TESCAN Scanning Electron Microscope

SharkSEM Remote Control

DrawBeam Extension

Revision 2.0.10
25th July 2012

SEM Models: Vega 3, Mira 3, Vela 3, Lyra 3, Fera 3

TESCAN, a.s.
Brno, Czech Republic

Table of Contents

Preface

All TESCAN Scanning Electron Microscopes (SEMs) are equipped with remote control capability. It can be used by remote applications like an EDX system, or lithography system or other custom application.

The remote control interface is referred as SharkSEM Remote Control.

This document describes interface for DrawBeam module control. DrawBeam offers rich set of functions for electron beam and ion beam patterning. FIB systems are always equipped with DrawBeam module, other systems must have additional DrawBeam license.

For information about the SharkSEM protocol, refer to the main document. This document only describes the extension functions.

Using DrawBeam

There are only few control functions, which define the lithographic project and control the exposition process.

The project contains shape of exposition objects, as well as the parameters of the exposition process. Thus, the project file contains complete description of the whole exposition job. The project is encoded in XML.

The typical workflow is:

1. DrwLoadLayer() – load XML project
2. DrwStart() – start exposition
3. DrwGetStatus() – call repeatedly till the exposition is finished
4. DrwUnloadLayer() – discard the XML project

Currently, two process types are available, ion etching using FIB column and electron exposition using SEM column.

The exposition is restricted to single field and single exposition layer.

The XML project is not visible in DrawBeam GUI, the project storage is independent from the storage for the current GUI project.

DrawBeam Reference

DrwEstimateTime

Estimate exposition time.

Arguments

*int **DrwEstimateTime**(in int layer, out float total_time)*

layer	exposition layer (reserved, must be 0)
total_time	estimated exposition time [s]
return value	DrawBeam error code (Appendix A)

Timing

Milliseconds for simple projects, may take longer time if the exposition project is very complex (thousands of objects).

Remarks

The estimation accuracy is usually within few percent.

Call Context

XML layer must be loaded, exposition must be inactive.

Compatibility

1.x.x	2.x.x
no	2.0.10 and later

DrwGetConfig

Get DrawBeam configuration.

Arguments

*int **DrwGetConfig**(void)*

return value	0 – not installed 1 – offline version only 2 – DrawBeam Basic 3 – DrawBeam Advanced 4 – DrawBeam installed, but license is not valid

Timing

Executed immediately.

Remarks

This may be used to check the actual system configuration (installation status). If the code is 2 or 3, SharkSEM DrawBeam interface is functional. Otherwise, it cannot be used.

Call Context

Anytime.

Compatibility

1.x.x	**2.x.x**
no	2.0.10 and later

DrwGetStatus

Get DrawBeam processing status.

Arguments

*int **DrwGetStatus**(out float total_time, out float elapsed_time)*

total_time	total exposition time [s]
elapsed_time	exposition time elapsed [s]
return value	0 – project not loaded 1 – project loaded, exposition idle 2 – project loaded, exposition in progress 3 – project loaded, exposition paused

Timing

Executed immediately.

Remarks

The timing information is valid only if exposition is in progress. The total time is updated during the exposition process according to the latest progress information.

Call Context

Anytime.

Compatibility

1.x.x	**2.x.x**
no	2.0.10 and later

DrwLoadLayer

Load project (layer).

Arguments

*int **DrwLoadLayer**(in int layer, in char[] xml_project)*

layer	exposition layer (reserved, must be 0)
xml_project	XML project
return value	DrawBeam error code (Appendix A)

Timing

Executed immediately.

Remarks

The XML project is self-containing, no additional information is required. It must be written in valid XML, encoded in UTF-8.

This call breaks down the project into internal binary object list ant other structures. If there is an error in XML, error code is returned and more info is printed into the SharkSEM debugging console.

Structure of the XML file is described in Appendix B.

Call Context

DrawBeam project must not be loaded.

Compatibility

1.x.x	2.x.x
No	2.0.10 and later

DrwPause

Pause DrawBeam process.

Arguments

*int **DrwPause**(void)*

return value	DrawBeam error code (Appendix A)

Timing

Takes few tens of milliseconds till DrawBeam process is paused.

Remarks

Only I-Etching process can be paused. When paused, beam is blanked and scanning is stopped.

When paused, it is legal to eg. scan an image using *ScScanXY()*, modify some SEM or FIB parameters, etc. However, the pending exposition project cannot be modified in any way.

Call Context

Exposition process must be running.

Compatibility

1.x.x	**2.x.x**
no	2.0.10 and later

DrwResume

Resume paused DrawBeam process.

Arguments

*int **DrwResume**(void)*

return value	DrawBeam error code (Appendix A)

Timing

Depending on the project complexity, several milliseconds to several seconds.

Remarks

Internally, this procedure is very similar to *DrwStart()*. Ideally, the exposition is resumed exactly from the point where it was paused. Due to implementation limitations, the exact resume point may be slightly different.

Call Context

Exposition process must be paused.

Compatibility

1.x.x	**2.x.x**
no	2.0.10 and later

DrwStart

Resume paused DrawBeam process.

Arguments

*int **DrwStart**(in int layer)*

layer	exposition layer (reserved, must be 0)
return value	DrawBeam error code (Appendix A)

Timing

Depending on the project complexity, several milliseconds to several seconds.

Remarks

Exposition procedure is calculated and exposition is started. The progress can be then determined using *DrwGetStatus()*. Exposition can be stopped prematurely using *DrwStop()*. Some kinds of processes can be also paused using *DrwPuase()*.

Call Context

Project must be loaded.

Compatibility

1.x.x	**2.x.x**
no	2.0.10 and later

DrwStop

Stop DrawBeam process.

Arguments

*int **DrwStop**(void)*

return value	DrawBeam error code (Appendix A)

Timing

Takes few tens of milliseconds till DrawBeam process is stopped.

Remarks

This is only necessary if processing shall be stopped prematurely. If the process is finished in a standard way, there is no need to call *DrwStop()*.

Call Context

Exposition process must be either running or paused.

Compatibility

1.x.x	**2.x.x**
no	2.0.10 and later

DrwUnloadLayer

Throw away the current project (layer).

Arguments

*int **DrwUnloadLayer**(in int layer)*

layer	exposition layer (reserved, must be 0)
return value	DrawBeam error code (Appendix A)

Timing

Executed immediately.

Remarks

This must be called after the exposition is completed to clean up the internal structures.

Call Context

Exposition must be inactive.

Compatibility

1.x.x	**2.x.x**
no	2.0.10 and later

Appendix A – Error Codes

Code	Value	Description
`OK`	0	Success
`INVALID_REQ`	-1	Request is not valid in the current context
`GIS_NOT_READY`	-2	GIS not ready
`MEMORY`	-3	Out of memory
`VISIBILITY`	-4	Some objects are outside the write field
`LICENSE`	-5	The DrawBeam license is not valid
`EMISSION`	-6	Emission is off
`VIEW_FIELD`	-7	The specified view field is too large
`UNKNOWN`	-100	Error, the exact reason is not known

Appendix B – XML Project

Example E-Exposition Project

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<Layer Name="Abcd" Version="1.0">
      <Settings WriteFieldSize="0.001000" Process="E-Exposition" Dose="3.0"
            Accuracy="Fine" Spacing="1.0" BeamCurrent="1.0e-9" />
      <Object Type="RectangleOutline" Center="0.0 0.0" Width="0.000180"
            Height="0.000120" Angle="0.0" Depth="1" DepthUnit="scan"
            ExpositionFactor="1.0" />
      <Object Type="RectangleFilled" Center="0.0 0.0" Width="0.000180"
            Height="0.000120" Angle="0.0" ScanningPath="0" Depth="1"
            DepthUnit="scan" ExpositionFactor="1.0" />
      <Object Type="PolygonFilled" Center="0.0 0.0" Angle="0.0"
            ScanAngle="0.0" ScanningPath="0" Depth="1" DepthUnit="scan"
            ExpositionFactor="1.0">
            <Vertex Position="-0.0001 -0.0001" />
            <Vertex Position="0.0001 -0.0001" />
            <Vertex Position="0.0001 0.0001" />
            <Vertex Position="-0.0001 0.0001" />
      </Object>
</Layer>
```

Example I-Etching Project

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<Layer Name="Abcd" Version="1.0">
      <Settings WriteFieldSize="0.000250" Process="I-Etching"
            Parallel="0" Rate="3.0e-3" DwellTime="1.0e-6"
            Accuracy="Fine" SpotSize="5.0e-8" Spacing="1.0"
            BeamCurrent="1.0e-9" />
      <Object Type="RectangleFilled" Center="0.0 0.0" Width="0.000180"
            Height="0.000120" Angle="0.0" ScanningPath="0" Depth="5.0e-6"
            DepthUnit="m" ExpositionFactor="1.0" />
      <Object Type="RectanglePolish" Center="0.0 0.0" Width="0.000180"
            Height="0.000120" Angle="0.0" ScanningPath="0" Depth="5.0e-6"
            DepthUnit="m" ExpositionFactor="1.0" />
</Layer>
```

XML Project Structure

`<?xml version="1.0" encoding="utf-8" standalone="yes"?>`

Each file must start with XML declaration. Recommended encoding is UTF-8.

```
<Layer Name="Abcd" Version="1.0">
…
</Layer>
```

This is the top-level (root) tag. `Name` is mandatory and allows specifying the layer name. `Version` is mandatory as well, the current version is 1.0.

```
<Settings WriteFieldSize="0.001000" Process="E-Exposition" Dose="3.0"
Accuracy="Fine" Spacing="1.0" BeamCurrent="1.0e-9" />
```

The `Settings` tag contains attributes with the exposition process parameters. There must be exactly one `Settings` tag. Parameters are described later in this document.

```
<Object Type="RectangleOutline" Center="0.0 0.0" Width="0.000180"
Height="0.000120" Angle="0.0" Depth="1" DepthUnit="scan"
ExpositionFactor="1.0" />
```

Following `Settings`, there is a list of `Object` tags. There should be at least one object, the number of objects is unlimited. The objects are processed sequentially. Most objects are just single tags with attributes, but for example polygon has also a list of child elements which contain coordinates of the vertices.

E-Exposition Settings

Attribute	Mandatory	Description
`Process`	yes	`E-Exposition` must be specified.
`WriteFieldSize`	no	Write field size in meters. Default value is 0.001 m.
`Accuracy`	no	Exposition mesh accuracy. Default is `Fine`. Available options are `Coarse`, `Medium`, `Fine`, `ExtraFine`.
`BeamCurrent`	no	Electron beam current in Amps. Default is 1 nA.
`Spacing`	no	Exposition mesh spacing. Default is 1.0.
`Dose`	no	Resist exposition dose in C/ m^2. Default is 2.0 C/m^2.

The `Accuracy` specifies rasterization precision of the generator. If the accuracy is very fine, DrawBeam has higher memory consumption and slower process start.

If `Spacing` is 1.0, the pitch of the discrete exposition points is same as the beam spot size. If it is different than 1.0, pitch distance is calculated as a product of spot size and spacing.

Spot size is calculated by the SEM software. Pixel dwell time is automatically calculated from spot size, spacing and exposition dose. The default behavior is that each object is exposed in a single pass.

I-Etching Settings

Attribute	Mandatory	Description
`Process`	yes	`I-Etching` must be specified.
`WriteFieldSize`	no	Write field size in meters. Default value is 0.001 m.
`Accuracy`	no	Exposition mesh accuracy. Default is `Fine`. Available options are `Coarse`, `Medium`, `Fine`, `ExtraFine`.
`BeamCurrent`	no	Ion beam current in Amps. Default is 1 nA.
`Spacing`	no	Exposition mesh spacing. Default is 1.0.

`SpotSize`	no	Ion beam spot size in meters. Default is 50 nm.
`Parallel`	no	Parallel processing flag. Default is `0`. Available options are `0` (serial etching) or `1` (parallel etching).
`Rate`	no	Ion etching rate in m^3/A/s. Default is 0.3 μm^3/nA/s.
`DwellTime`	no	Pixel dwell time in seconds. Default is 1 μs.

`Accuracy` and `Spacing` behave same way as for E-Exposition process.

Spot size is not calculated by the FIB optics, but it should be rather specified. Pixel dwell time should be specified as well. From all known parameters, number of scans is calculated.

Objects and Their Attributes

The exposition view field geometry is two-dimensional, Cartesian coordinates. The center of the field has coordinates x=0.0, y=0.0.

Each object is defined in a way like this:

```
<Object Type="RectangleOutline" Center="0.0 0.0" Width="0.000180"
Height="0.000120" Angle="0.0" Depth="1" DepthUnit="scan"
ExpositionFactor="1.0" />
```

`Object` is the tag name. `Type` attribute specifies the object type. Each object has arbitrary attribute list.

Available objects

`RectangleOutline` – outline rectangle. It is scanned from bottom left corner, counterclockwise.

`RectangleFilled` – filled rectangle. Scanning starts in the bottom left corner, from left to right.

`RectanglePolish` – polishing rectangle. Only suitable for I-Etching. To achieve the required depth, scanning is not repeated for the whole object, but rather for each scan line. Scanning starts in the bottom left corner, from left to right.

`PolygonOutline` – outline polygon.

`PolygonFilled` – filled polygon. During preprocessing, polygon is sliced into regular thin lines (slices) and exposed. The slicing goes from bottom up, but this behavior can be modified by `ScanAngle` parameter.

`ArcOutline` – outline circle or arc.

`AnnulusFilled` – filled circle, annulus or pie. Scanned from outside to the center.

`AnnulusPolish` – polishing circle, annulus or pie. The typical use of this object is polishing of thin pillars.

Attributes

Type	DepthUnit	Depth	Center	Width	Height	Angle	Radius	RadiusA	RadiusB	AngleA	AngleB	Orientation	ScanAngle	ExpositionFactor	LineSettleFactor	ScanningPath	*Vertex list*
RectangleOutline	●	●	●	●	●	●	○	○	○	○	○	○	○	●	●	○	○
RectangleFilled	●	●	●	●	●	●	○	○	○	○	○	○	○	●	●	●	○
RectanglePolish	●	●	●	●	●	●	○	○	○	○	○	○	○	●	●	●	○
PolygonOutline	●	●	●	○	○	●	○	○	○	○	○	○	●	●	●	○	●
PolygonFilled	●	●	●	○	○	●	○	○	○	○	○	○	●	●	●	●	●
ArcOutline	●	●	●	○	○	○	●	○	○	●	●	●	○	●	●	○	○
AnnulusFilled	●	●	●	○	○	○	○	●	●	●	●	●	○	●	●	○	○
AnnulusPolish	●	●	●	○	○	○	○	●	●	●	●	●	○	●	●	○	○

DepthUnit – either **scan** (number of scans) or **meter** (metric definition). Optional, default unit is meter.

Depth – depth of the objects, works in conjunction with **DepthUnit**. Mandatory attribute.

Center – center of the object. There are two values (x, y) separated by whitespace. The unit is meter. Mandatory attribute. For polygon, this specifies the reference point for the polygon rotation.

Width – width of a rectangular object in meters. Mandatory attribute.

Height – height of a rectangular object in meters. Mandatory attribute.

Angle – rotation of the object in degrees (counterclockwise). Optional, default angle is zero degrees.

Radius – radius of a circle in meters. Mandatory attribute.

RadiusA – outer radius of an annulus, in meters. Mandatory attribute.

RadiusB – inner radius of an annulus, in meters. Mandatory attribute.

AngleA – pie start angle of circular objects, in degrees. Optional, default is zero degrees.

`AngleB` – pie end angle of circular objects, in degrees. Optional, default is zero degrees.

`Orientation` – pie orientation flag, whether it is drawn clockwise (**`1`**) or counterclockwise (**`0`**). Optional, default is counterclockwise.

`ScanAngle` – filled polygon vectorization angle, in degrees. 0.0 deg means that the polygon is sliced from bottom up, 90.0 deg means slicing from right to left, etc. Optional, default is zero degrees.

`ExpositionFactor` – multiplication factor of the exposition dose. In case of I-Etching, exposition factor modifies the scan count. In case of E-Exposition, exposition factor modifies the pixel dwell time. Optional, default is 1.0.

`LineSettleFactor` – multiplication factor of the settle time. The settle time is time for the beam stabilization after flyback. Optional, default is 1.0.

`ScanningPath` – scanning strategy flag. **`0`** – scan with flyback, **`1`** – zig-zag scanning. Optional, default is 0.

`Vertex list` – list of polygon vertices. Since number of polygon vertices is variable, this is not an XML attribute, but rather list of child elements. Each vertex is described by **`Vertex`** tag, its only attribute is **`Position`**. The **`Position`** consists of two, whitespace separated values, which specify the point coordinates in meters.

```
<Object Type="PolygonFilled" …>
      <Vertex Position="-0.0001 -0.0001" />
      <Vertex Position="0.0001 -0.0001" />
      <Vertex Position="0.0001 0.0001" />
      <Vertex Position="-0.0001 0.0001" />
</Object>
```