Il TMR2

Il TMR2 è il terzo dei temporizzatori di cui dispongono i microcontroller del tipo del PIC16F870. Ha 8 bit e funziona solo in un modo, come temporizzatore dipendente dal clock del sistema. Questa era già una funzione del TMR0, però il TMR2 conta in modo "normale", dato che non avvisa quando va in overflow, come faceva il suo predecessore, ma solo quando arriva al numero che desideriamo.

Diagramma di funzionamento

La figura riporta uno schema piuttosto preciso di come si comporta il TMR2 "al suo interno". Dato che dispone di un solo modo di funzionamento, la configurazione consiste nel regolare i valori del predivisore e del postdivisore, controllati dai bit T2CKPS1 – T2CKPS0 e TOUTPS3 – TOUTPS0 rispettivamente.

Funzionando con l'oscillatore interno, il suo conteggio si fermerà se il microcontroller entra in stato di basso consumo. Una volta attivato il temporizzatore, gli impulsi passano tramite un

T2CKPSI	T2CKPS0	RANGE DEL PREDIVISORE
0	0	1:1
0	1	1:4
1	x	1:16

Tabella dei valori del predivisore del TMR2.

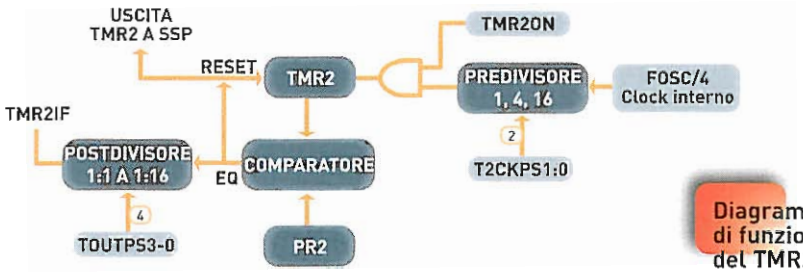


Diagramma di funzionamento del TMR2.

predivisore, questo fa sì che ogni 1, 4 o 16 cicli, il TMR2 si autoincrementi di 1. Arriverà un momento in cui il valore del TMR2 sarà uguale a quello del registro di periodo PR2, con il quale si resetterà il temporizzatore perché torni a iniziare, e questo impulso passerà al postdivisore, che attenderà X impulsi (1←=X←=16) per avvisarci impostando il flag corrispondente (TMR2IF) a 1, e producendo inoltre un interrupt, se lo avremo abilitato. Ogni volta che PR2 equivale al TMR2, si produce un evento che può attivare il modulo SSP (porta seriale sincrona). Inoltre, il TMR2 interviene nel funzionamento dei moduli CCP, ma questi sono argomenti di un capitolo successivo. Tutti i valori dovranno essere inseriti nel registro di controllo T2CON.

Il valore massimo che potremo contare con questo temporizzatore sarà Predivisore * valore – massimo – TMR2 * Postdivisore o, in altre parole, 16 * 255 * 16 = 65.280.

TOUTPS3-TOUTPS0	RANGE DEL POSTDIVISORE
0000	1:1
0001	1:2
0010	1:3
...	...
1111	1:16

Tabella dei valori del postdivisore del TMR2.

Definizione del problema

Per spiegare il funzionamento del TMR2 realizzeremo lo stesso programma fatto con il TMR0, perché è il modo migliore per notare le differenze. Simuleremo successivamente un allarme con orologio, dove l'utente inserirà un numero di ore e minuti al termine dei quali vuole essere avvisato. Questo si farà mediante interruttori e pulsanti, e servirà per dare l'ordine d'inizio del conteggio. Al termine del tempo si azionerà un cicalino per tre secondi.

–	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
7							0

Registro di controllo T2CON.

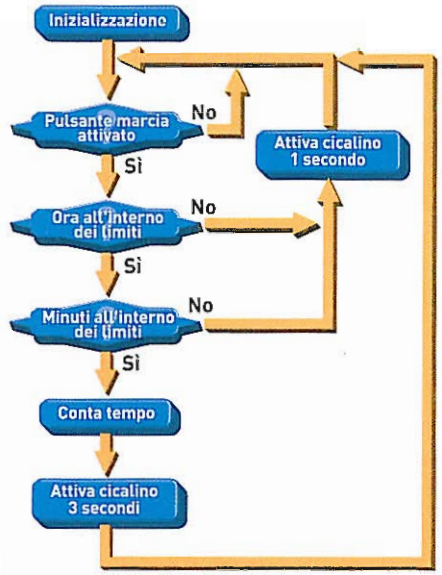
Schema elettronico

Lo schema elettronico è simile a quello già visto, però ora dobbiamo lavorare con il PIC16F87X, dato che il PIC16F84 non dispone del TMR2. Notate anche che la Porta A dispone di una linea in più, dato che ne ha 6.

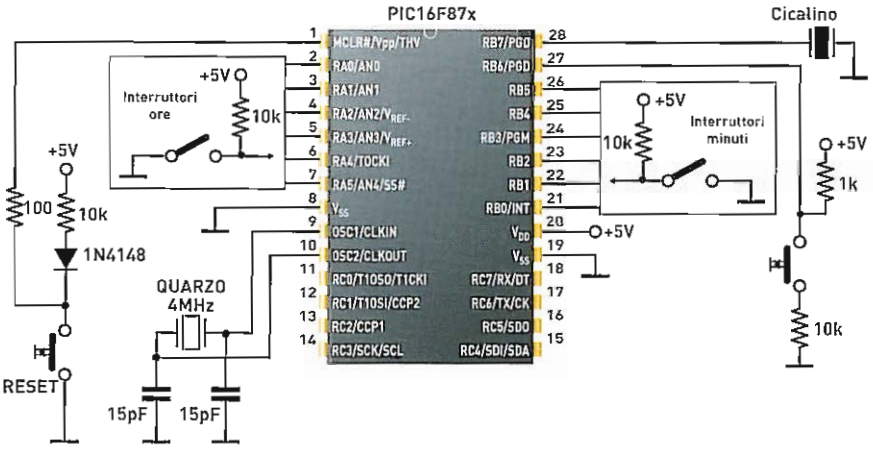
Organigramma e programma commentato

Nell'organigramma è stato necessario cambiare la prima azione dopo l'inizializzazione, perché a differenza del LetPicBasic, dove avevamo un'istruzione che controllava un impulso, in assembler questo deve essere fatto "a mano", testando il valore del pulsante fino a quando non cambia. La comparazione per determinare se le ore e i minuti sono all'interno dei limiti, si continuerà a fare, anche se risulta un po' più complicata, dato che in assembler le comparazioni non possono essere eseguite direttamente con un operatore, ma con l'unione di diverse istruzioni. "Contare il tempo" è il tema che ci interessa in questo caso. Nel LetPicBasic esisteva un'istruzione di "delay" che realizzava una temporizzazione in millisecondi, e ripetendo questa temporizzazione diverse volte si otteneva il secondo, i minuti, le ore, ecc. I cicli per contare i minuti e le ore sono simili; la differenza è nel modo di ottenere il secondo. Per calcolarlo è importante tener presente il quarzo con cui sta lavorando il PIC, dato che impone la velocità di esecuzione delle istruzioni e, quindi, la velocità di evoluzione dei temporizzatori, che seguono il suo ciclo. Quindi, per una frequenza di 4 MHz, ogni istruzione "di salto" in assembler, che ha bisogno di due cicli, impiega 2 µs, e nel resto delle istruzioni impiega un solo ciclo (1 µs). Se non applichiamo al TMR2 né il predivisore né il postdivisore, esso impiegherà 255 cicli ad andare in overflow, in altre parole, 255 µs, che è una quantità insufficiente. Tuttavia, se applichiamo i valori massimi otteniamo 65.280 µs, e se ripetiamo questo valore16 volte ci avviciniamo a sufficienza a 1 s.

Nella figura sottostante è riportato come fare questo in pseudo-codice, che si può facilmente trasformare, conoscendo le istruzioni assembler. Il predivisore e il postdivisore si impostano a 0 scrivendo il TMR2, perciò bisogna caricare i valori in questo ordine. Testando il valore del bit TMR2IF si verifica quando il registro PR2 equivale al TMR2 e quindi bisogna ricominciare un'altra volta.



Organigramma dell'esercizio proposto.



Schema elettronico dell'esercizio proposto.

```
TMR2=0
PR2=255
T2CON=011111111

Ripete 16 volte
        Ciclo 2: se TMR2IF=0
        Continuare nel ciclo 2
                TMR2IF=0
Fine Ripetizione
```

Pseudo-codice per la temporizzazione di un secondo.