

mercoledì 2 gennaio 2002

Traduzione effettuata da Demaldè Gastone

Chapter 2. Tutorial - PIC16F877

2.1 Introduction

MPLAB ICD è un programmatore per la famiglia PIC16F87X, ed un "incircuit debugger". Questo tutorial ti aiuterà ad usare MPLAB ICD hardware e software per programmare e fare il debug del codice sorgente.

Questo tutorial utilizza un semplice progetto (project), Tut877. Il programma `tut877.asm` è una applicazione del convertitore analog-to-digital (A/D) del PIC16F877 che utilizza MPLAB ICD Demo Board. Questo programma configura il modulo A/D per convertire l' A/D channel 0 (connesso al potenziometro sulla MPLAB ICD Demo Board) e visualizza il risultato sui LED del PORTC.

Questo tutorial richiede l'utilizzo della MPLAB ICD Demo Board. Di conseguenza, se tu hai solo l' MPLAB ICD Module (DV164002) piuttosto che MPLAB ICD Evaluation Kit (DV164001), non puoi seguire le istruzioni di questo capitolo.

2.2 Highlights

Argomenti trattati in questo capitolo:

- Running MPLAB IDE
- Creating a Hex File for Debugging
- Setting up MPLAB ICD
- Setting Programming and Debugging Options
- Programming the PIC16F877
- Setting up the Demo Board
- Running Tut877
- Debugging Tut877
- Tut877 Main Routine
- Tut877 Corrected Source Code

2.3 Running MPLAB IDE

Dopo avere installato l' MPLAB IDE software, mandare in esecuzione il file `mplab.exe`.

Per avere informazioni sull'installazione e l'utilizzo di MPLAB IDE, fai riferimento a *MPLAB IDE User's Guide* (DS51025) ed al file `README.LAB` posto nella directory di installazione di MPLAB.

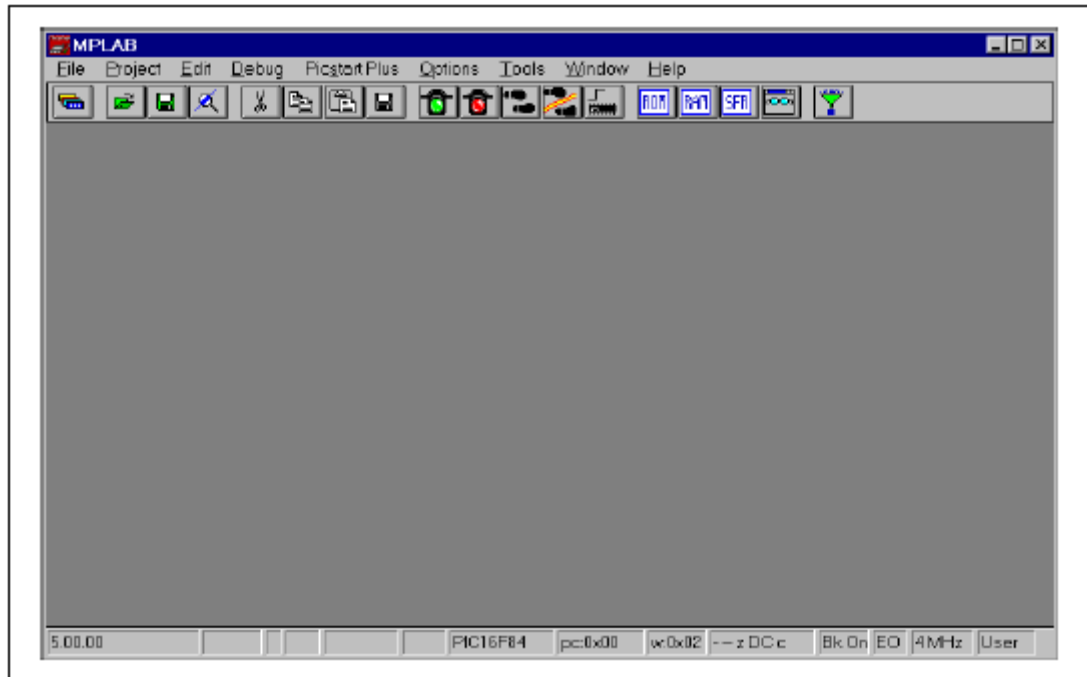


Figure 2.1: MPLAB IDE

2.4 Creating a Hex File for Debugging

Dovrai creare un new project, includere il source file `tut877.asm`, e costruire l' hex file `tut877.hex`. Guarda nella sezione seguente per le istruzioni su come creare un new project, `tut877.pjt`.

2.4.1 New Project Directory

In File Manager (Windows 3.1) o Windows Explorer, crea una subdirectory per il new project, `\MPLAB\tut877`. Copia il file `tut877.asm` da `\MPLAB` a questa subdirectory.

2.4.2 New Project

Seleziona *Project > New Project*, seleziona la directory per il new project, e quindi scrivi `tut877.pjt` nel File Name box. Clicca **OK**.

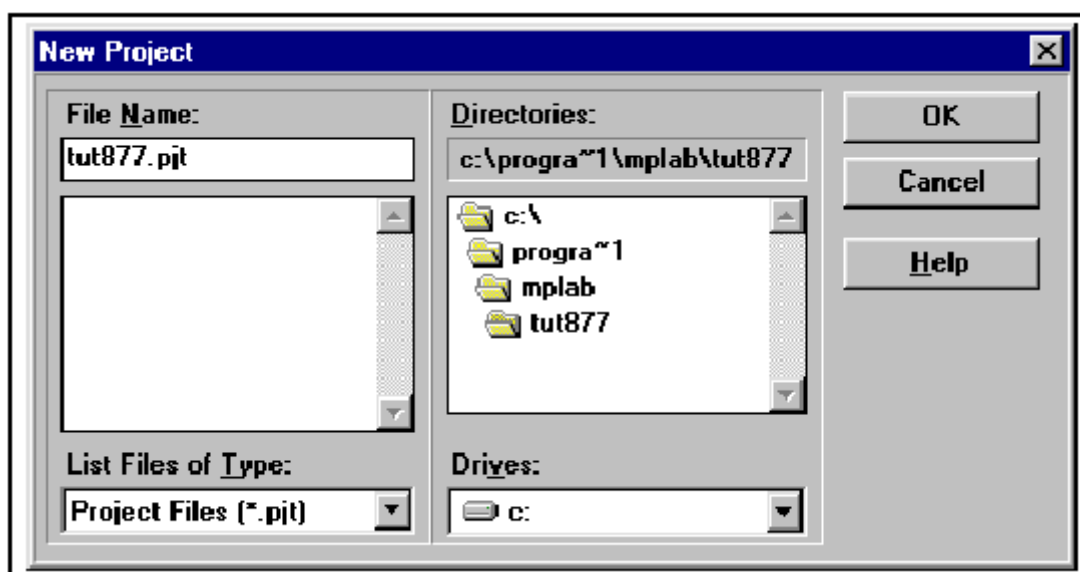


Figure 2.2: New Project - `tut877.pjt`

2.4.3 Project Dialog

Devi aprire l' Edit Project dialog.

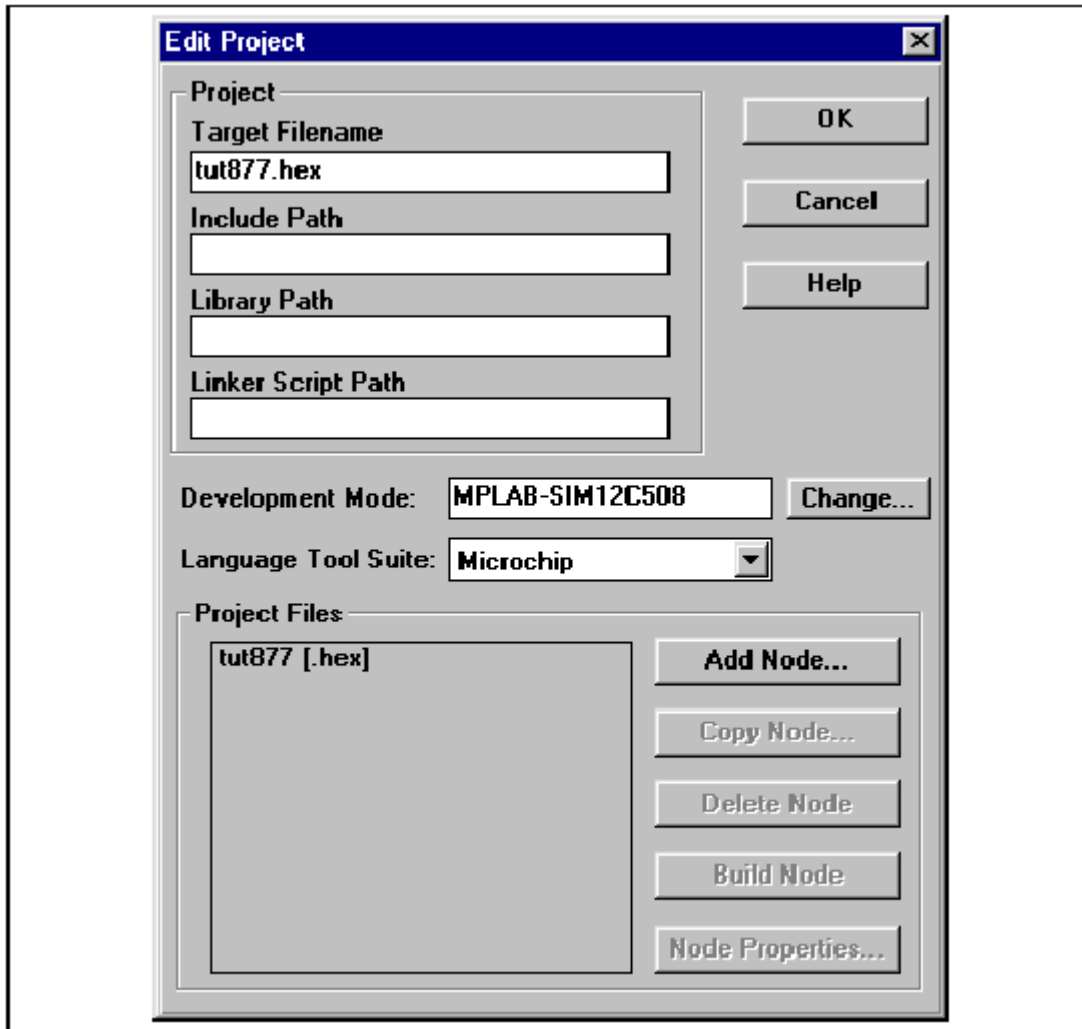


Figure 2.3: Edit Project Dialog Before Setting Development Mode

Osserva il development mode. Esso indica che noi stiamo lavorando con MPLAB SIM simulator e un PIC12C508 processor. Il tuo display indicherà il development mode ed il processore utilizzati in precedenza in MPLAB IDE. Noi dobbiamo cambiare tale settaggio ora.

Premi il pulsante **Change**.

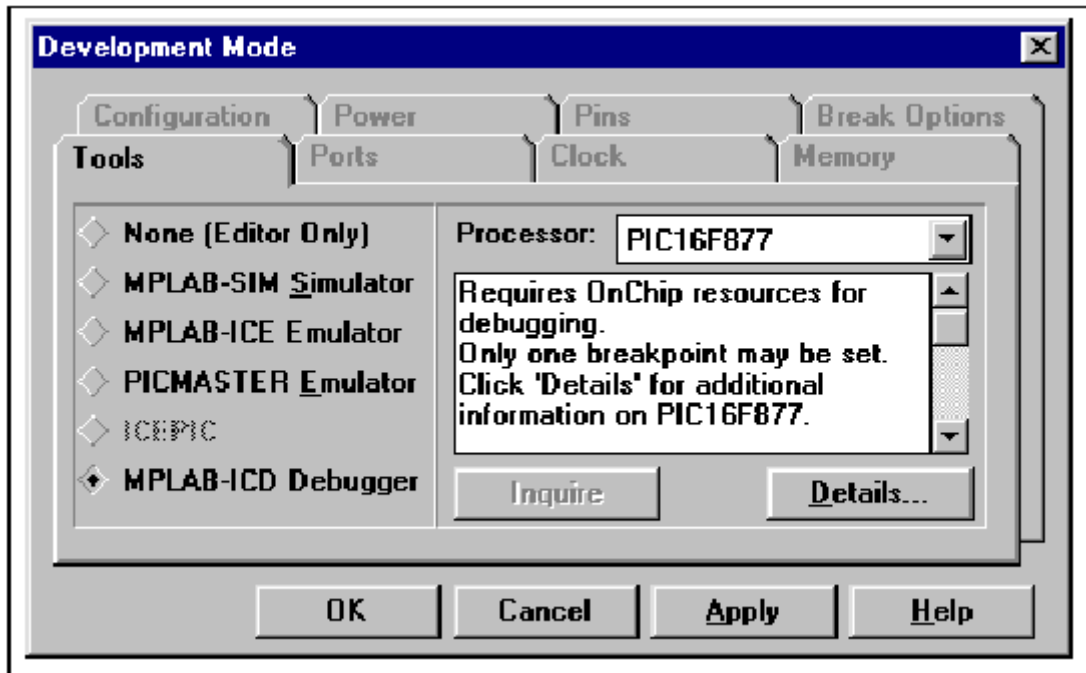


Figure 2.4: Setting the Development Mode

Seleziona MPLAB ICD Debugger nella cartella Tools. Controlla di avere selezionato il PIC16F877 processor. Premi **OK**.

Una window t'informerà che nessun hex file è stato creato per questo project. Noi creeremo il file .hex più tardi. Premi **OK**.

MPLAB IDE ora stabilirà una comunicazione con MPLAB ICD. MPLAB ICD dialog apparirà momentaneamente durante questo processo. Se ricevi un error message, allora controlla le connessioni del power supply, degli integrati e dei cavi. Per avere dettagliate informazioni sulla localizzazione dei guasti, leggi il Capitolo 5.

MPLAB ICD dialog deve rimanere aperta sempre durante le operazioni descritte nel tutorial. Non chiudere MPLAB ICD dialog quando essa appare. Se vuoi, puoi muoverla.

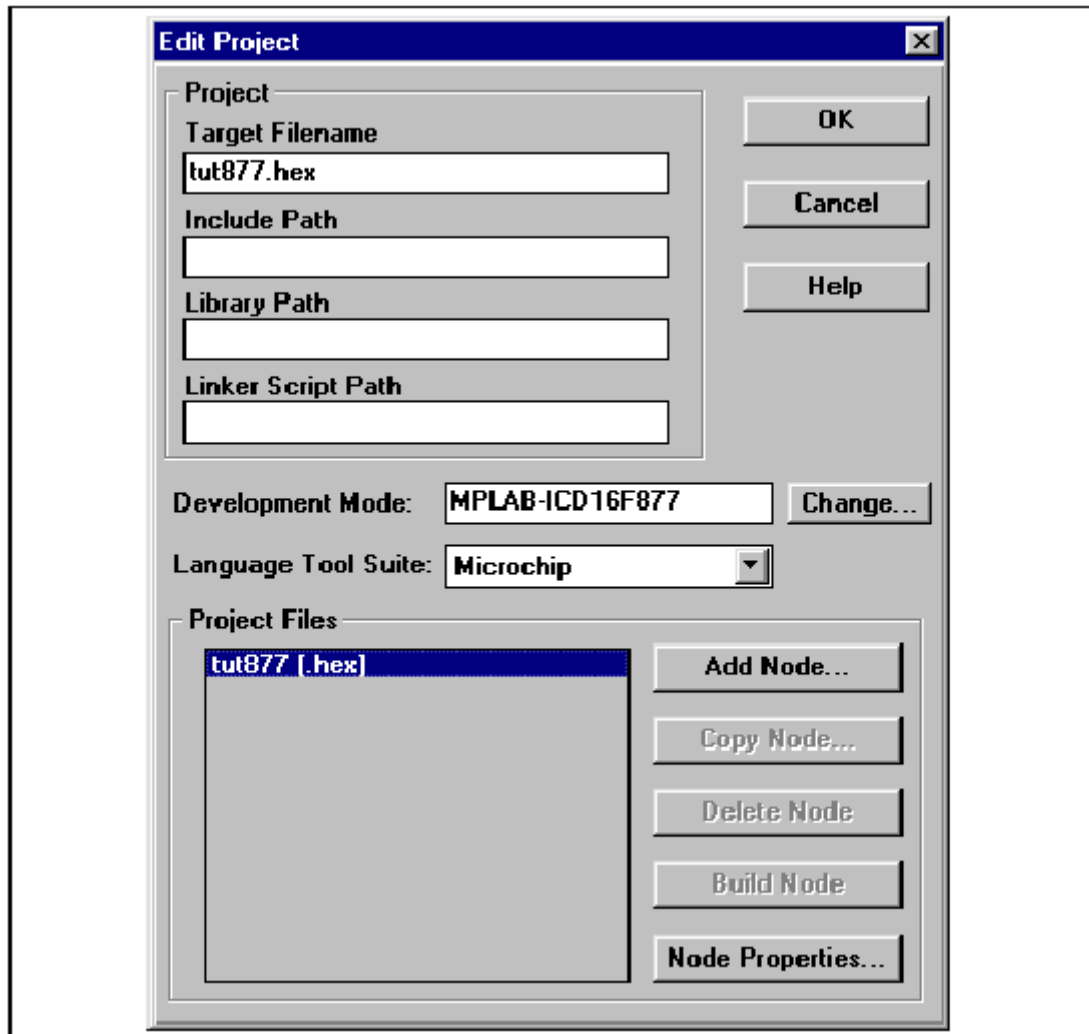


Figure 2.5: Edit Project Dialog

Nota che i corretti development mode e processor sono mostrati nell' Edit Project dialog.

Premi sulla linea `tut877 [.hex]` nell'area Project Files dell' Edit Project dialog, quindi premi il bottone **Node Properties**.

2.4.4 Set Node Properties

Node Properties dialog mostra gli switches della command line (opzioni della riga di comando) per lo strumento (tool), in questo caso l'MPASM assembler. Quando tu apri per la prima volta questa dialog, le checked boxes rappresentano i valori di default per il tool. In questo tutorial, questi settaggi non necessitano di essere cambiati. Vanno quindi bene i valori di defaults.

Click **OK** per tornare alla Edit Project dialog.

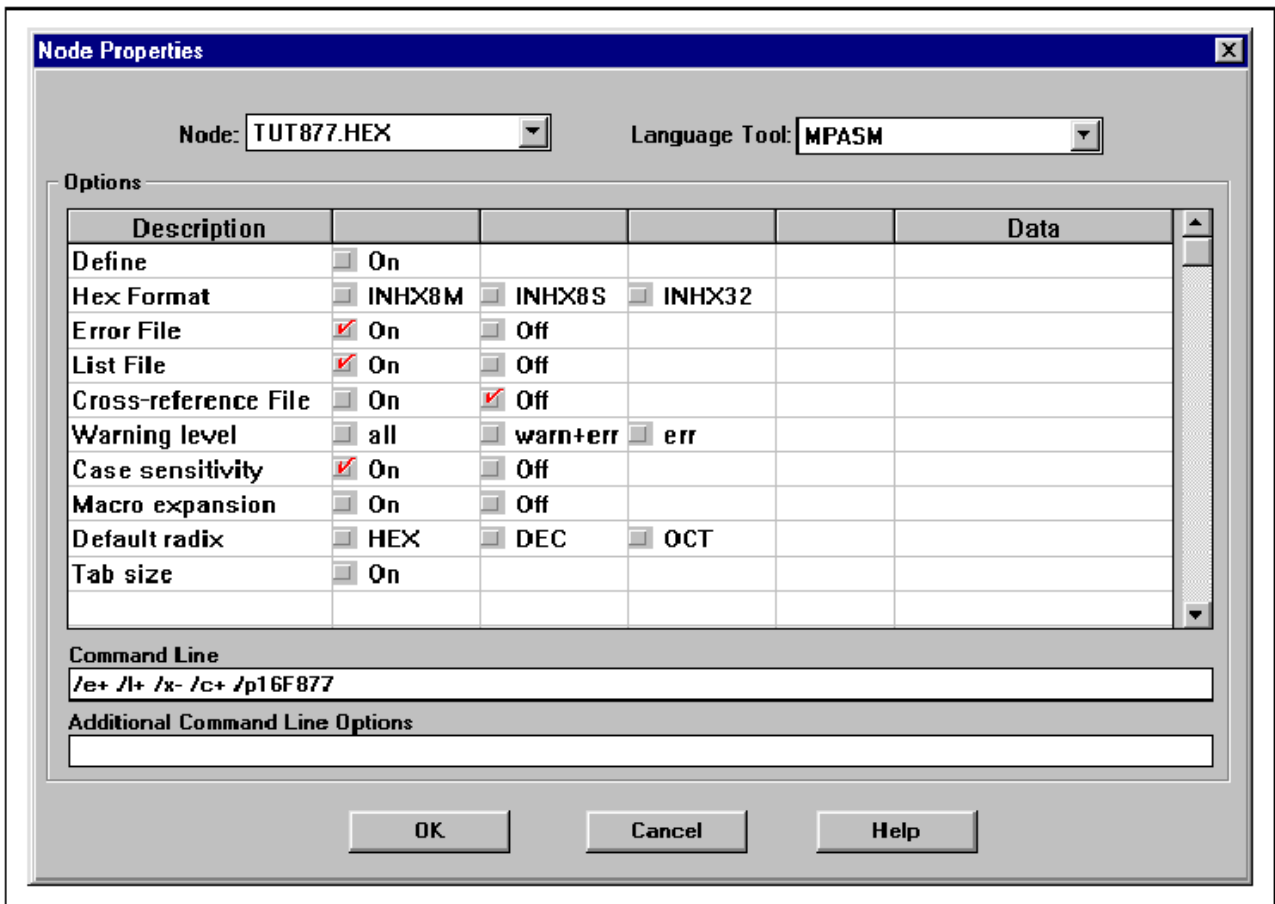


Figure 2.6: Node Properties Dialog

2.4.5 Add Node

Premi **Add Node** dall' Edit Project dialog per aprire l' Add Node dialog.

Seleziona `tut877.asm` per questo tutorial, quindi premi **OK**.

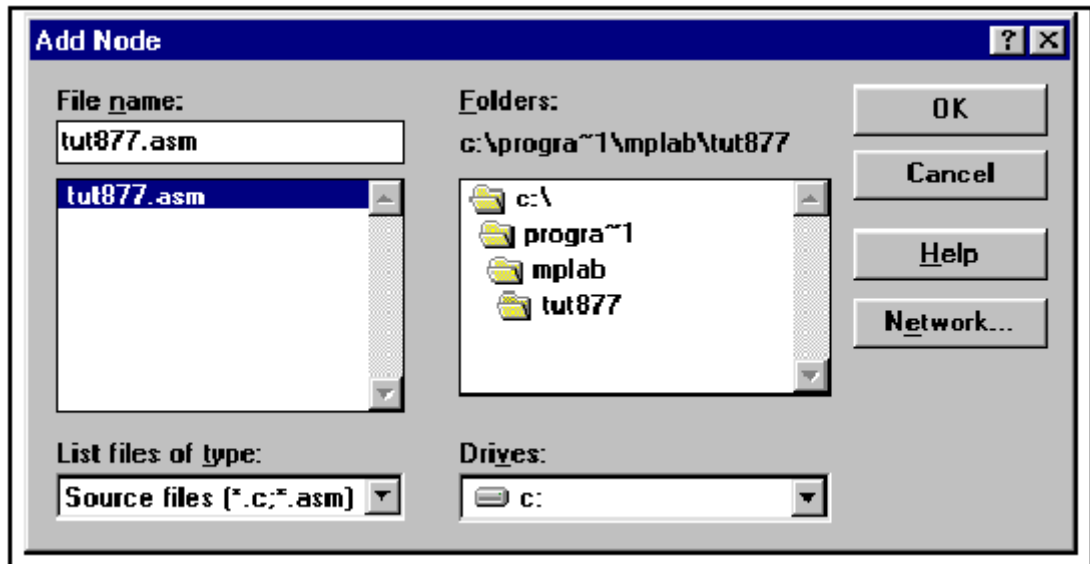


Figure 2.7: Add Node

2.4.6 Complete Project Setup

In questo semplice esempio, non immettere nulla nelle Path boxes. Quando la tua applicazione diventerà complessa, tu potrai avere la necessità di immettere le directories dei tuoi Include Files nelle appropriate boxes.

L' MPASM assembler crea sempre un .hex file con lo stesso nome del file sorgente. Il Project Manager creerà un file `tut877.hex` quando verrà fatta l'operazione di build.

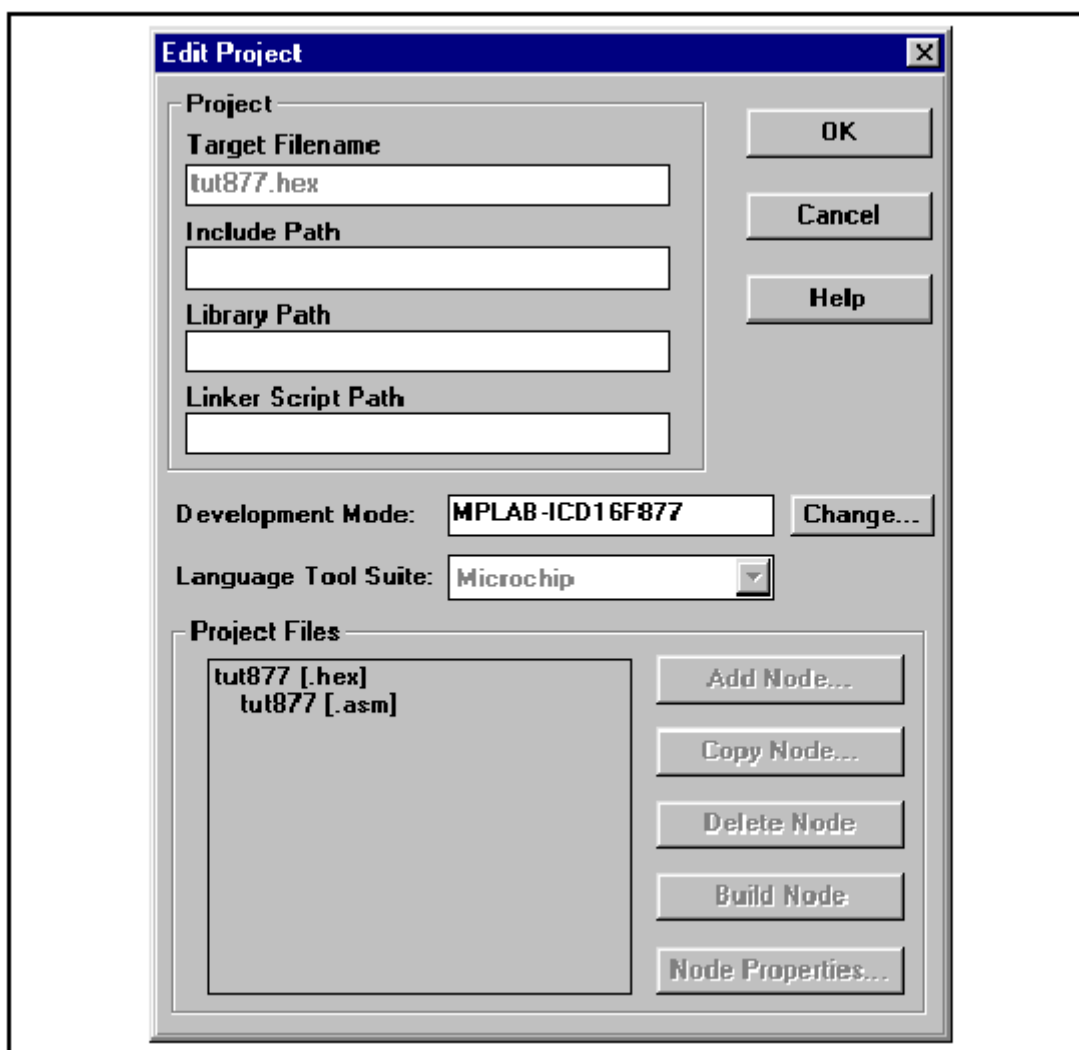


Figure 2.8: Edit Project Dialog with Node

Premi **OK** per chiudere la Edit Project Dialog.

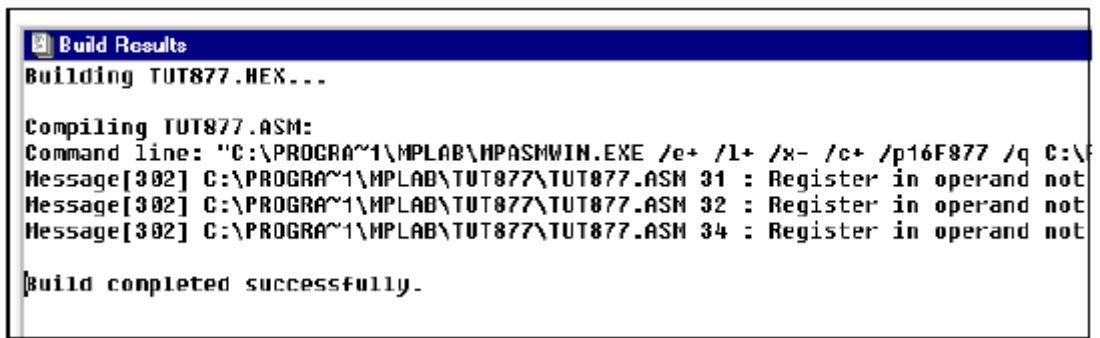
Ora seleziona *Project > Save Project* dal menù dell' MPLAB IDE per salvare il tuo progetto.

2.4.7 Make Project

Seleziona *Project > Make Project* dal menù per compilare l'applicazione utilizzando l' MPASM assembler.

Nota: Quando fai il build del project con successo, una window ti informa che il project è stato rebuilt e che devi riprogrammare l' ICD allo scopo di rendere effettivi i cambiamenti. Se tu non vuoi vedere questo warning tutte le volte che effettui un rebuild, check la checkbox per ignorare questo warning. Noi faremo questo più tardi nel tutorial. Premi sul bottone **Close**.

Una Build Results window mostra la command line spedita all' assembler. Essa dovrebbe essere come questa:



```
Build Results
Building TUT877.HEX...

Compiling TUT877.ASM:
Command line: "C:\PROGRA~1\MPLAB\MPASMWIN.EXE /e+ /l+ /x- /c+ /p16F877 /q C:\
Message[302] C:\PROGRA~1\MPLAB\TUT877\TUT877.ASM 31 : Register in operand not
Message[302] C:\PROGRA~1\MPLAB\TUT877\TUT877.ASM 32 : Register in operand not
Message[302] C:\PROGRA~1\MPLAB\TUT877\TUT877.ASM 34 : Register in operand not

Build completed successfully.
```

Figure 2.9: Build Results Window

Nota: Il messaggio [302] semplicemente ricorda che, alla linea segnalata (i.e. number, 31, 32, and 34) tu hai specificato un file register che non si trova nel bank zero. Questo non è un errore ed il tuo codice è stato compilato correttamente.

Clicca sull' X posta nell'angolo alto a destra della Build Results window per chiuderla. Oppure, clicca l'icona Restore vicina alla X per ridimensionare la window. I successivi builds utilizzeranno questa piccola finestra per visualizzare i risultati.

2.5 Setting Up MPLAB ICD

A questo punto, l' MPLAB ICD dialog dovrebbe essere sul tuo desktop. Fai quello che viene descritto per settare l' MPLAB IDE in modo da poterlo utilizzare con l' MPLAB ICD hardware.

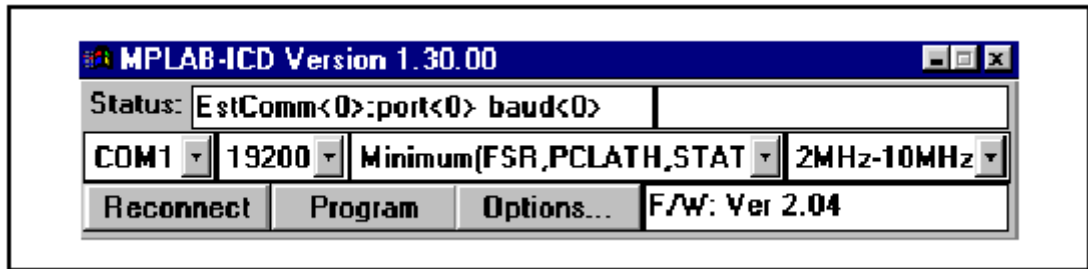


Figure 2.10: MPLAB ICD Dialog

Table 2.1: MPLAB ICD Dialog

Item	Options
Status	La Status bar visualizza la funzione eseguita da MPLAB ICD e lo stato. Quando tu programmi un dispositivo, puoi controllare le celle programmate in quest'area. Quanto l'operazione è completa, la Status box visualizza il messaggio "Waiting for user command."
COM Port and Baud Rate Pull-Down Menus	Non cambiare questo valore in questo tutorial.
Upload Options Pull-Down Menu	Seleziona Minimum per ora. Più tardi nel tutorial cambieremo questa voce per poter effettuare il debugging.
Operating Frequency Range Pull-Down Menu	Seleziona la frequenza di funzionamento nel range 2 MHz - 10 MHz.
Options	Clicca questo bottone per far comparire la ICD Options dialog.

2.6 Setting Programming and Debugging Options

Per programmare il micro PIC16F877, la ICD Options dialog deve prima essere settata per la programmazione. La piccola MPLAB ICD dialog fu aperta quando tu entrasti nel MPLAB ICD development mode. Click **Options** nella MPLAB ICD dialog per aprire la ICD Options dialog.

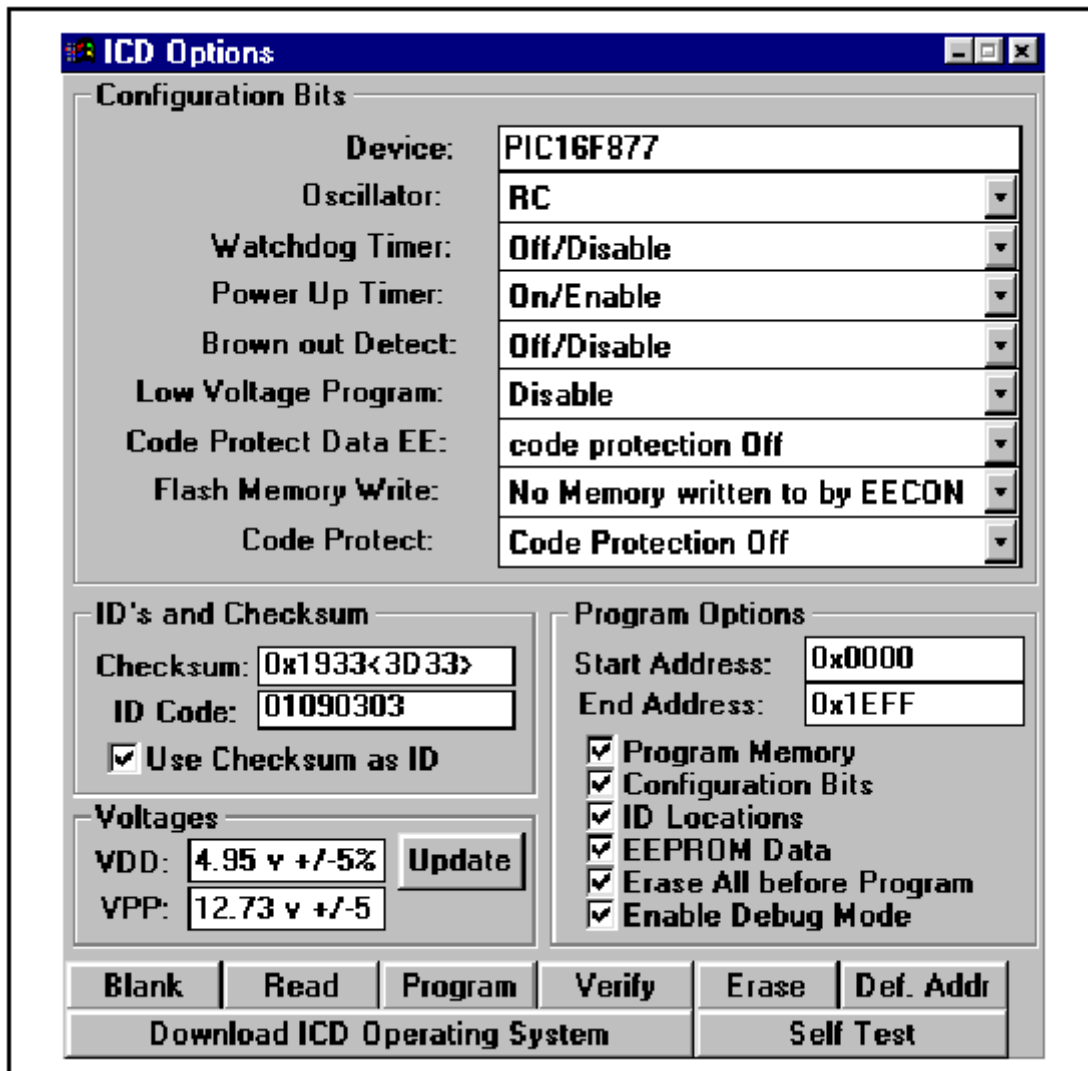


Figure 2.11: MPLAB ICD Options Dialog

2.6.1 Configuration Bits and Device Selection

Questa sezione della ICD Options dialog permette di settare i vari bit di configurazione sui processori PIC16F87X. Clicca sulla freccia e seleziona dalla lista.

Table 2.2: Configuration Bits and Device Selection

Item	Options
Device	Il dispositivo PIC16F877 dovrebbe essere mostrato, come selezionato nel Development Mode dialog. Se no, seleziona <i>Options > Development Mode</i> , seleziona il dispositivo corretto, e premi OK . Poi usa la voce del Project menu <i>Project > Save</i> per salvare il tuo project.
Oscillator	RC è utilizzato in questo tutorial. Controlla la MPLAB ICD Demo Board per essere sicuro che l'oscillator jumper JP1 sia correttamente posizionato per la RC OSC opzione (guarda la voce 8 nella Figura 1.3). Per ulteriori informazioni sull'oscillatore, guarda la Sezione A.3.8.
Watchdog Timer	In questo tutorial, il Watchdog Timer (WDT) deve essere off/disabled.
Power Up Timer	In questo tutorial, the Power-Up Timer (PWRT) deve essere on/enabled.
Brown Out Detect	In questo tutorial, the Brown-Out Detect (BOD) deve essere off/disabled.
Low Voltage Program	Low voltage ICSP programming deve essere disabilitato quando usi l'MPLAB ICD. Questo significa che tu puoi usare RB3 come digital I/O and tu puoi usare HV su MCLR per la programmazione.
Code Protect Data EE	Metti in off il code protection in questo tutorial.
Flash Memory Write	No memory will be written to EECON in questo Tutorial.
Code Protect	Metti in off il code protection in questo tutorial.

2.6.2 IDs and Checksum

Il checksum per i dati e l'ID code sono visualizzati qui. In questo tutorial, Use Checksum as ID deve essere selezionato spuntando la checkbox.

2.6.3 Voltages

Permette di controllare le tensioni VDD e VPP sul circuito applicativo cliccando sul bottone **Update**.

MPLAB ICD necessita di una VPP = 13V ricavata dalla tensione VDD del circuito da controllare (target) attraverso l'uso di uno switching boost converter.

2.6.4 Program Options

Il program address range (Start Address e End Address) è il range del programma o della memoria dati che sarà letto, programmato o verificato.

Verifica che tutte le checkboxes poste sotto Program Options siano checked (spuntate).

Questo significa che tutta la memoria, ID, ed i bit di configurazione saranno programmati. Significa anche che tutta la memoria sarà cancellata prima della programmazione e che il Debug Mode sarà abilitato.

Clicca sul bottone **Def. Addr** per settare il range degli indirizzi disponibile per il programma in base al dispositivo che hai selezionato.

2.6.5 Function Buttons

Durante il tutorial tu cliccherai su questi bottoni per far eseguire le funzioni assegnate sul PIC16F87X posto nel MPLAB ICD Header o nella Demo/Target board.

2.7 Programming the PIC16F877

Clicca sul bottone **Program** per programmare `tut877.hex` ed inserire il debug code all'interno del PIC16F87X posto nel MPLAB ICD Header o nel MPLAB ICD Demo Board.

La programmazione può durare un paio di minuti. Durante la programmazione, la Status box mostra la fase corrente dell'operazione. Quando la programmazione è completata, la Status box mostra il messaggio "Waiting for user command."

Nota: Il debug code è uno speciale codice posto nelle locazioni 1F00h-1FFFh del PIC16F877, tale codice deve essere presente per utilizzare le capacità di in-circuit debugging dell' MPLAB ICD.

Tu puoi minimizzare o muovere la MPLAB ICD dialog, ma non puoi chiuderla. Chiudendo la MPLAB ICD dialog disabiliti l'ICD. Per riabilitare l'ICD, tu devi selezionare Options > Development Mode. Seleziona Editor Only e clicca su **Apply**.

Quindi seleziona MPLAB ICD e clicca **OK** per riabilitare l'ICD.

2.8 Setting Up the Demo Board

Prima di cominciare il debugging, verifica che la MPLAB ICD Demo Board sia settata nel seguente modo:

- la RC OSC opzione deve essere selezionata utilizzando il jumper JP1.
- Il DIP switch (SW3) deve avere tutti gli switch nella posizione ON position per connettere tutti i LED ai loro rispettivi pin del PORTC.

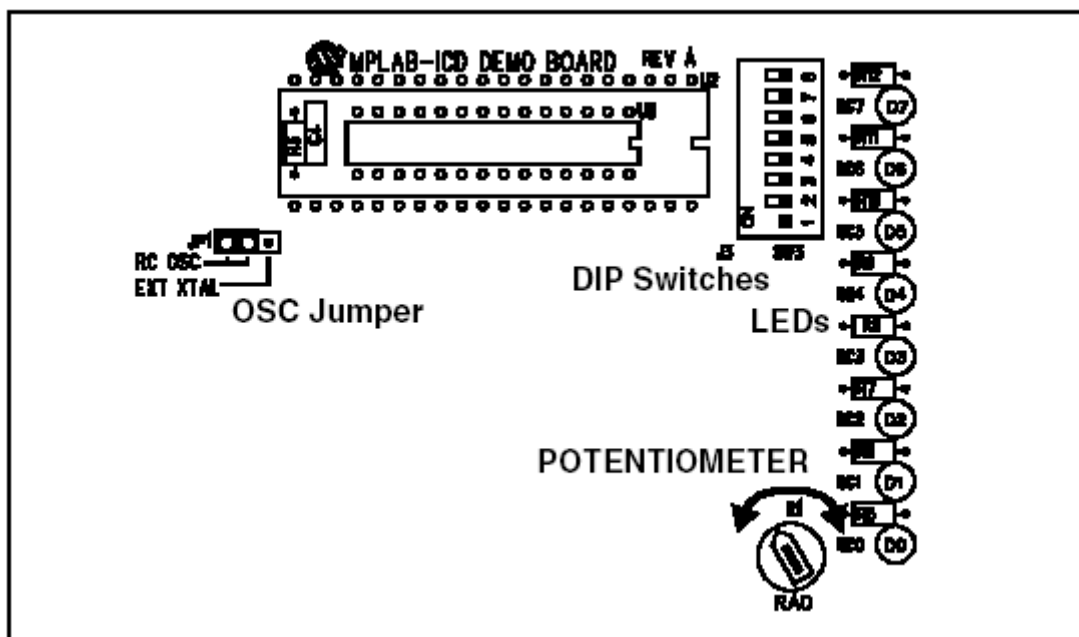


Figure 2.12: Setting Up the Demo Board

2.9 Running Tut877

MPLAB ICD esegue il programma in real-time mode oppure in step mode.

- L'esecuzione Real-time avviene quando il PIC16F87X posto nel MPLAB ICD Header è messo nel Run mode dell'MPLAB IDE.
- L'esecuzione Step mode può essere effettuata dopo che il processore è stato fermato (halted).

Per cominciare il real-time mode:

- Apri il file `tut877.asm` per guardarlo (*File > Open*)
- Clicca sul Run toolbar button oppure dai il comando *Debug > Run > Run*

La Status bar posta al di sotto dell'MPLAB IDE desktop dovrebbe diventare gialla.

Per cambiare il colore associato al program run, seleziona *Options > Environment Setup* e clicca il **Colors** tab.

Sulla MPLAB ICD Demo Board, regola la freccia del potenziometro (RA0) in modo da puntare verso il DIP switch (SW3). Osserva i LED. Se il programma lavorasse correttamente, tu dovresti vedere la rappresentazione binaria della tensione applicata dal potenziometro sul pin RA0 del micro. Tuttavia, un insidioso bug è stato posto nel programma Tut877 ! Naturalmente, questo ci darà l'opportunità di fare un debug del codice.

Clicca sul bottone Halt posto sulla toolbar oppure dai il comando *Debug > Run > Halt* per fermare l'esecuzione del programma. Resetta il programma selezionando *Debug > Run > Reset*.

2.10 Debugging Tut877

Una qualsiasi delle seguenti condizioni potrebbe impedire al programma Tut877 di lavorare:

- Il valore del convertitore A/D non è scritto correttamente sul PORTC (LEDs)
- Il convertitore A/D non è on o non è stato settato per convertire
- Un errore nel codice sorgente è la causa del funzionamento errato del program

Per esplorare la prima possibilità, tu devi settare un breakpoint nella linea del programma che scrive il valore del risultato della conversione A/D sul PORTC. Evidenzia o poni il cursore sulla seguente linea del codice sorgente tut877.asm :

```
movf ADRESH,W ;Write A/D result to PORTC
```

Clicca sul bottone destro del mouse per accedere al shortcut menu. Seleziona *Break Point(s)* dallo shortcut menu come mostrato nella Figura 2.13. Questa linea sarà ora marcata come breakpoint. Per cambiare il colore che evidenzia un breakpoint, seleziona *Options > Environment Setup* e clicca sul **Colors** tab.

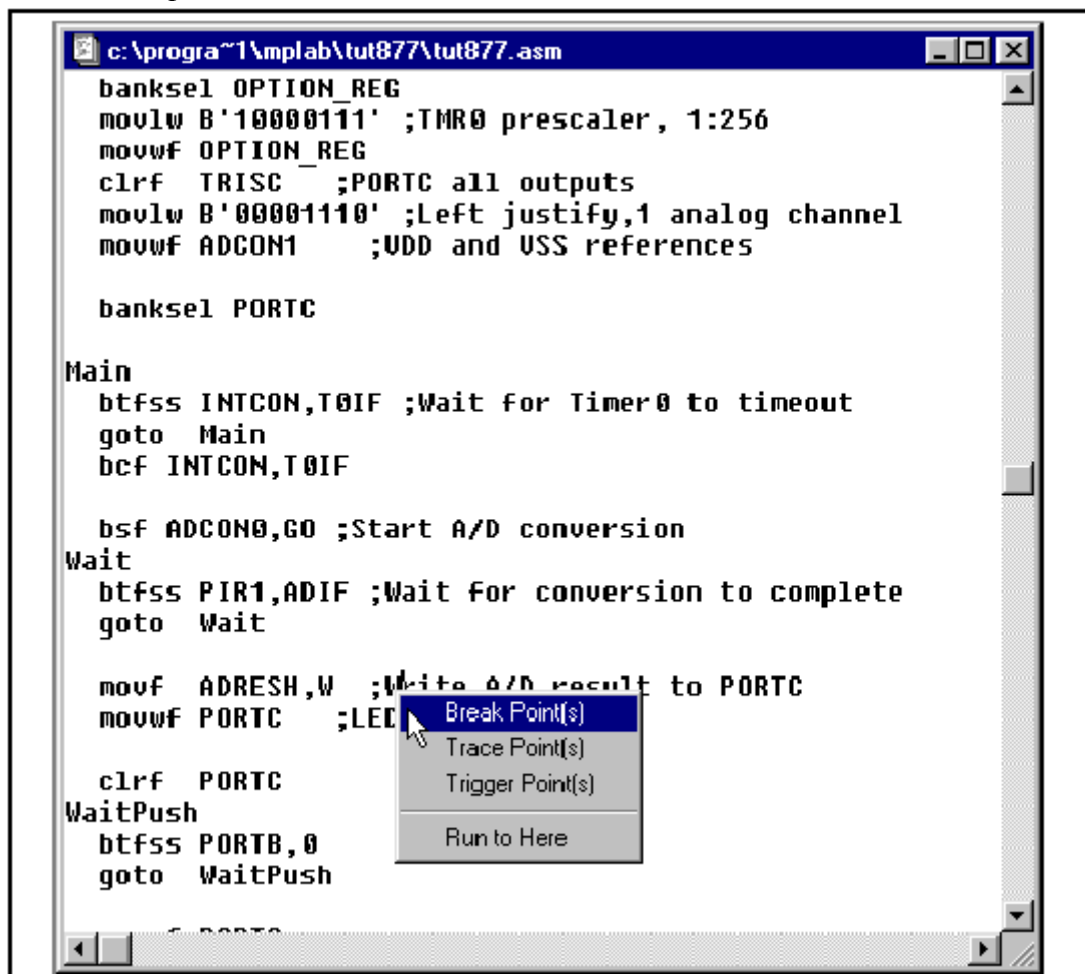


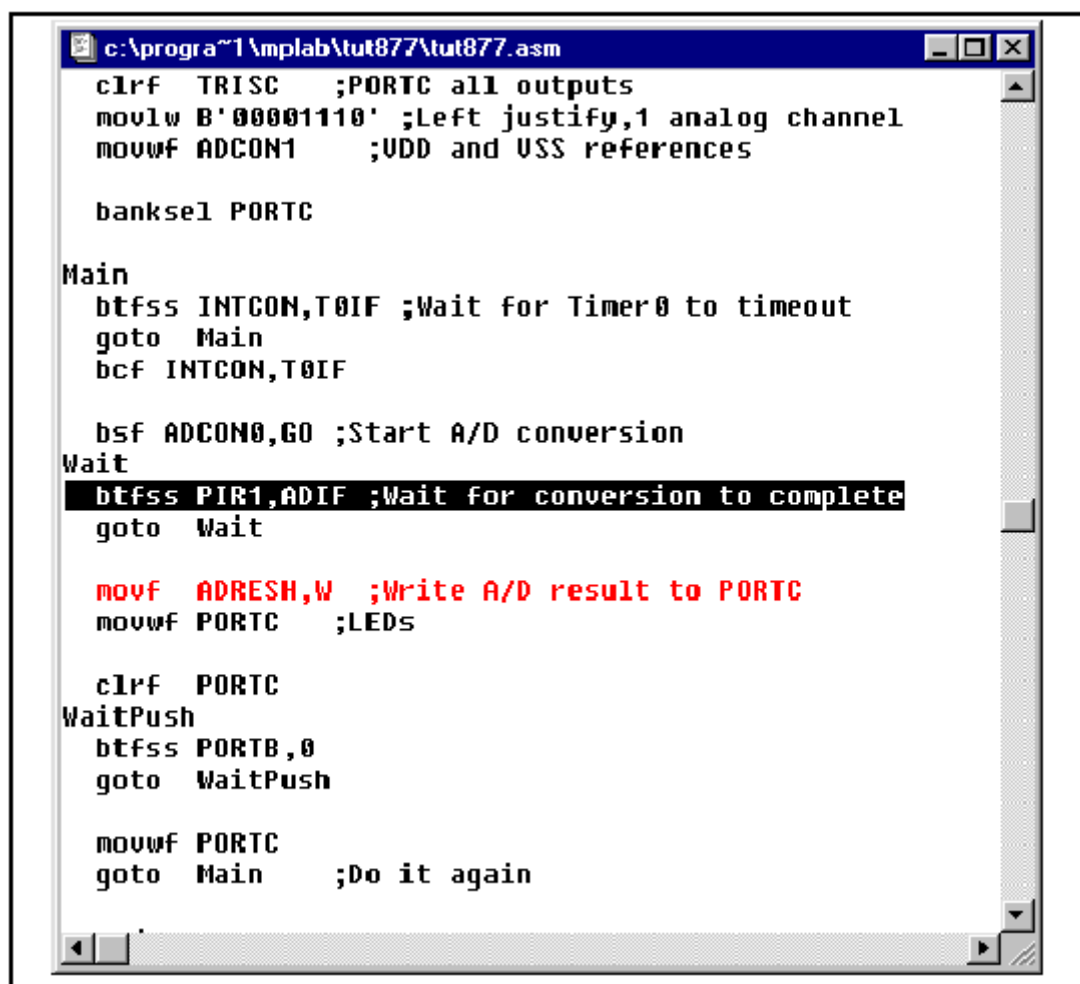
Figure 2.13: Set Breakpoint

Clicca sul bottone Run posto sulla toolbar o dai il comando Debug > Run > Run per eseguire il programma ancora una volta in real-time mode.

Un breakpoint ferma l'esecuzione del programma quando il programma esegue la linea macata come breakpoint. Tuttavia, il nostro programma esempio non si ferma.

Perciò, fermalo ora cliccando il bottone Halt della toolbar o dando il comando Debug > Run > Halt .

Osserva la finestra del codice sorgente (tut877.asm) e osserva dove il programma si è fermato. Il nostro programma esempio è fermo su una delle due linee nella routine Wait come mostrato nella Figura 2.14. Basandoci sulla locazione dell'halt e sul fatto che il programma non raggiunge mai il breakpoint, noi concludiamo che il problema è nella conversione A/D, in quanto il flag che segnala il completamento della conversione A/D non viene settato.



```
c:\progra~1\mplab\tut877\tut877.asm
clrfs TRISC ;PORTC all outputs
movlw B'00001110' ;Left justify,1 analog channel
movwf ADCON1 ;UDD and USS references

banksel PORTC

Main
btfss INTCON,T0IF ;Wait for Timer0 to timeout
goto Main
bcf INTCON,T0IF

bsf ADCON0,GO ;Start A/D conversion
Wait
btfss PIR1,ADIF ;Wait for conversion to complete
goto Wait

movf ADRESH,W ;Write A/D result to PORTC
movwf PORTC ;LEDs

clrfs PORTC
WaitPush
btfss PORTB,0
goto WaitPush

movwf PORTC
goto Main ;Do it again
```

Figure 2.14: Program Halted

Il settaggio e l'inizializzazione della conversione A/D sono posti all'inizio del programma. Per controllare questo codice, prima resetta il programma selezionando Debug > Run > Reset. La prima istruzione dopo Start dovrebbe essere evidenziata.

Apri una nuova watch window così puoi osservare i cambiamenti dei valori dei registri dell'A/D mentre il programma è in esecuzione. Seleziona *Window > Watch Windows > New Watch Window*. Si aprirà la Add Watch Symbol dialog, con la Watch_1 new watch window dietro ad essa. Aggiungi i simboli ADCON0 e ADCON1 come mostrato in Figura 2.15.

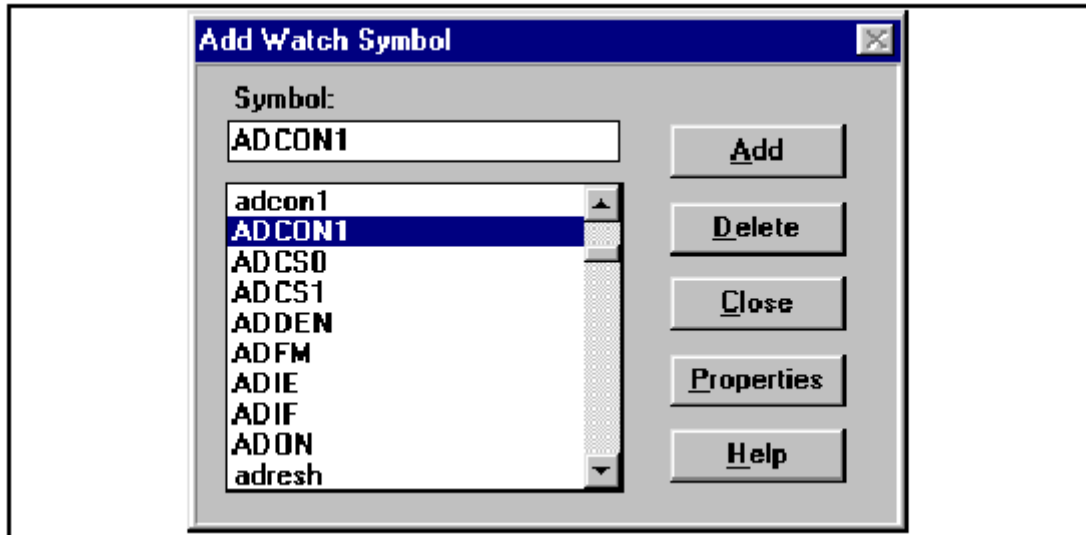


Figure 2.15: Adding Watch Symbols

Clicca **Close** quando hai finito. I simboli selezionati dovrebbero ora essere visibili nella watch window come mostrato nella Figura 2.16.

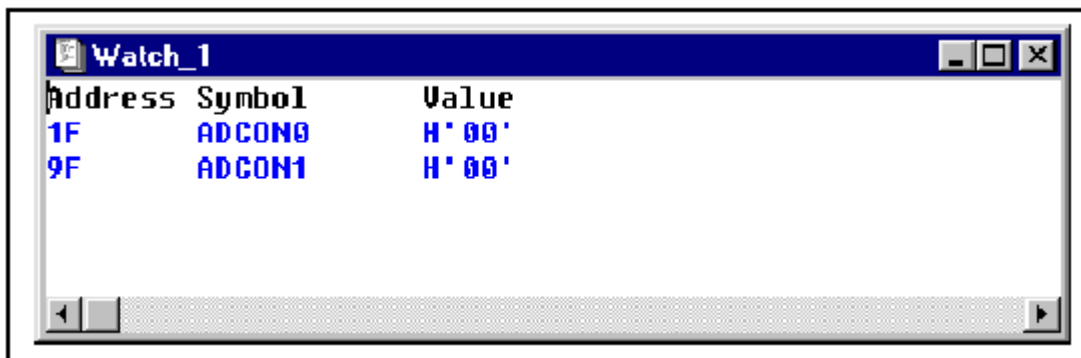


Figure 2.16: Watch Window

Nella MPLAB ICD dialog, setta l'upload options al valore Minimum & Watch Windows. Le operazioni di single stepping e di halting potrebbero ora essere lente, ma questo settaggio ci permette di vedere la Watch windows aggiornata durante il debugging. Chiudi o minimizza la MPLAB ICD Options dialog.

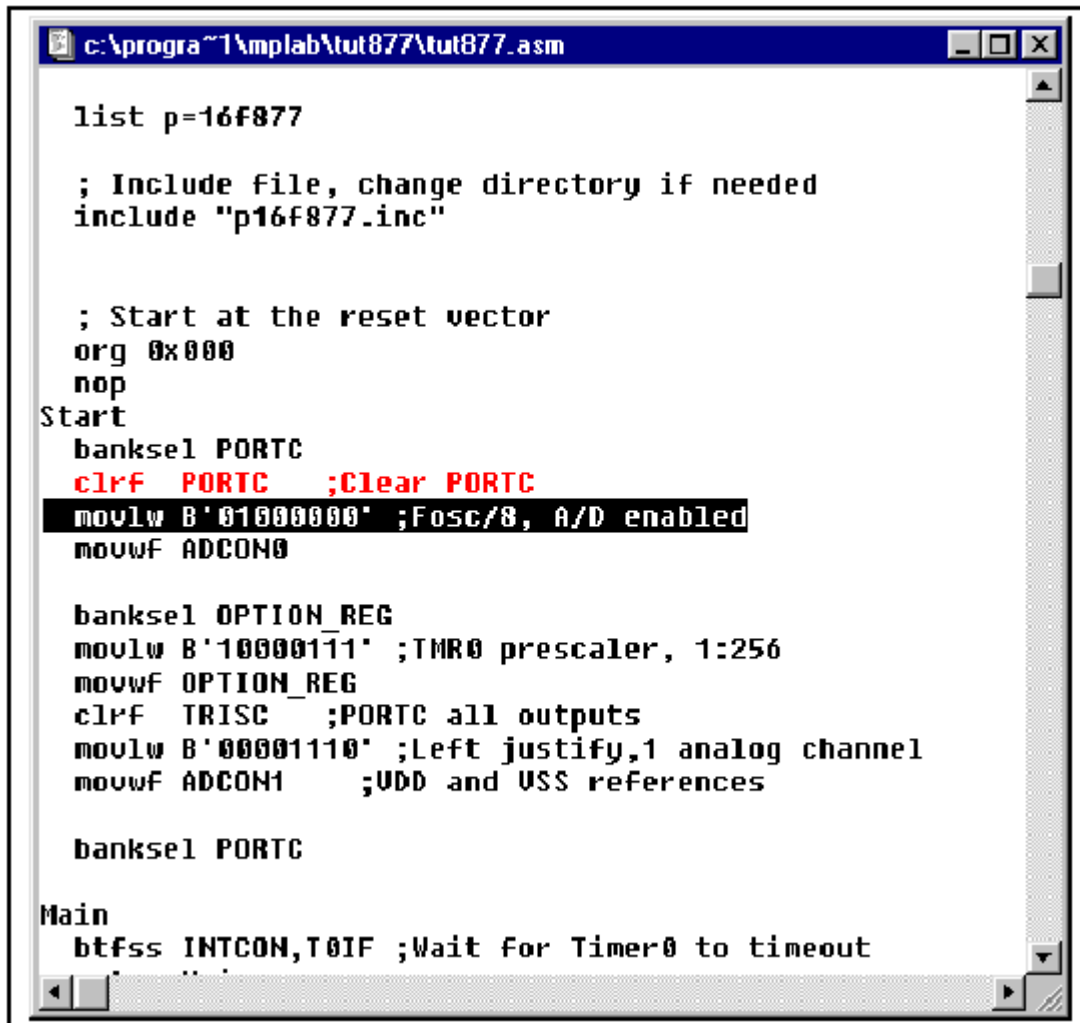
Nel tut877.asm codice sorgente, setta un breakpoint nella seconda istruzione dopo Start. Di nuovo, evidenzia o poni il cursore sulla seguente linea del codice sorgente tut877.asm:

```
clrf PORTC ;Clear PORTC
```

Clicca sul tasto destro del mouse per accedere al shortcut menu. Seleziona *Break Point(s)* dallo shortcut menu. Questa linea ora sarà marcata come un breakpoint.

Alla fine, clicca sul bottone Run posto sulla toolbar oppure dai il comando Debug > Run > Run per eseguire il programma in real-time mode.

Questa volta il programma si fermerà dopo avere eseguito la linea di codice con il breakpoint, e l'istruzione dopo il breakpoint sarà evidenziata come mostrato nella Figura 2.17.



```
c:\progra~1\mplab\tut877\tut877.asm

list p=16f877

; Include file, change directory if needed
include "p16f877.inc"

; Start at the reset vector
org 0x000
nop
Start
banksel PORTC
clrf PORTC ;Clear PORTC
movlw B'01000000' ;Fosc/8, A/D enabled
movwf ADCON0

banksel OPTION_REG
movlw B'10000111' ;TMR0 prescaler, 1:256
movwf OPTION_REG
clrf TRISC ;PORTC all outputs
movlw B'00001110' ;Left justify,1 analog channel
movwf ADCON1 ;UDD and USS references

banksel PORTC

Main
btfss INTCON,T0IF ;Wait for Timer0 to timeout
```

Figure 2.17: Program Halted After Break

Ora esegui una istruzione alla volta (single step) cliccando sul bottone Step posto sulla toolbar o selezionando il comando Debug > Run > Step. Esegui l'operazione Single step due volte e quindi esamina i valori dei registri ADCON0 e ADCON1 nella Watch window. Tu dovresti notare che ADCON0 vale 40 hex come mostrato nella Figura 2.18.

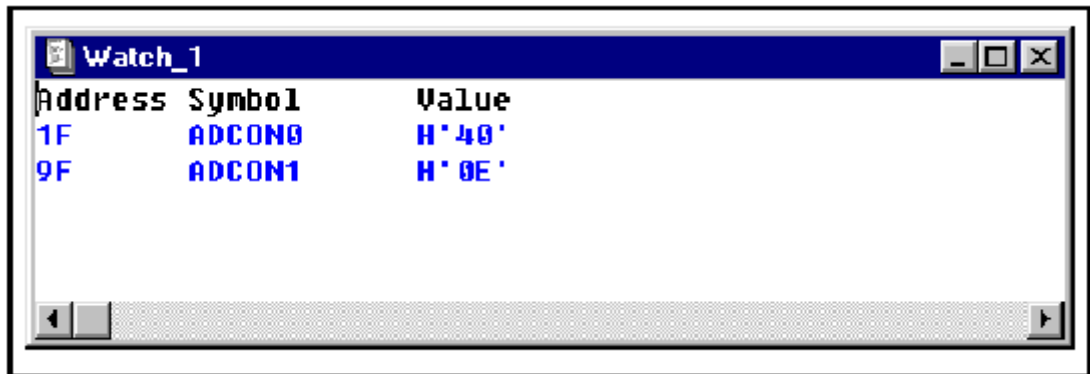


Figure 2.18: Updated Watch Window

Questo corrisponde al valore binario assegnato nel programma, ma è corretto questo valore ? Esaminando il Data Sheet(DS30292) del PIC16F87X nella sezione sull'A/D, tu potrai vedere che l'ultimo bit deve essere posto ad uno, non a zero, per mettere ad on il modulo A/D.

Per correggere questo bug, sostituisci:

```
movlw B'01000000' ;Fosc/8, A/D enabled
```

con

```
movlw B'01000001' ;Fosc/8, A/D enabled
```

Seleziona *File > Save* per salvare il cambiamento e seleziona *Project > Make Project* per fare un rebuild del project.

Un messaggio ti dirà che il programma ha avuto un rebuilt e che tu devi riprogrammare l'ICD per far in modo che il cambiamento abbia effetto. Clicca il bottone **Program** nella MPLAB ICD dialog per riprogrammare l'ICD in modo da rendere effettivo il cambiamento.

Quando la Status box dell'MPLAB ICD indica che esso sta aspettando il tuo successivo comando, tu sei pronto per mandare ancora in esecuzione il tuo programma.

Premi il bottone **Run** sulla toolbar o dai il comando *Debug > Run > Run* per eseguire il progamma in real-time mode. Alcuni LED dovrebbero ora essere accesi. Regola il potenziometro per cambiare il valore visualizzato dai LED.

Il codice sorgente di questo tutorial conteneva solo un bug. Tuttavia, un codice reale può averne di più. Usando l'MPLAB ICD e le funzioni di debugging dell'MPLAB IDE, tu puoi trovare e correggere con successo i bug del tuo codice.

2.11 Tut877 Main Routine

La routine main di `tut877.asm` (Figure 2.19) comincia con la configurazione di PORTC, del A/D module e del Timer0. Quindi essa aspetta l'overflow del Timer0 per dare lo start alla conversione A/D del valore proveniente dal potenziometro. Quando la conversione è completa, il valore è visualizzato sui LED, ed il programma chiude il loop tornando ad aspettare un altro overflow del Timer0 per poter far partire un'altra conversione A/D.

Per avere informazioni dettagliate sul funzionamento del modulo A/D ed una lista delle relative application notes, fai riferimento al documento *PICmicro Mid-Range MCU Family Reference Manual* (DS33023).

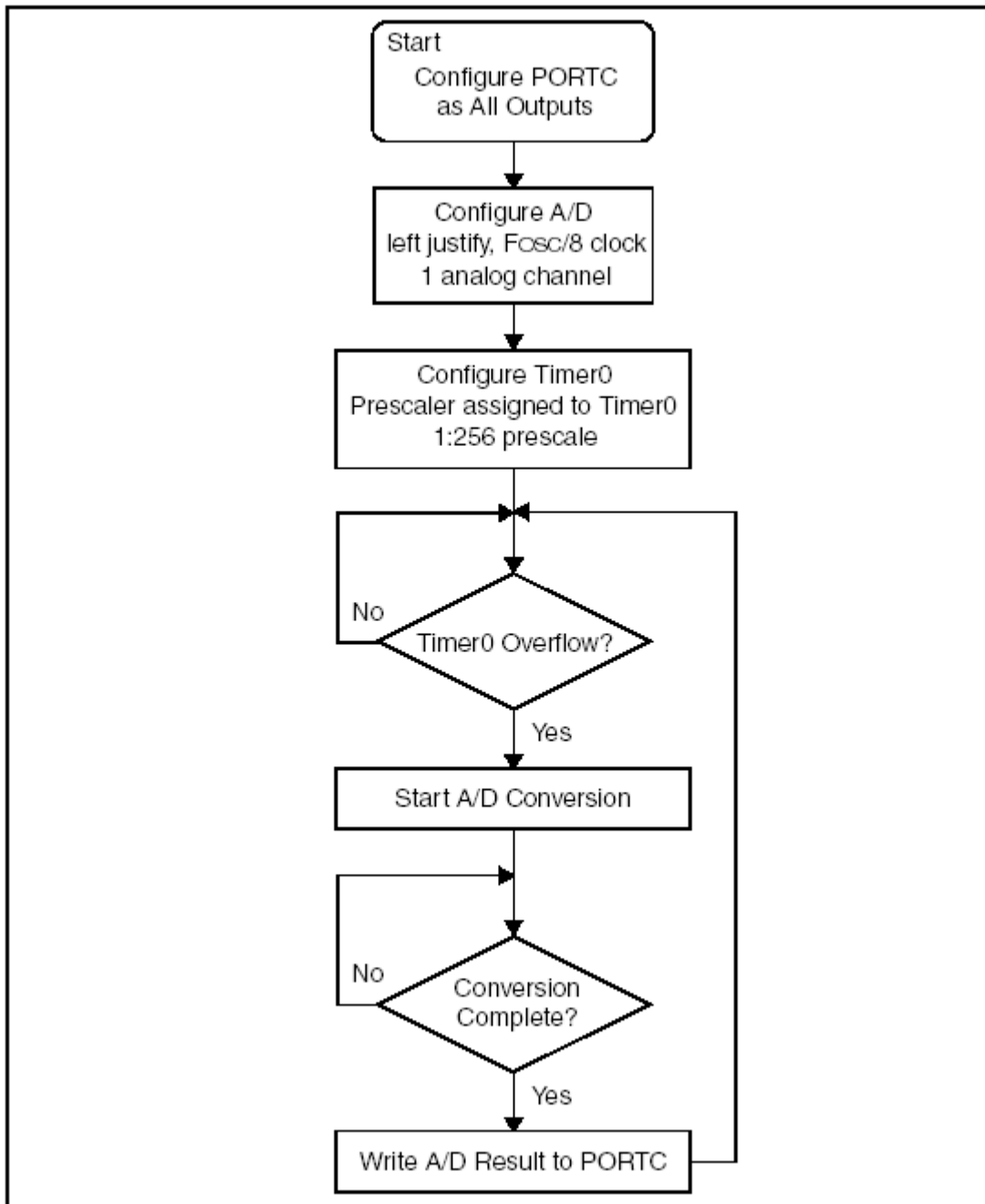


Figure 2.19: Program Flow Chart

2.12 Tut877 Corrected Source Code

This is a functional version of tut877.asm.

```
*****  
;* TUT877.ASM  
*****  
;* Microchip Technology Incorporated  
;* 16 December 1998  
;* Assembled with MPASM V2.20  
*****  
;* This program configures the A/D Module to convert on  
;* A/D channel 0 (the potentiometer) and display the  
;* results on the LEDS on PORTC. Make sure that the DIP  
;* switch SW3 has all switches in the ON position.  
*****  
  
list p=16f877  
  
; Include file, change directory if needed  
include <p16f877.inc>  
  
; Start at the reset vector  
  
org 0x000  
nop  
Start  
banksel PORTC  
clrf PORTC ;Clear PORTC  
movlw B'01000001' ;Fosc/8, A/D enabled  
movwf ADCON0  
  
banksel OPTION_REG  
movlw B'10000111' ;TMR0 prescaler, 1:256  
movwf OPTION_REG  
clrf TRISC ;PORTC all outputs  
movlw B'00001110' ;Left justify,1 analog channel  
movwf ADCON1 ;VDD and VSS references  
  
banksel PORTC  
  
Main  
btfss INTCON,T0IF ;Wait for Timer0 to timeout  
goto Main  
bcf INTCON,T0IF  
  
bsf ADCON0,GO ;Start A/D conversion  
Wait  
btfss PIR1,ADIF ;Wait for conversion to complete  
goto Wait
```

```
movf ADRESH,W ;Write A/D result to PORTC
movwf PORTC ;LEDs

    clrf PORTC
WaitPush
    btfss PORTB,0
    goto WaitPush

    movwf PORTC
    goto Main ;Do it again

end
```