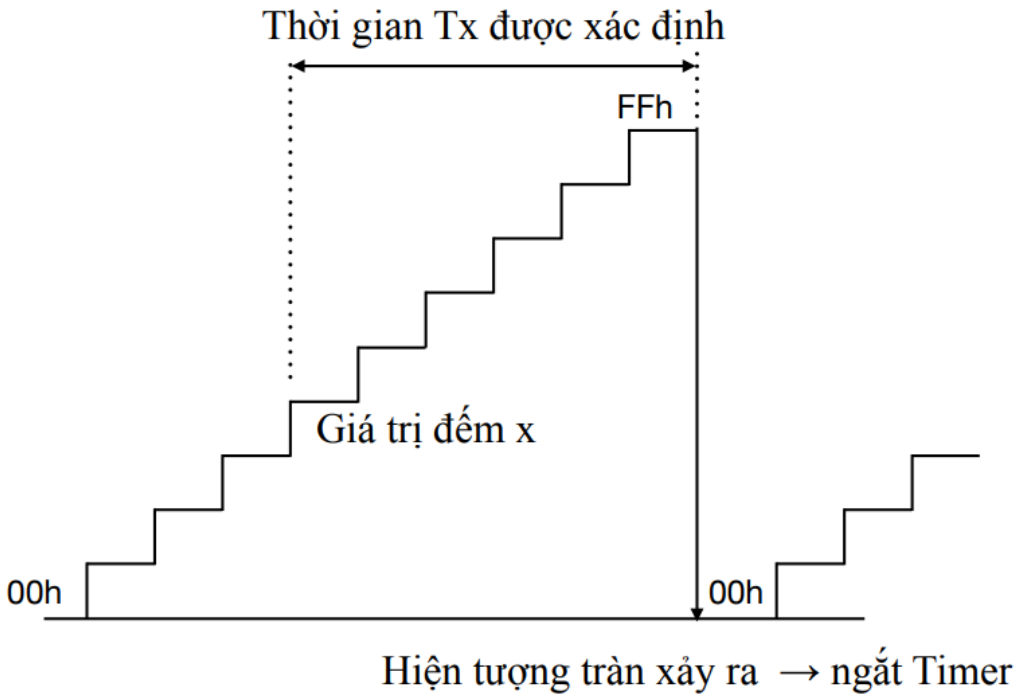
Timer & Counter.

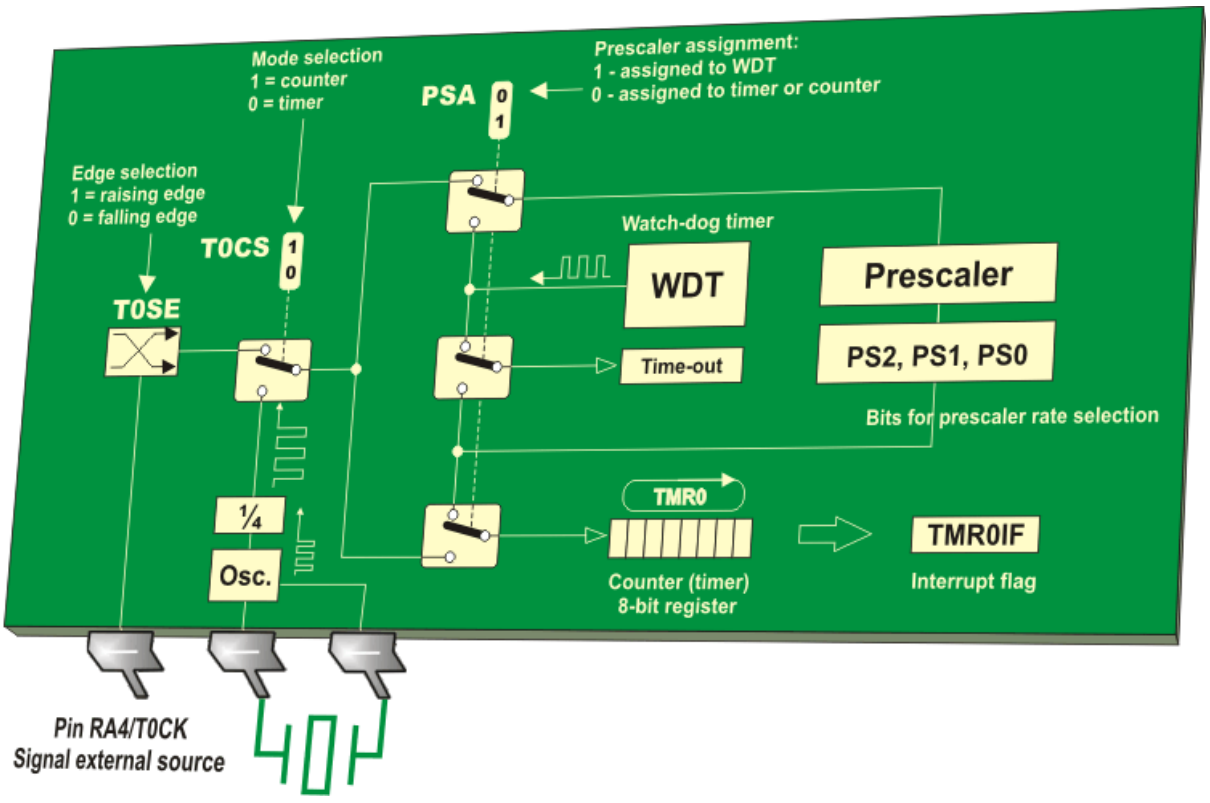
- Timer trong PIC16F877A
- Có 3 timer được đánh số Timer0, Timer1 và Timer2
- Timer 0 ,2 là timer/counter 8 bit (Timer1 là 16 bit)
- Có thể đọc và ghi
- Có bộ chia trước 8 bit có thể lập trình bằng phần mềm
- Cho phép lựa chọn xung nguồn clock bên trong hoặc bên ngoài
- Phát sinh ngắt khi bị tràn từ FFh xuống 00h
- Cho phép lựa chọn các tác động theo sườn xung clock bên ngoài

Hoạt động của Timer



Câu hỏi đặt ra

- Giá trị đếm nằm ở đâu ?
- Khi nào thì giá trị đếm tăng ?
- Thời gian Tx được xác định như thế nào ?
- Khi tràn thì điều gì xảy ra ?



Timer0

- Giá trị đếm nằm ở đâu ? ⟶ Thanh ghi TMR0
- Khi nào thì giá trị đếm tăng ?

Hoạt động chế độ Timer

- Clear bit TOSC tức là bit thứ 5 của POTION_REG)
- TMR0 sẽ tăng ứng với mỗi chu kỳ máy (tức là gấp 4 lần chu kỳ dao động)

Hoạt động chế độ Counter

- Set bit TOSC tức là bit thứ 5 của OPTION_REG)
- TMR0 sẽ tăng ứng với mỗi xung tác động vào chân RA4/TOCK1

  TOSE=0 sườn lên
  TOSE=1 sườn xuống

  TOSE là bit 4 của OPTION_REG

Timer0

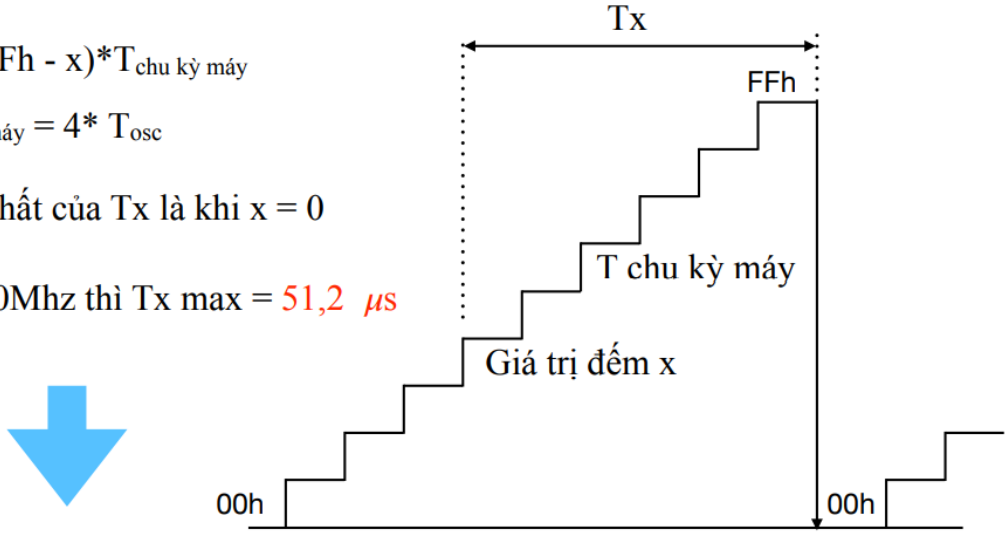
- Thời gian Tx được xác định như thế nào ?

$Tx = (FFh - x)*T_{chu\ kỳ\ máy}$

$T_{chu\ kỳ\ máy} = 4* T_{osc}$

Giá trị lớn nhất của Tx là khi x = 0

Nếu f_{osc} = 20Mhz thì Tx max = 51,2 μs



Làm thế nào để tăng thời gian Tx lên ?

Timer0

Sử dụng bộ chia trước prescale

Khi gán bộ chia trước prescale cho Timer0 thì không thể sử dụng nó cho WDT và ngược lại

OPTION_REG REGISTER (ADDRESS 81h, 181h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
$\overline{\text{RBPU}}$	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

3 bit xác định prescale

PS2:PS0: Prescaler Rate Select bits

Bit Value	TMR0 Rate	WDT Rate
000	1 : 2	1 : 1
001	1 : 4	1 : 2
010	1 : 8	1 : 4
011	1 : 16	1 : 8
100	1 : 32	1 : 16
101	1 : 64	1 : 32
110	1 : 128	1 : 64
111	1 : 256	1 : 128

$Tx = Prescale*(FFh - x)*T_{\text{chu kỳ máy}}$

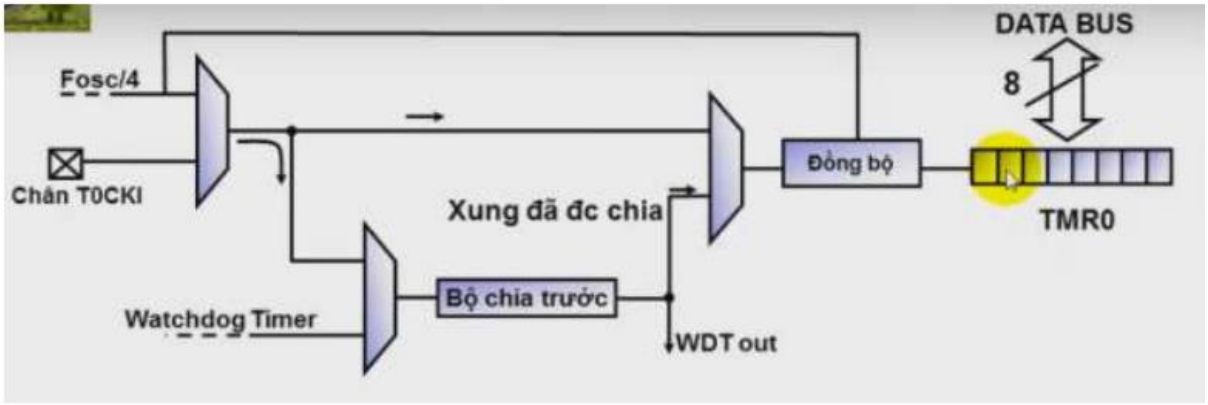
PSA: Prescaler Assignment bit

1 = Prescaler is assigned to the WDT

0 = Prescaler is assigned to the Timer0 module

Lặp đi lặp lại hiện tượng tràn Timer cho đến khi cần

Sử dụng timer có khả năng đếm lớn hơn



- Giải thích: Nguồn xung vào của Timer0 có thể lấy từ nguồn dao động nội(Fosc/4) hoặc nguồn xung đưa vào chân RA4/T0CKI. Xung này được đưa thẳng đến bộ đếm hoặc thông qua bộ chia trước (bộ chia trước có thể sử dụng cho Timer0 hoặc watchdog timer). Giá trị đếm được lưu vào thanh ghi TMR0(8bit) Khi bộ chia trước sử dụng cho watchdog timer thì xung sẽ được đưa thẳng vào bộ đếm luôn(coi như bộ chia 1:1)

3. Thanh ghi –

- **Các thanh ghi liên quan**
- Thanh ghi INTCON

REGISTER 2-3: INTCON REGISTER (ADDRESS 0Bh, 8Bh, 10Bh, 18Bh)

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-x
GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF
bit 7							bit 0

bit 7: GIE: Global Interrupt Enable bit: **ngắt toàn cục**
1 = Enables all unmasked interrupts: cho phép ngắt toàn cục
0 = Disables all interrupts: không cho phép ngắt
bit 6: PEIE: **ngắt cục bộ**
1 = Enables all unmasked peripheral interrupts
0 = Disables all peripheral interrupt
bit 5: T0IE: TMR0 Overflow Interrupt Enable bit: **ngắt tràn timer0**
1 = Enables the TMR0 interrupt// cho phép
0 = Disables the TMR0 interrupt// không cho phép
bit 4: INTE: RB0/INT External Interrupt Enable bit //**bit cho phép ngắt ngoài trên RB0**
1 = Enables the RB0/INT external interrupt// **cho phép**
0 = Disables the RB0/INT external interrupt// **không cho phép**
bit 3: RBIE: RB Port Change Interrupt Enable bit **// ngắt thay đổi trạng thái của các chân PortB (RB4-RB7).**
1 = Enables the RB port change interrupt// **cho phép**
0 = Disables the RB port change interrupt// **không cho phép**
bit 2: T0IF: TMR0 Overflow Interrupt Flag bit// **cờ báo ngắt khi tràn timer0**
1 = TMR0 register has overflowed
0 = TMR0 register did not overflow
bit 1: INTF: RB0/INT External Interrupt Flag bit// **cờ báo ngắt ngoài**
1 = The RB0/INT external interrupt occurred // **ngắt ngoài xảy ra**
0 = The RB0/INT external interrupt did not occur// **không xảy ra**
bit 0 **RBIF**: RB Port Change Interrupt Flag bit// **cờ báo ngắt thay đổi trạng thái portb**
1 = At least one of the RB7:RB4 pins changed state// **ít nhất 1 trong 4 chân thay đổi trạng thái**
0 = None of the RB7:RB4 pins have changed state// **không chân nào thay đổi trạng thái**

- **Thanh ghi OPTION**

REGISTER 2-2: OPTION_REG REGISTER (ADDRESS 81h, 181h)

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
RBPU	INTEDG	T0CS	T0SE	PSA	PS2	PS1	PS0
bit 7							bit 0

bit 6 :**INTEDG**: Interrupt Edge Select bit //**bit chọn chế độ ngắt cạnh lên hay cạnh xuống**

1 = Interrupt on rising edge of RB0/INT pin // **ngắt cạnh lên**

0 = Interrupt on falling edge of RB0/INT pin // **ngắt cạnh xuống**

Các bit còn lại phục vụ chức năng khác:

Bit 7: RBPU: PORTB pull-up enable bit

=1:Không cho phép chức năng pull-up của portB

=0: Cho phép

Bit 5: TOCS Timer0 Clock Source select bit

=1: clock lấy từ chân RA4/TOCKI

=0: Dùng xung clock bên trong

Bit 4: TOSE Timer0 Source Edge Select bit

=1: tác động cạnh lên

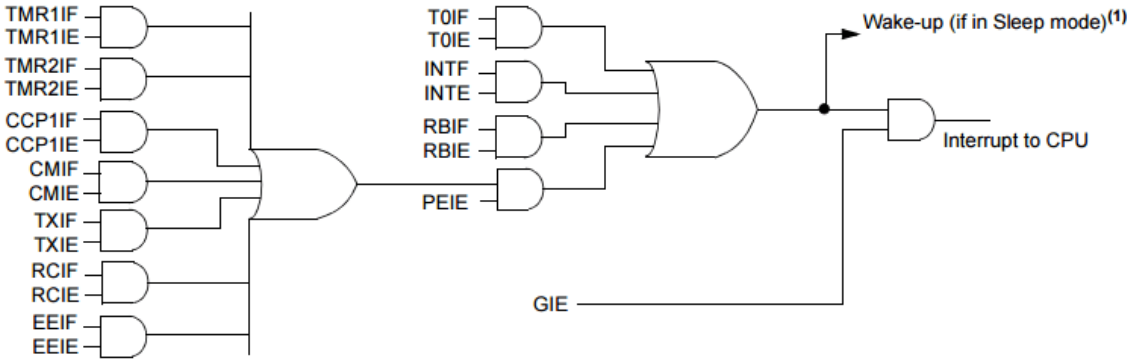
=0: tác động cạnh xuống

Bit 3: PSA Prescaler Assignment Select bit: chọn bộ chia tần

Bit 2-0: PS2:PS0 Prescaler Rate Select bit: thiết lập tỉ số chia tần

- **bit 7 RBPU:** PORTB Pull-up Enable bit // **bit điều khiển điện trở treo của portb**.
  1 = PORTB pull-ups are disabled
  0 = PORTB pull-ups are enabled by individual port latch values
- **bit 6 INTEDG**: Interrupt Edge Select bit// bit chọn ngắt cạnh lên hay xuống của ngắt ngoài
  1 = Interrupt on rising edge of RB0/INT pin
  0 = Interrupt on falling edge of RB0/INT pin
- **bit 5 T0CS**: TMR0 Clock Source Select bit: **bit lựa chọn nguồn xung cho TMR0**
  1 = Transition on RA4/T0CKI pin // **đếm xung ngoại đưa đến chân T0CKI.**
  0 = Internal instruction cycle clock (CLKOUT) // **đếm xung clock nội bên trong.**
- **bit 4 T0SE**: TMR0 Source Edge Select bit // **bit lựa chọn cạnh tích cực**
  1 = Increment on high-to-low transition on RA4/T0CKI pin // **tích cực cạnh xuống ở chân T0CKI.**
  0 = Increment on low-to-high transition on RA4/T0CKI pin// **tích cực cạnh lên ở chân T0CKI.**
- **bit 3 PSA**: Prescaler Assignment bit //**bit gán bộ chia**
  1 = Prescaler is assigned to the WDT // **gán bộ chia cho WDT**
  0 = Prescaler is assigned to the Timer0 module// **Gán bộ chia cho timer0**
- **bit 2-0 PS2:PS0**: Prescaler Rate Select bits // **bit chọn tỉ lệ bộ chia trước**

FIGURE 14-14: INTERRUPT LOGIC PIC16F628A



Note 1: Some peripherals depend upon the system clock for operation. Since the system clock is suspended during Sleep, only those peripherals which do not depend upon the system clock will wake the part from Sleep. See **Section 14.8.1 "Wake-up from Sleep"**.

NOTE

Nếu bit T0CS bằng 1 thì chọn chế độ đếm xung ngoài Counter. Trong chế độ đếm xung ngoài thì xung đếm đưa đến chân RA4/T0CKI. Bit T0SE = 0 thì chọn cạnh lên, ngược lại thì chọn cạnh xuống.

Bộ chia trước không thể đọc/ghi có mối quan hệ với Timer0 và Watchdog Timer.

a. Ngắt của Timer0

Khi giá trị đếm trong thanh ghi TMR0 tràn từ FFh về 00h thì phát sinh ngắt, cờ báo ngắt TMR0IF lên 1. Ngắt có thể ngăn bằng bit cho phép ngắt TMR0IE.

Trong chương trình con phục vụ ngắt Timer0 phải xóa cờ báo ngắt TMR0IF. Ngắt của TMR0 không thể kích CPU thoát khỏi chế độ ngủ vì bộ định thời sẽ ngừng khi CPU ở chế độ ngủ.

b. Timer0 đếm xung ngoại

Muốn đếm xung ngoại thì xung được đưa đến ngõ vào T0CKI, việc đồng bộ tín hiệu xung ngõ vào T0CKI với xung clock bên trong được thực hiện bằng cách lấy mẫu ngõ ra bộ chia ở những chu kì Q2 và Q4 của xung clock bên trong. Điều này rất cần thiết cho T0CKI ở trạng thái mức cao ít nhất 2 TOSC và ở trạng thái mức thấp ít nhất 2 TOSC

c. Bộ chia trước

Bộ chia trước có thể gán cho Timer0 hoặc gán cho Watchdog Timer. Các bit PSA và PS2:PS0 chọn đối tượng gán và tỉ lệ chia.

Khi được gán cho Timer0 thì tất cả các lệnh ghi cho thanh ghi TMR0 (ví dụ CLRF 1, MOVWF 1, BSF 1, …) sẽ xoá bộ chia trước.

Khi được gán cho WDT thì lệnh CLRWDT sẽ xoá bộ chia trước cùng với Watchdog Timer.

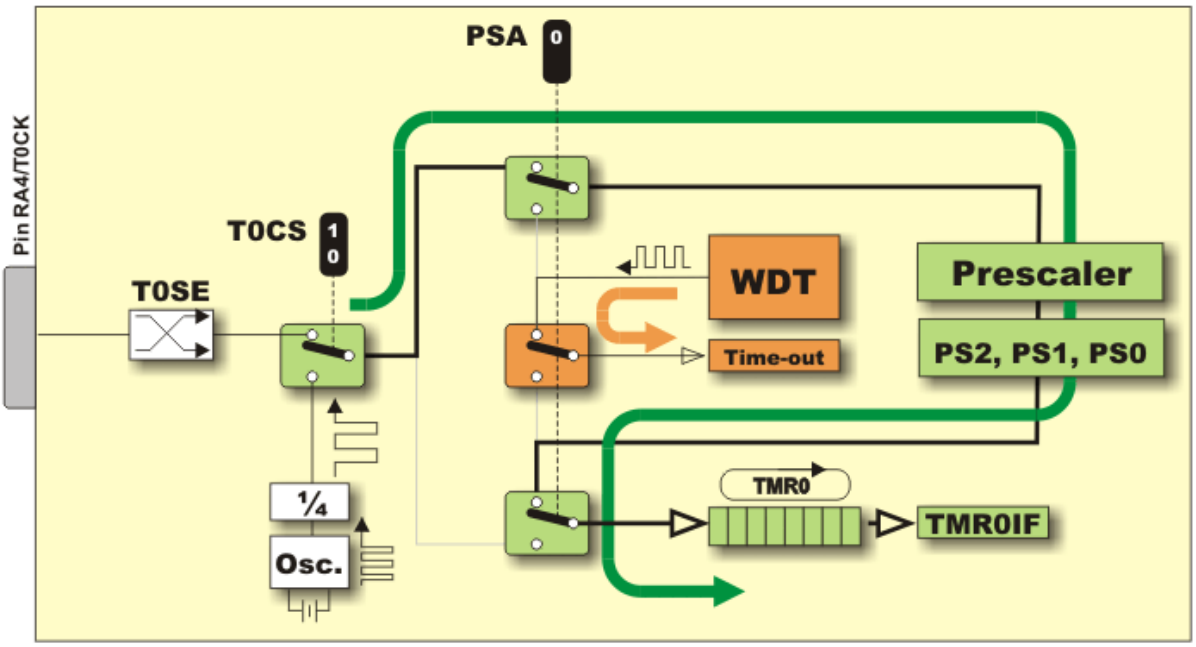
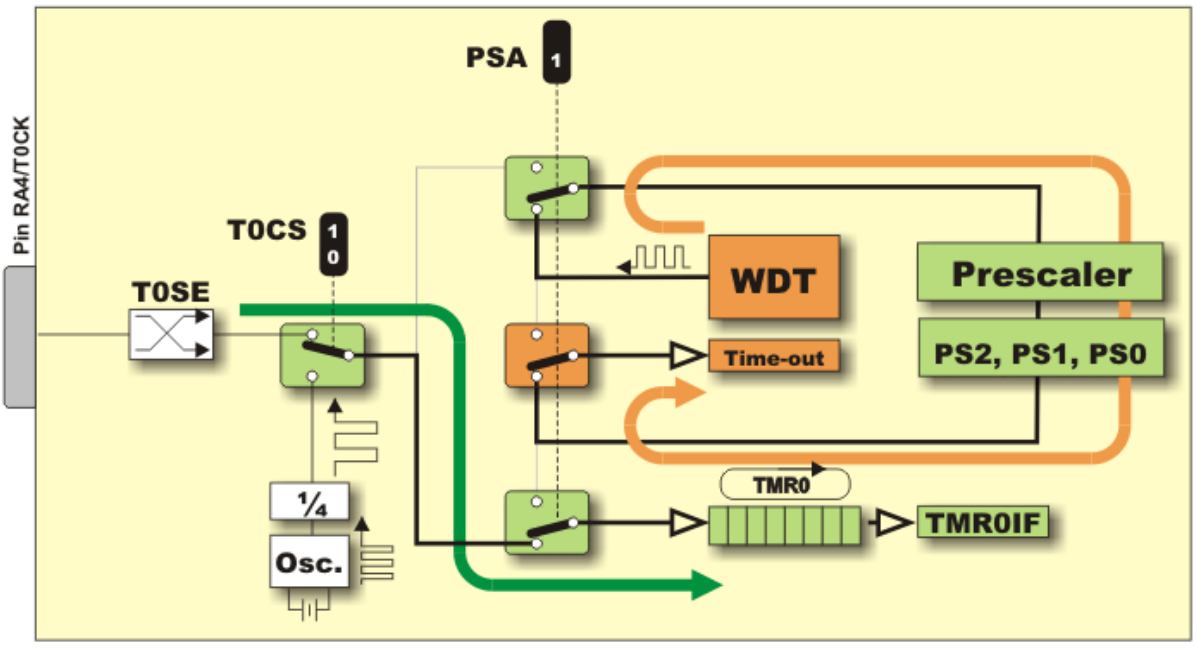
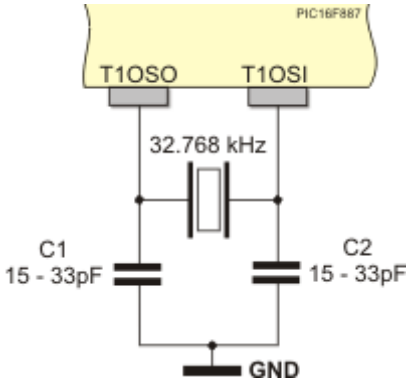


Fig. 4-3 The function of the PSA bit 0



2. Timer0



Chúng ta sẽ có công thức tính thời gian để cài đặt cho Timer 0 như sau:
*(8bit) 256 = giá trị nạp + (giá trị mong muốn/(tỉ lệ * chu kì lệnh))*
Chu kì lệnh = 4 * chu kì máy
Chu kì máy = 1/ tần số của vdk

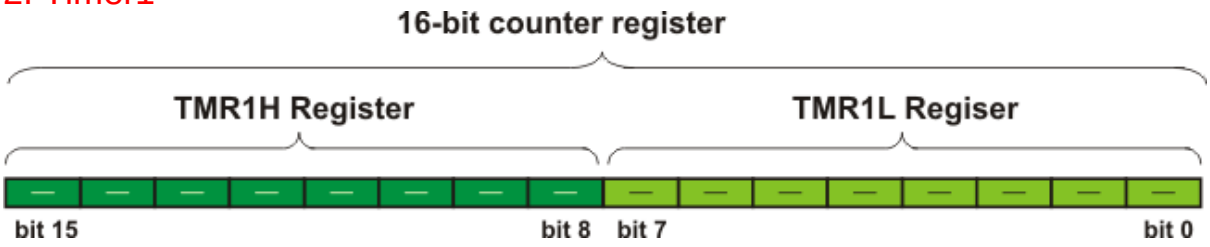
VD: Thạch anh 8MHz ~ F=8MHz
B1.
F = 8MHz = 8*10^6 Hz suy ra T = 1/F = 1/8*10^-6 (s)=1/8us
Suy ra: chu kì lệnh = 4*1/OSC = 4*1/8 = 0.5 us
B2. áp dụng công thức trên
256 = giá trị nạp + (giá trị mong muốn/(tỉ lệ chia*chu kì lệnh)) (us)
với:
- tỉ lệ chia: 8
- chu kì lệnh: 0.5 us

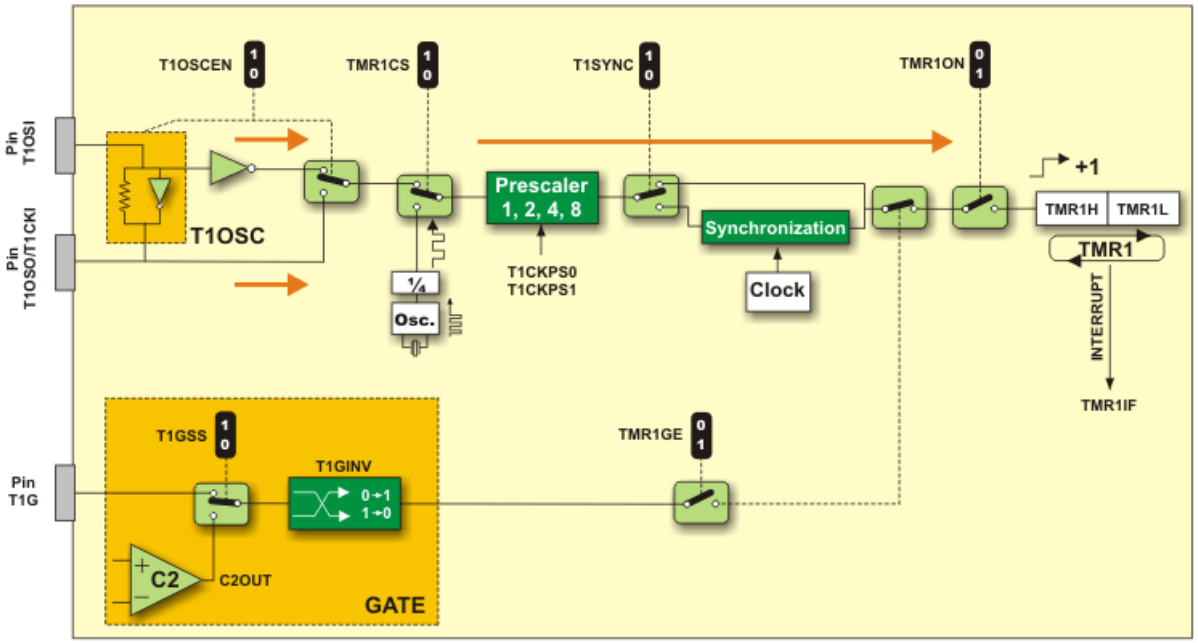
từ công thức suy ra
(256 - giá trị nạp) * 8*0.5 = giá trị mong muốn (us)
Chọn giá trị nạp sao cho giá trị mong muốn lớn nhất và là số **chia hết** cho 1.000.000 us (~1s)

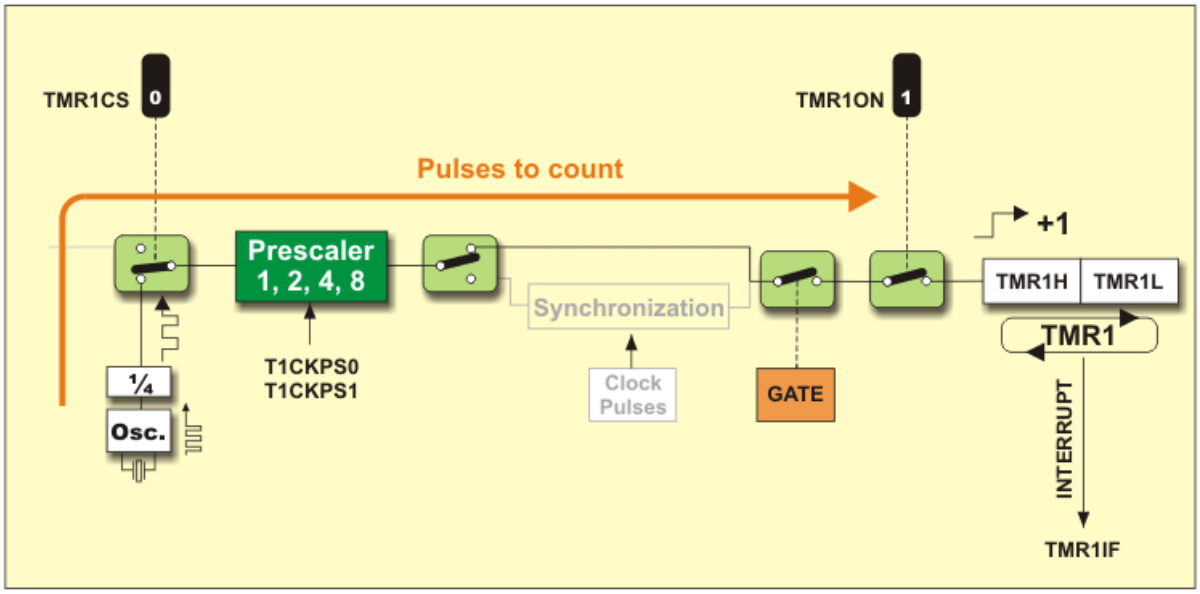
giá trị nạp	giá trị mong muốn	số lần tràn để được 1 giây
Chọn 6	1.000 us	1.000.000/1000=1000 lần

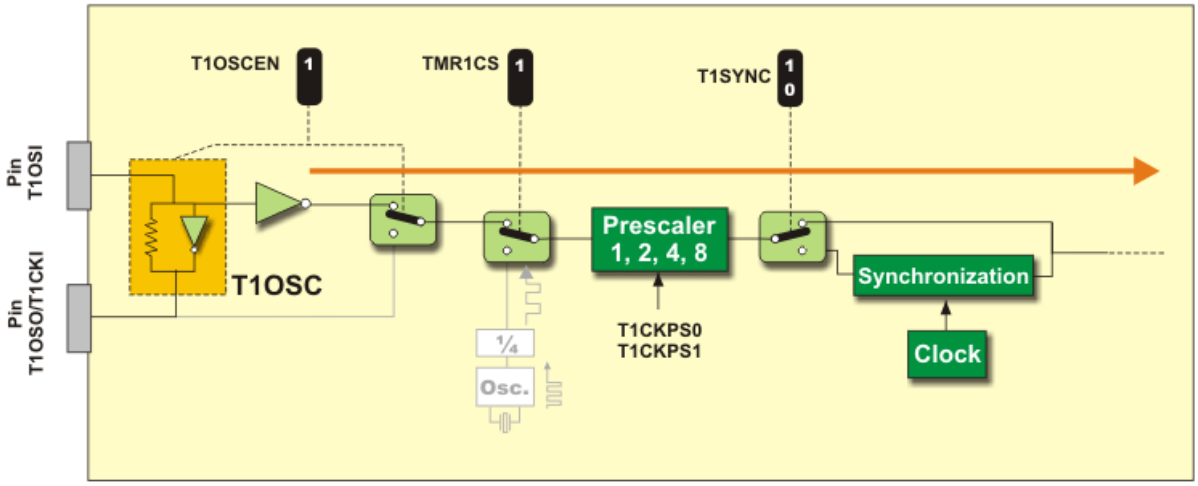
Chọn 56	800 us	1000000/800=1250 lần

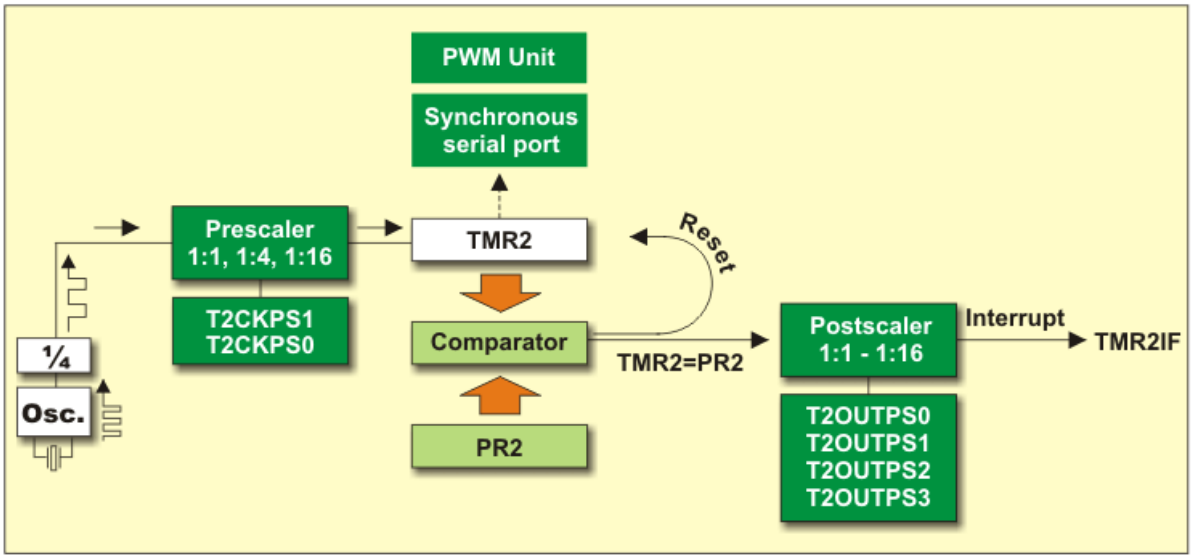
2. Timer1






TMR1CS
0
TMR1ON
1
Pulses to count
+1
Prescaler
1, 2, 4, 8
Synchronization
TMR1H
TMR1L
TMR1
T1CKPS0
T1CKPS1
Clock
Pulses
GATE
1/4
Osc.
INTERRUPT
TMR1IF


T1OSCEN
1
TMR1CS
1
T1SYNC
1
0
Pin
T1OSI
Pin
T1OSO/T1CKI
T1OSC
Prescaler
1, 2, 4, 8
Synchronization
T1CKPS0
T1CKPS1
Clock
1/4
Osc.


PWM Unit
Synchronous
serial port
Prescaler
1:1, 1:4, 1:16
TMR2
Reset
T2CKPS1
T2CKPS0
Comparator
TMR2=PR2
Postscaler
1:1 - 1:16
Interrupt
TMR2IF
1/4
Osc.
PR2
T2OUTPS0
T2OUTPS1
T2OUTPS2
T2OUTPS3

T2CON	-	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	Features
	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Bit name

Legend

-	Bit is unimplemented
R/W	Readable/Writable bit
(0)	After reset, bit is cleared

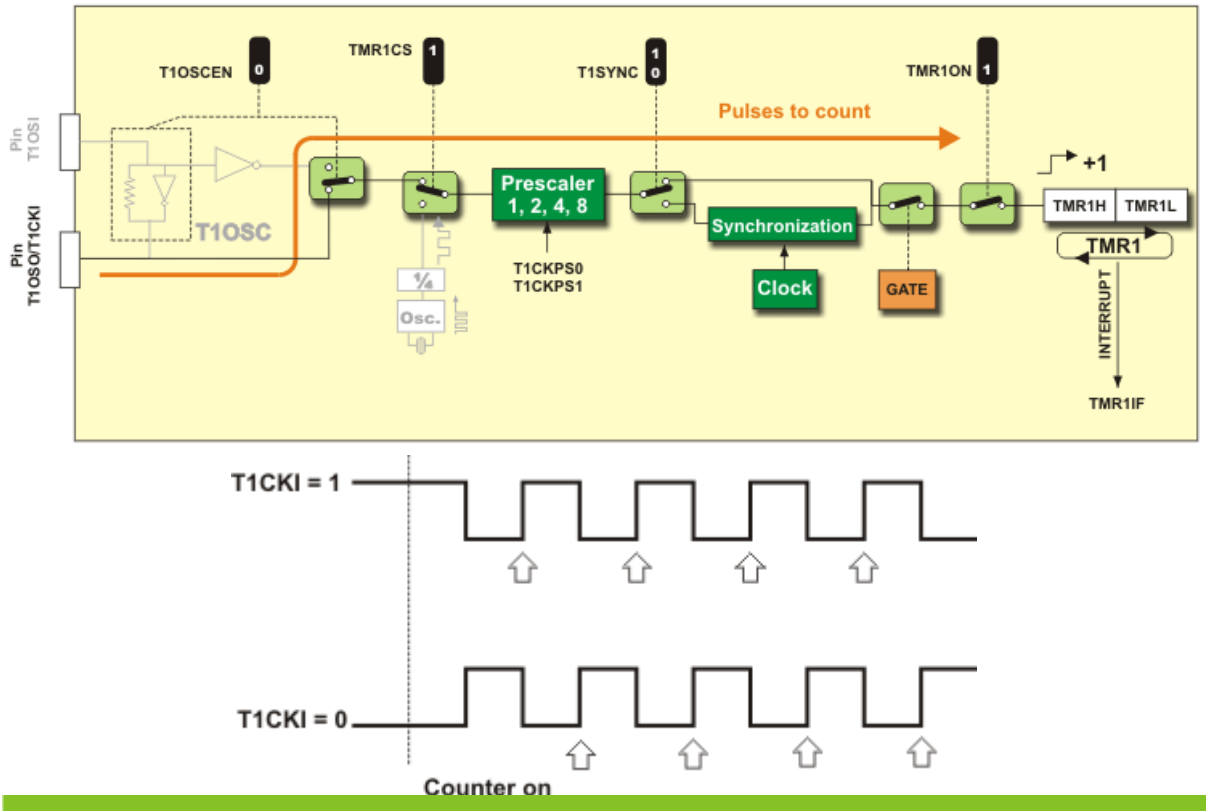
Chúng ta sẽ tính toán các thông số để cài đặt cho Timer 1 theo công thức sau : Ví dụ muốn định thời 50ms = 50000us sử dụng prescaler 1:8, Fosc = 4 Mhz thì ta có công thức tính như sau :

GT = 65536 - Tdelay.Fosc/(4.Kprescaler)

Từ công thức trên thay số vào ta được như sau :

GT = 65536 - 50.10^-3.4.10^6.(4/8) = 59286

3. Counter



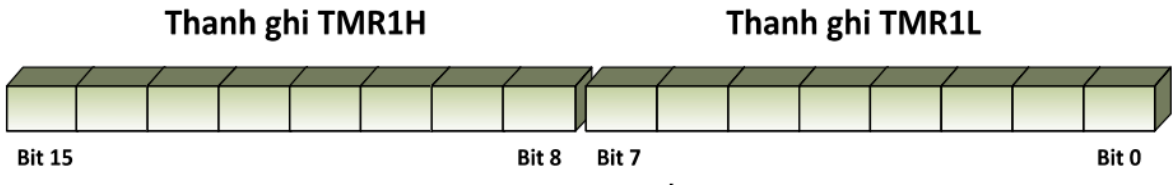
T1CON	T1GINV	TMR1GE	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC	TMR1CS	TMR1ON	Features
	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	R/W (0)	
	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Bit name

Legend

R/W	Readable/Writable bits
(0)	After reset, bit is cleared

1. Các thanh ghi quan trọng của timer 1

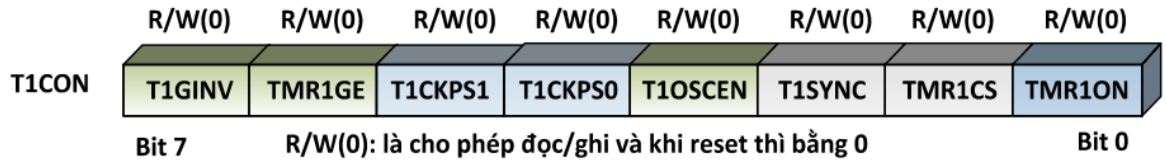
a. Thanh ghi TMR1L và TMR1H:



2 thanh ghi này dùng để ghi giá trị counter của timer 1.
Ngoài ra, khi khối CCP hoạt động ở chế độ compare/ capture thì 2 thanh ghi này còn có tác dụng như sau:

- Compare: 1 sự kiện được kích hoạt nếu giá trị của cặp thanh ghi CCPRxH:CCPRxL trùng khớp với giá trị của TMR1L:TMR1H
- Capture: giá trị của cặp thanh ghi TMR1L:TMR1H được sao chép vào cặp CCPRxH:CCPRxL trên 1 sự kiện đã đặt trước

Thanh ghi T1CON (Timer 1 control register):



- Bit 7: Bit đảo cổng của timer 1 (T1GINV)
- T1GINV = 1: timer 1 đếm khi cổng ở mức 1
- T1GINV = 0: timer 0 đếm khi cổng ở mức 0
- Bit 6: Bit cho phép cổng của timer 1 (TMR1GE), bit này chỉ có tác dụng khi TMR1ON = 1
- TMR1GE = 1: timer 1 mở nếu cổng của nó không tích cực
- TMR1GE = 0: timer 1 mở
- Bit 5-4: Các bit chọn bộ chia trước timer 1 (T1CKPS1, T1CKPS0)

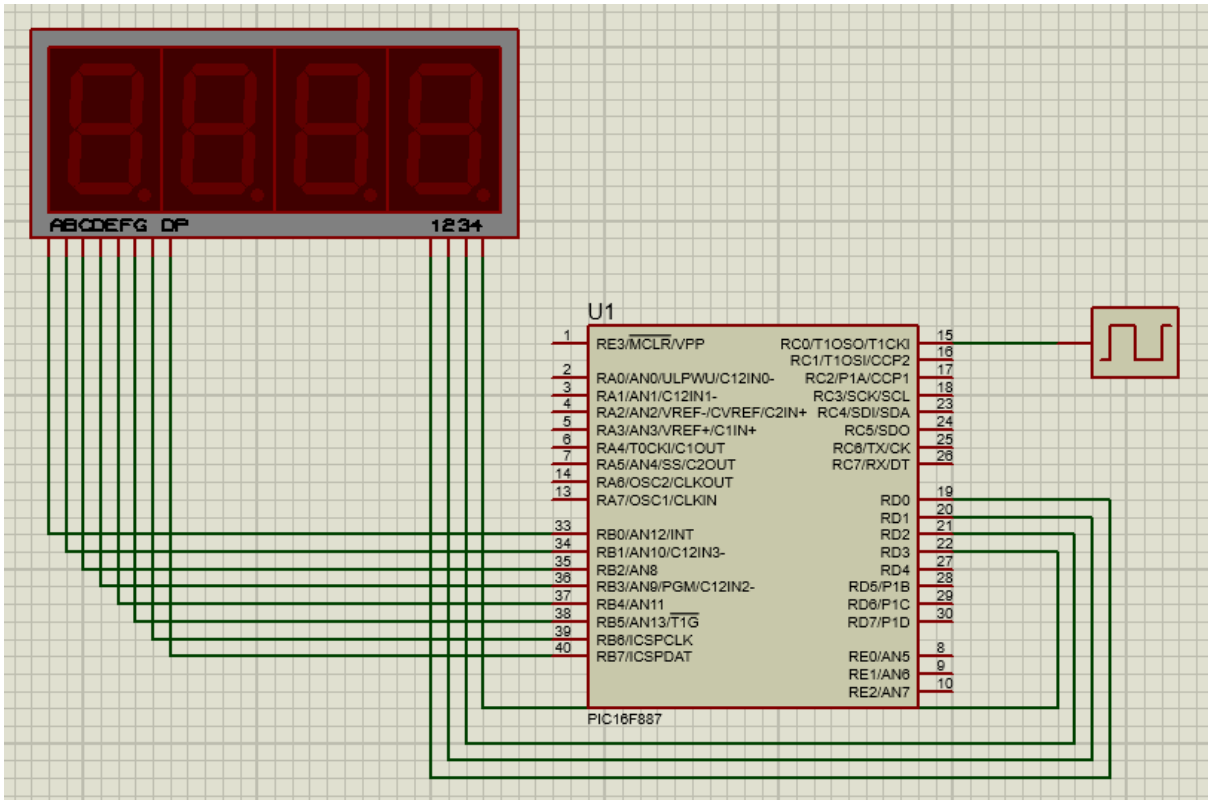
PS1	PS0	Giá trị chia
0	0	1
0	1	2
1	0	4
1	1	8

- Bit 3: bit cho phép bộ dao động timer 1 (T1OSCEN)
- T1OSCEN = 1: cho phép
- T1OSCEN = 0: tắt bộ dao động
- Bit 2: Bit đồng bộ ngõ vào xung clock ngoại timer 1 (T1SYNC), bit này chỉ có tác dụng khi TMR1CS = 1
- T1SYNC = 1: không đồng bộ ngõ vào clock từ bên ngoài Bộ đếm tiếp tục tăng bất đồng bộ với xung bên trong. Bộ đếm vẫn đếm khi CPU ở trong chế độ ngủ và khi tràn sẽ phát sinh ngắt và đánh thức CPU. Ngắt T1 có thể ngăn được.
- T1SYNC = 0: đồng bộ ngõ vào clock từ bên ngoài, nếu CPU ở chế độ ngủ thì Timer1 sẽ không đếm vì mạch đồng bộ ngừng hoạt động.
- Bit 1: Bit chọn nguồn xung timer 1 (TMR1CS)
- TMR1CS = 1: chọn nguồn xung ngoại từ chân T1CKI
- TMR1CS = 0: chọn xung nội Fosc/4
- Bit 0: Bit điều khiển timer 1 (TMR1ON)
- TMR1ON = 1: cho phép timer 1 đếm
- TMR1ON = 1: timer 1 ngừng đếm

II. Ứng dụng timer 1 ở chế độ đếm không đồng bộ

Đọc và ghi Timer1 trong chế độ đếm không đồng bộ: Timer T1 cho phép đọc giá trị các thanh ghi TMR1H hoặc TMR1L khi timer đang đếm xung bất đồng bộ bên ngoài. Khi ghi thì nên ngừng timer lại rồi mới ghi giá trị mong muốn vào các thanh ghi. Nếu ghi mà timer đang đếm vẫn có thể được nhưng có thể tạo ra một giá trị đếm không chính xác.

Ví dụ này, mình sẽ dùng timer 1 để thực hiện đếm xung ngoại và hiển thị số chu kỳ xung qua led 7 đoạn với sơ đồ mạch như sau:



Trước hết cần khởi tạo nguồn xung timer 1 lấy từ chân T1CKI của vi điều khiển với bộ chia tỷ lệ 1:1 để đo chính xác chu kỳ của xung ngoại

```
        T1CONbits.TMR1CS = 1; // nguon xung ngoai
    T1CONbits.T1CKPS0 = 0;
    T1CONbits.T1CKPS1 = 0;// bo chia ty le 1:1
```

Tiếp theo, mình cần phải bắt đầu cho timer 1 đếm từ giá trị 0 bằng cách tác động vào bit TMR1ON và reset giá trị của 2 thanh ghi TMR1H và TMR1L:

```
    T1CONbits.TMR1ON = 1;   // bat dau cho timer 1 dem
    TMR1L = TMR1H = 0;
```

Mình sẽ tạo 1 biến cnt dùng để lưu giá trị của 2 thanh ghi TMR1H và TMR1L sau đó dùng cnt để hiển thị lên led 7 thanh. Dưới đây là code mẫu:

```
// CONFIG1
#pragma config FOSC = HS        // Oscillator Selection bits (HS oscillator: High-speed
crystal/resonator on RA6/OSC2/CLKOUT and RA7/OSC1/CLKIN)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled and can be
enabled by SWDTEN bit of the WDTCON register)
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT disabled)
#pragma config MCLRE = OFF      // RE3/MCLR pin function select bit (RE3/MCLR pin function
is digital input, MCLR internally tied to VDD)
#pragma config CP = OFF         // Code Protection bit (Program memory code protection is
disabled)
#pragma config CPD = OFF        // Data Code Protection bit (Data memory code protection is
disabled)
#pragma config BOREN = OFF      // Brown Out Reset Selection bits (BOR disabled)
#pragma config IESO = OFF       // Internal External Switchover bit (Internal/External
Switchover mode is disabled)
#pragma config FCMEN = OFF      // Fail-Safe Clock Monitor Enabled bit (Fail-Safe Clock
Monitor is disabled)
```

```
#pragma config LVP = OFF        // Low Voltage Programming Enable bit (RB3 pin has digital
I/O, HV on MCLR must be used for programming)
// CONFIG2
#pragma config BOR4V = BOR40V   // Brown-out Reset Selection bit (Brown-out Reset set to
4.0V)
#pragma config WRT = OFF        // Flash Program Memory Self Write Enable bits (Write
protection off)
#include <xc.h>
#include "led7.h"

void main(void)
{
    TRISC0 = 1;             // chan C0 la input de lay nguon xung ngoai
    T1CONbits.TMR1CS = 1;   // nguon xung ngoai
    T1CONbits.T1CKPS0 = 0;
    T1CONbits.T1CKPS1 = 0;  // bo chia ty le 1:1
    T1CONbits.TMR1ON = 1;   // bat dau cho timer 1 dem
    TMR1H = TMR1L = 0;
    TRISD = 0;
    TRISB = 0;
    ADCON1 |= 0x07; // khong cho phep che do ADC while (1)
    {
        unsigned int cnt;
        cnt = TMR1H;  // ghi 8 bit dau tien
        cnt <<= 8;
        cnt |= TMR1L; // ghi 8 bit cuoi
        Led7_display(cnt);
    }
}
```

Hướng dẫn về Counter của Timer 0 của Vi Điều Khiển PIC16F877A dử dụng trình biên

Code Timer 0 nhấp nháy đèn LED đơn

```
#include <stdio.h>
#include <stdlib.h>
#define _XTAL_FREQ 4000000
#include <xc.h>
// CONFIG
#pragma config FOSC = XT        // Oscillator Selection bits (XT oscillator)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled)
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT disabled)
#pragma config BOREN = OFF      // Brown-out Reset Enable bit (BOR disabled)
#pragma config LVP = OFF        // Low-Voltage (Single-Supply) In-Circuit Serial
Programming Enable bit (RB3 is digital I/O, HV on MCLR must be used for programming)
#pragma config CPD = OFF        // Data EEPROM Memory Code Protection bit (Data
EEPROM code protection off)
#pragma config WRT = OFF        // Flash Program Memory Write Enable bits (Write
protection off; all program memory may be written to by EECON control)
#pragma config CP = OFF         // Flash Program Memory Code Protection bit (Code
protection off)
unsigned int Count = 0;
void main(void) {
  TRISCbits.TRISC0 = 0;                 //RC0 pin as output
  //config timer0
  TMR0 = 55;
  OPTION_REGbits.PSA = 0;
  OPTION_REGbits.PS2 = 0;
  OPTION_REGbits.PS1 = 1;
  OPTION_REGbits.PS0 = 0;
  OPTION_REGbits.T0CS = 0;
  while (1)
  {
    if(INTCONbits.TMR0IF == 1)
    {
      INTCONbits.TMR0IF = 0;
      TMR0 = 55;
      Count++;
      if(Count == 125)
      {
        Count = 0;
        PORTCbits.RC0 ^= 1;             //toggle the LED
      }
    }
  }
}
```

VÍ dụ 2

```
/*
 * File:   main.c
 * Author: namgu
 *
 * Created on February 28, 2022, 10:04 PM
 */
// CONFIG
#pragma config FOSC = HS        // Oscillator Selection bits (HS oscillator)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled)
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT disabled)
#pragma config BOREN = ON       // Brown-out Reset Enable bit (BOR enabled)
#pragma config LVP = ON         // Low-Voltage (Single-Supply) In-Circuit Serial Programming Enable bit (RB3/PGM pin has PGM function; low-voltage programming enabled)
#pragma config CPD = OFF        // Data EEPROM Memory Code Protection bit (Data EEPROM code protection off)
#pragma config WRT = OFF        // Flash Program Memory Write Enable bits (Write protection off; all program memory may be written to by EECON control)
#pragma config CP = OFF         // Flash Program Memory Code Protection bit (Code protection off)
#define _XTAL_FREQ 20000000
// #pragma config statements should precede project file includes.
// Use project enums instead of #define for ON and OFF.

#include <xc.h>

//===============INTERRUPTS BY TPN - UTC - 2022====================

void timer_isr()
{
  if (TMR0IF==1)
   {
```

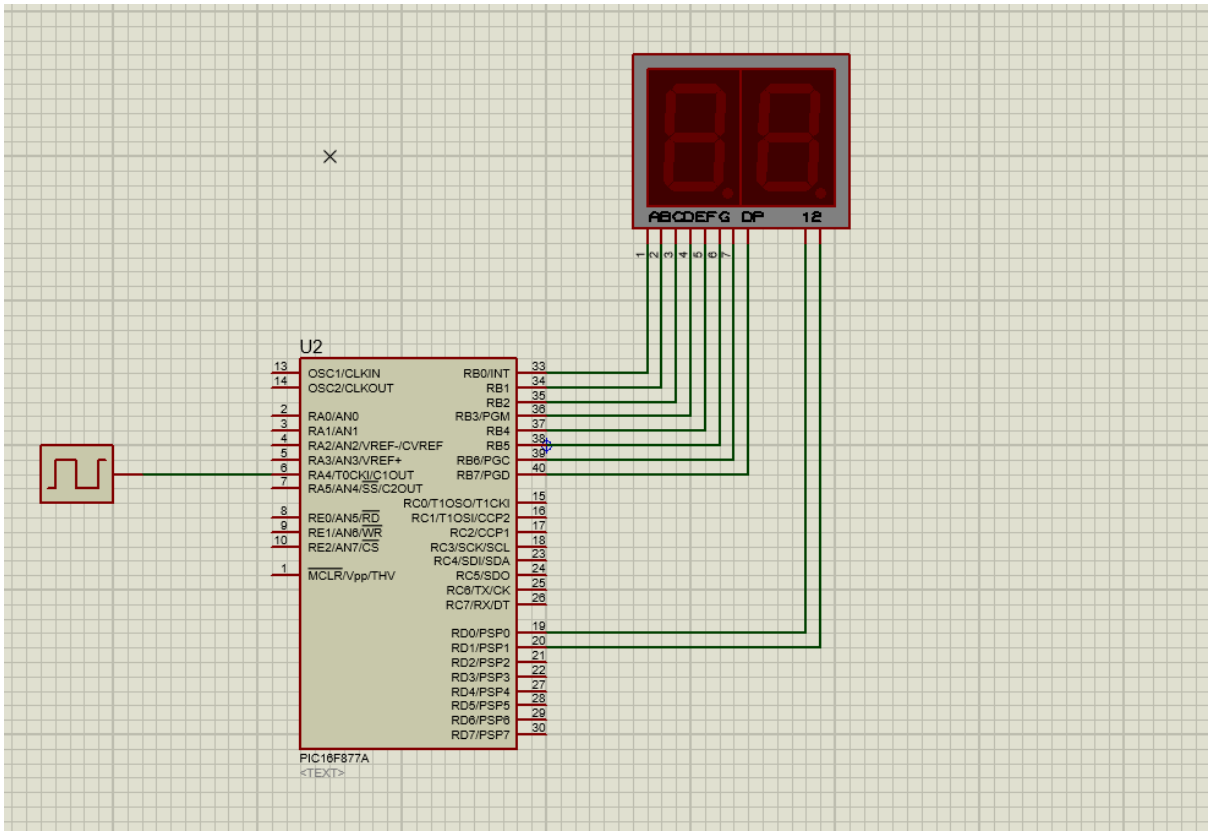
```
      RD0 =~RD0;
       INTCONbits.TMR0IF = 0;
      TMR0 = 59;     /*Load the timer Value, (Note: Timervalue is 101 instaed of 100 as the
                  TImer0 needs two instruction Cycles to start incrementing TMR0 */
          // Clear timer interrupt flag
   }
}
void timer_setup();
void main(void) {
   timer_setup();
   TRISD = 0X00;
   TMR0 = 59;
   while(1)
   {
   timer_isr();
   }

}

void timer_setup(){
   OPTION_REG = 0X07; // b? chia 256
    INTCON = 0XA0; // GIE   TOIE

}
```

COUNTER CỦA TIMER 0 PIC16F877A XC8



OPTION_REGbits.T0CS = 1; // Dem xung ngoai

OPTION_REGbits.PSA = 1;   // Su dung bo chia truoc WDT

//

```
#include <stdio.h>
#include <stdlib.h>
#define _XTAL_FREQ 8000000
#include <xc.h>
// CONFIG
#pragma config FOSC = HS        // Oscillator Selection bits (HS oscillator)
#pragma config WDTE = OFF       // Watchdog Timer Enable bit (WDT disabled)
#pragma config PWRTE = OFF      // Power-up Timer Enable bit (PWRT disabled)
#pragma config BOREN = OFF      // Brown-out Reset Enable bit (BOR disabled)
#pragma config LVP = OFF        // Low-Voltage (Single-Supply) In-Circuit Serial
Programming Enable bit (RB3 is digital I/O, HV on MCLR must be used for
programming)
#pragma config CPD = OFF        // Data EEPROM Memory Code Protection bit (Data
EEPROM code protection off)
#pragma config WRT = OFF        // Flash Program Memory Write Enable bits (Write
protection off; all program memory may be written to by EECON control)
#pragma config CP = OFF         // Flash Program Memory Code Protection bit (Code
protection off)
unsigned int dem = 0,nghin,tram,chuc,donvi;
const unsigned char maled[] = {0x3F, 0x06, 0x5B, 0x4F, 0x66, 0x6D, 0x7D, 0x07, 0x7F,
```

```
0x6F};
void display(void);
void main(void)
{
   OPTION_REGbits.T0CS = 1;          // Dem xung ngoai
   OPTION_REGbits.PSA = 1;          // Su dung bo chia truoc WDT
   TRISB = 0;
   PORTB = 0;
   TRISDbits.TRISD0 = 0;
   TRISDbits.TRISD1 = 0;
   PORTD = 0;
   while (1)
   {
      display();
      dem=TMR0;
      if(dem>=100)
      {
         TMR0=0;
         dem=0;
      }
   }
}
void display (void)
{
   chuc = dem/10;
   donvi = dem%10;

   PORTB = maled[chuc];
   RD0=0;
   __delay_ms(10);
   RD0=1;

   PORTB = maled[donvi];
   RD1=0;
   __delay_ms(10);
   RD1=1;
}
```

INTERRUPT

Trước hết ta cần cấu hình timer 0 như ví dụ trước. Và để cho phép chức năng ngắt timer 0, ta phải đặt giá trị 1 cho 2 bit T0IE và GIE ở thanh ghi INTCON để cho phép ngắt toàn cục và ngắt timer 0 ở trong hàm main:

```
void main(void)
{
    // config timer 0
    OPTION_REGbits.PSA = 0;// chon bo chia truoc cho timer 0
    OPTION_REGbits.PS2 = 0;
    OPTION_REGbits.PS1 = 1;
    OPTION_REGbits.PS0 = 0;// chon bo chia truoc 8
    OPTION_REGbits.T0CS = 0;// chon nguon xung clock noi
    INTCONbits.GIE = 1; // cho phep ngat
    INTCONbits.T0IE = 1; // cho phep ngat timer 0
    INTCONbits.T0IF = 0; // ghi gia tri co ngat = 0
    TRISC = 0;
    PORTC = 0;
    while (1);
}
```