introdotte delle istruzioni SLEEP al posto di DELAY. In questo modo si ottiene un importante risparmio di energia, dato che durante questo tempo il microcontroller non fa nulla. Inoltre questa istruzione è indipendente dall'oscillatore del sistema, quindi non serve dichiarare il valore di quest'ultimo. In assembler esiste un'istruzione con lo stesso nome, che però non necessita di parametri; anch'essa, infatti, porta il microcontroller in stato di riposo, ma la sua durata non viene specificata, si prolunga sino a quando si produce un determinato evento. Se non si specifica il tempo di riposo, si assume che si stia utilizzando l'istruzione assembler, della quale parleremo in un capitolo a parte. Abbiamo una seconda istruzione con cui portare il microcontroller nello stato di riposo: SNOOZE. Il suo utilizzo è simile a SLEEP, però il consumo non si riduce in egual modo (si abbassa fino a 50 mA). La principale differenza è che i tempi durante il quale il microcontroller può restare in riposo sono più ristretti e l'unica misura sono i millisecondi. I valori possibili del parametro (periodo) vanno da 0 a 7, quindi il tempo totale è dato dalla formula $2^{periodo} * 18$ ms. I valori risultanti sono riportati nella tabella in alto.

PERIODO	$2^{periodo}$	TEMPO TOTALE (in ms)
0	1	18
1	2	36
2	4	72
3	8	144
4	16	288
5	32	576
6	64	1152
7	128	2304

Tabella dei valori utilizzati nell'istruzione SNOOZE.

```
DIM     VAR as BYTE

VAR=4

SYMBOL  LED=PORTB.0             'Dichiarazione di un LED sul bit 0 della PORTB

IF VAR>5 THEN   STOP            'Quest'istruzione impedisce che si esegua il
                                'codice che troviamo di seguito

LOOP:   HIGH    LED             'Si accende il LED
        SNOOZE  6               'Rimane in riposo 1152 ms
        LOW     LED             'Si spegne il LED
        SNOOZE  VAR             'Un altro intervallo uguale di riposo
        GOTO    LOOP            'Inizia nuovamente
```

```
PICBASIC PLUS COMPILED OK. 11 Words used
27 Variables used in the DEFAULT 16F84 from a possible 68
```

A partire dall'istruzione "stop" il codice non viene eseguito.

```
DIM     VAR as BYTE

VAR=4

SYMBOL  LED=PORTB.0             'Dichiarazione di un LED sul bit 0 della PORTB

END                             'Dopo questa istruzione non si compila nulla

LOOP:   HIGH    LED             'Si accende il LED
        SNOOZE  6               'Rimane in riposo 1152 ms
        LOW     LED             'Si spegne il LED
        SNOOZE  VAR             'Un altro intervallo uguale di riposo
        GOTO    LOOP            'Inizia nuovamente
```

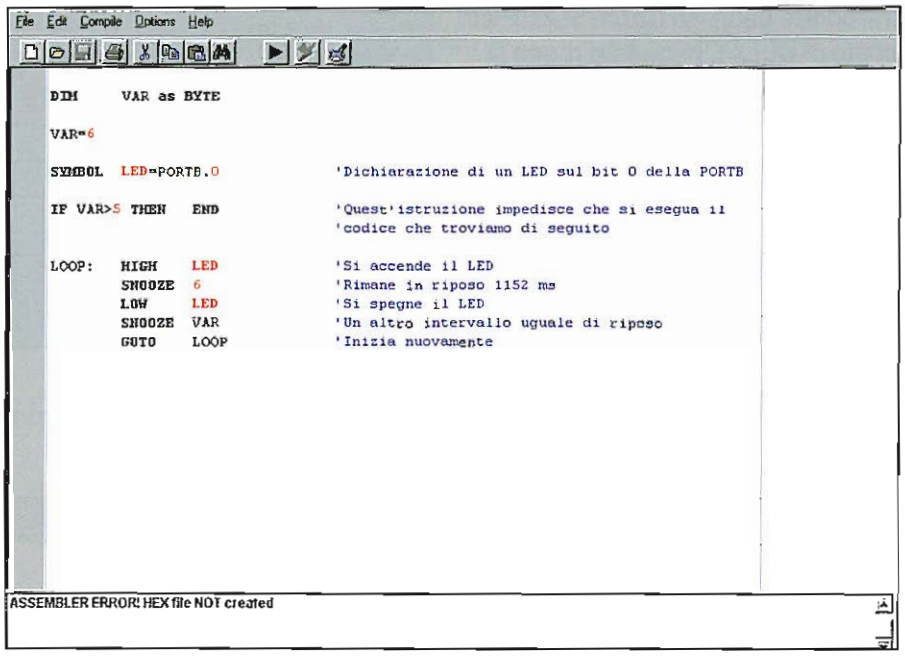
```
PICBASIC PLUS COMPILED OK. 8 Words used
27 Variables used in the DEFAULT 16F84 from a possible 68
```

Il codice a partire dall'istruzione "end" non viene compilato.

Arresto completo del microcontroller

Se desideriamo un arresto completo del sistema, non per un tempo definito, ma infinito, l'istruzione da utilizzare, come nel LetPicBasicLite, è STOP. In realtà con essa non si ferma l'esecuzione del programma, ma si fa in modo che il programma entri in un ciclo infinito senza più eseguire le istruzioni successive, con un conseguente spreco di energia. L'unico modo di uscire da questo stato è con un reset del sistema da parte dell'utente. Nel programma della figura, dato che l'istruzione "if" si compie, il programma rimarrà fermo dopo l'istruzione "stop" e non verrà eseguito il ciclo

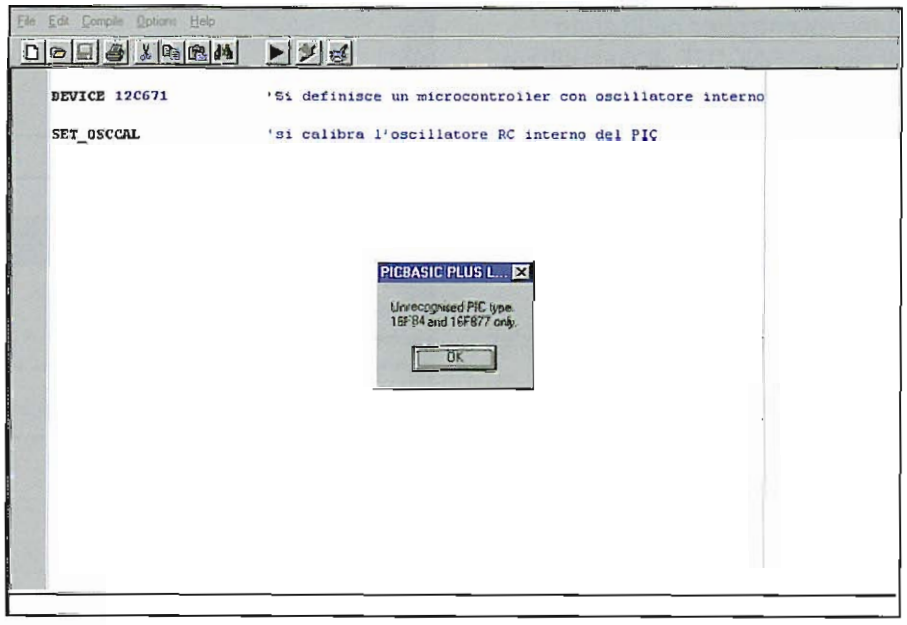
di lampeggio del LED. A differenza dell'istruzione END, con questa non solo non si esegue il codice che è scritto dopo, ma questo codice non viene neanche compilato, quindi non occupa spazio nella memoria. Anche se in alcuni casi potrebbe sembrare la stessa cosa, se facciamo attenzione al numero di word compilate con l'istruzione "stop" e "end" vedremo che può fare molta differenza. Bisogna tenere presente, quindi, che con "stop" si compila tutto il codice, ed è durante l'esecuzione che si decide se eseguire parte del programma o meno; con l'istruzione "end", invece, si decide se compilare o no, quindi in nessun caso si può inserire questa istruzione in una comparazione, altrimenti il software segnalerà un errore.



L'istruzione "end" non può fare parte di una comparazione.

Calibrazione del clock interno

L'ultima istruzione che abbiamo nel LetPicBasicPlus, relativa al tempo, è la SET_OSCCAL. Questa istruzione non è valida per tutti i microcontroller, dato che fa riferimento a un oscillatore RC interno presente solo in alcuni tipi, come quelli della serie PIC12C67x e PIC16F62x, con i quali si può lavorare con questo oscillatore al posto di quello esterno. A questo scopo, questi dispositivi hanno nell'ultima cella delle istruzioni un fattore di calibrazione che bisogna portare sul registro OSCCAL. Con l'istruzione "set_osccal" si muove questo valore automaticamente. Per questo di solito l'istruzione viene posta all'inizio di ogni programma come mostrato nella figura. Bisogna tener presente il fatto che se si cancella il microcontroller, si cancella anche il fattore di calibrazione, quindi dovrà essere letto prima per assegnarlo completamente al registro OSCCAL. Il LetPicBasicPlus non permette l'utilizzo di questa istruzione, dato che nella versione di prova non sono inseriti questi PIC.



Utilizzo dell'istruzione "set_osccal".