Tutorial 3: Firmware Debugging with PICKit2

I. INTRODUCTION

It is a fact that, it does not matter whether you are an engineer or not, everyone does mistakes in life ☺. We as engineers can skip small details in developing a system for example. The important thing is that we should figure out and correct those mistakes in the application we implement. However, sometimes it becomes really frustrating when we try to find out what the problem really is. We need a procedure more than a "reprogram and try again" style. That is why "debugging" becomes necessary in our engineering tools.

As you may possibly be familiar from MATLAB programming, you can pause the execution of your code and investigate the states of each variable you use inside. The debugging utility you use in MATLAB is a highly featured tool in which you can visually inspect every item by processing them within functions in the command window or array inspector. With a step-by-step trace, you can observe and locate the problems in your code and hence fix them easily. However, you should note that you are using your computer's resources to perform this task which includes a highly organized structure when compared to the hardware limitations of a microcontroller. Therefore, debugging a microcontroller firmware is usually harder than the software you develop in PC. You need additional components to perform debugging PICs which is the main subject of this tutorial.

Here, the debugging utility of PICKit2 will be demonstrated with an example using MPLAB IDE and CCS C compiler. CCS C is a commercial PIC C compiler with very limited 30 day trial versions; you can buy it if you like. The reason why this compiler is used to demonstrate the debugging example is to explain also the integration of another compiler (other than C18 as discussed in tutorial 1) with MPLAB IDE. You can, for sure, use C18 or other compilers to perform this debugging task.

II. The firmware example

Now let's implement a simple blinking using PIC16F877. Open MPLAB IDE. Select
Project -> Project Wizard

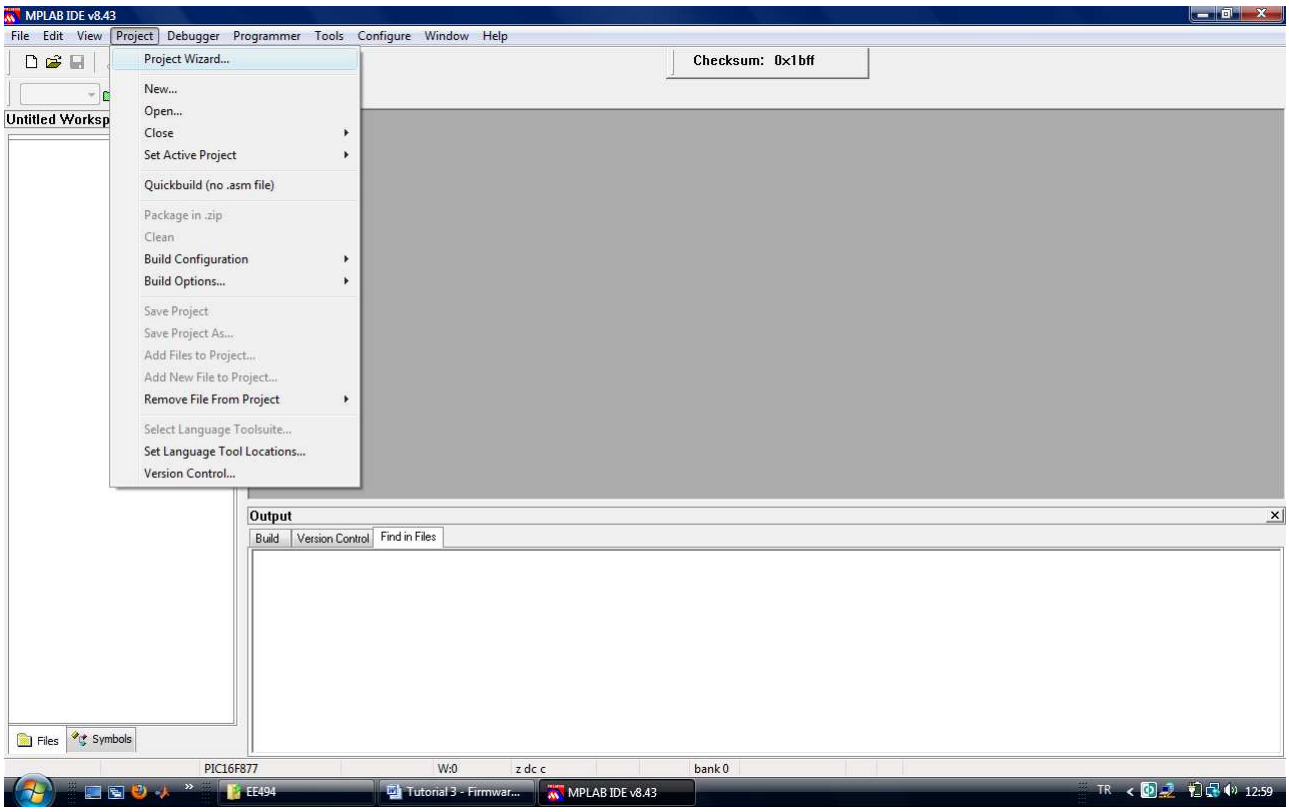

Figure 1: Creating a Project in MPLAB with Project Wizard

Click on Next and Select PIC16F877 from the device list.

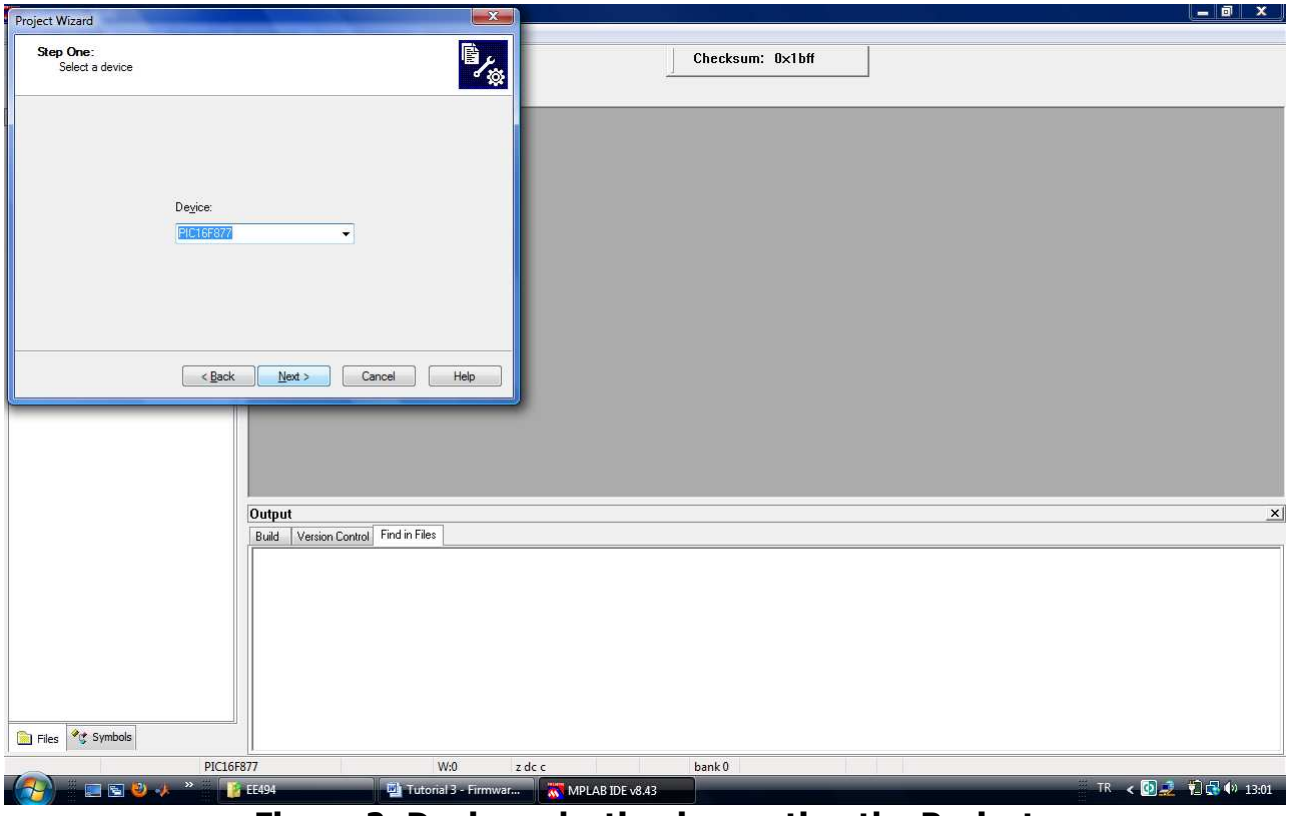

Figure 2: Device selection in creating the Project

Select CCS C toolsuite as the compiler. Check for the directory of the compiler provided that MPLAB can not locate the toolsuite. Click Next.

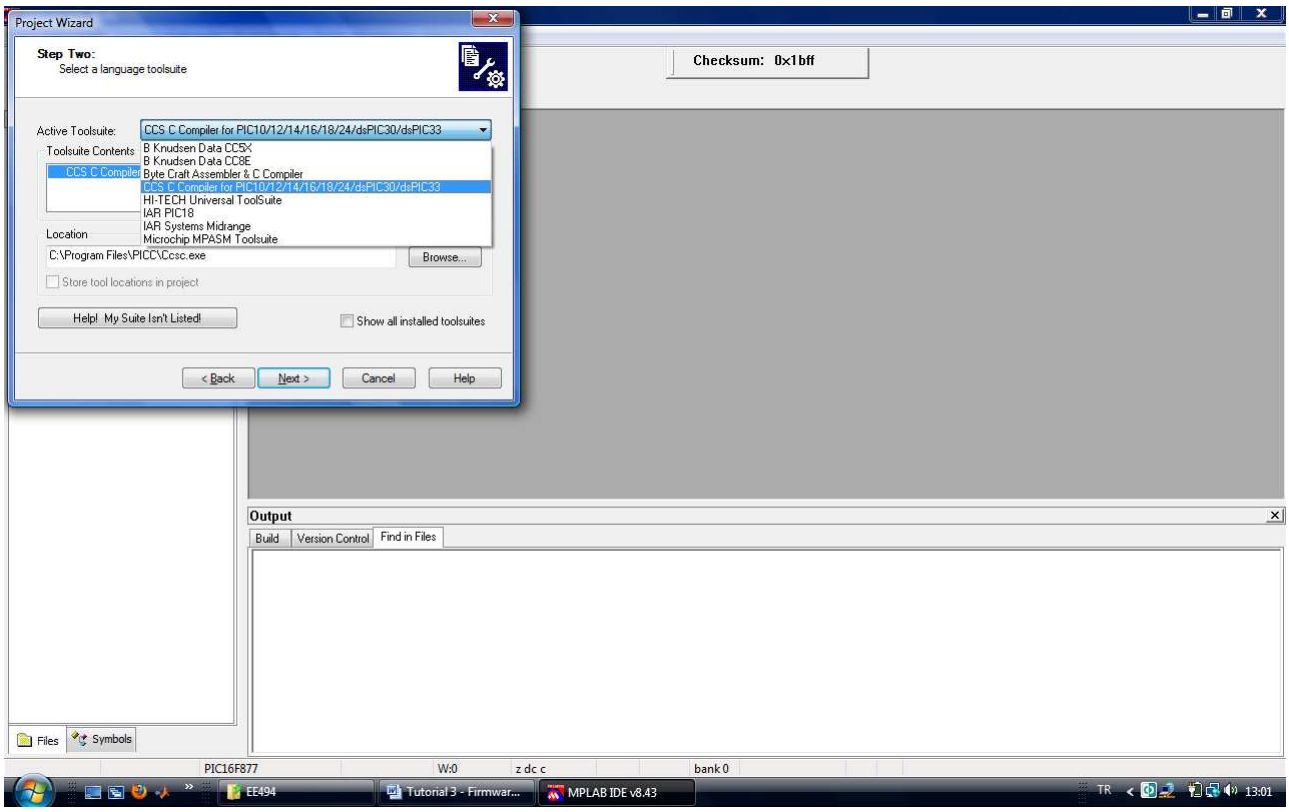

Figure 3: Selection of the Compiler toolsuite

Create your project directory and the project file. Click Next.

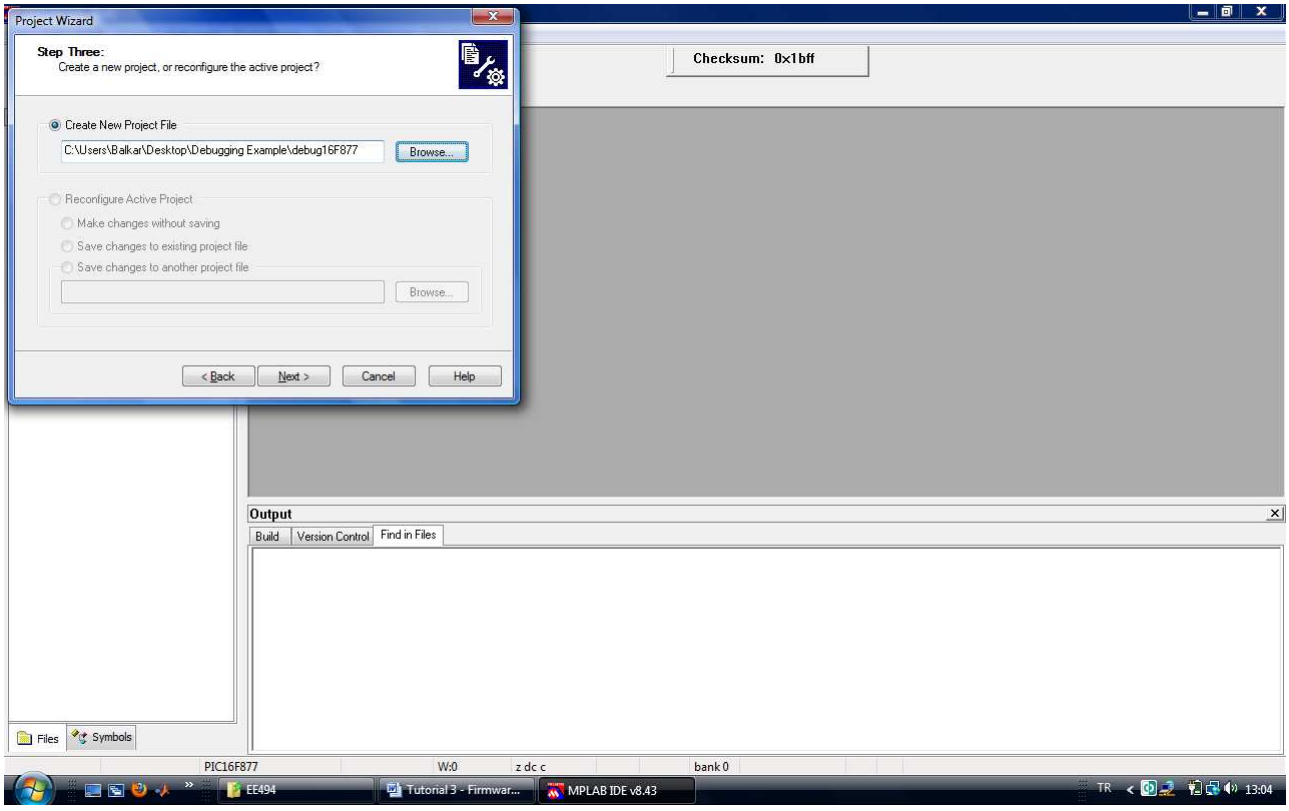

Figure 4: Specifying the Project Location

For now, do not add any files to your project; let's just create an empty project. Click Next.

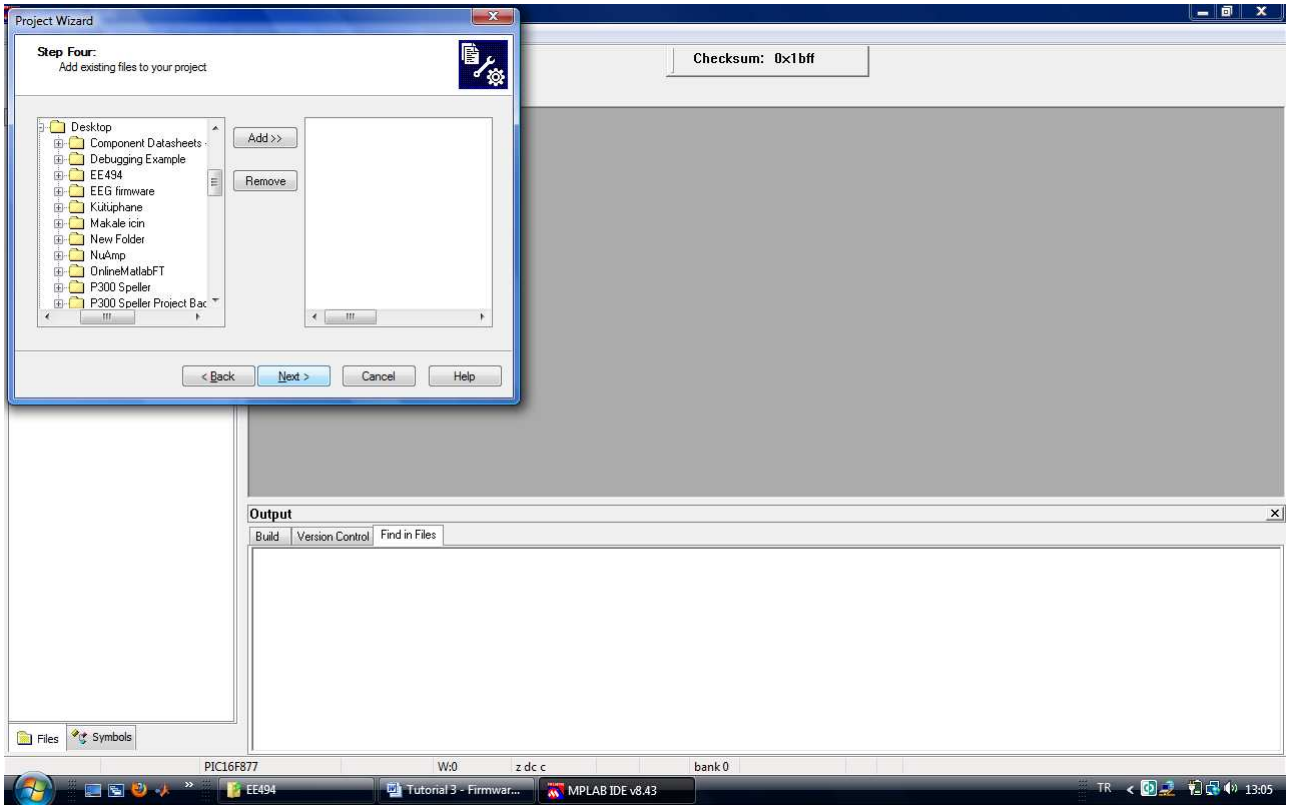

Figure 5: Addition of existing files

Check your project configuration and finish the wizard.

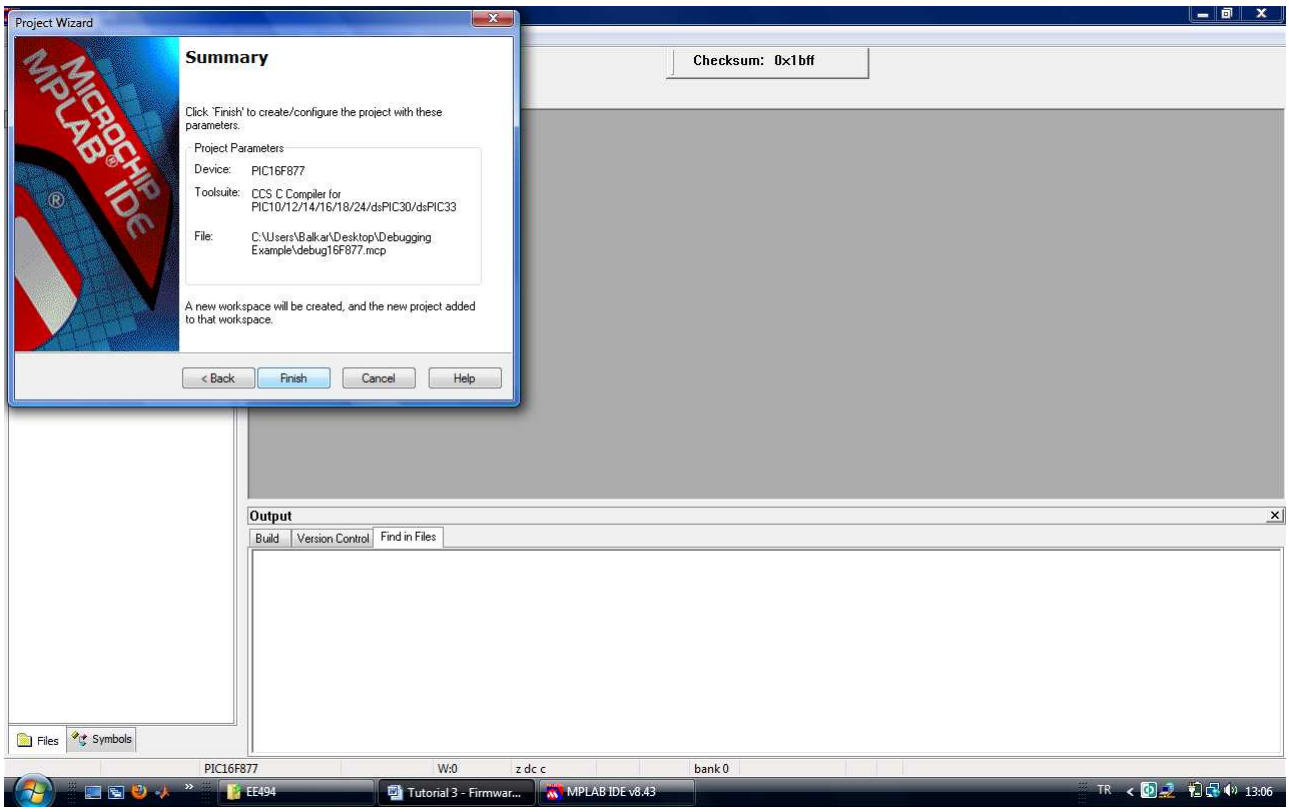

Figure 6: Creating an empty project - end of the Project Wizard

Dock your project and output windows. Click on the blank sheet (New File).

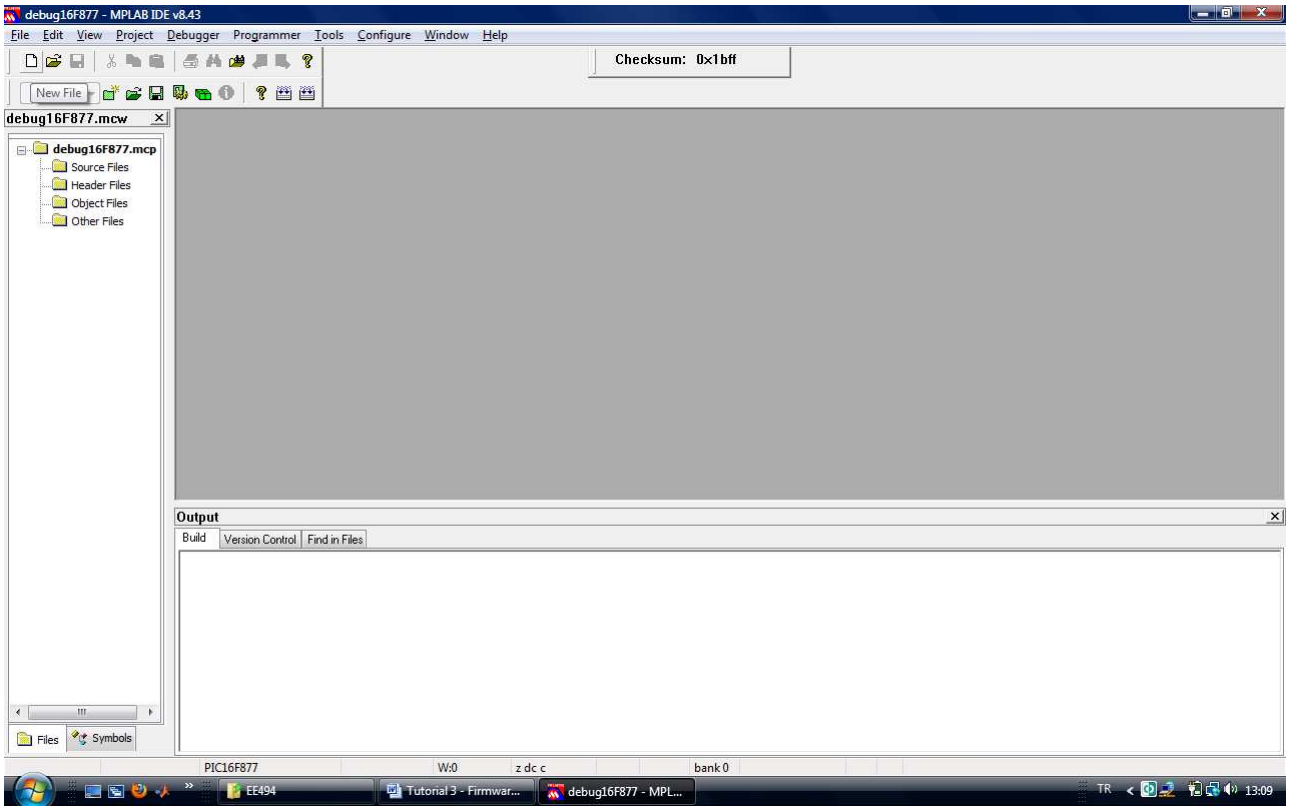

Figure 7: Formatting the Workspace and adding a new file

In the untitled document, copy and paste the following code

```
#include <16F877.h>
#fuses HS,NOWDT,NOPROTECT,NOLVP
#use delay(clock=20000000)

void main()
{
  int k;
  int j;
  double t;

  for (k = 1;k<10;k++)
  {
          output_high(PIN_D2);
          delay_ms(500);
          output_low(PIN_D2);
          delay_ms(500);
          j = 2*k+1;
          t = j/2;
  }
}
```

Save your file as main.c in your project directory.

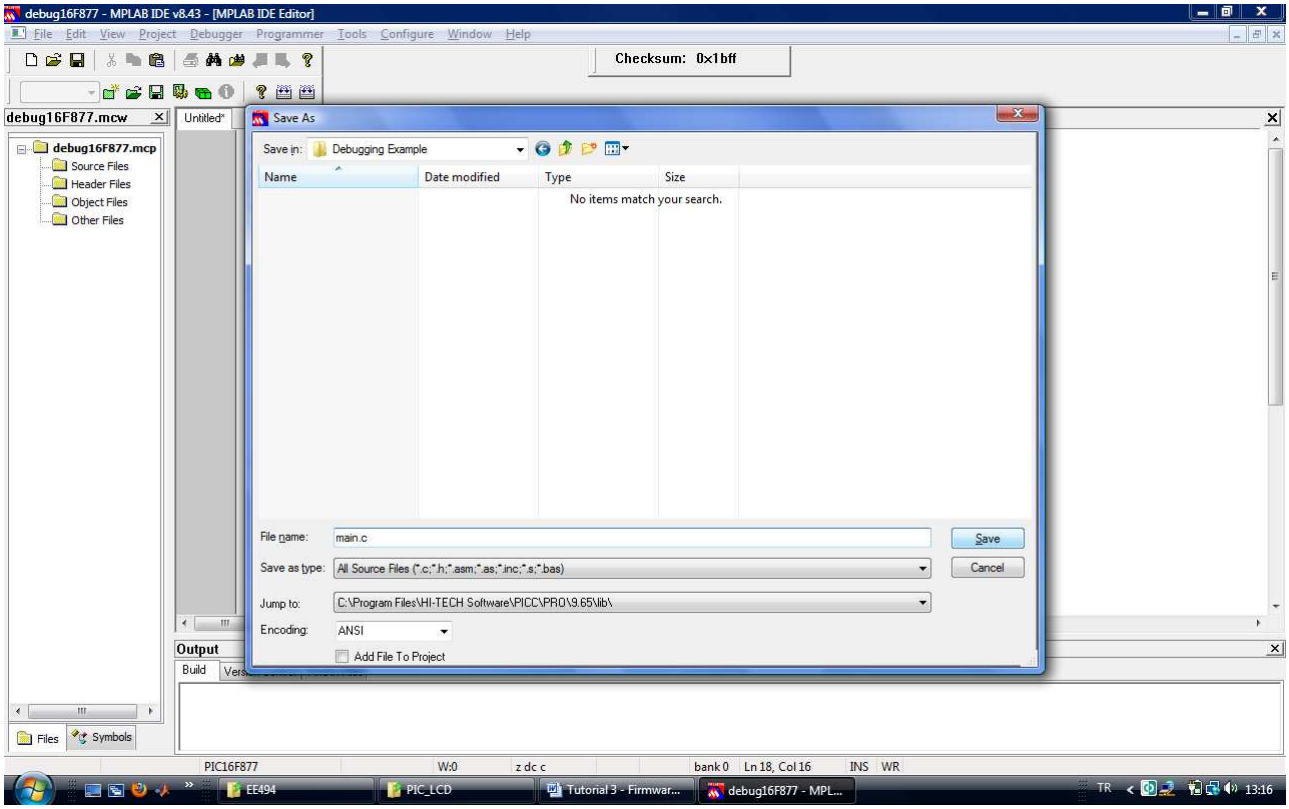

Figure 8: Saving the main program in the project directory

Now, right click on the Source Files->Add Files. Select the file "main.c" you just created.

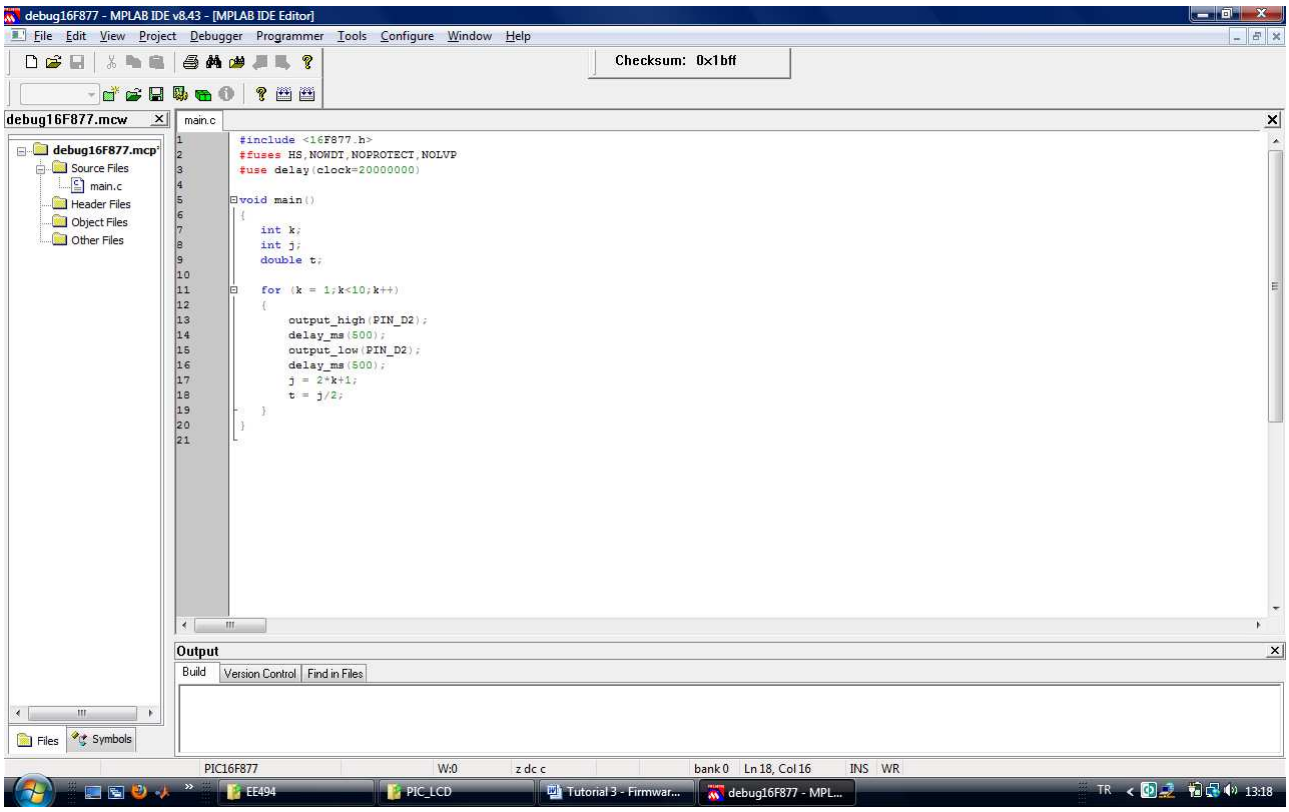

Figure 9: Inclusion of the created "main.c" file in the project

Click on "Build All", and you will see that new files are created in your project window.

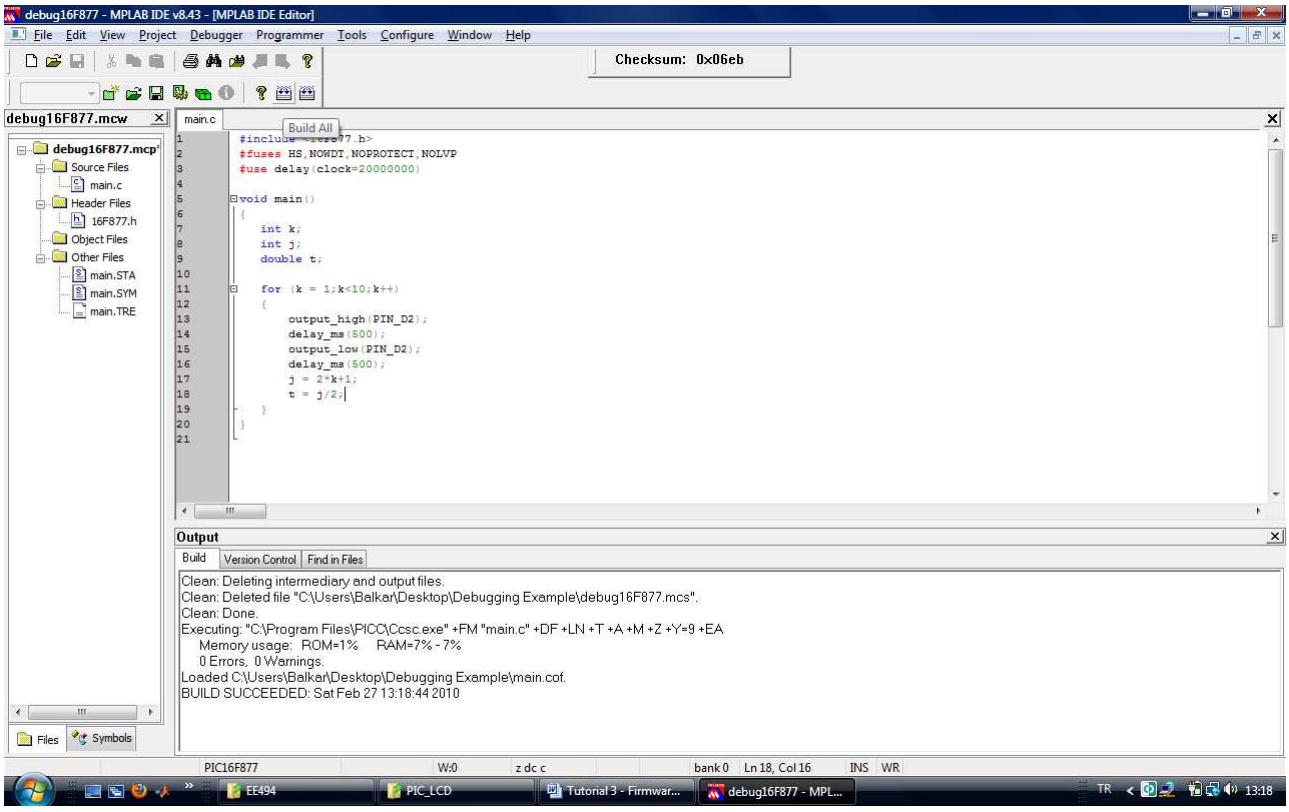

Figure 10: Building the code with CCS compiler

Now, let's program this code to our PIC16F877 device. First of all, you should construct the programming hardware (with PICKit2) discussed in tutorial 2 (Figure 5 of ICSP tutorial). You should use the PIC16F877 device as the target microcontroller and place a resistor (10k) and a LED pair to pin D2 of the microcontroller.

Plug your PICKit2 device to one of the USB ports of your PC and connect your application circuit and PICKit2 with the ICSP cable (an ordinary cable with 6 independent lines). Now, we will first program the device from MPLAB. In MPLAB, click on "Programmer", choose "Select Programmer" and select PICKit2 (Figure 11).

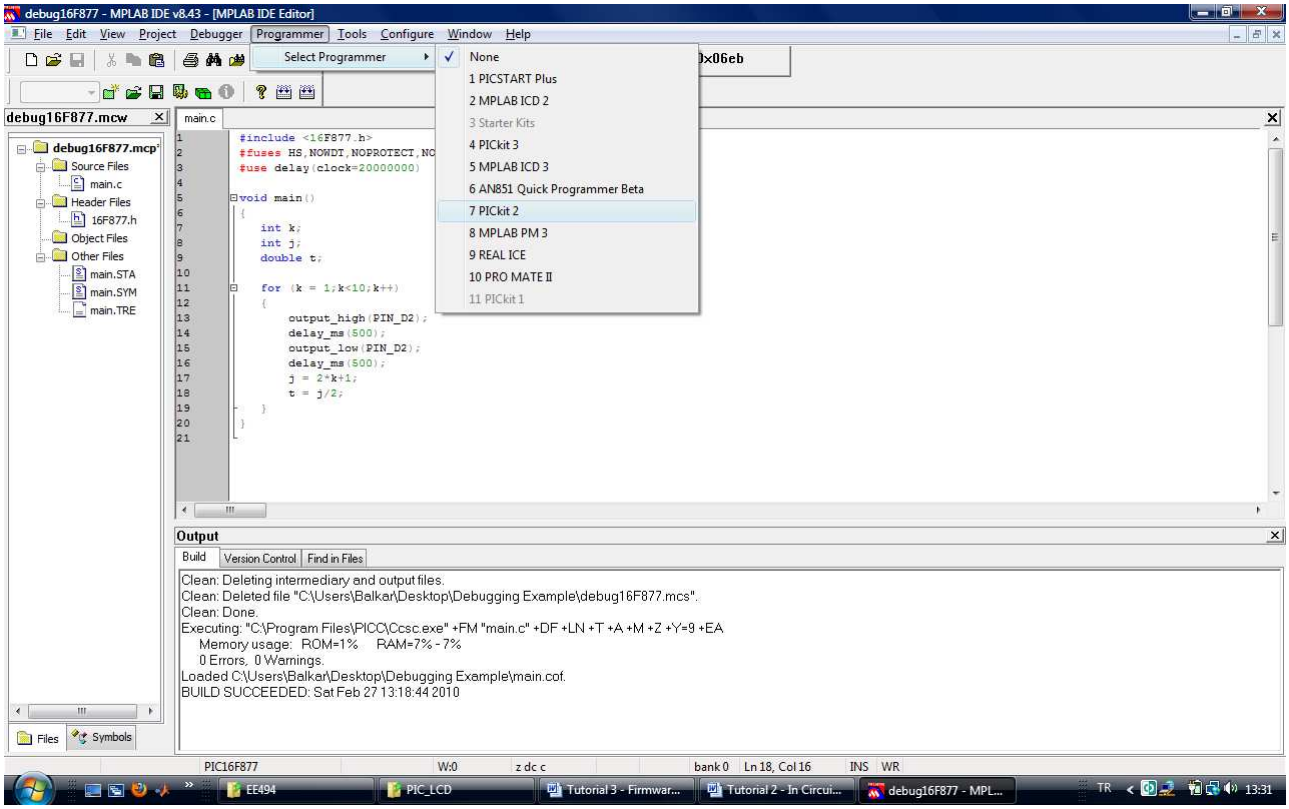

Figure 11: Programming the target device from MPLAB with PICKit2

Note that your PICKit2 device is identified by MPLAB (in PICKit2 tab, see Figure 12

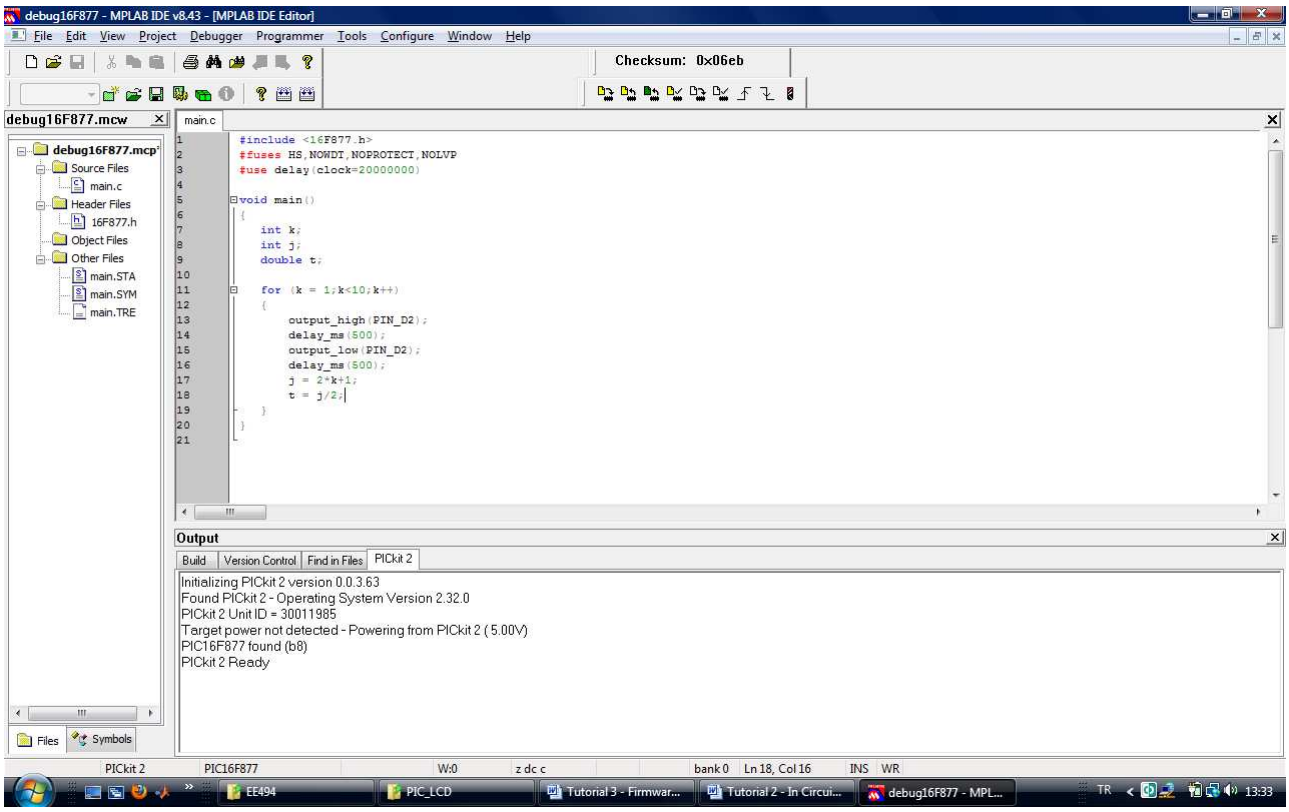

Figure 12: Identification of PICKit2 in MPLAB

Click on the "program the target device" icon (in Figure 13)

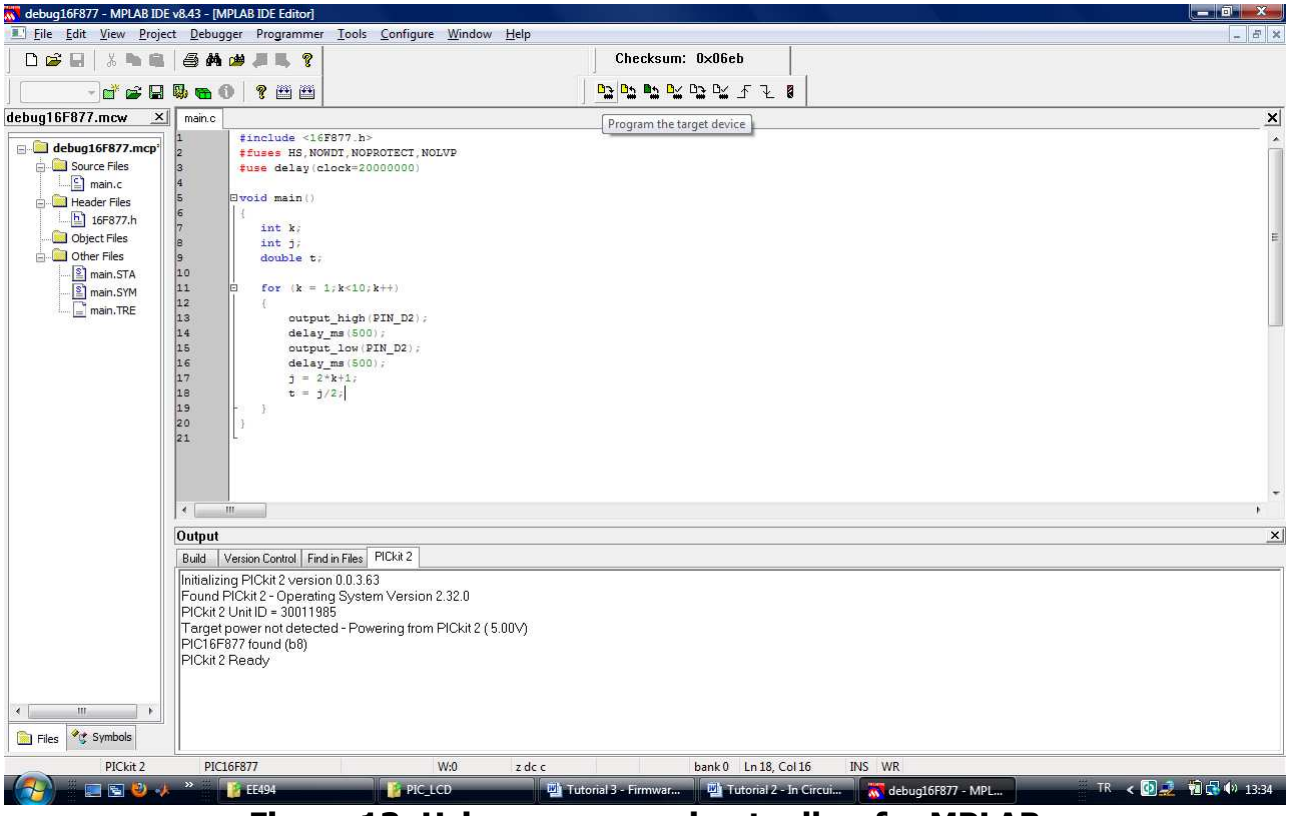

Figure 13: Using programming toolbar for MPLAB

Once the programming is verified, you can use your PICKit2 as a debugger.

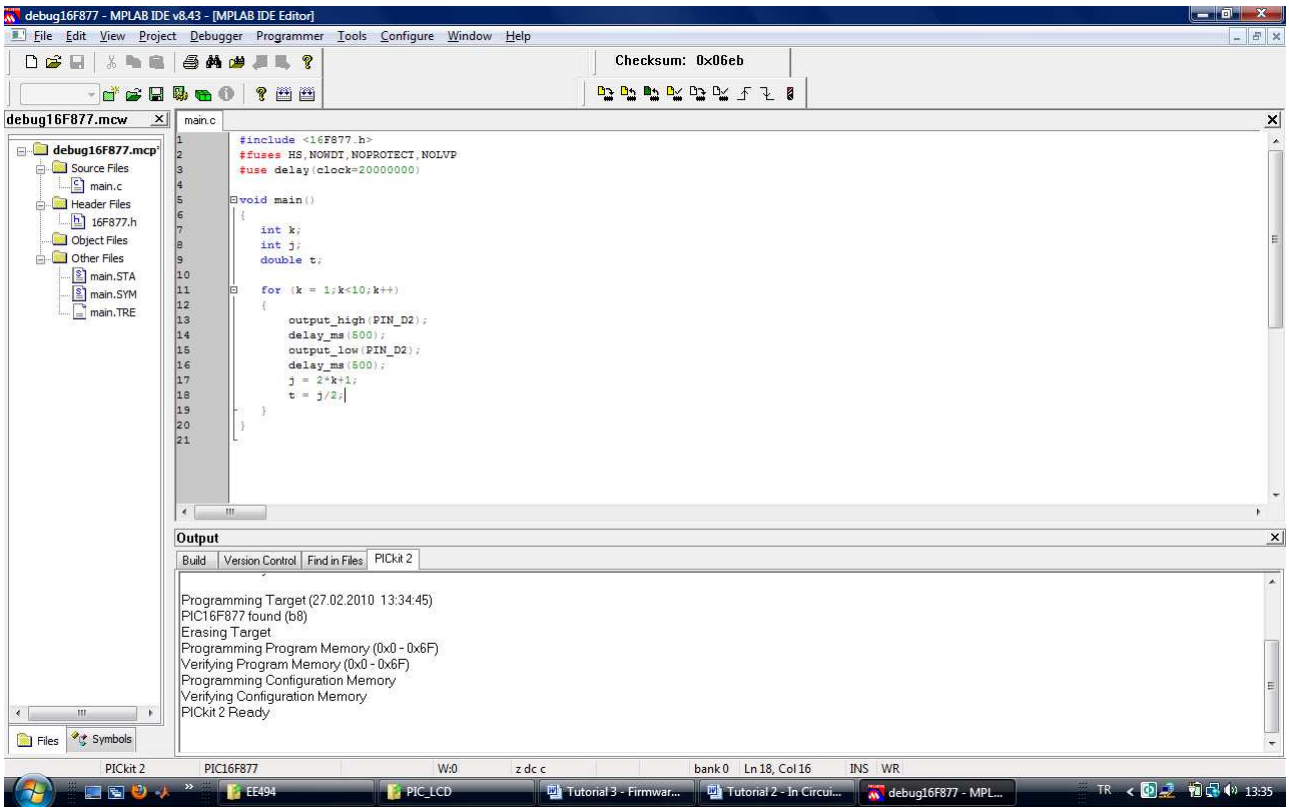

Figure 14: Verification of the programmed device

Select "Debugger", click on "Select Tool" and select PICKit2.

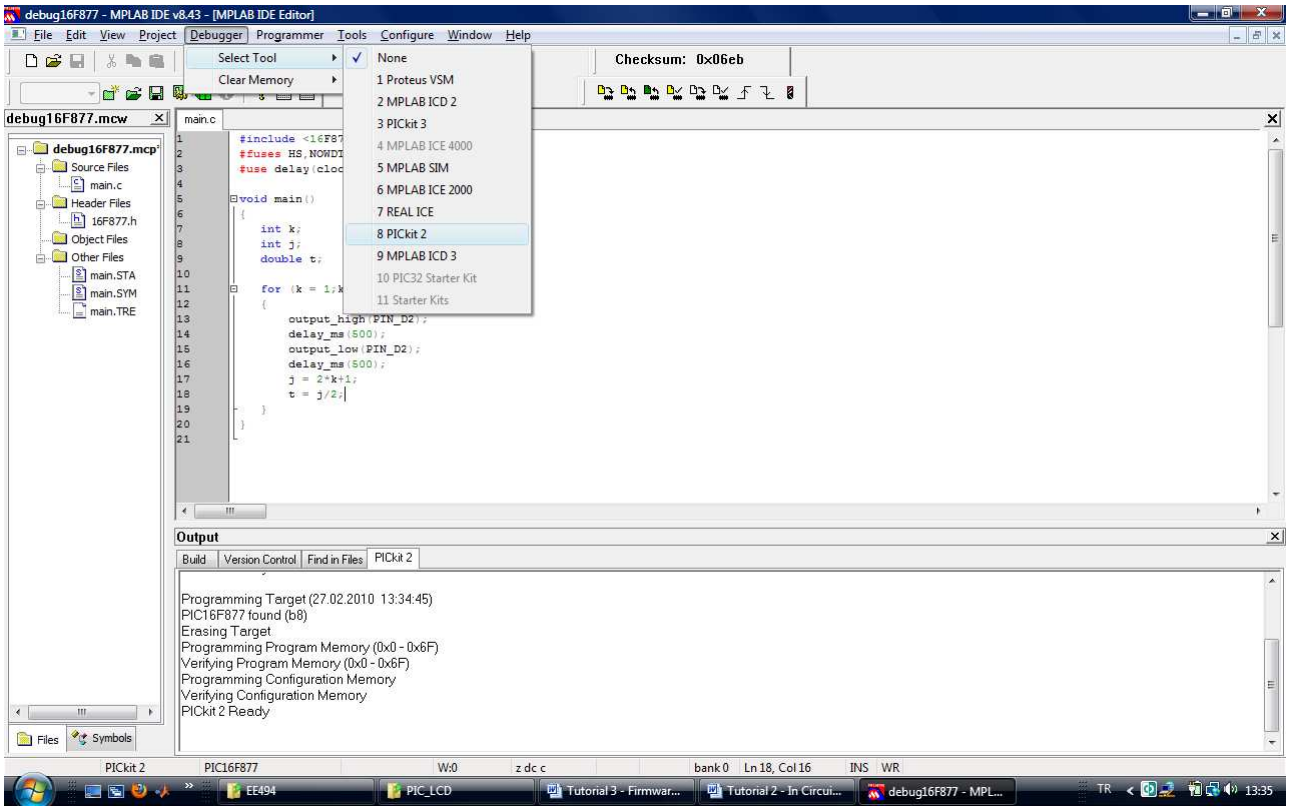

Figure 15: Switching PICKit2 to a debugger

You will be reminded that "a programmer and a debugger can not be used at the same time" and asked to switch to the debugger. Click OK.

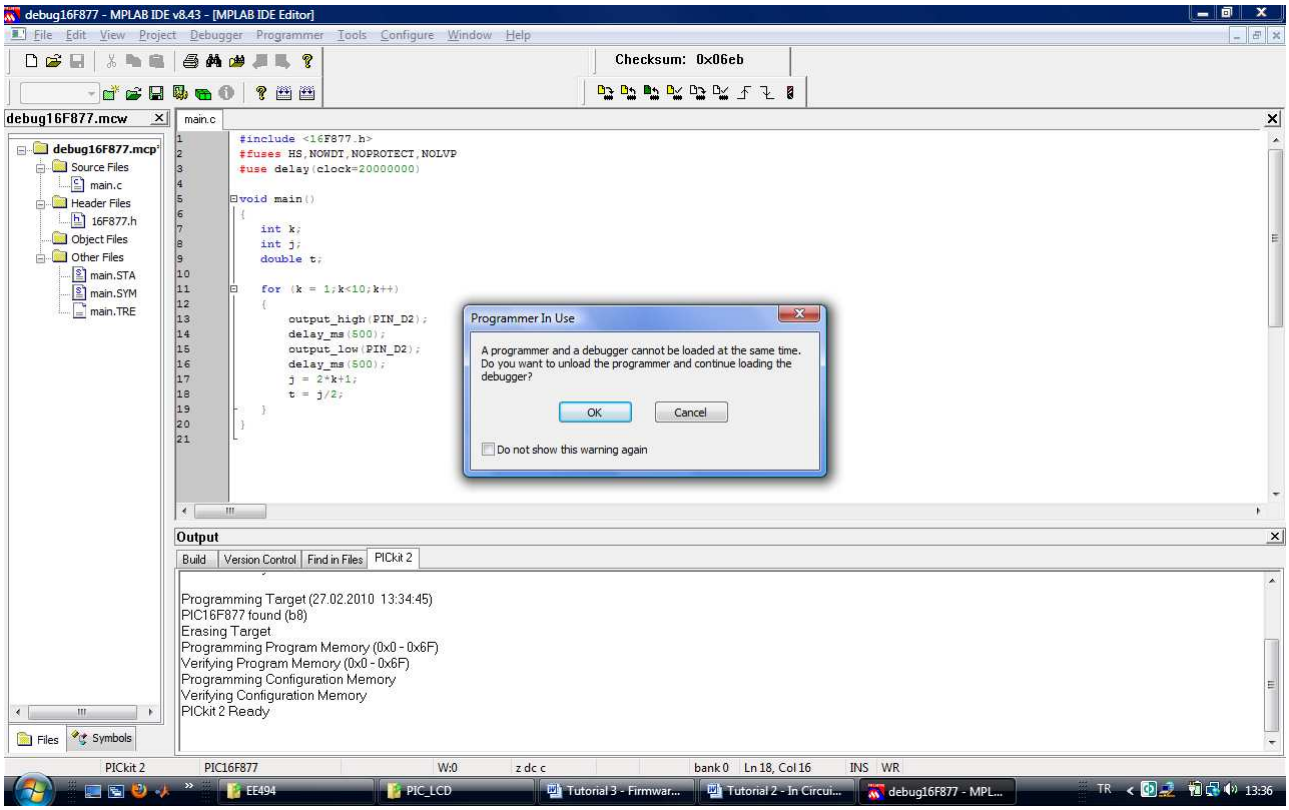

Figure 16: Configuration of PICKit2 as a debugger

Now, your PICKit2 serves as a debugger. Place a breakpoint on line 19 by double clicking on the gray bar including the line numbers (Figure 17)

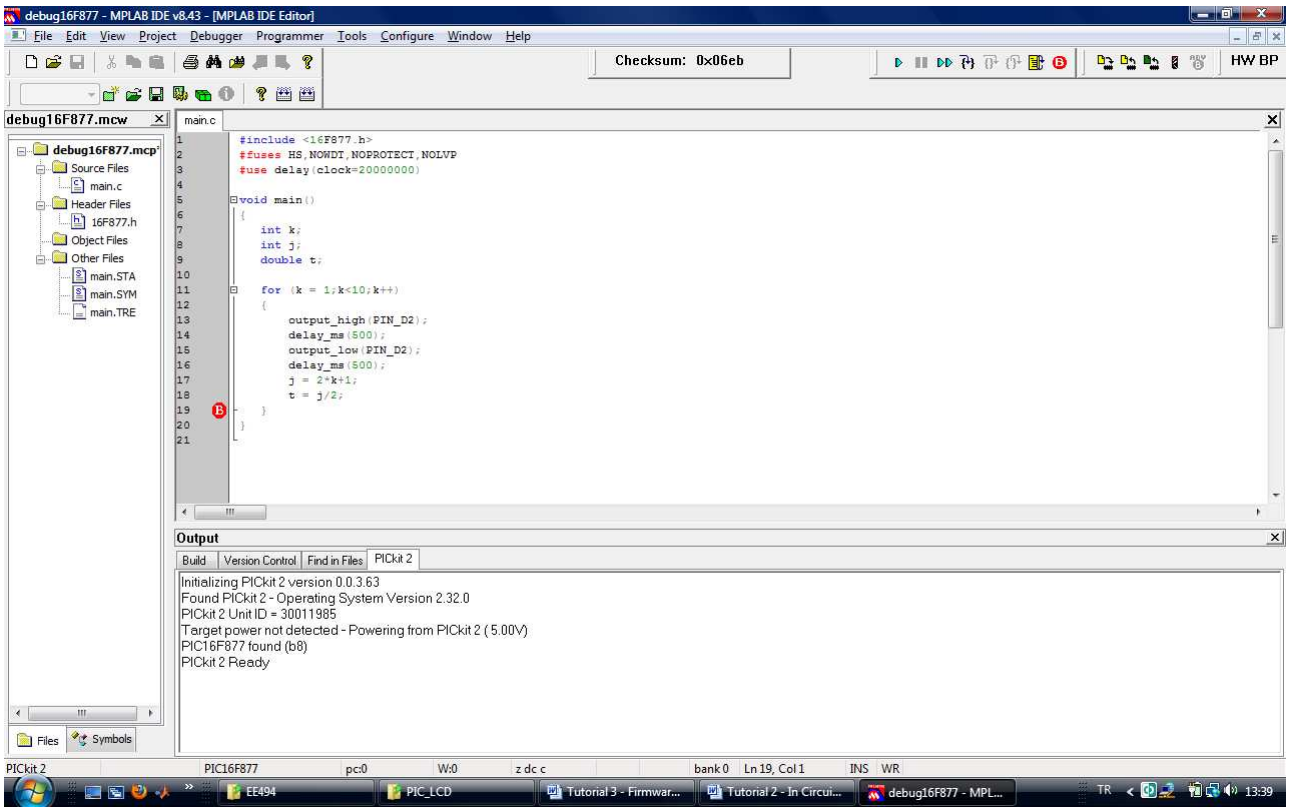

Figure 17: Placing a breakpoint

Click on the "Run" (play) icon on the debugging toolbar to go to line 19. You may be asked to recompile the code for the debugger as in Figure 18. Select "Yes" for that message.

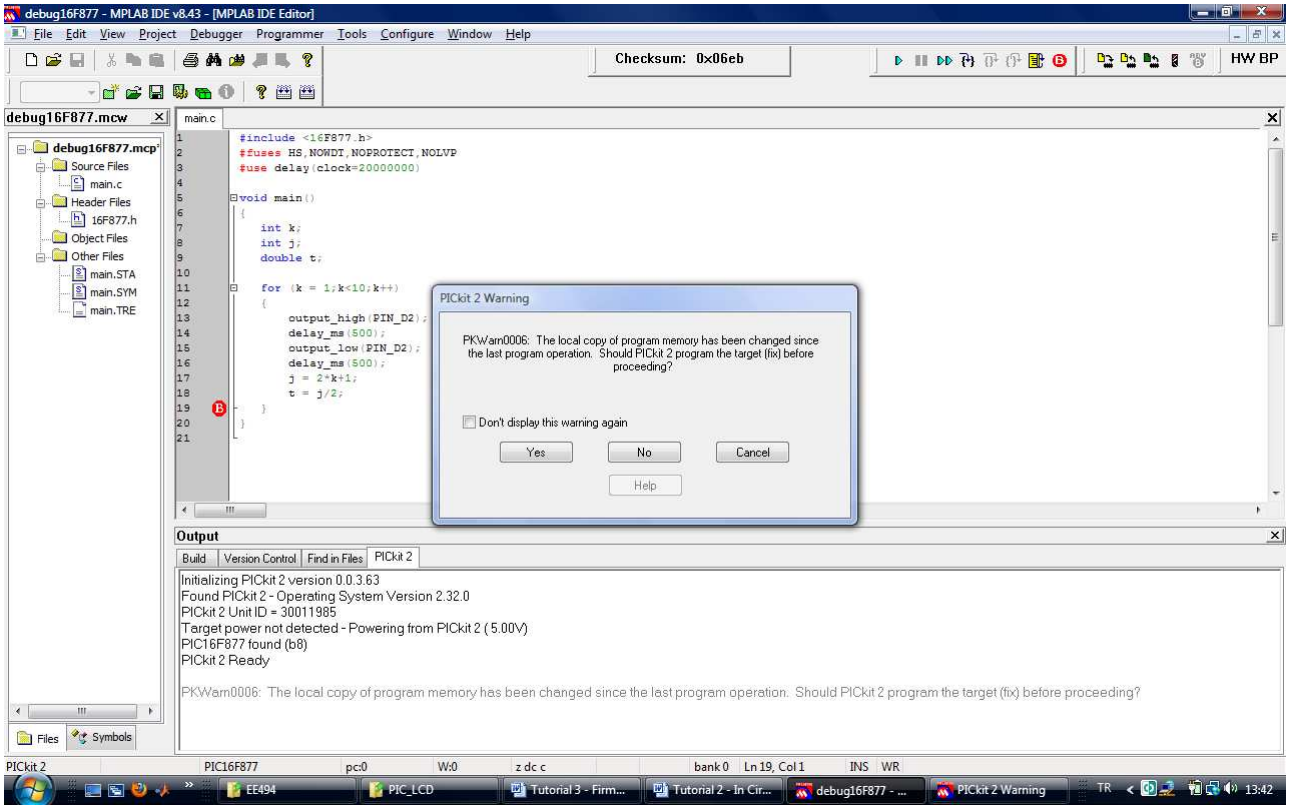

Figure 18: Configuring the firmware for debugging utilities.

Now, you can see that the operation is halted at line 19 and you can analyze every variable by placing your mouse on the variables.

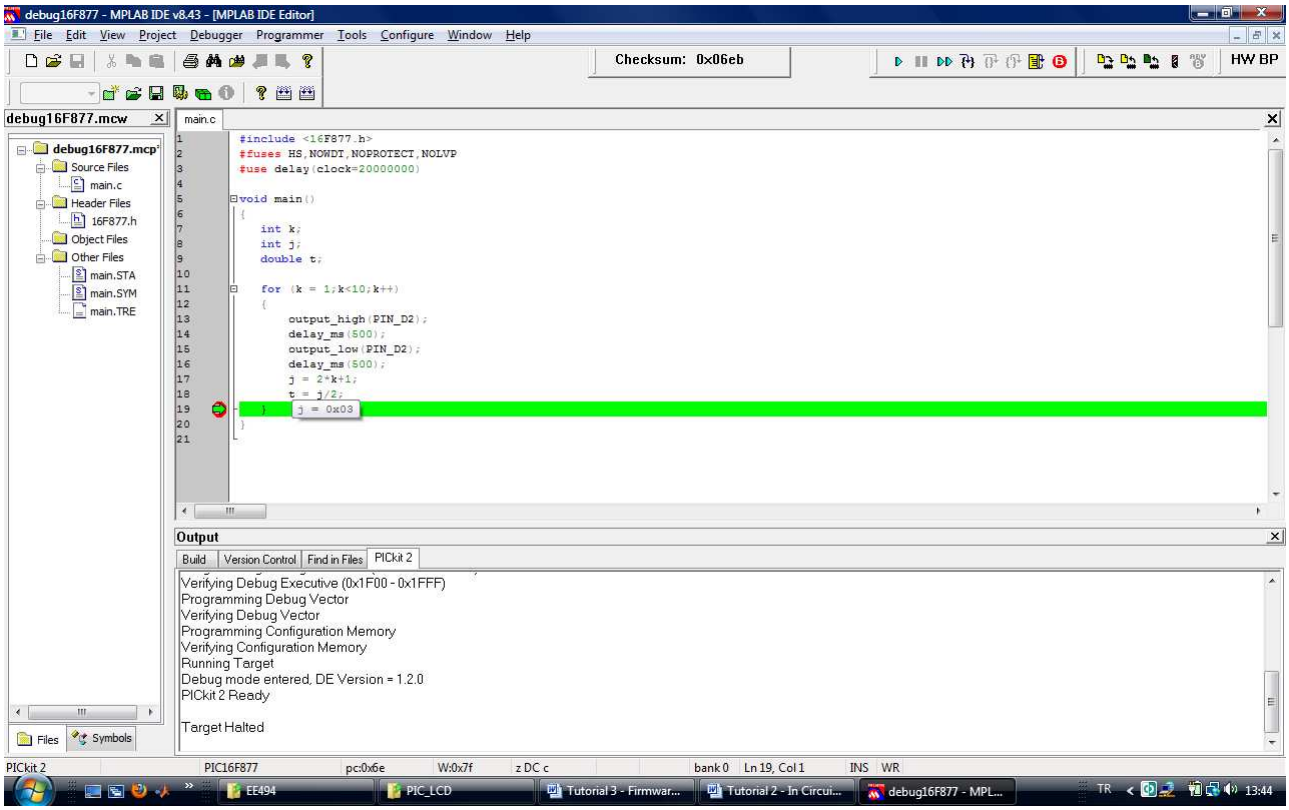

Figure 19: Investigation of variables in debugger

You can also see every variable and physical memory values (including stack, EEPROM etc.) from "View"->"Program Memory", "View"->"Locals" etc.

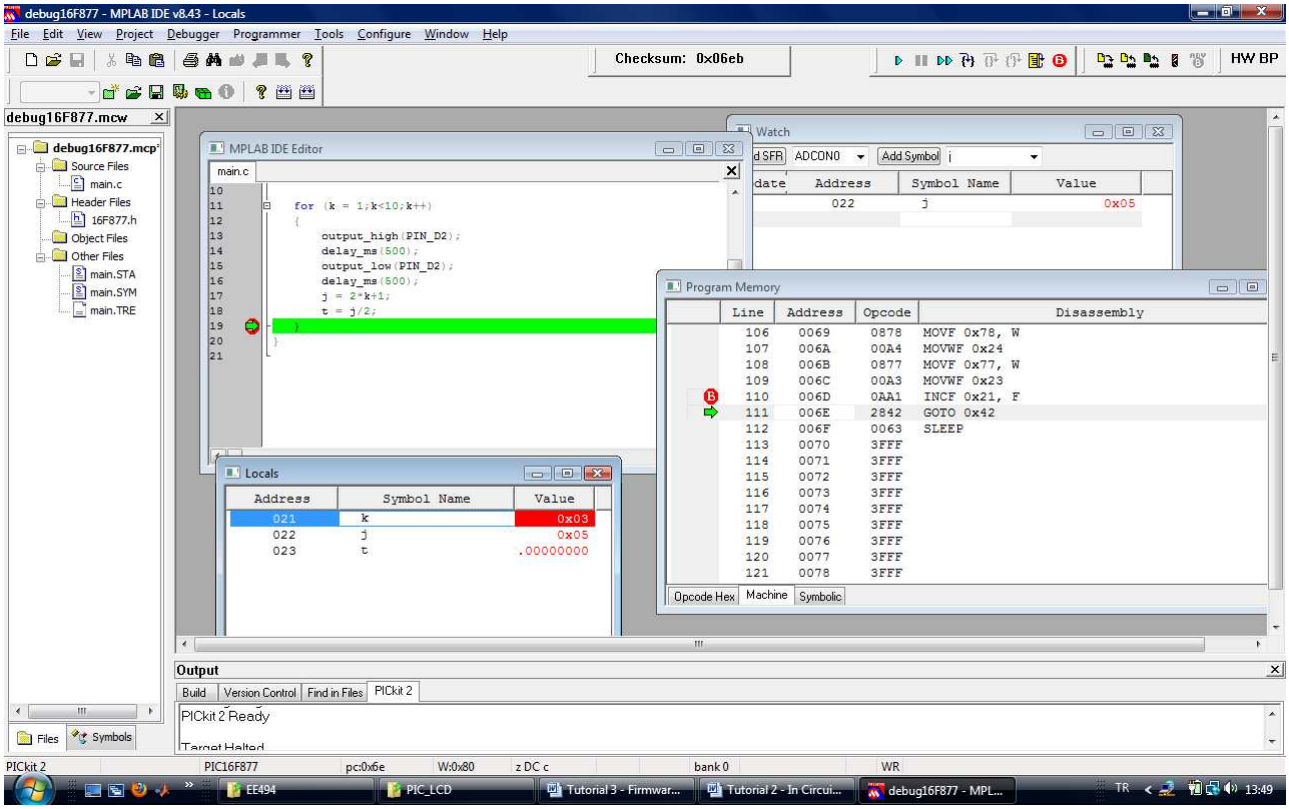

Figure 20: Using "Program Memory", "Locals" and "Watch" windows in debugger