Calling AxsunOCTControl.dll from unmanaged C++ code in Windows Visual C++

Background

Calling methods in the .NET-based *AxsunOCTControl.dll* from a managed environment (.NET/C#, MATLAB, LabVIEW, Java, etc.) usually involves a straightforward constructor call. However, to call methods in the managed *AxsunOCTControl.dll* from an unmanaged C++ environment, one needs to perform some additional configuration steps described here.

Part 1: Assembly Registration using RegAsm.exe (via Windows Command Line)

1. If desired, copy *AxsunOCTControl.dll* (and all of its .dll dependencies, e.g. *LibUsbDotNet.dll*) to the desired location on your disk, such as your project directory. Alternatively, you can leave the library where it is currently installed alongside the *OCTHost.exe* application or the *AxsunAdvancedHardwareTool.exe* application.

2. A Microsoft Windows tool called *RegAsm.exe* registers managed libraries as a COM object in the Windows Registry. On Windows 7, *RegAsm.exe* is located at the following paths (the version part of the path "\vX.Y.ZZZZZ\" might be different than shown here):
   a. 32-bit: *C:\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe*
   b. 64-bit: *C:\Windows\Microsoft.NET\Framework**64**\v4.0.30319\RegAsm.exe*

   Use the version appropriate for your target application (i.e. WIN32 or x64).

3. Open a Windows **cmd** prompt and change the current directory to that of your *AxsunOCTControl.dll* file, for example:
```
cd c:\Program Files (x86)\Axsun\Axsun OCT Control
```

4. Now execute the RegAsm command as appropriate for your desired bitness:
```
c:\Windows\Microsoft.NET\Framework\v4.0.30319\RegAsm.exe AxsunOCTControl.dll
/tlb:AxsunOCTControl.tlb /codebase
```
-or-
```
c:\Windows\Microsoft.NET\Framework64\v4.0.30319\RegAsm.exe AxsunOCTControl.dll
/tlb:AxsunOCTControl.tlb /codebase
```

5. This will create the *AxsunOCTControl.tlb* file in the same directory. Move the .tlb file to the same project directory as your .cpp source code which will be calling the *AxsunOCTControl.dll* library functions. (NOTE: Do NOT move the *AxsunOCTControl.dll* file or its dependencies! Doing so will require you to repeat the RegAsm of the library from its new location.)

Part 2: Importing the .tlb file, starting COM, and accessing .dll functions (in Visual C++)

6. Import the .tlb file using the #import directive at the top of your source code:

```
#import "AxsunOCTControl.tlb"
```

7. Use the AxsunOCTControl namespace:
```
using namespace AxsunOCTControl;
```

8. Wrap the segment of your code that contains calls to AxsunOCTControl.dll with calls to initialize and uninitialize COM. For example, initialize COM at the beginning of your main() function and uninitialize it just before completing your main() function:
```
main () {
        CoInitialize(NULL);           // init COM
        // AxsunOCTControl.dll calls go here
        CoUninitialize();             // un-init COM
}
```

9. Create a smart pointer to an AxsunOCTControl structure:
```
IAxsunOCTControlPtr pAxsunOCTControl(__uuidof(struct AxsunOCTControl));
```

10. Dereference this smart pointer to access the functions available in the AxsunOCTControl structure, such as:
```
long numDevices = pAxsunOCTControl->GetNumberOfOCTDevicesPresent();
```

Example C++ Source Code (no error checking shown)

```
#include <iostream>
#import "AxsunOCTControl.tlb"                     // import the assembly's tlb
using namespace std;
using namespace AxsunOCTControl;

int main() {

    // initialize COM
    CoInitialize(NULL);

    // create smart pointer
    IAxsunOCTControlPtr pAxsunOCTControl(__uuidof(struct AxsunOCTControl));

    // check how many OCT devices are connected
    long numDevices = pAxsunOCTControl->GetNumberOfOCTDevicesPresent();

    // report status
    cout << "num of devices is " << numDevices << '\n';

    // uninitialize COM
    CoUninitialize();

    return 0;
}
```