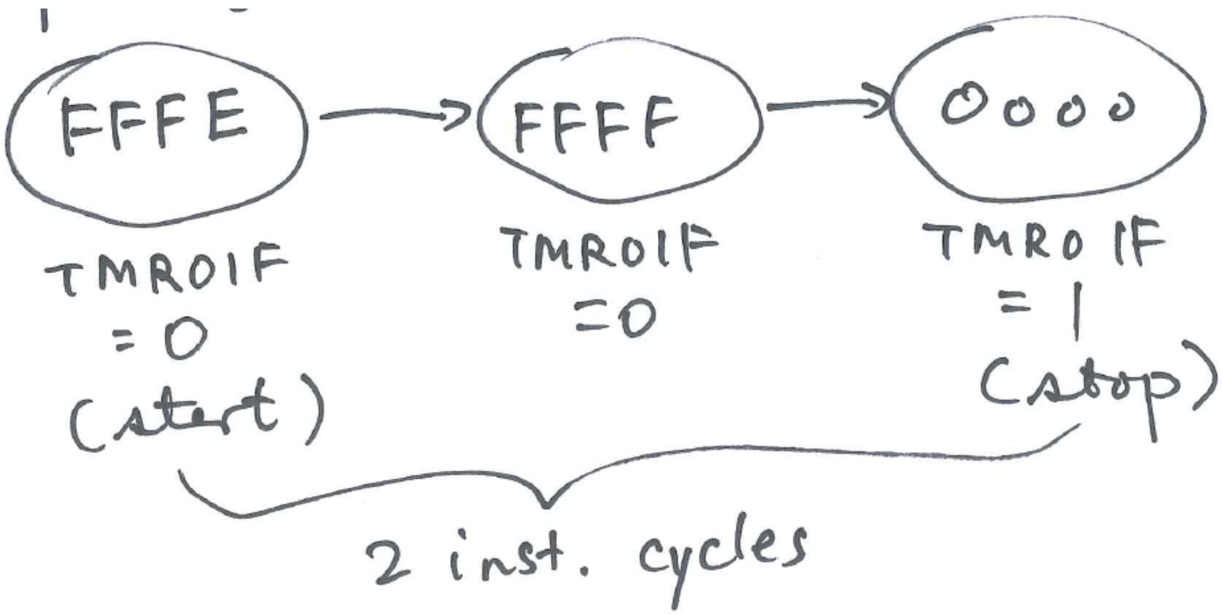
1. Set T08BIT = T0CON<6> = 0 (use 16-bit mode)
2. Load register TMR0H followed by TMR0L with initial count values
3. Start the timer: `bsf T0CON, TMR0ON`
4. Keep monitoring TMR0IF to see whether it is raised. Get out of loop if raised.
5. Stop the timer: `bcf T0CON, TMR0ON`
6. Clear TMR0IF for next round
7. Repeat from Step 2



Create delay using Timer 0 16-bit Mode

- TMR0 increments once each instruction cycle.
- Time delay obtained depends on the initial value.
- We stop when counter overflow, which is indicated by the TMR0IF flag.

<u>Simple Example</u>

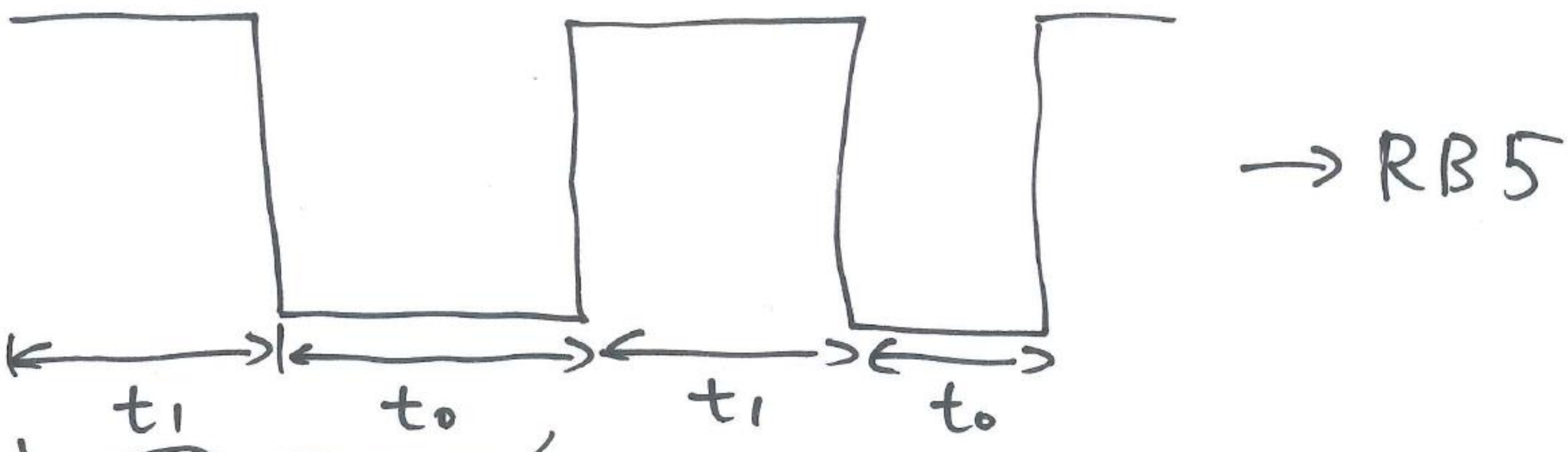


Example

Create a square wave of 50% duty cycle on RB5 bit.

```
      bcf     TRISB, 5          ; Set PB5 as output
      bsf     PORTB, 5          ; init. sq. wave to high (1)
      movlw   B '00001000'      ; 16-bit, int clk, no prescaler
      movwf  T0CON
Here  movlw  0xFF               ; [TMR0H] = 0xYY
      movwf  TMR0H
      movlw  0xF2               ; [TMR0L] = 0xXX
      movwf  TMR0L
      bcf    INTCON, TMR0IF     ; Clear timer interrupt flag
      bsf    T0CON, TMR0ON      ; start Timer 0
Again btfss  INTCON, TMR0IF
      bra    Again
      bcf    TOCON, TMR0ON
      btg    PORTB, 5
      bra    Here               ; load TMR0H and TMR0L again
```

Duty Cycle



whole period $= t_1 + t_0$

$$\text{Duty cycle} = \frac{t_1}{t_1 + t_0}$$

50% duty cycle $\longrightarrow t_1 = t_0$

Time Delay Calculation

- F_{osc} = clock frequency = 4MHz
- Instruction Cycle = 1μs
- Instruction Cycle required to reach FFFF = FFFF-FFF2 = 13 (in decimal)
- Need one more instruction cycle to rollover to 0000 where TMR0IF is triggered.
- Total time delay generated by Timer 0= 14 x 1μs = 14μs

Formula to Compute Time Delay

- In Hex, total time delay generated by Timer 0 = (FFFF – YYXX + 1) x 1μs

where YY and XX are the initial values of TMR0H and TMR0L respectively.

- In decimal, convert YYXX into its decimal equivalence NNNNN first and then use: (65536 – NNNNN) x 1μs

Use of Prescaler

- Longest delay that can be produced is 65.536 ms using a 16-bit timer. What if we want longer delay?
- Prescaler can be used to increase period of internal clock by a factor of 2^N, where N = 1, ..., 8 and specified by T0PS2:T0PS0 = T0CON<2:0>.
- e.g., When N = 6, the period of the clock is increased by a factor of 64 → 64μs. Max delay = 65536 x 64 μs = 4.194 s

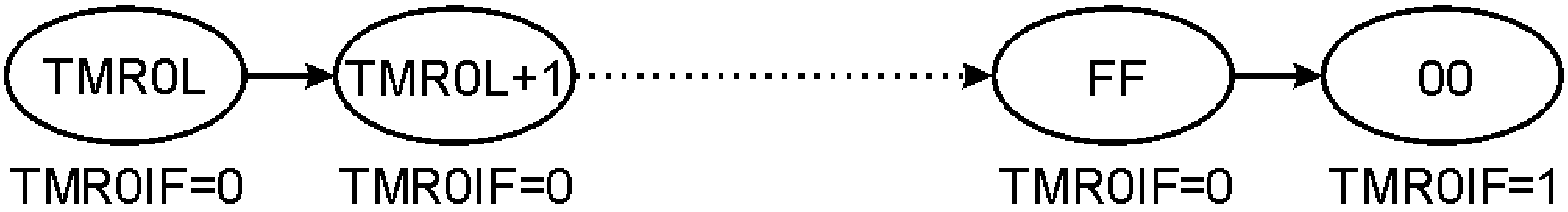
Examples

Assume F_{osc} = 4MHz and Instruction Cycle = 1MHz

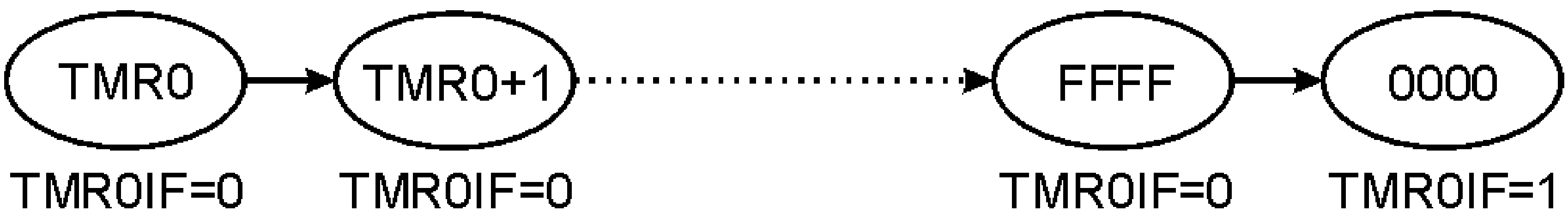
(FFFF - FC84 + 1) x 1μs = 892 μs

Initial TMR0	Prescaler	Time Delay (μs)
FC84	1 (no prescaler)	892
	2	1,784
	4	3,568
	8	7,136
	16	14,272
	32	28,544
	64	57,088
	128	114,176

Create delay using Timer 0 8-bit Mode



8-bit Mode



16-bit Mode

Timer 0 as counter

- If driven by an external clock (T0CS = 1), Timer 0 will increment, either on every rising (T0SE = 0) or falling (T0SE = 1) edge of pin RA4/T0CKI.

Example

Assuming clock pulses are fed into Pin T0CKI (RA4), use Timer 0 as a 8-bit counter of pulses and display the state of TMR0L count on PORTB

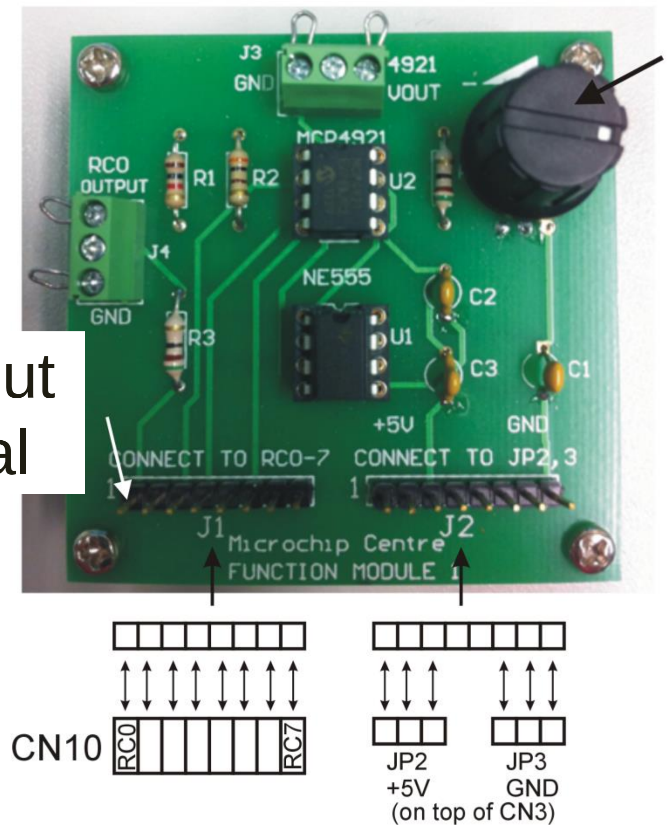
```
        bsf TRISA, RA4; PORTA.4 set as input
        clrf TRISB; PORTB set as output
        movlw 0x68; 01101000: Timer 0, 8-bit, ext clk, no prescale
        movwf T0CON
HERE   movlw 0x00
        movwf TMR0L
        bcf INTCON, TMR0IF
        bsf T0CON, TMR0ON
AGAIN movff TMR0L, PORTB
        btfss INTCON, TMR0IF
        bra AGAIN
        bcf T0CON, TMR0ON
        goto HERE
```

Timer 1

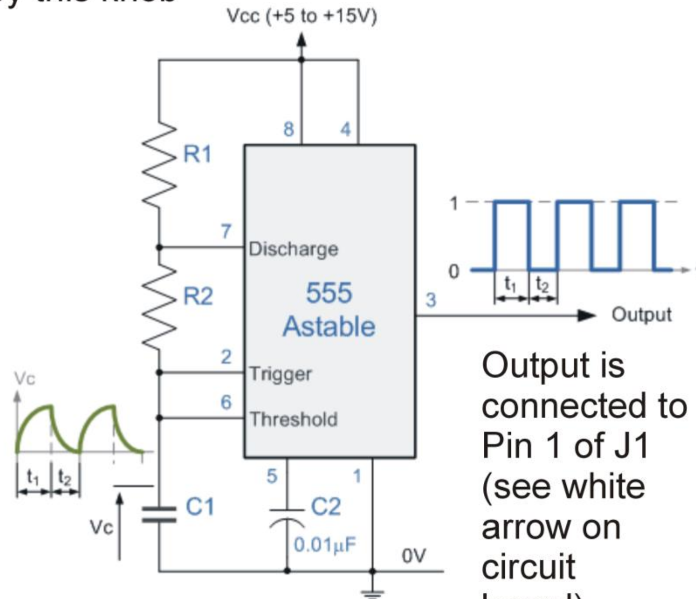
Timer 0	**Timer 1**
8-bit or 16-bit timer or counter.	16-bit timer or counter.
Supports prescaling factors of 1, 2, 4, 8, 16, 32, 64, 128, 256.	Supports prescaling factors of 1, 2, 4 and 8.
Timer 0 counts clock pulses fed into RA4	Timer 1 counts clock pulses fed into RC0
When counter rolls over, TMR0IF in INTCON SFR is raised	When counter rolls over, TMR1IF in PIR1 SFR is raised

Lab 4 Task 2: Frequency Counter

Output signal



R2 is controlled by this knob



Output is connected to Pin 1 of J1 (see white arrow on circuit board)

Variable Frequency Generator: Output frequency depends on the charging/discharging time of C1, which is controlled by varying the resistor value R2 through turning a knob on the circuit board

Lab 4 Task 2: Frequency Counter

Write a subroutine `MeasureFrequency` to perform the following operations:

- Set up Timer 0 to generate a 1-second delay
- Use Timer 1 as a counter to count the rising edges of an unknown signal (fed to RC0) within the second
- Start Timer 0 and Timer 1
- Wait for IF flag for Timer 0 to raise
- Stop Timer 0 and Timer 1
- TMR1H:TMR1L would then be the frequency of the signal.

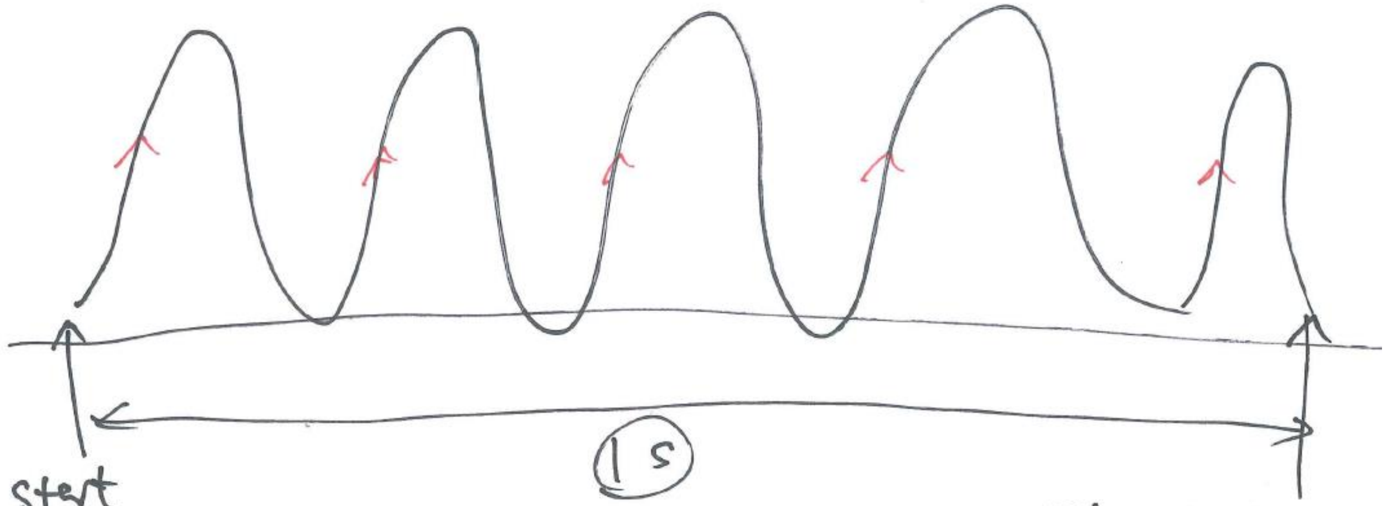
Overall structure of Lab 4

The algorithm consists of two components:

1. <u>Generate a time delay of 1s using Timer 0</u>
   - Lab 4 Task 1
   - Prescaler = 16
   - Expected total time delay = (FFFF-YYXX+1) x 16 x 1µs = 1s
   - You will determine YYXX in prelab, but you know that (FFFF-YYXX+1) = 62500 (i.e., Timer 0 increments 62500 times before overflow)
   - Does the program generate a delay of <u>exactly</u> 1s?

2. <u>Count total number of rising edges using Timer 1</u>
   - The example code gives all information of how it can be done.

Overall structure of Lab 4



start Timer0 Timer1

TMR1 = # of rising edges in 1s

= # of period in 1s

= frequency of signal.

Timer0 is expected to raise IF after 1s has been elapsed. Stop timer1 counter

1,000,000 inst. cycle

= 62500 x 16

Lab 4 Task 1

You are given the following code fragment that uses TMR0 to count 1 second. Follow the instructions below to modify the code so that the LED turns on for exactly 1 second.

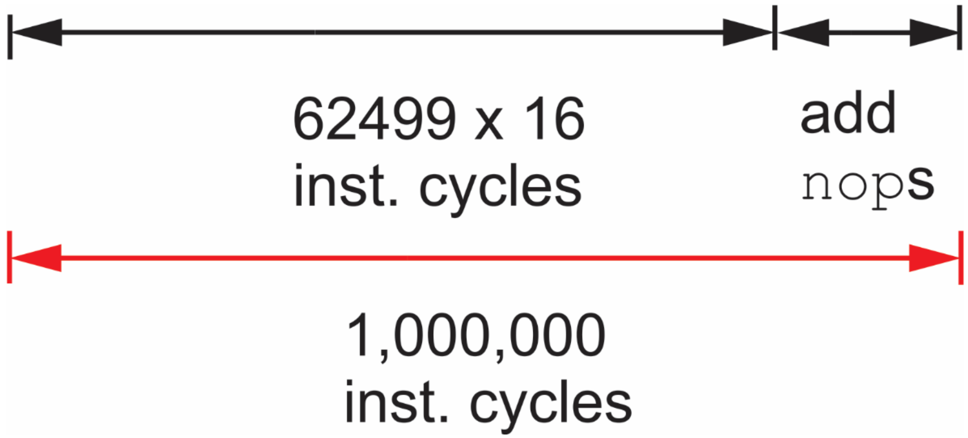
```
Main:           clrf   TRISD              ;set PortD output
                clrf   PORTD

                movlw  0x03               ;TMR0, 16-bit, Int clk, 1:16 prescale
                movwf  T0CON
                movlw  0xYY   ;Students will determine initial YYXX to put in TMR0H: TMR0L
                movwf  TMR0H
                movlw  0xXX
                movwf  TMR0L

                bcf    INTCON, TMR0IF     ;
                bsf    PORTD, RD0         ;turn LED on

                bcf    INTCON, TMR0IF ; Set Breakpoint 1 here right after LED was turned on
                bsf    T0CON,  TMR0ON     ;start TMR0
WaitT0done:     btfss  INTCON, TMR0IF
                bra    WaitT0done

                bcf    PORTD, RD0         ;turn LED off.
                bra    $              ;set Breakpoint 2 here right after LED was turned off
                                      ; jump to current PC, loop forever;
```

Lab 4 Task 1

Code	Total Time Delay	
	Expected	Actual
`bsf PORTD, RD0`		
`bcf INTCON, TMR0IF`	1,000,000 inst. cycles = 1 s	1
`bsf T0CON, TMR0ON`		1
`WaitT0done: btfss INTCON, TMR0IF`		999999 + 2 • 333333 repetitions x 3 inst. cycle/ reps → TMR0 overflows → TMR0IF = 1 • btfss skips (additional 2 inst. cycles
`bra WaitT0done`		
`bcf PORTD, RD0`		1
Total time LED has turned on (inst. cycles)	1,000,000	1,000,004

Lab 4 Task 1

- Need exact time delay for accurate frequency measurement
- Solution
  - Determine the initial value YYXX again so that Timer 0 increments 62499 times (instead of 62500) before overflow.
  - Time delay generated before TMR0IF is raised = 62499 x 16 inst. cycles
  - Add a few `nop` to fill up the time.
  - Objective: Decide how many `nop`s are needed to generate a time delay of exactly 1s.

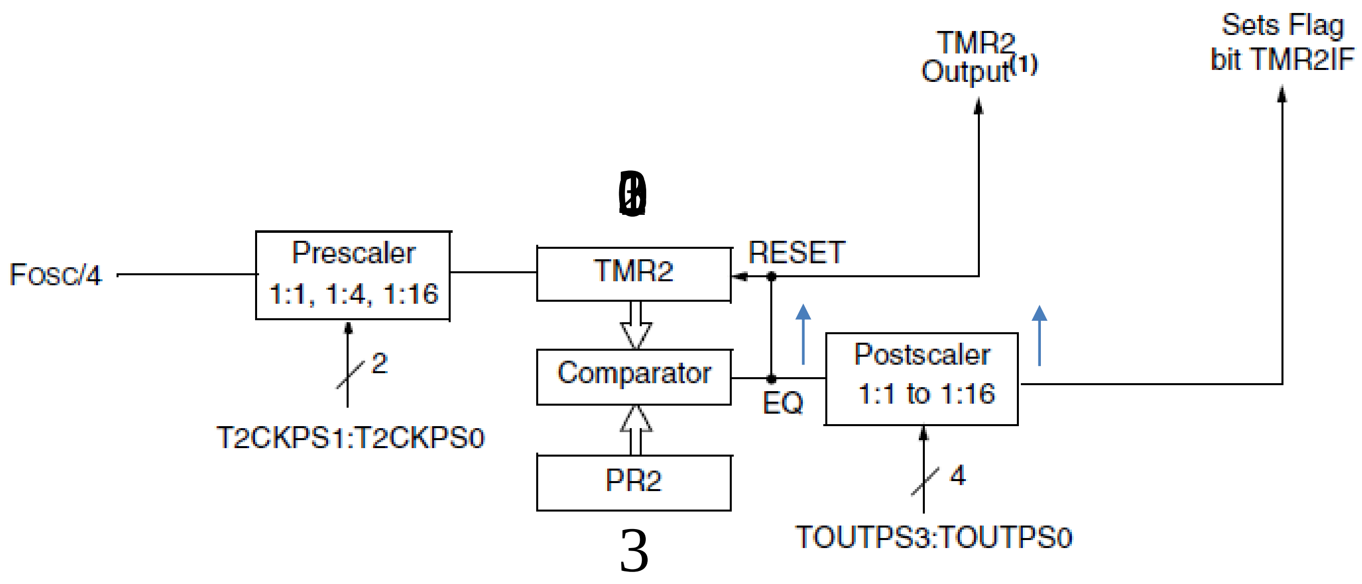


Overall structure of Lab 4

- <u>Validation of your measurement</u>
  - Use debugger mode of PICKit 3. Set a breakpoint at the location where you exit the subroutine `MeasureFrequency`.
  - TMR1H and TMR1L store your frequency measurement.

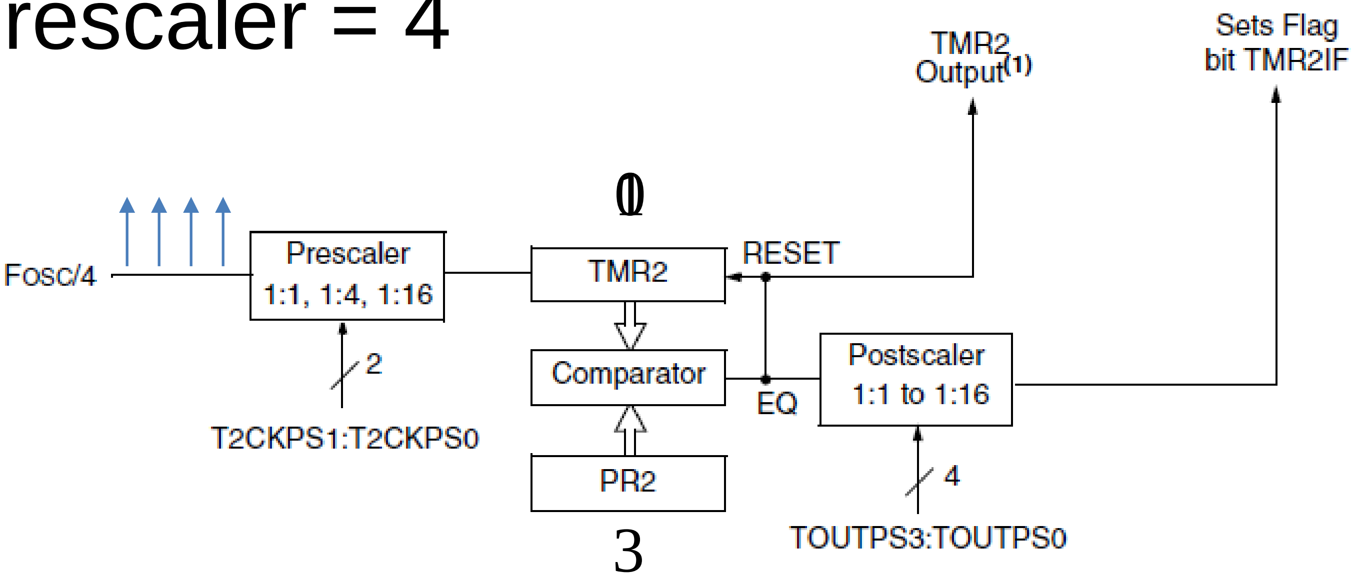
Timer 2

- Can only be used as a timer to create time delay – No external clock source can fed in.
- Prescaler and Postscaler to lengthen delays.



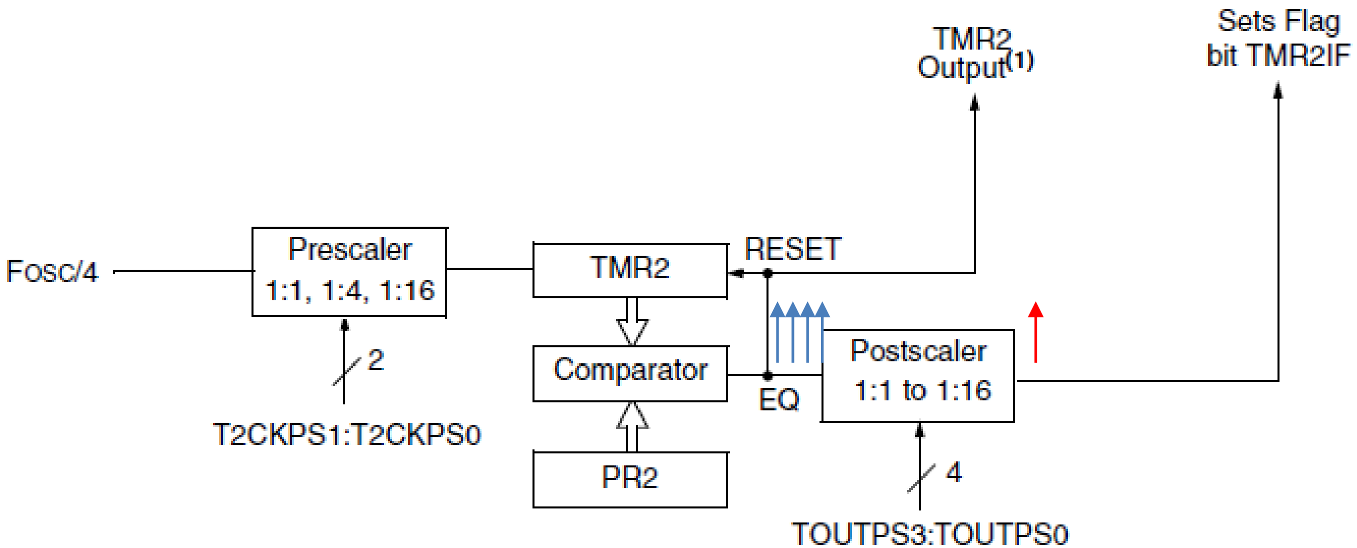
Timer 2

- Prescaler = 4



Timer 2

- Postscaler = 4



T2CON: Timer 2 Control Register

REGISTER 12-1: T2CON: TIMER2 CONTROL REGISTER

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
bit 7							bit 0

bit 7 **Unimplemented:** Read as '0'

bit 6-3 **TOUTPS3:TOUTPS0**: Timer2 Output Postscale Select bits

`0000` = 1:1 Postscale
`0001` = 1:2 Postscale
•
•
•
`1111` = 1:16 Postscale

bit 2 **TMR2ON**: Timer2 On bit

`1` = Timer2 is on
`0` = Timer2 is off

bit 1-0 **T2CKPS1:T2CKPS0**: Timer2 Clock Prescale Select bits

`00` = Prescaler is 1
`01` = Prescaler is 4
`1x` = Prescaler is 16

TMR2 and PR2 SFR

- TMR2: 8-bit register of Timer 2
- PR2 (Period Register) is set to be a fixed value.
- The input period can be lengthened by selecting the prescaling factor.
- When TMR2 == PR2, EQ signal will reset TMR2 to 0.
- The output of the comparator is divided by postscaling factor.
  - i.e., If postscaler = 1:16, the EQ signal will need to be set 16 times before TMR2IF is set.

Example 1: Timer 2

- Write a program to turn on pin PORTB.4 when TMR2 reaches the value of 100 (decimal).

```
        bcf    TRISB, 4; make PORTB.4 as output
        bcf    PORTB, 4; turn off PORTB.4
        movlw  0x00
        movwf  T2CON; no prescale and postscale
        movlw  0x00
        movwf  TMR2
        movlw  D‘100’
        movwf  PR2
        bcf    PIR1, TMR2IF
        bsf    T2CON, TMR2ON
Again:  btfss  PIR1, TMR2IF
        bra     Again
        bsf    PORTB, 4
        bcf    T2CON, TMR2ON
Here:   bra    Here
```

Example 2: Timer 2

- Use the prescaler and postscaler to create the longest possible delay by Timer 2.
- Total delay = (255+1) x 16 x 16 x 1μs = 65.536ms

```
            bcf    TRISB, 4
            bcf    PORTB, 4
            movlw B'01111011'
            movwf T2CON; prescale = 1:16, postscale = 1:16
            movlw 0x00
            movwf TMR2
            movlw D'255'
            movwf PR2
            bcf    PIR1, TMR2IF
            bsf    T2CON, TMR2ON
Again:      btfss PIR1, TMR2IF
            bra    Again
            bsf    PORTB, 4
            bcf    T2CON, TMR2ON
Here:       bra    Here
```

Timer 0, Timer 1 and Timer 2

Timer 0	Timer 1	Timer 2
8-bit or 16-bit timer or counter.	16-bit timer or counter.	8-bit timer. Cannot be used as counter
Supports prescaling factors of 1, 2, 4, 8, 16, 32, 64, 128, 256.	Supports prescaling factors of 1, 2, 4 and 8.	Supports prescaling and postscaling factors.
Timer 0 counts clock pulses fed into PORTA.4	Timer 1 counts clock pulses fed into PORTC.0	Cannot be used as counter
When counter rolls over, TMR0IF in INTCON SFR is raised	When counter rolls over, TMR1IF in PIR1 SFR is raised	When counter rolls over, TMR2IF in PIR1 SFR is raised

Time Delay Equation

- Time delay are expressed in instruction cycles.
- Timer 0/1 16-bit mode
  – Time Delay = (FFFF – YYXX + 1) x Prescaler
  – where YY and XX are the initial values of TMRxH and TMRxL, respectively. x = 0, 1.
- Timer 0 8-bit mode
  – Time Delay = (FF – XX + 1) x Prescaler
- Timer 2
  – Time Delay = (PR2+1) x Prescaler x Postscaler

You should be able to ...

- List the PIC18 timers and their associated registers.
- Describe various modes of the PIC18 timer.
- Program the PIC18 timers to generate time delays