

Basic per PIC

esempio, l'esercizio della figura successiva. Si inizia come nei programmi precedenti però si introduce una richiesta sul valore del bit 0 della PortaA.

A seconda se vale 1 o 0 si va da una parte del programma o dall'altra. Dopo ogni ciclo è stata posta un'istruzione STOP, in modo che non si prosegua eseguendo il codice che appare dopo di essa.

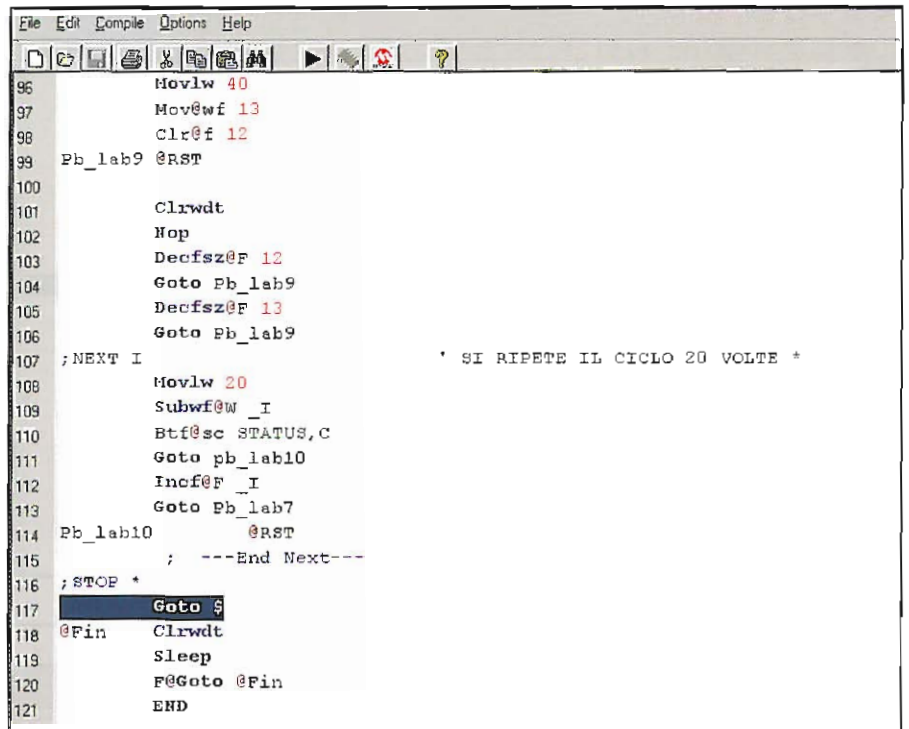
Tuttavia, se facciamo la stessa cosa fatta precedentemente, e apriamo il codice assembler generato, vedremo che il programma è stato compilato completamente.

L'istruzione è stata tradotta in una "goto \$", ciò che succederà durante l'esecuzione, sarà che se si arriva in questa parte del programma, si resta fermi alla stessa istruzione in modo permanente. Si può cambiare l'istruzione STOP in un'altra che già conosciamo, la "goto", visto che l'istruzione STOP si comporta allo stesso modo della "goto" per inserirci in un ciclo infinito su se stessa.

Tuttavia, anche se si produce lo stesso risultato, la compilazione del programma non sarà identica.

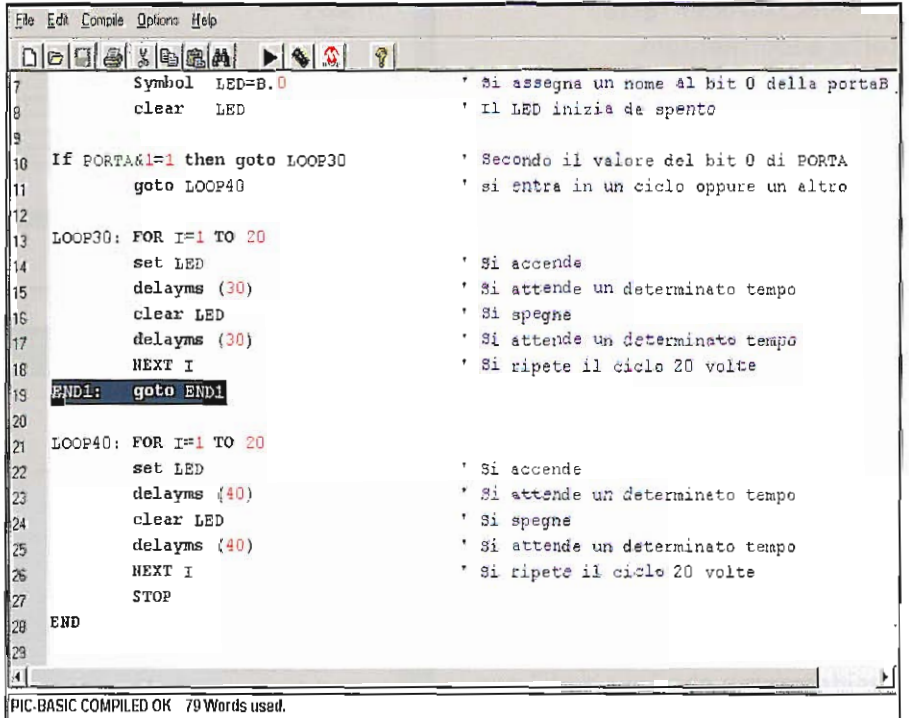
Arresto del microcontroller

La terza istruzione per fermare un programma è l'istruzione SLEEP. Questa istruzione non solo ferma l'esecuzione di un programma, come la precedente, ma porta il microcontroller in quello che si chiama "stato di riposo"



```
File Edit Compile Options Help
[Icons]
96      Movlw 40
97      Mov@wf 13
98      Clr@f 12
99      Pb_lab9 @RST
100
101      Clrwdt
102      Hop
103      Decfsz@F 12
104      Goto Pb_lab9
105      Decfsz@F 13
106      Goto Pb_lab9
107 ;NEXT I          * SI RIPETE IL CICLO 20 VOLTE *
108      Movlw 20
109      Subwf@W _I
110      Btf@sc STATUS,C
111      Goto pb_lab10
112      Incf@F _I
113      Goto Pb_lab7
114      Pb_lab10 @RST
115      ; ---End Next---
116 ;STOP *
117      Goto $
118 @Fin      Clrwdt
119           Sleep
120           F@Goto @Fin
121           END
```

L'istruzione STOP non è un'istruzione che verrà compilata.



```
File Edit Compile Options Help
[Icons]
7      Symbol LED=B.0          * Si assegna un nome al bit 0 della portaB
8      clear LED                * Il LED inizia da spento
9
10     If PORTA&1=1 then goto LOOP30 * Secondo il valore del bit 0 di PORTA
11     goto LOOP40              * si entra in un ciclo oppure un altro
12
13     LOOP30: FOR I=1 TO 20
14         set LED                * Si accende
15         delays (30)            * Si attende un determinato tempo
16         clear LED              * Si spegne
17         delays (30)            * Si attende un determinato tempo
18     NEXT I                    * Si ripete il ciclo 20 volte
19     END1: goto END1
20
21     LOOP40: FOR I=1 TO 20
22         set LED                * Si accende
23         delays (40)            * Si attende un determinato tempo
24         clear LED              * Si spegne
25         delays (40)            * Si attende un determinato tempo
26     NEXT I                    * Si ripete il ciclo 20 volte
27     STOP
28     END
29
30     PIC-BASIC COMPILED OK 79 Words used.
```

Si può sostituire l'istruzione STOP con una "goto".

```
File Edit Compile Options Help
Goto Pb_lab5
;NEXT I          * SI RIPETE IL CICLO 20 VOLTE *
  Movlw 20
  Subwf@w _I
  Btfsc STATUS,C
  Goto pb_lab6
  Incf@F _I
  Goto Pb_lab3
Pb_lab6 @RST
; ---End Next---
END1 RESET@BANK
  Goto END1
LOOP40 RESET@BANK
; FOR I=1 TO 20 *
  Movlw 1
  Mov@wf _I
Pb_lab7 @RST
;SET LED          * SI ACCENDE *
  Set@Bit PORTE,00
; DELAYS (40)     * SI ATTENDE UN DETERMINATO TEMPO *
  Movlw 40
  Mov@wf 13
  Clr@f 12
```

STOP: fermando l'esecuzione con SLEEP si ottiene un risparmio di energia, dato che si ferma l'esecuzione, mentre con STOP non rimane realmente fermo, ma continua a eseguire delle istruzioni — anche se si tratta sempre della stessa — consuma ugualmente corrente. Inoltre, dopo che si è fermato a una istruzione STOP, si può ripartire solo facendo un reset del sistema, mentre da una istruzione SLEEP si può uscire mediante un interrupt. Il tema dell'interrupt verrà trattato in un altro capitolo.

Due istruzioni con lo stesso utilizzo, possono avere compilazioni diverse.

o "stato di basso consumo".

In questo modo di funzionamento il microcontroller resta come addormentato.

Le linee di ingresso/uscita non cambiano di valore rimanendo come sono, e cessano di essere eseguite le istruzioni. Il consumo passa così da circa 2 mA del funzionamento normale, a meno di 10 µA.

Questo risparmio di energia è raccomandato nei sistemi che stanno molto tempo in attesa, ad esempio, che venga premuto il pulsante di una tastiera come succede con i telecomandi: il sistema rimane "addormentato" sino a che noi desidereremo che "si svegli" per continuare il suo funzionamento abituale.

Questa è la prima differenza rispetto all'istruzione

```
File Edit Compile Options Help
Symbol LED=B.0
clear LED
If PORTA0=1 then goto LOOP30
goto LOOP40
LOOP30: FOR I=1 TO 20
  set LED
  delays (30)
  clear LED
  delays (30)
NEXT I
SLEEP
LOOP40: FOR I=1 TO 20
  set LED
  delays (40)
  clear LED
  delays (40)
NEXT I
SLEEP
END
C-BASIC COMPILED OK 79 Words used.
```

Utilizzo dell'istruzione SLEEP in un programma.

```
File Edit Compile Options Help
Wop
Decfsz@F 12
Goto pb_lab5
Decfsz@F 13
Goto Pb_lab5
;NEXT I          * SI RIPETE IL CICLO 20 VOLTE *
  Movlw 20
  Subwf@w _I
  Btfsc STATUS,C
  Goto pb_lab6
  Incf@F _I
  Goto Pb_lab3
Pb_lab6 @RST
; ---End Next---
Sleep
LOOP40 RESET@BANK
; FOR I=1 TO 20 *
  Movlw 1
  Mov@wf _I
Pb_lab7 @RST
;SET LED          * SI ACCENDE *
  Set@Bit PORTE,00
; DELAYS (40)     * SI ATTENDE UN DETERMINATO TEMPO *
```

L'istruzione basic SLEEP si traduce automaticamente nella SLEEP dell'assembler.