C# Development Guide, v1.0

We provide samples in a variety of languages. One of the newest is Microsoft C#, a .NET language.

All samples we provide are pre-compiled for your convenience – you do not need to own a compiler, let alone know how to use one. However, it is necessary for .NET languages to have the .NET framework installed, and for most Microsoft languages to have the equivalent run-times installed.

To run the C# samples, please make sure you have the correct .NET framework installed. You can find the Microsoft .NET frameworks here: http://www.microsoft.com/net

.NET languages attempt to lockdown the programing environment to prevent certain types of security flaws from being introduced. This is called "Managed" programming, and really refers to the fact that these languages are very high-level scripting languages that perform much of the low-level programming chores for the developer. This can be a good thing (more secure, easier) but it can also be a drawback (larger code that executes slowly, difficulty integrating with hardware, quirks when integrating with other languages).

This "managed code" environment prevents C# code from calling directly into certain types of drivers and DLLs, like the DLLs used to control data acquisition hardware in other languages, without violating the "managed" wrapper. To avoid this violation, we have provided a C# language wrapper for the driver DLLs.

The USB driver API, AIOUSB.DLL, is wrapped up in AIOUSBNet.DLL. This DLL is simply a little piece of code written in C# that marshals the parameters used into forms .NET is more comfortable with calling as "managed". The full source is provided under your installation path's /win32 directory, so you can take a look at it if you'd like.

This AIOUSBNet.DLL replaces certain file types used in other languages, things like "header files" "lib files" "interface files" etcetera. This type of .NET DLL is often referred to as a "Class Library" – every function from our AIOUSB.DLL is provided in the form of a C# .NET compatible "Class".

The only provided support at this time is for 32-bit systems. So, the first tip in this guide:

1) Make sure your Application target development system is "x86", not "any"

Please note: some versions of Visual Studio may not have a convenient way to set the x86 configuration (they are coded always to "any"). Here's an article on how to modify your project file in those cases: http://social.msdn.microsoft.com/Forums/nl-BE/vblanguage/thread/d4fa83dc-eed1-4ead-96a1-78bbd9ba6d3a

These samples were built using Visual Studio 2010, but can be converted to compile in older versions.

2) Create a brand new project in your version, then copy and paste the source code into that new project to build our VS2010 code in your older version.

3) If you're rebuilding one of our Class Libraries (AIOUSBNet.DLL for example), make sure you select a Class Library Project when you create your new project to get all the settings correct.

4) The Class Library DLL can be made much more convenient to use if you install it into the GAC (Global Assembly Cache). This process is difficult, but we made it easy – Check out the sub-project in the AIOUSBNet.DLL solution which will create an installer .MSI file for you. By simply building this project and running the resultant .MSI file, the dll and settings will be properly installed into the GAC.
As always, check for the latest versions of our code at our website, and feel free to chat, email, or call for technical support. We're here to help!