Il TMR0 come temporizzatore

Abbiamo visto l'importanza che hanno i temporizzatori nei programmi quando si lavora in Basic. Praticamente in qualsiasi programma un po' complesso è necessario inserire qualche ritardo, per questo o quell'altro motivo. Il Basic ci forniva tre istruzioni differenti per realizzare delle temporizzazioni, fra le quali sceglieremo quella più adatta per realizzare il programma.

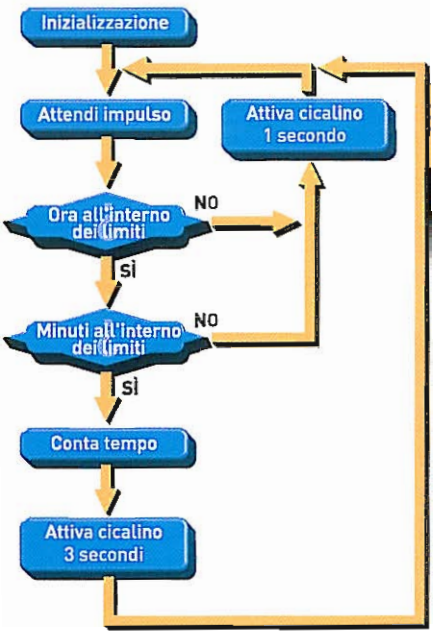
Definizione del problema

Simuleremo un allarme da orologio. L'utente avrà il compito, mediante interruttori, di inserire il numero di ore e minuti al termine dei quali vuole essere avvisato. Infine, l'ordine di iniziare il conteggio verrà dato con un pulsante. Al termine del tempo si attiverà un cicalino per la durata di tre secondi.

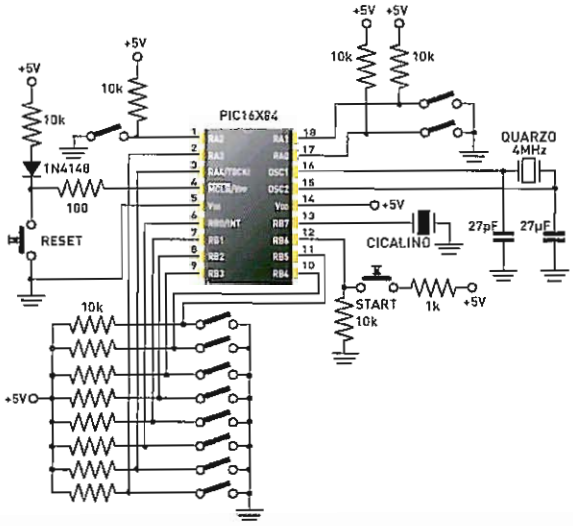
Schema elettronico

Per conoscere il numero degli interruttori che saranno necessari, dobbiamo prendere in considerazione il formato delle ore e dei minuti.

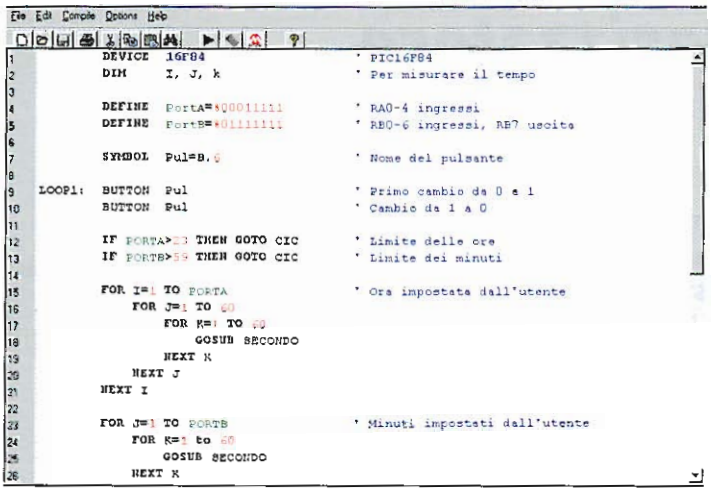
Le ore possono arrivare a 24, da 0 a 23, e i minuti a 60, da 0 a 59. Per le ore, quindi, saranno sufficienti 5 interruttori, per i minuti ne avremo bisogno di 6, in entrambi i casi conteremo in binario. In questo modo, supponendo di lavorare con un PIC16F84, potremo dedicare la porta A agli interruttori per le ore, e la porta B agli interruttori dei minuti, al pulsante d'inizio e al cicalino. Lo schema elettronico risultante è riportato nella figura. Come sempre, abbiamo cinque piedini occupati con gli elementi fissi: alimentazione, clock e pulsante di reset. Gli interruttori sono stati collegati per livello basso, anche se dal punto di vista della programmazione questo non è importante, poiché dovremo prendere in considerazione solamente il dato che viene acquisito; saranno gli interruttori a dover essere etichettati per fare in modo che l'utente non cada in errore, e imposti un codice per livello alto. Sia il pulsante che il cicalino saranno collegati per livello alto, e questo dovrà essere tenuto nella debita considerazione al momento di programmare.



Organigramma dell'esercizio proposto.



Schema elettronico dell'esercizio proposto.



```
File Edit Compile Options Help
        DEVICE  16F84                   ' PIC16F84
        DIM     I, J, k                 ' Per misurare il tempo

        DEFINE  PortA=%00011111         ' RA0-4 ingressi
        DEFINE  PortB=%01111111         ' RB0-6 ingressi, RB7 uscita

        SYMBOL  Pul=B.0                 ' Nome del pulsante

LOOP1:  BUTTON  Pul                     ' Primo cambio da 0 a 1
        BUTTON  Pul                     ' Cambio da 1 a 0

        IF PORTA>23 THEN GOTO CIC       ' Limite delle ore
        IF PORTB>59 THEN GOTO CIC       ' Limite dei minuti

        FOR I=1 TO PORTA                ' Ora impostata dall'utente
            FOR J=1 TO 60
                FOR K=1 TO 60
                    GOSUB SECONDO
                NEXT K
            NEXT J
        NEXT I

        FOR J=1 TO PORTB                ' Minuti impostati dall'utente
            FOR K=1 to 60
                GOSUB SECONDO
            NEXT K
```

Prima parte del programma dell'esercizio proposto.

Organigramma

L'idea è stata plasmata nell'organigramma della figura della pagina precedente. Notate che, anche se l'enunciato non lo specifica, è necessario verificare che il valore introdotto tramite gli interruttori, sia per le ore sia per i minuti, si trovi all'interno dei limiti, in caso contrario bisogna avvisare l'utente. Un modo semplice di generare questo avviso, è attivare il cicalino con un impulso di durata inferiore rispetto al caso in cui tutto è corretto.

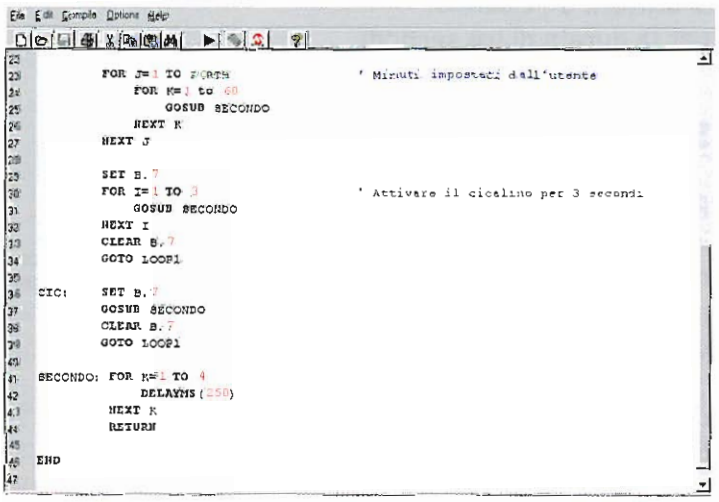
Programma commentato

Le figure riportano il programma diviso in due videate. Nella prima si specifica il modello di PIC, le variabili che si vogliono utilizzare, e si definiscono le porte come ingressi/uscite, così come abbiamo spiegato nello schema elettronico. Dovremo quindi attendere un impulso per iniziare a conteggiare il tempo. Dato che si utilizza l'istruzione "button", è necessario eseguirla due volte per fare in modo che il pulsante si attivi e si disattivi. Se le ore o i minuti non sono all'interno dei limiti ci sarà una subroutine per attivare il cicalino per un secondo, e tornare al ciclo principale. Se i valori sono corretti si passa a contare il tempo. Il modo scelto è l'annidamento di due cicli. Quello esterno è il ciclo delle ore, che parte da 1 e arriva al limite impostato dall'utente, e quello interno è il ciclo dei minuti contenuti nell'ora, il cui valore è sempre 60. Al centro del ciclo viene chiamata una subroutine che realizza una temporizzazione di un secondo. Con un terzo ciclo vengono conteggiati i minuti che l'utente ha impostato. Nella seconda videata del programma si vedono le routines utilizzate. Nella prima parte si attiva il cicalino per tre secondi, per avvisare che è terminata la temporizzazione e si ritorna al ciclo principale. Il ciclo seguente è la routine che attiva il cicalino in caso di errore per superamento dei limiti. Entrambi fanno uso della routine "secondo", che è basata sull'istruzione DELAYMS, con una durata di 250 millisecondi, che viene ripetuta quattro volte per ottenere un secondo.

Modifiche al programma proposto

Vi proponiamo due modifiche al programma. Avete pensato a cosa potrebbe succedere se l'utente introduce 0 ore o 0 minuti? Il programma non è pronto per questo, e la variante non si risolve cambiando semplicemente i limiti dei cicli. Abbiamo utilizzato l'istruzione "delayms" perché era la più semplice in questo caso.

Provate a sostituirla con le altre due che abbiamo visto a suo tempo per realizzare le temporizzazioni.



```
File Edit Compile Options Help
        FOR J=1 TO PORTB                ' Minuti impostati dall'utente
            FOR K=1 to 60
                GOSUB SECONDO
            NEXT K
        NEXT J

        SET B.7
        FOR I=1 TO 3                    ' Attivare il cicalino per 3 secondi
            GOSUB SECONDO
        NEXT I
        CLEAR B.7
        GOTO LOOP1

CIC:    SET B.7
        GOSUB SECONDO
        CLEAR B.7
        GOTO LOOP1

SECONDO: FOR K=1 TO 4
             DELAYMS (250)
         NEXT K
         RETURN

END
```

Seconda parte del programma dell'esercizio proposto.