

Building JSBSim under Visual C++ 2008 Express, as Executable and Library

By Bill Galbraith
billg (at) holycows dot net

October 29, 2010

Preface: Okay, I'm an idiot sometimes. I'm not that good in Visual C++ 2008 Express, so it took me a while, with the help of some other people, namely Csaba Halász, to get this down. I took baby steps, and show them here, in case you are like me. These steps were written on a 32-bit version of Windows XP. It may vary slightly with different versions of Windows.

Preinstalled:

- Visual C++ 2008 Express edition
- CVS

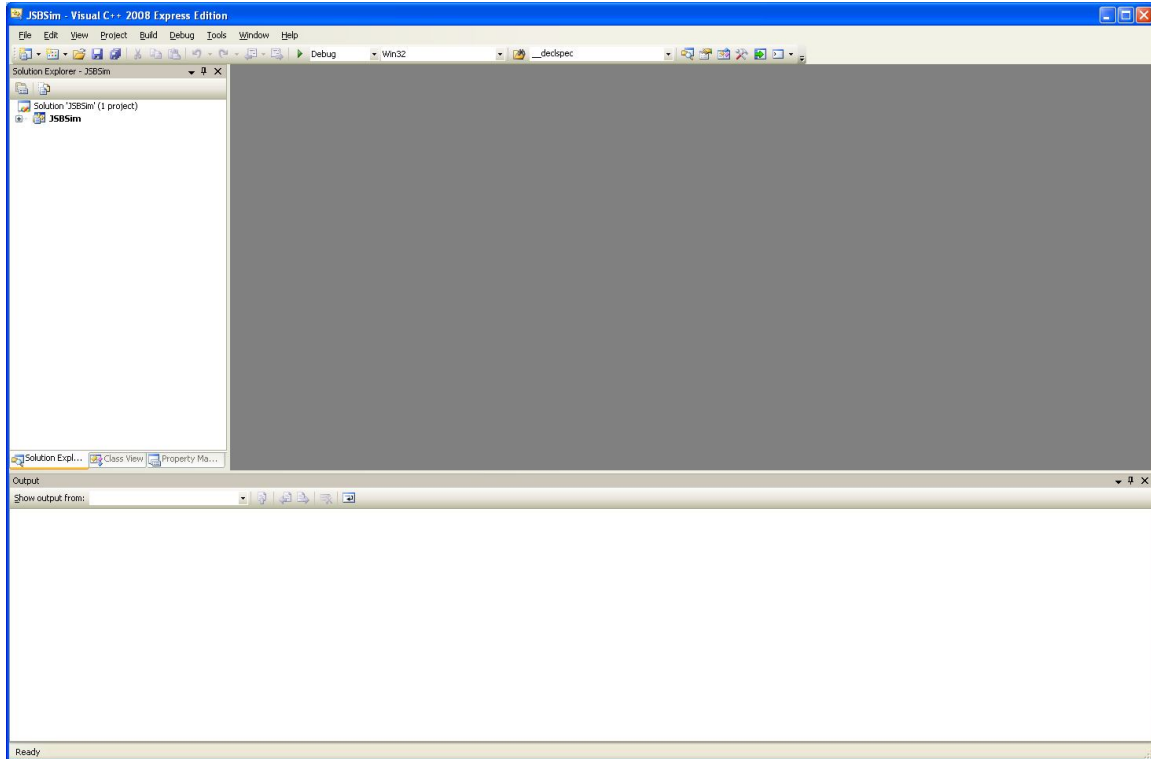
Building the JSBSim program

1. Download JSBSim from CVS, using the normal checkout method as described on the web page. At the time of this tutorial, here were the command that I used:

```
cvs -d:pserver:anonymous@jsbsim.cvs.sourceforge.net:/cvsroot/jsbsim login  
cvs -z3 -d:pserver:anonymous@jsbsim.cvs.sourceforge.net:/cvsroot/jsbsim co -P JSBSim
```

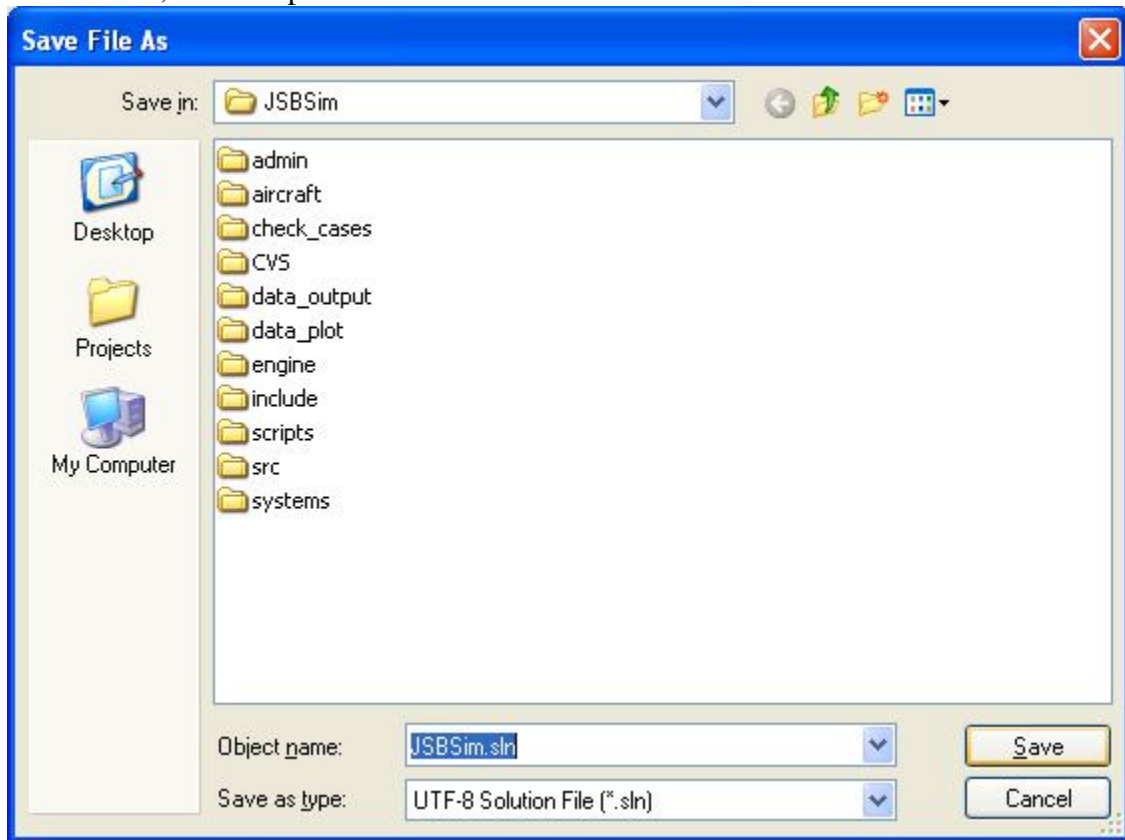
Note that I used CygWin to do this, as it has all of the tools such as CVS that I needed. You'll have to figure this out yourself. For the password asked after the first command, just press Enter.

- Open up the Visual C project file in the JSBSim directory, called JSBSim.vcproj.
This is what you should see, with the Solution Explorer on the left side being the only window showing any significant information.



3. We will build the executable program first, so just press F7, or pull down the **Build** menu and select either **Build Solution** or **Rebuild Solution**, the difference being that **Rebuild Solution** cleans out all the generated files from a previous build. This is our first build, so there is nothing to clean out.

When the build starts for the first time, Visual C++ will ask for a file name for the solution file, and will provide a default name.



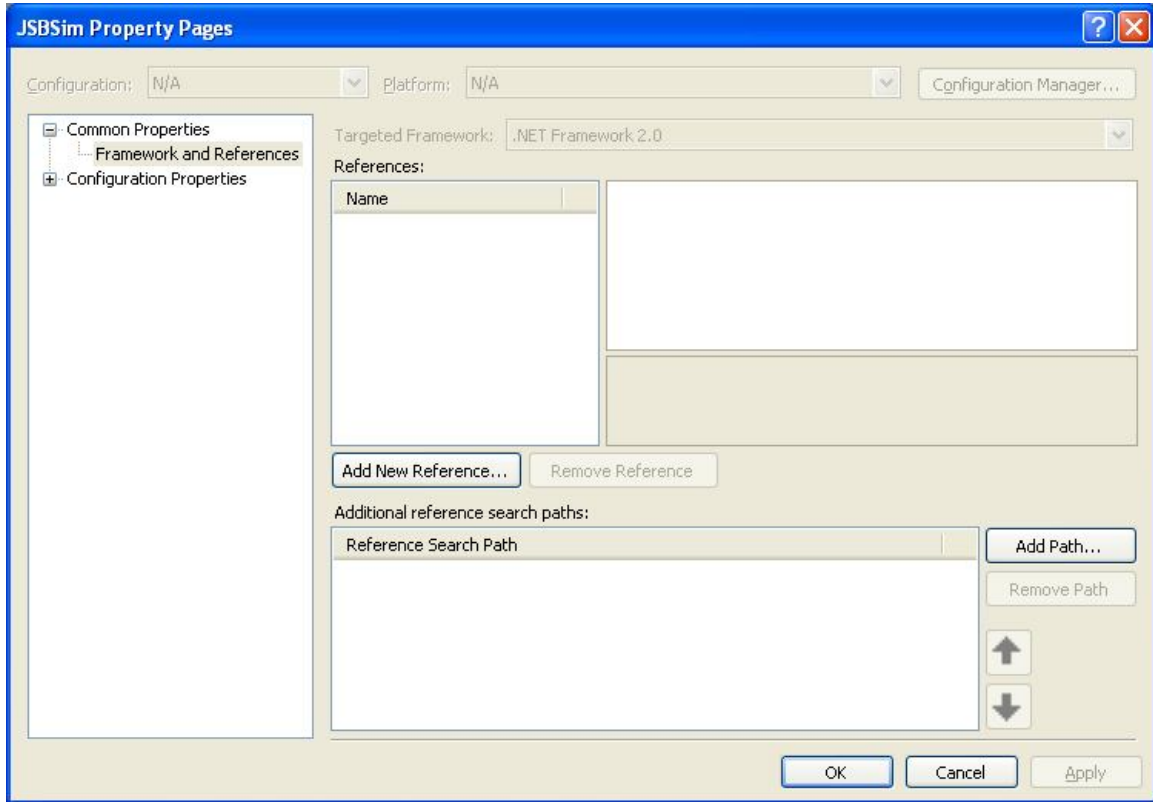
You can accept the default name. I like to prepend an underscore on the name, to make it **_JSBSim.sln**, just so that it is close to the top of the Explorer list of file, and therefore easier to find. Name it whatever you would like, or accept the default name.

With the source code that I used, there were 4 warning issues, but no errors.

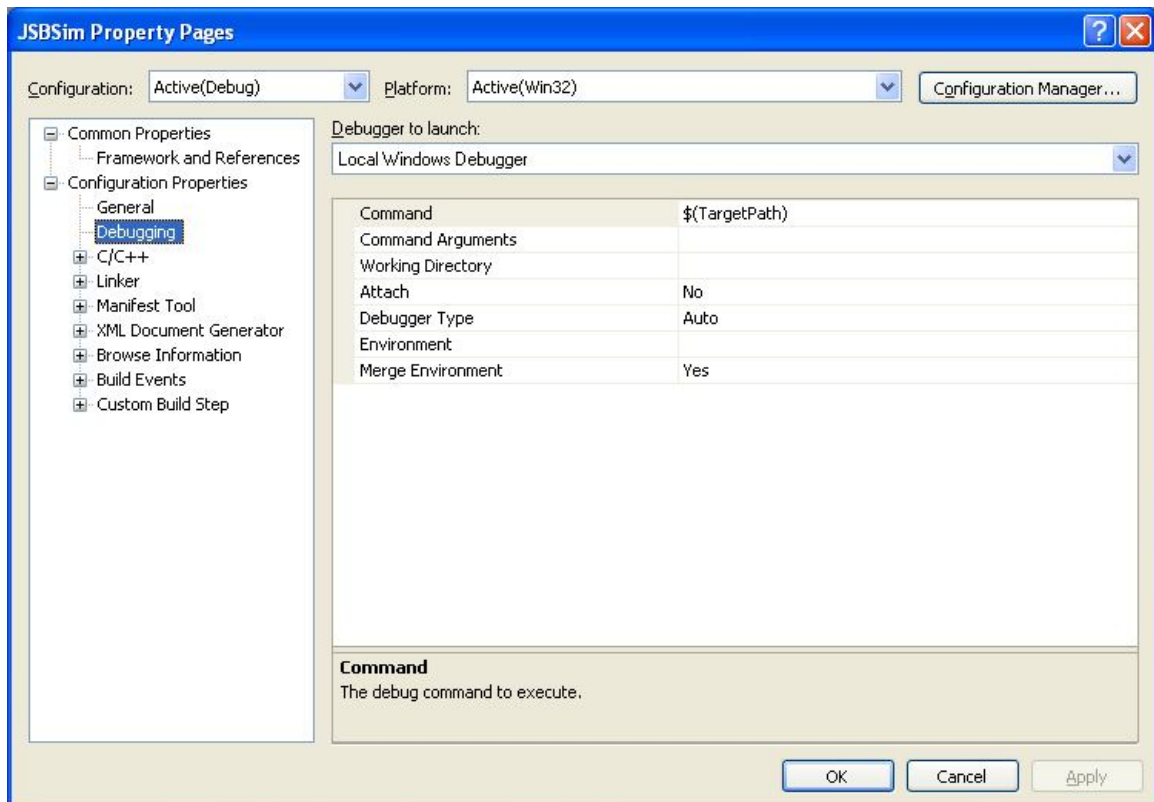
4. One way to run the program is directly from Visual C++, however, the program accepts a script file name as an argument, so we have to set up Visual C++ to pass this argument in when it is started. This method is fine if you want to always run JSBSim with the same script, or rarely change the script file. If you want to run JSBSim with different script files, skip to the next step and run it from the command line.

Open up the the Project Properties. This menu box is easily called up by pressing Alt-F7, or by pulling down the **Project** menu bar and selecting **Properties**, or by right

clicking the mouse on the JSBSim in the solution Explorer on the left side, and selecting **Properties**. However you do it, this is the menu box that pops up.



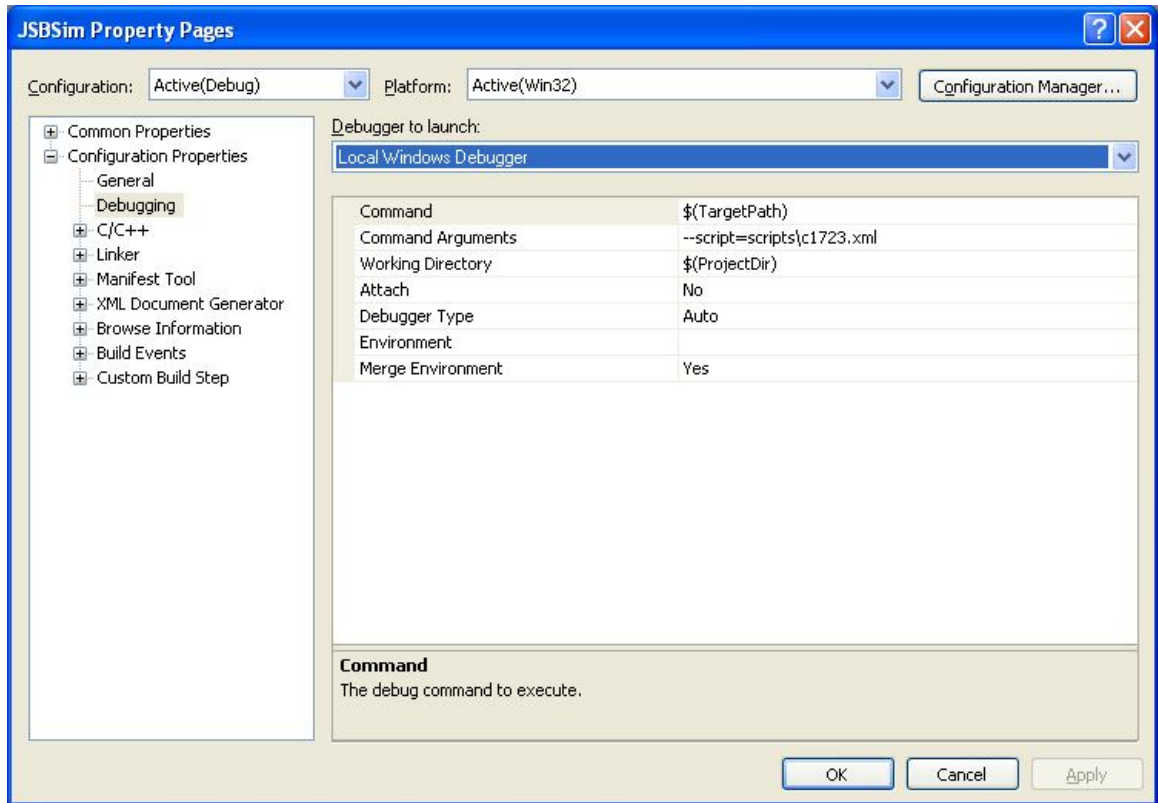
Press the “+” sign beside the **Configuration Properties** on the left side to expand the menu, then click on the **Debugging** line. You should see this:



When the program is run from Visual C++, the command is issued to run the JSBSim.exe program using the full path to the executable, which is all contained in the replaceable parameter **\$(TargetPath)**. By default, the program starts in the default Windows working directory, but we need to be in the project directory. You could hard code this path, but if your project ever got moved, it would be wrong, so the easier and better way to do it is to put in a replaceable parameter. On the right side, find the line that says **Working Directory**. Edit this line to say **\$(ProjectDir)**.

On the right side, find the line that says **Command Arguments**, and enter the command line argument **--script=scripts\c1723.xml**, which will execute that script when the program is started.

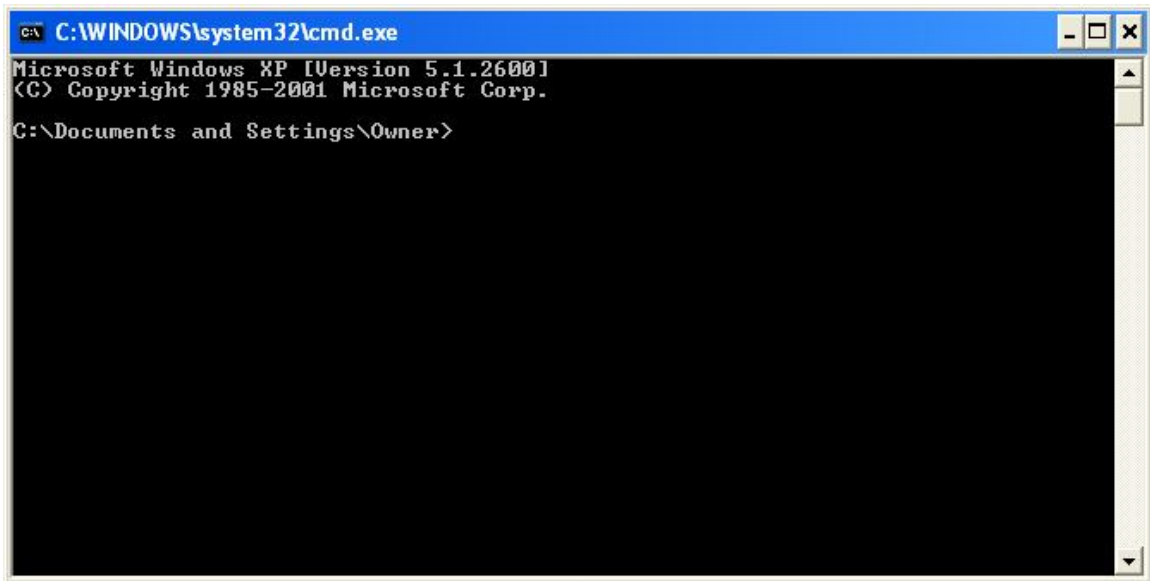
Your Property Page should now look like this:



You can now press **OK** to close the JSBSim Property Page.

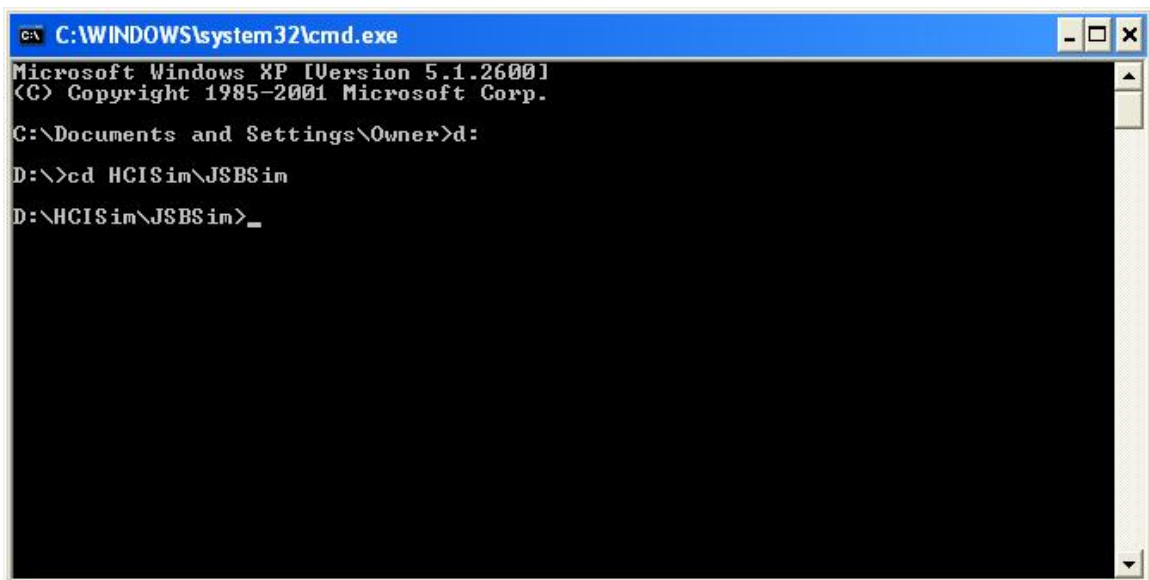
You can now run the program, either by pressing **Ctrl-F5**, or pulling the **Debug** menu and selecting **Start without Debugging**. A command window will pop up and execute the JSBSim program. It runs for a couple of minutes with this script, which starts the C-172 and performs a take-off. It generates a comma-separated values (.csv) file in the JSBSim directory called **JSBout172B.csv**. You will have to press **Enter** when it finishes, and the window is closed automatically.

- Another way to run the JSBSim program is from a command line. Visual C++ provides a command line prompt entry in the Windows Start menu under the Visual Studio tools, which provides a path to all of the Visual C++ tools, etc. However, we don't need this, so you can either open that up, or just open a command window by pressing the **Windows key** and **R**, or by pulling up the start menu and typing in **cmd** and pressing **Enter** or the **OK** button. That will cause a command window to pop up.



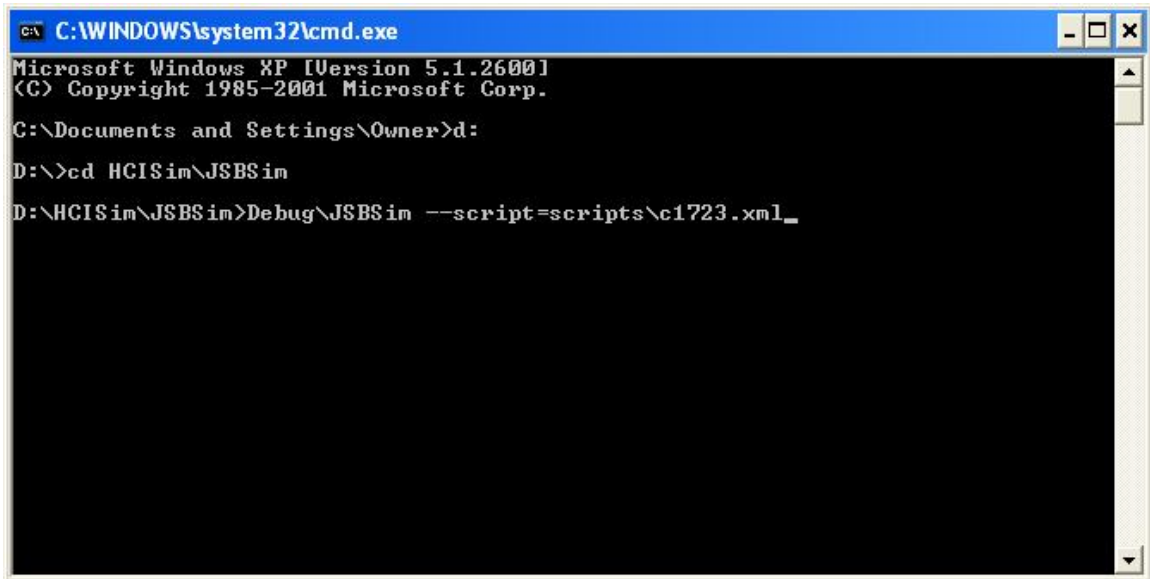
```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\Owner>
```

We need to go to the JSBSim directory. On my computer, with is on the D: drive, so I would type the commands shown:



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.
C:\Documents and Settings\Owner>d:
D:\>cd HCISim\JSBSim
D:\HCISim\JSBSim>_
```

We can now run the JSBSim program, which is located in the Debug directory. We need to pass the command line argument, so type in the command as shown:



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

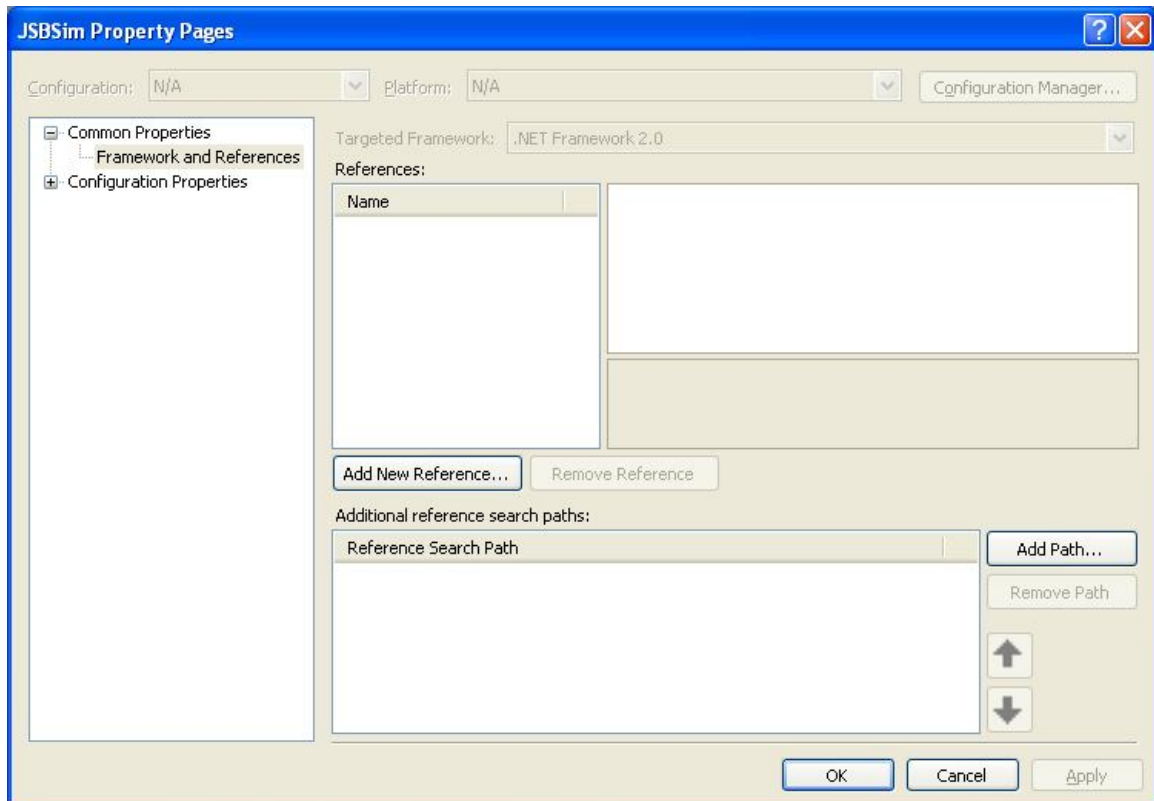
C:\Documents and Settings\Owner>d:
D:\>cd HCISim\JSBSim
D:\HCISim\JSBSim>Debug\JSBSim --script=scripts\c1723.xml_
```

The program will start, load the script and execute, and generate the same **JSBout172B.csv** file as if we ran it from Visual C++. However, if you want to run a different script, it is easier to do it from this command box, since you have to type it in. Either way produces the same results.

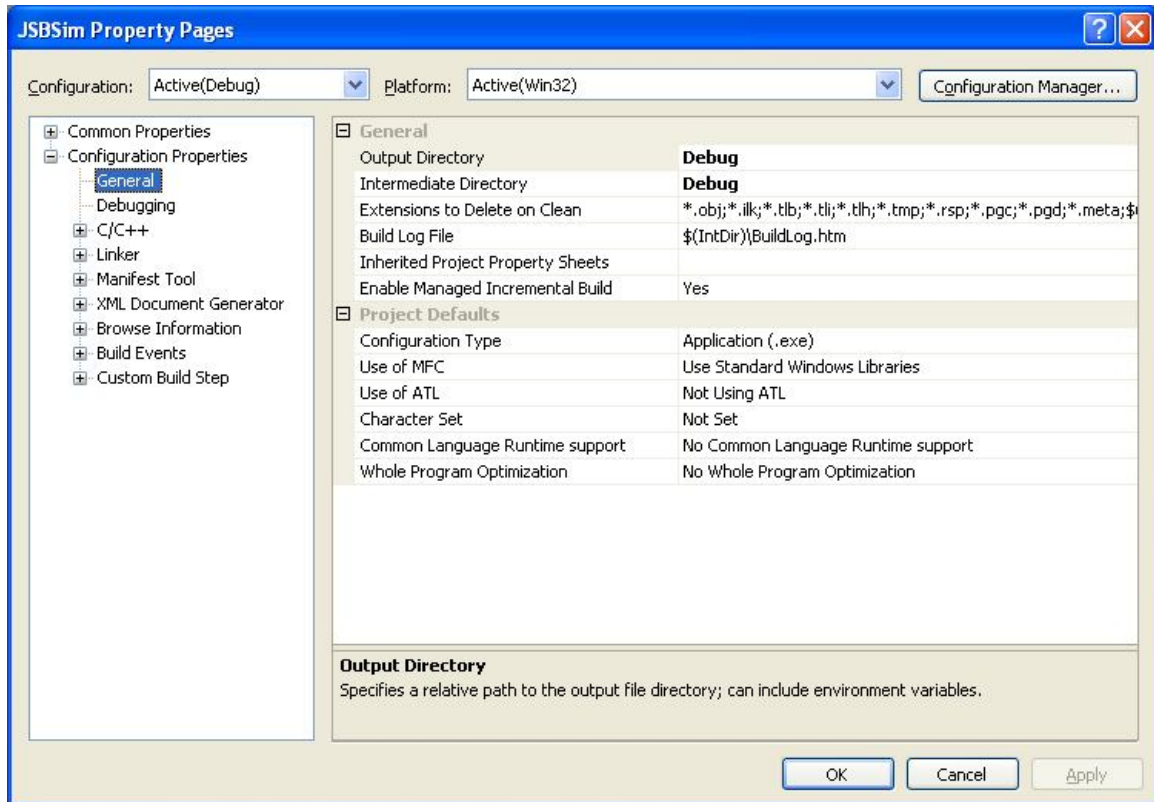
Building the JSBSim library

In this section, we will build the JSBSim into a static library, and using the same main program as above, use that code and the library to build an executable program. This shows the steps you would follow if you wanted to call JSBSim from your own program.

1. Our purpose here is to build a static library of JSBSim code. Therefore, we need to change the Project Properties. Called up the Project Properties box as before, by pressing Alt-F7, or by pulling down the **Project** menu bar and selecting **Properties**, This is the menu box that pops up:

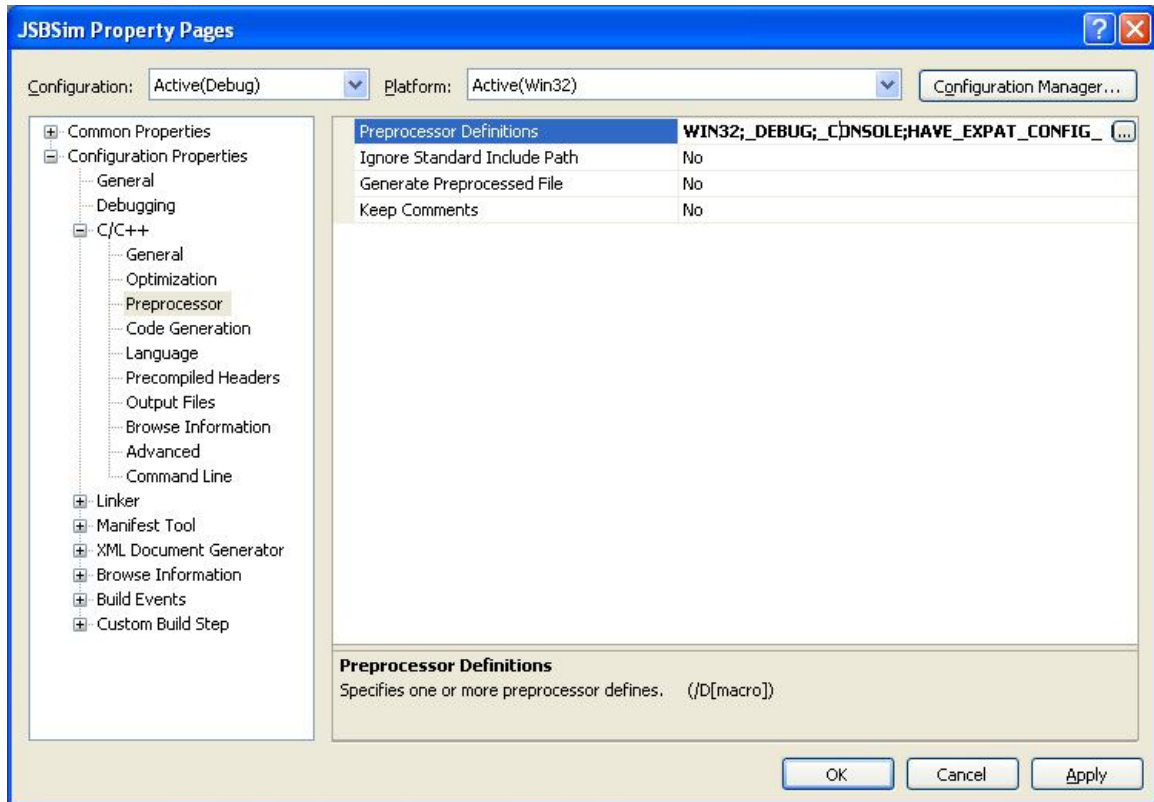


If the menu on the left side is not already expanded, press the “+” sign beside the **Configuration Properties** on the left side to expand the menu, and select the **General** entry from the left side. The box should look like this now:

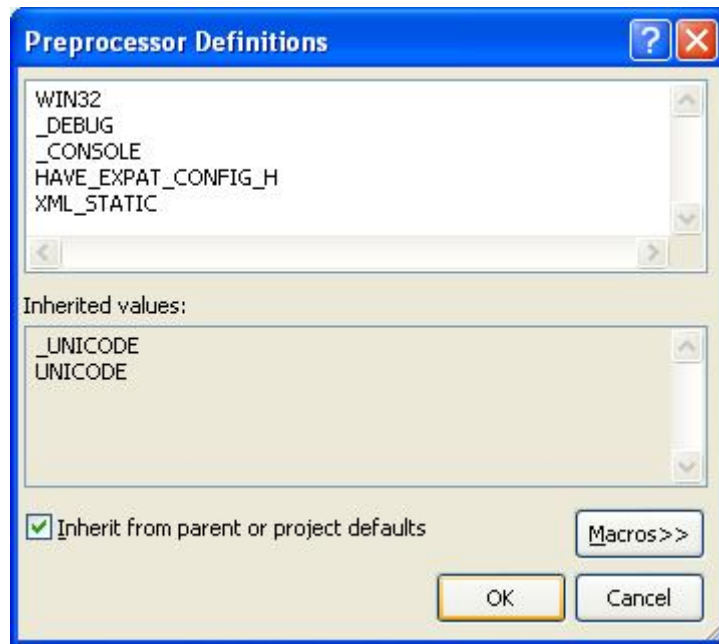


On the right side, under **Project Defaults**, change the line for **Configuration Type** from **Application (.exe)** to **Static Library (.lib)**, by clicking on that box and using the menu selection. On the line marked **Character Set**, if this says **Use Unicode Character Set**, change it to **Not Set**.

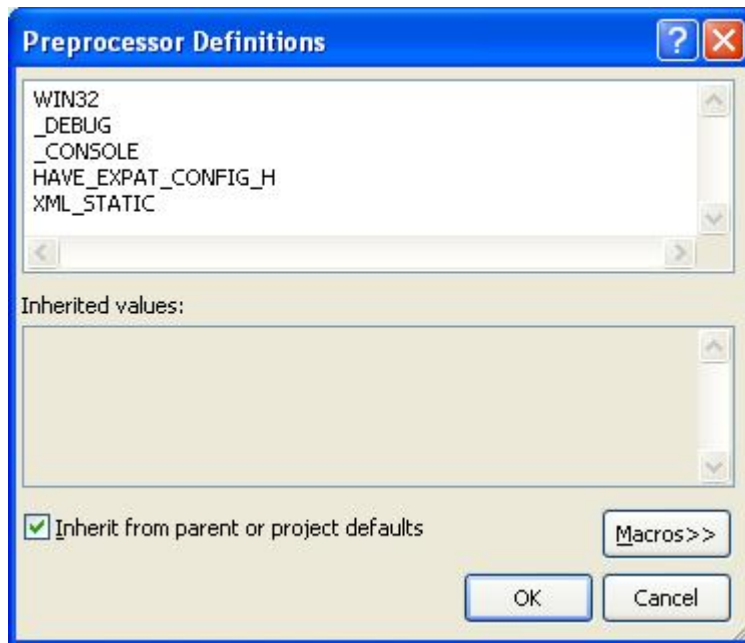
On the left side, expand the **C/C++** entry and select the **Preprocessor** entry, as shown:



Select the top line, **Preprocessor Definitions** line by clicking in the box that has the entries. The pull-down box with the ellipses (,,) will appear. Press that box, and the following screen will appear:

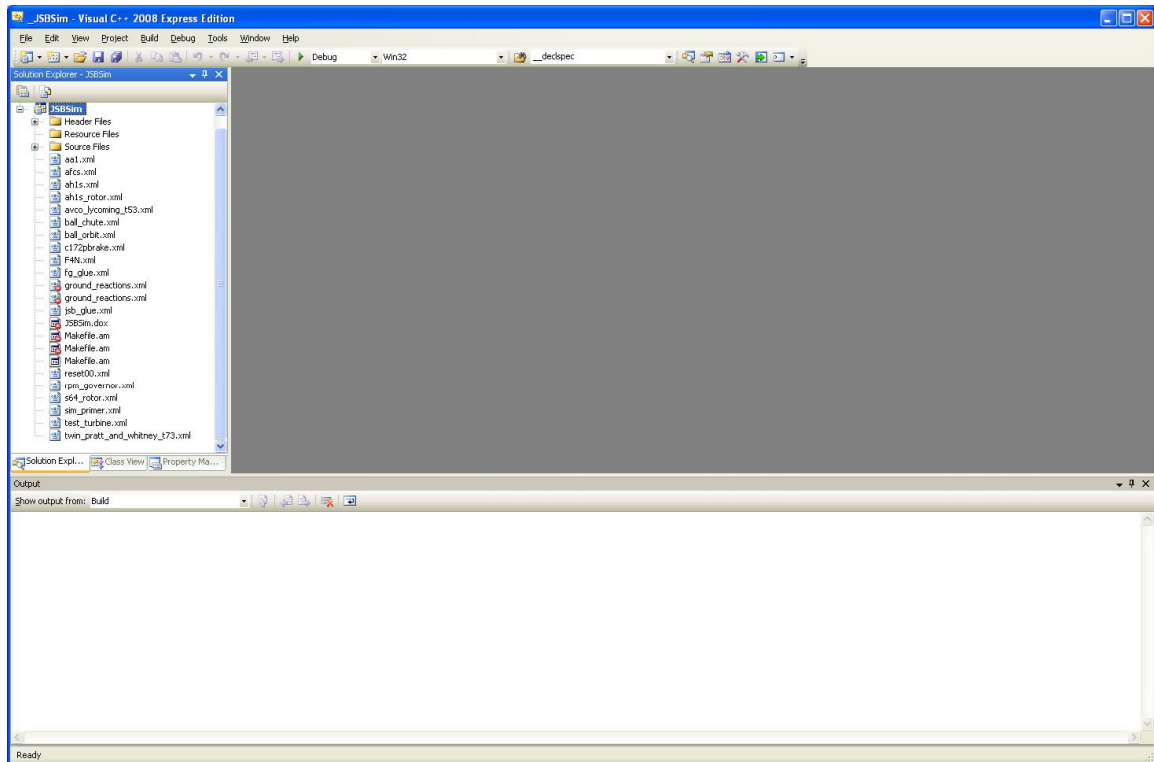


At the bottom, add the entry **XML_STATIC** (this has to be in all capital letters), as shown



Press the **OK** button twice to get back to the main Visual C++ window.

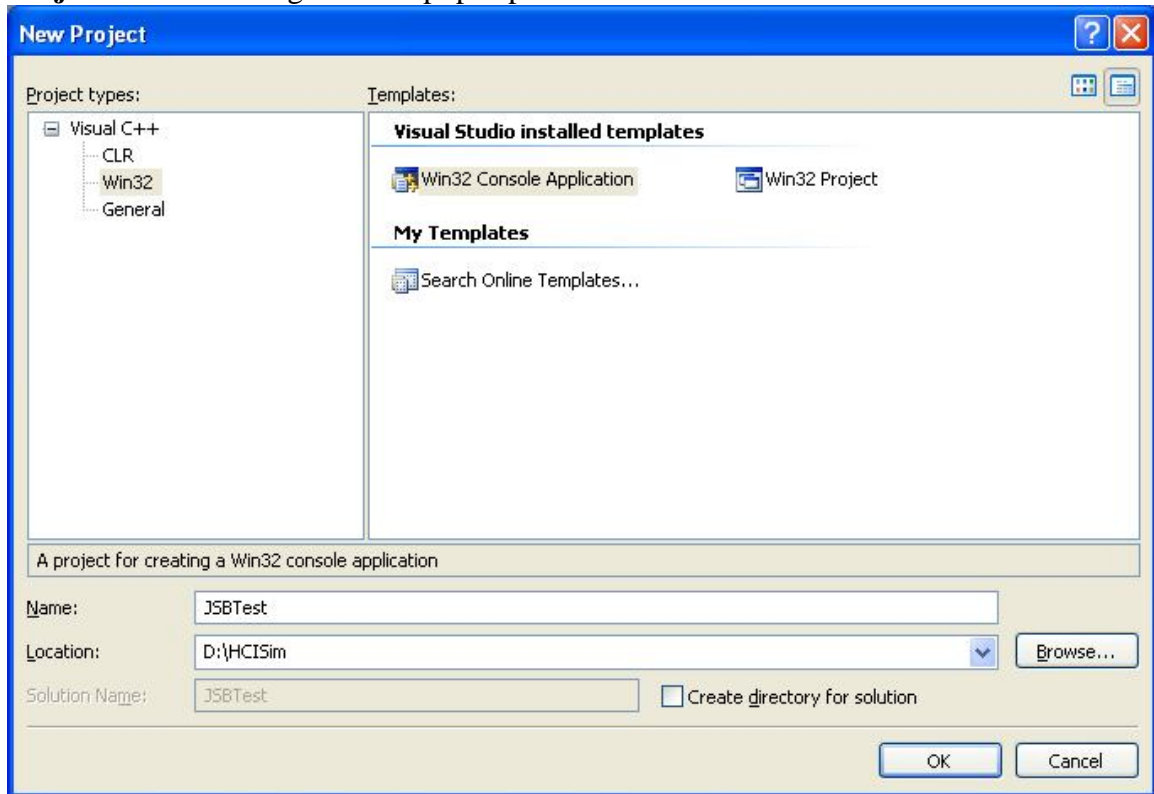
- Expand the JSBSim Solution Explorer by pressing the “+” sign on the left side, in the Solution Explorer window, beside **JSBSim**



Also expand the **Source Files** entry and scroll down to close to the bottom of this list, and find the file **JSBSim.cpp**. Right-click on this file, and select **Exclude from Project**.

- You can now build the library in a method similar to before when we were building the executable, either by pressing **F7** or by pulling down the **Build** menu line and selecting **Build Solution** or **Rebuild Solution**. The library will be created and populated with the object code. The library will be in the **JSBSim** directory subdirectory **Debug**, such as **JSBSim\Debug\JSBSim.lib**.

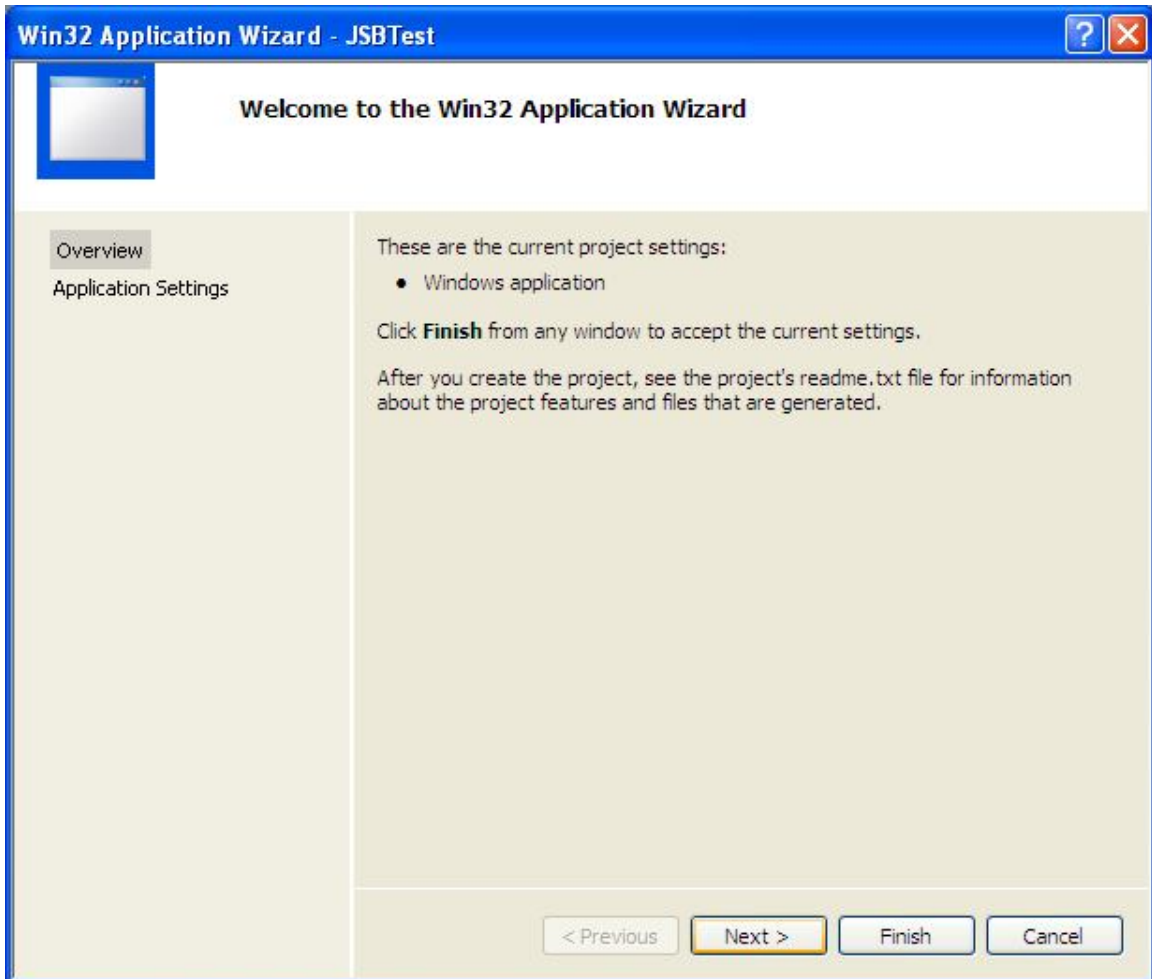
4. We are now ready to build a program using the newly created **JSBSim.lib** library. From the Visual C++, pull down the **File** menu and select **Close Solution**, which will close the JSBSim library solution. Again from the **File** menu, select **New**, then **Project**. The following window pops up:



Make sure that the **Win32 Console Application** is selected. In the **Name** box, type in the name of the project. In this case, we will call it **JSBTest**.

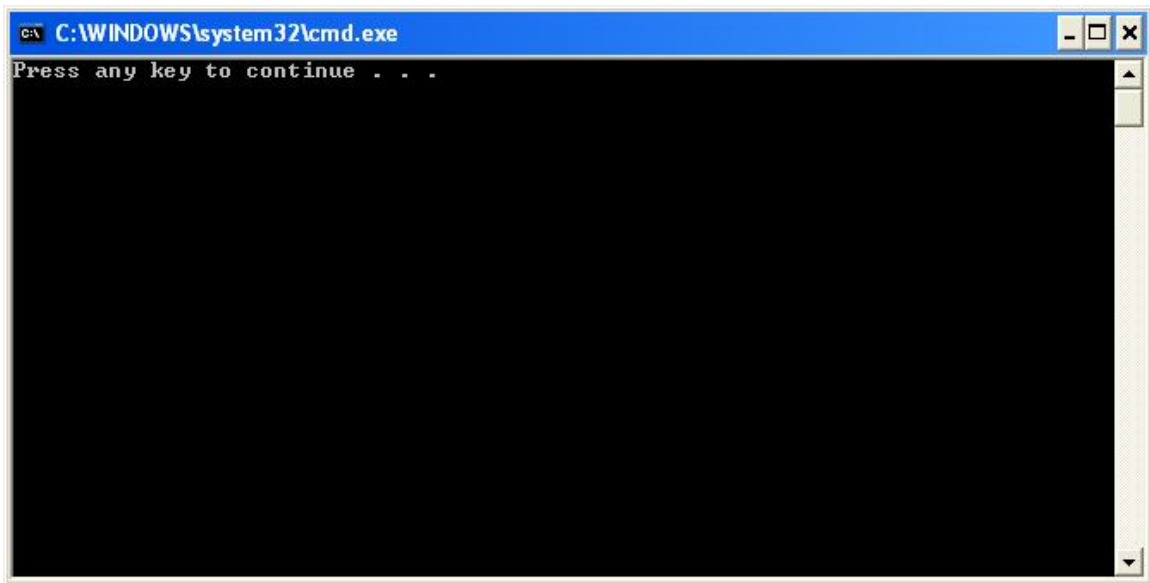
IMPORTANT STEP: In the **Location** box, you have to enter the directory where the new project will be located. I want to make this at the same level as the JSBSim directory, which is at **D:\HCISim\JSBSim**, so in this box, we want to type in **D:\HCISim**. All the steps that follow assume this directory structure. If you vary from this layout now, you will have to make adjustments in subsequent steps. If this is your first experience, set it up this way. You can always change it later.

Make sure the box marked **Create directory for solution** is NOT checked, and press **OK**. The following box appears:



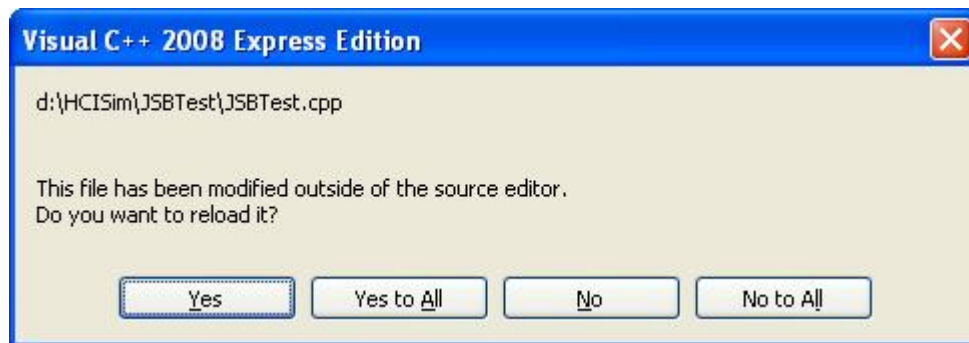
Click **Finish**, and the project solution is created. The following is what Visual C++ will look like now.

5. If you wish, you can compile and run this program, although it doesn't really do anything. Press **F7** to compile it, and when that finishes, press **Ctrl F5** to run it. A command window pops up:



Press **Enter** to close this box.

6. We are now ready to copy over the JSBSim main program into this project.. From a Windows Explorer window, find the main program file in the JSBSim\src directory. The file name is **JSBSim.cpp**. Copy this over to the JSBTest directory. (note that there isn't a **src** directory in the JSBTest directory).
7. In Windows Explorer, in the **JSBTest** directory, delete the file **JSBTest.cpp**. Rename the file **JSBSim.cpp** to **JSBTest.cpp**. You will probably get a pop-up in Visual C++ like this:



Press **Yes**, and Visual C++ will reflect the new source code for JSBTest.cpp

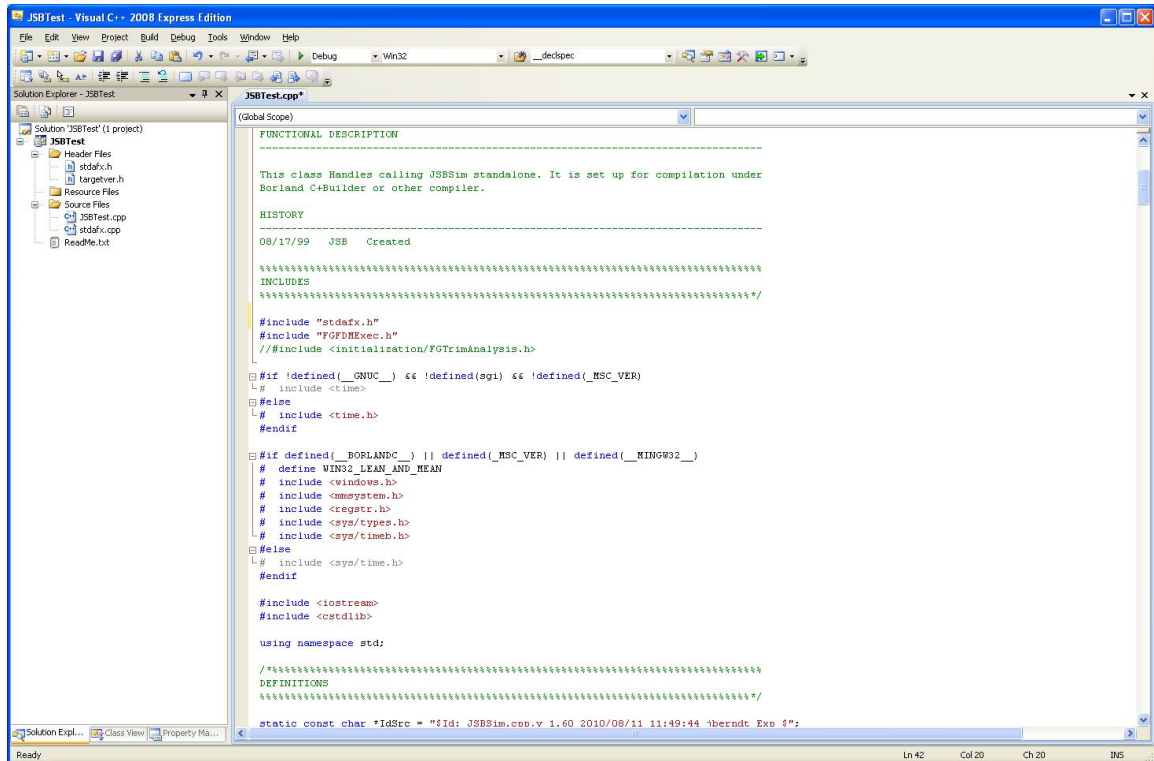
8. In the Visual C++ window for **JSBTest.cpp**, scroll down to the first include statement, which should be:

#include "FGFDMEExec.h"

and ABOVE that line, add the following line:

#include "stdafx.h"

Your code should now look like this:



```
FUNCTIONAL DESCRIPTION
-----
This class Handles calling JSBSim standalone. It is set up for compilation under
Borland C++Builder or other compiler.

HISTORY
-----
08/17/99 JSB Created

*****
INCLUDES
*****

#include "stdafx.h"
#include "FGFDMEExec.h"
// #include <initialization/FGTrimAnalysis.h>

#ifdef __GNUC__
#include <time.h>
#else
#include <time.h>
#endif

#ifdef _BORLANDC_ || defined(_MSC_VER) || defined(_MINGW32_)
#define WIN32_LEAN_AND_MEAN
#include <windows.h>
#include <memory.h>
#include <regstr.h>
#include <sys/types.h>
#include <sys/time.h>
#else
#include <sys/time.h>
#endif

#include <iostream>
#include <stdlib.h>

using namespace std;

*****
DEFINITIONS
*****

static const char *IdSrc = "$Id: JSBSim.cpp.v 1.60 2010/08/11 11:49:44 tberndt Exp $";
```

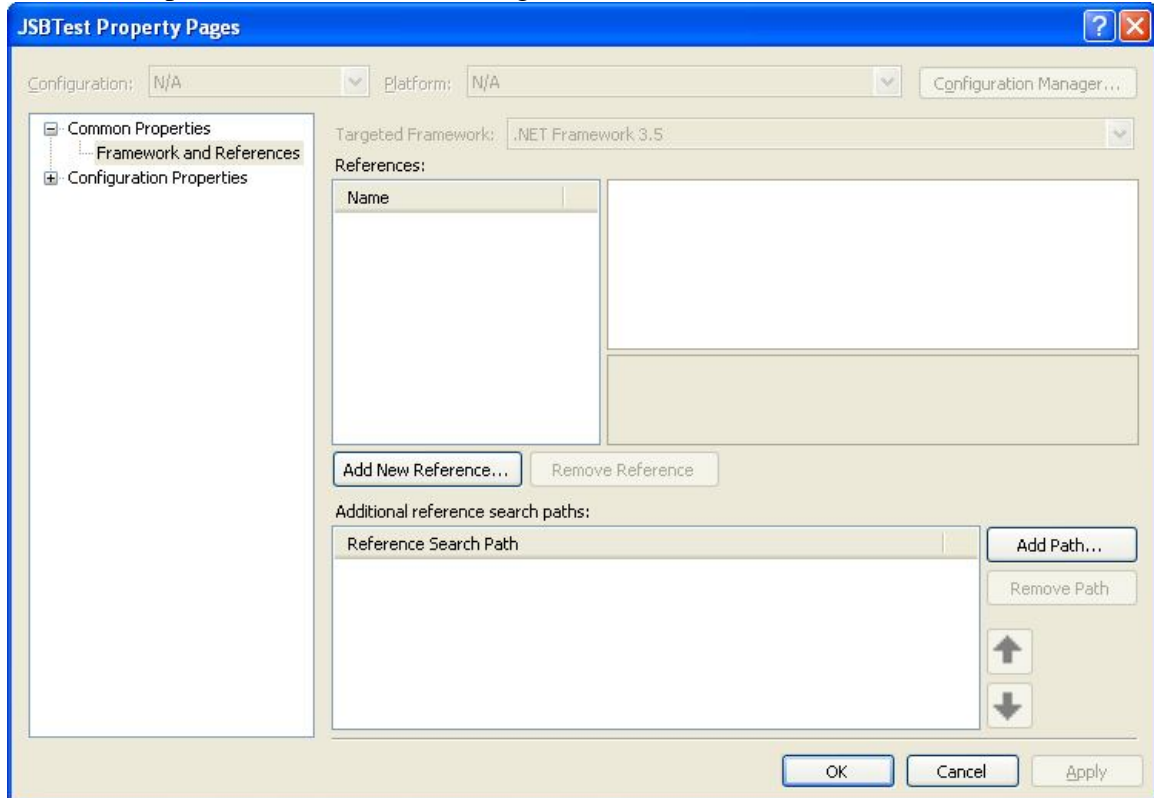
Press **Ctrl S** to save the file modifications. They are also automatically saved when you start a compile.

9. Now, we need to tell the compiler where to find the JSBSim include and library files. You can do this in one of two ways, either with a relative path or a hard path. For example, a relative path would say: "From the current directory, go up one level, then down to JSBSim", like this **"..\JSBSim"**. A hard path would be if the JSBSim directory is always going to be in the same location, such as **"C:\JSBSim"**.

For this example, we will use relative paths. If you didn't set up your directory structure where **JSBTest** and **JSBSim** are at the same level, this is where this will go wrong, unless you must adjust accordingly.

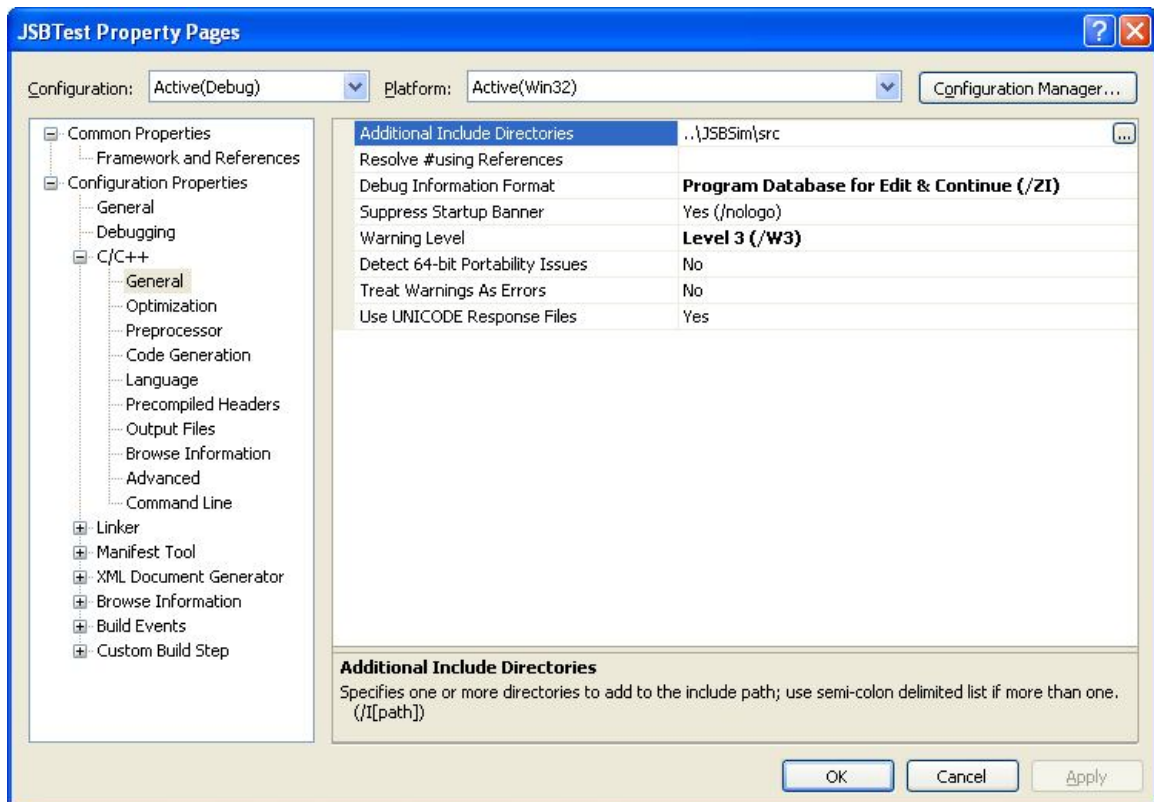
To add the paths to the include and library files, you have to change the project configuration properties. To do so, on the left side, right-click where it says JSBTest

and select **Properties**, or from the pull-down menus, **Project -> JSBTest Properties**. You will be presented with the following menu:



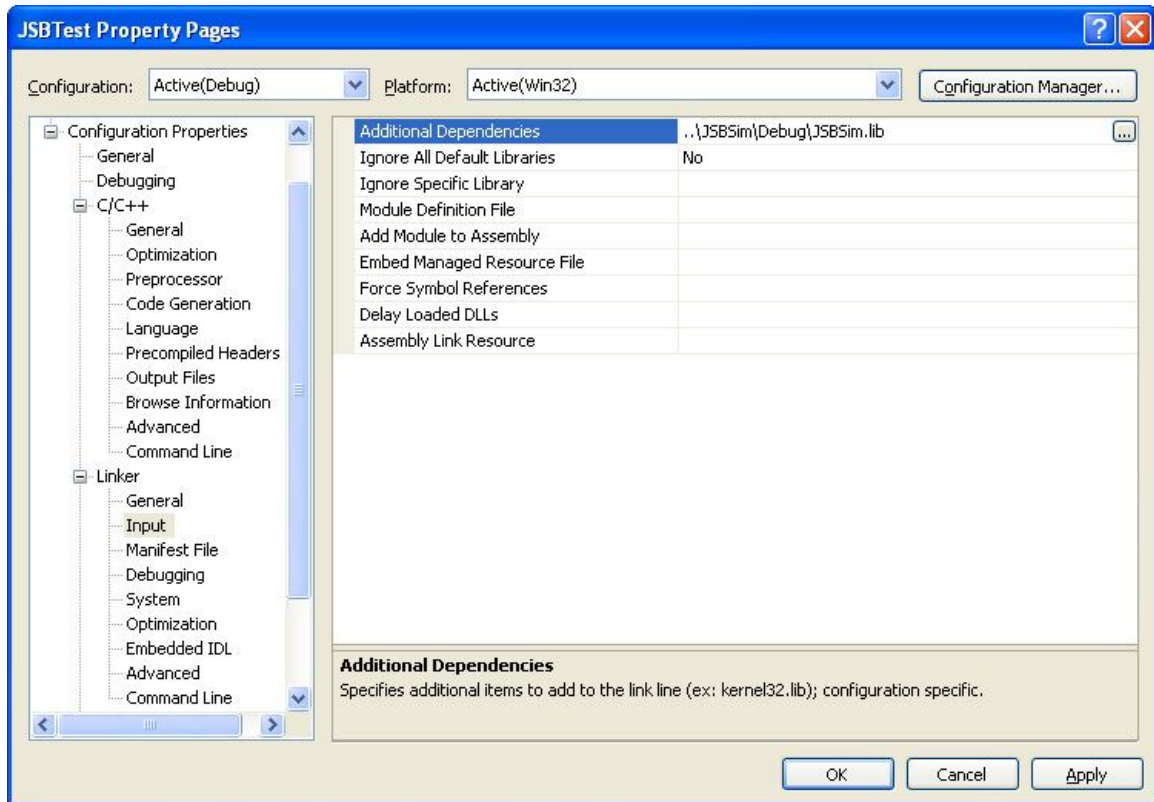
Press the “+” beside “Configuration Properties” to expand that menu. Under that, also expand “C/C++” and “Linker”.

Under the C/C++ entry, click on General. On the line that says “Additional Include Directories”, add the path to the JSBSim include directory, either as a hard path or a soft path (as shown), such as shown.

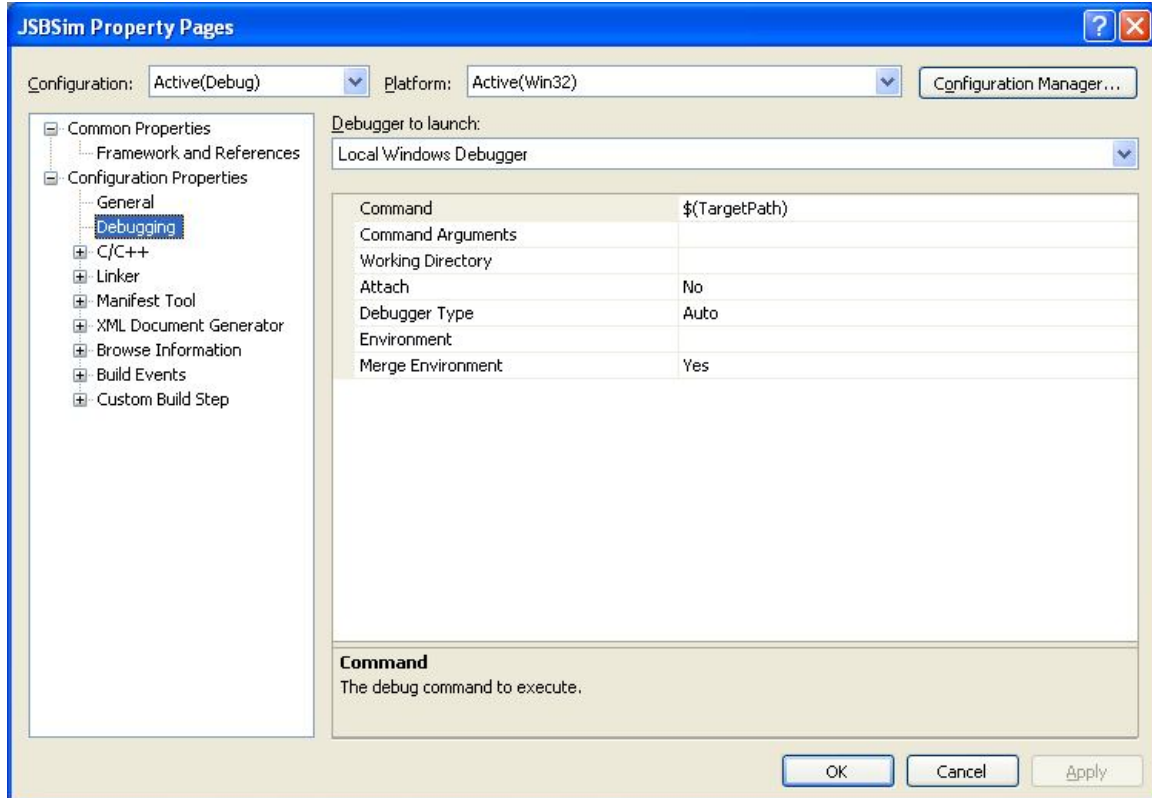


On the **Preprocessor** entry, on the **Preprocessor Definitions** line, add the entry **XML_STATIC**, separated from the previous entries with a semicolon (“;”)

Under Linker, click on the Input line entry. On the top line, Additional Dependencies, enter the path AND library name to the JSBSim library, either as a hard path or a soft path (as shown), such as shown.



then click on the **Debugging** line. You should see this:



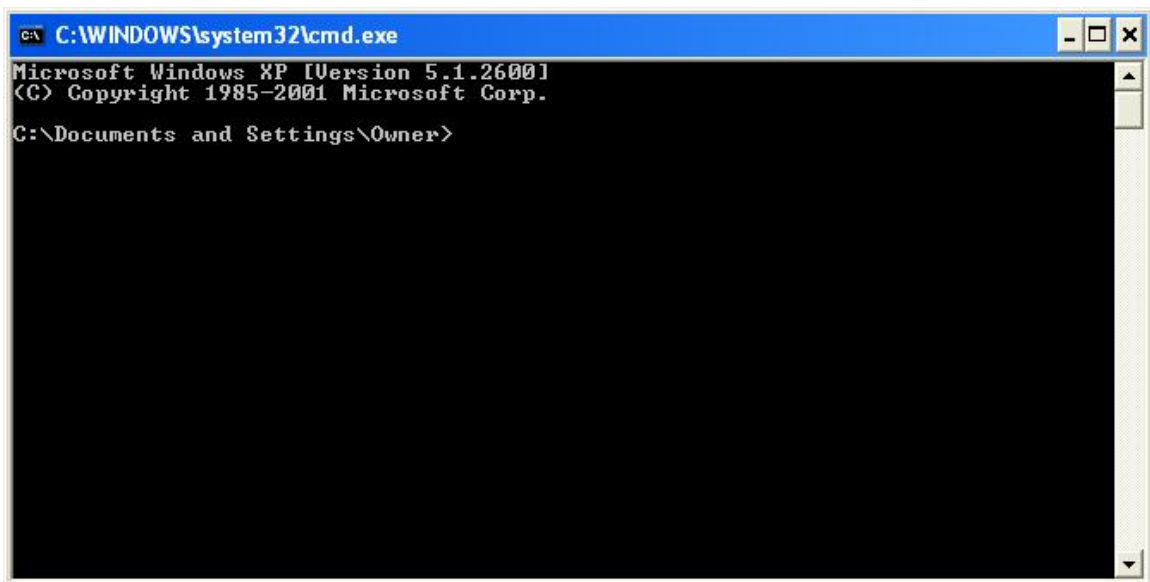
When the program is run from Visual C++, the command is issued to run the JSBSim.exe program using the full path to the executable, which is all contained in the replaceable parameter **\$(TargetPath)**. By default, the program starts in the default Windows working directory, but in this case, we want to be in the JSBSim directory, where the library was built, so that we can pick up all of the directories under JSBSim, such as **scripts**, **aircraft**, and other directories. You could copy those directories to the **JSBTest** directory, but I'll show it another way. We are going to put in a relative path to the JSBSim directory as the working directory. On the right side, find the line that says **Working Directory**. Edit this line to say **\$(ProjectDir)\..\JSBSim**.

On the right side, find the line that says **Command Arguments**, and enter the command line argument **--script=scripts\c1723.xml**, which will execute that script when the program is started. You can now press **OK** to close the JSBSim Property Page.

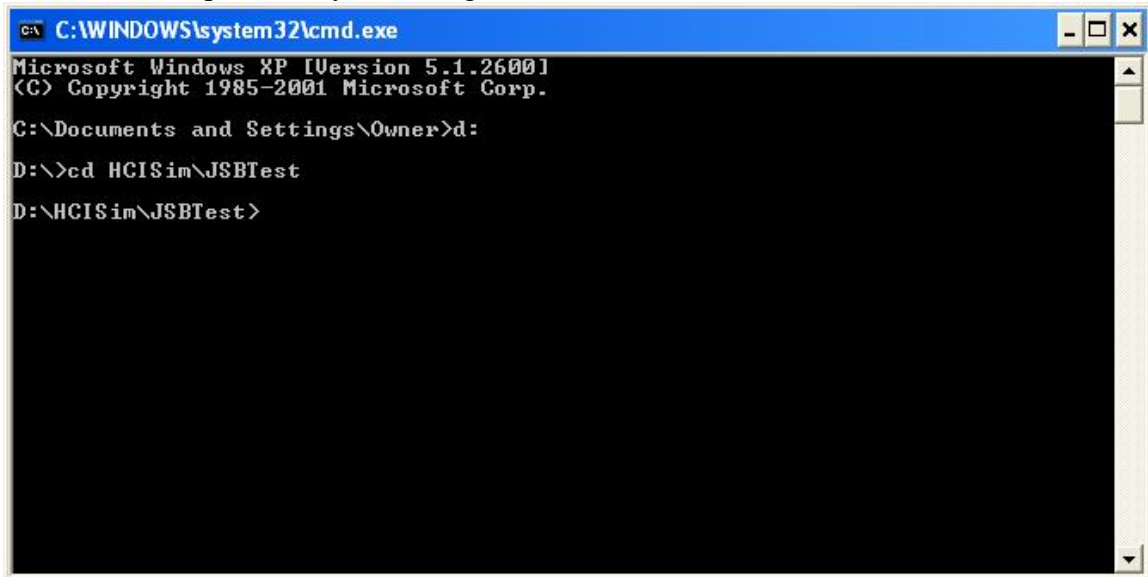
Click "**Ok**"

You are ready to compile. From the pull-down menu, select Build -> Build Solution, or just press F7.

10. You can now run the program, either by pressing **Ctrl-F5**, or pulling the **Debug** menu and selecting **Start without Debugging**. A command window will pop up and execute the JSBSim program. It runs for a couple of minutes with this script, which starts the C-172 and performs a take-off. It generates a comma-separated values (.csv) file in the JSBSim directory called **JSBout172B.csv**. You will have to press **Enter** when it finishes, and the window is closed automatically.
11. Another way to run the JSBSim program is from a command line. Visual C++ provides a command line prompt entry in the Windows Start menu under the Visual Studio tools, which provides a path to all of the Visual C++ tools, etc. However, we don't need this, so you can either open that up, or just open a command window by pressing the **Windows key** and **R**, or by pulling up the start menu and typing in **cmd** and pressing **Enter** or the **OK** button. That will cause a command window to pop up.



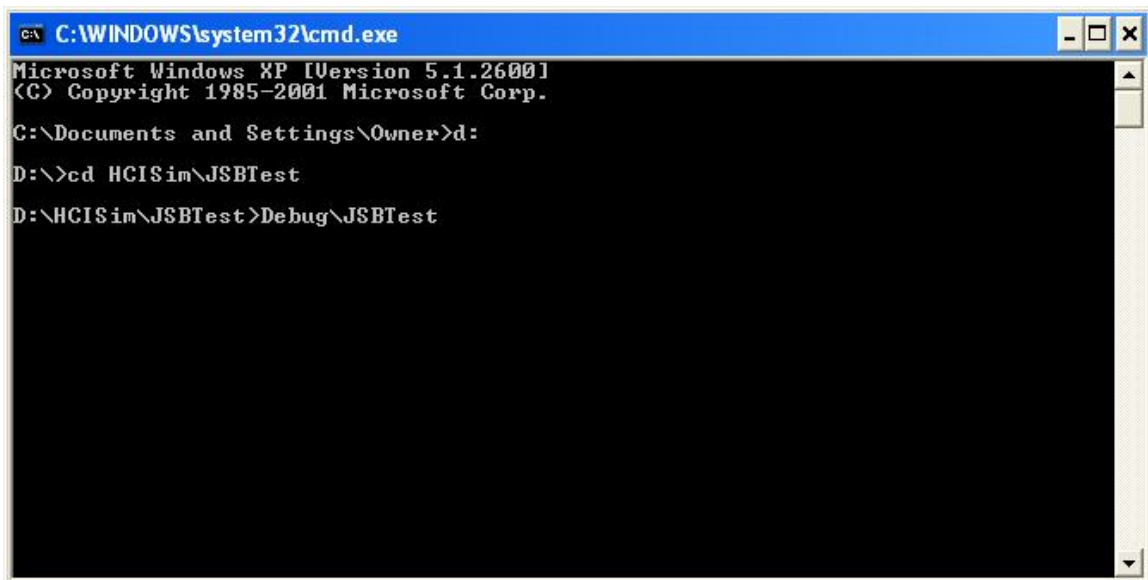
Let's go to the JSBTest directory and run the program, so type in the commands as shown, or as required for your configuration:



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Owner>d:
D:\>cd HCISim\JSBTest
D:\HCISim\JSBTest>
```

Expand the window to maximum by pressing the box next to the “X” box in the upper right corner of this command box, then run the program, which is located in the **Debug** directory



```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows XP [Version 5.1.2600]
(C) Copyright 1985-2001 Microsoft Corp.

C:\Documents and Settings\Owner>d:
D:\>cd HCISim\JSBTest
D:\HCISim\JSBTest>Debug\JSBTest
```

This will display the help menu for the JSBSim program, which we have renamed to JSBTest. We didn't specify any command line arguments, so by default, it presents this information.

In order for the command line arguments to be correct, the **scripts** and other directories need to be located under where we are running. We could copy those file here, but let's do it another way, just for another example.

Change the directory to the **JSBSim** directory, where we built the library and the **scripts** and other directories reside. In my case, I used:

```
cd ..\JSBSim
```

Now, to run the **JSBTest** program, we have to issue that path, so use this command:

```
..\JSBTest\Debug\JSBTest -script\scripts\c1723.xml
```

and as before, the **JSBSim** program runs, spitting out a lot of messages, and produces the same **JSBout172B.csv** file as if we ran it from Visual C++.

Close this window by typing **Exit**.