USE OF THE SFBP232.DLL LIBRARY IN VISUAL BASIC

February 2009

To use the SFBP232 library, which allows access to a COM 232 serial communication port (even through possible USB adapters), you need to include the following files in your Visual Basic project:

- **NSCCOMhelper.bas** – Provides basic functions and prototypes for linking to the dynamic library.
- **frmSFBP232Settings.frm** – Configuration settings form (*Optional*)
- **frmSFBP232Settings.frx** – (*Optional*)
- **INsccom.cls** – *Optional.* A class to be used as a template if you prefer to use an object instead of inserting the event procedures (called back by the library) into a standard form.

  ⚠ **WARNING!** If you choose to use this file, implement the necessary code in the various procedures and save it privately in your application, leaving the original template unchanged.
  Additionally, you must set the following compilation variable in the project:
  `NSCCOMHLPR_USE_CLASS = 1`

  If you want to use a regular form, do **not** load this class file.

How SFBP.DLL Works

When calling ***`OpenCOM`*** a *hWnd* argument is required. This is because the DLL begins to subclass the given Window handle.

In background SFBP.DLL start an apartment thread and then registers for private messages that Windows will send to the application, which in turn are intercepted by the DLL via subclassing.

Once the communication channel needs to be closed, calling ***`CloseCOM`*** automatically destroy the thread and released the allocated resources.

This also keep the DLL from calling back the host application (no more events are generated).

Now the application should call ***`SetClientHWND(0)`*** to unsubclass the window. In the BAS stub this is achieved by calling `cleanNSCCOMLIB`.

Therefore before calling ***`OpenCOM`*** you should be sure that the *Form* is actually loaded so the handle is valid. In addition before calling ***`OpenCOM`*** you should also call `setUpNSCCOMLIB` to register your functions callbacks.

Inserting into a Standard Form

If, instead of using an object built from the template class, you want to use a standard form, copy the following procedures and paste them into the form to be used.
Note: In this case, you do **not** need to set any compilation variable.

Copy and paste the following into the form:

```
Public Sub Nsccom1_COMerror()
End Sub

Public Sub Nsccom1_DataReady(srcAddress As Integer, destAddress As Integer, datalen As Integer,
msgType As Integer, pBuf As Long, pckType As enumPacketType, buf() As Byte)
End Sub

Public Sub Nsccom1_FirmwareInfo(Address As Integer, Serial As String, devID As Integer, Device As
String, MinorVersion As Integer, MajorVersion As Integer, Version As String)
End Sub

Public Sub Nsccom1_Map(Address As Integer, Serial As String, devID As Integer, MinorVersion As
Integer, MajorVersion As Integer)
End Sub

Public Sub Nsccom1_MapProgress(level As Integer)
End Sub

Public Sub Nsccom1_NetError(ErrCode As Integer, ErrMsg As String)
End Sub

Public Sub Nsccom1_ProgramCompleted()
End Sub

Public Sub Nsccom1_ProgramProgress(level As Integer, status As String)
End Sub

Public Sub Nsccom1_QueryReply(Address As Integer, lpData As Long, queryID As Integer)
End Sub

Public Sub Nsccom1_RemoteError(Address As Integer, ErrorCode As Integer, ErrorString As String)
End Sub
```

Then, populate each procedure as needed, or leave them empty.

To start using the library, you must invoke the `setUpNSCCOMLIB` function, passing the form in which the procedures have been inserted. This will also be the form used for subclassing to handle serial port event messages.

> *(Note: If you're using an object created from the class instead of inserting the functions into a form, initializing the object will also initialize the library, and you do not need to call* `setUpNSCCOMLIB`*.)*

When you're finished using the library, you must call the `cleanNSCCOMLIB` function to remove the reference to the object (either the class-derived object or the form).

In a form, typically insert the following in `Form_Load` and `Form_Unload`:

```
Private Sub Form_Load()
    setUpNSCCOMLIB Me   ' Setup callback functions (Nsccom1_...) and bind this window to the helper
module
End Sub

Private Sub Form_Unload(Cancel As Integer)
    Me.CloseCOM
    cleanNSCCOMLIB       ' Unbind the window from the helper module
End Sub
```

In the same form, add the following functions:

```
Public Function OpenCOM() As Boolean
    ' Helper module calls the DLL OpenCOM, giving the bound window to start subclassing
    OpenCOM = CBool(NSCCOMhelper.OpenCOM() = 1)
    If NSCCOMhelper.IsOpen = 1 Then
        ' OK open
    Else
        ' Failed
    End If
End Function

Public Sub CloseCOM()
    NSCCOMhelper.closeCOM
    NSCCOMhelper.SetClientHWND 0   ' This releases window subclassing
End Sub
```

```
Public mApplicationPort As Integer
Public mDefaultPort As Integer
```

These are two public variables corresponding to:

- The application-specific port (not saved as a default; must be stored privately if you want to recall it later).
- The default port (used if a specific application port has been set).

To programmatically set the application port, use the `SetApplicationPort` function.

By setting the application port, multiple communication channels can operate simultaneously on different ports.

You can also set the station address *before* opening the communication. This won't be saved in the default values unless you explicitly call `SaveSFBPdriverSettings`.

> ⚠ Be aware that this can also be done by the user when opening the settings form.

Settings Form

==========

To open the settings form, simply create and show `frmSFBP232Settings` (typically as a modal window).

Example:

```
Private Sub cmdSettings_Click()
    frmSFBP232Settings.Show 1
End Sub
```