

Using the NCPS Library DLL

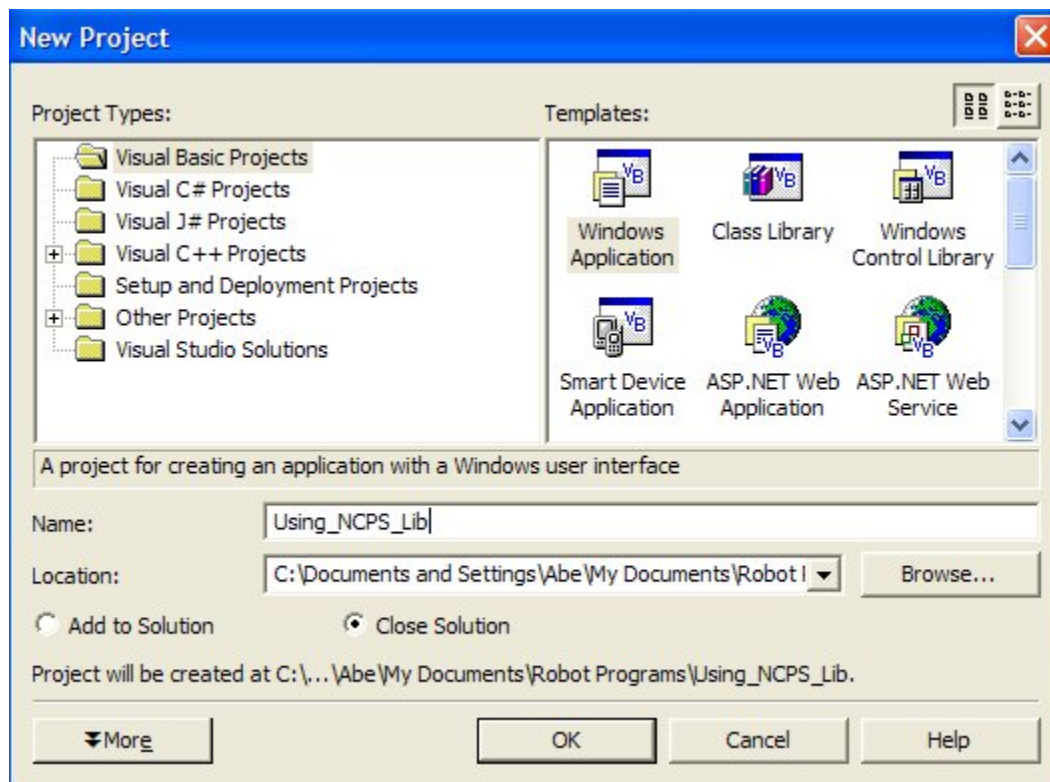
By Abraham L. Howell

Abstract:

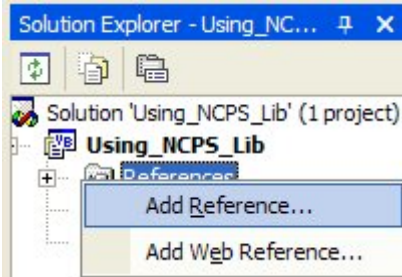
This paper will explain how to utilize the NCPS_Lib, which is a Visual Basic Class Library that compiles an OOPic II+ program using the OOPicMK.exe. This library hooks into the OOPicMK.exe to compile an OOPic II+ program and then reads in and properly parses the *.oex OOPic Executable file. The OOPicMK.exe was created by Savage Innovations and is freely distributed with the OOPic Multi-Language Compiler, which is used to create and upload programs to their line of [OOPic micro-controllers](#). After using the NCPS_Lib to compile an OOPic II+ program, the resulting *.oex.txt file can be wirelessly uploaded to an [EmbeddedBlue Transceiver](#) equipped OOPic II+ micro-controller. I make no guarantees with respect to this paper or the freely supplied source-code and will not assume responsibility for any damage caused. All Trademarks belong to their respective owner.

Using the NCPS_Lib:

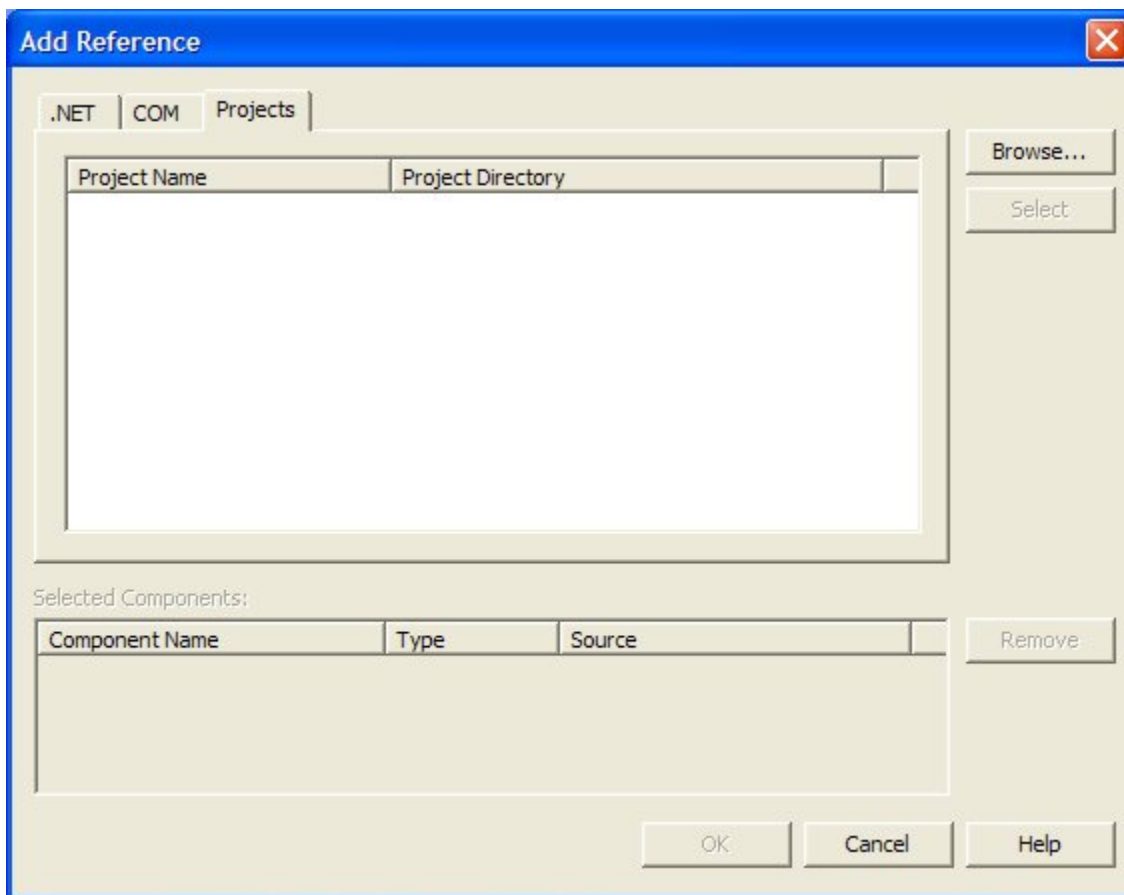
I will demonstrate how to use the NCPS_Lib using Visual Basic.Net®, but you can utilize any other .Net programming language if you are familiar with using Class Libraries. To start, simply create a new Visual Basic Windows® Application Project as shown below. As you can see, I've named the project Using_NCPS_Lib, but you can give your project any name.



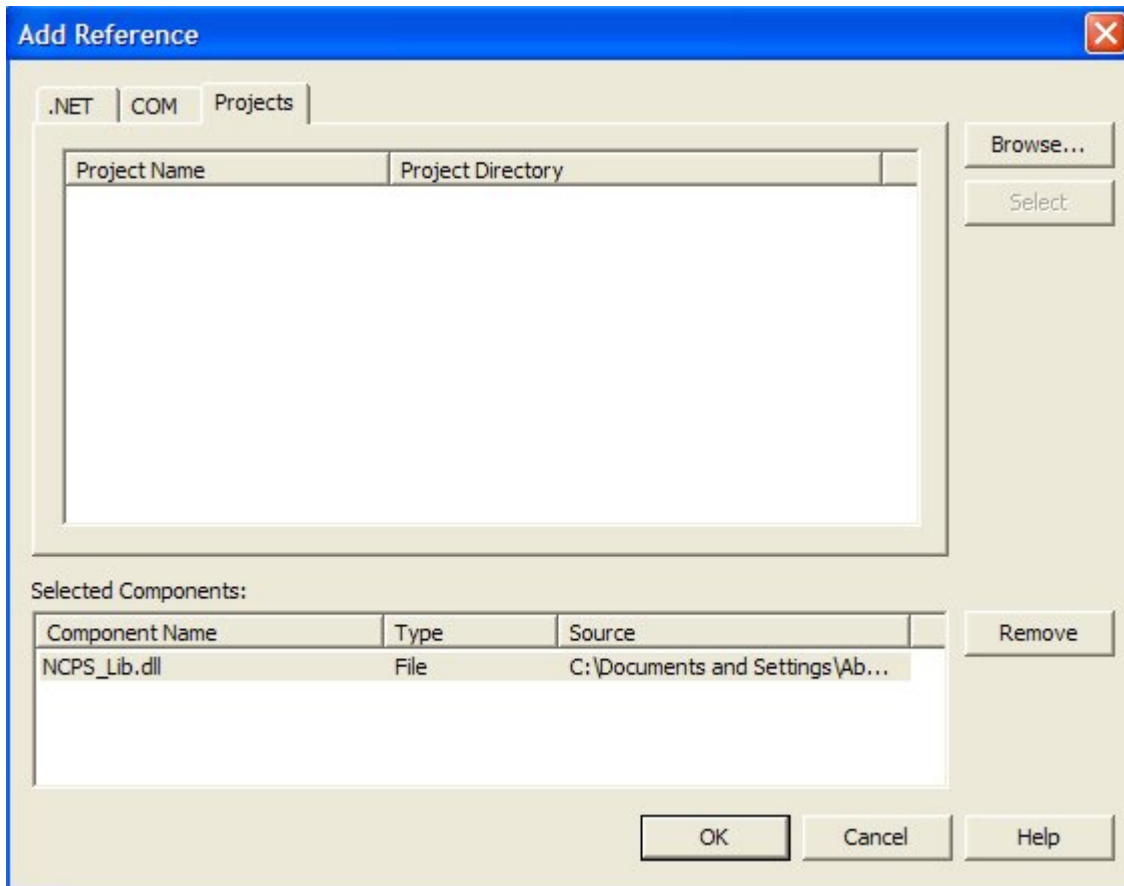
After creating the new project, the first task will be to add a reference to the NCPS_Lib. Before we do this you must copy the NCPS_Lib.dll to the bin directory of this project. To add a reference to the NCPS_Lib simply right click on references under the Solution Explorer and select Add Reference as shown below.



Next, click on the Projects tab and select Browse.



Now double-click on the bin directory and finally click on the NCPS_Lib.dll and click Open. You will be brought back to the Add Reference dialog and should notice that the NCPS_Lib.dll has been added to the Selected Components section. Finally click OK to exit.



Now that we have a reference to the NCPS_Lib we need to declare an instance of this dll for use in our application. Simply add the following code.

```
Dim NCPS_Blue As New NCPS_Lib.ncps_compile
```

Add the above line of code just after the class declaration of form1. To get to the form1 code, simply right-click on the form and select View Code. Once you've added the above line of code you should have the following.

```
Public Class Form1
    Inherits System.Windows.Forms.Form

    Dim NCPS_Blue As New NCPS_Lib.ncps_compile
```

Now we are actually ready to compile an OOPic II+ program. First, you will need to have a test program and the OOPicMK.exe available in the project's bin directory. I will use a program named, EmbeddedBlue.osc for demonstration purposes. We need to create a button on form1 that will perform the actual compilation of our test program. I am going to change the text of our button to "Compile". Now we are ready to add the appropriate code to our button so that our test program will be compiled. To do this double-click on the button and add the following code.

```

App_Path = CurDir() & "\"
Current_Path = CurDir() & "\"
OOPicFileName = App_Path & "EmbeddedBlue.osc"

'Now compile the program
strCompiledString = NCPS_Blue.Compile_Program(OOPicFileName,
Current_Path, App_Path)
If strCompiledString(0) = "ERROR" Then
    MsgBox(strCompiledString(1))
End If

```

The NCPS.Compile_Program method requires the OOPic file name, current path, and application path and will return the properly formatted string array, which can be uploaded to the OOPic II+. The OOPicFileName string must contain the OOPic source code and path. For example, suppose our application has the following bin directory location and the OOPic source code file is also located in this directory and is named “EmbeddedBlue.osc.

C:\NCPS_Bluetooth_Uploader\bin

OOPicFileName would then consist of the following string:

“C:\NCPS_Bluetooth_Uploader\bin\EmbeddedBlue.osc”

Current_Path and App_Path would then consist of the following string:

“C:\NCPS_Bluetooth_Uploader\bin”

We still need to add two more lines of code after our NCPS_Lib declaration, which should appear as follows.

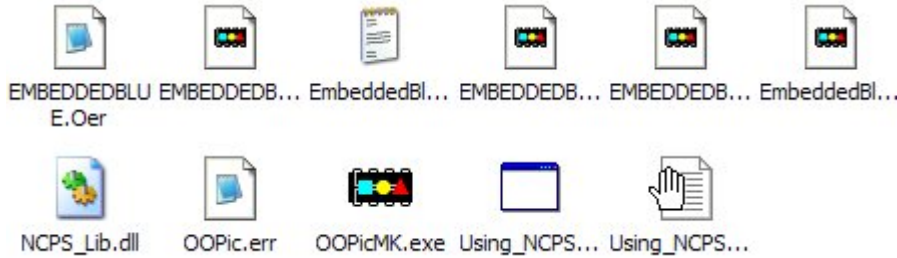
```

Public Class Form1
    Inherits System.Windows.Forms.Form

    'NCPS Compiler Library DLL
    Dim NCPS_Blue As New NCPS_Lib.ncps_compile
    Dim strCompiledString() As String
    Dim OOPicFileName, App_Path, Current_Path As String

```

You are now ready to test out your application. Simply Start Without Debugging by pressing Ctrl+F5 and then click the button, which in my case reads “Compile”. You should immediately see the compile progress bars. Afterwards, go into the bin directory to see what files were created.



You should find the following new files: EmbeddedBlue.Oer, EmbeddedBlue.Oex, EmbeddedBlue.Oex.txt, EmbeddedBlue.Omp, EmbeddedBlue.Ops, OOPic.err. The file that we will be using next is the EmbeddedBlue.Oex.txt file, which contains the properly formatted hex strings that will be uploaded to the OOPic II+ using Bluetooth.

Upload *.oex.txt file using Bluetooth:

To be able to upload the *.oex.txt using Bluetooth we'll need to add methods for accessing the computer's serial port. I will not be detailing the steps necessary to accomplish this since I am making the source-code for this project readily available as a download from www.abotics.com However, I will focus on exactly how the compiled program is uploaded to the OOPic II+ controller.

The "Upload_Program_Wirelessly" Sub is responsible for uploading the compiled program. If you look at this Sub the first thing you'll notice is that it checks for a valid open serial port and if one isn't found an exit occurs. After that it can be seen that the file path and name for the compiled and parsed program is attained. The remaining code is responsible for preparing the OOPic II+ to receive the program wirelessly. First the OOPic II+ must be placed into verbose mode.

```
'This command places the OOPic in verbose mode
oCP.Write("\0v")
'Read in the response
If GetOOPicResponse() = True Then
    GoTo ReadError
End If
```

The above code places the OOPic II+ into verbose mode and also checks for the appropriate response from the OOPic II+ using the GetOOPicResponse function, which signifies that the OOPic II+ actually received the command.

Next, the OOPic II+ must be instructed to stop executing the current program and all virtual circuits.

```
'This command stops the currently running program
oCP.Write("S")
'Read in the response
If GetOOPicResponse() = True Then
    GoTo ReadError
End If
```

```

'This command halts all VCs
oCP.Write("X")
'Read in the response
If GetOOPicResponse() = True Then
    GoTo ReadError
End If

```

We also need to properly configure the OOPic II+ program memory before sending the compiled program.

```

'This command sets the address register to the beginning
oCP.Write("0J")
If GetOOPicResponse() = True Then
    GoTo ReadError
End If

```

```

'This command sets the I2C for 8-byte transfers and
'auto-increment on the eeprom location
oCP.Write("71H")
'Read in the response
If GetOOPicResponse() = True Then
    GoTo ReadError
End If

```

```

'This command sets the I2C address
oCP.Write("80L")
'Read in the response
If GetOOPicResponse() = True Then
    GoTo ReadError
End If

```

Now that the OOPic II+ is properly configured we are ready to upload the compiled program, but first need to open up the appropriate *.oex.txt file. We also need to setup the progress bar and make it visible as well.

```

'First get free file number
filenum = FreeFile()
FileOpen(filenum, strOexFile, OpenMode.Input, OpenAccess.Read,
OpenShare.Shared, )

'Get the number of lines to be read in so we can
'set the max on the progress bar and increment as
'the data is downloaded
pgbDownload.Maximum = LOF(filenum) / 19
pgbDownload.Value = 0
pgbDownload.Visible = True

```

Now we are finally ready to send the compiled code to the OOPic II+ using Bluetooth. We will loop through and send out the compiled program line by line until we reach the end of the file. We will also increment the progress bar for each pass and also check for the appropriate response from the OOPic II+.

```

'Now download the program to the robot!
While Not EOF(filenum)
    zLine = LineInput(filenum)
    'send it out the serial port
    oCP.Write(zLine)
    'Read in the response
    If GetOOPicResponse() = True Then
        GoTo ReadError
    End If
    'Increment progress bar
    pgbDownload.Value = pgbDownload.Value + 1
    Application.DoEvents()
End While
'Close the file!
FileClose(filenum)

```

After sending the entire file we now need to reset the OOPic II+, start running the new program and take it out of verbose mode.

```

'This command restarts the OOPic and runs the uploaded program
'No response for this command
oCP.Write("W")
'This command sets takes the OOPic out of verbose mode
'No response for this command
oCP.Write("\A")

```

You are now ready to test out the application by pressing Ctrl+F5 and then selecting the appropriate serial port and finally clicking the Upload button. You may need to read over the following paper, http://www.abotics.com/AddBluetooth_NewCDBot.pdf, to setup and wire your OOPic II+ controller with an EmbeddedBlue Transceiver and also to configure the Bluetooth Serial Connection. This will conclude my discussion regarding the use of the NCPS_Lib Class Library.

I hope that this paper helps fellow robotics enthusiasts and as always send an email to abe@abotics.com if you have any questions or feedback.