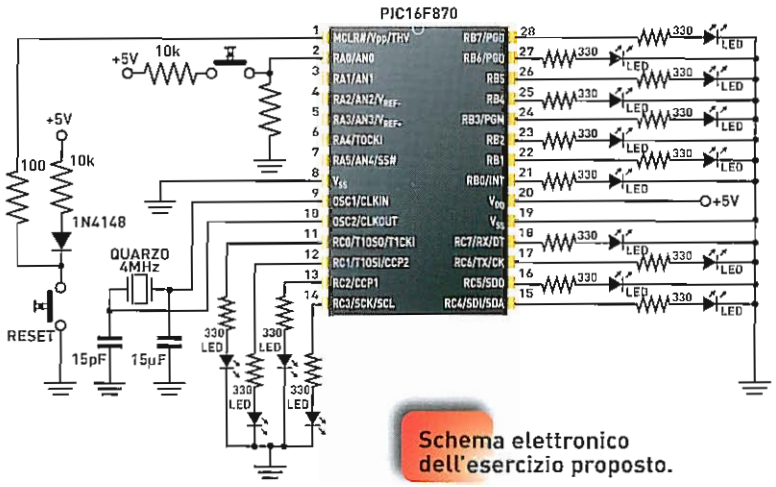
Il TMR1 come temporizzatore

Tutti i programmi e le istruzioni con cui abbiamo lavorato sinora fanno riferimento al PIC16F84, che ha delle risorse limitate per cui si è reso necessario passare a un microcontroller più avanzato come il caso del PIC16F870. Fra le altre cose questo PIC non possiede solo un temporizzatore come il PIC16F84, ma ne ha tre: TMR0, TMR1 e TMR2.



Schema elettronico dell'esercizio proposto.

Differenza fra i temporizzatori

Il TMR0 del PIC16F84 è un contatore a 8 bit che può funzionare in due modi: come temporizzatore e come contatore di impulsi.

Il PIC16F870 dispone anch'esso del TMR0 che funziona nello stesso modo ed è identico a quello del PIC16F84. Per cui un programma fatto per questo funziona anche per l'altro, inoltre dispone anche di un secondo temporizzatore, il TMR1 e anche di un terzo il TMR2. Il TMR1 è quello che fornisce più possibilità, e in questo capitolo lavoreremo con una delle sue modalità: come temporizzatore.

Definizione del problema

Si vuole misurare il tempo che passa fra due impulsi provenienti da una tastiera. Il risultato sarà riportato su diodi LED. La temporizzazione si farà utilizzando il TMR1.

Schema elettronico

Tenendo conto che il PIC da utilizzare ha 28 pin, le porte di ingresso/uscita diventano tre: PORTA, PORTB e PORTC, quindi il numero dei LED per mostrare il risultato può essere elevato.

Si possono utilizzare le porte B e C per mostrare la temporizzazione, e una delle linee della Porta A può diventare quella che supporta il pulsante per iniziare e fermare il conteggio. Lo schema elettronico è riportato nella figura. Il pulsante e i LED sono stati collegati a livello alto, quindi il pulsante porterà un livello 1 al momento dell'attivazione (seguito da uno 0 al rilascio) e i LED avranno bisogno di livelli 1 per essere attivati. Se il pulsante si collega al livello basso, il modo di fare il programma sarà simile, però nel caso dei LED il loro collegamento per il livello alto è più immediato, dato che semplifica la visualizzazione del valore ottenuto senza doverlo modificare con un passaggio successivo.

Organigramma dell'esercizio proposto.

–	–	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC#	TMR1CS	TMR1ON
7				T1CON			0

- T1CKPS1-0: 11. Valore massimo del divisore di frequenza (1/8)
- T1OSCEN,T1SYNC#:00. Perdono significato se il TMR1 diventa temporizzatore con clock interno
- TMR1CS:0. Utilizzo del clock interno per temporizzazioni
- TMR1ON:0. Temporizzatore scollegato. Dovremo impostarlo a 1 quando si preme il pulsante di marcia

Valori da inserire in T1CON.

Organigramma

L'organigramma è piuttosto semplice da organizzare, anche se è di poco aiuto alla successiva codificazione del programma, perché non esiste in Basic (il linguaggio che per il momento conosciamo), alcuna istruzione per lavorare con il TMR1.

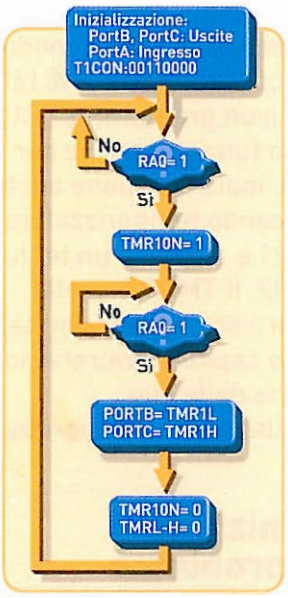
Realizzazione del programma

Come abbiamo detto, questo programma non può essere fatto in Basic e le istruzioni assembler sono ancora sconosciute; cerchiamo di fare una prima dichiarazione delle cose da fare all'interno del programma per far sì che, più avanti, voi stessi siate capaci di trasformarlo in codice. Il riquadro di inizializzazione rappresenta tutte le dichiarazioni e le definizioni che bisogna fare prima di iniziare con il programma in sé. Si definirà il microcontroller da utilizzare e si dichiareranno le variabili (registri generali) così come i registri specifici. Le porte saranno definite come ingresso o uscita.

Nel nostro caso la porta B e la porta C diventeranno uscite e la porta A sarà di ingresso. Questi microcontroller hanno il convertitore analogico digitale e i suoi canali appartengono alla porta A, perciò bisognerà configurare questa come digitale per il nostro pulsante, dato che per default alla partenza del programma è analogica. L'assembler esige anche che siano configurati i modi di lavoro dei temporizzatori, quindi dovremo inserire un valore nel registro T1CON. Dobbiamo attendere un impulso. Se nel Basic questo si faceva con una sola istruzione (button) dalla quale non si usciva sino a che non si aveva il valore ricercato, in assembler bisogna fare un ciclo per acquisire questo valore, continuando solo dopo che il valore viene prodotto.

Il concetto di programma si amplia quando si lavora in assembler. Come avrete notato in Basic le istruzioni vengono eseguite una a una in modo che, ad esempio, un lasso di tempo si può contare fra due istruzioni, ma non mentre sono eseguite altre istruzioni. In assembler, come si può vedere nell'organigramma, si può iniziare "a contare il tempo" e passare all'istruzione successiva mentre il temporizzatore continua a fare il suo lavoro. Questo lavoro consisterà nell'autoincrementarsi ogni X cicli di istruzione, dove X sarà il valore definito dal divisore di frequenza, 8 nel nostro esempio. Trattandosi di un temporizzatore da 16 bit il valore massimo che si potrà contare sarà di 216, che non è molto dato che un ciclo in un PIC a 4 MHz è di 1 ms. Arrivando al valore massimo tornerà a 0 e continuerà a contare, quindi se noi vogliamo avere valori maggiori dovremo incrementare un'altra variabile ad ogni "giro".

Dopo aver messo in marcia il temporizzatore, bisogna entrare in un altro ciclo, per attendere l'attivazione del pulsante che serve sia a mettere in marcia il tempo che a fermarlo. I 16 bit del temporizzatore possono essere letti direttamente dai loro registri TMR1L (parte meno significativa del dato) e TMR1H (parte più significativa del dato) ed essere passati alle porte B e C. Se si vuole ripetere il ciclo bisogna scollegare il temporizzatore e caricare nuovamente il valore 0. Nella figura è riportato un organigramma più vicino al programma assembler.



Organigramma dell'esercizio proposto per la sua codificazione in assembler.