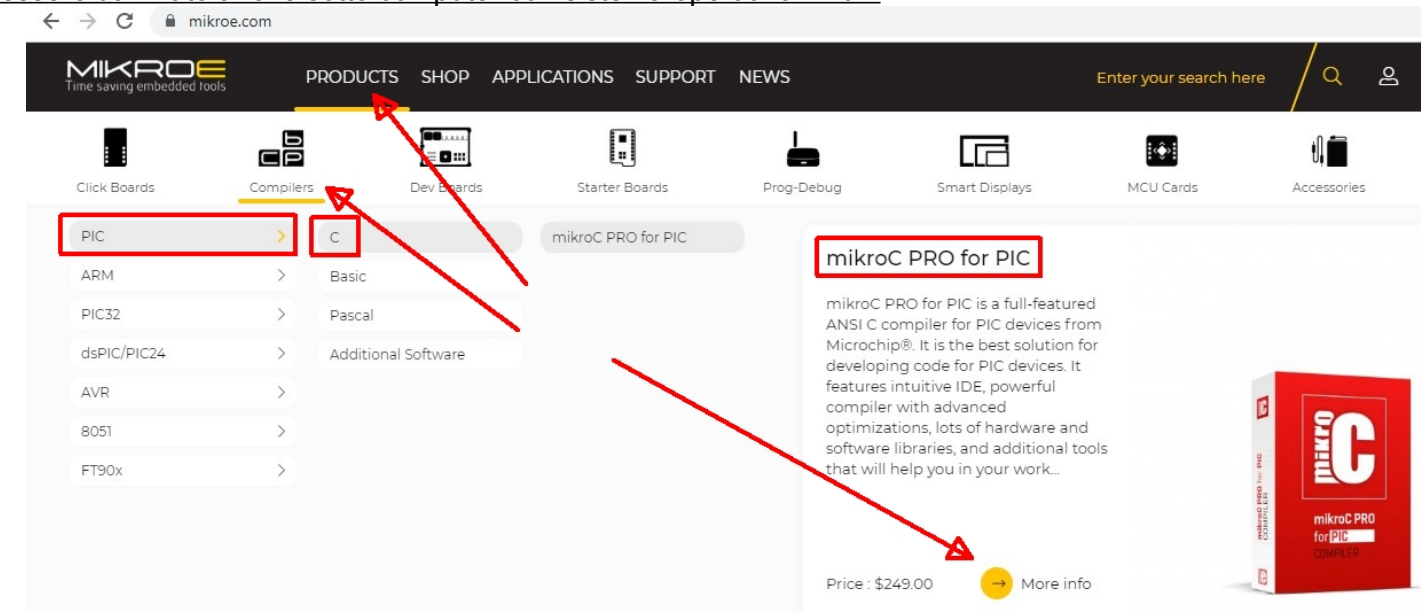
AMBIENTE DI SVILUPPO MIKROC

L'ambiente di sviluppo mikroC è realizzato e distribuito dalla ditta serba mikroelektronica.

Il software può essere scaricato al seguente link: www.mikroe.com

Purtroppo il software viene fornito solo per sistemi operativi Windows, ma installando una macchina virtuale potrà essere utilizzato anche sotto computer con sistema operativo Linux.



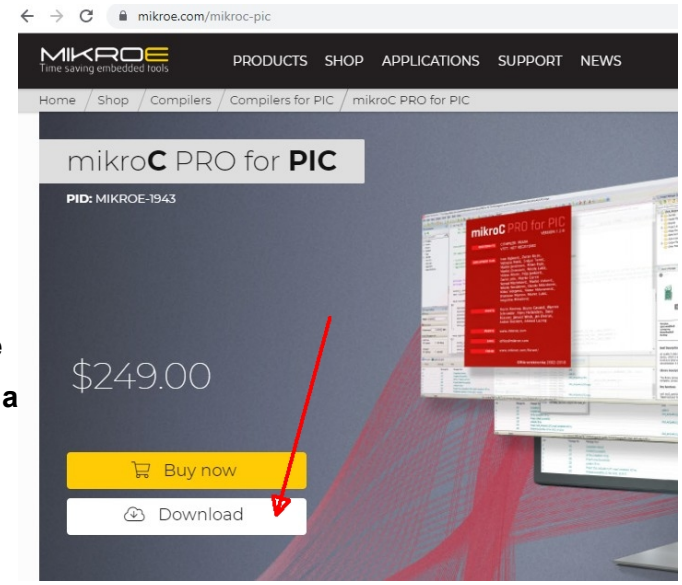
Come si può vedere nell'immagine sopra, la ditta offre degli ambenti di sviluppo per varie tipologie di microcontrollori; ***PIC, ARM, PIC32, dsPIC/PIC24, AVR,8051,FT90x.***

E si possono scegliere 3 linguaggi di programmazione; ***C, BASIC, PASCAL.***

Nel nostro caso sceglieremo di scaricare l'ambiente di sviluppo per microcontrollori **PIC** in linguaggio **C.**

Questo ambiente di sviluppo è adatto solo per I microcontrollori PIC ad 8 bit, ma la sua struttura è praticamente ugiale a quella degli altri microcontrollori, fatta eccezione per le librerie, che saranno in qualche caso diverse.

Una volta scaricato con l'apposito pulsante, il software dovrà essere installato seguendo le istruzioni a video, confermando tutte le schermate.



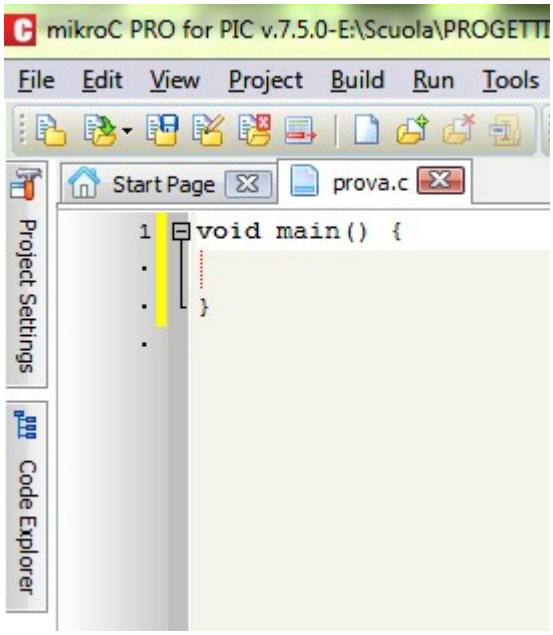
Se non si acquista la licensa, il software offre comunque tutte le potenzialità, l'unica limitazione è nelle dimensioni del file prodotto dal compilatore.

La versione demo senza licenza, consente di compilare programmi che producono file binari (.hex) fino a 2kByte.

Questo ci permetterà comunque di realizzare semplici programmi al fine di acquisire la conoscenza del software prima di acquistarlo.

Nelle precedente dispense abbiamo visto come realizzare un progetto, e per chi ha già un po' di dimestichezza, abbiamo descritto come realizzare un programma per gestire una comunicazione seriale RS232.
Ma facciamo un passo indietro, e cominciamo a vedere come strutturare e realizzare programmi più semplici.
Diamo per scontato che sia stata letta la seconda dispensa sul compilatore mikroC sulla creazione di un progetto.
Ci troviamo perciò al punto in cui bisogna cominciare a scrivere un codice in linguaggio C che esegua una qualsiasi funzione.

Dopo aver creato un progetto ci troviamo il file con estensione .C aperto, con il seguente contenuto.
In pratica c'è il solo ciclo main senza alcun codice.

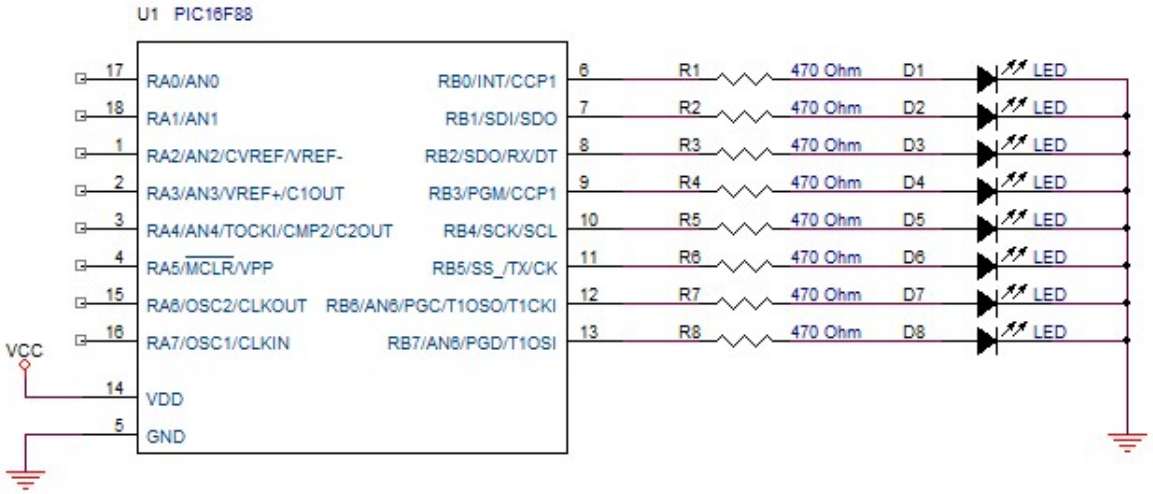


Il ciclo main, nel linguaggio C è il ciclo principale, ed una delle possibilità è quella di scrivere il nostro codice interamente nel ciclo main.

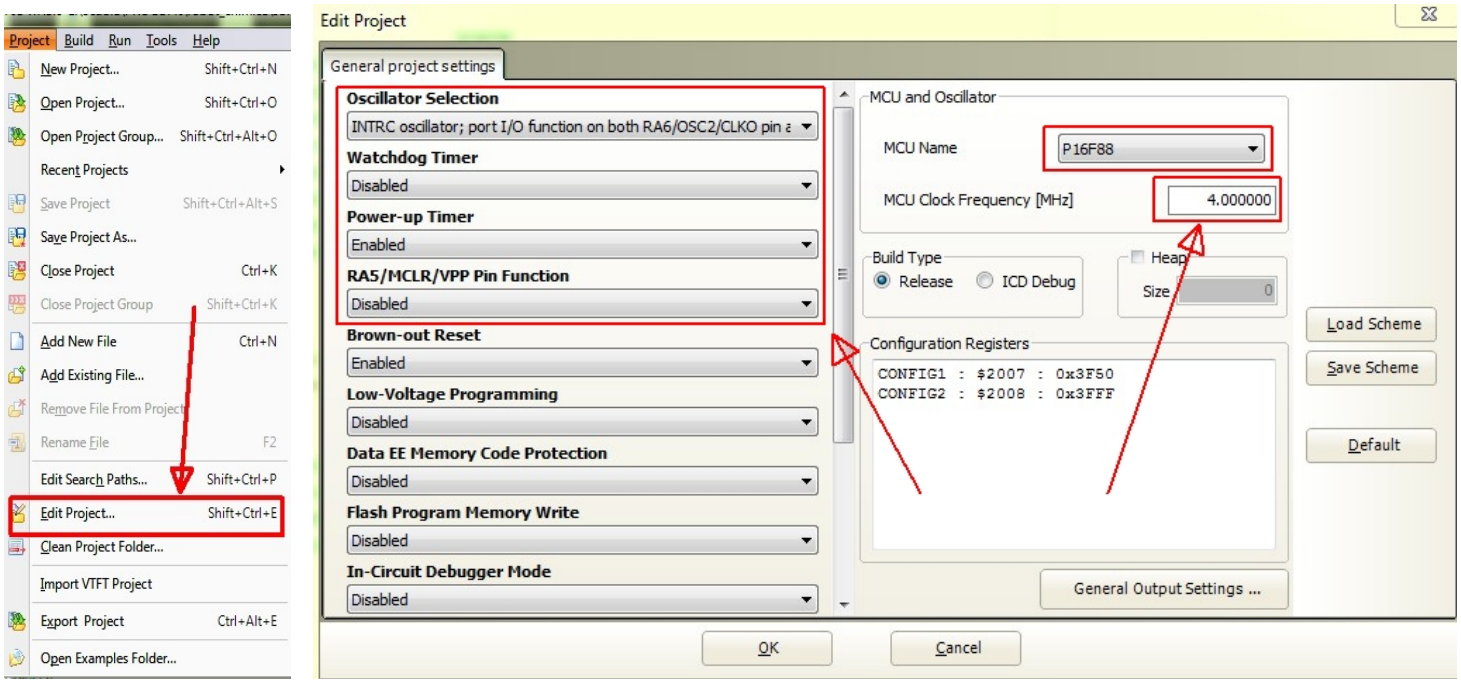
Ovviamente ciò si può fare, se il codice che scrivamo, è composto da poche righe ed istruzioni, in caso contrario, è consigliabile sviluppare delle funzioni, o addirittura aggiungere altri file contenenti del codice, che verranno però compilati tutti insieme per ottenere un solo file binario eseguibile dal microcontrollore.

Cominciamo con il caso più semplice, e cioè quello del codice scritto interamente nel main in un solo file.

Supponiamo ad esempio di sviluppare un programma che esegua una semplice operazione, e cioè accendere e spegnere continuamente le uscite della PORTB del PIC, dove magari potranno essere collegati dei LED come nel seguente schema.

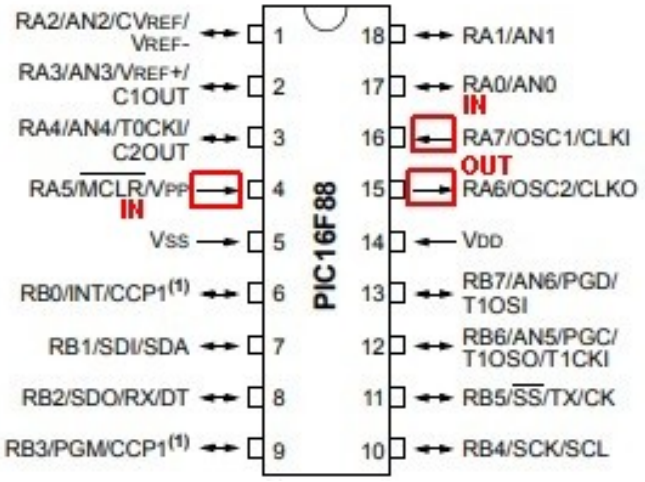


Come si può notare nello schema, oltre ai LED ed alle resistenze limitatrici, non ci sono altri componenti esterni come l'oscillatore per generare il clock, dovremo perciò ricordarci di attivare il clock interno del microcontrollore.
La definizione del progetto e della configurazione del microcontrollore, è la prima cosa che dev'esser fatta, per questo bisogna accedere al settaggio del progetto come descritto nel precedente documento.
Di seguito la schermata dei settaggi, ed evidenziati nel rettangolo rosso, quelli da modificare rispetto al default.



Come si può notare, risulta attivato il clock interno a 4MHz, di conseguenza I pin RA6 ed RA7 rimarranno utilizzabili come input/output generici, risulta anche disattivato il MasterCLeaR (MCLR) ed anche il pin RA5 sarà disponibile.

I pin RA5,RA6 ed RA7, se non utilizzati nelle funzioni di MCLR o CLOCK, possono essere utilizzati come ingressi/uscite ma a differenza degli altri pin che offrono entrambe le possibilità, questi 3 pin possono essere solo ingresso (RA5, RA7) o solo uscita (RA6) vedi figura.

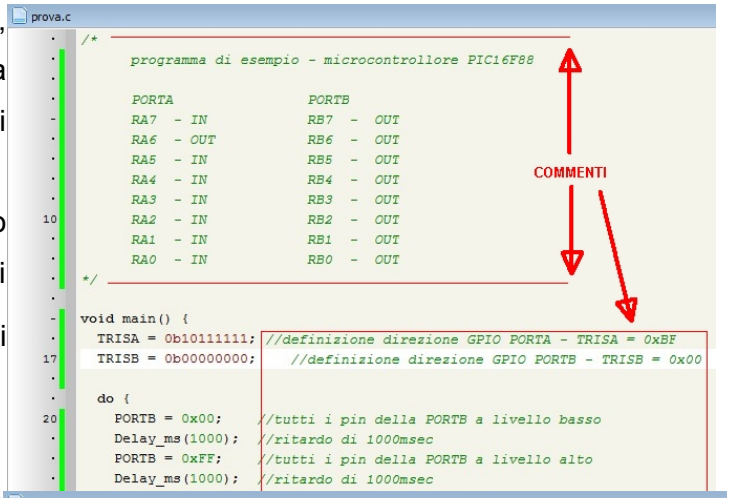


Questa configurazione sarà probabilmente utilizzata in gran parte dei programmi che seguiranno, pertanto andrà sempre ripetuta allo stesso modo.
Proseguiamo con la stesura del codice.

Come detto precedentemente, vogliamo scrivere tutto il codice dentro alla routine principale del main.

E' buona norma quando si scrive un programma, riassumere inizialmente le caratteristiche del sistema come ad esempio; il tipo di microcontrollore, le funzioni dei suoi piedini ecc….

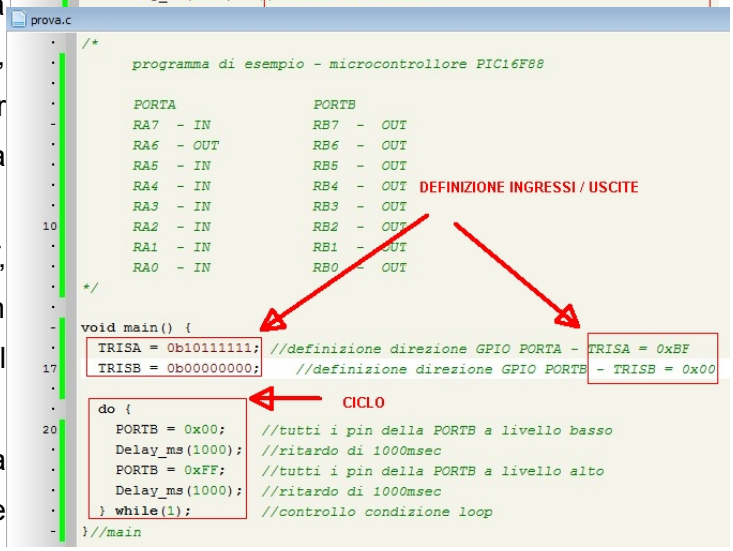
Ricordiamoci che i commenti nel linguaggio C possono essere scritti su più righe se compresi tra i caratteri di apertura ***/**** *e di chiusura* ****/*** , o possono essere scritti su una riga seguiti dai caratteri ***//*** .



La prima cosa da fare nel nostro programma, è quella di definire gli ingressi e le uscite del microcontrollore, che viene effettuata scrivendo nei registri TRISA per quanto riguarda la PORTA e TRISB per quanto riguarda la PORTB.

Ricordiamoci che se vogliamo un uscita nella PORT, dobbiamo scrivere 0 nel registro TRIS, se vogliamo un ingresso nella PORT dobbiamo invece scrivere 1 nel registro TRIS.

La PORTA non è utilizzata nel nostro schema, ma la impostiamo comunque tutta in ingresso, fatta eccezione per il pin RA6 che, come detto prima, potrà solo essere un'uscita.

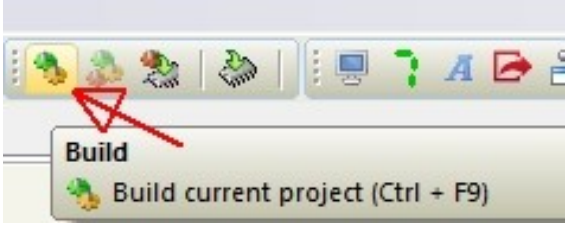


Il valore da scrivere in TRISA sarà perciò **10111111**, per esprimere questo valore in binario, dobbiamo farlo seguire dai caratteri 0b, se invece volessimo esprimerlo in esadecimale dovremmo farlo seguire dai caratteri 0x e pertanto avremmo **0xBF**.

Dopo l'impostazione degli input e degli output, dobbiamo inserire un ciclo che dovrà essere eseguito continuamente in loop, e per fare questo possiamo utilizzare un ciclo ***do….while(),*** dove la condizione deve consentire una ripetizione continua del ciclo.

Pertanto sarà sufficente mettere nel while il valore 1, come nel programma. All'interno del ciclo invece la PORTB verrà messa tutta ad 1 e tutta a 0 con un ritardo di 1sec.

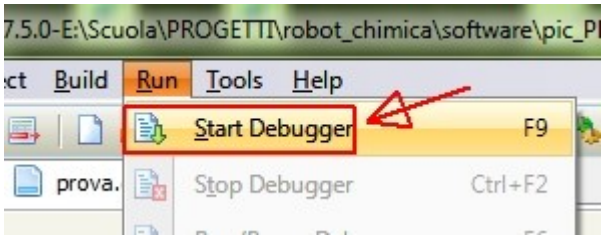
Il passo successivo è quello di compilare il programma, verificando nella finestra in basso la presenza di eventuali errori.



Prima di procedere con altri esempi,(magari diversi da questo con il codice completamente all'interno del ciclo main, situazione non consigliabile e scarsamente utilizzata per programmi mediamente complessi) vediamo come procedere con la simulazione e successivamente con la programmazione.

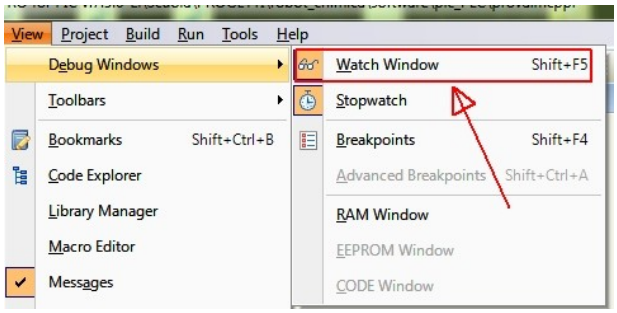
SIMULAZIONE----------

Dopo aver compilato si può procedere alla simulazione con l'apposito pulsante "**Start Debugger**".

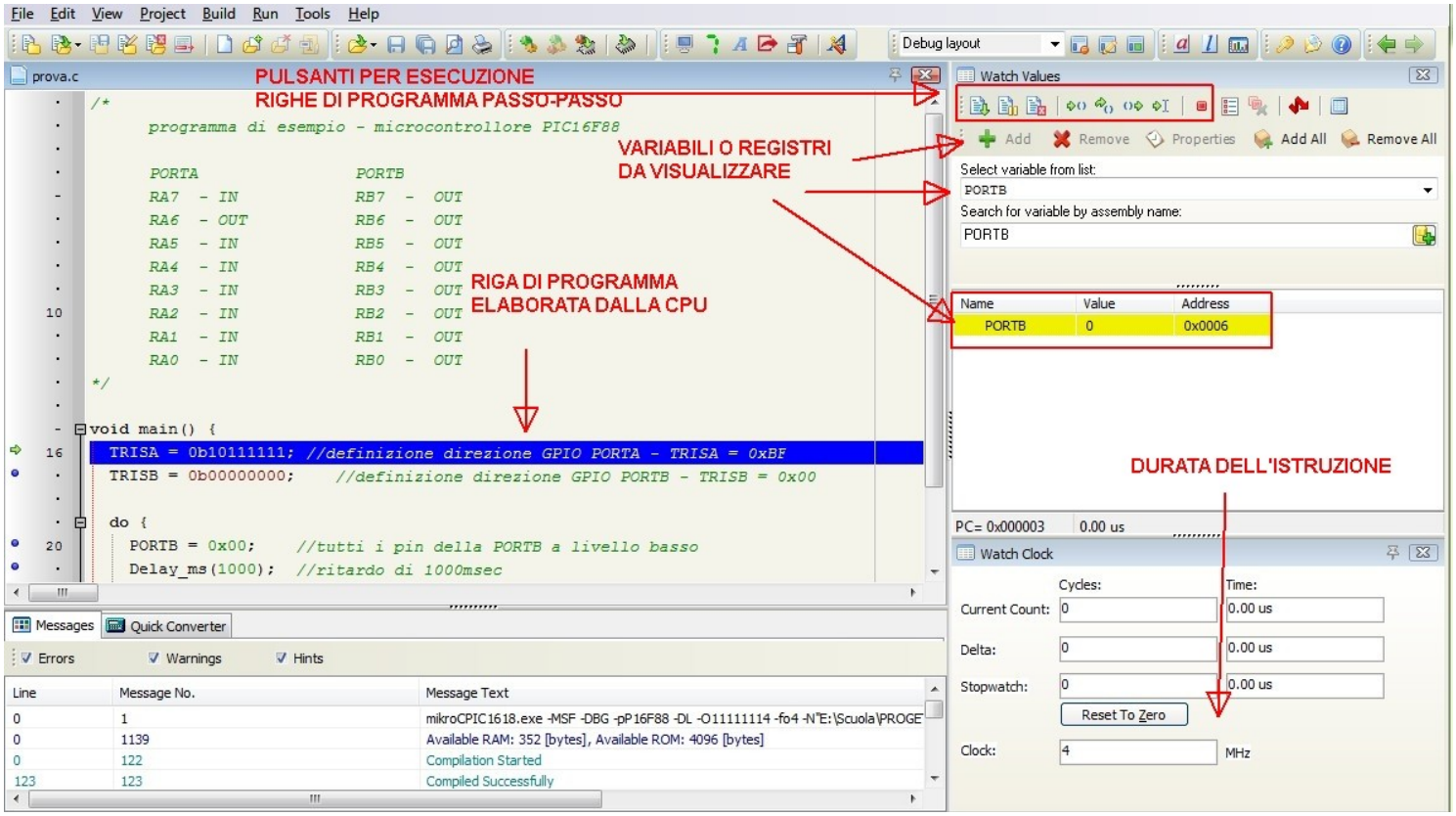


All'avvio della simulazione, si evidenzierà la prima riga del programma e si aprirà la finestra "**Watch Values**" alla destra.

Se questa finestra non si dovesse aprire in automatico, è possibile farlo dal menù ***View-Debug Windows.***



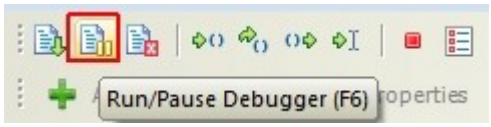
Avviata la simulazione avremo la seguente situazione.



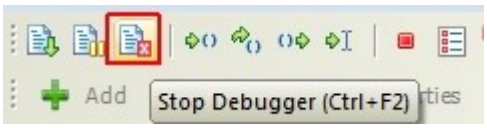
Per avviare il programma si possono utilizzare i pulsanti indicati nell'immagine in alto.

- Avvio dell'esecuzione del programma.

- Programma in pausa.



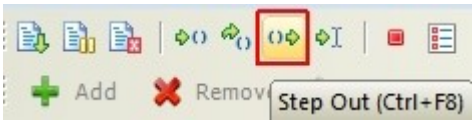
- Stop della simulazione.



- Esecuzione di un'istruzione, con entrata in una eventuale subroutine o funzione.

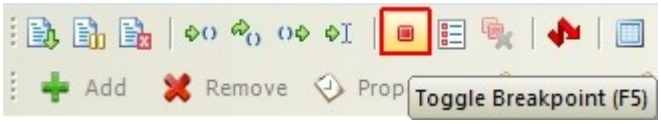
- Esecuzione di un'istruzione, saltando l'entrata nell'eventuale subroutine o funzione.

- Esecuzione di un'istruzione, uscendo dalla funzione o routine.

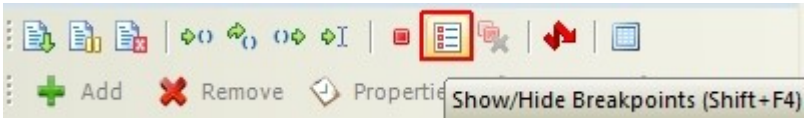


- Esecuzione del programma fino alla posizione del cursore.

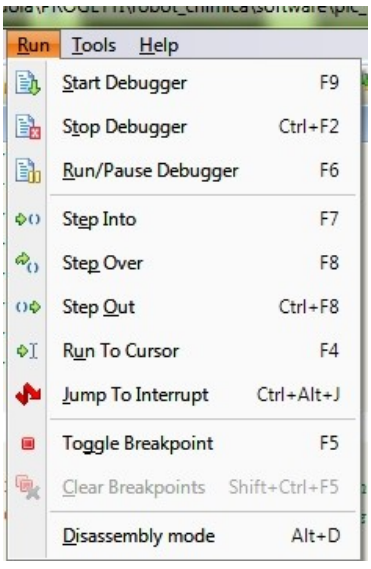
- Inserimento di un breakpoint, cioè un punto di interruzione all'interno dei programa.



- Visualizzazione breakpoint inseriti.



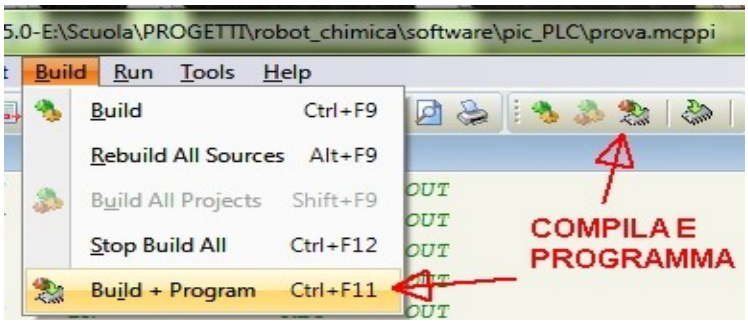
Gli stessi pulsanti sono raggiungibili dal menù "**Run**".



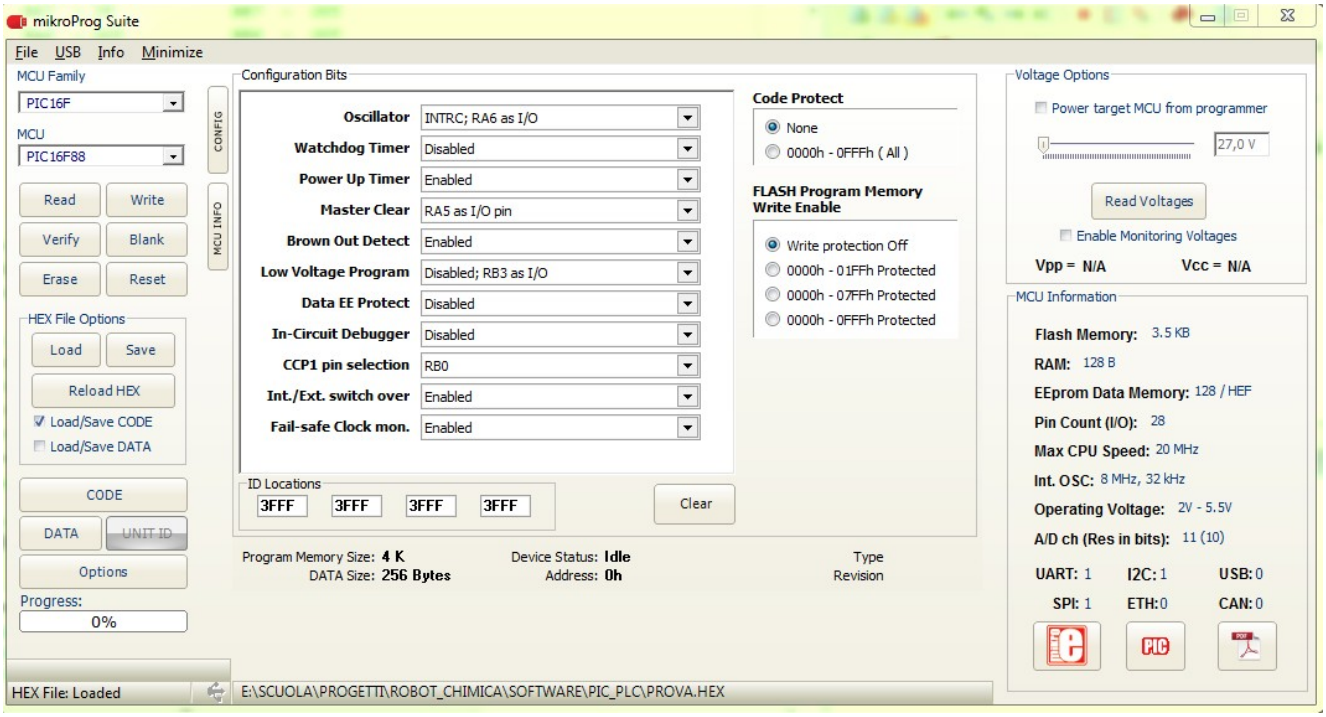
Gestendo i pulsanti sopra descritti, si può simulare passo-passo, o per blocchi, il programma, verificando in ogni occasione lo stato dei registri e delle variabili.

PROGRAMMAZIONE----------

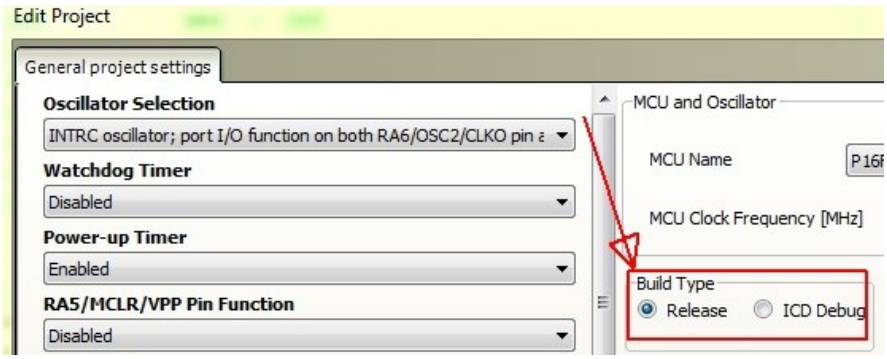
Dopo aver collegato il programmatore della ditta Mikroelektronika, si procede alla programmazione con l'apposito pulsante.



In questo caso si aprirà la seguente finestra e si avvierà la programmazione.



Nella finestra di configurazione del programma "**Edit Project**" , si può scegliere inoltre se compilare il programma in modalità **Release** o **ICD Debug.**

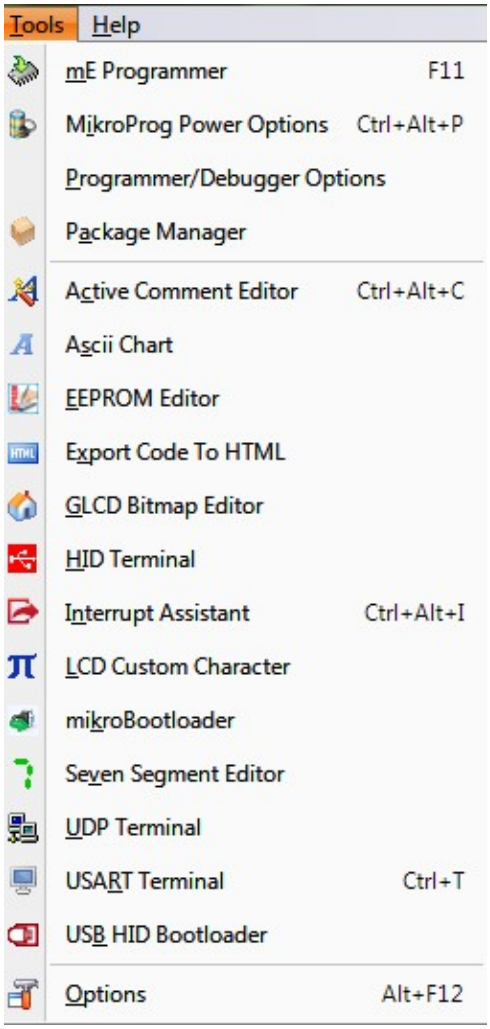


Se si utilizza la prima modalità il programma girerà sul micro in maniera normale, senza la possibilità di eseguire un debug sul microcontrollore.

Se si utilizza invece la seconda modalità, il programma invece girerà sul micro con la possibilità di eseguire un debug. Ciò significa che è possibile collegare il programmatore direttamente sul circuito con l'apposito connettore, ed utilizzando la modalità vista prima per la simulazione, si può eseguire il programma passo passo e visualizzare lo stato delle variabili e dei registri.

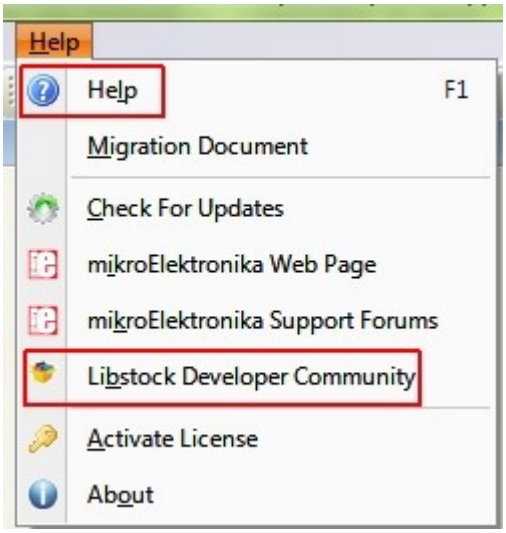
TOOLS

Sotto questo menù possiamo trovare diversi strumenti utili per sviluppare il nostro progetto. Nel corso delle successive dispense probabilmente verranno approfonditi, ma brevemente vediamone qualcuno.



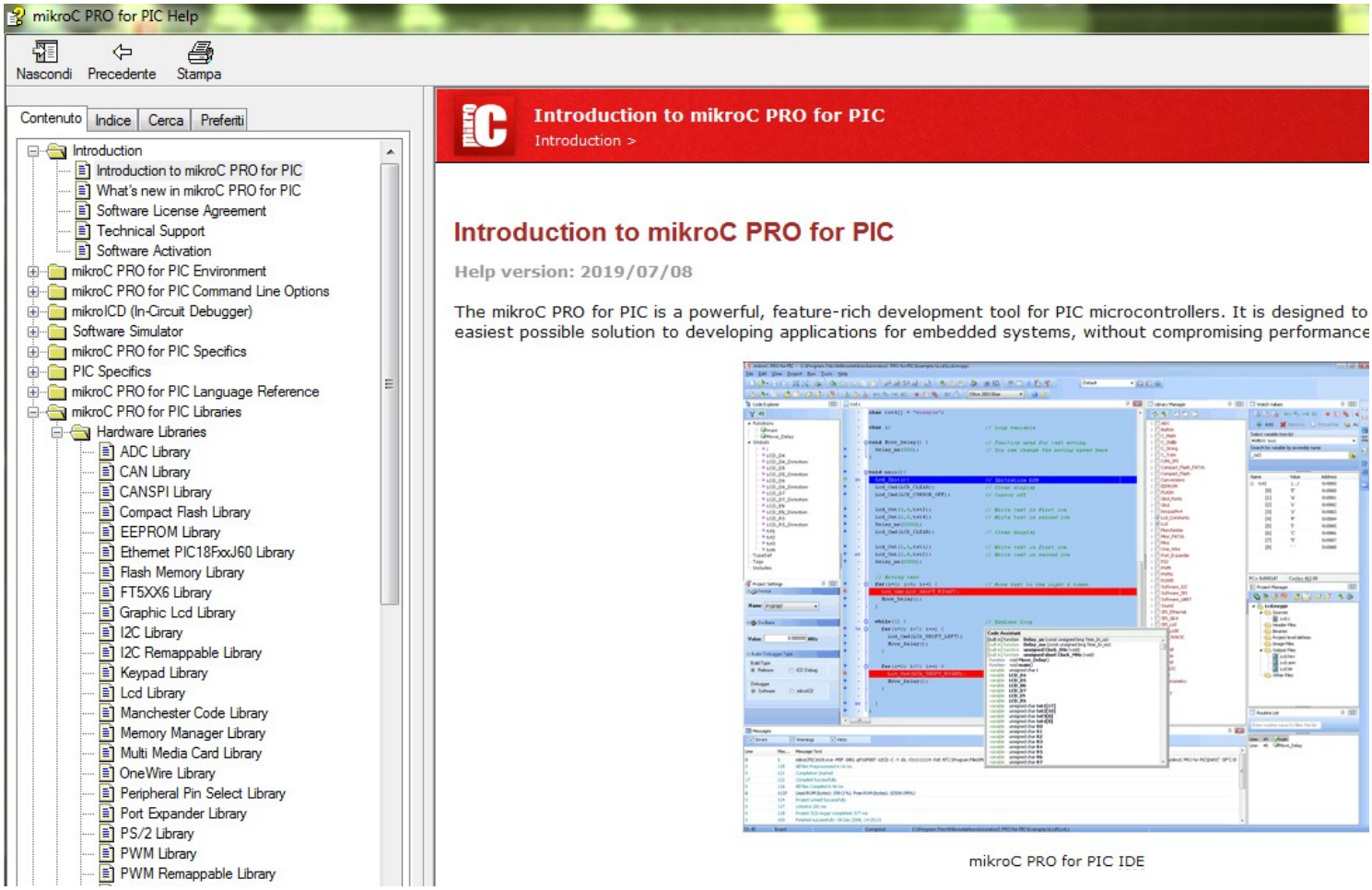
- **Active Comment Editor**
  Un semplice editor dei commenti da inserire nel programma.
- **ASCII Chart**
  E' la tabella dei caratteri ASCII molto utile in diverse occasioni.
- **EEPROM Editor**
  E' un editor per scrivere il contenuto della EEPROM in fase di programmazione del microcontrollore.
- **Export Code to HTML**
  Crea una pagina html con il codice scritto.
- **GLCD Bitmap Editor**
  Consente di caricare un'immagine per creare l'array da utilizzare nelle librerie di controllo del display grafico.
- **HID Terminal**
  Terminale per comunicazioni USB
- **Interrupt Assistant**
  Strumento per creare le routine di interrupt.
- **LCD Custom Character**
  Permette di creare caratteri personalizzati per i display LCD alfanumerici.
- **MikroBootloader e USB HID Bootloader**
  Permettono di creare un bootloader sul microcontrollore. Ciò significa inserire nel microcontrollore un programma che verrà automaticamente avviato all'accensione, il quale resterà in ascolto su una linea del micro e se riceverà gli appositi comandi su questa linea, consentirà di programmare il microcontrollore senza alcun programmatore esterno.
- **Seven Segment Editor**
  Un editor che restituisce il valore esadecimale degli 8 bit che gestiscono un display ad anodo o a catodo comune.
- **UDP Terminal**
  Uno strumento per inviare pacchetti dati con protocollo UDP.
- **USART Terminal**
  Strumento per inviare e ricevere dati sulla seriale RS232.

HELP



Una delle parti migliori di questo ambiente di sviluppo, è proprio l'help.
Oltre ai pulsanti per l'attivazione della licenza o il link al sito ed alla pagina di supporto, ci sono i due pulsanti indicati in figura.
Il primo aprirà la guida del programma con tutte le informazioni e gli esempi sulle varie librerie.
Il secondo consente l'accesso ad un interessante spazio, dove è possibile trovare progetti e librerie sviluppare dalla community mondiale.

Nella guida del programma, troviamo davvero di tutto, di seguito uno screenshot, con le varie librerie disponibili.



mikroC PRO for PIC IDE

Nei successivi tutorial, svilupperemo dei programmi più complessi, e quando utilizzate, descriveremo il funzionamento delle librerie hardware disponibili.

Chiudo questa panoramica delle varie funzioni dell'ambiente di sviluppo della mikroelektronika, in attesa dei successivi tutorial.