1.2 - Introduction to the IAR Workbench IDE*

Naren Anand

Based on *Introduction to CrossStudio MSP430 IDE*† by

Kileen Cheng

This work is produced by OpenStax-CNX and licensed under the
Creative Commons Attribution License 2.0‡

Abstract

A quickstart tutorial to the IAR Workbench IDE. Learn how to create a project, edit files, build solutions, and use the debugger.

1 Goals

To develop applications to run on the ez430 chip, we use the IAR Embedded Workbench IDE (integrated development environment). Not only does this application provide a powerful code editor, but it also allows a simple one-click deployment of the source code onto the MSP chip using USB as well as hardware debugging capabilities that allow you to trace through actual stack calls. This module is intended to give you an introduction to the IAR Workbench application so that you may create and develop your own ez430 applications.

2 Create a Project

The very first thing you must do before you can start downloading any code onto the MSP, is to create a project in Workbench that will contain all of the relevant files for your application. When you open Workbench you should see the following window (This window is equivalent to selecting **File->New->Workspace** and then **Project->Create New Project**. This window may also be reached through Help->Startup Screen):

*Version 1.1: May 18, 2006 10:54 am -0500
†http://cnx.org/content/m11692/1.2/
‡http://creativecommons.org/licenses/by/2.0/

Startup Screen

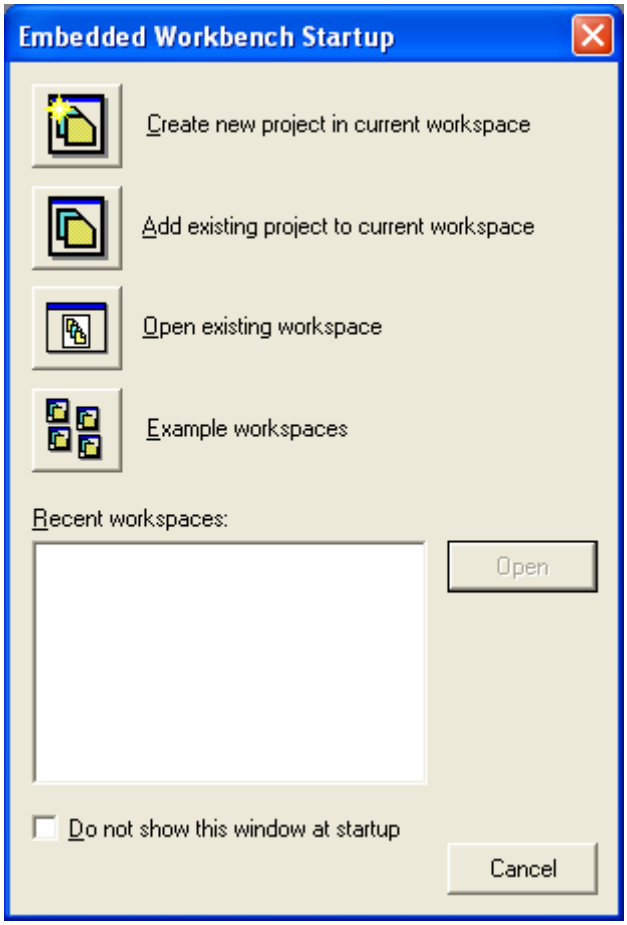

Figure 1: Select "Create new project in current workspace" to begin.

Next, the following window will appear:

Project Template Selection

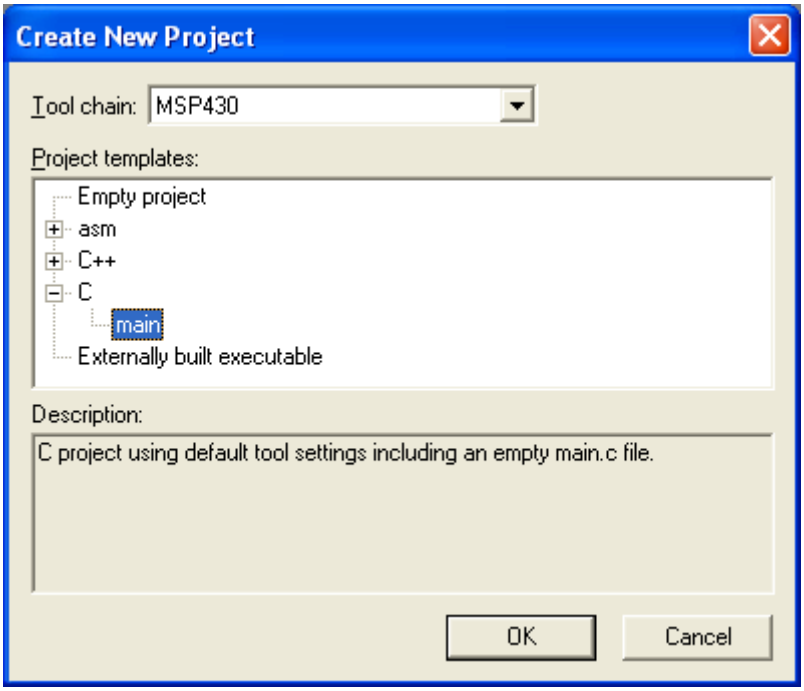

Figure 2: Here we select the language in which we shall write our code. Select the 'C' file and then the 'main' option. In one lab, we will program in assembly; for that lab, select the "asm" option.

Now a Windows Save dialog should appear. Give your project a name and save it in its own **seperate folder**. A project is not only a C file that contains your code but also several other files (a project file with the ".eww" extention, a project settings folder, a debug folder which contains compiled code outputted by Workbench etc). Keeping the files for your current project in their own folder will help you stay organized. If you save and close your project, opening it again is as simple as opening this folder, and doubleclicking on the Workbench project file, which has a ".eww" extention.

3 Configure Project Options

Now, we must configure the project options. Make sure the project is selected in the "Files" pane to the left of the screen (the project is the top of the workspace's file tree). Then select **Project->Options** to get the following screen and make the indicated selections:

Project Options Setup: General Options

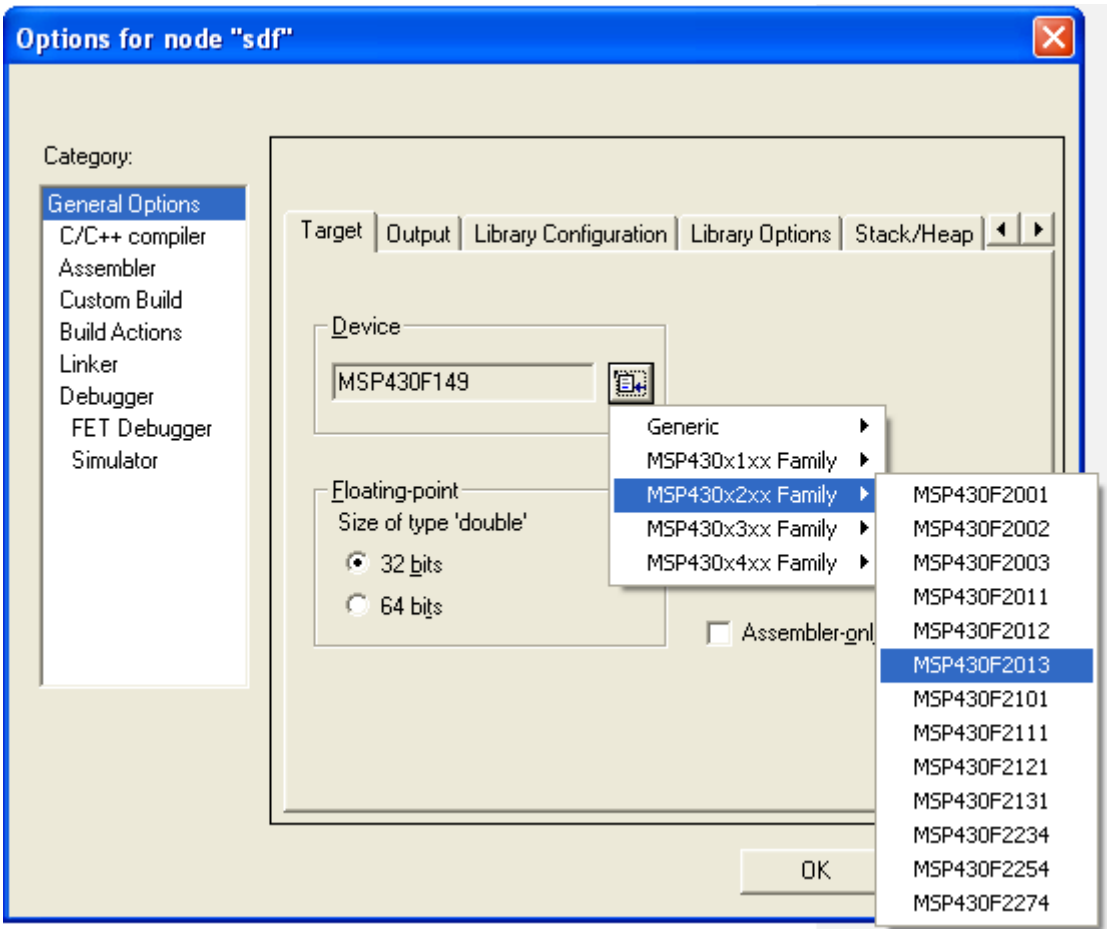

Figure 3: Make sure you choose the right processor for your application! Here, I've selected the MSP4302013 (a member of the ez430 series) since that is the chip I plan to use.

Now select the "Debugger" option and make the following selections:

Project Options Setup: Debugger

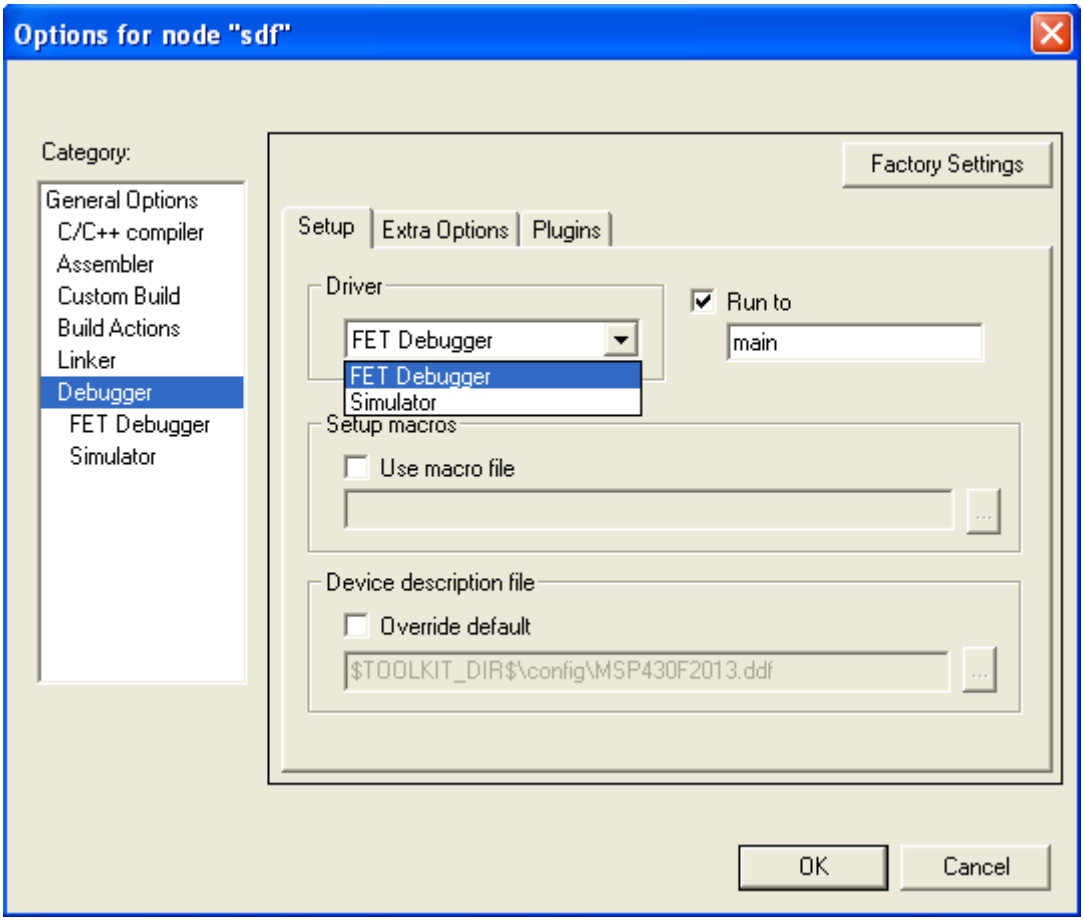

Figure 4: Here we must select to use the FET (Flash Emulation Tool) Debugger to run our code.

Finally select the "FET Debugger" option and make the following selections:

Project Options Setup: FET Debugger

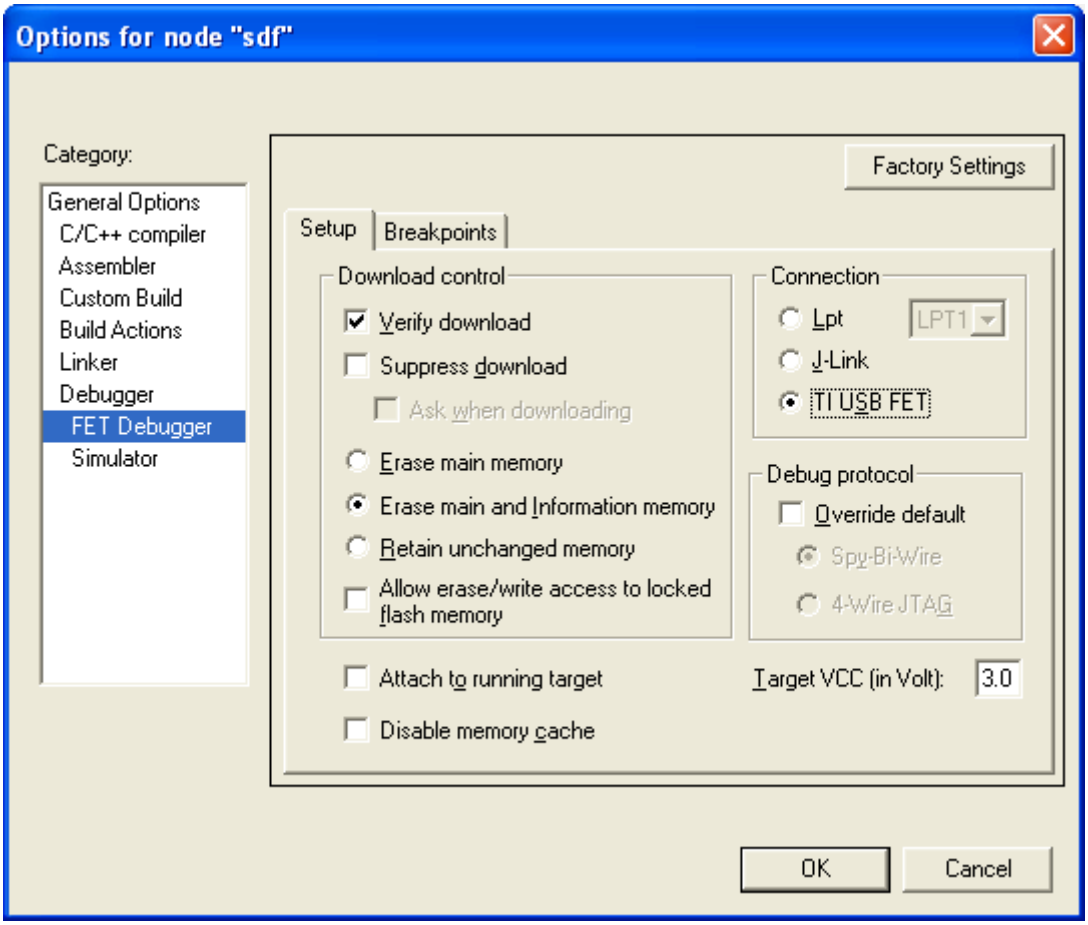

Figure 5: Now we select the "Verify Download" check box and the "TI USB FET" radio button (while leaving other options as they are) so that our program is properly transferred to the chip.

Your project is created, saved, and now you may write your code. Only one project may be open at a time.

4 Adding Files to the Project

If your project consists of multiple files, you will need to add them to the current project. To add a file to your project, choose **Project->Add Files**.

5 Run Your Program!

This is the part you've been waiting for: actually testing your program! Use the **Project->Debug** menu item to load the currently active project and set your program running. You can also set any breakpoints beforehand; by default, the very first line of executed code is set as a breakpoint. Now that you have started the debug process, the layout of the Workbench program changes slightly. A Debug menu and the "Disassembly", the assembly/machine code instructions at their specific memory addresses on the ez430,

appear. To continue the execution of the program, select **Debug->Go**. If you would like to set breakpoints (in order to view the "path" of program execution or the values of specific variables or registers at a certain point during the program execution) click to the left of a line of code such that a red 'X' appears.

You can pause the debugger when the target is running if you would like to look at the value of certain variables. Select **Debug->Break** and open a watch window to examine the value of certain variables. It is suggested that you copy the variables you are interested in to temporary global variables. Because local variables go out of scope, it is uncertain if their correct value is maintained when the debugger is paused.

Debugging can be stopped using **Debug->Stop**. At this point, the watch window will not display your variable values. At this point, you may make any modifications to your program as necessary and restart the debugger from the beginning.

All of the necessary commands have shortcut icons as the following diagram shows:

Workbench: Debug View

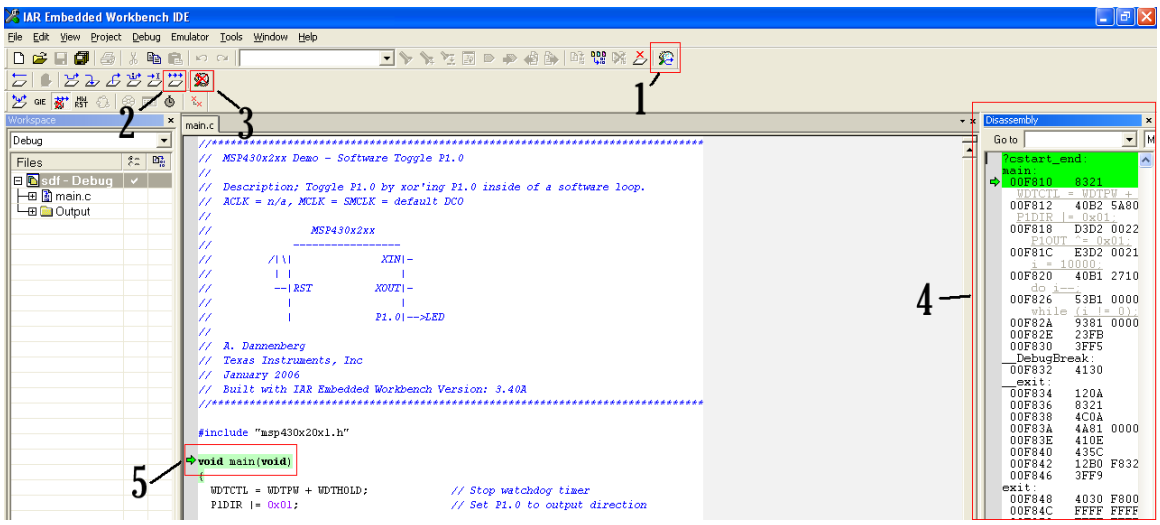

Figure 6: The program layout during debugging

Legend:

1. **Make/Debug-** This button will compile, load, and run the code onto the chip. This process happens rather rapidly so if the execution of this seems to hang, then something is wrong. Usually disconnecting the tool and/or restarting Workbench solves this problem. Don't forget to save first!
2. **Go-** This button is for the Go command in the debug menu; it tells the program to continue execution until the next breakpoint. The buttons to its left are other ways of traversing through breakpoints that you may experiment with.
3. **Stop Debug-** Selecting this will end the debugging process. If you want to modify your code then you must stop debugging first.
4. **Dissassembly-** This is a map of the memory space of the ez430 and where your code has been placed (in its compiled form).
5. **Current Location-** The Green Arrow/Green Highlighting indicates the current instruction being executed.

6 Good luck!

You're all set to start using the CrossStudio compiler to write embedded microcontroller code. Save often and remember that disconnecting and reconnecting or restarting the program will solve almost all interface problems that you encounter. Don't worry if breakpoints seem a little confusing now. We will go into more depth shortly and a good bit of practice is necessary to use them effectively.