

Tools required.

Now's a good time to setup your environment for generating CFunctions.

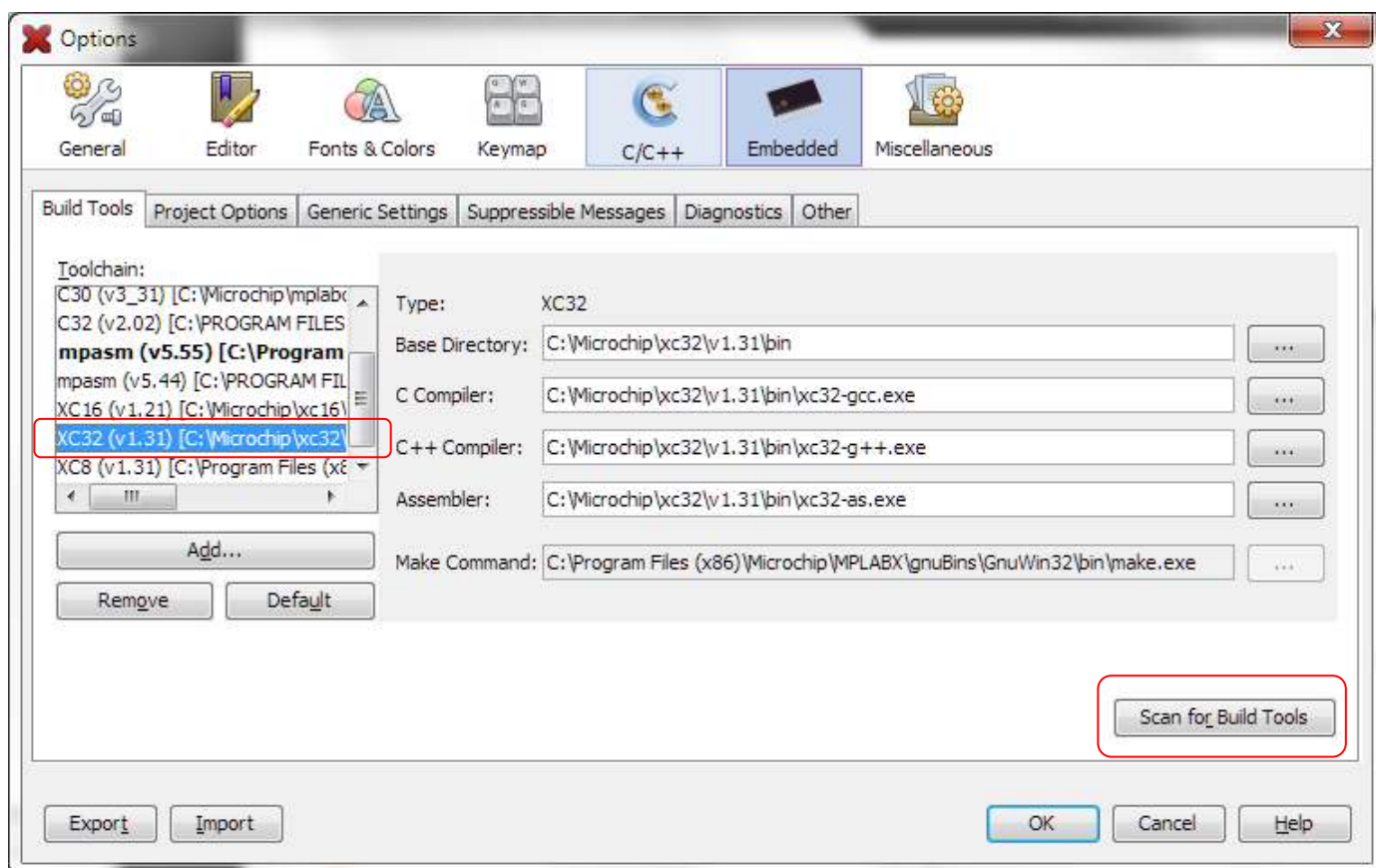
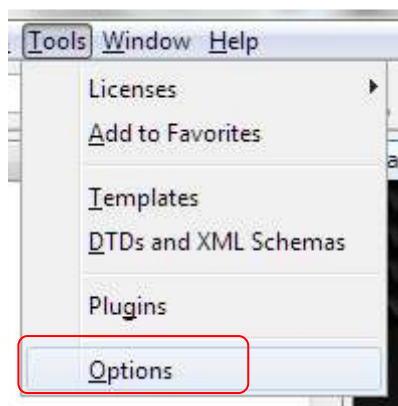
Compiler

You will need a PIC32 C Compiler. The easiest and quickest route is to download and install MPLAB X and the XC32 compiler from Microchip - <http://www.microchip.com/pagehandler/en-us/family/mplabx/>. The 'free' version is quite suitable for writing CFunctions. For the adventurous, xc32 is really just a Microchip packaged version of gcc, and it is possible to install gcc which will then give you full access to all of the Optimization levels without paying Microchip. Just GOOGLE for 'PIC32 gcc windows install' and you should turn up several How-To's. All the tutorials in this document have been compiled with xc32 under MPLabX.

MPLab/X and XC32 setup check

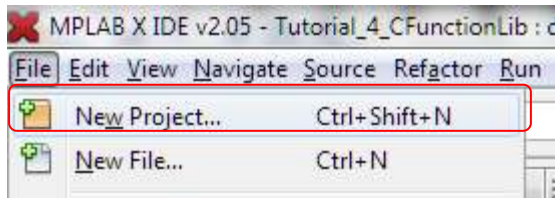
After installing XC32 and MPLabX, makes sure that they are setup correctly.

In MPLAB/X, do Tools | Options and select the Build Tools tab

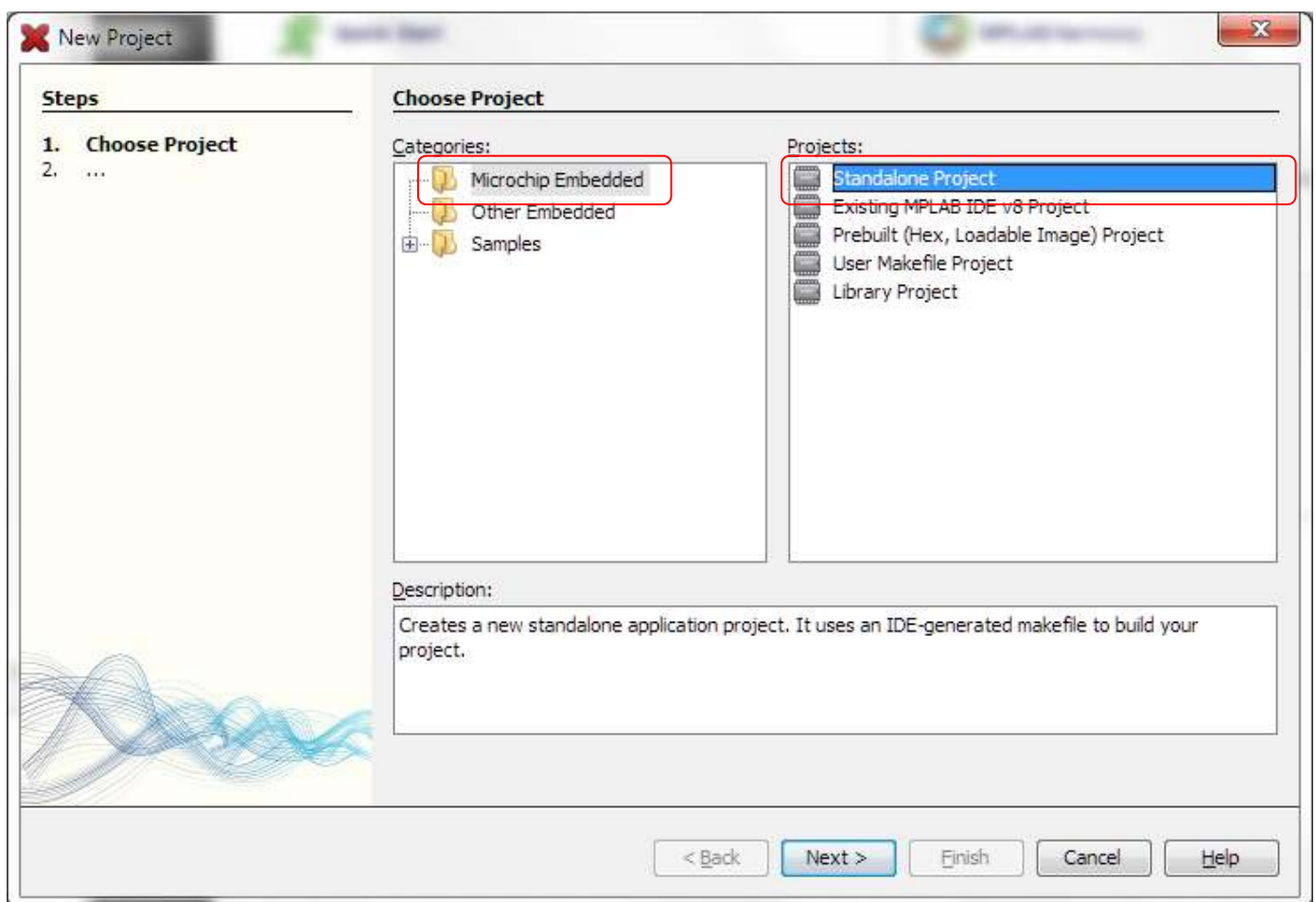


if XC32 is correctly configured with MPLAB/X you should see XC32 in the Toolchain list. If XC32 is not displayed in the Toolchain list, then click Scan for Build Tools which will cause MPLab/X to scan your harddrive looking for MPLab/X compatible toolchains, eg XC32. Until XC32 shows up as an item in the Toolchain list, MPLab/X will be unable to compile PIC32 C codes. In extremis, you will need to uninstall MPLab/X and XC32 and try re-installation. One thing I have noticed is that some applications which have their roots in UNIX/LINUX, eg GNU Tools, don't like directory names with spaces in them, eg "Program Files(x86)", so I tend to always install Microchip tools into the root directory, eg c:\microchip.

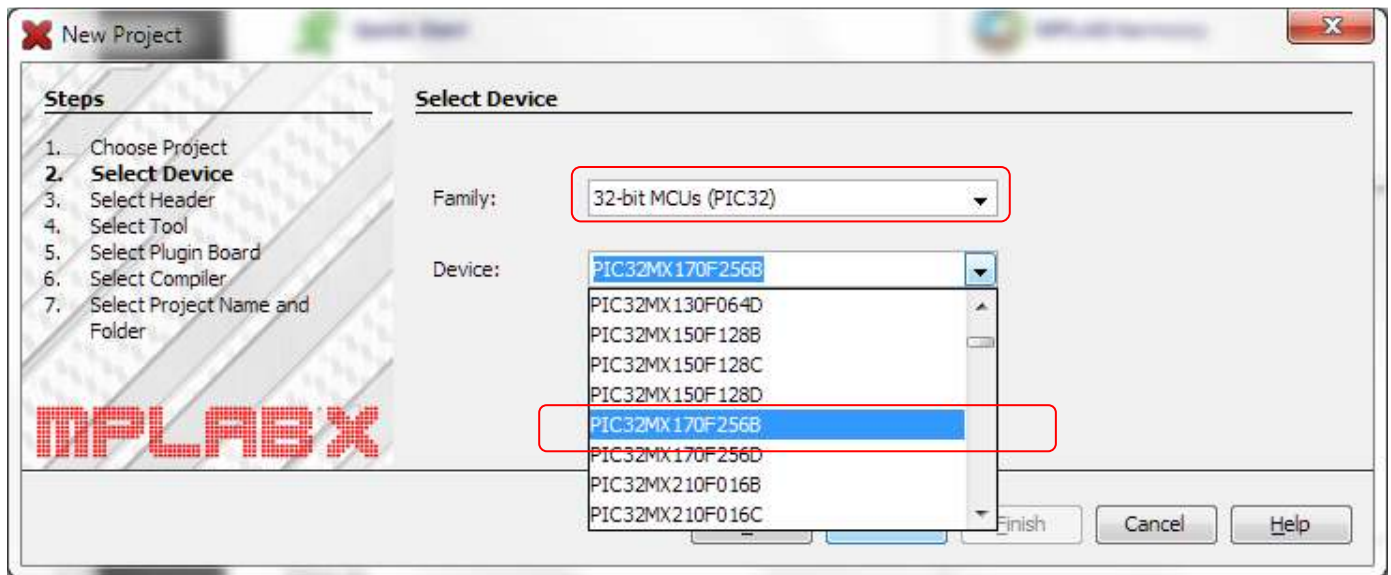
Next create a test project. Do File | New Project



Select Microchip- Embedded, and Standalone Project

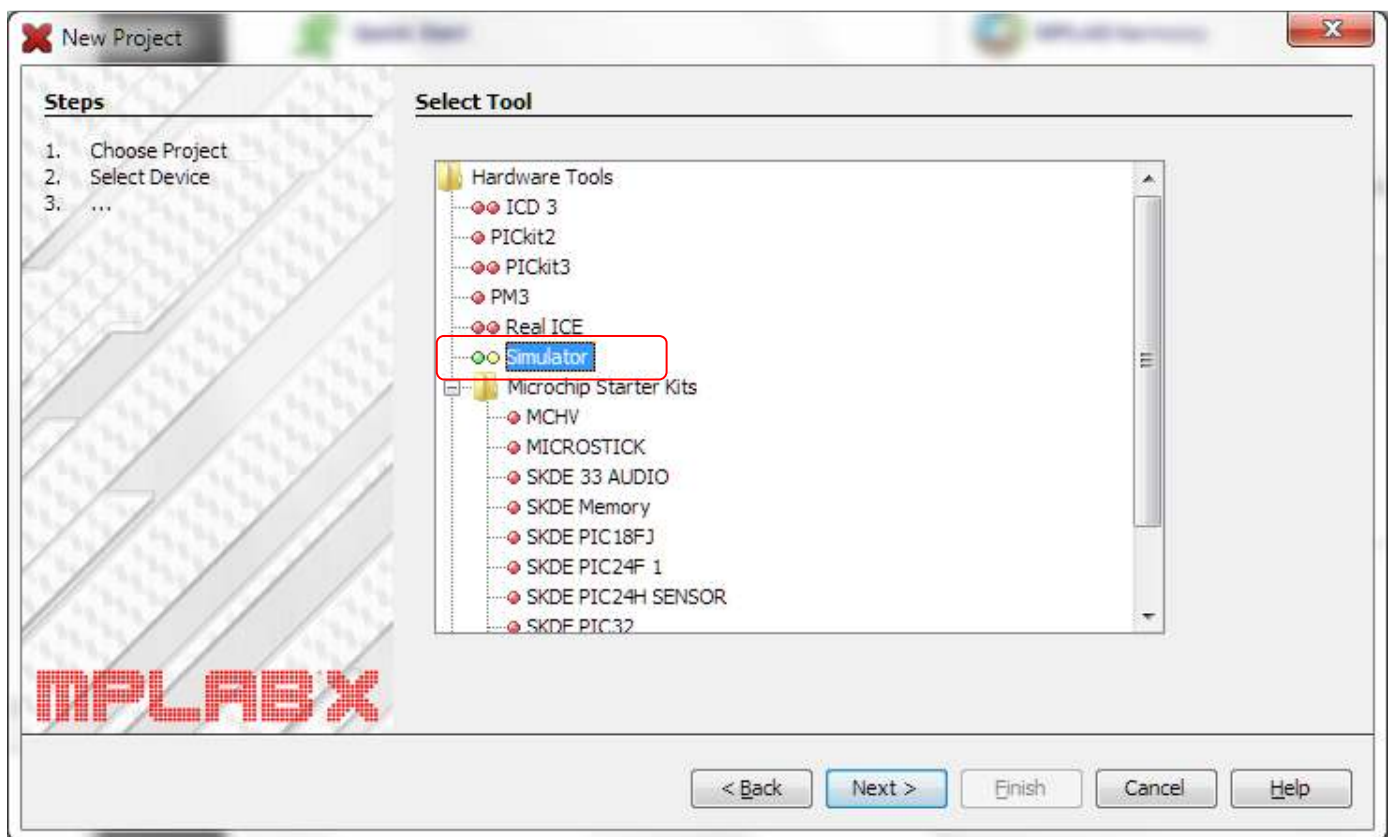


Click Next

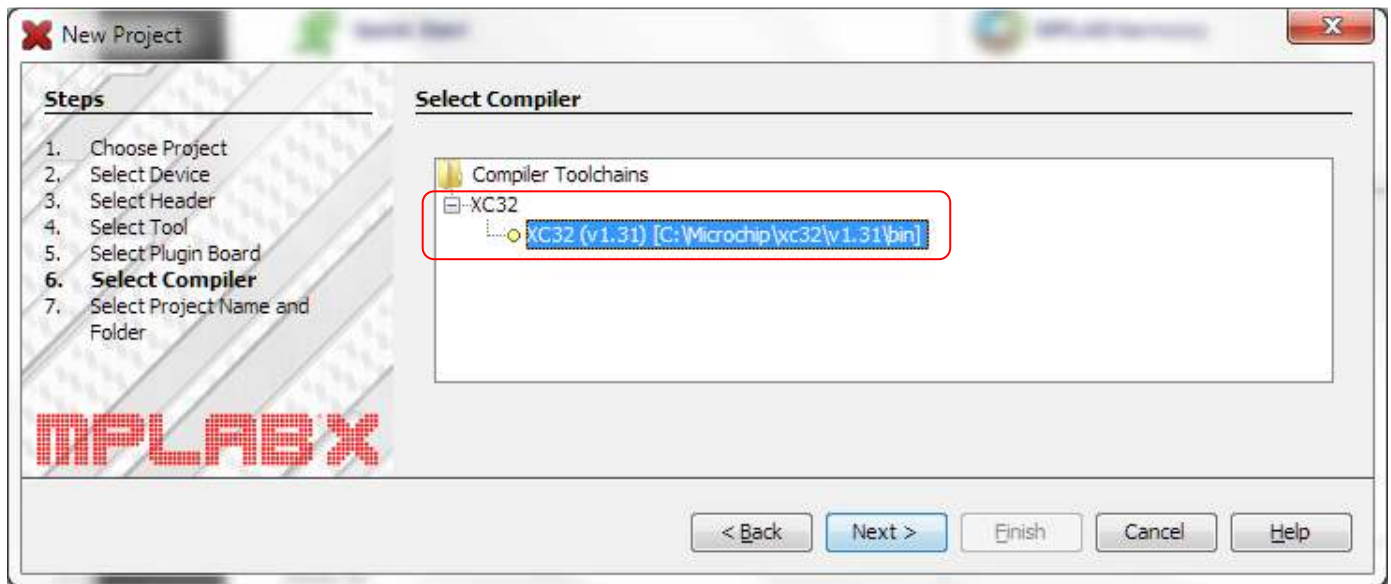


Select 32 bit MCUs (PIC32) and Device PIC32MX170F256B

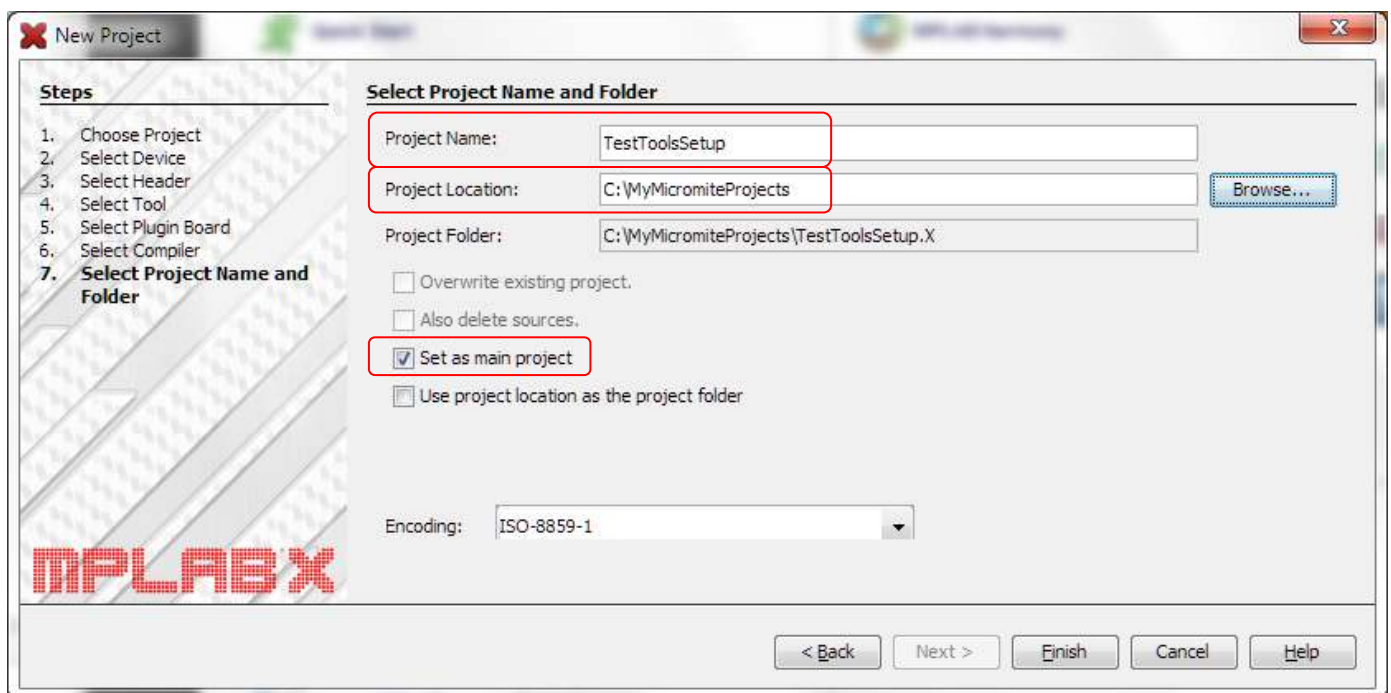
Click Next



Select Simulator. Click Next



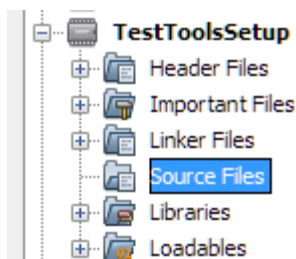
Select XC32. Click Next



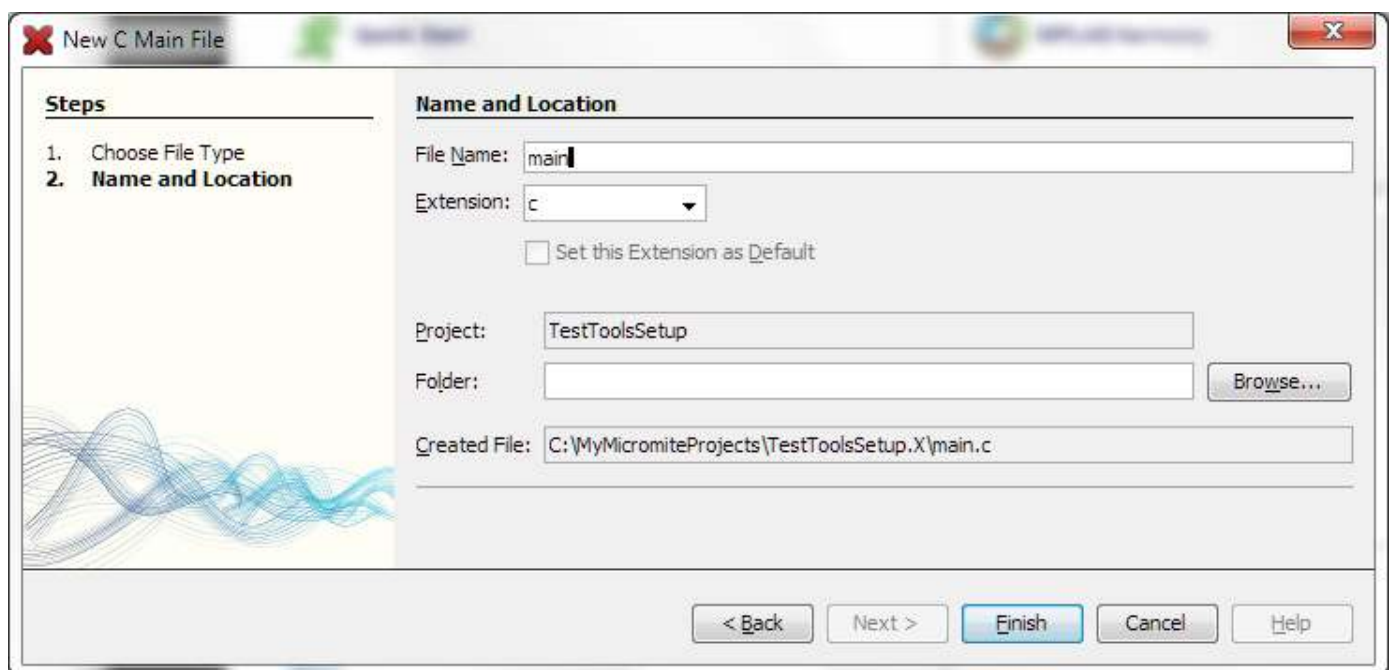
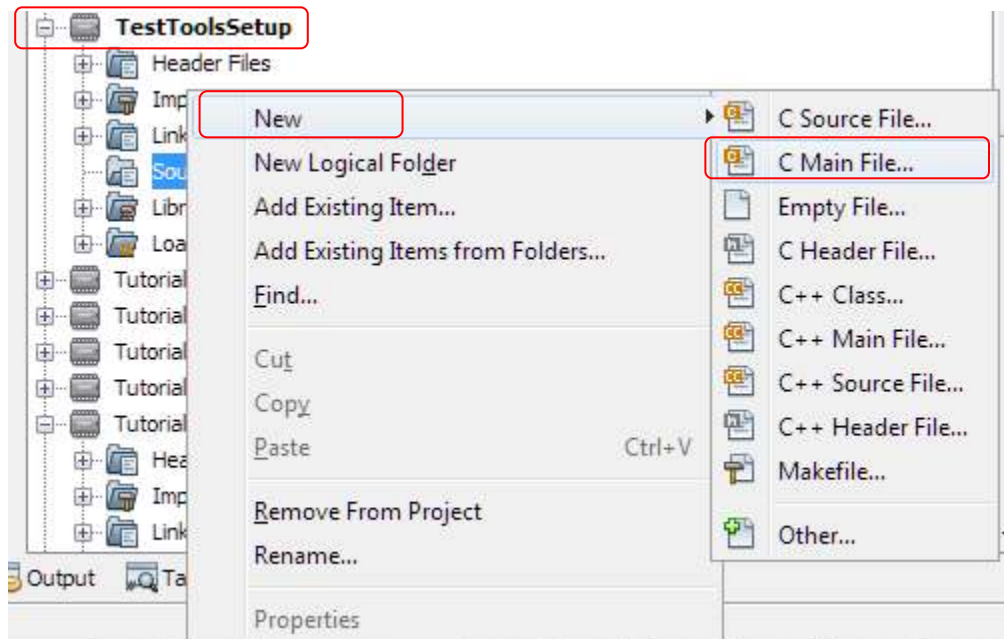
Enter Project Name TestToolsSetup, and set the Folder where you want to keep your Micromite projects. Check Set as main project.

Click Finish.

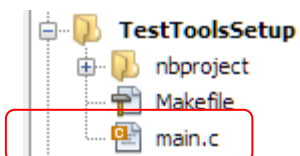
MPLab/X will now create a new project named TestToolsSetup



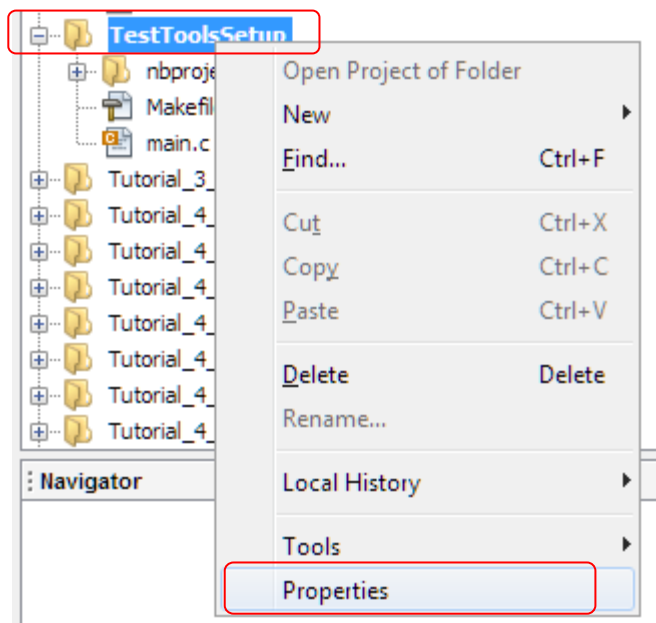
Right Click on TestToolsSetup and add a new C Main File



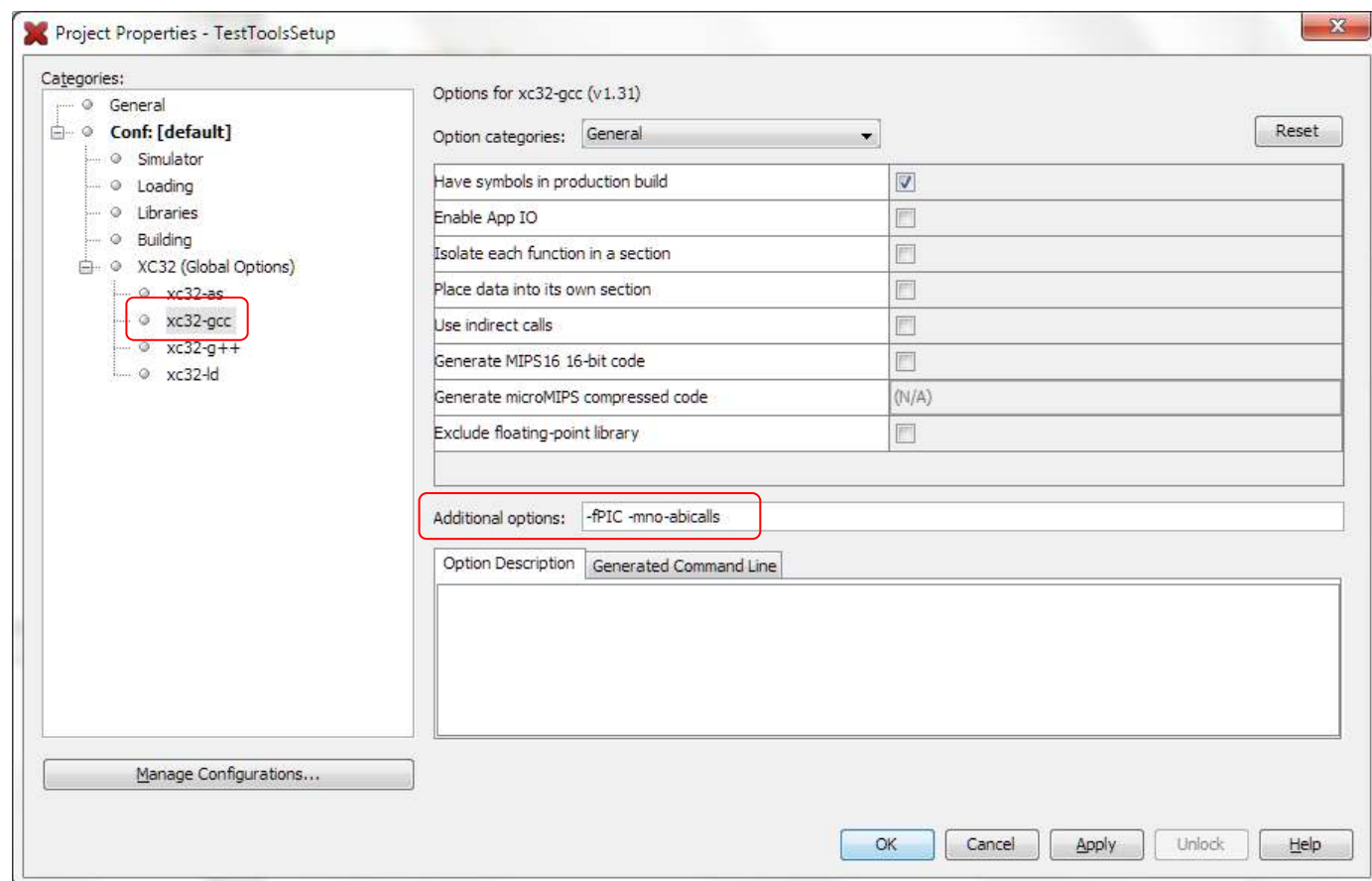
Change File Name to main.c and click Finish. MPLab/X will now generate main.c



Right Click TestToolsSetup.



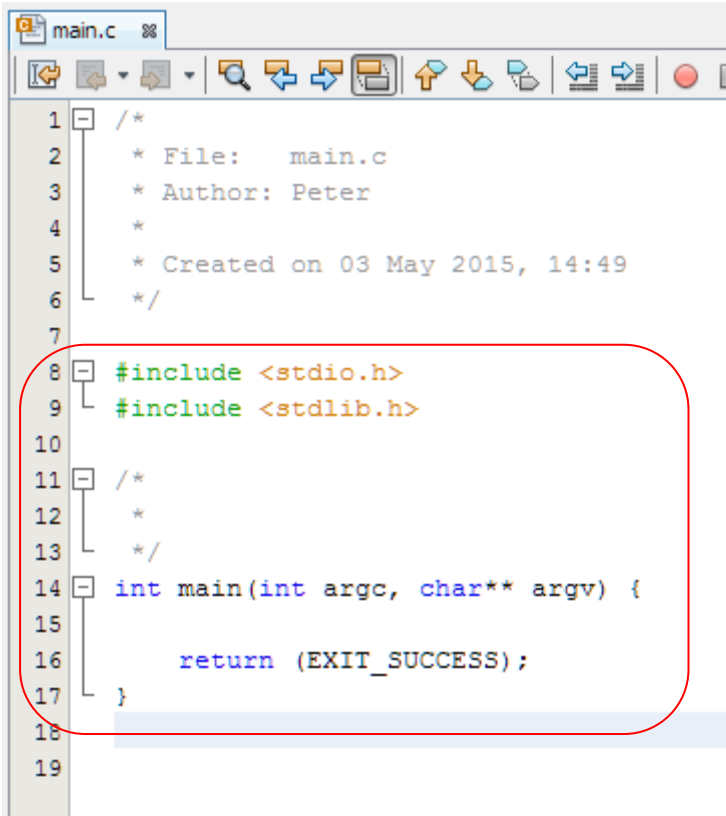
Select Properties



Select XC32-gcc in Categories list, and enter -fPIC -mno-abicalls in the Additional options box.

Click Apply, Click OK

Open main.c for editing, delete whatever MPLab/X has given as a default.



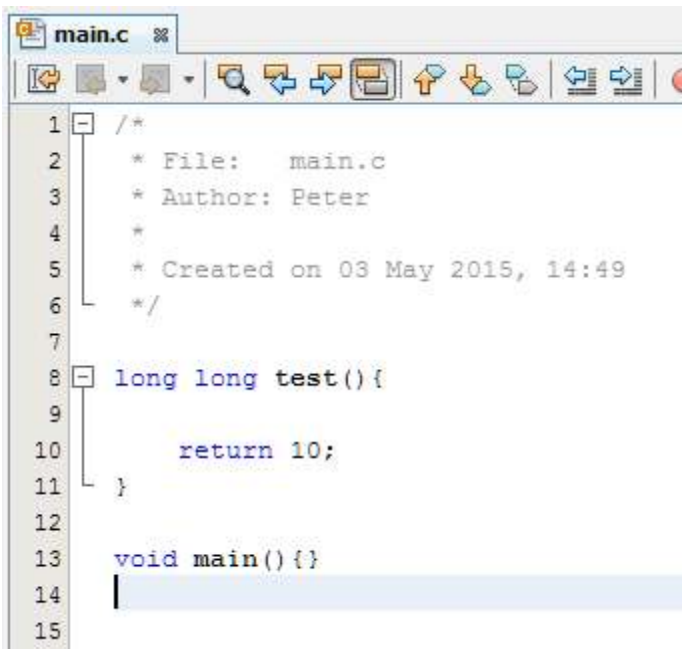
```
1  /*
2   * File:   main.c
3   * Author: Peter
4   *
5   * Created on 03 May 2015, 14:49
6   */
7
8  #include <stdio.h>
9  #include <stdlib.h>
10
11 /*
12  *
13  */
14 int main(int argc, char** argv) {
15
16     return (EXIT_SUCCESS);
17 }
18
19
```

and paste in the following code

```
long long test(){
    return 10;
}

void main(){}

```

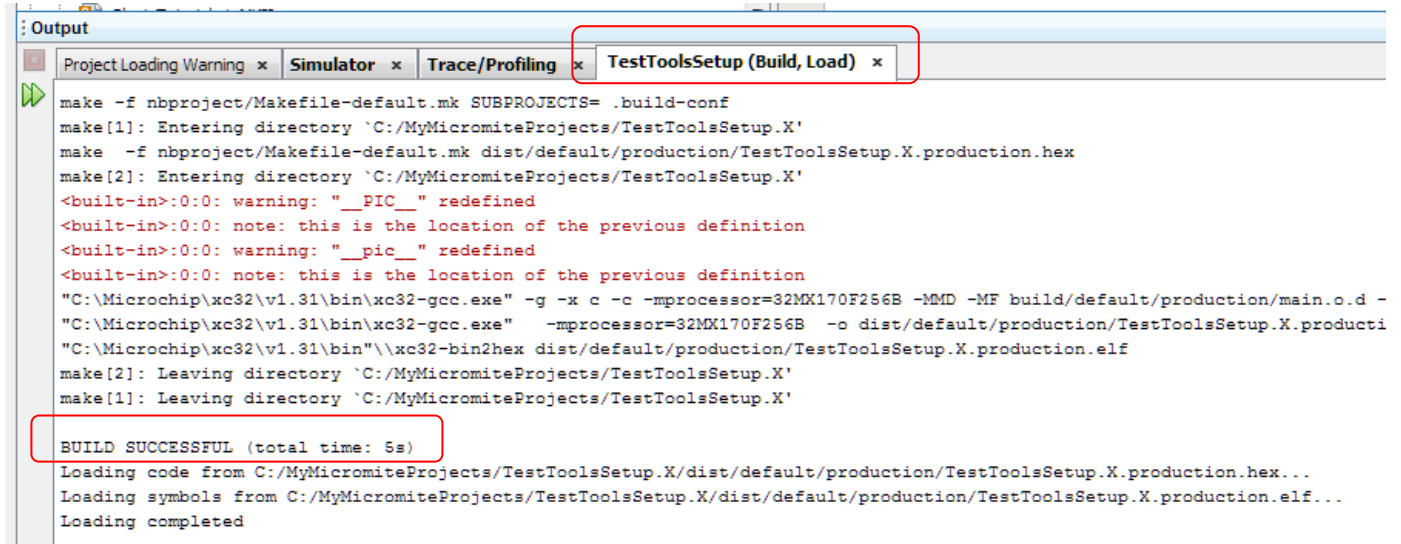


```
1  /*
2   * File:   main.c
3   * Author: Peter
4   *
5   * Created on 03 May 2015, 14:49
6   */
7
8  long long test(){
9
10     return 10;
11 }
12
13 void main(){}
14
15
```

Click Build



MPLab/X will then build your project, and if all the tools are correctly installed and configured you MPLab/X will display



```
Output
ProjectLoading Warning x Simulator x Trace/Profiling x TestToolsSetup (Build, Load) x
make -f nbproject/Makefile-default.mk SUBPROJECTS= .build-conf
make[1]: Entering directory `C:/MyMicromiteProjects/TestToolsSetup.X'
make -f nbproject/Makefile-default.mk dist/default/production/TestToolsSetup.X.production.hex
make[2]: Entering directory `C:/MyMicromiteProjects/TestToolsSetup.X'
<built-in>:0:0: warning: " __PIC__ " redefined
<built-in>:0:0: note: this is the location of the previous definition
<built-in>:0:0: warning: " __pic__ " redefined
<built-in>:0:0: note: this is the location of the previous definition
"C:\Microchip\xc32\v1.31\bin\xc32-gcc.exe" -g -x c -c -mprocessor=32MX170F256B -MMD -MF build/default/production/main.o.d -
"C:\Microchip\xc32\v1.31\bin\xc32-gcc.exe" -mprocessor=32MX170F256B -o dist/default/production/TestToolsSetup.X.producti
"C:\Microchip\xc32\v1.31\bin\xc32-bin2hex dist/default/production/TestToolsSetup.X.production.elf
make[2]: Leaving directory `C:/MyMicromiteProjects/TestToolsSetup.X'
make[1]: Leaving directory `C:/MyMicromiteProjects/TestToolsSetup.X'

BUILD SUCCESSFUL (total time: 5s)
Loading code from C:/MyMicromiteProjects/TestToolsSetup.X/dist/default/production/TestToolsSetup.X.production.hex...
Loading symbols from C:/MyMicromiteProjects/TestToolsSetup.X/dist/default/production/TestToolsSetup.X.production.elf...
Loading completed
```

This concludes checking that your MPLab/X and XC32 installation is correct and ready for use. In the unlikely event that you get to this last step, but the build fails, you will need to go back carefully over each step of this checkout and verify that your setup matches that described in this document.

Until you can successfully complete this checkout, you will not be able to develop CFunctions (or indeed any other kind of C program on your installation of MPLab/X and XC32).

MMBasic CFunction Generator

Next you will need to download and install the CFuncGen utility - <http://www.cfuncgen.dyndns.org/>

Text Editor

A text editor is not essential because whenever CFuncGen generates the CFunction file(s), it also puts the same text onto the Windows clipboard ready for you to paste, Ctrl+V, into your MMBasic program.

But, you may want to edit the CFunction text before you copy/paste it into your MMBasic program in which case you should use a text editor which will automatically watch for file changes - so that you can be sure that you always have the latest version of the CFunction(s) file in your text editor.

I like Notepad++ available from <http://notepad-plus-plus.org/>, but you can use whatever editor you like.

MMBasic Editor/IDE

You can use just about any text editor and terminal emulator to write/test/debug your Micromite MMBasic programs, but I strongly recommend MMEdit available from <http://www.c-com.com.au/MMedit.htm>.

C-Pre-processor

In Tutorial 4 we describe how using a C pre-processor in conjunction with MMEdit can really improve your CFunction development productivity. The C-Pre-processor package is available from <http://www.g8jcf.dyndns.org/mmbasic>