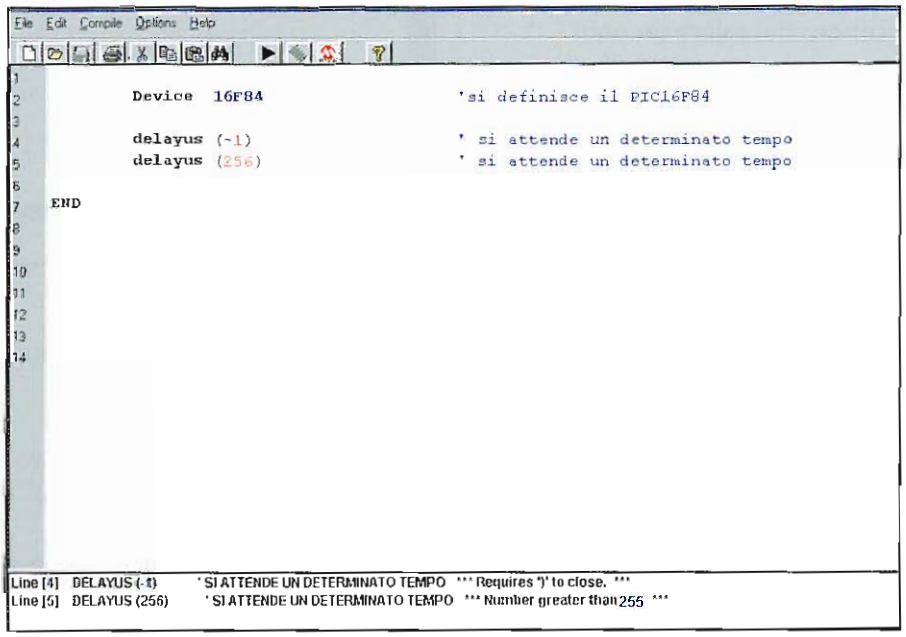
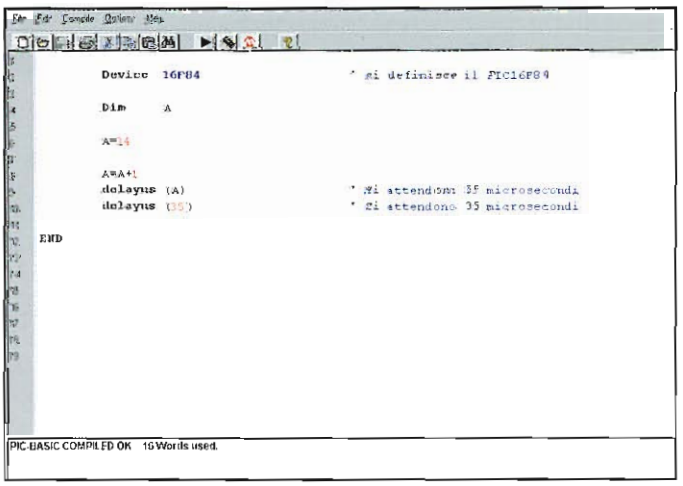
Fermare il tempo

Un microcontroller funziona a una velocità (4, 10, 20 MHz) che, se comparata a quella di un personal computer ci può apparire irrisoria, ma è molto superiore a quello che noi possiamo percepire lavorando con le periferiche.

Far lampeggiare un diodo LED, accenderlo e spegnerlo in modo consecutivo a una velocità di 4 MHz, fa sì che il diodo si accenda per 1 µs e che si spenga il microsecondo successivo, per cui noi non saremo in grado di percepire questo lampeggio. Come abbiamo già visto in altre occasioni, però, in alcuni casi è possibile utilizzare un piccolo trucco, cioè "fermare il tempo" per diversi microsecondi, oppure millisecondi, in modo che fra l'accensione e lo spegnimento questa "perdita di tempo" renda possibile notare la differenza; tale procedura è nota con il nome di introduzione di un "delay".



I limiti dei microsecondi sono compresi fra 0 e 255.

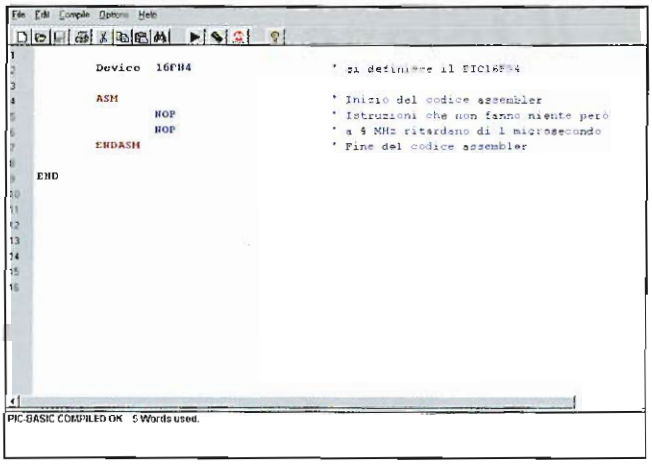


Metodi validi per passare un parametro all'istruzione.

Tempo in microsecondi

Il LetPicBasic possiede tre istruzioni per controllare il tempo. La prima di queste permette di realizzare ritardi di diversi microsecondi. L'unico parametro che accetta questa istruzione è il numero di microsecondi da perdere, che deve essere compreso fra 0 e 255. Un valore negativo o superiore a 255 ci darà un errore di compilazione. Come per altri casi il parametro si può inserire sotto forma di variabile oppure di costante, sempre tenendo conto dei limiti, che dovremo controllare qualora la variabile arrivi dall'esterno, dato che in questo caso l'errore non si produrrà nella compilazione, ma nell'esecuzione e sarebbe più difficile da trovare. Tuttavia il parametro non può essere un'espressione; se vogliamo realizzare delle operazioni con le

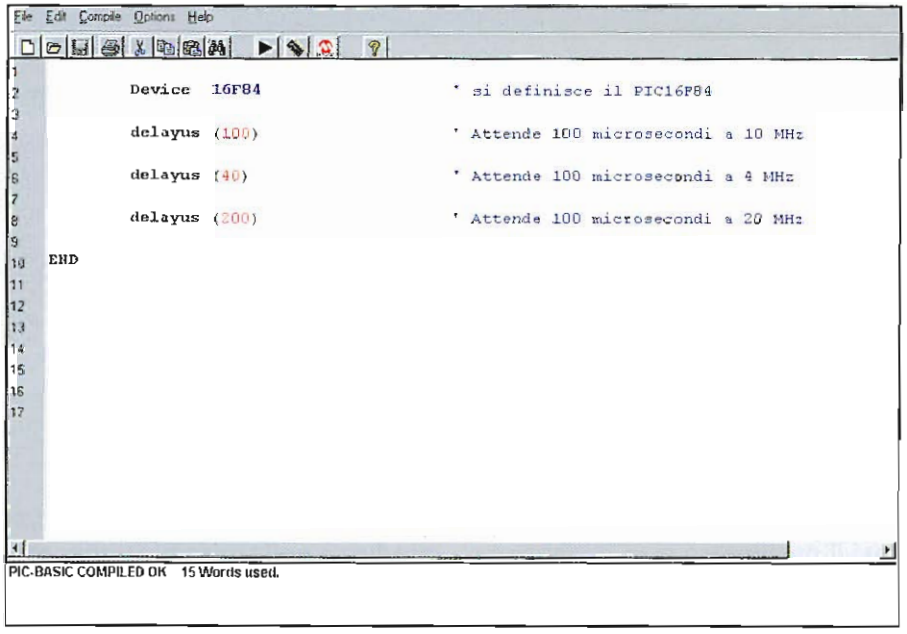
variabili, lo dobbiamo fare prima di eseguire la chiamata all'istruzione di delay. Non dobbiamo mai dimenticare le parentesi che contengono il parametro del numero dei microsecondi.



In alcuni casi bisogna utilizzare direttamente le istruzioni assembler.

Precauzioni nell'utilizzo di "delayus"

Anche se inizialmente l'utilizzo di questa istruzione può essere semplice, bisogna tenere sempre presenti un paio di precauzioni. Innanzitutto, il LetPicBasic presuppone che il microcontroller lavori con un quarzo da 10 MHz, in modo che a questa velocità l'istruzione "delayus (100)" produca una perdita di tempo di 100 microsencondi. Però se il PIC sta funzionando con un quarzo da 4, oppure da 20 MHz, la stessa istruzione genera dei ritardi diversi. La corrispondenza si può fare con una regola del tre, in tal modo le tre istruzioni dell'esempio generano lo stesso ritardo se ognuna di esse viene eseguita con un PIC a una velocità diversa. Come possiamo vedere, se la velocità del microcontroller diminuisce, lo fa anche il parametro dell'istruzione, dato che le istruzioni stesse impiegano di più a essere eseguite; se invece la velocità raddoppia, anche il parametro dovrà essere aumentato nella stessa proporzione, dato che ogni istruzione impiega la metà a essere eseguita. Dobbiamo considerare inoltre che le istruzioni in LetPicBasic sono una specie di "istruzioni composte", ognuna di esse, cioè, assemblando il programma, si scompone in varie istruzioni macchina. In alcuni casi questa particolarità non influenza la funzionalità del programma, ma il discorso cambia quando si devono gestire dei tempi: l'istruzione "delayus (1)", ad esempio, lavorando a 10 MHz genera un ritardo di 1 microsecondo, lo stesso valore di ritardo però non si può ottenere lavorando a 4 MHz, dato che la regola del tre ci dà 0,4 come parametro. In questo modo, la risoluzione minima (cioè il tempo minimo di ritardo ottenibile) con questa istruzione lavorando a 4 MHz è di 4 µs. Un valore minore, sarebbe "coperto" dal tempo di esecuzione dell'istruzione stessa, che è maggiore del tempo di ritardo che avremmo voluto ottenere; in un caso



Bisogna tener conto che il LetPicBasic assume per default una velocità di lavoro di 10 MHz.