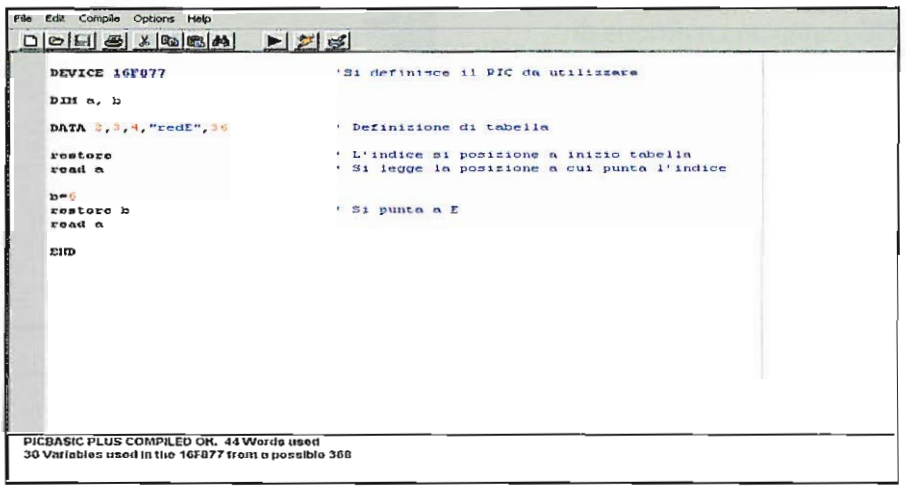
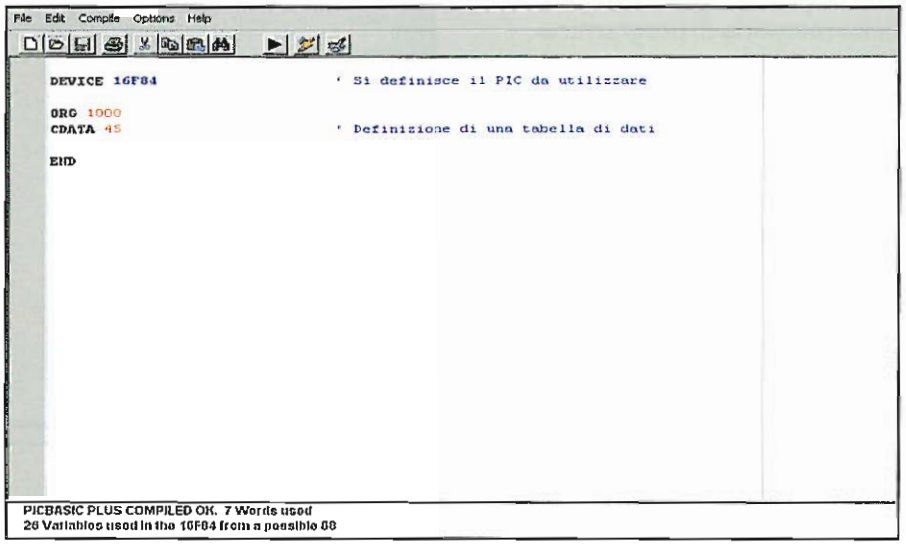
Tabelle nella memoria di programma

Abbiamo già visto i vantaggi che fornisce il lavoro con le tabelle, nella gestione di alcuni dati e le istruzioni che forniva il LetPicBasicLite a questo scopo. Il LetPicBasicPlus utilizza le stesse istruzioni e in modo uguale: DATA definisce la tabella dei dati alfanumerici, RESTORE colloca l'indice all'interno della tabella iniziando dal valore 0, e READ legge il dato che in quel momento è puntato dall'indice. In questo modo, i programmi che abbiamo in precedenza realizzato con il LetPicBasicLite, li possiamo passare al LetPicBasicPlus senza modificarli.



Le istruzioni di gestione delle tabelle sono uguali nel LetPicBasicLite e nel LetPicBasicPlus.



La memoria di codice non si può scrivere tramite programma in tutti i PIC. Però il compilatore non ci avvisa di questo.

Memoria di programma per immagazzinamento di dati

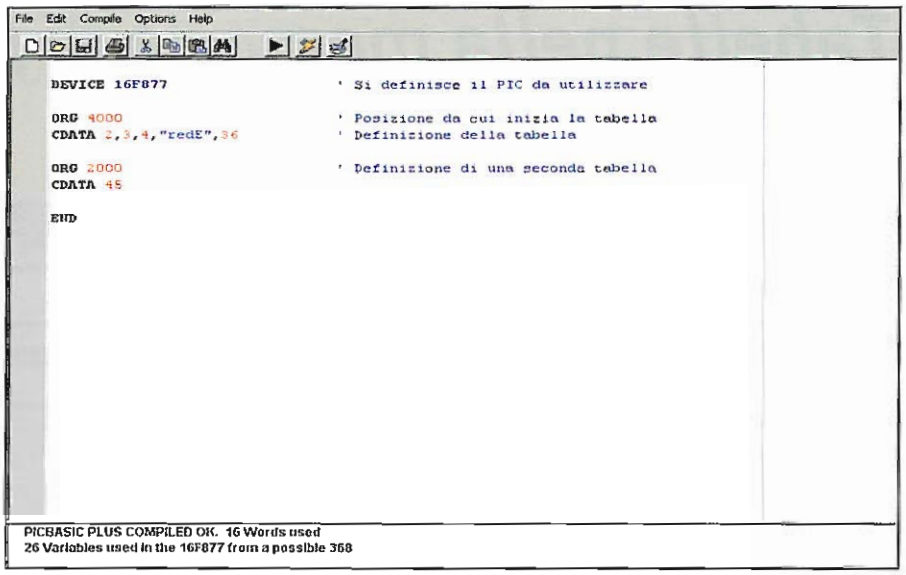
Sinora abbiamo commentato le memorie disponibili per l'immagazzinamento dei dati scoprendo i vantaggi e gli inconvenienti di utilizzare le memorie RAM e EEPROM.

Tuttavia i PIC più recenti prevedono la lettura e la scrittura della memoria FLASH di programma, come se si trattasse di una qualsiasi altra memoria. Anche se queste memorie non sono uguali alle altre, come si verificherà chiaramente lavorando con esse in assembler, facendolo in BASIC risulterà ugualmente semplice. Per cui le istruzioni per la gestione delle tabelle sono simili a quelle già viste.

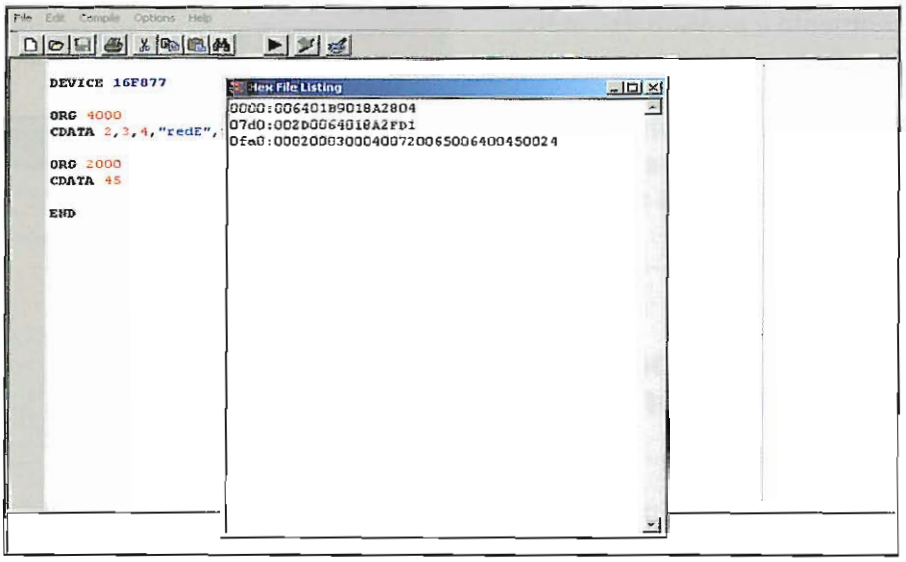
Creazione di una tabella di dati

L'istruzione per la creazione di una tabella di dati è CDATA, allo stesso modo di DATA accetta come

parametri dati alfanumerici chiusi fra virgolette (") e dati numerici senza virgolette, separati gli uni dagli altri mediante virgole. Tuttavia una differenza fondamentale è che non tutti i PIC hanno una memoria di programma che si può scrivere in questo modo, quindi mentre DATA si può utilizzare, ad esempio, con il PIC16F84 e con il PIC16F877, CDATA si può utilizzare solamente con quest'ultimo. Il compilatore però non genera alcun errore se si utilizza un PIC quando invece bisognerebbe utilizzarne un altro; tanto meno ci avviserà se scriveremo su di una cella di memoria che è occupata da una parte di programma. Il posizionamento su di un indirizzo anziché un altro, si realizza con la direttiva ORG. Il numero che appare di seguito è la posizione della memoria di programma come nel linguaggio assembler, in questo modo sarà possibile definire più di una tabella di dati. Ognuna in un indirizzo, diversamente da cosa succedeva con DATA che era unica. Per sapere se l'indirizzo che vogliamo utilizzare è libero o meno, la cosa migliore è vedere la videata del file HEX che ci mostra il codice già compilato, con le istruzioni che occupano la loro dimensione reale una volta tradotte.



Definizione corretta di una tabella di dati.



Si può vedere il file HEX per verificare le posizioni libere della memoria di codice.

Lettura di una tabella di dati

Il nome dell'istruzione di lettura di queste tabelle di dati è simile a quello di una tabella normale, però non il suo utilizzo. Si tratta di CREAD, e qui dobbiamo specificare sia la variabile su cui memorizzare il dato che l'indirizzo del dato stesso.

In realtà possiamo accedere a qualsiasi indirizzo del programma anche se non fa parte di una tabella, dato che l'indirizzo può essere uno qualsiasi. Per leggere diversi indirizzi consecutivi della memoria, senza la necessità di scrivere un'istruzione per ognuno di essi, è possibile realizzare un ciclo ripetitivo con un indice riferito a una variabile. In questa istruzione a seconda della dimensione della variabile a cui si assegna la lettura, si leggeranno più o