# 

# **Documentazione PIC16F73**

Sommario

[**Documentazione PIC16F73** 1](#_Toc17814262)

[**I/O Digitale** 3](#_Toc17814263)

[**TRIS:** 3](#_Toc17814264)

[**PORT:** 3](#_Toc17814265)

[**Datatypes** 4](#_Toc17814266)

[**ADC** 4](#_Toc17814267)

[**ADCON0** 5](#_Toc17814268)

[**ADCON1** 6](#_Toc17814269)

[**ADRES** 6](#_Toc17814270)

[**TIMER0** 7](#_Toc17814271)

[**OPTION\_REG** 7](#_Toc17814272)

[**INTCON** 8](#_Toc17814273)

[**TMR0** 8](#_Toc17814274)

[**TIMER1** 9](#_Toc17814275)

[**T1CON** 9](#_Toc17814276)

[**INTCON** 10](#_Toc17814277)

[**PIE1** 11](#_Toc17814278)

[**PIR1** 12](#_Toc17814279)

[**TMR1H** 12](#_Toc17814280)

[**TMR1L** 12](#_Toc17814281)

[**TIMER2** 13](#_Toc17814282)

[**T2CON** 13](#_Toc17814283)

[**PR2** 14](#_Toc17814284)

[**CCPRxL** 15](#_Toc17814285)

[**USART** 16](#_Toc17814286)

[**TXSTA** 16](#_Toc17814287)

[**TXREG** 17](#_Toc17814288)

[**SPBRG** 17](#_Toc17814289)

[**RCSTA** 18](#_Toc17814290)

[**RCREG** 19](#_Toc17814291)

[**INTCON** 19](#_Toc17814292)

[**PIE1** 20](#_Toc17814293)

[**PIR1** 21](#_Toc17814294)

## **I/O Digitale**

### **TRIS:**

**TRISA**: determina quali porte del registro A sono ingressi e quali sono uscite (6 bit)

**TRISB**: determina quali porte del registro B sono ingressi e quali sono uscite (8 bit)

**TRISC**: determina quali porte del registro C sono ingressi e quali sono uscite (8 bit)

**Note:**

* Se il bit è a 1 è un ingresso, se è a 0 è un’uscita

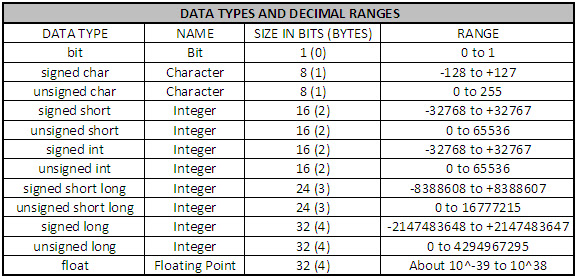
### **PORT:**

**PORTA**: si possono leggere o scrivere gli stati delle porte del registro A (6 bit)

**PORTB**: si possono leggere o scrivere gli stati delle porte del registro B (8 bit)

**PORTC**: si possono leggere o scrivere gli stati delle porte del registro C (8 bit)

## **Datatypes**



## **ADC**

Per utilizzare l’ADC si utilizzano 3 registri:

* ADCON0
* ADCON1
* ADRES

### **ADCON0**

Immagine che contiene screenshot, testo

Descrizione generata automaticamente

**Note:**

* Dopo aver selezionato il canale ed acceso l’ADC bisogna aspettare 5-10ms per dare modo al condensatore all’interno dell’ADC di caricarsi
* Dopo aver dato via alla conversione va inserita la seguente linea di codice per assicurarci che prima di eseguire ogni altra operazione la conversione sia finita:

while(GO\_nDONE==1);

* La il divisore di frequenza (ADCS) per un oscillatore da 20MHz è FOSC/32.

### **ADCON1**

Immagine che contiene screenshot

Descrizione generata automaticamente

### **ADRES**

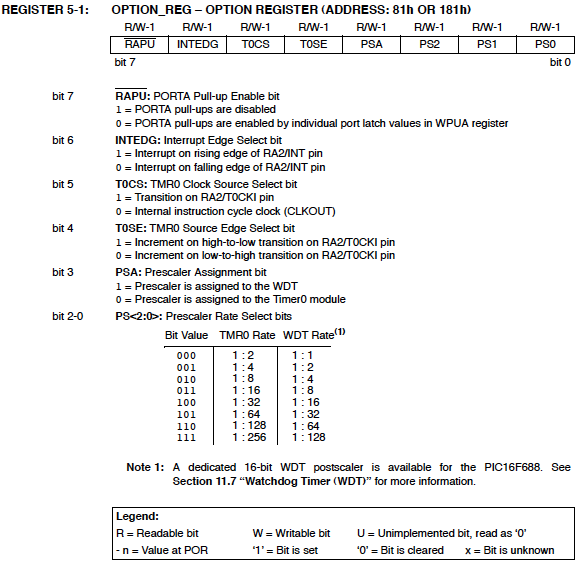
Questo registro memorizza il risultato della conversione analogico/digitale (8 bit)

## **TIMER0**

Per utilizzare il TIMER0 si utilizzano 3 registri:

* OPTION\_REG
* INTCON
* TMR0

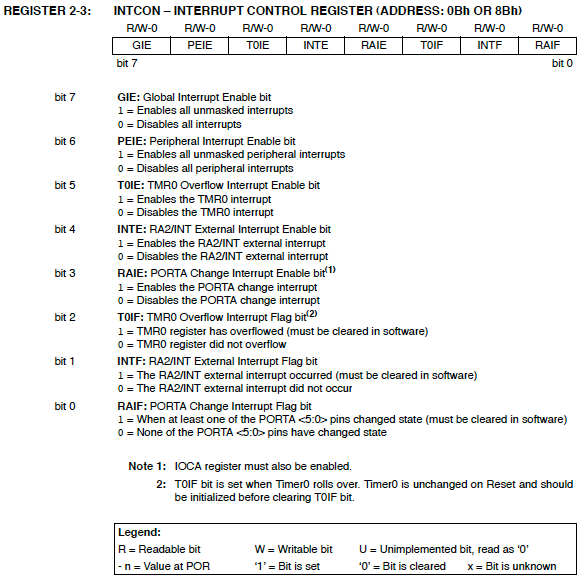
### **OPTION\_REG**



**Note:**

* Se non si desidera assegnare di prescaler conviene assegnare singolarmente i bit del registro, in tale caso in quanto l’oscillatore è sicuramente esterno conviene assegnare solamente il bit T0CS e lasciare inalterati gli altri

### **INTCON**



Questo registro serve per abilitare gli interrupt sul TIMER0. Ogni volta che il timer arriva a 255 ne lancia uno, che può essere catturato dalla funzione void \_\_interrupt() ISR(). Serve perciò abilitare T0IE(bit 5) e controllare nella routine di interrupt che T0IF sia a 1 (interrupt lanciato).

**Note:**

* Dopo la routine di interrupt T0IF va posto a 0
* Va abilitato anche il bit GIE(bit 8) per abilitare tutti gli interrupt

### **TMR0**

Contiene il valore raggiunto dal TIMER0. E’ un registro ad 8 bit perciò il valore va da 0 a 255. Viene incrementato (a meno che il prescaler non venga assegnato) ogni 4 cicli di clock, perciò ogni FOSC/4.

## **TIMER1**

Per usare il TIMER1 ci sono 6 registri:

* T1CON
* INTCON
* PIE1
* PIR1
* TMR1H
* TMR1L

### **T1CON**

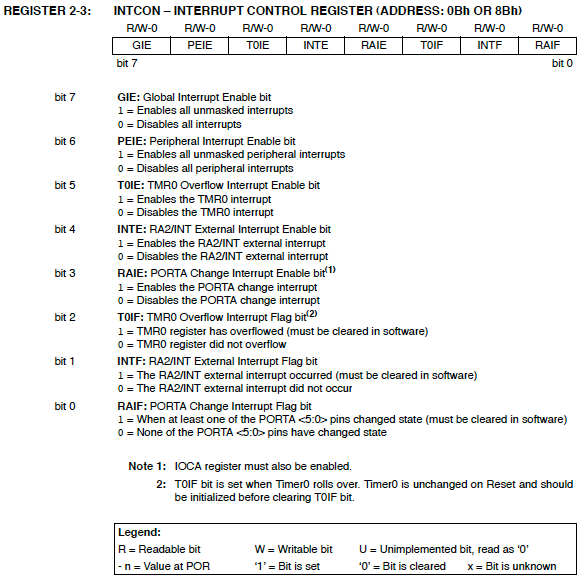
Immagine che contiene screenshot

Descrizione generata automaticamente

**Note:**

* T1OSCEN si riferisce all’oscillatore esterno
* Il conteggio parte quando TMR1ON è a 1
* Il timer è a 16 bit

### **INTCON**



Di questo registro vanno abilitati: l’interrupt globale(**GIE**) e quello sulle periferiche (**PEIE**).

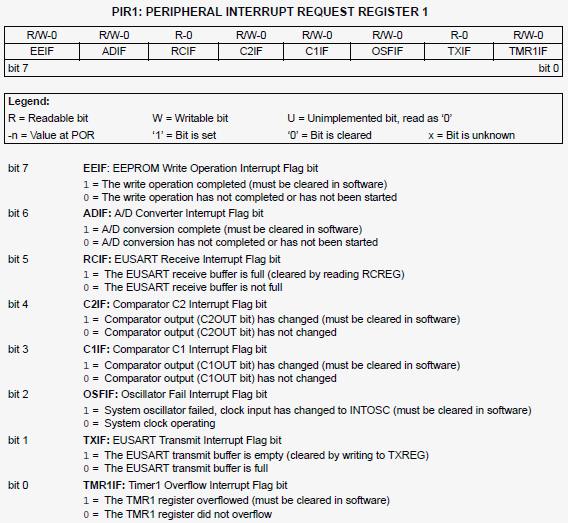
### **PIE1**

**Immagine che contiene screenshot, testo

Descrizione generata automaticamente**

Di questo registro va abilitato solamente **TMR1IE**.

### **PIR1**



Di questo registro va usato soltanto TMR1IF nella routine di interrupt. Tale bit va automaticamente a 1 quando il timer va in overflow.

### **TMR1H**

Registro ad 8 bit che contiene gli MSB del TIMER1.

### **TMR1L**

Registro ad 8 bit che contiene gli LSB del TIMER1.

**Note:**

* Per conoscere il valore totale del TIMER1 conviene usare la seguente linea di codice: (**TMR1H<<8) || TMR1L** anziché (TMR1H<<8) + TMR1L perché l’OR è più veloce della somma.

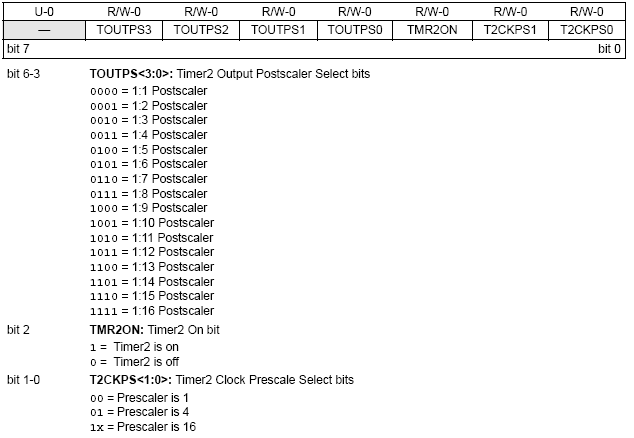
## **TIMER2**

Per utilizzare il TIMER2 (serve per generare segnali PWM sui pin CCP1 e CCP2) vi sono 4 registri:

* T2CON
* CCPxCON
* PR2
* CCPRxL

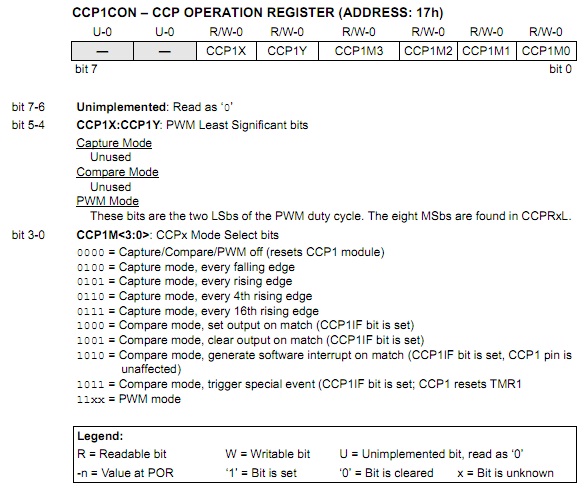
Dove la ‘x’ può essere sia 1 (segnale sul pin CCP1) che 2 (segnale sul pin CCP2)

### **T2CON**



Accende il timer ed imposta prescaler e postscaler.

**CCPxCON**



Seleziona la modalità con cui verrà usato il TIMER2 è da via al conteggio, perciò per dare il via al segnale PWM scrivo CCPxCON=0x0F

### **PR2**

Registro a 8 bit che serve per impostare la frequenza del segnale PWM.

La formula per determinare il valore è:

**Note:**

* Il prescaler deve essere regolato dimodoché il valore non superi 255, ma non deve neanche essere troppo basso

### **CCPRxL**

Serve per impostare il duty cycle. La formula per determinarne il valore è:

**Note:**

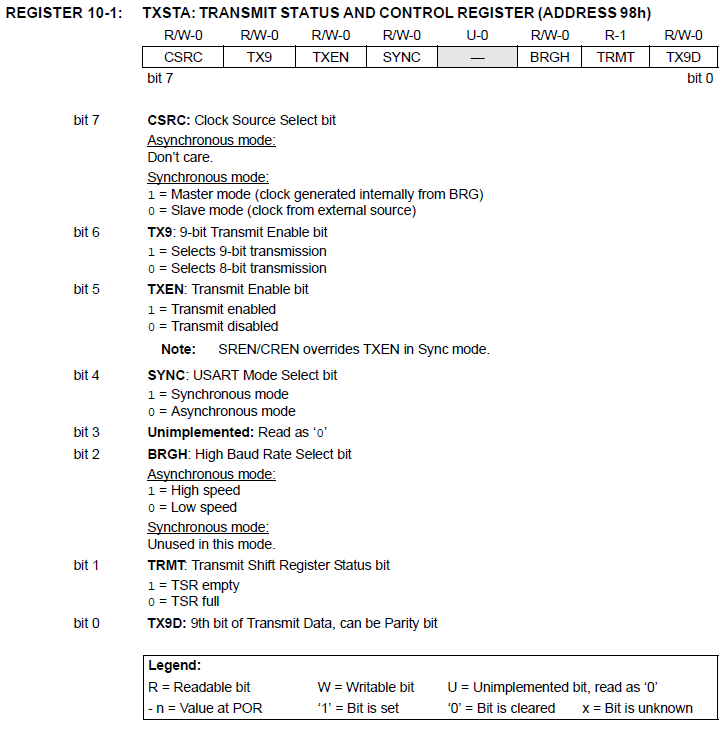
* Il DC è il DC%/100, perciò 0.25 per il 25%, 0.75 per il 75%

## **USART**

L’USART può essere usato sia in modalità sincrona (DT, CK) che asincrona (TX, RX). Vi sono 4 registri (pe l’asincrona):

* TXSTA
* TXREG
* SPBRG
* RCSTA
* RCREG
* INTCON
* PIE1
* PIR1

### **TXSTA**



**Note:**

* TRMT è a 1 quando lo shift register TSR è vuoto, perciò quando la trasmissione non è in corso.
* Per il corretto funzionamento della trasmissione abilitare anche il bit SPEN di RCSTA.

### **TXREG**

Registro a 8 bit nel quale viene memorizzato il byte da trasmettere, che poi passerà allo shift register.

**Note:**

* Quando il registro è vuoto, il bit TXIF va automaticamente a 1. Può essere catturato da una interrupt routine, tuttavia non è necessario.

### **SPBRG**

Registro a 8 bit con il quale si imposta il baud rate.

**Note:**

* Nel datasheet, alle tabelle 10-3 e 10-4, sono presenti alcuni dei valori più comuni per tale registro in funzione di FOSC, BRGH, e BAUD RATE. Per FOSC=20MHz, BRGH=1 e BAUD RATE=9600 il valore è **129**.

### **RCSTA**

Immagine che contiene screenshot, testo

Descrizione generata automaticamente

**Note:**

* Per abilitare la ricezione, porre a 1 il bit CREN.
* Per il corretto funzionamento della ricezione, impostare anche i bit SYNC e BRGH di TXSTA.

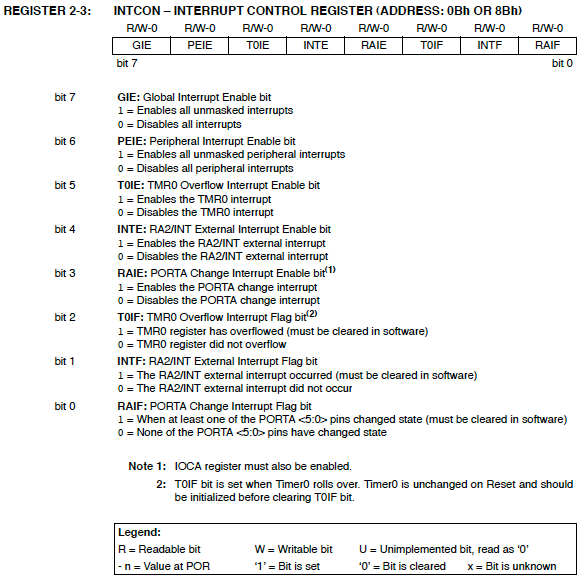
### **RCREG**

Registro a 8 bit nel quale viene memorizzato il byte ricevuto, una volta passato dallo shift register.

**Note:**

* Quando il registro è pieno, il bit RCIF va automaticamente a 1 e può essere catturato dalla routine di interrupt.

### **INTCON**



Di questo registro vanno abilitati (solo in ricezione) i bit GIE e PEIE.

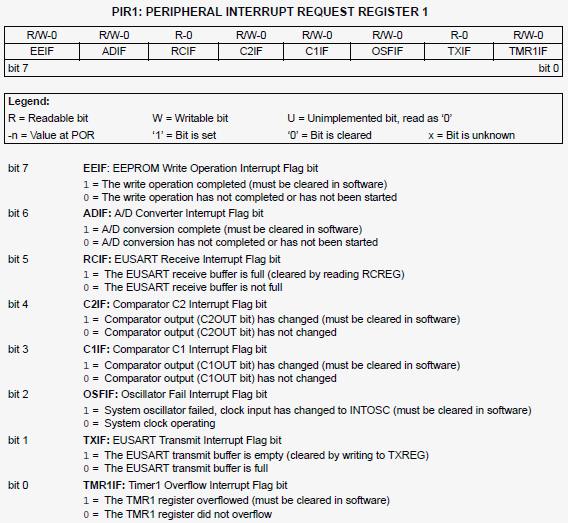
### **PIE1**

**Immagine che contiene screenshot, testo

Descrizione generata automaticamente**

L’unico bit che va attivato per la ricezione è RCIE che abilita l’interrupt di cui sopra descritto.

### **PIR1**



Di questo registro vanno usati i bit TXIF e RCIF per controllare rispettivamente:

* Se il registro TXREG è vuoto e perciò la trasmissione non è in corso (TXIF=1)
* Se il registro RCREG è pieno e perciò la ricezione è avvenuta (RCIF=1)

**Note:**

* TXIF controlla se il registro TXREG è vuoto, TRMT se lo shift register è pieno. In trasmissione il dato viene prima caricato nel registro e poi nello shift register. In ricezione al contrario: prima passa dallo shift register e poi viene caricato nel registro.
* TXIF si azzera automaticamente
* RCIF va azzerato manualmente