Esercizi di apprendimento

```
;
;Macchina di confezionamento. Eliminazione dei rimbalzi mediante una temporizzazione
;
;Due relè "H1" (RB0) e "H2" (RB1), comandano due motori che fanno muovere due nastri
;trasportatori. "H1" (RB0) trasporta pezzi e "H2" (RB1) imballaggi. Un sensore "DP" (RB1)
;rileva il passaggio dei pezzi e l'altro, "DE" (RB2), rileva il corretto posizionamento
;dell'imballo. Dopo che sono passati 10 pezzi, la confezione si considera piena, si attiva
;un segnale acustico, "A" (RB2), e il nastro che trasporta gli imballaggi si attiva, sino
;a posizionare un nuovo imballo vuoto. In questo momento si disattiva il segnale acustico
;("0") (RB2), e avanza  nuovamente il nastro dei pezzi ripetendo così il ciclo. Un interruttore
;"I" (RC0) attiva o disattiva l'intero sistema.

                List    p=16F870                ;Tipo di processore
                include "P16F870.INC"   ;Definizione dei registri interni

Conta_pezzi     equ     0x20            ;Variabile N° di pezzi

                org     0x00

Inizio          clrf    PORTB           ;Cancella i latch di uscita
                bsf     STATUS,RP0      ;Seleziona banco 1
                clrf    TRISB           ;Configura la Porta B come uscita
                movlw   0xFF
                movwf   TRISC           ;Configura la Porta C come ingresso
                movlw   b'00000110'
                movwf   OPTION_REG      ;Prescaler di 128 per il TMR0
                bcf     STATUS,RP0      ;Seleziona banco 0
```

L'esercizio che vi mostriamo si trova sul CDROM sotto il nome di es16.asm.
È un esercizio sequenziale, simile al programma es15.asm commentato in precedenza.
Questo esercizio simula una macchina di confezionamento, in cui abbiamo una serie di pezzi che si muovono su due nastri trasportatori. Quando vengono inseriti 10 pezzi all'interno del contenitore, questo si considera pieno e si completa il ciclo. Come abbiamo potuto verificare con il programma es15.asm, a causa dell'effetto rimbalzo dei sensori meccanici, anche se l'interruttore viene azionato meno di 10 volte, il programma considera che il contenitore sia pieno.

```
Loop_0          clrf    PORTB           ;Resetta le uscite
Loop:           clrwdt                  ;Aggiorna il WDT
                movlw   d'10'
                movwf   Conta_pezzi     ;Inizializza la variabile N° di pezzi
                btfss   PORTC,0         ;Testa lo stato dell'interruttore I
                goto    Loop_0          ;È su OFF, sistema fermo

                call    Delay_20_ms     ;Elimina i rimbalzi

                bcf     PORTB,0         ;Avanzamento dei pezzi a OFF
                bsf     PORTB,1         ;Avanzamento dei contenitori a ON

No_contenitore  clrwdt                  ;Aggiorna il WDT
                btfss   PORTC,2         ;Contenitore in posizione?
                goto    No_contenitore  ;No.

                call    Delay_20_ms     ;Elimina i rimbalzi

                bsf     PORTB,0         ;Sì. Avanzamento dei pezzi a ON
                bcf     PORTB,1         ;Avanzamento dei contenitori a OFF
                bcf     PORTB,2         ;Cicalino a OFF
```

Tutti i sensori meccanici sono soggetti all'effetto rimbalzo, il quale consiste in uno stato transitorio fra "0" e "1" che si manifesta all'uscita del sensore ogni qual volta il sensore cambi stato. A causa della velocità di elaborazione del microcontroller, è possibile leggere questi valori di transizione, che possono provocare errori nel programma. I sensori meccanici tipo finecorsa di Pathfinder sono soggetti agli stessi rimbalzi degli interruttori della scheda di ingressi e uscite che stiamo manipolando ora.

```
;Questa sequenza di istruzioni attende che il pezzo passi completamente
;davanti al sensore "DP"

Pezzi_0         clrwdt                  ;Aggiorna il WDT
                btfsc   PORTC,1         ;È arrivato il pezzo?
                goto    Pezzi_0         ;Non ancora

                call    Delay_20_ms     ;Elimina i rimbalzi

Pezzi_1         clrwdt                  ;Ora sì. Aggiorna il WDT
                btfsc   PORTC,1         ;Il pezzo ha completato il passaggio davanti al sensore "DP"?
                goto    Pezzi_1         ;Non ancora

                call    Delay_20_ms     ;Elimina i rimbalzi

                decfsz  Conta_pezzi,F   ;Ora sì. Decrementa il contatore dei pezzi

                goto    Pezzi_0         ;Attendi il passaggio di un nuovo pezzo

                bcf     PORTB,0         ;Sono passati 10 pezzi, avanzamento dei pezzi a OFF
                bsf     PORTB,1         ;Avanzamento dei contenitori a ON
                bsf     PORTB,2         ;Cicalino a ON

Si_contenitore  clrwdt                  ;Aggiorna il WDT
                btfsc   PORTC,2         ;Il contenitore in posizione è sempre quello di prima?
                goto    Si_contenitore  ;Per ora sì

                call    Delay_20_ms

                goto    Loop            ;Ora no
```

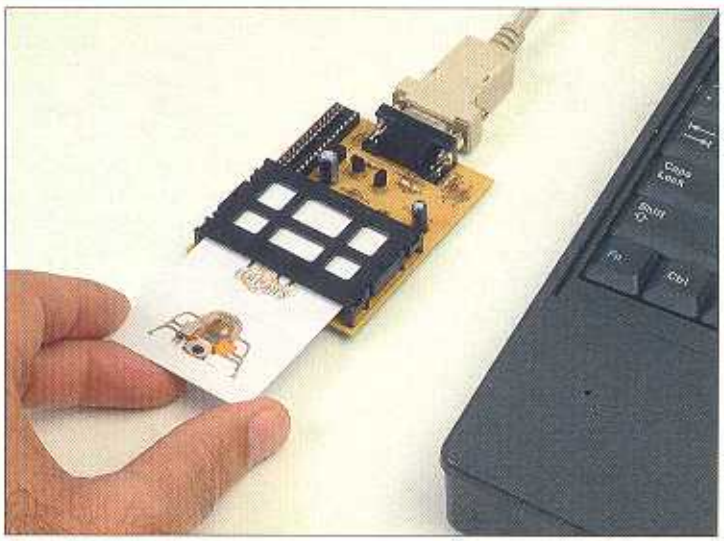
Per correggere l'effetto rimbalzo possiamo utilizzare una soluzione software, ovvero realizzare una temporizzazione di 20 ms ogni volta che viene rilevato il primo cambio di stato di un sensore meccanico. Trascorso il periodo di 20 ms, l'uscita del sensore sarà già stabilizzata sul nuovo valore. Si può verificare che il codice sorgente del programma es16.asm sia simile a quello del file es15.asm, però dopo ogni istruzione di salto che testa lo stato di un interruttore, viene eseguita una chiamata a una funzione, che eseguirà una temporizzazione di 20 ms.

```
;Delay_20_ms è una routine che realizza una temporizzazione di 20 ms, con l'obiettivo
;di eliminare "l'effetto rimbalzo", caratteristico dei componenti elettromeccanici.
;
;Se il PIC lavora ad una frequenza di 4 MHz, il TMR0 si aggiorna ogni µS. Se si desidera
; temporizzare 20000 µS (20 ms) con un prescaler di 128, il TMR0 dovrà contare 156 eventi.
;(156 * 128 = 19968). Il valore 156 equivale a 0x9c e, dato che  il TMR0 è ascendente,
;lo dovremo caricare con il suo complemento (0x63)

Delay_20_ms:    bcf     INTCON,T0IF     ;Resetta il flag di overflow del TMR0
                movlw   0x63
                movwf   TMR0            ;carica il TMR0
Delay_20_ms_1   clrwdt                  ;Aggiorna il WDT
                btfss   INTCON,T0IF     ;Overflow del TMR0?
                goto    Delay_20_ms_1   ;Non ancora
                return                  ;Ora sì


                end                     ;Fine del programma sorgente
```

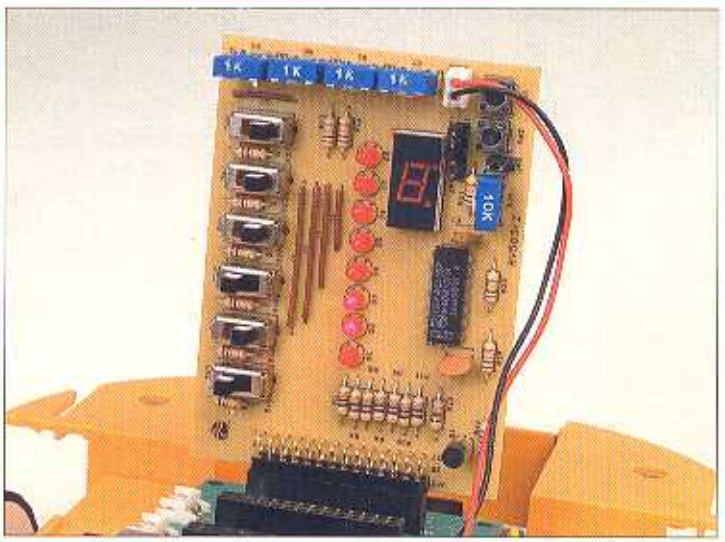
Questa è la routine di temporizzazione. È stato utilizzato il Timer 0 del microcontroller, che è un temporizzatore interno da 8 bit. Bisogna notare che all'inizio del programma, nel ciclo Inizio, è stato anche configurato il registro Option tramite il quale assegniamo un prescaler di 128 al Timer 0, quindi i suoi incrementi saranno 128 volte più lenti della sua velocità nominale. Per calcolare il valore del tempo che impiega il Timer 0 bisogna applicare la seguente formula:

Tempo_attesa (s) = (4 / frequenza_lavoro) * n°_incrementi_Timer0 * prescaler.

Nei commenti del programma sono evidenziati i calcoli realizzati e i valori inseriti per ogni parametro.



Dopo aver compilato il programma con MPLAB e ottenuto il file es16.hex, lo scriveremo sulla Smartcard. Come in tutti gli esercizi, utilizzeremo il programma ICPROG, selezionando il dispositivo 24C16. Collegheremo la scheda di scrittura alla porta seriale del PC, apriremo il file es16.hex e procederemo alla scrittura della Smartcard. Dopo aver caricato il programma, estrarremo la Smartcard e la inseriremo nella scheda di alimentazione di Pathfinder con l'orientamento adeguato.



Per provare l'esercizio, il jumper JP1 che attiva i LED della scheda di ingressi e uscite deve essere chiuso. Gli altri due jumper della scheda rimarranno aperti. Dopo aver alimentato il robot, il microcontroller impiegherà alcuni secondi a leggere la Smartcard e a iniziare l'esecuzione del programma. A questo punto verificheremo che il risultato dell'esercizio corrisponda fedelmente a quanto enunciato: