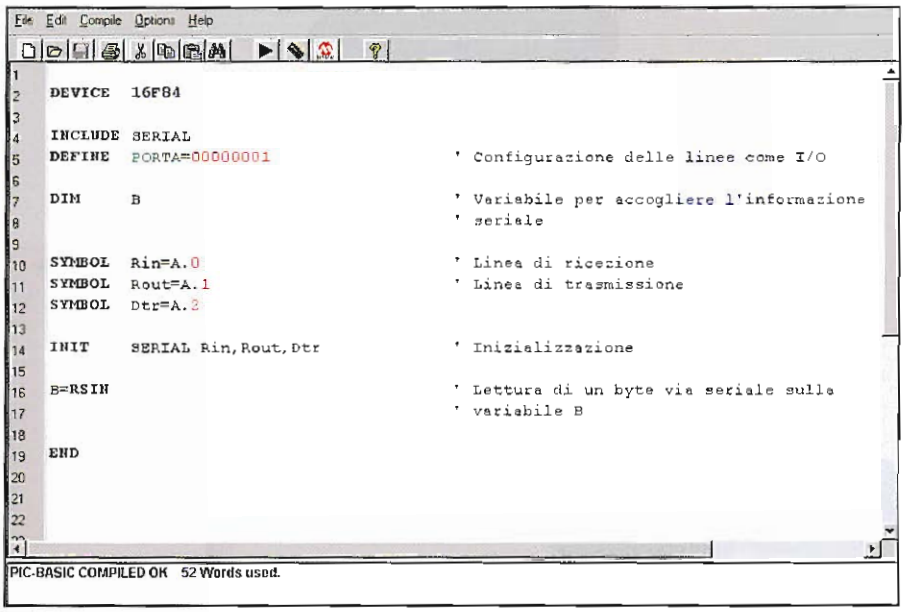
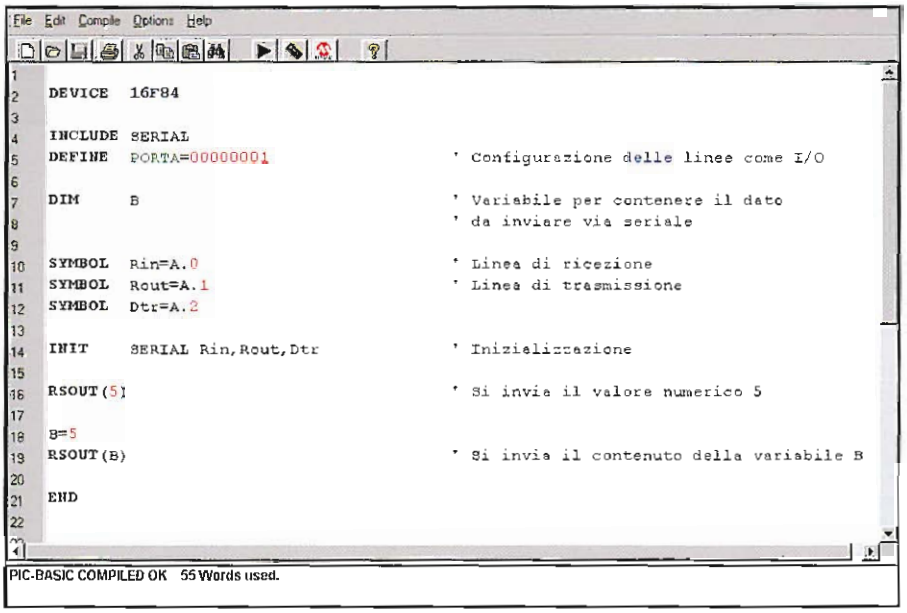


```
File  Edit  Compile  Options  Help

1
2   DEVICE   16F84
3
4   INCLUDE  SERIAL
5   DEFINE   PORTA=00000001                ' Configurazione delle linee come I/O
6
7   DIM      B                             ' Variabile per accogliere l'informazione
8                                          ' seriale
9
10  SYMBOL   Rin=A.0                       ' Linea di ricezione
11  SYMBOL   Rout=A.1                      ' Linea di trasmissione
12  SYMBOL   Dtr=A.2
13
14  INIT     SERIAL Rin,Rout,Dtr           ' Inizializzazione
15
16  B=RSIN                                 ' Lettura di un byte via seriale sulla
17                                         ' variabile B
18
19  END
20
21
22

PIC-BASIC COMPILED OK   52 Words used.
```

Acquisizione di un dato via seriale, con l'istruzione RSIN.



```
File  Edit  Compile  Options  Help

1
2   DEVICE   16F84
3
4   INCLUDE  SERIAL
5   DEFINE   PORTA=00000001                ' Configurazione delle linee come I/O
6
7   DIM      B                             ' Variabile per contenere il dato
8                                          ' da inviare via seriale
9
10  SYMBOL   Rin=A.0                       ' Linea di ricezione
11  SYMBOL   Rout=A.1                      ' Linea di trasmissione
12  SYMBOL   Dtr=A.2
13
14  INIT     SERIAL Rin,Rout,Dtr           ' Inizializzazione
15
16  RSOUT(5)                               ' Si invia il valore numerico 5
17
18  B=5
19  RSOUT(B)                               ' Si invia il contenuto della variabile B
20
21  END
22

PIC-BASIC COMPILED OK   55 Words used.
```

Trasmissione di un dato seriale con l'istruzione RSOUT.

comunicazione seriale, quindi questa comunicazione si fa via software, grazie alle istruzioni speciali. Bisognerà includere le routines corrispondenti con il comando INCLUDE e, prima di inizializzare il canale seriale, specificare le linee del PIC che si vogliono utilizzare. In questo caso, sono stati scelti i piedini RA0 per la ricezione dei dati e RA1 per la trasmissione, quindi dovranno essere definiti rispettivamente come ingresso e uscita.

Con SYMBOL si assegna un nome simbolico per avere una visione più chiara del funzionamento. DTR è una linea opzionale che possiamo omettere se non viene utilizzata, ma che dobbiamo definire come linea di uscita, se pensiamo di utilizzarla. Nell'esempio è stata dichiarata sul piedino RA2. Come possiamo vedere, anche se il programma per ora non fa nulla si può compilare senza errori.

Istruzione per la ricezione dei dati

Delle due istruzioni che ci fornisce il LetPicBasic una è per la ricezione dei dati da un altro dispositivo al PIC, l'altra per la trasmissione dei dati in senso inverso. La prima è la RSIN. Dopo che è stato configurato in modo adeguato il canale, non resta che definire una variabile dell'utente, e assegnare a questa variabile ciò che trasmetterà l'istruzione RSIN, che sarà un byte mandato via seriale. Nella figura è stata chiamata B la variabile dell'utente.

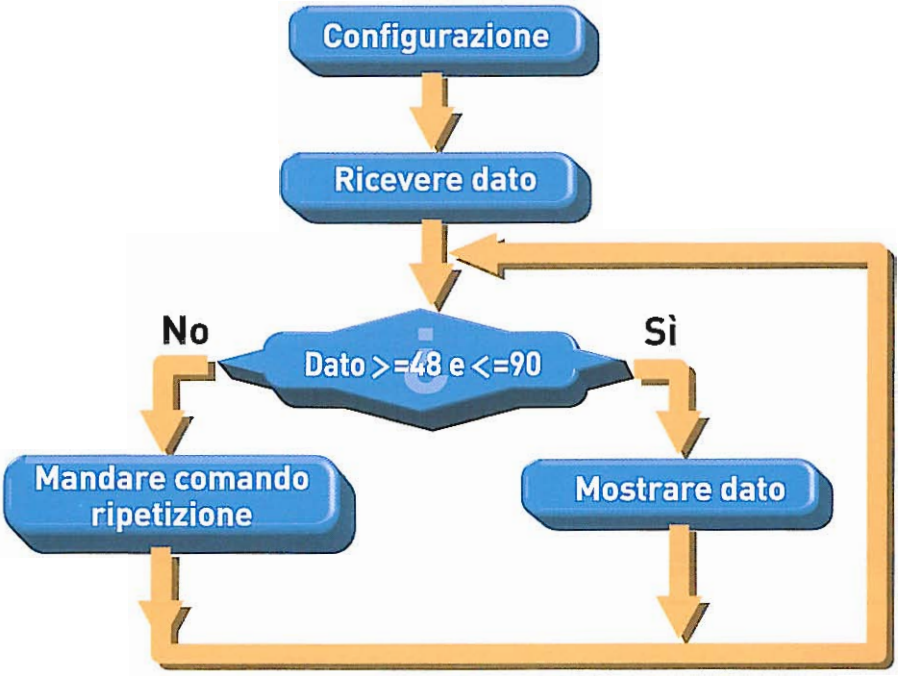
Istruzione per la trasmissione dei dati

L'utilizzo dell'istruzione per la trasmissione, è semplice come quello precedente. Con la stessa configurazione e inizializzazione che serve per l'istruzione di ricezione, si indicherà fra parentesi il dato da inviare, che può essere una variabile o un numero. Nell'esempio della figura si inviano due dati, il primo

come valore letterale, poi utilizzando una variabile che contiene lo stesso numero.

Un esempio utile

Gli esempi precedenti erano solamente indicativi di come si realizza la comunicazione seriale in entrambi i versi, però questo si può applicare in situazioni molto diverse, dove la comunicazione fra dispositivi ricopre una grande importanza. Immaginate un sistema di controllo nel quale, a seconda dei dati di ingresso, si realizzano delle istruzioni, oppure altre. Il subsistema di raccolta dei dati potrebbe essere un PC, un altro PIC o un sistema indipendente. In tutti i casi questo subsistema comunicherà con il nostro PIC via seriale, per inviare/ricevere dati e comandi. Nell'esempio mostrato nell'organigramma, il PIC riceve un dato e lo compara con i valori compresi fra 48 e 90, ipotizzati come valori validi per questa applicazione. Se il valore è valido, si visualizza, ad esempio, tramite diodi LED, e se non è valido si manda un comando per fare in modo che si ripeta la trasmissione del dato. Questo viene inserito in un ciclo infinito. Di seguito riportiamo la soluzione dell'esercizio. Come potete osservare si utilizzano strutture e istruzioni viste fino ad ora. Al posto del diodo LED si potrebbe usare il display LCD, che già conosciamo, e differenziare non solo fra un range di valori, ma assegnare a ogni valore specifico un significato particolare, e in base ad esso fare eseguire un'azione differente. Volete provare? Tenete conto che questo programma corrisponde solamente a una parte della comunicazione, dopo bisognerà progettare il programma del sistema che si trova dall'altro lato, per fare in modo che capisca e risponda correttamente ai comandi di quest'ultimo programma.



Organigramma dell'esercizio proposto.

```
File  Edit  Compile  Options  Help

DEVICE  16F84

INCLUDE SERIAL
DEFINE  PORTA=00000001          ' Configurazione delle linee come I/O
DEFINE  PORTB=00000000          ' Configurazione della PORTB come uscita
DIM     B                       ' Variabile per contenere il dato ricevuto
SYMBOL  Rin=A.0                 ' Linea di ricezione
SYMBOL  Rout=A.1                ' Linea di trasmissione
INIT    SERIAL Rin,Rout         ' Inizializzazione

BUC:    B=RSIN                  ' Si riceve un dato

        ' Se il dato ricevuto rientra nel range lo si visualizza
        IF B>=48 & B<=90 THEN GOTO L1

        ' altrimenti si invia un codice di ripetizione
        GOTO    L2

        L1:     PORTB=B         ' Si visualizza il dato ricevuto
                GOTO BUC        ' Si ritorna al ciclo principale

        L2:     RSOUT(91)       ' Si invia un codice di ripetizione
                GOTO BUC        ' Si ritorna al ciclo principale

END
```

Programma che corrisponde all'organigramma precedente.