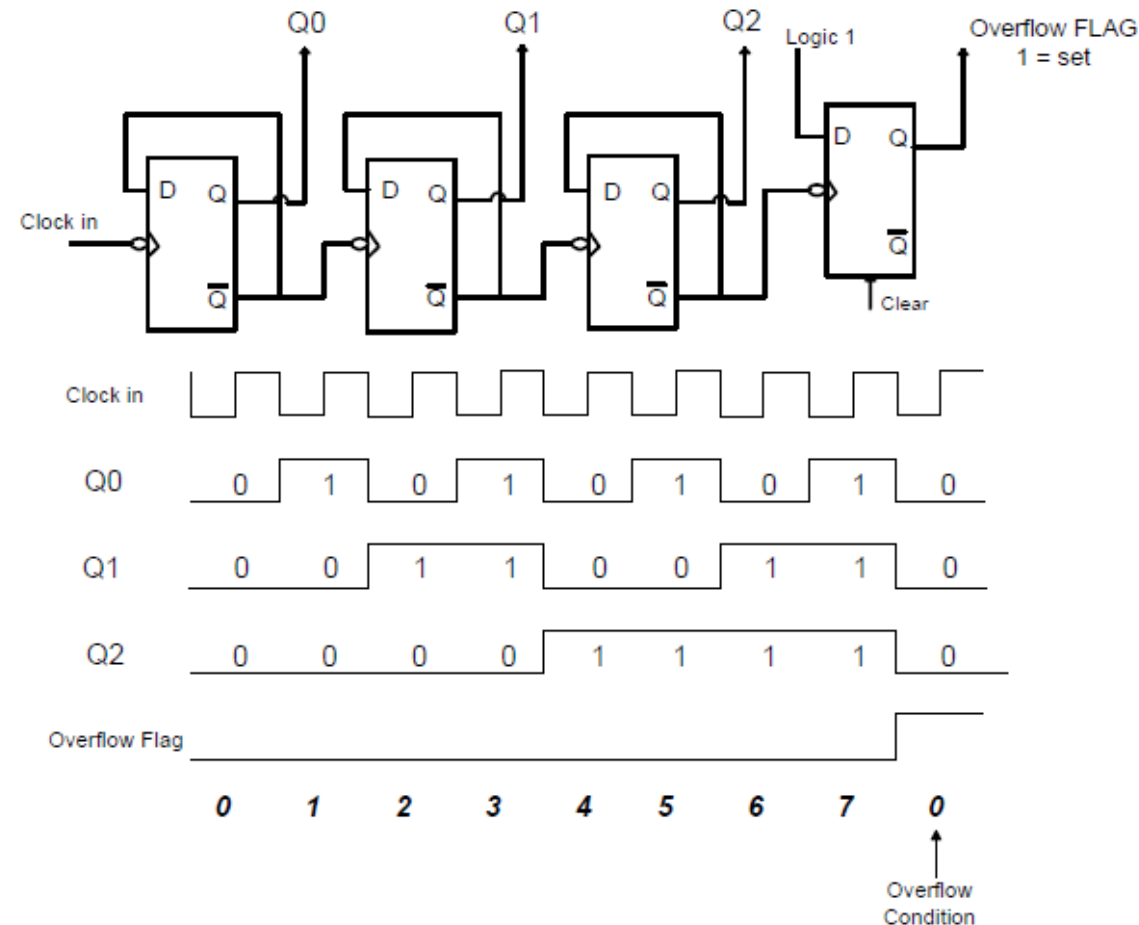


## 4.4 timer

- PIC18 has two to five timers (Timer0, Timer1, etc.) – depending on family member
- timer can be used to
  - generate time delay
  - count event
  - record event arrival time
  - generate waveform with certain duty cycle and frequency
  - ....

A timer/counter can be configured in various modes, typically based on 8-bit or 16-bit operation. It is used to count the number of input pulses.



## Timer

- Set the initial value of registers (by software)
- Start the timer (by software) and then the timer counts up
- Input from internal system clock (machine cycle)
- When the timer: FFFF --> 0, the timer sets a bit to denote time out

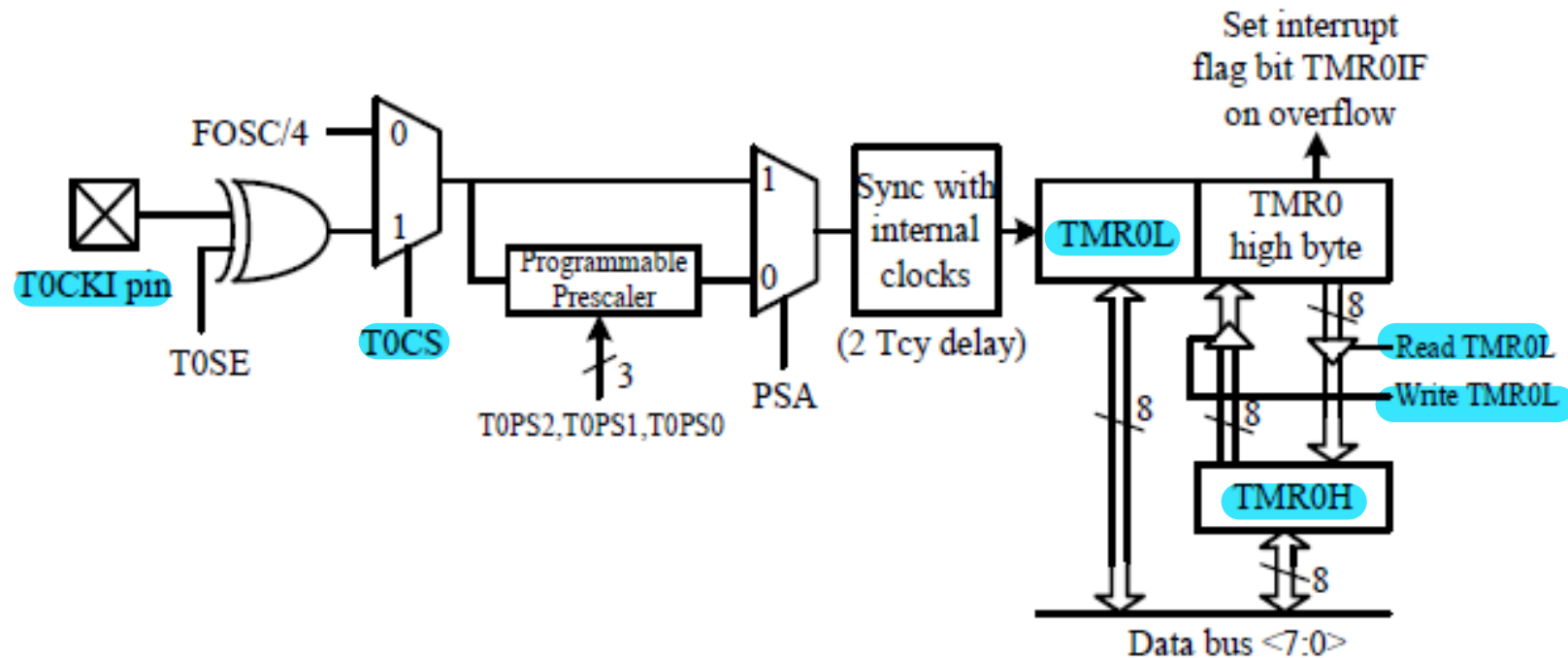
## Counter

- Set the initial value of registers (by software)
- Start the timer (by software) and then the timer counts up
- Input from external signal
- When the timer: FFFF --> 0, the timer sets a bit to denote time out

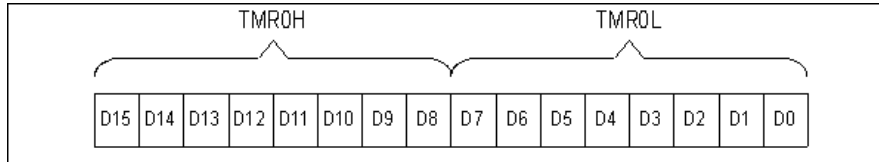
- Every timer needs a clock pulse to tick
- Clock source can be
  - **Internal** → 1/4th of the frequency of the crystal oscillator on OSC1 and OSC2 pins ( $F_{osc}/4$ ) is fed into timer
  - **External** pulses are fed through one of the PIC18's pins → Counter
- Timers are 16-bit wide ( $TMR_{xL}$  &  $TMR_{xH}$ )
  - Each timer has **TCON** (Timer CONtrol) register

### 4.4.1 Timer0

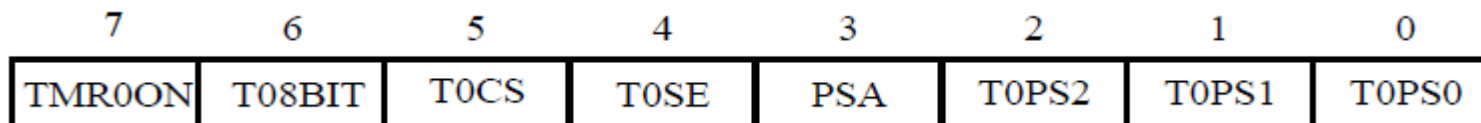
- can select the internal instruction cycle clock ( $F_{osc}/4$ ) – time delay
- can use the RA4/T0CKI signal as the clock signal – count event
- user can choose to divide the clock signal by a prescaler before connecting it to the clock input to Timer0



- can be configured as an 8-bit or 16-bit timer or counter



- **TMR0H** acts as a **buffer** between TMR0 high byte and data bus
- you can move value in WREG into TMR0 – **load TMR0H** first, then **load TMR0L**
- when microcontroller reads TMR0L, contents of TMR0 high byte is transferred to TMR0H
- **T0CON** register controls the operation of Timer0



	7	6	5	4	3	2	1	0
value after reset	TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0
	1	1	1	1	1	1	1	1

**TMR0ON:** *Timer0 on/off control bit*

0 = stops Timer0

1 = Enables Timer0

**T08BIT:** *Timer0 8-bit/16-bit control bit*

0 = Timer0 is configured as a 16-bit timer

1 = Timer0 is configured as an 8-bit timer

**T0CS:** *Timer0 clock source select*

0 = Instruction cycle clock

1 = Transition on T0CKI pin

**T0SE:** *Timer0 source edge select bit*

0 = Increment on falling edge transition on T0CKI pin

1 = Increment on rising edge transition on T0CKI pin

**PSA:** *Timer0 prescaler assignment bit*

0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

1 = Timer0 prescaler is not assigned. Timer0 clock input bypasses prescaler.

**T0PS2:T0PS0:** *Timer0 prescaler select bits*

000 = 1:2 prescaler value

001 = 1:4 prescaler value

010 = 1:8 prescaler value

011 = 1:16 prescaler value

100 = 1:32 prescaler value

101 = 1:64 prescaler value

110 = 1:128 prescaler value

111 = 1:256 prescaler value

# Example

Find the value for T0CON if we want to program Timer0 in 16-bit mode, no prescaler. Use PIC18 Fosc/4 crystal oscillator for the clock source, increment on positive-edge.

7	6	5	4	3	2	1	0
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0

T0CON = 0000 1000



# Example

Find the timer's clock frequency and its period for various PIC18-based systems, with the following crystal frequencies. Assume that no prescaler is used.

(1) 10 MHz      (2) 16 MHz      (3) 4 MHz

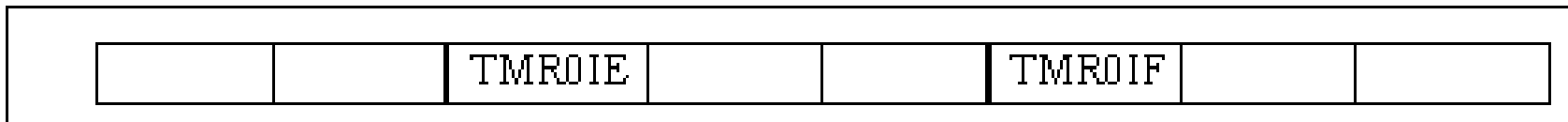
(1)  $\frac{1}{4} \times 10 \text{ MHz} = 2.5 \text{ MHz}$  and  $T = 0.4 \text{ } \mu\text{s}$

(2)  $\frac{1}{4} \times 16 \text{ MHz} = 4 \text{ MHz}$  and  $T = 0.25 \text{ } \mu\text{s}$

(3)  $\frac{1}{4} \times 4 \text{ MHz} = 1 \text{ MHz}$  and  $T = 1 \text{ } \mu\text{s}$

## 16-bit timer programming

1. Values from 0000 to FFFFH.
2. After loading TMR0H and TMR0L, the timer must be started.
3. Count up, till it reaches FFFFH, then it rolls over to 0000 and sets TMR0IF bit to 1.
4. Then TMR0H and TMR0L must be reloaded with the original value and TMR0IF bit must be reset to 0.



INTCON register with Timer0 Interrupt Enable and Interrupt Flag

## generate time delay

1. Load the value into the T0CON register
2. Load register TMR0H followed by register TMR0L with initial values
3. Start the timer with instruction  
**BSF T0CON, TMR0ON**
4. Keep monitoring the timer flag (TMR0IF) to see if it is raised
5. Stop the timer with instruction  
**BCF T0CON, TMR0ON**
6. Clear the TMR0IF flag
7. Go back to step 2

# Example

Generate a square wave of 50% duty cycle on the PORTB.5 pin.

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

1. T0CON is loaded
2. FFF2H is loaded into TMR0H-TMR0L
3. The Timer0 interrupt flag is cleared
4. PORTB.5 is toggled
5. Timer0 is started
6. Timer0 counts up from FFF2H until it reaches FFFFH, one more clock rolls it to 0, then TMR0IF = 1, falling through



7. Timer0 is stopped
8. The process is repeated

# Example

In the previous example, calculate the amount of time delay generated by the timer. Assume that the clock frequency is 10MHz.

$$F_{osc}/4 = 2.5 \text{ MHz}$$

$$\text{clock cycle time} = 0.4 \mu\text{s}$$

$$\text{number of counts} = \text{FFFFH} - \text{FFF2H} + 1 = 0EH = 14$$

$$\text{half the pulse} = 14 \times 0.4 \mu\text{s} = 5.6 \mu\text{s}$$

$$\text{one cycle} = 11.2 \mu\text{s}$$

# Example

Calculate the frequency of the square wave generated on pin PORTB.5.

Assume that the clock frequency is 10MHz.

	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	2
	BTG	PORTB, 5	1
	BRA	HERE	2
DELAY:	BSF	T0CON, TMR0ON	1
AGAIN:	BTFSS	INTCON, TMR0IF	1 or 2
	BRA	AGAIN	2
	BCF	T0CON, TMR0ON	1
	BCF	INTCON, TMR0IF	1
	RETURN		2
			---
			17 or 18

$$T \cong 2 \times (48 + 17) \times 0.4 \mu s = 52 \mu s$$

$$F = 19.23 \text{ KHz}$$

# Example

Calculate the frequency of the square wave generated on pin PORTB 5.

Assume that the clock frequency is 10MHz. Do not include the overhead due to instructions in the loop.

HERE:	BCF TRISB,5 MOVLW 0x08 MOVWF T0CON  MOVLW 0x76 MOVWF TMR0H MOVLW 0x34 MOVWF TMR0L BCF INTCON, TMR0IF CALL DELAY BTG PORTB,5 BRA HERE	FFFFH - 7634H + 1 = 89CCH = 35,276 Delay 35,276 x 0.4 μs = 14.11 ms Period = 28.22 ms Frequency = 35.44 Hz
DELAY:	BSF T0CON, TMR0ON	
AGAIN:	BTFSS INTCON, TMR0IF BRA AGAIN BCF T0CON, TMR0ON RETURN	



# Example

Assuming that clock = 10 MHz, write a program to generate a square wave with period of 10 ms on pin PORTB.3.

$T = 10 \text{ ms} \Rightarrow \text{Delay} = 5 \text{ ms}$ .  $5 \text{ ms} / 0.4 \mu\text{s} = 12,500$ .  $65,536 - 12,500 = 53,036 = \text{CF2CH}$ .  
 $\text{TMR0H} = \text{CFH}$ ,  $\text{TMR0L} = 2\text{CH}$

HERE	BCF	TRISB,3	Modify the program to get the lowest frequency.
	MOVLW	0x08	
	MOVWF	T0CON	
	MOVLW	0xCF	
	MOVWF	TMR0H	
	MOVLW	0x2C	
DELAY	MOVWF	TMR0L	If we set $\text{TMR0} = 0000\text{H}$ , the delay generated is $65,536 \times 0.4 \mu\text{s} = 26.214 \text{ ms}$ .
	BCF	INTCON, TMR0IF	
	CALL	DELAY	
	BTG	PORTB,3	
	BRA	HERE	
	BSF	T0CON, TMR0ON	
AGAIN	BTFSS	INTCON, TMR0IF	The smallest frequency = 19.07 Hz.
	BRA	AGAIN	
	BCF	T0CON, TMR0ON	
	RETURN		

# Prescaler and generating larger delay

The size of time delay depends on  
crystal frequency  
timer's 16-bit register

The largest time delay happens when  $TMR0L = TMR0H = 0$

Prescaler option is used to duplicate the delay by dividing the clock by a factor of 2, 4, 8, 16, 32, 64, 128, 256

If  $T0CON = 0000\ 0101$ , then  $T = 4 \cdot 64 / F_{osc}$

7	6	5	4	3	2	1	0
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0

*PSA: Timer0 prescaler assignment bit*

0 = Timer0 prescaler is assigned. Timer0 clock input comes from prescaler output.

1 = Timer0 prescaler is not assigned. Timer0 clock input bypasses prescaler.

*T0PS2:T0PS0: Timer0 prescaler select bits*

000 = 1:2 prescaler value

001 = 1:4 prescaler value

010 = 1:8 prescaler value

011 = 1:16 prescaler value

100 = 1:32 prescaler value

101 = 1:64 prescaler value

110 = 1:128 prescaler value

111 = 1:256 prescaler value

# Example

Examine the following program and find the time delay in second and frequency of the wave generated. Assume that XTAL = 10 MHz.

<div>HERE:</div> <div>BCF TRISB, 2 MOVLW 0x05 MOVWF T0CON  MOVLW 0x01 MOVWF TMR0H MOVLW 0x08 MOVWF TMR0L BCF INTCON, TMR0IF CALL DELAY BTG PORTB, 2 BRA HERE</div> <div>DELAY:</div> <div>AGAIN:</div> <div>BSF T0CON, TMR0ON  BTFSS INTCON, TMR0IF BRA AGAIN BCF T0CON, TMR0ON RETURN</div>	<div>Prescaler = 64.</div> <div>TMR0 = 0108H = 264 65,536 – 264 = 65,272</div> <div>65,272 x 64 x 0.4 ms = 1.671 s</div> <div>frequency = 0.299 Hz</div> <div>If prescaler 256 is chosen, the prescaler clock period to the timer is 0.4 ms x 256 = 0.1024 ms.</div> <div>The largest delay is 6.7 seconds.</div>
--	---

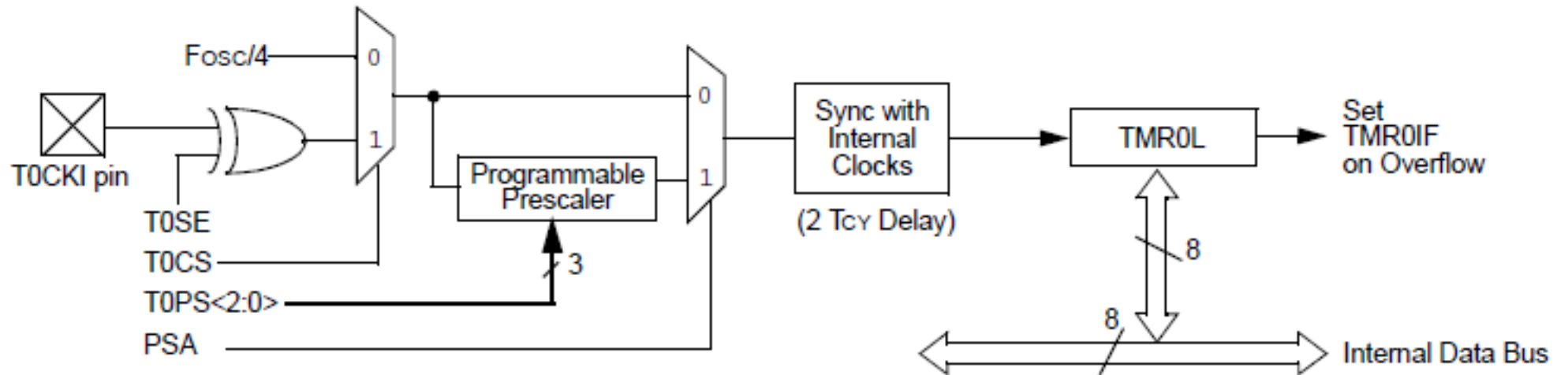
# Example

Write a program to generate a square wave of 50Hz on pin PORTB.7. Use Timer0, 16-bit mode with prescaler = 128. Assume that XTAL = 10 MHz.

HERE:	BCF	TRISB,7	(1) $F = 50 \text{ Hz}$ , $T = 20 \text{ ms}$ (2) Delay = 10 ms (3) $10 \text{ ms} / 0.4 \mu\text{s} / 128 = 195$ (4) $65,536 - 195 = 65,341 = \text{FF3DH}$
	MOVLW	0x06	
	MOVWF	T0CON	
	MOVLW	0xFF	
	MOVWF	TMR0H	
	MOVLW	0x3D	
	MOVWF	TMR0L	
	BCF	INTCON, TMR0IF	
	CALL	DELAY	
	BTG	PORTB, 7	
DELAY:	BRA	HERE	
	BSF	T0CON, TMR0ON	
	AGAIN:		
AGAIN:	BTFSS	INTCON, TMR0IF	
	BRA	AGAIN	
	BCF	T0CON, TMR0ON	
	RETURN		

# Timer0 8-bit mode

1. Load the T0CON value
2. Load TMR0L only
3. Start timer
4. Keep monitor the timer flag (TMR0IF)



# Example

Assume that XTAL=10 MHz, find (a) the frequency of the square wave generated on PORTB.0 in the following program, and (b) the smallest frequency achievable in this program, and the TMR0L value to do that.

			255-5+1
	BCF	TRISB, 0	a) Delay = 251 x 0.4 μs = 100.4 ms
	MOVLW	0x48	Cycle = 200.8 ms
	MOVWF	T0CON	Frequency = 4.98 KHz
	BCF	INTCON, TMR0IF	
HERE:	MOVLW	0x5	b) To get the smallest frequency, we set
	MOVWF	TMR0L	TMR0L = 00
	CALL	DELAY	Cycle = 256 x 0.4 μs x 2 = 204.8 ms
	BTG	PORTB, 0	Frequency = 4.88 KHz
	BRA	HERE	
DELAY:	BSF	T0CON, TMR0ON	
AGAIN:	BTFSS	INTCON, TMR0IF	
	BRA	AGAIN	
	BCF	T0CON, TMR0ON	
	BCF	INTCON, TMR0IF	
	RETURN		

# Example

Assume XTAL = 10MHz. Find the clock cycle time if Timer0 prescaler is set to 256. Determine the largest time delay for this prescaler option.

$$F_{osc}/4 = 2.5 \text{ MHz}$$

$$\text{clock cycle time} = 256/2.5 \text{ MHz} = 102.4 \mu\text{s}$$

$$\text{Set TMR0L} = 0$$

$$\text{Time delay} = 256 \times 102.4 \mu\text{s} = 26.21 \text{ ms}$$

# Load negative value

MOVLW -D'100'

MOVWF TMR0L

is identical to move 9CH into TMR0L. Then  $FFH - 9CH + 1 = 100_{10}$

Assume 10 MHz

```
HERE:    BCF      TRISB,3
          BCF      INTCON, TMR0IF
          MOVLW    0x48
          MOVWF    T0CON
          MOVLW    -D'150'
          MOVWF    TMR0L
          BSF      PORTB, 3
          CALL     DELAY
          MOVWF    TMR0L
          CALL     DELAY
          BCF      PORTB, 3
          MOVWF    TMR0L
          CALL     DELAY
          BRA      HERE
```

```
DELAY:   BSF      T0CON, TMR0ON
AGAIN:   BTFSS    INTCON, TMR0IF
          BRA      AGAIN
          BCF      T0CON, TMR0ON
          BCF      INTCON, TMR0IF
          RETURN
```

Delay =  $150 \times 0.4 \mu s = 60 \mu s$

High portion of the pulse =  $120 \mu s$

Low portion of the pulse =  $60 \mu s$

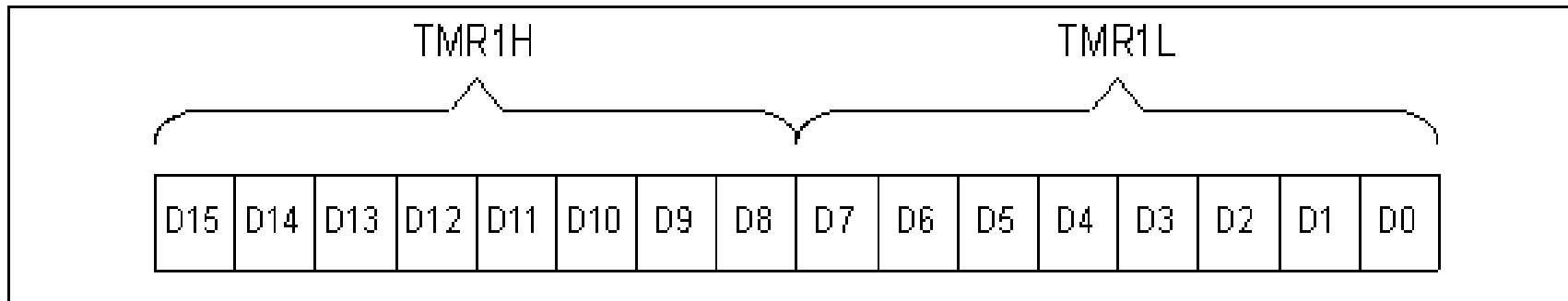
Frequency = 5.56 KHz

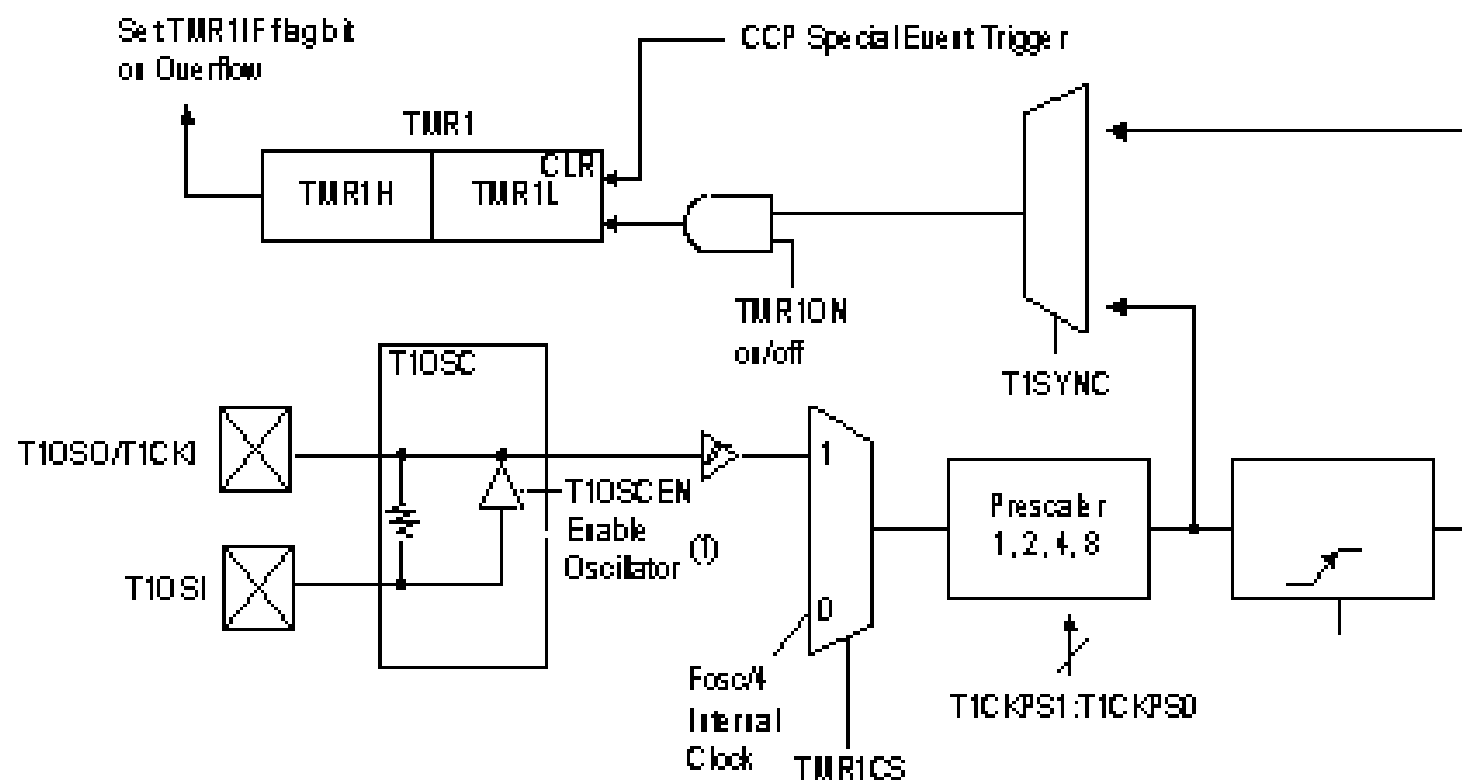
$(120e-6 + 60e-6)^{-1}$



## 4.4.2 Timer1

- can be programmed in 16-bit mode only
- it has 2 bytes named as **TMR1L** and **TMR1H**
- it also has **T1CON** and **TMR1IF**
- TMR1IF is in PIR1 register
- TMR1IF goes HIGH when TMR1H:TMR1L overflows (**FFFFH** → **0000H**)
- prescaler options: 1, 2, 4, 8





RD16	...	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON
------	-----	---------	---------	---------	--------	--------	--------

RD16	D7	16-bit read/write enable bit 1 = Timer1 16-bit is accessible in one 16-bit operation. 0 = Timer1 16-bit is accessible in two 8-bit operations.
	D6	Not used
T1CKPS2: T1CKPS0	D5 D4	Timer1 prescaler selector 0 0 = 1:1 Prescale value 0 1 = 1:2 Prescale value 1 0 = 1:4 Prescale value 1 1 = 1:8 Prescale value
T1OSCEN	D3	Timer1 oscillator enable bit 1 = Timer1 oscillator is enabled. 0 = Timer1 oscillator is shutoff
T1SYNC	D2	Timer1 synchronization (used only when TMR1CS = 1 for counter mode to synchronize external clock input) If TMR1CS = 0 this bit is not used.
TMR1CS	D1	Timer1 clock source select bit 1 = External clock from pin RC0/T1CKI 0 = Internal clock (Fosc/4 from XTAL)
TMR1ON	D0	Timer1 ON and OFF control bit 1 = Enable (start) Timer1 0 = Stop Timer1

# PIR1 (Interrupt Control Register 1)

							TMR1IF
--	--	--	--	--	--	--	--------

**TMR1IF**      D1      Timer1 Interrupt overflow flag bit  
0 = Timer1 did not overflow  
1 = Timer1 has overflowed (FFFF to 0000).

**The importance of TMR1IF:** When TMR1H:TMR1L overflows from FFFF to 0000, this flag is raised. We monitor this flag bit before we reload the TMR1H:TMR1L registers.

# Example

Write a program to generate a square wave of 50Hz on pin PORTB.5. Use Timer1 in 16-bit mode with the maximum prescaler. Assume that the clock frequency is 10 MHz.

HERE:	BCF	TRISB,5	(1) $F = 50 \text{ Hz}$ , $T = 20 \text{ ms}$ (2) Delay = 10 ms (3) $10 \text{ ms} / 0.4 \mu\text{s} / 8 = 3125 = \text{C35H}$ (4) $\text{FFFF} - \text{C34H} = \text{F3CBH}$
	MOVLW	0x30	
	MOVWF	T1CON	
	MOVLW	0xF3	
	MOVWF	TMR1H	
	MOVLW	0xCB	
	MOVWF	TMR1L	
	BCF	PIR1, TMR1IF	
	CALL	DELAY	
	BTG	PORTB, 5	
DELAY:	BRA	HERE	
	BSF	T1CON, TMR1ON	
AGAIN:	BTFSS	PIR1, TMR1IF	
	BRA	AGAIN	
	BCF	T1CON, TMR1ON	
	RETURN		

### 4.4.3 counter

- use timer to count events happening outside PIC18
  - increments the TMR0H and TMR0L registers
- T0CS in T0CON register decides the clock source
  - If T0CS = 1, the timer is used as a counter
    - TMR0 count up (NOT count down)
  - count up as pulses are fed from pin RA4/T0CKI
  - What does T0CON = 0110 1000 mean?

68h

7	6	5	4	3	2	1	0
TMR0ON	T08BIT	T0CS	T0SE	PSA	T0PS2	T0PS1	T0PS0

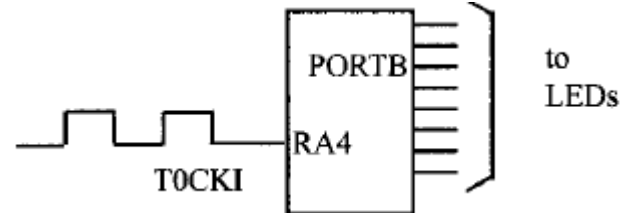
# Example

Assuming that a signal is fed into pin T0CKI, write a program for counter 0 in 8-bit mode to count the pulses and display the state of the TMR0L count on PORTB.

```
BSF      TRISA, RA4      ; PORTA.4 as external input for timer0
CLRF     TRISB
MOVLW    0x68            ; control word for timer0, 8 bit and counter model
MOVWF    T0CON

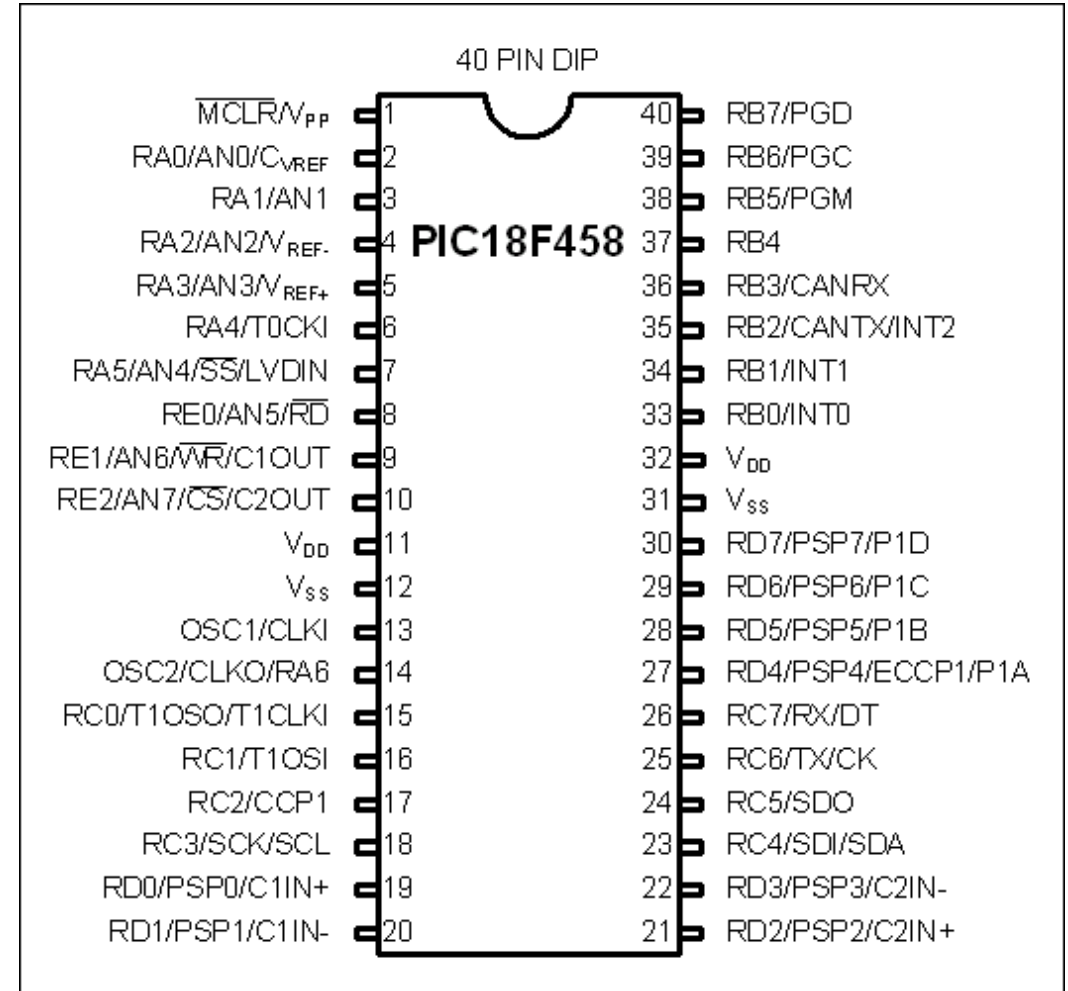
HERE:
MOVLW    0x0             ; TMR0L=0
MOVWF    TMR0L
BCF      INTCON, TMR0IF
BSF      T0CON, TMR0ON

AGAIN:
MOVFF    TMR0L, PORTB
BTFSS    INTCON, TMR0IF
BRA      AGAIN
BCF      T0CON, TMR0ON
GOTO     HERE
```



# Using external crystal for Timer1 clock

- for Timer1, when **TMR1CS = 1**, clock pulse coming from pin **RC0/T1CKI** makes the counter count up
- alternatively, set **T1OSCEN = 1**, feed clock pulse from crystal to **T1OSI** and **T1OSO** pins





RD16	...	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON
------	-----	---------	---------	---------	--------	--------	--------

<b>RD16</b>	D7	16-bit read/write enable bit 1 = Timer1 16-bit is accessible in one 16-bit operation. 0 = Timer1 16-bit is accessible in two 8-bit operations.
	D6	Not used
<b>T1CKPS2:T1CKPS0</b>	D5 D4	Timer1 prescaler selector 0 0 = 1:1      Prescale value 0 1 = 1:2      Prescale value 1 0 = 1:4      Prescale value 1 1 = 1:8      Prescale value
<b>T1OSCEN</b>	D3	Timer1 oscillator enable bit 1 = Timer1 oscillator is enabled. 0 = Timer1 oscillator is shutoff
<b>T1SYNC</b>	D2	Timer1 synchronization (used only when TMR1CS = 1 for counter mode to synchronize external clock input) If TMR1CS = 0 this bit is not used.
<b>TMR1CS</b>	D1	Timer1 clock source select bit 1 = External clock from pin RC0/T1CKI 0 = Internal clock (Fosc/4 from XTAL)
<b>TMR1ON</b>	D0	Timer1 ON and OFF control bit 1 = Enable (start) Timer1 0 = Stop Timer1

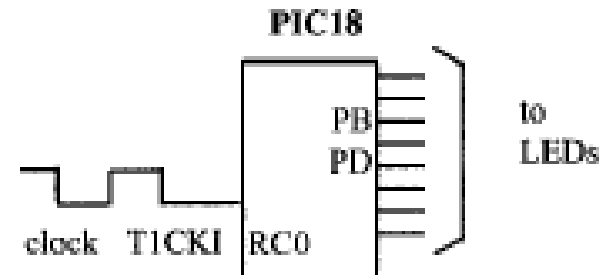
# Example

Assuming that a signal is fed into pin T1CKI, write a program for counter 1 to count the pulses and display the state of the count on PORTB and D

```
BSF      TRISC, 0      ; PORTC.0 as external input for timer1
CLRF     TRISB
CLRF     TRISD
MOVLW    0x02          ; control word for timer1
MOVWF    T1CON

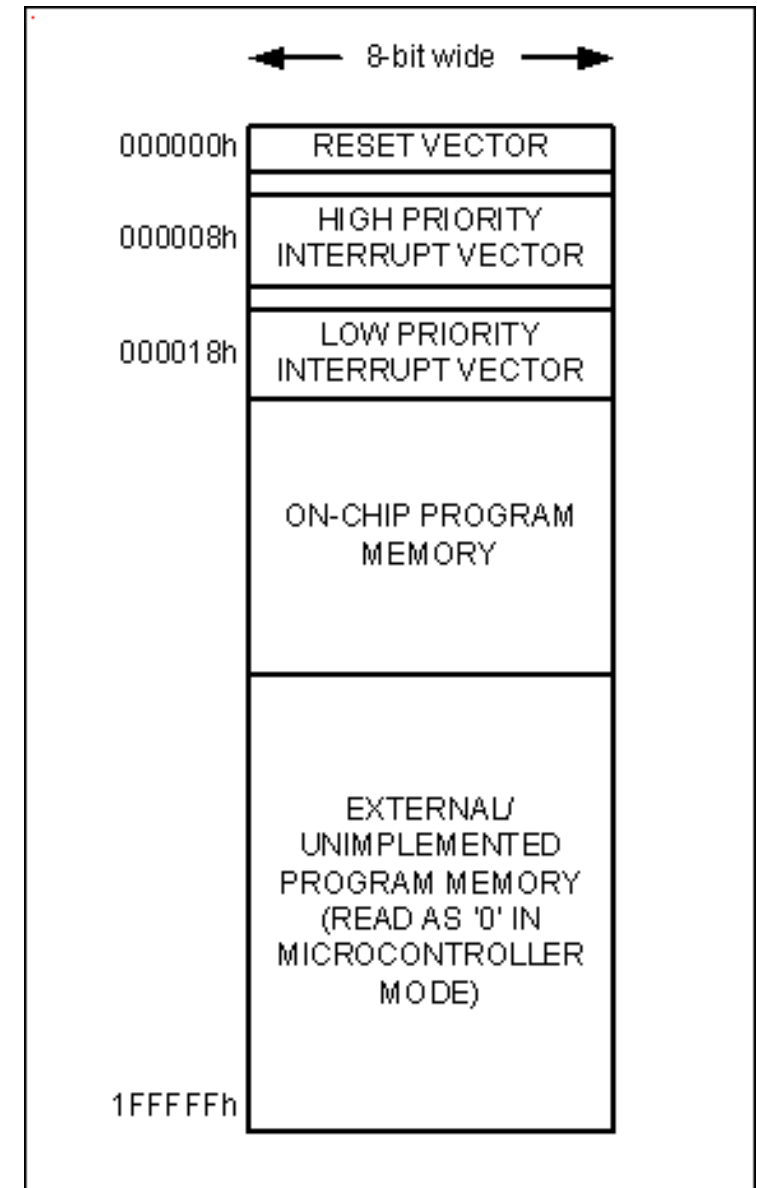
HERE:
MOVLW    0x0           ; TMR1L=0
MOVWF    TMR1L
MOVWF    TMR1H
BCF      PIR1, TMR1IF
BSF      T1CON, TMR1ON

AGAIN:
MOVFF    TMR1L, PORTB
MOVFF    TMR1H, PORTD
BTFSS    PIR1, TMR1IF
BRA      AGAIN
BCF      T1CON, TMR1ON
GOTO     HERE
```



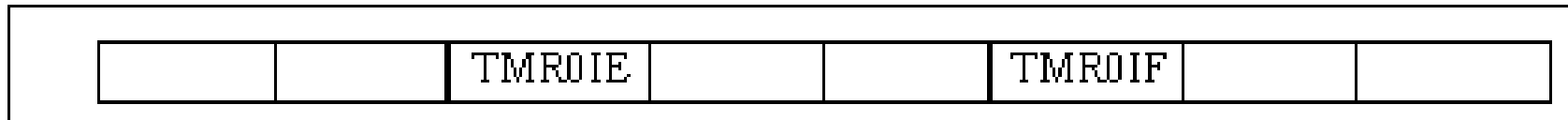
## 4.4.4 timer interrupt

- previously, we program timer with polling method
  - monitor TMR0IF
  - wait until TMR0IF is raised
- here, we use interrupt to program the timer
  - when the timer rolls over, TMR0IF is raised
  - microcontroller jumps to interrupt vector table
  - service the ISR
- TMR0IE enables the interrupt for Timer0



Interrupt	Flag Bit	Register	Enable Bit	Register
Timer0	TMR0IF	INTCON	TMR0IE	INTCON
Timer1	TMR1IF	PIR1	TMR1IE	PIE1
Timer2	TMR2IF	PIR1	TMR2IE	PIE1
Timer3	TMR3IF	PIR3	TMR3IE	PIE2

### Timer Interrupt Flag Bits and Associated Registers



INTCON Register with Timer0 Interrupt Enable and Interrupt Flag

# INTCON

## ENABLES

## FLAGS

Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTCON	GIE	PEIE	TOIE	INTE	GPIE	TOIF	INTF	GPIF

global  
interrupt  
enable

- INT pin interrupt
- TMR0 overflow interrupt

• GP port  
change  
interrupt

- INT pin interrupt
- TMR0 overflow interrupt

• GP port  
change  
interrupt

**GIE/GIEH:** Global Interrupt Enable bit

When IPEN = 0:

1 = Enables all unmasked interrupts

0 = Disables all interrupts

When IPEN = 1:

1 = Enables all high-priority interrupts

0 = Disables all interrupts

**PEIE/GIEL:** Peripheral Interrupt Enable bit

When IPEN = 0:

1 = Enables all unmasked peripheral interrupts

0 = Disables all peripheral interrupts

When IPEN = 1:

1 = Enables all low-priority peripheral interrupts

0 = Disables all low-priority peripheral interrupts

**TMR0IE:** TMR0 Overflow Interrupt Enable bit

1 = Enables the TMR0 overflow interrupt

0 = Disables the TMR0 overflow interrupt

**INT0IE:** INT0 External Interrupt Enable bit

1 = Enables the INT0 external interrupt

0 = Disables the INT0 external interrupt

**RBIE:** RB Port Change Interrupt Enable bit

1 = Enables the RB port change interrupt

0 = Disables the RB port change interrupt

**TMR0IF:** TMR0 Overflow Interrupt Flag bit

1 = TMR0 register has overflowed (must be cleared in software)

0 = TMR0 register did not overflow

**INT0IF:** INT0 External Interrupt Flag bit

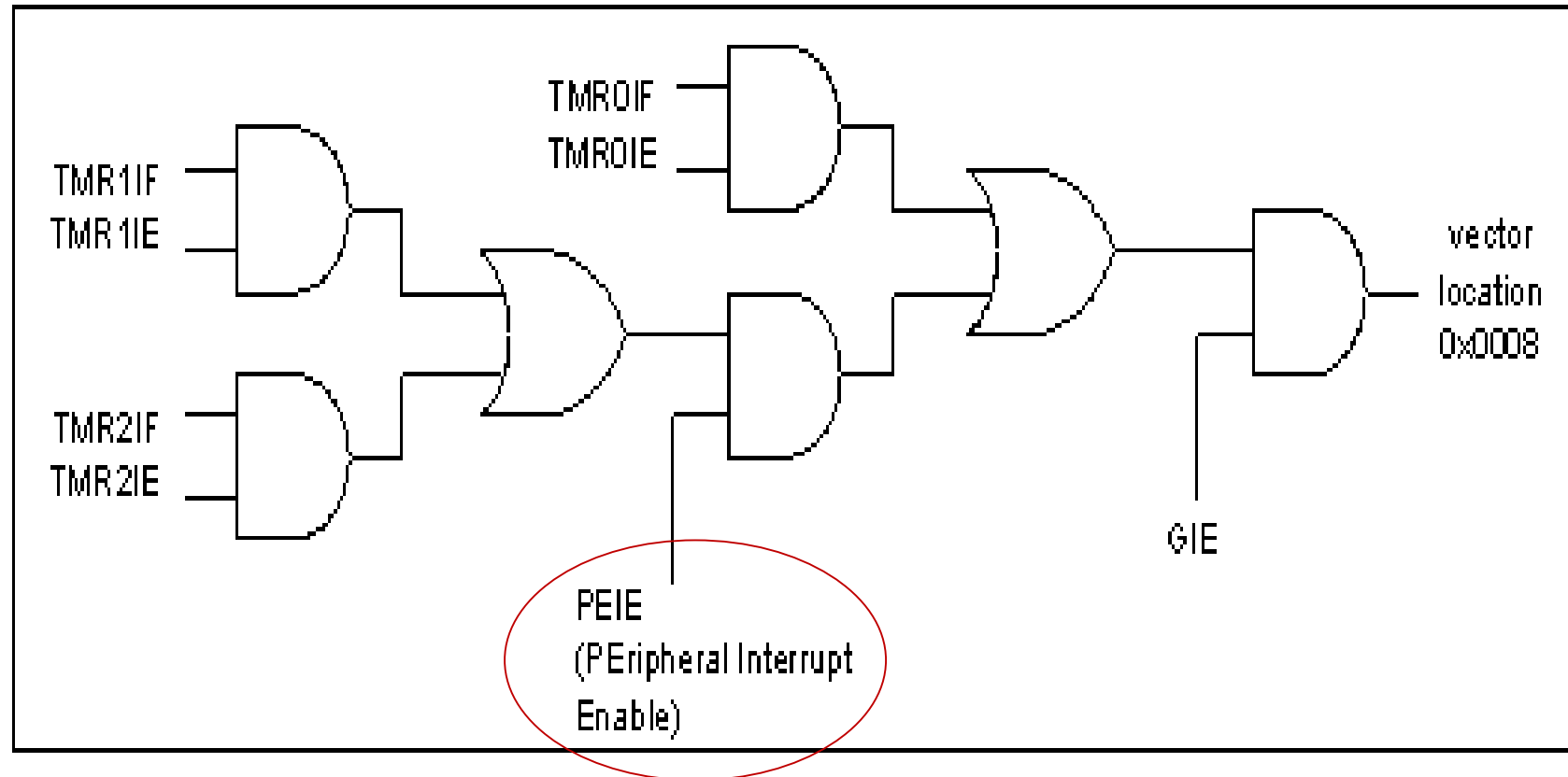
1 = The INT0 external interrupt occurred (must be cleared in software)

0 = The INT0 external interrupt did not occur

**RBIF:** RB Port Change Interrupt Flag bit<sup>(1)</sup>

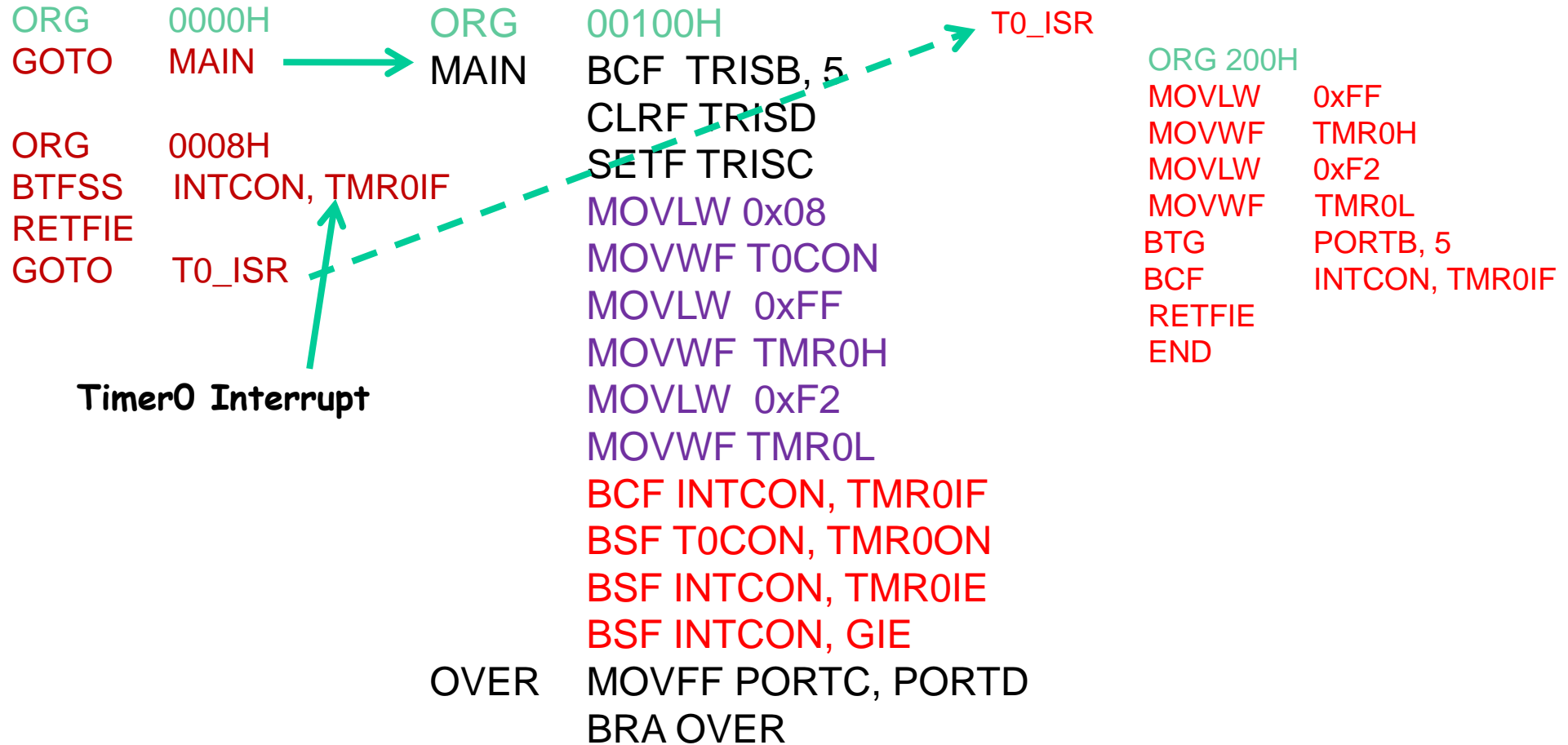
1 = At least one of the RB<7:4> pins changed state (must be cleared in software)

0 = None of the RB<7:4> pins have changed state





# Program



# Example

Use **Timer0** and **Timer1** interrupts to generate square waves on pins **RB1** and **RB7** respectively, while data is being transferred from PORTC to PORTD.

```
ORG 0000H
GOTO MAIN
ORG 0008H
GOTO CHK_INT
ORG 0040H

CHK_INT
    BTFSC INTCON, TMR0IF
    CALL T0_ISR
    BTFSC PIR1, TMR1IF
    CALL T1_ISR
    RETFIE

MAIN
    ORG 0100H
    BCF TRISB, 1
    BCF TRISB, 7
    CLRF TRISD
    SETF TRISC
    MOVLW 0x08
    MOVWF T0CON
    MOVLW 0xFF
    MOVWF TMR0H
    MOVLW 0xF2
    MOVWF TMR0L
    BCF INTCON, TMR0IF
```

```
MOVLW 0x0
MOVWF T1CON
MOVLW 0xFF
MOVWF TMR1H
MOVLW 0xF2
MOVWF TMR1L
BCF PIR1, TMR1IF
BSF INTCON, TMR0IE
BSF PIE1, TMR1IE
BSF INTCON, PEIE
BSF INTCON, GIE
BSF T0CON, TMR0ON
BSF T1CON, TMR1ON

OVER
    MOVFF PORTC, PORTD
    BRA OVER
```

```
T0_ISR
    ORG 200H
    MOVLW 0xFF
    MOVWF TMR0H
    MOVLW 0xF2
    MOVWF TMR0L
    BTG PORTB, 1
    BCF INTCON, TMR0IF
    RETURN

T1_ISR
    ORG 300H
    MOVLW 0xFF
    MOVWF TMR1H
    MOVLW 0xF2
    MOVWF TMR1L
    BTG PORTB, 7
    BCF PIR1, TMR1IF
    RETURN
END
```

## Summary

- ◆ programming Timer0
- ◆ programming Timer1
- ◆ counter programming
- ◆ timer interrupt