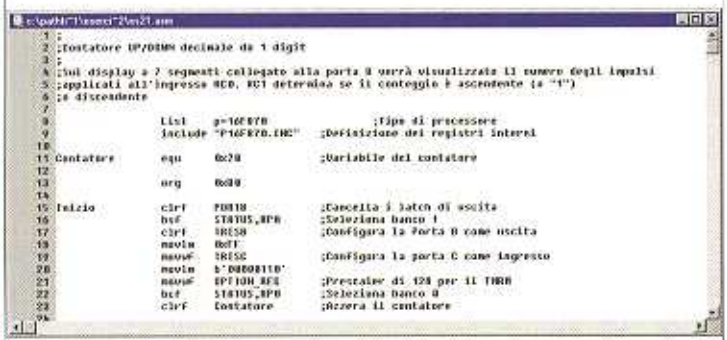
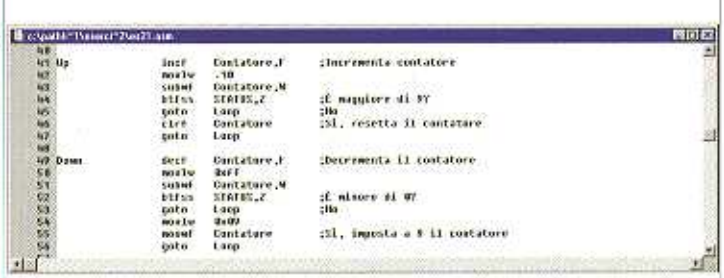


```
;
;Contatore UP/DOWN decimale da 1 digit
;
;sul display a 7 segmenti collegato alla porta B verrà visualizzato il numero degli impulsi
;applicati all'ingresso RC0. RC1 determina se il conteggio è ascendente (a "1")
;o discendente

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

Contatore       equ     0x20            ;Variabile del contatore

                org     0x00

Inizio          clrf    PORTB           ;Cancella il 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
                clrf    Contatore       ;Azzera il contatore
```

Questo è un ampliamento dell'esercizio dei contatori es20.asm, spiegato in precedenza. Si tratta di implementare un contatore ascendente e discendente, che però visualizzi il numero sul display a sette segmenti. Questo esercizio introdurrà due novità, in primo luogo l'utilizzo del display invece della barra dei diodi LED, e poi la necessità di implementare delle maschere, perché il numero da visualizzare dovrà essere compreso fra 0 e 9.

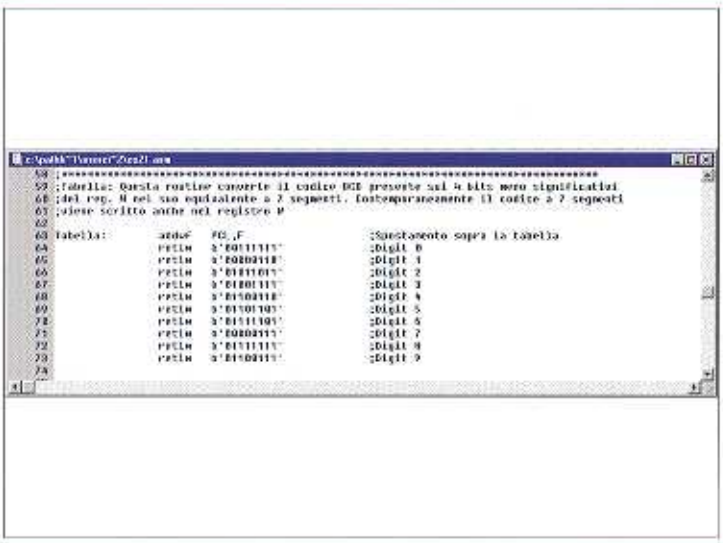
```
Loop            movf    Contatore,W
                call    Tabella         ;Converte BCD a 7 segmenti
                movwf   PORTB           ;Visualizza il valore del contatore
Wait_0          clrwdt                  ;Aggiorna il WDT
                btfsc   PORTC,0         ;RC0 = 1?
                goto    Wait_0          ;No
                call    Delay_20_ms     ;Elimina rimbalzi

Wait_1          clrwdt                  ;Aggiorna il WDT
                btfss   PORTC,0         ;RC0 = 0 (impulso)?
                goto    Wait_1          ;No
                call    Delay_20_ms     ;C'è un impulso, eliminare i rimbalzi

                btfss   PORTC,1         ;RC1 = 1
                goto    Down            ;No, conteggio discendente
```

Il ciclo principale del programma è abbastanza simile a quello del contatore binario che abbiamo già realizzato. Abbiamo una variabile chiamata Contatore che contiene il valore del contatore e che deve essere visualizzata sul display a 7 segmenti. Esistono inoltre due cicli per attendere che il segnale RC0 passi a valore '1' e dopodiché torni a '0', completando il ciclo dell'impulso. Dopo aver inserito un impulso con RC0 si testa il pin RC1 per decidere se eseguire un conteggio ascendente o discendente.



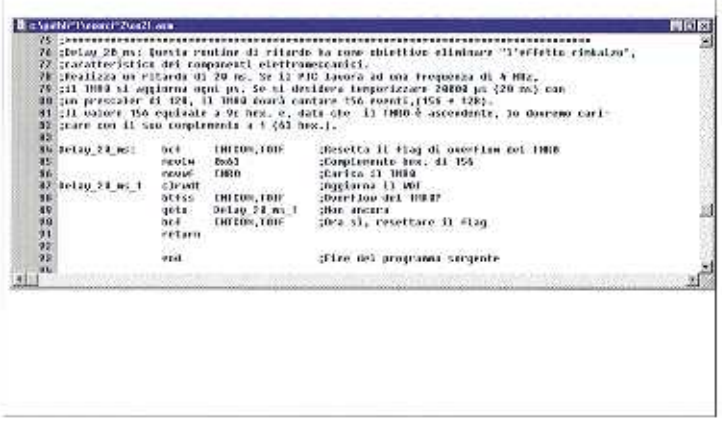
```
Up              incf    Contatore,F     ;Incrementa contatore
                movlw   .10
                subwf   Contatore,W
                btfss   STATUS,Z        ;È maggiore di 9?
                goto    Loop            ;No
                clrf    Contatore       ;Sì, resetta il contatore
                goto    Loop

Down            decf    Contatore,F     ;Decrementa il contatore
                movlw   0xFF
                subwf   Contatore,W
                btfss   STATUS,Z        ;È minore di 0?
                goto    Loop            ;No
                movlw   0x09
                movwf   Contatore       ;Sì, imposta a 9 il contatore
                goto    Loop
```

Sia nel ciclo di conteggio ascendente che in quello discendente, l'operazione si realizza nella variabile Contatore. Dopo aver eseguito l'operazione si richiama una maschera, perché il numero sia sempre compreso fra 0 e 9, che rappresentano il valore minimo e massimo visualizzabili sul display. In questo modo, se la variabile contatore passa a valore 10 la si visualizza con il numero 0, e se dopo un decremento passa a valore 255, la si visualizza con il numero 9.



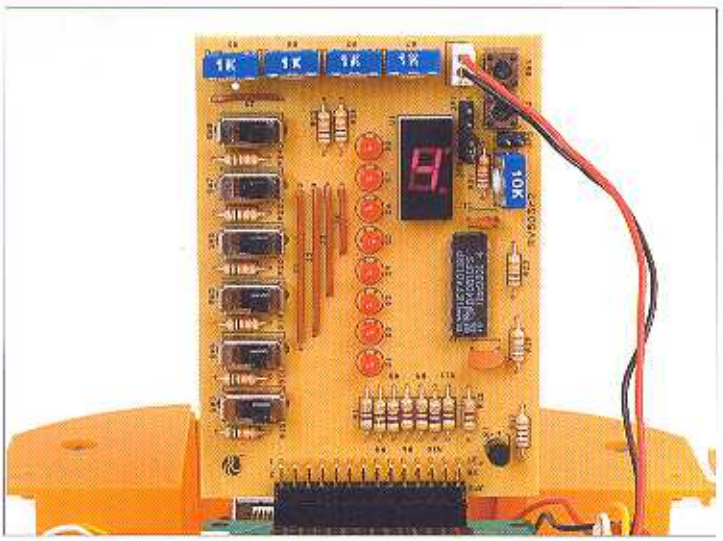
```
;*******************************************************************************
;Tabella: Questa routine converte il codice BCD presente sui 4 bits meno significativi
;del reg. W nel suo equivalente a 7 segmenti. Contemporaneamente il codice a 7 segmenti
;viene scritto anche nel registro W
;
Tabella:        andwf   PCL,F                   ;Spostamento sopra la tabella
                retlw   b'00111111'             ;Digit 0
                retlw   b'00000110'             ;Digit 1
                retlw   b'01011011'             ;Digit 2
                retlw   b'01001111'             ;Digit 3
                retlw   b'01100110'             ;Digit 4
                retlw   b'01101101'             ;Digit 5
                retlw   b'01111101'             ;Digit 6
                retlw   b'00000111'             ;Digit 7
                retlw   b'01111111'             ;Digit 8
                retlw   b'01100111'             ;Digit 9
```

Questa è la funzione tabella che converte i numeri binari in codice numerico per il display a 7 segmenti. Nella variabile Contatore si memorizza il valore del contatore sotto forma di numero binario, che grazie alla maschera sarà sempre compreso fra 0 e 9. Prima di inviare il valore del contatore alla porta B nel ciclo Loop, è necessario richiamare la funzione Tabella per fare in modo che il valore binario sia convertito nel codice adatto al display a 7 segmenti.



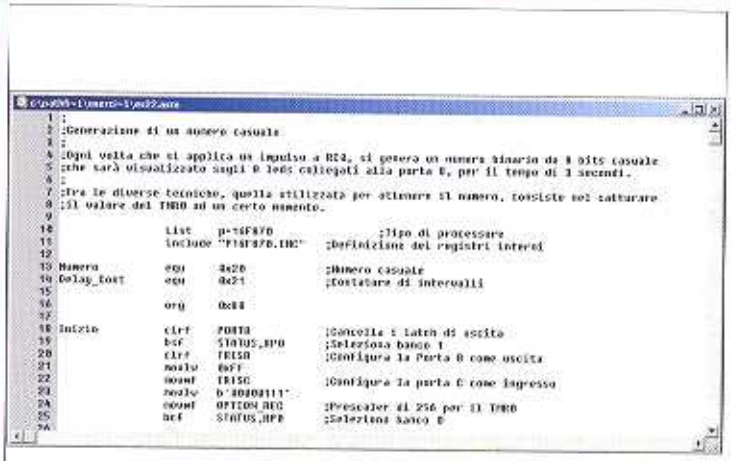
```
;*******************************************************************************
;Delay_20_ms: Questa routine di ritardo ha come obiettivo eliminare "l'effetto rimbalzo",
;caratteristica dei componenti elettromeccanici.
;Realizza un ritardo di 20 ms. Se il PIC lavora ad una frequenza di 4 MHz,
;il TMR0 si aggiorna ogni µs. Se si desidera temporizzare 20000 µs (20 ms) con
;un prescaler di 128, il TMR0 dovrà contare 156 eventi,(155 + 128).
;Il valore 156 espresso a 9c hex. e, dato che  il TMR0 è ascendente, lo dovremo cari-
;care con il suo complemento a 1 (63 hex.).

Delay_20_ms:    movlw   B'11010111'             ;Resetta il flag di overflow del TMR0
                movwf   TMR0                    ;Carica il TMR0
Delay_20_ms_1   clrwdt                          ;Aggiorna il WDT
                btfss   INTCON,T0IF             ;Overflow del TMR0?
                goto    Delay_20_ms_1           ;No: ancora
                bcf     INTCON,T0IF             ;Sì: sì, resettare il flag
                return

                end                             ;Fine del programma sorgente
```

Questo programma utilizza degli interruttori per fornire gli impulsi di ingresso, e come già sappiamo, i dispositivi meccanici sono soggetti all'effetto rimbalzo. Questo effetto, ogni volta che un sensore cambia stato, genera un periodo di instabilità all'uscita del sensore stesso. Per contrastarlo, ogni volta che RC0 cambia stato si richiama questa funzione; essa temporizza 20 ms, cosicché quando si ritorna al ciclo principale, l'uscita dell'interruttore si è già stabilizzata.



Questo esercizio si trova sul CDROM sotto il nome es21.asm. Dopo aver compilato l'esercizio lo memorizzeremo sulla Smartcard. Nell'immagine possiamo vedere l'esecuzione di questo esercizio. Per vedere i numeri sul display dobbiamo chiudere il jumper JP4 della scheda di ingressi e uscite, e aprire il resto dei jumper della scheda. Con l'interruttore SW3 inseriremo gli impulsi per il contatore; mediante l'interruttore SW4 selezioneremo se il conteggio dovrà essere ascendente o discendente.

Esercizi di apprendimento

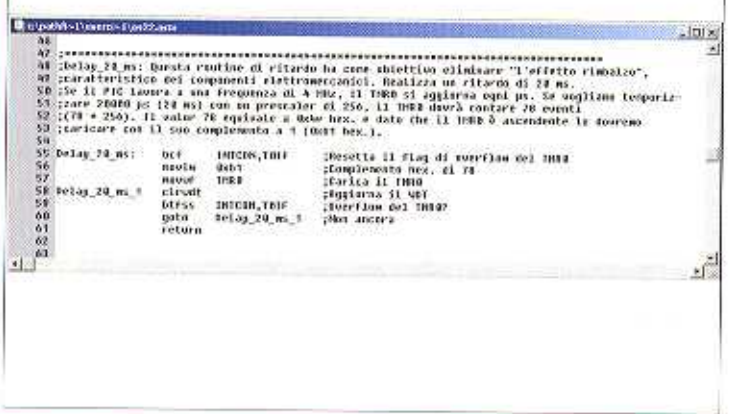


```
;
;Generazione di un numero casuale
;
;Ogni volta che si applica un impulso a RC3, si genera un numero binario da 8 bits casuale
;che sarà visualizzato sugli 8 leds collegati alla porta B, per il tempo di 3 secondi.
;
;Tra le diverse tecniche, quella utilizzata per ottenere il numero, consiste nel catturare
;il valore del TMR0 ad un certo momento.

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

Numero          equ     0x20            ;Numero casuale
Delay_Cost      equ     0x21            ;Costatore di intervalli

                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'00000111'
                movwf   OPTION_REG      ;Prescaler di 256 per il TMR0
                bcf     STATUS,RP0      ;Selezione banco 0
```

Il presente esercizio è un generatore di numeri casuali. Il programma è il preludio a un altro esercizio che consisterà in un dado elettronico. Per poter sviluppare questo programma dobbiamo gestire interruttori e led come ingressi e uscite e anche il temporizzatore interno del microcontroller Timer0. Ogni volta che premeremo l'interruttore SW3 della scheda di ingressi e uscite si visualizzerà un numero binario casuale sulla barra dei diodi led, che rimarrà visibile per tre secondi.

```
Loop            clrwdt                  ;Aggiorna il WDT
                btfsc   PORTC,0         ;E' attivo RC0?
                goto    Loop            ;Non ancora
                movf    TMR0,W          ;Sì.
                movwf   Numero          ;Catturo il valore del TMR0 (NP casuale)
                call    Delay_20_MS     ;Elimina rimbalzi

RC0_1           clrwdt                  ;Aggiorna il WDT
                btfsc   PORTC,0         ;RC0 è stato disattivato?
                goto    RC0_1           ;Non ancora
                movf    Numero,W        ;E' stato generato un impulso su RC0, si legge
                movwf   PORTB           ;il valore catturato dal TMR0 (NP casuale)
                                        ;e lo si visualizza sui leds della PORTB 0

                movlw   d'66'
                movwf   Delay_Cost      ;Inizializza la variabile di temporizzazione
                call    Delay_sec       ;Temporizza 3 secondi
                clrf    PORTB           ;Resetta le uscite
                goto    Loop
```

Il Timer0 è un contatore interno del microcontroller che si utilizza per realizzare temporizzazioni. Questo contatore è sempre attivo, per questo motivo ogni volta che l'interruttore RC0 passa a "1" si acquisisce il valore del Timer0 che sarà un numero casuale fra 0 e 255, essendo il Timer0 un contatore a 8 bit. Quando vogliamo realizzare funzioni casuali, un buon metodo è acquisire il valore del temporizzatore del microcontroller, dato che per ogni acquisizione si potrà avere un valore qualsiasi.

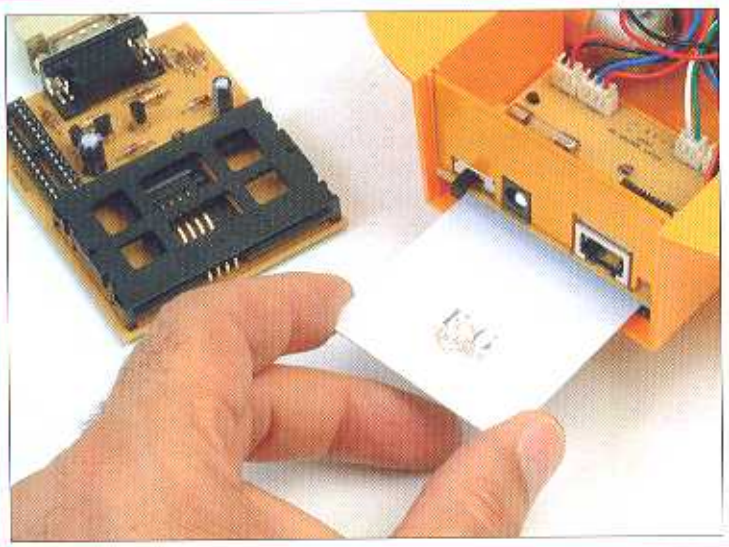


```
;*****************************************************************************
;Delay_20_ms: Questa routine di ritardo ha come obiettivo eliminare "l'effetto rimbalzo",
;caratteristico dei componenti elettromeccanici. Realizza un ritardo di 20 ms.
;Se il PIC lavora a una frequenza di 4 MHz, il TMR0 si aggiorna ogni µs. Se vogliamo temporiz-
;zare 20000 µs (20 ms) con un prescaler di 256, il TMR0 dovrà contare 78 eventi
;(78 * 256). Il valore 78 equivale a 0x4e hex. e dato che il TMR0 è ascendente lo dovremo
;caricare con il suo complemento a 1 (0x31 hex.).

Delay_20_ms:    bcf     INTCON,T0IF     ;Resetta il Flag di overflow del TMR0
                movlw   0x31            ;Complemento hex. di 78
                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
```

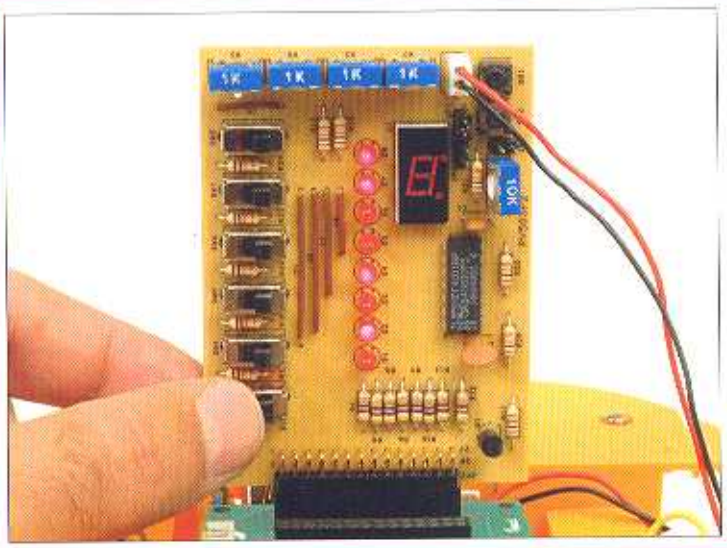
Il numero casuale da mostrare sulla barra dei diodi led si produce quando si introduce un impulso completo mediante l'interruttore SW3 della scheda, che invia segnali al pin RC0 del microcontroller. Bisogna eliminare l'effetto rimbalzo dei sensori meccanici, e per questo motivo quando RC0 cambia il suo valore a 1, dopo aver acquisito il valore casuale del Timer0, si chiama una routine di temporizzazione di 20 ms che elimina l'effetto rimbalzo dell'interruttore.

```
;Delay_var: Questa routine di utilizzo generale realizza una temporizzazione variabile
;fra 50 ms e 12.8 sec. Si utilizza un prescaler di 256 e sul TMR0 si carica 195.
;La velocità di lavoro è di 4 Mhz e quindi il TMR0 si incrementa ogni µs. In
;questo modo, il TMR0 deve contare 195 eventi che, con un prescaler di 128 danno un
;intervallo totale di 50000 µs = 50 ms (195 * 256). Il valore 195 lo dobbiamo caricare
;in hex. (C3) e, dato che il TMR0 è ascendente dovremo caricare il suo complemento a 1 (3C hex.)
;Questo intervallo di 50 ms si ripete tante volte quante indicate dalla variabile "Delay_cont",
;per questo il ritardo minimo è di 50 ms ("Delay_cont=1) e quello massimo di 12.8 sec.
;(Delay_cont=255).

Delay_var:      bcf     INTCON,T0IF     ;Resetta il flag di overflow
                movlw   0x3C            ;Complemento hex. di 195
                movwf   TMR0            ;Carica il TMR0
Intervallo      clrwdt                  ;Aggiorna il WDT
                btfss   INTCON,T0IF     ;Overflow del TMR0?
                goto    Intervallo      ;Non ancora
                decfsz  Delay_cont,f    ;Decrementa il contatore degli intervalli
                goto    Delay_var       ;Ripete l'intervallo di 50 ms
                return

                end                     ;Fine del programma sorgente
```

Al ritorno dalla routine di temporizzazione per annullare i rimbalzi, è sufficiente attendere che l'interruttore RC0 torni a valore "0" per mostrare il dato acquisito sulla variabile "Numero" tramite i diodi, cosa che eseguiremo passando il contenuto della variabile "Numero" alla porta B; dobbiamo visualizzare il numero casuale per 3 secondi e poi spegnere i led. Per fare questo eseguiremo una chiamata a una routine di temporizzazione che attende 3 secondi. Nella routine si realizza una temporizzazione base di 50 ms con il Timer0, e la si ripete 60 volte. 60 è il contenuto caricato nella variabile Delay_cont.
Il risultato sarà una temporizzazione di tre secondi dopo la quale spegneremo i led e torneremo ad attendere un altro cambio di stato sull'interruttore SW3 per ripetere il ciclo.

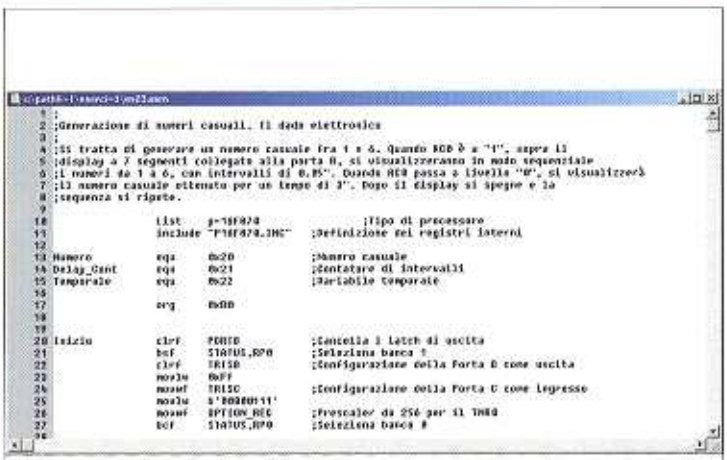


Questo esercizio si trova sul CDROM, sotto il nome di es22.asm. Dopo aver compilato l'esercizio e ottenuto il file es22.hex, lo scriveremo sulla Smartcard. Dobbiamo collegare la scheda di scrittura alla porta seriale del PC ed eseguire il programma ICPROG. Selezioneremo il dispositivo modello 24C16 e apriremo il file es22.hex. Ora selezioneremo il comando programma tutto. Dopo aver programmato la Smartcard la inseriremo nella scheda di alimentazione di Pathfinder con l'orientamento uguale a quello mostrato nell'immagine.



Quando realizzeremo gli esercizi di apprendimento con la scheda di ingressi e uscite, nessun altro sensore deve essere collegato al robot tramite la scheda di controllo o la scheda di interfaccia. Tutti i connettori dei sensori rimarranno senza collegamento. Il microcontroller impiegherà alcuni minuti a leggere il contenuto della Smartcard e ad eseguire l'esercizio. Mediante l'interruttore SW3 inseriremo gli impulsi in grado di generare numeri casuali che visualizzeremo per tre secondi sulla barra dei diodi led. Il jumper JP1 della scheda di ingressi e uscite rimarrà chiuso per attivare i led.

Esercizi di apprendimento

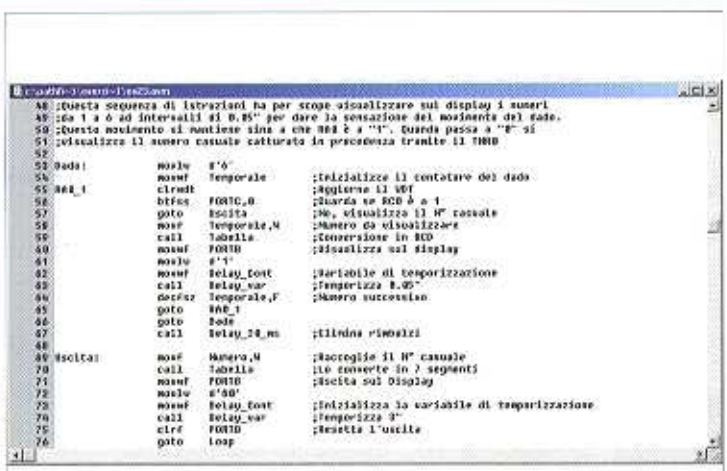


```
;
;Generazione di numeri casuali. Il dado elettronico
;
;Si tratta di generare un numero casuale fra 1 e 6. Quando RC0 è a "1", apre il
;display a 7 segmenti collegato alla porta B, si visualizzeranno in modo sequenziale
;i numeri da 1 a 6, con intervalli di 0.05". Quando RC0 passa a livello "0", si visualizzerà
;il numero casuale ottenuto per un tempo di 3". Dopo il display si spegne e la
;sequenza si ripete.

                list    p=16F870A               ;Tipo di processore
                include "P16F870A.INC"  ;Definizione dei registri interni

Numero          equ     0x20            ;Numero casuale
Delay_Cont      equ     0x21            ;Contatore di intervalli
Temporale       equ     0x22            ;Variabile temporale

                org     0x00


Inizio          clrf    PORTB           ;Cancella i Latch di uscita
                bsf     STATUS,RP0      ;Seleziona banco 1
                clrf    TRISB           ;Configurazione della Porta B come uscita
                movlw   0xFF
                movwf   TRISC           ;Configurazione della Porta C come ingresso
                movlw   b'00000111'
                movwf   OPTION_REG      ;Prescaler da 256 per il TMR0
                bcf     STATUS,RP0      ;Seleziona banco 0
```

Questo esercizio combina quasi tutte le conoscenze sulla programmazione che abbiamo acquisito sino a questo momento.
Si tratta di sviluppare un dado elettronico che genererà numeri casuali fra 1 e 6, visualizzati tramite il display a 7 segmenti della scheda di ingressi e uscite.
Avremo bisogno di leggere l'interruttore SW3 per attivare il dado, eliminare l'effetto rimbalzo, gestire una routine di generazione di numeri casuali, fare in modo che il numero sia compreso fra 1 e 6 e realizzare una temporizzazione per fare spegnere il display dopo 3 secondi.

```
Loop            clrwdt                  ;Aggiorna il WDT
                btfss   PORTC,0         ;Si è attivato RC0?
                goto    Loop            ;Non ancora
                movf    TMR0,W          ;Sì!
                movwf   Numero          ;Cattura il valore del TMR0 (N° casuale)
                call    Delay_20_ms     ;Elimina i rimbalzi

;Il numero casuale è, mediante sottrazioni consecutive, diviso per 6. In questo modo
;l'ultimo resto sarà fra 0 e 5 che sarà incrementato di una unità in modo che alla
;fine risulti un numero fra 1 e 6
Dividi          movlw   d'6'
                subwf   Numero,W        ;Sottrai 6 al numero casuale
                movwf   Numero          ;memorizzalo
                sublw   d'5'
                btfss   STATUS,C        ;Guarda se è minore di 5
                goto    Dividi          ;No
                incf    Numero,F        ;Il numero è fra 1 e 6
```

Nella routine di inizio realizzeremo le configurazioni dei registri per il programma. La porta C come ingresso, la porta B come uscita e il prescaler a 256 per il Timer0. Nel ciclo "Loop" si attende che l'interruttore RC0 passi a 1. In questo momento si acquisisce il valore del Timer0, portandolo sulla variabile "Numero", detto valore sarà un numero casuale fra 0 e 255. Il dato da visualizzare sul display a 7 segmenti deve essere compreso fra 1 e 6, per questo motivo nella routine "Dividi" si opera con la variabile "Numero" sino a che il suo valore casuale sia compreso in questo intervallo.



```
;Questa sequenza di istruzioni ha per scopo visualizzare sul display i numeri
;da 1 a 6 ad intervalli di 0.05" per dare la sensazione del movimento del dado.
;Questo movimento si mantiene sino a che RA4 è a "1". Quando passa a "0" si
;visualizza il numero casuale catturato in precedenza tramite il TMR0

Dado:           movlw   d'6'
                movwf   Temporale       ;Inizializza il contatore del dado
RA4_1           clrwdt                  ;Aggiorna il WDT
                btfss   PORTC,0         ;Guarda se RC0 è a 1
                goto    Uscita          ;No, visualizza il N° casuale
                movf    Temporale,W     ;Numero da visualizzare
                call    Tabella         ;Conversione in BCD
                movwf   PORTB           ;Visualizza sul display
                movlw   d'1'
                movwf   Delay_Cost      ;Variabile di temporizzazione
                call    Delay_var       ;Temporizza 0.05"
                decfsz  Temporale,F     ;Numero successivo
                goto    RA4_1
                goto    Dado
                call    Delay_20_ms     ;Elimina rimbalzi

Uscita:         movf    Numero,W        ;Raccoglie il N° casuale
                call    Tabella         ;Lo converte in 7 segmenti
                movwf   PORTB           ;Uscita sul Display
                movlw   d'60'
                movwf   Delay_Cost      ;Inizializza la variabile di temporizzazione
                call    Delay_var       ;Temporizza 3"
                clrf    PORTB           ;Resetta l'uscita
                goto    Loop
```

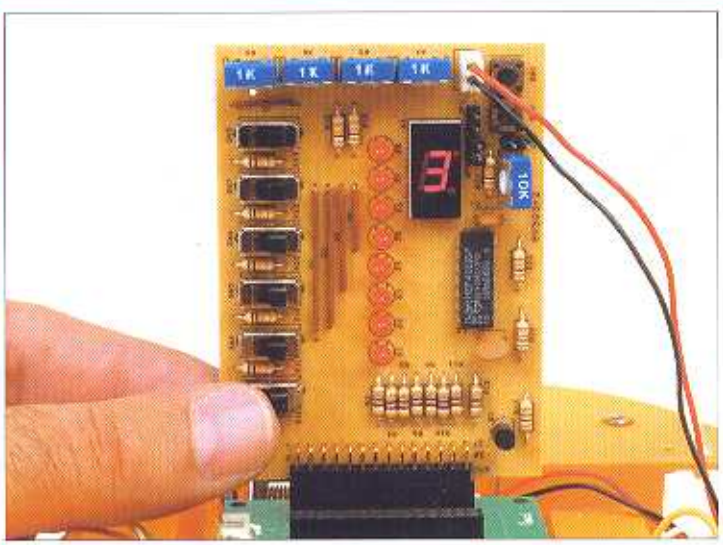
Sino a quando l'interruttore RC0 non passa a valore 0, non si deve visualizzare il numero casuale sul display a 7 segmenti. Nell'attesa che ciò accada, la routine dado opera sul display visualizzando numeri casuali ad alta velocità; all'interno di questo ciclo si testa RC0, in modo che quando l'interruttore SW3 passa a valore 0, termini il ciclo e si visualizzi il numero casuale generato, scritto nella variabile "Numero", tramite il display a 7 segmenti. Dopo aver visualizzato il numero casuale si temporizzano tre secondi e si spegne il display.

```
;*****************************************************************************
;Tabella: Questa routine converte il codice binario presente sui 4 bits meno significativi
;del reg. W nel suo equivalente a 7 segmenti. Il codice a 7 segmenti viene anche scritto
;sul reg. W
;
Tabella:        addwf   PCL,F           ;Spostamento sopra la tabella
                retlw   b'00111111'     ;Digit 0
                retlw   b'00000110'     ;Digit 1
                retlw   b'01011011'     ;Digit 2
                retlw   b'01001111'     ;Digit 3
                retlw   b'01100110'     ;Digit 4
                retlw   b'01101101'     ;Digit 5
                retlw   b'01111101'     ;Digit 6

;*****************************************************************************
;Delay_20 ms: Questa routine di ritardo ha come obiettivo eliminare "l'effetto rimbalzo",
;caratteristico dei componenti elettromeccanici. Realizza un ritardo di 20 ms.
;Se il PIC lavora ad una frequenza di 4 MHz, il TMR0 si aggiorna ogni µs. Se vogliamo
;temporizzare 20000 µs (20 ms) con un prescaler di 256, il TMR0 dovrà contare 78 eventi
;(78 * 256). Il valore 78 equivale a 0e4e hex. e dato che il TMR0 è ascendente lo dovremo
;caricare con il suo complemento a 1 (0b1 hex.).

Delay_20_ms:    bcf     INTCON,T0IF     ;Resetta il flag di overflow del TMR0
                movlw   0xb1            ;Complemento hex. di 78
                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
```

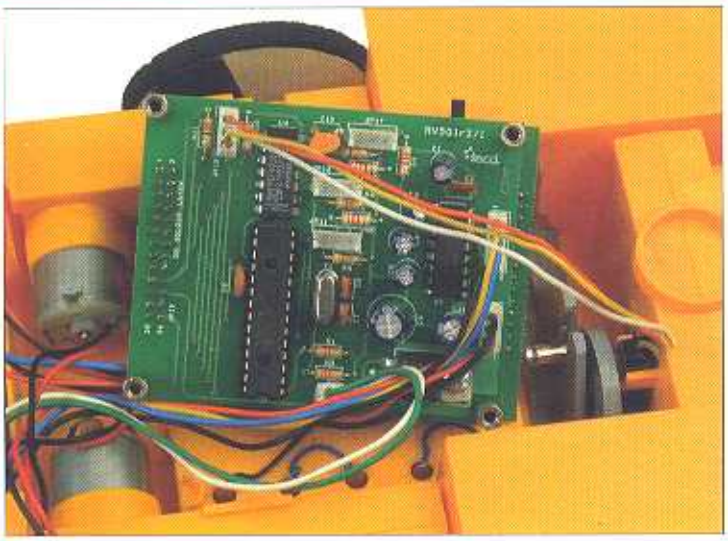
Negli esercizi precedenti abbiamo utilizzato le due routines mostrate nell'immagine. La prima di esse è la routine di conversione dei numeri binari a numeri in codice per il display a 7 segmenti. È necessario richiamare questa routine prima di inviare il numero casuale alla porta B. L'altra funzione è una temporizzazione di 20 ms utilizzata per eliminare l'effetto rimbalzo dell'interruttore.

```
;*****************************************************************************
;Delay_var: Questa routine di utilizzo generale realizza una temporizzazione variabile
;fra 50 ms e 12.8". Si utilizza un prescaler di 256 e sul TMR0 si carica 195.
;La velocità di lavoro è di 4 Mhz e quindi il TMR0 si incrementa ogni µs. In
;questo modo, il TMR0 deve contare 195 eventi che, con un prescaler di 128 danno un
;intervallo totale di 50000 µs = 50 ms (195 * 256). Il valore 195 lo dobbiamo esprimere
;in hex. (c3) e, data che il TMR0 è ascendente dovremo caricare il suo complemento a 1 (3C hex.)
;Questo intervallo di 50 ms si ripete tante volte quante indicate dalla variabile "Delay_cont".
;Per questo il ritardo minimo è di 50 ms ("Delay_cont=1) e quello massimo di 12.8 sec.
;(Delay_cont=255).

Delay_var:      bcf     INTCON,T0IF     ;Resetta il flag di overflow
                movlw   0x3c            ;Complemento hex. di 195
                movwf   TMR0            ;Carica il TMR0
Intervallo      clrwdt                  ;Aggiorna il WDT
                btfss   INTCON,T0IF     ;Overflow del TMR0?
                goto    Intervallo      ;Non ancora
                decfsz  Delay_Cont,F    ;Decrementa il contatore degli intervalli
                goto    Delay_var       ;Ripete l'intervallo di 50 ms
                return

                end                     ;Fine del programma sorgente
```

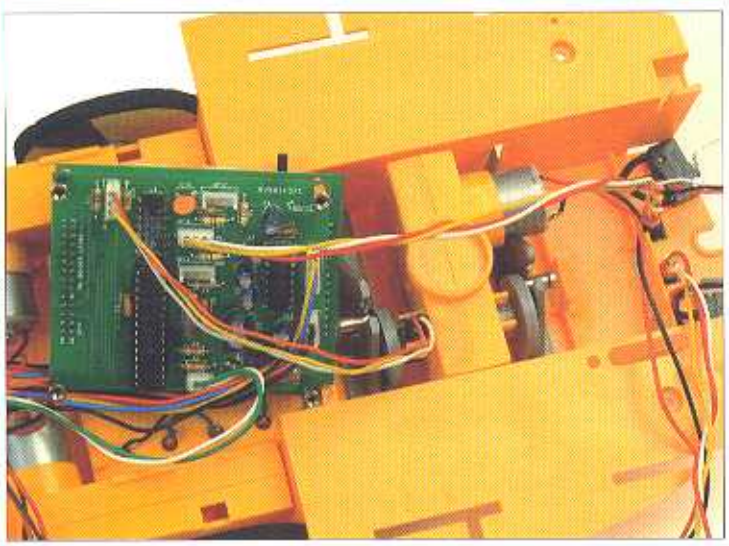
Questa funzione è già stata utilizzata in precedenti programmi di generazione di numeri casuali, con essa realizziamo una temporizzazione di tre secondi che sarà il tempo per cui rimarrà il dato visualizzato sul display a 7 segmenti. Questa funzione realizza una temporizzazione sulla base di 50 ms che si ripete tante volte quante indicate nella variabile Delay_cont. Nel caso si volessero temporizzare tre secondi, è necessario inizializzare questa variabile con il valore 60, prima di entrare nella funzione. Se vogliamo temporizzare solamente un secondo, caricheremo il dato 20 in questa variabile.



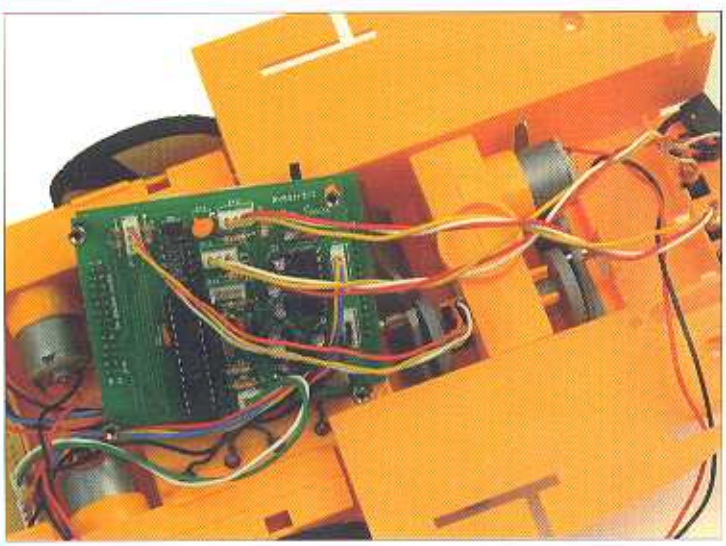
Dopo aver compilato il programma procederemo alla sua scrittura sulla Smartcard, utilizzando il software ICPROG e la scheda di scrittura, seguendo la stessa procedura già applicata per il resto degli esercizi. Per provare il programma dobbiamo aprire i jumper JP1 e JP5 della scheda di ingressi e uscite e chiudere il jumper JP4 della stessa scheda. In questo modo attiveremo il display a 7 segmenti. Utilizzeremo l'interruttore SW3 per iniziare la generazione e la visualizzazione dei numeri casuali sul display.



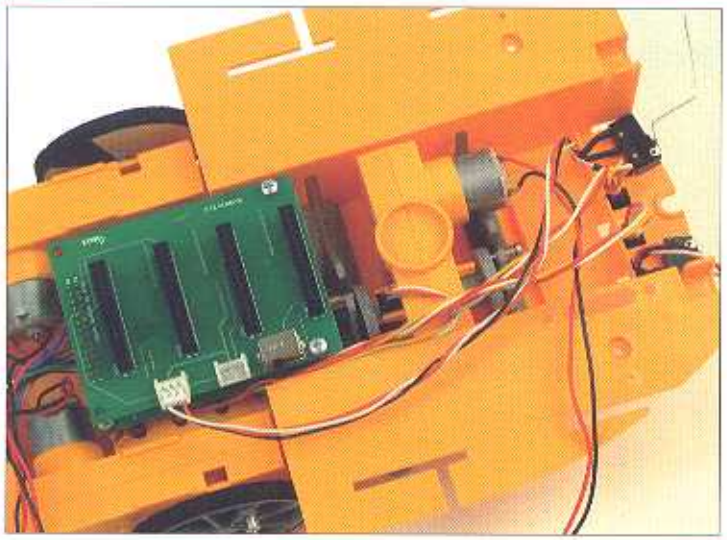
Inizieremo realizzando alcuni programmi di apprendimento con i motori e sensori di Pathfinder. Per questo avremo bisogno di utilizzare la scheda di potenza che abbiamo già montato.
Però prima di realizzare i programmi il primo passo consisterà nel collegare adeguatamente i sensori alle diverse schede del robot. Il primo sensore da collegare sarà il sensore ottico CNY70, che abbiamo montato per il controllo del motore centrale e che si collegherà sul connettore JP18 della scheda di controllo.



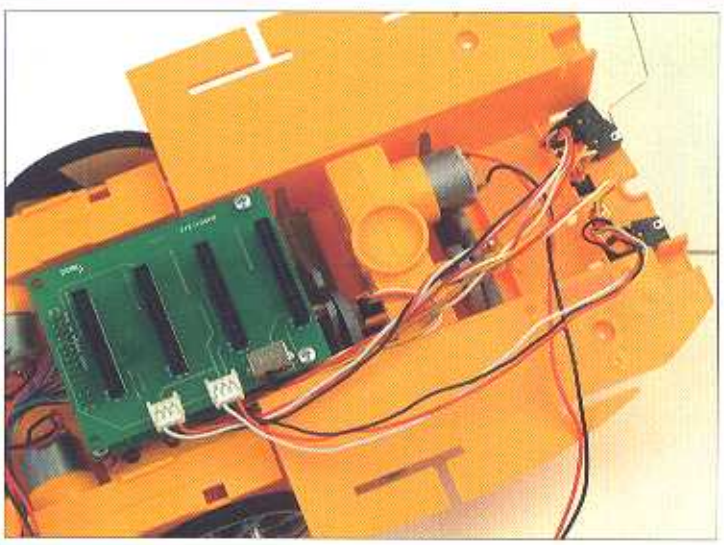
Collegheremo ora gli altri due sensori ottici che abbiamo montato nella parte anteriore del telaio del robot. Il riferimento per distinguere se il sensore è sinistro o destro consiste nel guardare il robot dalla sua parte posteriore. Il sensore situato nel foro sinistro del telaio si collegherà al connettore JP12 della scheda di controllo.



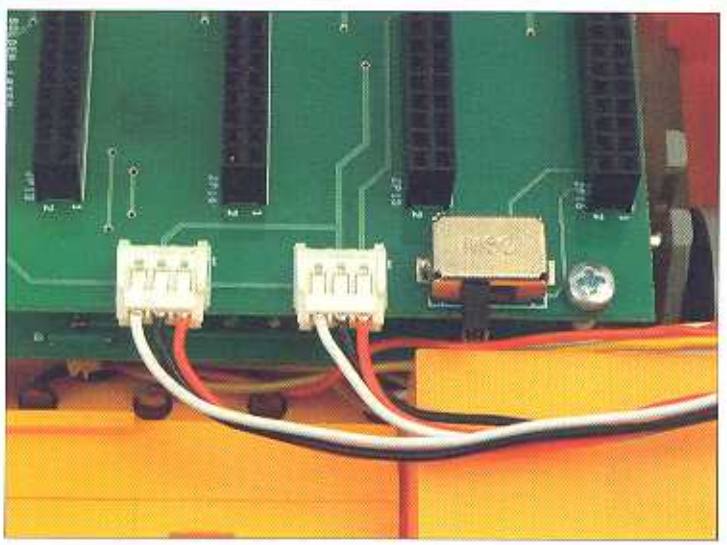
Per ultimo collegheremo il sensore ottico CNY70, che è l'altro sensore della parte anteriore del telaio ed è montato sulla parte destra. Questo sensore verrà collegato sul connettore JP7 della scheda di controllo.
I connettori hanno solo un verso di inserzione corretto, e dispongono di alcuni impedimenti meccanici mediante i quali è permessa l'inserzione solo nell'orientamento adeguato.



Ora collegheremo i sensori meccanici tipo finecorsa. Per distinguere il sensore sinistro dal destro utilizzeremo lo stesso riferimento dei sensori ottici, cioè osserveremo il robot dalla sua parte posteriore.
Il sensore finecorsa sinistro verrà collegato al connettore JP7 posizionato sulla scheda di interfaccia.



Infine l'ultimo sensore tipo finecorsa è quello situato sulla parte destra del robot.
Si collegherà al connettore JP8 della scheda di interfaccia. Questi connettori così come i sensori ottici CNY70, accettano solo il collegamento con l'orientamento corretto. Il finecorsa sinistro comunicherà con il pin RA1 del microcontroller, mentre il finecorsa destro invierà il suo stato al pin RA2.



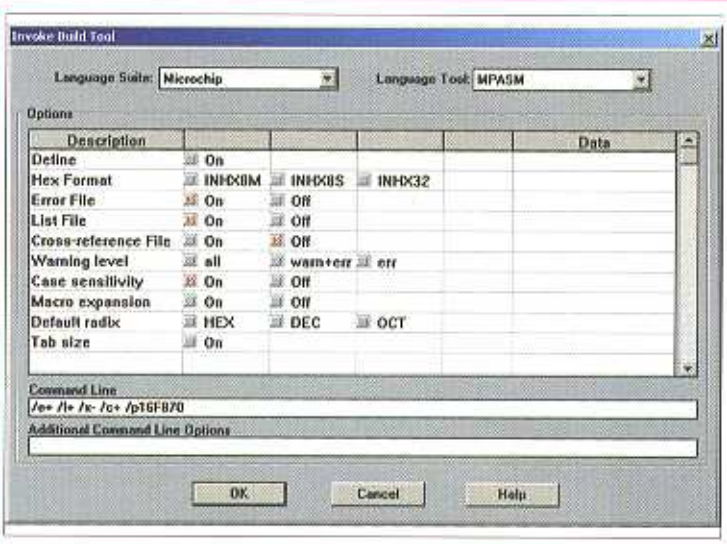
A lato dei connettori per i finecorsa sulla scheda di interfaccia troviamo il commutatore a due vie SW3. Questo commutatore serve a fare in modo che il robot funzioni con i sensori meccanici tipo finecorsa o con i sensori a ultrasuoni. Nei primi esercizi utilizzeremo i finecorsa. Per fare in modo che i finecorsa siano attivi la posizione del commutatore deve coincidere con quella mostrata nell'immagine.

```
;
;POT1.asm
;Programma che esegue l'accensione o lo spegnimento dei motori posteriori
;in funzione dello stato del finecorsa sinistro (connettore JP7, segnale su RA1)

                LIST    p=16F870                ;Tipo di processore
                include "p16F870.INC"   ;Definizione dei registri interni

                ORG     0x00

Inizio
                bsf     STATUS,RP0      ;Selezione del banco 1
                movlw   0x07
                movwf   ADCON0  ;Porta A come ingresso digitale
                clrf    TRISB           ;La Porta B si configura come uscita
                bcf     STATUS,RP0      ;Selezione del banco 0
                clrf    PORTB           ;Si spengono i motori
```

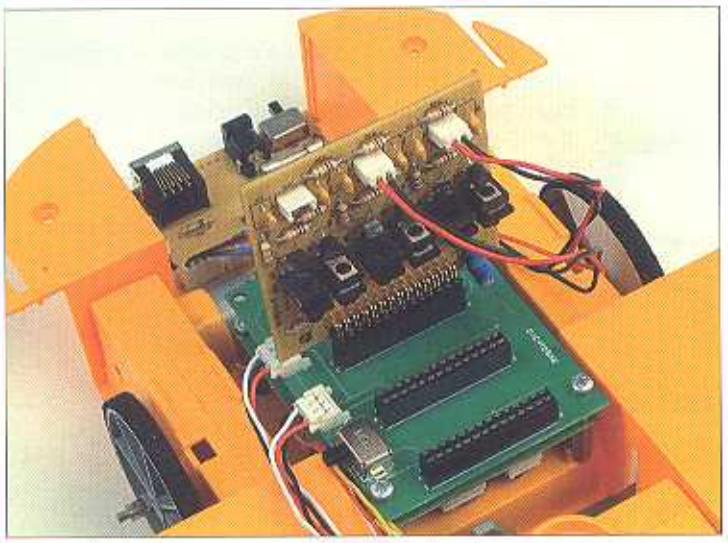
Il primo esercizio che realizzeremo consisterà nell'accensione e spegnimento dei motori posteriori del robot, mediante lo stato del finecorsa sinistro. Quando il finecorsa si attiva i motori cominceranno a funzionare e quando si trova disattivato si fermeranno. Il finecorsa sinistro invierà i segnali al piedino RA1 del microcontroller, i motori si controllano mediante la porta B del microcontroller. Per questo motivo la porta A si configura come ingresso e la porta B come uscita.

```
;RB2 e RB3 sono per il motore della ruota sinistra
;RB2: 0
;RB3: 1 -> Motore avanza
;RB4 e RB5 sono per il motore della ruota destra
;RB4: 1
;RB5: 0 -> Motore avanza

Loop    btfss   PORTA, 1       ;Si legge un '1' o uno '0' su RA1
                goto    OFF
                goto    ON

OFF     clrf    PORTB
                goto    Loop

ON              movlw   b'00011000'
                movwf   PORTB
                goto    Loop            ;Ciclo infinito

                END                     ;Fine del programma sorgente
```

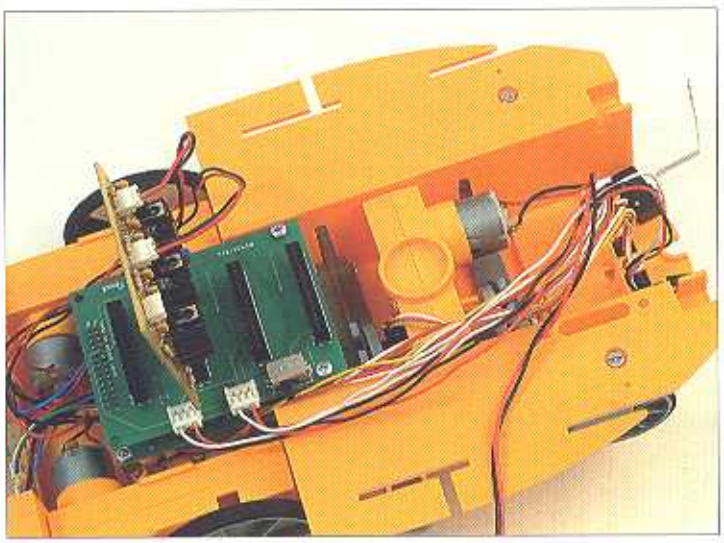
Nell'immagine del programma possiamo vedere la connessione dei motori posteriori alla porta B. Per accendere il motore sinistro e farlo muovere in avanti è necessario inviare un "1" logico tramite RB3 e uno "0" tramite RB2. Per accendere il motore destro nel verso dell'avanzamento bisogna inviare un "1" tramite RB4 ed uno "0" tramite RB5. Il ciclo principale del programma testa lo stato del finecorsa, quando il finecorsa si chiude invia un "1" logico al pin RA1 del microcontroller e quando rimane aperto uno "0". Nel momento in cui il finecorsa si chiude si salta alla routine ON per accendere i motori.



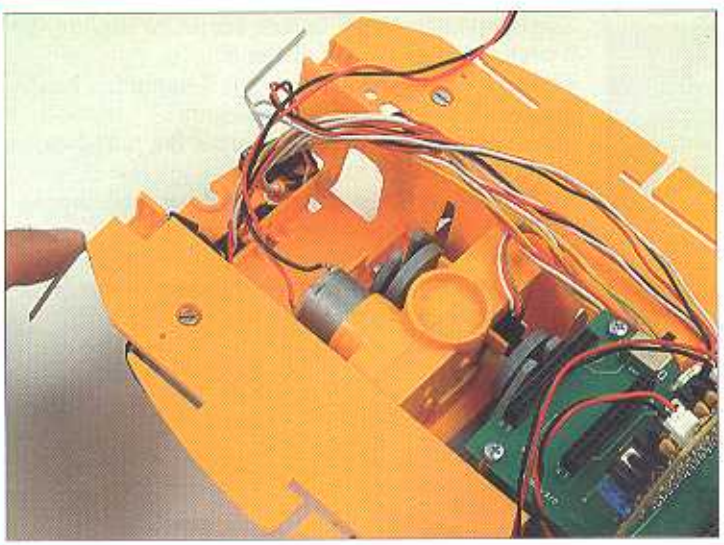
Dobbiamo scrivere il codice sorgente utilizzando il programma MPLAB. Dopo averlo scritto lo memorizzeremo con il nome pot1.asm, dobbiamo compilare il programma senza errori, al fine di ottenere il file pot1.hex; che sarà il file da scrivere sulla Smartcard utilizzando per questo il programma ICPROG. Il procedimento per la programmazione è lo stesso che abbiamo utilizzato per tutti gli esercizi di apprendimento con la scheda di ingressi e uscite.



Per verificare l'esercizio, la scheda di ingressi e uscite, deve essere inserita nel connettore JP13 della scheda di interfaccia. Collegheremo la scheda di potenza sul connettore JP14 della scheda di interfaccia con lo stesso orientamento mostrato dall'immagine. Collegheremo anche il motore destro del robot al connettore JP2 della scheda di potenza e il motore sinistro al connettore JP4 della stessa scheda. Per capire qual è il motore destro e qual è quello sinistro il riferimento è guardare il robot dalla sua parte posteriore, lato in cui si inserisce la Smartcard.



Il finecorsa sinistro deve essere inserito sul connettore JP7 della scheda di interfaccia, la posizione del commutatore SW2 della scheda di interfaccia deve essere la stessa di quella che appare nell'immagine, in modo che i segnali dei finecorsa siano attivi verso il microcontroller della scheda di controllo.



Dopo aver scritto il programma pot1.hex sulla Smartcard la inseriremo nella scheda di alimentazione del robot. Dobbiamo alimentare il robot mediante cinque pile collocate nel porta batterie o con un alimentatore esterno avente un'uscita di circa 6 V.
Quando il microcontroller legge il contenuto della Smartcard inizierà l'esecuzione del programma, attivando il finecorsa sinistro i motori cominceranno a girare e rilasciando il finecorsa si fermeranno.