Documentation for the Visual Basic COM Wrapper of the Zeiss and Fibics API Functions

(Kenneth Hayworth Feb. 10, 2011)

Zeiss provides an Application Programming Interface (API) with its SEM so that third party developers can automate the SEM's functions. This Zeiss API (see pdf document SmartSEM_Remote_API_Manual.pdf) is in the form of an ActiveX control (CZEMAPI.OCX) that can be directly used in programs like VB, C++, etc. Unfortunately the functions in this Zeiss API use a variant data type passed by reference and because of this Matlab cannot directly access the API. I have therefor written a 'wrapper' COM library around this Zeiss API using Visual Basic. This wrapper provides Matlab users with functions taking specific data types and passes these on to the Zeiss API functions.

The Fibics frame grabber hardware/software also provides an API as a Type Library, registered as "NPVE3Z Library". I encountered some problems trying to access these directly through Matlab as well so I included wrapper functions to them in the same COM wrapper as the Zeiss API.

The following is a complete listing of the functions made available to Matlab via my VBComObjectWrapperForZeissAPI Class Library:

```
Function Initialise(ByVal Machine As String) As Integer

Function InitialiseRemoting() As Integer

Function GetLimits_MaxValue(ByVal Param As String) As Single

Function GetLimits_MinValue(ByVal Param As String) As Single

Function GetMag() As Double

Function Get_ReturnTypeSingle(ByVal CommandStr As String) As Single

Function Get_ReturnTypeString(ByVal CommandStr As String) As String

Function MoveStage(ByVal x As Single, ByVal y As Single, ByVal z As Single,
ByVal t As Single, ByVal r As Single, ByVal m As Single) As Integer

Function Grab(ByVal xoff As Short, ByVal yoff As Short, ByVal width As Short,
ByVal height As Short, ByVal reduction As Short, ByVal Filename As String) As
Integer

Function GetLastError() As String

Function ClosingControl() As Integer

Function GetLastRemotingConnectionError() As String
```

```
Function SetSuppressRemotingConnectionErrors() As Integer

Function Set_PassedTypeSingle(ByVal CommandStr As String, ByVal PassedValue
As Single) As Integer

Function Set_PassedTypeString(ByVal CommandStr As String, ByVal PassedValue
As String) As Integer

Function Execute(ByVal CommandStr As String) As Integer

Function Fibics_Initialise() As Integer

Function Fibics_AcquireImage(ByVal W As Long, ByVal H As Long, ByVal Dwell As
Single, ByVal Filename As String) As Integer

Function Fibics_IsBusy() As Boolean

Function Fibics_ReadFOV() As Single

Function Fibics_WriteFOV(ByVal FOV As Single)

Function Fibics_Cancel()
```

Using the COM Wrapper in Matlab:

To use the COM wrapper in Matlab first call the following function to create the object MyCZEMAPIClass in the Matlab workspace:

```
MyCZEMAPIClass =
actxserver('VBComObjectWrapperForZeissAPI.KHZeissSEMWrapperComClass');
```

Then initialize as follows:

```
MyCZEMAPIClass.InitialiseRemoting();
```

This initializes the communications between the ActiveX control and the SmartSEM server on the SEM PC. This must be called before any other Zeiss API functions.

The key functions that a typical Matlab SEM automation program will be using are:

```
Function Get_ReturnTypeSingle(ByVal CommandStr As String) As Single

Function Get_ReturnTypeString(ByVal CommandStr As String) As String

Function Set_PassedTypeSingle(ByVal CommandStr As String, ByVal PassedValue
As Single) As Integer

Function Set_PassedTypeString(ByVal CommandStr As String, ByVal PassedValue
As String) As Integer
```

```
Function Execute(ByVal CommandStr As String) As Integer

Function MoveStage(ByVal x As Single, ByVal y As Single, ByVal z As Single,
ByVal t As Single, ByVal r As Single, ByVal m As Single) As Integer
```

These functions are wrappers around the Zeiss API functions Get(), Set(), Execute(), and MoveStage(). The only difference between calling the wrapper functions vs. the original API functions is that if one is calling a Get() command that returns a string one needs to use the Get_ReturnTypeString() function etc. See the pdf document SmartSEM_Remote_API_Manual.pdf for a listing of all command arguments that can be used with these functions.

Here is some example Matlab code using these functions:

```
MyCZEMAPIClass =
actxserver('VBComObjectWrapperForZeissAPI.KHZeissSEMWrapperComClass');

MyCZEMAPIClass.InitialiseRemoting();

MyMag = MyCZEMAPIClass.Get_ReturnTypeSingle('AP_MAG');

MyCZEMAPIClass.Set_PassedTypeSingle('AP_MAG',100);

MyCZEMAPIClass.Execute('CMD_AUTO_FOCUS_FINE');
```

To use the Fibics API from Matlab you must have already initialized the Zeiss API as shown above. You must then make sure that the ATLAS software GUI is shut down and then call:

```
MyCZEMAPIClass.Fibics_Initialise();
```

The Visual basic code behind this call is the following:

```
    Public Function Fibics_Initialise() As Integer

        veobj = New NPVE3Z.FibicsSEMVE()

        Return 0
    End Function
```

As you can see this call creates a new NPVE3Z.FibicsSEMVE object within the COM wrapper called veobj. This object is used for all subsequent calls to the Fibics API.

The key Fibics functions that a typical Matlab SEM automation program will be using are:

```
Function Fibics_Initialise() As Integer

Function Fibics_AcquireImage(ByVal W As Long, ByVal H As Long, ByVal Dwell As
Single, ByVal Filename As String) As Integer

Function Fibics_IsBusy() As Boolean
```

```
Function Fibics_ReadFOV() As Single

Function Fibics_WriteFOV(ByVal FOV As Single)
```

Here is some example Matlab code using these functions to grab an image:

```
FOV_microns = 4096;

MyCZEMAPIClass.Fibics_WriteFOV(FOV_microns);

ImageWidthInPixels = 4096;

ImageHeightInPixels = 4096;

DwellTimeInMicroseconds = 1;

FileNameStr = 'C:\TestImage.tif';

MyCZEMAPIClass.Fibics_AcquireImage(ImageWidthInPixels, ...

   ImageHeightInPixels, DwellTimeInMicroseconds, FileNameStr);

while(MyCZEMAPIClass.Fibics_IsBusy)

        pause(1);

end
```

See the Fibics API documentation (ATLAS_API_Draft_User_Guide_April_2010.pdf) for more information on these functions.

NOTE: I have noticed a bug in the Fibics FOV-to-Magnification calibration such that the mag for a particular FOV will be slightly different based on what mag the SEM was in when Fibics initialized. The solution to this that I have been using is to always call the Zeiss API to set the microscope to 25x just before calling MyCZEMAPIClass.Fibics_Initialise();

****

Creating and installing the VBComObjectWrapperForZeissAPI Class Library:

Note: I have already created this COM library and installed it on the Fibics PC. The following steps are listed here simply as documentation on how this was done.

1. Create in VB.net a new project of type Class Library
2. Delete the existing Class1.vb from the solution explorer

3. Right click on project and Add.. New... COM Object. Call it "KHZeissSEMWrapperComClass.vb"
4. Project -> Add Reference... under COM add CZ EM API OLE Control
5. Add a form to the project (only way to get to activex control). Call it Form1
6. In the Form1.vb [Design] use the toolbox to drag a CZ EM API control onto the form
7. In the top of the KHZeissSEMWrapperComClass.vb put the line: "Dim MyForm As New Form1"
8. Any function call to the api will use the following syntax: MyForm.AxApi1.InitialiseRemoting()
9. Build this DLL and then register using the RegAsm.exe program with /tlb option:
   C:\>C:\WINDOWS\Microsoft.NET\Framework\v2.0.50727\RegAsm
   C:\Users\Hayworth\VBComObjectWrapperForZeissAPI\VBComObjectWrapperForZeissAPI\bin\Debug\VBComObjectWrapperForZeissAPI.dll /tlb
10. In Matlab create an object with this: MyCZEMAPIClass =
    actxserver('VBComObjectWrapperForZeissAPI.KHZeissSEMWrapperComClass')