PMC Precision MicroControl Corp.

TechNOTE

Title: Programming Support for the .NET Platform on Windows
Products(s): MCAPI
Keywords: .NET, C#, VB.NET, assembly
ID#: TN1064
Date: May 4, 2006

Summary

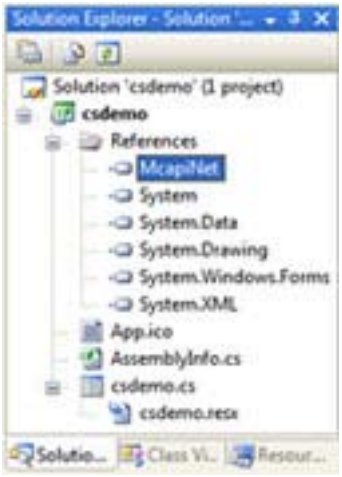
Support for Microsoft's .NET platform in the form of a .NET assembly is included starting with version 4.0.0 of the Motion Control API (MCAPI). This TechNote provides additional details.

More Information

Support for the MCAPI under .NET is implemented as a number of related .NET classes, implemented in the assembly McapiNet.dll. In addition, the file McapiNet.xml contains MCAPI online help in the Intellisense format. This format is supported by recent version of Visual Studio. Both files are normally installed to:

```
C:\Program Files\Motion Control\Motion Control API
```

Using McapiNet requires that the .NET Framework, version 2.0 or later, be installed on your PC. The .NET framework may be downloaded free of charge from http://www.microsoft.com, or by using the Windows Update tool in Windows XP or Vista.



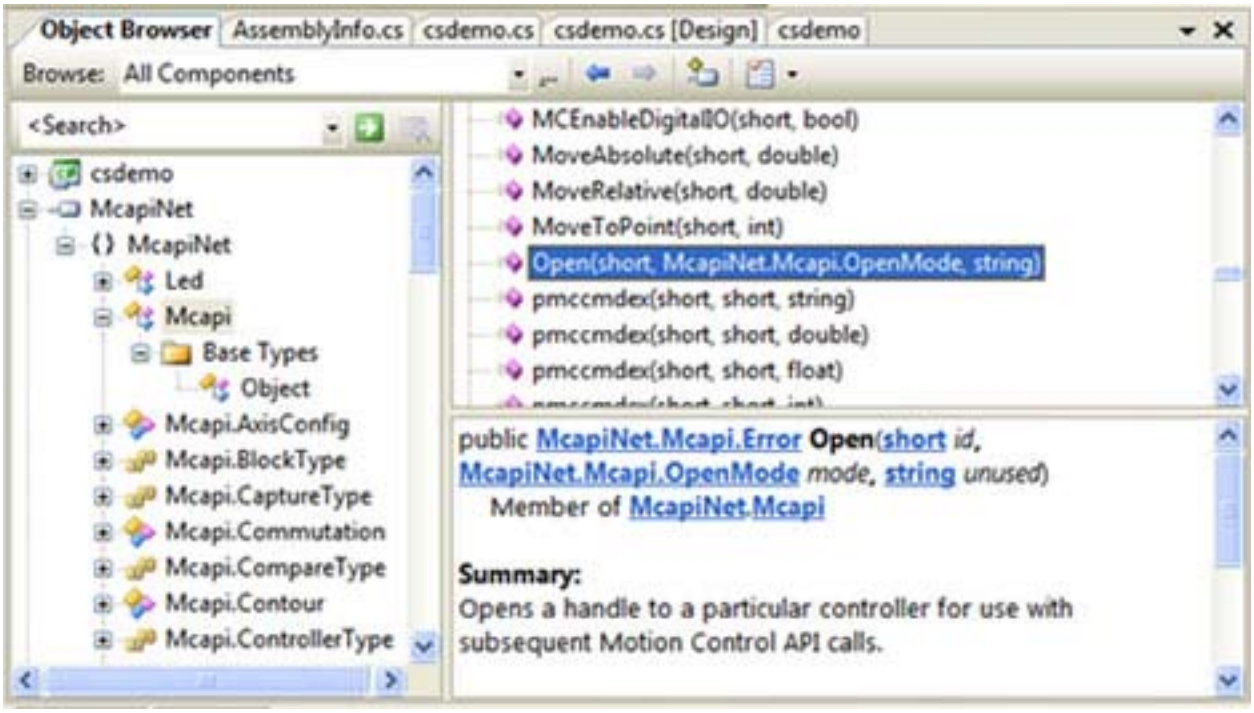
To begin programming with McapiNet you must first add the McapiNet assembly to the references for your project. In Visual Studio, right click on References in the Solution Explorer and select Add Reference. Click on the Browse tab, navigate to the location where the Motion Control API was installed, and select the McapiNet.dll. Visual Studio will automatically detect and use the Intellisense help file McapiNet.xml.

McapiNet contains two main classes. The Mcapi class implements the Motion Control API functions as methods, while the Mcdlg class implements the Motion Dialogs (motor setup dialog, select controller dialog, etc.) as methods. In addition to these main classes there are numerous smaller objects and enumerations that facilitate using McapiNet in a typesafe object-oriented manner.

Begin your program by creating an instance of the Mcapi class and calling its Open() method to open a motion controller. Unlike the standard MCAPI MCOpen() function there is no handle returned, the handle is managed internally by the Mcapi class:

```
Controller = new Mcapi();
if (Controller.Open(id, Mcapi.OpenMode.Binary, null) ==
Mcapi.Error.NoError)
{
      Controller.MoveRelative(1, 1000.0);
}
```

One other thing to notice here is that the traditional MCAPI function names such as MCOpen() and MCMoveRelative() have all had there names shortened by removing the “MC” prefix. This prefix makes sense in a functional programming environment where it is often difficult to differentiate local functions from library functions, but it is redundant in an object oriented framework. Use the Object Browser in Visual Studio to see all the methods in McapiNet.



All the programming constants of the MCAPI function library have been migrated to .NET enumerations. McapiNet methods require the use of the correct enumeration types. This makes it almost impossible to pass a bad value for a constant to a McapiNet method in your program. The Open() method for instance will only except a Mcapi.OpenMode value for the open mode, Visual Studio will consider any other value a compile time error.

Error handling is improved under .NET with all methods returning an error status as their return value. If you prefer you may set the ThrowExceptions member variable of the Mcapi class to True and McapiNet methods will throw .NET exceptions when an error is detected.

C#Demo, included with MCAPI 4.0.0 and later, is a C#/.NET implementation of our classic demo program that uses McapiNet. Source code for C#Demo is located in the Sources folder of your MCAPI installation.