Controllo dei tempi

Nella programmazione con il LetPicBasicLite e nel LetPicBasicPlus abbiamo istruzioni specifiche molto semplici per il controllo dei tempi. Lavorando con i dispositivi dei PIC abbiamo conosciuto le caratteristiche dei tre temporizzatori che possiamo utilizzare in Assembler, vediamo ora le routines specifiche per il controllo di questi temporizzatori e di quali istruzioni Assembler hanno bisogno.

I tre temporizzatori

Dei tre temporizzatori che possiedono i PIC, due di essi funzionano per overflow e il terzo segnala il raggiungimento di un valore desiderato. Questi temporizzatori possono contare eventi esterni, però ciò che a noi interessa è misurare il tempo.

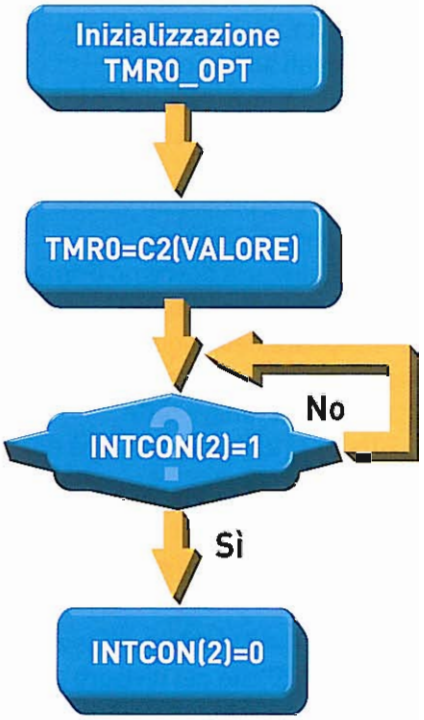
In definitiva, a parte qualche differenza, il metodo usato è simile per tutti i temporizzatori e le istruzioni sono le stesse, quindi ci baseremo su come saranno le routines per uno di questi: il TMR0.

Definizione del problema

In questo caso non si tratta di realizzare un codice che compia un enunciato, ma di programmare e verificare routines individuali che contano il tempo.

Organigramma e routines per un primo approccio

Cominceremo con il progetto più semplice. Il TMR0 si autoincrementa a ogni ciclo d'istruzione, quindi a seconda dell'oscillatore del sistema possiamo supporre che, lavorando ad esempio a 4 MHz, si incrementi ogni microsecondo. Dato che si tratta di un contatore a 8 bit, il suo valore massimo è di 255, che si può incrementare con l'utilizzo del divisore di frequenza, il quale si definisce nel registro TMR0_OPT. Per via del suo modo di funzionare, al posto del valore che vogliamo far contare, bisogna inserire il complemento a due del valore stesso. Sapremo che ha terminato di contare perché quando va in overflow, il bit 2 del registro INTCON viene impostato a 1, quindi



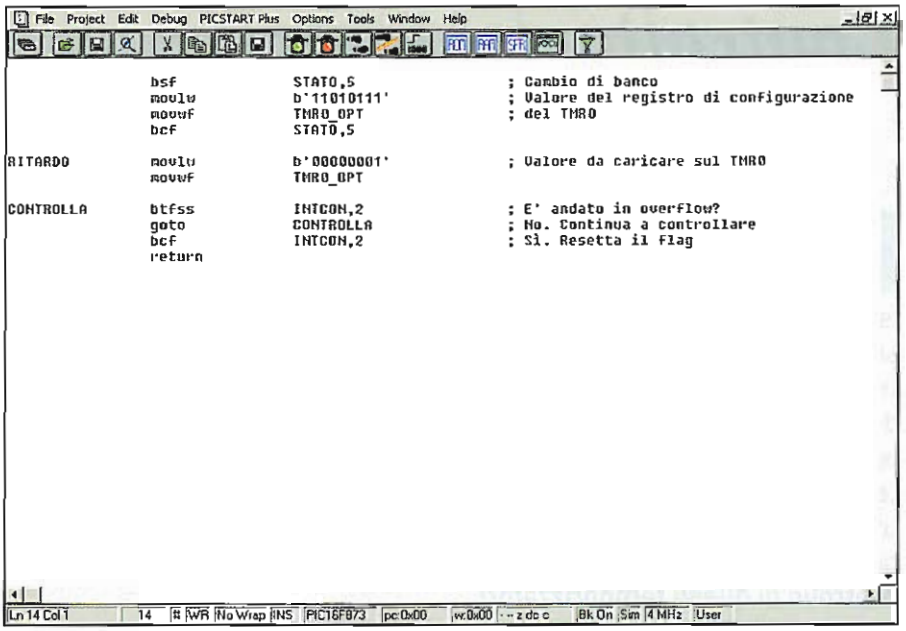
Organigramma della routine di funzionamento del TMR0.

TMR0 (8 bit)	CONTATORE (fronte di salita/discesa)	
	TEMPORIZZATORE	
TMR1 (16 bit)	CONTATORE (fronte di salita)	SINCRONO
		ASINCRONO
	TEMPORIZZATORE	
TMR2 (8 bit)	TEMPORIZZATORE	

Ci baseremo sulle routines costruite per il TMR0 come temporizzatore, dato che per i rimanenti casi si procede in modo simile.

noi potremo rimanere all'interno del ciclo sino a quando questo succede. Questo bit dovrà essere nuovamente impostato a 0, affinché la volta successiva che si utilizza la routine di tempo non parta a valore 1 già dall'inizio. I passaggi commentati sono riportati nell'organigramma della figura. Se trasformiamo questi passaggi in istruzioni Assembler, otteniamo la routine che possiamo vedere nell'immagine a fianco.

Per utilizzarla dovremo prima configurare il registro OPTION così come appare nella figura, dopo di che potremo "chiamare" la routine. Come potete vedere tutte le istruzioni utilizzate sono già note.

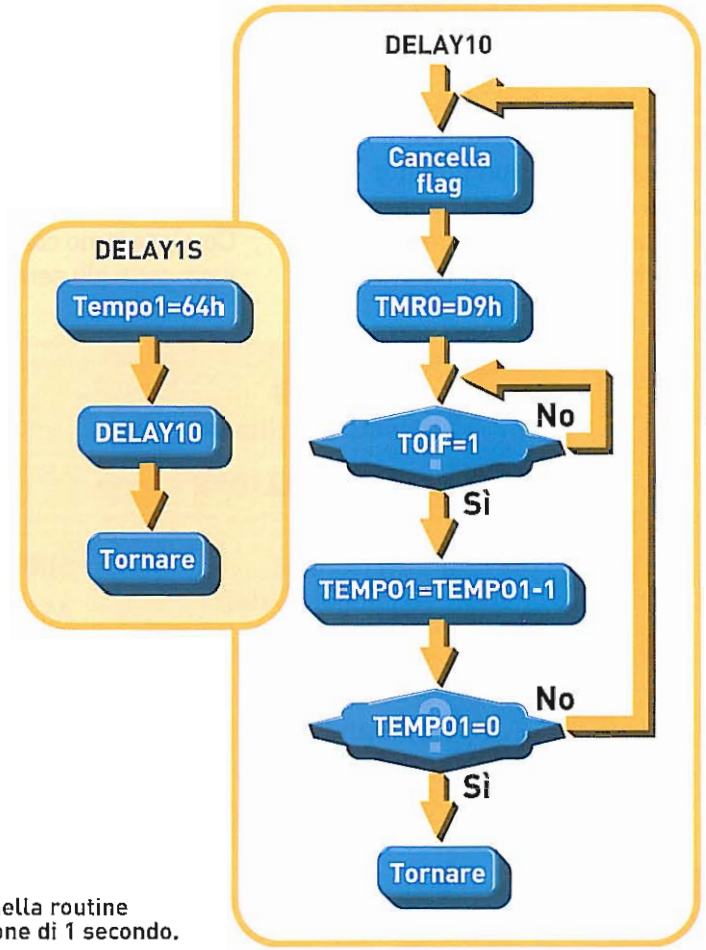


Routine di temporizzazione con il TMR0.

Secondo organigramma

Pur impostando i valori massimi, sia nel TMR0 che nel divisore di frequenza, la temporizzazione che si ottiene è di pochi millisecondi. Quando sono richieste temporizzazioni maggiori è necessario realizzare un ciclo in cui il TMR0 vada in overflow diverse volte, inoltre potrebbe essere necessario annidare un ciclo all'interno di un altro.

I due organigrammi della figura mostrano come realizzare una temporizzazione da 1 secondo a partire da una routine di temporizzazione di 10 ms. Potremmo inserire tutto all'interno della stessa routine, però in questo modo, cambiando il valore impostato a TEMPO1 nella routine più esterna, è possibile ottenere diverse temporizzazioni. Per la sua implementazione in codice possiamo utilizzare istruzioni già viste, però ne utilizzeremo alcune nuove che facilitano il lavoro.



Organigrammi della routine di temporizzazione di 1 secondo.