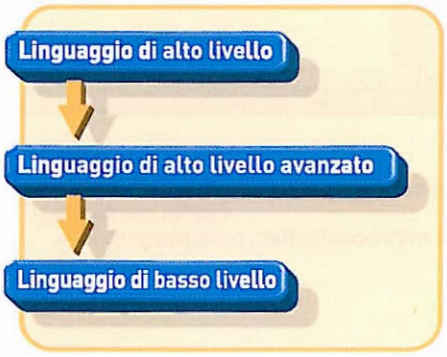
Linguaggi di programmazione da utilizzare

Allo stesso modo in cui una cosa si può dire in molti modi e in molte lingue, nel mondo dei computer, e per inciso, in quello dei microcontroller, lo stesso lavoro si può programmare in diversi modi e in diversi linguaggi; in altre parole possiamo indicare al computer



Linguaggi che utilizzeremo per imparare a programmare i microcontroller PIC.

le azioni da eseguire in svariati modi.

Nel caso dei microcontroller PIC, essi capiscono solo il linguaggio degli "uno e zero", cioè il codice macchina, però così come per le persone che conoscono una sola lingua, esistono dei traduttori che rendono possibile il passaggio da un linguaggio ad un altro. In questa sezione impareremo a lavorare con alcuni di questi linguaggi, per vedere quali vantaggi e quali inconvenienti presenta ognuno di essi.

Un linguaggio per prendere confidenza

Mentre alcune altre sezioni ci aiuteranno a conoscere un po' meglio ciò su cui si basano i piccoli circuiti che più tardi governeranno il nostro microrobot, noi inizieremo dalle tecniche di programmazione di un linguaggio di alto livello per microcontroller, il LetPicBasicLite.

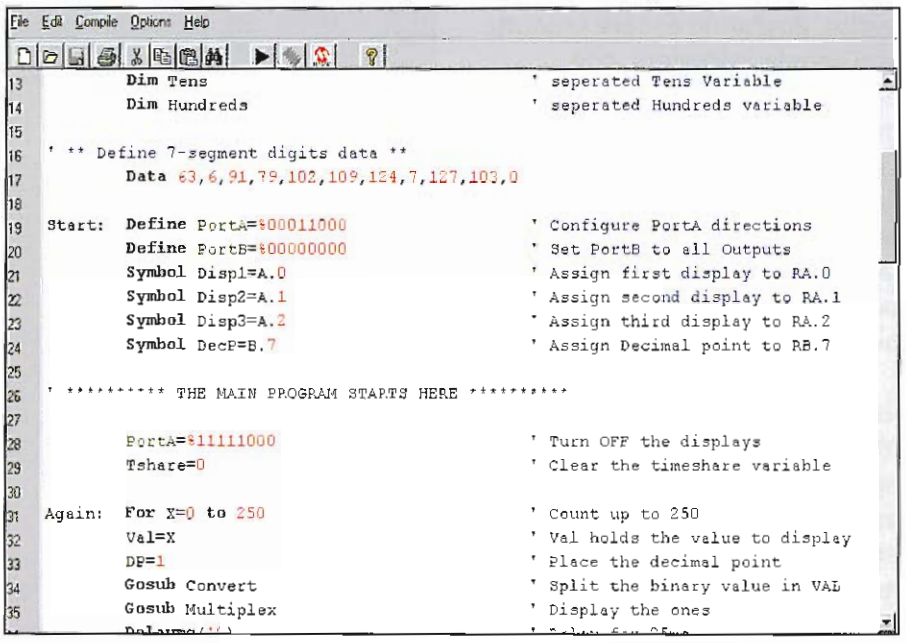
In questo modo, quando l'hardware sarà montato, avremo a disposizione dei piccoli esercizi da poter provare.

Trattandosi di un linguaggio di alto livello, le istruzioni e le strutture utilizzate sono facilmente comprensibili, essendo, nella maggioranza dei casi, vocaboli di lingua inglese dal significato concreto.

Anche se questo tipo di linguaggio si avvicina abbastanza alla forma strutturale del nostro modo di pensare, bisognerà rispettare una serie di regole del linguaggio di programmazione, così come si fa con le lingue, ma in modo più rigido.

Ampliamento del linguaggio

Per fare in modo che l'apprendimento sia progressivo, dopo il LetPicBasicLite, che ci permette di lavorare solo con il PIC16F84, avremo bisogno di un linguaggio adatto al lavoro con le risorse più avanzate dei PIC



Il linguaggio LetPicBasicLite è facilmente comprensibile.

16F87x. Per questo utilizzeremo il LetPicBasicPlus, che aggiungerà alle istruzioni già conosciute del linguaggio precedente, alcune nuove per la gestione delle suddette periferiche.

Ricordiamo, però, che questo linguaggio ha alcune restrizioni, derivate dal fatto che, trattandosi di una versione gratuita e libera, i programmi non potranno superare le 20 linee.

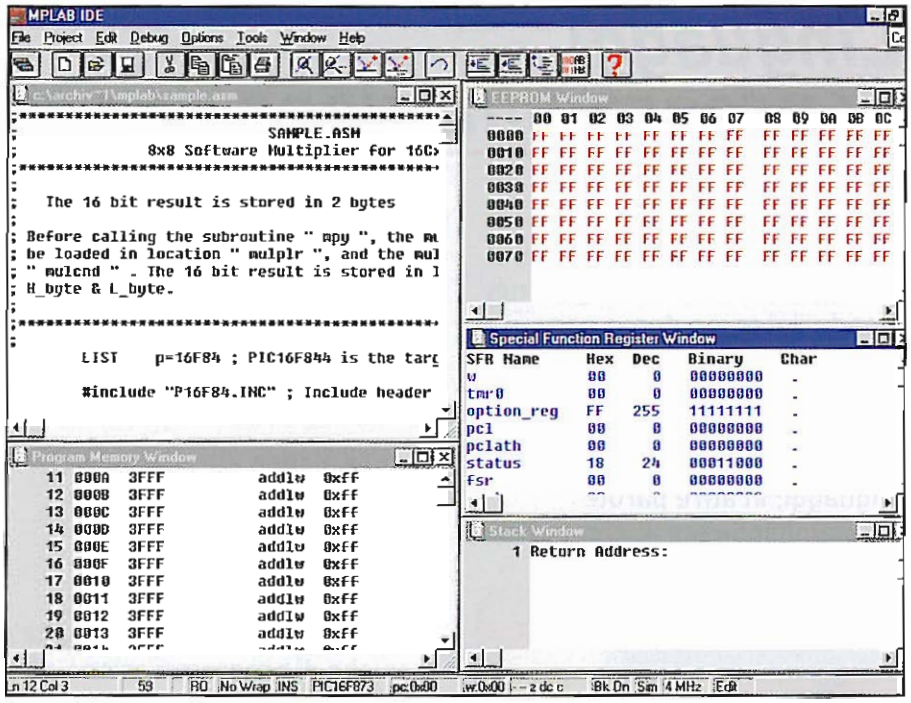
Scendiamo sino all'assembler

Dopo due linguaggi di alto livello, una volta che si conosce la struttura interna del microcontroller, siamo pronti per programmare in assembler, il linguaggio più vicino al codice macchina.

Anche qui dovremo seguire alcune regole fondamentali, legate al linguaggio stesso.

Passando da un linguaggio di alto livello ad uno di basso livello, dovranno essere tradotte non solo le istruzioni, che naturalmente saranno diverse, ma anche le strutture di controllo, dato che mentre il primo linguaggio le gestisce automaticamente, facilitandoci il lavoro di programmazione, il secondo non le gestisce, proprio per semplificare le sue regole.

Come contropartita i linguaggi di basso livello sono migliori sotto il profilo dello spazio occupato e del tempo di esecuzione, caratteristiche entrambe essenziali nella programmazione di microcontroller.



Il passaggio all'assembler, linguaggio di basso livello più vicino al microcontroller, sarà progressivo.

Con un ambiente simile al LetPicBasicLite, il LetPicBasicPlus permette di lavorare con risorse avanzate come il PWM.

```
' Adjust the brightness of an LED attached to PORTB.0 using the PWM command
        Device 16F877                   ' We'll use a 16F877 for this experimemt
        Remarks ON                      ' Enable Assembler remarks
        Declare XTAL 20                 ' Choose a crystal frequency of 20MHz
        Dim Loop as Byte

Inf:    Loop = 0                        ' Start the loop at 0
        Repeat                          ' Beginning of the loop
            PWM PortB.0,Loop,20         ' Output the value in loop for 20ms
            Inc Loop                    ' Increment the loop counter
        Until Loop = 255                ' Loop until 255 revolutions

        Repeat                          ' Loop still contains 255
            PWM PortB.0,Loop,20         ' Output the value in loop for 20ms
            Dec Loop                    ' Decrement the loop counter
        Until Loop = 0                  ' Loop until 255 revolutions
        Goto Inf                        ' Do it forever
```