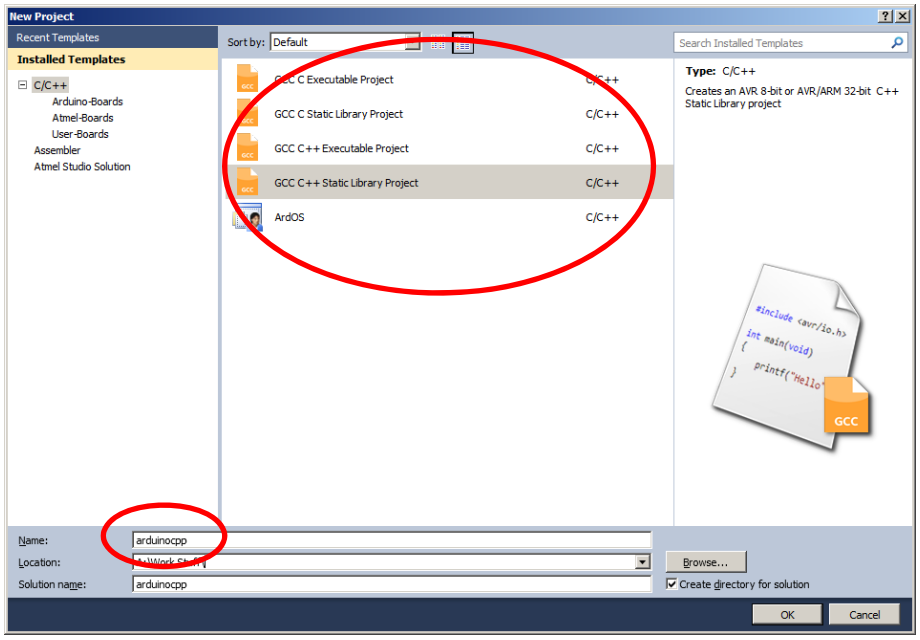
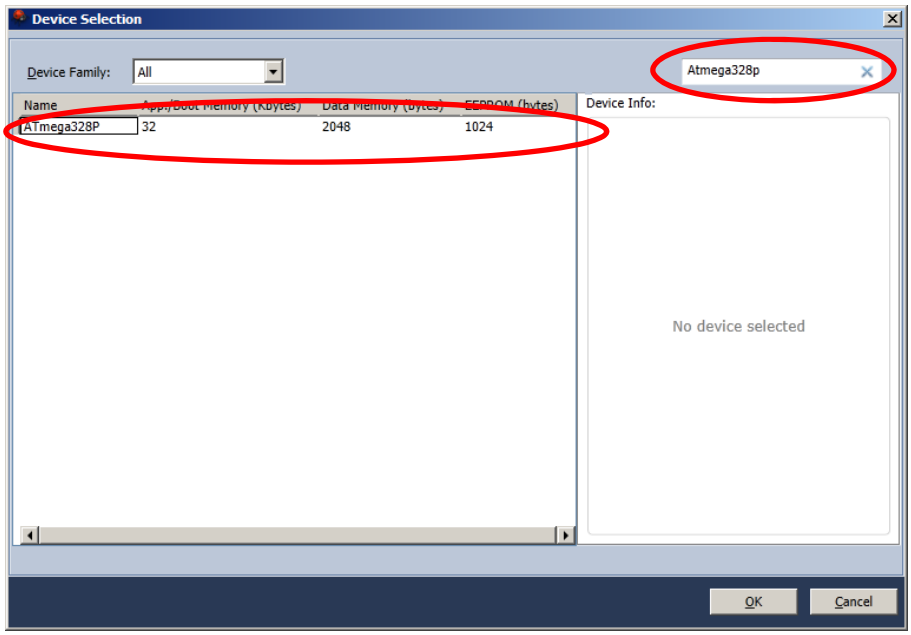
Part 1. <u>Instructions for Compiling Arduino Libraries under AVR/GNU C++</u>

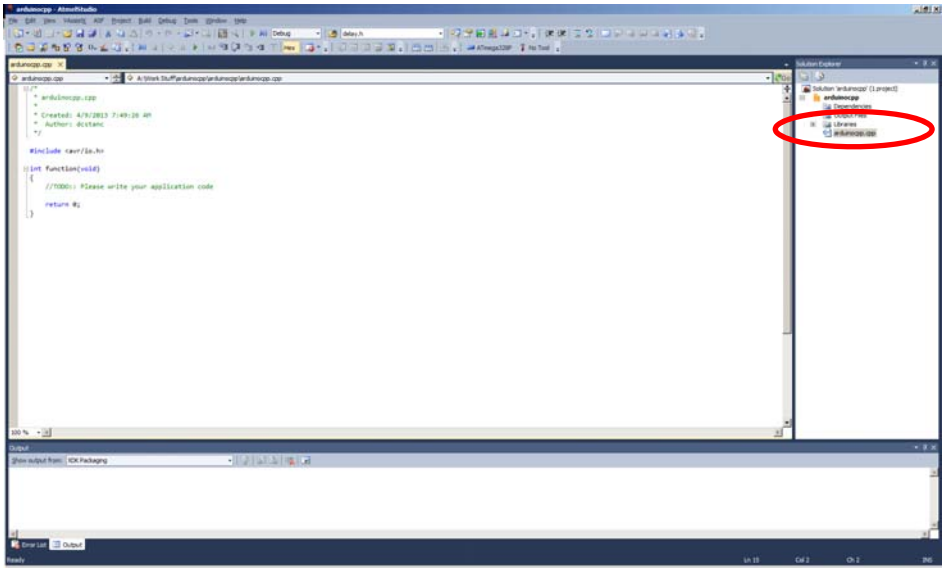
1. Create a new project by File->New Project.
2. Choose “GCC C++ Static Library Project” from the middle panel, and enter “arduinocpp” in the Name field at the bottom.



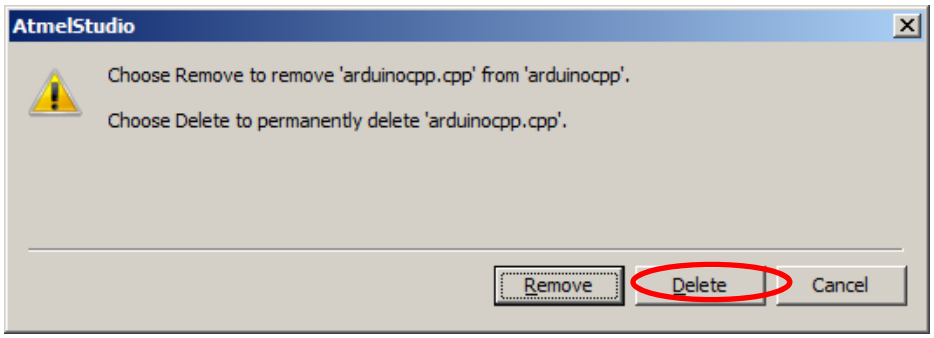
3. Click OK. This will bring up a Device Selection panel. Enter ATMega328P in the search box on the top right of the screen. Click on the Atmega328P in the bottom left panel and click OK.



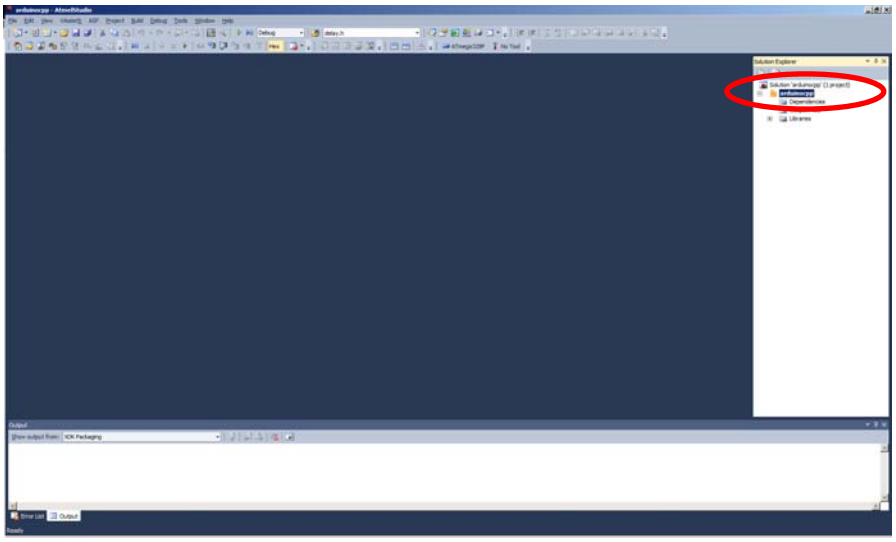
4. You will now be in the editing screen. Delete arduinocpp.cpp in the right panel by right clicking on the filename, and choosing “Remove”.



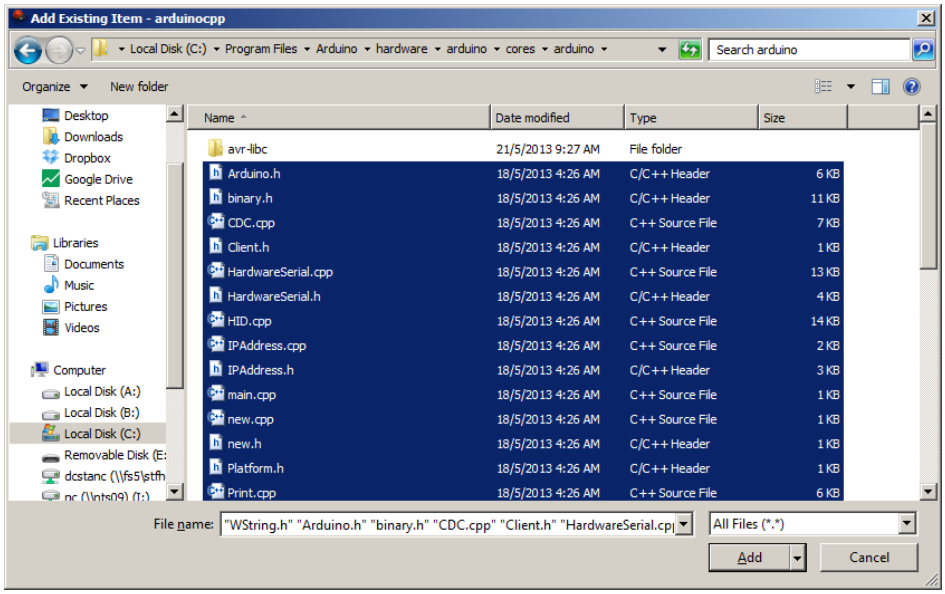
This will bring up a small dialog box that lets you remove or delete the file. Click on delete.



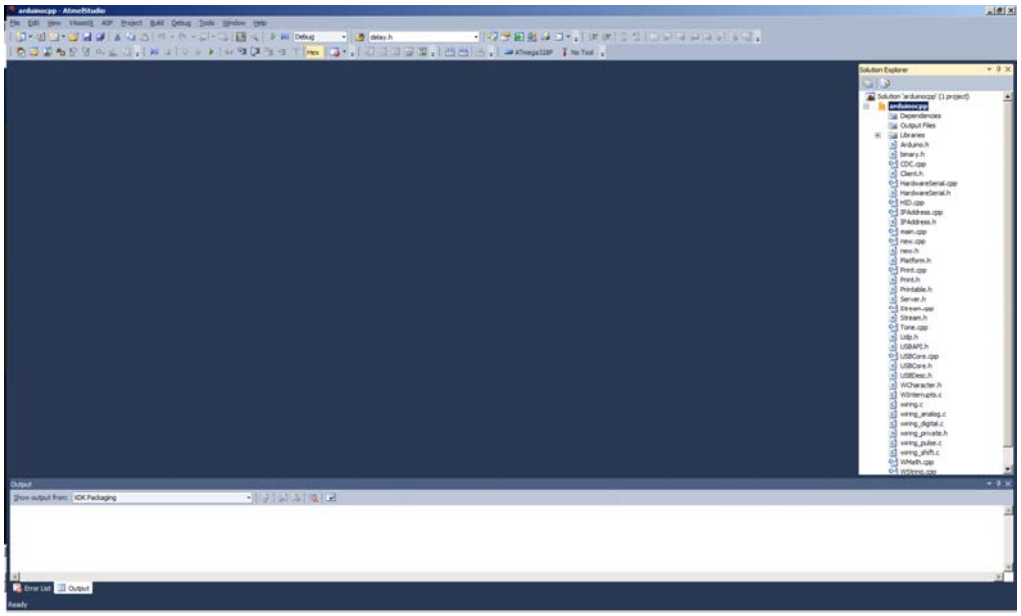
5. Now right click on the line that says “arduinocpp” with the yellow icon next to it. Choose Add->Existing Item.



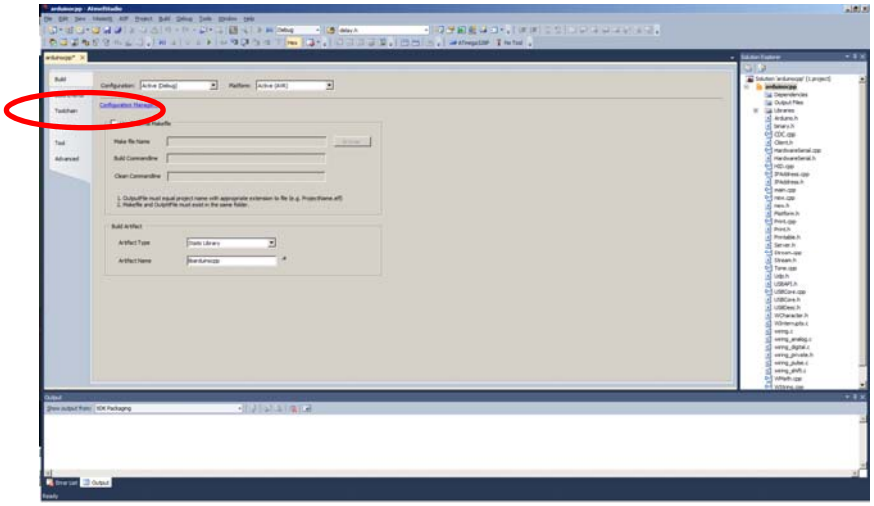
6. This will bring up a dialog box like so. Navigate to the directory that contains the Arduino library source code. In my case it is at “c:\program files\Arduino\hardware\arduino\cores\arduino”. Select every file in that directory except avr-libc. Click add.



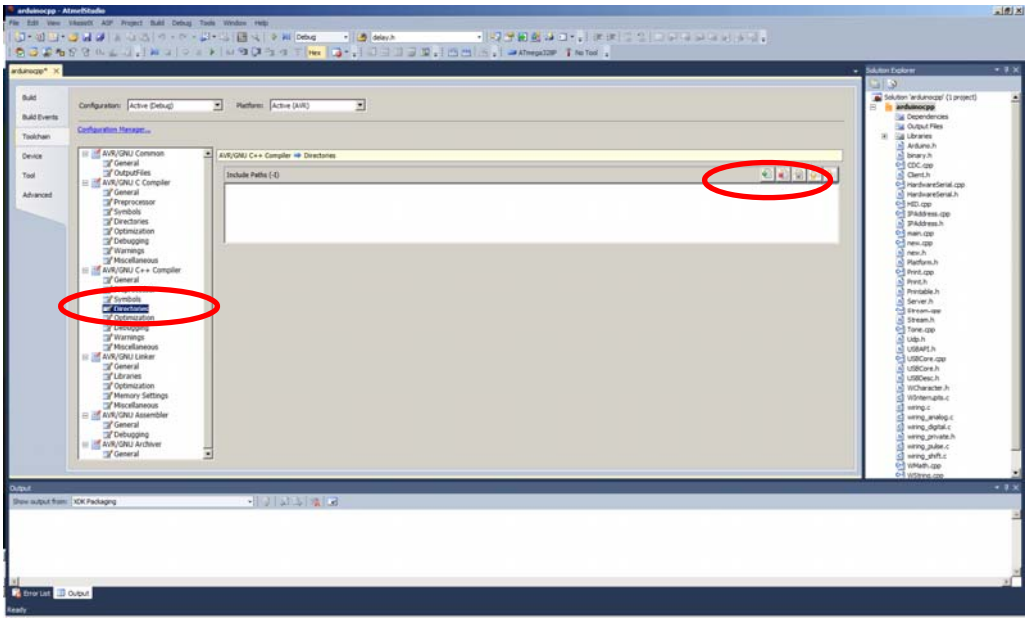
7. You will now be brought back to the edit screen with a long list of files on the right.



8. You now need to set up the include directories. To do so click Project->Properties. This will bring up this dialog box:

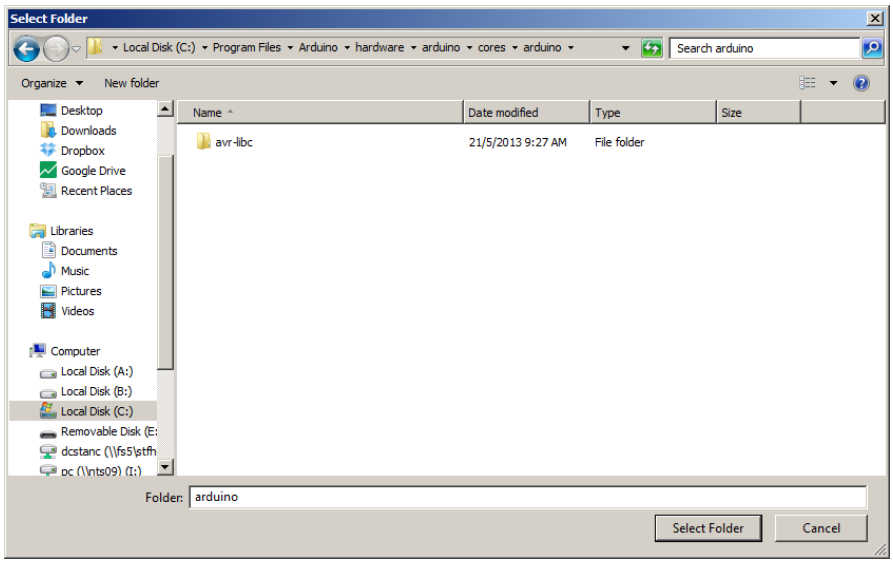


9. Click “Toolchains” on the tab on the left (circled above). This will bring up the toolchains setup screen. Click on “Directories” under “AVR/GNU C++ Compiler”.



Click on the green + button to bring up this dialog box:

10. Click on the … button and navigate to the source code directory for the arduino libraries. In my case it is “C:\Program Files\Arduino\hardware\arduino\cores\arduino”.

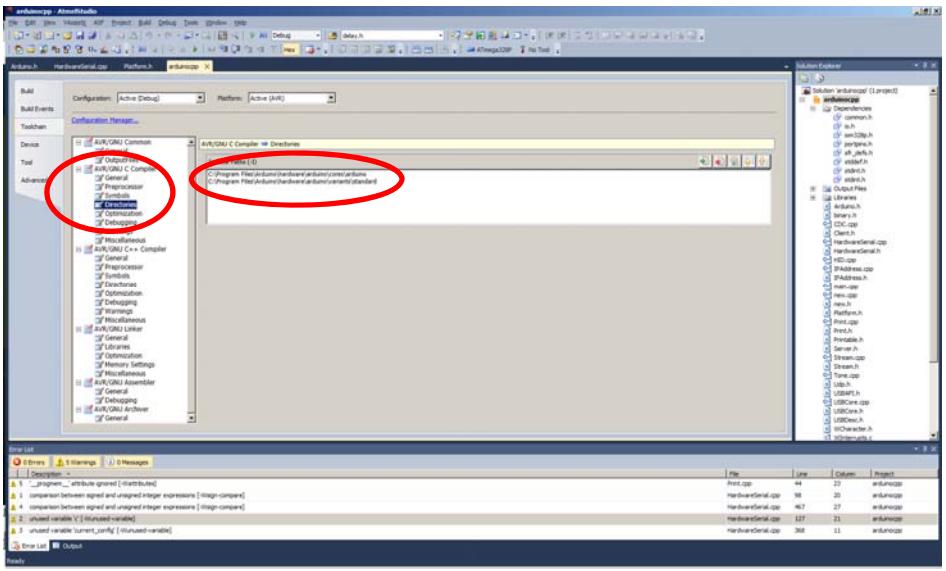


Click on Select Folder, and then OK at the small “Add Include Paths (-I)” dialog box in step 9.

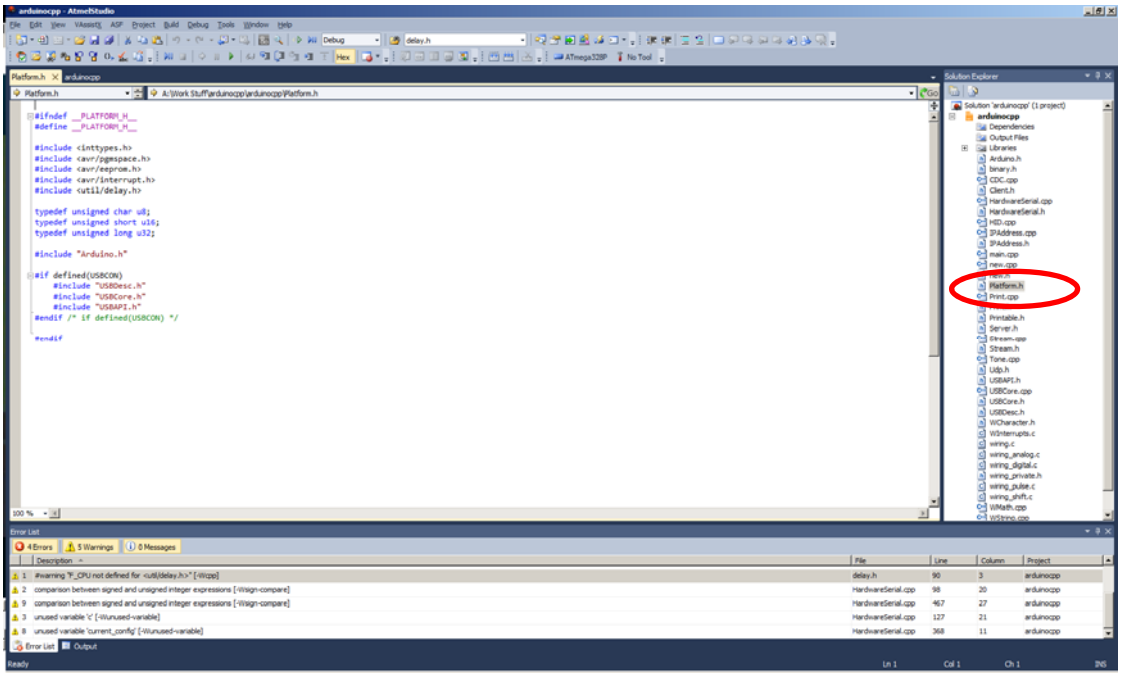
11. When you are back at the tool chain configuration screen, click on the green “+” and on “…” again, and this time navigate to the “variants\standard” directory. In my case it is at “C:\Program Files\Arduino\hardware\arduino\variants\standard”. Click “Select Folder” and “OK” to add this directory.

12. Repeat and add the same two directories to the “AVR/GNU C Compiler”. You can cut and paste the directory names from the “AVR/GNU C++ Compiler” side.

    (Yes, you actually have to add the directories to both the C and C++ compilers. This was my oversight. My apologies).



13. Now locate the file called “Platform.h” on the right panel. Double click on it to open it.



14. Locate the line that says “#include <util/delay.h>” and insert the following line just before it:

    #define F_CPU 16000000UL

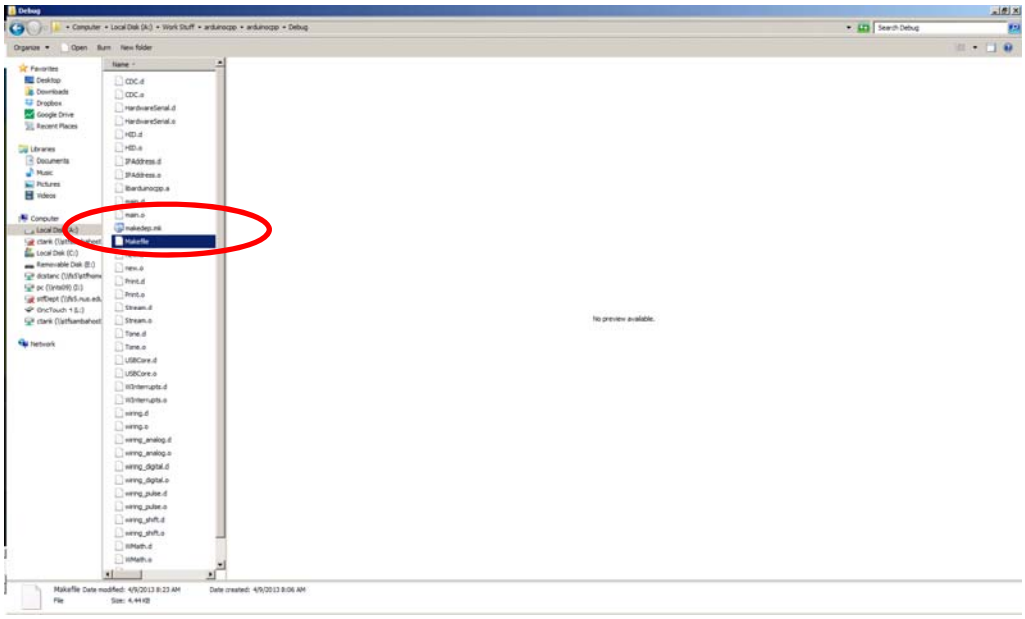
    Your Platform.h should now look like this:

```
#ifndef __PLATFORM_H__
#define __PLATFORM_H__

#include <inttypes.h>
#include <avr/pgmspace.h>
#include <avr/eeprom.h>
#include <avr/interrupt.h>
#define F_CPU 16000000UL
#include <util/delay.h>

typedef unsigned char u8;
typedef unsigned short u16;
typedef unsigned long u32;

#include "Arduino.h"

#if defined(USBCON)
	#include "USBDesc.h"
	#include "USBCore.h"
	#include "USBAPI.h"
#endif /* if defined(USBCON) */

#endif
```

15. Locate the file called “Arduino.h” and double click on the filename. Add in again “#define F_CPU 16000000UL” just before the line “#define HIGH 0x1”. Your Arduino.h should look like this:

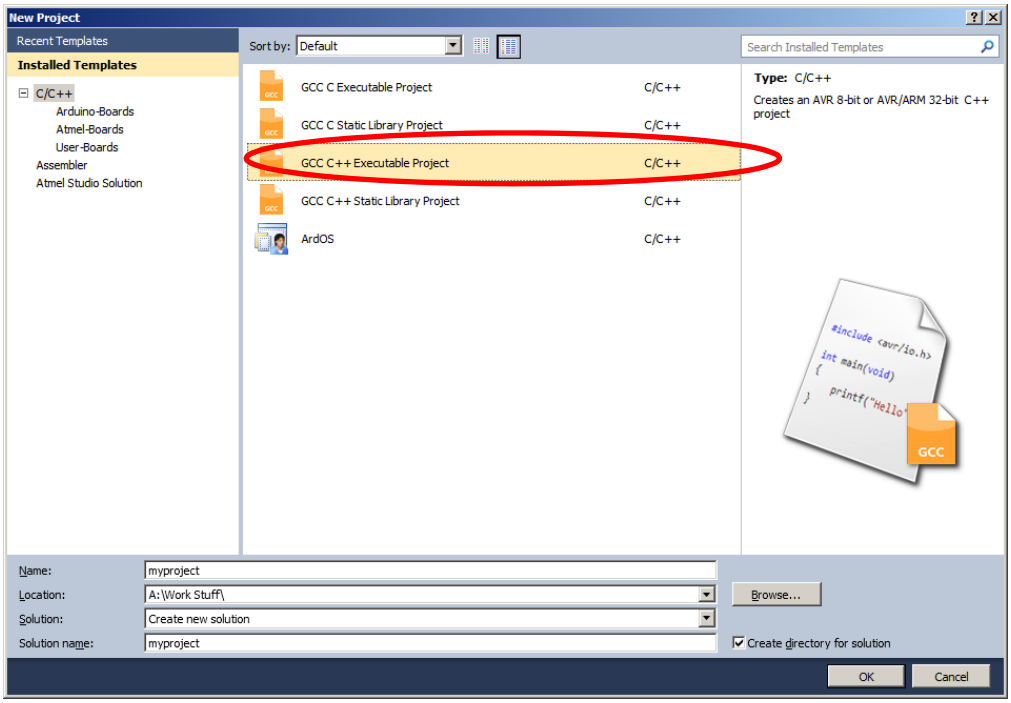
```
…
#include <avr/pgmspace.h>
#include <avr/io.h>
#include <avr/interrupt.h>

#include "binary.h"

#ifdef __cplusplus
extern "C"{
#endif

#define F_CPU 16000000UL

#define HIGH 0x1
#define LOW  0x0
…
```

16. Click Build->Build Solution to compile the library. You will get some warnings but the compile will succeed. To verify, navigate to the debug directory of your library project. In my case it is at “a:\work stuff\arduinocpp\arduinocpp\debug”. You should see a file called “libarduino.a” in that directory.



Part 2. <u>Compiling an Arduino Application</u>

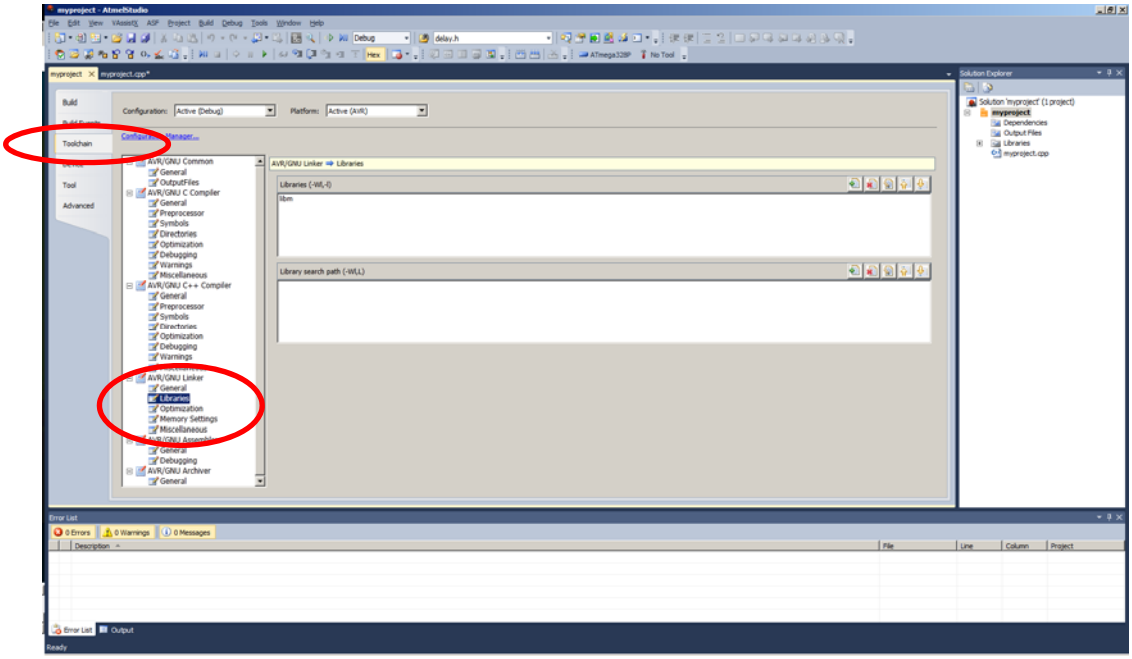
1. Create a new project by clicking File->New Project. Select “GCC C++ Executable Project” from the center panel, and name your project in the “Name” field. Click OK.



2. As before select the Atmega328P.
3. Edit the project’s .cpp file to implement your project. In this case I used the following “blinky” source code.

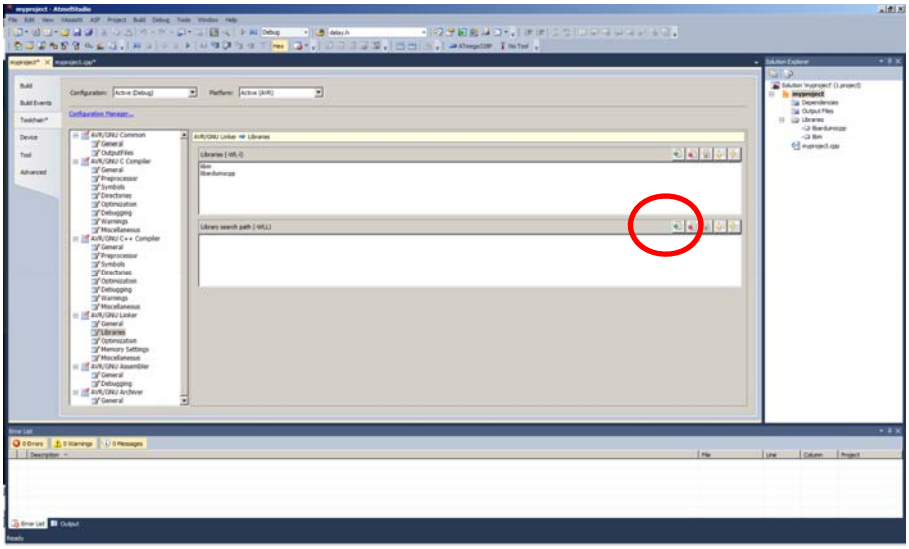
```
#include <avr/io.h>
#include <Arduino.h>

void setup()
{
	pinMode(13, OUTPUT);
}
void loop()
{
	digitalWrite(13, HIGH);
	delay(500);
	digitalWrite(13, LOW);
	delay(500);
}
int main(void)
{
    init();
    setup();
    while(1)
    {
		loop();
		if(serialEventRun)
			serialEventRun();
    }
}
```

4. Set up the include directories for the “AVR/GNU C++” and “AVR/GNU C” compilers as before.
5. Set up the “Library” directory and name. To do this, click “Project->myProject Properties”, choose Toolchain from the left tab bar, and click “Libraries” under “AVR/GNU Linker”:

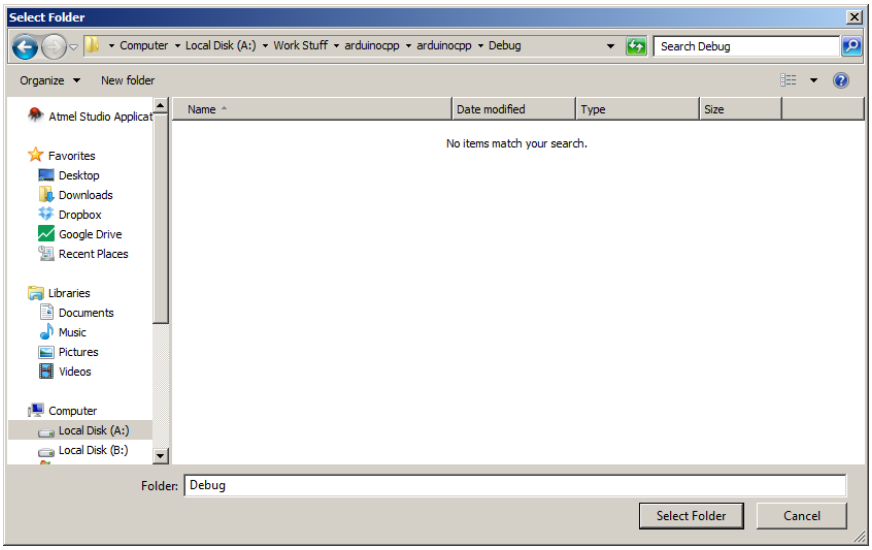


6. In the “Libraries (-Wl, -l)” field click on the green “+” button and enter “libarduinocpp” and click “OK” on the dialog box that pops out.

7. Now click the green “+” button at the “Library Search Path (-WI,L)” field.



8. This will produce a dialog box. Click on “…” in the dialog box and navigate to the directory that contains your “libarduinocpp.a” file. Click “Select Folder”, then “OK” in the “Add Library Search Path” dialog box.



9. Click “Build Solution”. Your program should compile fully and you will be able to upload the .hex file to the Arduino using avrdude.