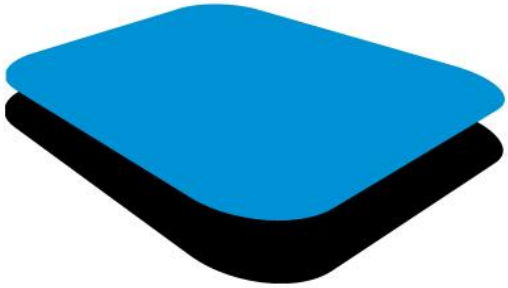


lyncée tec

Koala Remote

Users Manual

Version 8.6 - November 2023

Versions and modifications of this manual

Version	Date	Main modifications
8.0.x	March 2019	First version of this manual
8.1	September 2019	Small fixes
8.2	September 2020	LynceeTec.ToolBox library replaced by LynceeTec.LogTools library Difference in GetIntensity32fImage and GetPhase32fImage when a mask is active
8.4	January 2022	Added new functions for Stroboscope, and renamed RecordStroboFixFrequency and RecordStroboFrequencyScan
8.4	April 2022	Added new functions for phase correction AddCorrZone and ResetCorrZone
8.4	May 2022	Added script examples for C#
8.5	July 2022	Added phase offset adjustment zone functions, and wavelength filter functions.
8.5	August 2022	Added open, close and save reconstruction settings functions.
8.5	March 2023	Added filter functions for mask window, for reconstruction to disk, fast sequence acquisition, intensity profile, stroboscopic frequency approach.
8.5	May 2023	Added missing function description for SetFastHologramsSequenceRecordPeriodicallyEveryXHologram
8.6	October 2023	Added functions to set and reset the reconstruction ROI and to do a lateral scanning acquisition
8.6	November 2023	Added functions to realize an acquisition in spiral (StartSpiralScanning) or with given points (StartScanningFromPoints)

Table of contents

Introduction

The *Remote Module* (also known as *Production Mode*) allows sending commands to the Koala interface from a separate user-defined client program, executing at the same time as Koala.

The remote module is optional, and is usually sold separately.

To use the remote module, it must have been enabled in the factory settings of your system, and the remote option must have been selected at installation. If you did not purchase the module, it is always possible to buy and activate this option at a later point.

The remote commands communication takes place over a TCP-IP connection.

Koala side (server side)

The Koala software acts as a server for the remote commands.

For more details on the *Remote Module* from the Koala side, please see the *Controlling Koala via the Remote interface* section in the *Koala Manual*

Client side

There are two options for the client-side, either use the *Koala Remote Test App* (see the specific manual), or implement your own client. See section *How to implement your own remote client application* below.

How to implement your own remote client application

References

Lyncée Tec provides .NET libraries to help you implement your own client without the need to implement the details of the TCP-IP communication protocol.

The necessary libraries and configuration files are located in *C:\Program Files\LynceeTec\Koala\Remote\RemoteLibraires* (make sure you take the correct processor version, x64 or x86, according to your system and application).

File name	Description
log4net.dll	3rd party library for logging. See the *Troubleshooting your application with the logs* section below.
LynceeTec.KoalaRemote.Client.dll	Utility for the implementation of a remote client application. This is the main library that you will have to reference in your code to implement your client application. See the *Implementation* section below.
LynceeTec.KoalaRemote.dll	Base definitions for the remote protocol
LynceeTec.LogTools.dll	Utility library for logging
messages.xml	Signed list of remote commands

The reference libraries and files must all be copied next to your application executable even if you do not directly refer to them in your code.

The Lyncée Tec reference libraries version number must always be the same as the version number of your Koala program.

Implementation

Your application needs to create an instance of the `LynceeTec.KoalaRemote.Client.KoalaRemoteClient` class which is defined in the *LynceeTec.KoalaRemote.Client.dll* library, using the parameterless constructor as follows:

```
KoalaRemoteClient m_client = new KoalaRemoteClient();
```

Through this instance you can call all the remote functions described in the *The remote functions* section of this manual.

Your application must first call the *Connect* function, followed by the *Login* function.

```
string userName = "";
bool success = m_client.Connect("localhost", out userName, true);
success = m_client.Login(password);
```

You can then call any remote function.

```
m_client.OpenConfig(137);
```

When closing, your application should call *Logout*, to notify Koala that it is disconnecting.

```
m_client.Logout();
```

For more details, please also refer to the provided examples for *Python*, *LabView* and *Matlab*. You can find a copy of the Python example in the Python remote client application example section of this manual.

Error handling

In case of an error due to a remote command, the command function throws an exception.

If the error is directly linked to the remote command, the exception will be of type `LynceeTec.KoalaRemote.Client.KoalaRemoteFunctionException` and will contain the following information:

- ErrorCode (RemoteReturnCodes) : The remote command return code.
- ErrorMessage (String): The error message
- FunctionName (String): The name of the remote command which failed

The remote return codes

The remote return codes are defined in the `LynceeTec.KoalaRemote.RemoteReturnCodes` enumeration in the *LynceeTec.KoalaRemote.dll*. The following table describes the possible values:

Error code	Error Name	Definition
-1000	*Unknown*	Unknown result. Should not happen. If this happens, please contact your Lyncée Tec support.
-6	*SystemBusy*	The command could not be executed because another one is executing
-5	*ExecutionFailed*	The command started execution, but an error occurred

-4	*MessageNotFound*	The command sent a message unknown in the remote protocol. Should not happen if you use the libraries corresponding to your current version of Koala
-3	*InvalidOperation*	The command was not executed because Koala is not in a state where it's execution was possible
-2	*TargetHandlerNotFound*	Deprecated. Handler object necessary for the command execution does not exist.
-1	*TargetObjectNotFound*	Deprecated. Object necessary for the command execution does not exist.
0	*OK*	The command executed successfully.
1	*LongAsyncOperation*	Deprecated. The command was not executed because a previous asynchronous command is still executing. In the current version, every remote command is synchronous and blocking until execution is finished.

Obviously, no exception is thrown when the result is *OK*.

In a standard situation, your code should only receive `KoalaRemoteFunctionException` with return codes of value *SystemBusy*, *ExecutionFailed* and *InvalidOperation*.

Troubleshooting your application with the logs

The Koala log

The Koala log can be very helpful to debug your remote client application and understand what happened in case of an error.

The Koala log is located in *C:\Users*`{userName}`*\AppData\Local\LynceeTec\Koala*, where `{userName}` is your Windows login name. It can also directly be displayed in Koala, via the *Help* ⟶ *Show Log* menu.

Adding a log4net log in your client application

All Lyncée Tec libraries use *log4net* for logging. We recommend that you implement logging with *log4net* in your client application as well. By doing that, your client application log will automatically contain the log entries from the *LynceeTec.KoalaRemote.Client.dll* and *LynceeTec.KoalaRemote.dll* reference libraries.

The *log4net* library is included in the set of Lyncée Tec remote reference libraries (see the *References* section above). It is also available as a standard NuGet package on nuget.org.

How to update your client application to make it compatible with Koala 8.x

Some libraries have been renamed in the Koala versions 8.x and additional libraries are now mandatory.

To update your application, you need to update the reference libraries used by your application and rename their reference in your code.

Update the reference libraries used by you application

In your application reference folder, remove the *KoalaRemoteBase.dll*, *TCPClient.dll* and *message.xml files*.

Replace them by the new references located in *C:\Program Files\LynceeTec\Koala\Remote\RemoteLibraires* (make sure you take the correct processor version, x64 or x86, according to your system and application):

- *log4net.dll* (version 2.0.8.0)
- *LynceeTec.KoalaRemote.Client.dll* (same version as your current Koala)
- *LynceeTec.KoalaRemote.dll* (same version as your current Koala)
- *LynceeTec.LogTools.dll* (same version as your current Koala, replaces the *ToolBox* library)
- *messages.xml*

Update the references to the Lyncée Tec library in your code

Modify your code references according to the following table:

	Old	**New (version >=8.0)**
Library name	TCPClient.dll	LynceeTec.KoalaRemote.Client.dll
namespace	KoalaClient	LynceeTec.KoalaRemote.Client
Class name	KoalaTCPClient	KoalaRemoteClient

Example in Python:

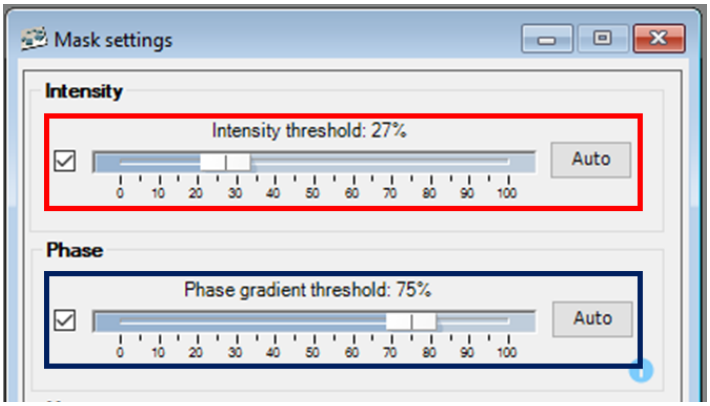
Old code	**New code**
`import clr`	`import clr`
`clr.AddReference("TCPClient")`	`clr.AddReference("LynceeTec.KoalaRemote.Client")`
`import KoalaClient`	`from LynceeTec.KoalaRemote.Client import KoalaRemoteClient`
`host = KoalaClient.KoalaTCPClient()`	`host = KoalaRemoteClient()`
`host.OpenConfig(configNumber)`	`host.OpenConfig(configNumber)`

The remote command interfaces have not changed, the rest of your code should remain compatible.

The remote functions by theme

Mask

- *CloseMaskSettingsWin*
- *OpenMaskSettingsWin*

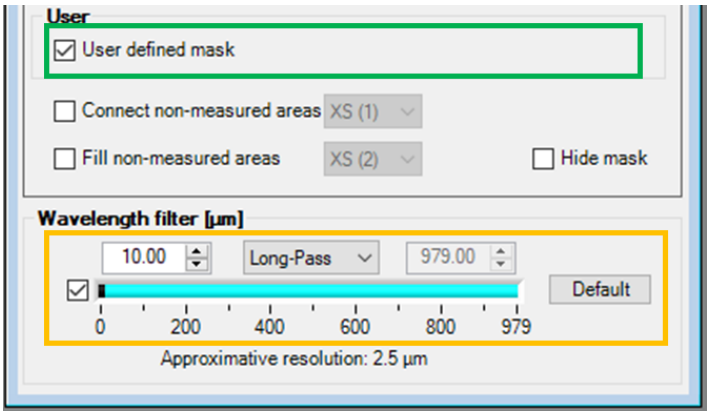


Intensity threshold Filter

- *SetAutomaticIntensityThresholdFilterToEnabledState*
- *SetIntensityThresholdFilterState*
- *SetIntensityThresholdFilterValueInPercent*

Phase gradient threshold filter

- *SetAutomaticPhaseGradientThresholdFilterToEnabledState*
- *SetPhaseGradientThresholdFilterState*
- *SetPhaseGradientThresholdFilterValueInPercent*



User defined Mask

- *AddEllipticalUserDefinedMaskZoneToPhase*
- *AddRectangularUserDefinedMaskZoneToPhase*
- *ResetUserDefinedMaskZone*
- *SetInteractionModeWhenAddingMaskZone*
- *SetUserDefinedMaskState*

Wavelength filter

- *SetWavelengthFilterMinimalCutOffValue*
- *SetWavelengthFilterMaximalCutOffValue*
- *SetWavelengthFilterState*
- *SetWavelengthFilterType*

Profile

Intensity

- *ExtractIntensityProfile*
- *GetIntensityProfile*
- *GetIntensityProfileLength*
- *SetIntensityProfileState*

Phase

- *ExtractPhaseProfile*
- *GetPhaseProfile*
- *GetPhaseProfileLength*
- *SetPhaseProfileState*

Sequence

Fast Holograms Record

- *CloseFastHologramsRecordWin*
- *OpenFastHologramsRecordWin*
- *SetFastHologramsSequenceRecordNumberOfHolograms*
- *SetFastHologramsSequenceRecordingModeBuffer*
- *SetFastHologramsSequenceRecordPath*
- *SetFastHologramsSequenceRecordPeriodicallyEveryXHologram*
- *StartFastHologramsSequenceRecord*
- *StopFastHologramsSequenceRecord*

Reconstruction to disk

- *CloseReconstructionToDiskSequenceWin*
- *OpenReconstructionToDiskSequenceWin*
- *SetReconstructionToDiskDataType*
- *SetReconstructionToDiskSequencePath*
- *StartReconstructionToDisk*

The remote functions

AccWDSearch

Performs an accurate working distance search.
Note that this function is blocking, like all remote functions, and might take several minutes, depending on the chosen parameters.

Koala UI equivalent:
Accurate working distance search in the *Working Distance window*. Open the working distance window, via the *Tools* ⟶ *Working Distance* menu.

Parameters:

- distUM (Int32): Full range of the search, in [um].
- stepUM (Double): Size of a step between two working distance measurements, in [um].

Return: void (nothing)

Requirements:

- The stage must be available and the axes must be initialized
- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

See also:

- *FastWDSearch*

Acquisition2L

Acquires an image (or several if temporal averaging is on) and reconstruct it.

Koala UI equivalent:
(No exact Koala UI equivalent)

(No parameters)

Return: void (nothing)

Requirements:

- A configuration must be loaded
- Live mode must be available
- The intern camera must be available
- Video mode must not be running
- A strobo measurement must not be running
- The sequence reconstruction to display window must not be opened
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

See also:

- *SingleReconstruction*

AddCorrSegment

Adds a phase correction segment (for 1D tilt correction)

Note: The start and end points can be outside the phase ROI, but it is recommended to avoid it.

Koala UI equivalent:
Tracing a phase correction segment in the *Phase Image window*, when the *Horizontal and Vertical profiles for phase mask adjustment* button  is activated.

Parameters:

- top (Int32): Y coordinate of the top left point of the segment, in pixel
- left (Int32): X coordinate of the top left point of the segment, in pixel
- length (Int32): Length of the segment, in pixel
- Orientation (Int32): Orientation of the segment: 0 = horizontal, 1 = vertical

Return: void (nothing)

Requirements:

- A configuration must be loaded
- The phase window must be opened and must contain an image
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

See also:

- AddCorrZone
- *AlgoResetPhaseMask*
- *ComputePhaseCorrection*
- *ResetCorrSegment*
- *ResetCorrZone*

AddCorrZone

Adds a phase correction zone (for 2D tilt correction)

Koala UI equivalent:
Tracing a phase correction zone in the *Phase Image window*, when the *Horizontal and Vertical profiles for phase mask adjustment* button is activated.

Parameters:

- top (Int32): Y coordinate of the top left point of the zone, in pixel
- left (Int32): X coordinate of the top left point of the zone, in pixel
- Width (Int32): Width of the zone, in pixel
- Height (Int32): Height of the zone, in pixel

Return: void (nothing)

Requirements:

- A configuration must be loaded
- The phase window must be opened and must contain an image
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

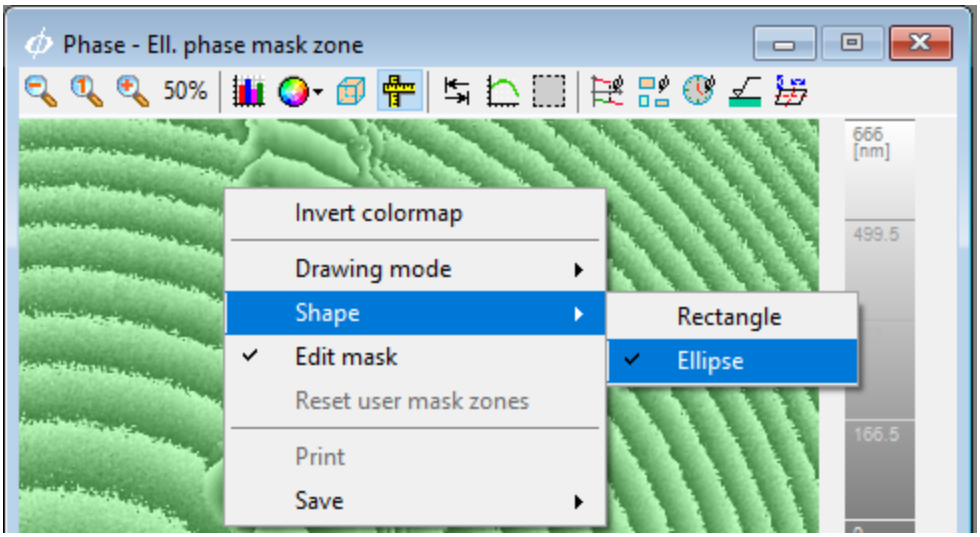
See also:

- *AddCorrSegment*
- *AlgoResetPhaseMask*
- *ComputePhaseCorrection*
- *ResetCorrSegment*
- *ResetCorrZone*

AddEllipticalUserDefinedMaskZoneToPhase

Adds an elliptical user defined mask zone on the Phase.

Koala UI equivalent:
Tracing an elliptical user defined mask zone in the *Phase Image window*, when the **Edit mask** is activated (right-click to access the drop-down menu), and the **Shape** selected is **Ellipse**.



Parameters:

- centerX (Int32): X coordinate in the phase image of the center point of the zone, in pixel
- centerY (Int32): Y coordinate in the phase image of the center point of the zone, in pixel
- radiusX (Int32): Radius in the X-direction of the zone, in pixel
- radiusY (Int32): Radius in the Y-direction of the zone, in pixel

Return: void (nothing)

Requirements:

- A configuration must be loaded
- The phase window must be opened and must contain an image
- A reconstruction object must be created
- A user must be logged in
- The mask window must be opened

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

See also:

- *AddRectangularUserDefinedMaskZoneToPhase*
- *CloseMaskSettingsWin*
- *OpenMaskSettingsWin*

- *ResetUserDefinedMaskZone*
- *SetInteractionModeWhenAddingMaskZone*
- *SetUserDefinedMaskState*

AddPhaseOffsetAdjustmentZone

Adds a phase offset adjustment zone

Koala UI equivalent:
Tracing a phase offset adjustment zone in the *Phase Image window*, when the *phase offset adjustment zone* button  is activated.

Parameters:

- top (Int32): Y coordinate of the top left point of the zone, in pixel
- left (Int32): X coordinate of the top left point of the zone, in pixel
- Width (Int32): Width of the zone, in pixel
- Height (Int32): Height of the zone, in pixel

Return: void (nothing)

Requirements:

- A configuration must be loaded
- The phase window must be opened and must contain an image
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

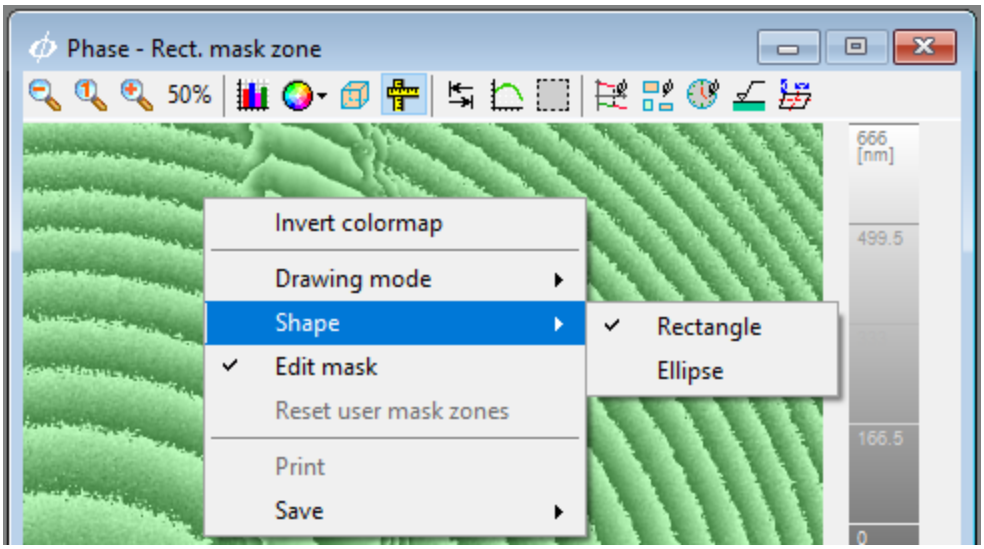
See also:

- *ResetPhaseOffsetAdjustmentZone*

AddRectangularUserDefinedMaskZoneToPhase

Adds a rectangular user defined mask zone on the Phase.

Koala UI equivalent:
Tracing a rectangular user defined mask zone in the *Phase Image window*, when the **Edit mask** is activated (right-click to access the drop-down menu), and the **Shape** selected is **Rectangle**.



Parameters:

- top (Int32): Y coordinate in the phase image of the top left point of the zone, in pixel
- left (Int32): X coordinate in the phase image of the top left point of the zone, in pixel
- Width (Int32): Width of the zone, in pixel
- Height (Int32): Height of the zone, in pixel

Return: void (nothing)

Requirements:

- A configuration must be loaded
- The phase window must be opened and must contain an image
- A reconstruction object must be created
- A user must be logged in
- The mask window must be opened

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

See also:

- *AddEllipticalUserDefinedMaskZoneToPhase*
- *CloseMaskSettingsWin*
- *OpenMaskSettingsWin*

- *ResetUserDefinedMaskZone*
- *SetInteractionModeWhenAddingMaskZone*
- *SetUserDefinedMaskState*

AlgoResetPhaseMask

Resets the current user mask for the phase tilt correction to the values saved in the database.

Koala UI equivalent:
Click on the *Reset mask* button in the *Basic settings* tab of the *Reconstruction settings window*.

(No parameters)

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

See also:

- *AddCorrSegment*
- *AddCorrZone*
- *ComputePhaseCorrection*
- *ResetCorrSegment*
- *ResetCorrZone*

AxisInstalled

Gets a value indicating if an axis is installed for the current stage.

Koala UI equivalent:
(Not applicable)

Parameters:

- axisId (Int32): Id of the axis to check. Possible values are
  - 0: X axis
  - 1: Y axis
  - 2: Z axis
  - 3: Theta axis (rotation, system-dependent)
  - 4: Phi axis (rotation, system-dependent)
  - 5: Psi axis (rotation, system-dependent)

Return: Boolean

Requirements:

- A stage must be defined
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

See also:

- -

ComputePhaseCorrection

Computes the 1D phase correction using a specific method.
Note that if the *Reconstruction Settings* window is not opened, the function will open it automatically.

Koala UI equivalent:
Click on the *Perform fit* button in the *Basic settings* tab of the *Reconstruction settings window*.

Parameters:

- fitMethod (Int32): Method used to compute the phase correction. Possible values are:
  - 0: Tilt. In this case, *degree* must be equal to 1
  - 1: Polynomial.
  - 4: Polynomial 2D
- degree (Int32): Degree of the correction. Value must be a positive non-zero integer and can only be 1 *fitMethod* is *Tilt*.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A reconstruction object must be created
- The phase window must be opened and must contain an image
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*
- Incoherent parameters ⇒ *Invalid Operation*

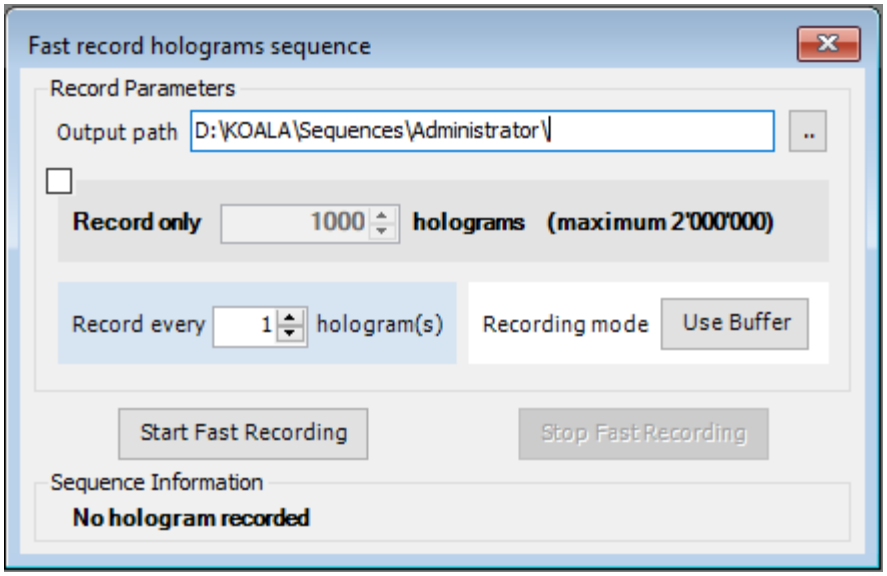
See also:

- *AddCorrSegment*
- *AddCorrZone*
- *AlgoResetPhaseMask*
- *ResetCorrSegment*
- *ResetCorrZone*

CloseFastHologramsRecordWin

Close the *Fast record holograms sequence* window

Koala UI equivalent:
Corresponds to closing the Fast record holograms sequence window with the top right red cross.



(No parameters)

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in
- The window must be opened.

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *OpenFastHologramsRecordWin*

CloseIntensityWin

Close the intensity (amplitude) window

Koala UI equivalent:

Corresponds to clicking on the *Intensity window* button  in the toolbar, when the intensity window is opened.

Parameters:

- None.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in
- The Intensity window must be opened.

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *OpenIntensityWin*

CloseMaskSettingsWin

Close the *Mask Settings* window

Koala UI equivalent:
Corresponds to unchecking on the *Mask* option in the *Settings* menu.

(No parameters)

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in
- The Mask settings window must be opened

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *OpenMaskSettingsWin*

ClosePhaseWin

Close the phase window.

Koala UI equivalent:

Corresponds to clicking on the *Phase window* button  in the toolbar, when the phase window is opened.

Parameters:

- None.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in
- The Phase window must be opened.

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *OpenPhaseWin*

CloseReconstructionSettingsWin

Close the *Reconstruction Settings* window

Koala UI equivalent:
Corresponds to unchecking on the *Reconstruction* option in the *Settings* menu.

(No parameters)

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

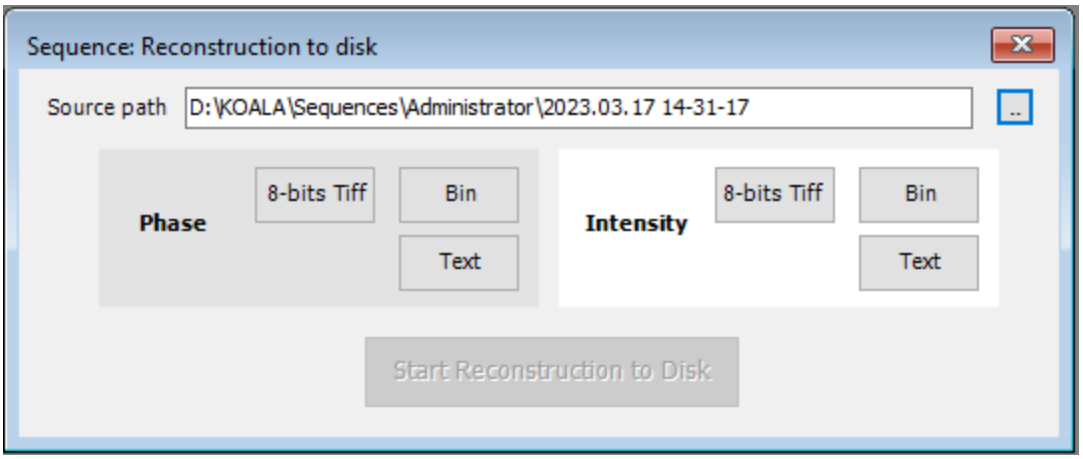
See also:

- *OpenReconstructionSettingsWin*

CloseReconstructionToDiskSequenceWin

Closes the reconstruction settings window

Koala UI equivalent:
Corresponds to clicking on the red cross in the ***Sequence: Reconstruction to Disk*** window, or unticking the ***Sequence → Reconstruction → To disk*** option in the ***Mode*** menu.



(No parameters)

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in
- The reconstruction to disk window must be already opened.

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *OpenHoloWin*
- *OpenIntensityWin*
- *OpenPhaseWin*
- *OpenReconstructionToDiskSequenceWin*
- *OpenStroboWin*

Connect

Connects the client remote application to the Koala remote server.
Note that connection and login is handled automatically in the *Koala Remote Test App*. It is only needed if you implement your own client application.

Koala UI equivalent:
(not applicable)

Parameters:

- hostName (String): The IP of the computer where Koala is running. Use `localhost` if running on the same computer
- userName (String): Output parameter returning the name of the user currently logged in Koala
- quiet (Boolean): Deprecated parameter, will be removed in a later version. Set either to `true` or `false`

Return: `true` if the connection was successful.

Requirements:

- A user must be logged in

Possible errors:

- The *Remote Log Message window* is not opened or needs to be reopened because of a previous login failure

See also:

- *Login*
- *Logout*

DigitizerAcquiring

Gets a value indicating if the camera is currently in continuous grab mode or not.

Koala UI equivalent:
(not applicable)

(No parameters)

Return: Boolean: `true` if the camera is in continuous grab mode

Requirements:

- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

See also:

- *ResetGrab*

ExtractIntensityProfile

Extracts a profile from the intensity image and plots it in the intensity profile window.

Note: The start and end points can be outside the intensity ROI, but it is recommended to avoid it.

Koala UI equivalent:

Tracing a profile in the *Intensity Image window*, when the *Draw a profile line* button  is activated.

Parameters:

- startX (Int32): X coordinate of the starting point of the profile in the intensity ROI
- startY (Int32): Y coordinate of the starting point of the profile in the intensity ROI
- endX (Int32): X coordinate of the ending point of the profile in the intensity ROI
- endY (Int32): Y coordinate of the ending point of the profile in the intensity ROI

Return: void (nothing)

Example: (See example for *GetIntensityProfile*)

Requirements:

- A configuration must be loaded
- The intensity window must be opened and must contain an image
- The intensity profile window must be opened
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

See also:

- *GetIntensityProfile*
- *GetIntensityProfileLength*
- *SetIntensityProfileState*

ExtractPhaseProfile

Extracts a profile from the phase image and plots it in the phase profile window.

Note: The start and end points can be outside the phase ROI, but it is recommended to avoid it.

Koala UI equivalent:

Tracing a profile in the *Phase Image window*, when the *Draw a profile line* button  is activated.

Parameters:

- startX (Int32): X coordinate of the starting point of the profile in the phase ROI
- startY (Int32): Y coordinate of the starting point of the profile in the phase ROI
- endX (Int32): X coordinate of the ending point of the profile in the phase ROI
- endY (Int32): Y coordinate of the ending point of the profile in the phase ROI

Return: void (nothing)

Example: (See example for *GetPhaseProfile*)

Requirements:

- A configuration must be loaded
- The phase window must be opened and must contain an image
- The phase profile window must be opened
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

See also:

- *GetPhaseProfile*
- *GetPhaseProfileLength*
- *SetPhaseProfileState*

ExtUIGoodBye

This function is documented for backward compatibility, but it should be replaced by Logout.
This function notifies the server side that a client application is closing, and it should close the connection.
This function is not blocking and will return before execution is finished without waiting for an answer.
After a call to *ExtUIGoodBye*, it is possible to call *Connect* and then *Login* again, without the need to restart the *Remote Message Log window* in Koala.
Note that disconnection is handled automatically in the *Koala Remote Test App*. It is only needed if you implement your own client application.

Koala UI equivalent:
(not applicable)

(No parameters)

Return: void (nothing)

Requirements:

- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

See also:

- *Connect*
- *Login*
- *Logout*

FastWDSearch

Performs a fast working distance search.
Note that this function is blocking, like all remote functions, and might take several seconds.

Koala UI equivalent:
Fast working distance search in the *Working Distance window*. Open the working distance window, via the *Tools* ⟶ *Working Distance* menu.

(no parameters)

Return: void (nothing)

Requirements:

- The stage must be available and the axes must be initialized
- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

See also:

- *AccWDSearch*

GetAxesPosMu

Copies the current positions of the stage axes, in [um], in the given memory location.

Koala UI equivalent:
(not applicable)

Parameters:

- buffer (Double[]): Array of doubles of size 4, where the data for the X, Y, Z and Theta axis respectively will be copied

Return: void (nothing)

Requirements:

- A stage must be available and initialized
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error
- The buffer array does not have the correct dimension ⇒ this will not result in an error on the server (Koala) side, but on the client application.

See also:

- *MoveAxis*
- *MoveAxes*
- *MoveAxesArr*

GetCameraShutterUs

Get the value of the camera shutter parameter, in [us].

Koala UI equivalent:
(Not applicable, but the current shutter value is displayed in the *Camera Settings window*, accessible via the *Settings / Camera* menu, or the *Camera configurations* button )

(No parameters)

Return: Int32: The shutter value, in [us]

Requirements:

- The intern camera must be available
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *SetCameraShutterUs*

GetChosenOPLPosition

Gets the non-corrected position of the OPL1 or OPL2 motor in [qc], depending on oplId (1 or 2).

Koala UI equivalent:
(not applicable)

Parameters:

- oplId (Int32): The OPL ID (1 or 2).

Return: Int32: The position of the OPL1 or OPL2 motor, in [qc]

Requirements:

- The OPL1 or OPL2 option must be enabled
- The OPL1 or OPL2 motor must be available and initialized
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

See also:

- *MoveOPL*
- *MoveChosenOPLToPosition*
- *GetOPLPos*

GetDHMSerial

Gets the numerical part of the serial number of the DHM device.

Koala UI equivalent:
(Not applicable, but the full serial number is displayed at the bottom right of the Koala main window, in the status bar.)

(No parameters)

Return: Int32

Requirements:

- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- -

GetHoloContrast

Gets a value representing the contrast of the last hologram (either last one grabbed or last loaded from disk, whichever occurred last) in an area half the size of the hologram, centered in the middle of the image.

Koala UI equivalent:
(Not applicable, but the hologram contrast is displayed in the *Hologram Histogram window*, which can be opened by clicking the *Draw a hologram*  button in the *Hologram Image window*.)

(No parameters)

Return: Double

Requirements:

- A configuration must be loaded
- An hologram must have been grabed since program start
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- -

GetHoloHeight

Gets the height of the hologram image, according to the current configuration.

Koala UI equivalent:
(not applicable)

(No parameters)

Return: Int32

Requirements:

- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *GetHoloWidth*

GetHoloImage

Copies the current hologram image in the given memory location.

Koala UI equivalent:
(not applicable)

Parameters:

- buffer (*Byte[]*): Array of bytes of dimension `stride*height` where the data will be copied. (See the *GetPhaseImage* function for an example of how to compute the stride)

Return: void (nothing)

Requirements:

- A configuration must be loaded
- The hologram window must be opened and must contain an image
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *OpenHoloWin*

GetHoloWidth

Gets the width of the hologram image, according to the current configuration.

Koala UI equivalent:
(not applicable)

(No parameters)

Return: Int32

Requirements:

- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *GetHoloHeight*

GetIntensity32fImage

Copies the current intensity (amplitude) image in the given memory location, as a floating point image (32 bits).
Note: Contrary to the GetPhase32fImage function, the image does not contain any mask information, even if a mask is active.

Koala UI equivalent:
(not applicable)

Parameters:

- buffer (*Single[]*): Array of singles of dimension `stride*height` where the data will be copied

Return: void (nothing)

Requirements:

- A configuration must be loaded
- The intensity (amplitude) window must be opened and must contain an image
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *GetIntensityImage*
- *GetPhase32fImage*
- *OpenIntensityWin*

GetIntensityImage

Copies the current intensity (amplitude) image in the given memory location, as a grayscale image (8 bits)

Koala UI equivalent:
(not applicable)

Parameters:

- buffer (*Byte[]*): Array of bytes of dimension `stride*height` where the data will be copied

Return: void (nothing)

Requirements:

- A configuration must be loaded
- The intensity (amplitude) window must be opened and must contain an image
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *GetIntensity32fImage*
- *OpenIntensityWin*

GetIntensityProfile

Copies the current intensity profile data in the given memory location

Note: the length of the array to receive the data is given by the function *GetIntensityProfileLength*

Koala UI equivalent:
(not applicable)

Parameters:

- buffer (Double[]): Array of doubles where the data will be copied

Return: void (nothing)

Example (C#):

```
host.SetIntensityProfileState(true);
host.ExtractIntensityProfile(100, 100, 200, 200);
int profileLength = host.GetIntensityProfileLength();
double[] profileData = new double[profileLength];
host.GetIntensityProfile(profileData);
```

Requirements:

- A configuration must be loaded
- A reconstruction object must be created
- An intensity profile must be selected
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *ExtractIntensityProfile*
- *GetIntensityProfileLength*
- *SetIntensityProfileState*

GetIntensityProfileLength

Gets the length of the current intensity profile array.

Note: this function is needed to know the length of the array to give to *GetIntensityProfile*.

Koala UI equivalent:
(not applicable)

(No parameters)

Return: Int32

Example: (See example for *GetIntensityProfile*)

Requirements:

- A configuration must be loaded
- A reconstruction object must be created
- An intensity profile must be selected
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *ExtractIntensityProfile*
- *GetIntensityProfile*
- *SetIntensityProfileState*

GetKoalaVersion

Gets the current Koala version number.

Koala UI equivalent:
It can be seen via the *Help* ⟶ *About your DHM* menu.

Parameters:

- None.

Return: string: The version number as a string value.

Requirements:

- A user must be logged in

Possible errors:

- -

See also:

- -

GetLambdaNm

Gets the wavelength of a laser source, in [nm].

Koala UI equivalent:
(Not applicable, but the wavelength of the sources is displayed in the *Sources* tab of the *About your DHM window*, which can be opened via the *Help* ⟶ *About your DHM* menu.)

Parameters:

- srcId (Int32): The id of the source (logical or physical according to *useLogicalId*)
- useLogicalId (Boolean): Set to `true` to use logical id, to `false` to use physical id. Optional, default value is `true`.

Return: Single: The wavelength of the source, in [nm]

Requirements:

- If logical ids are used, a configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

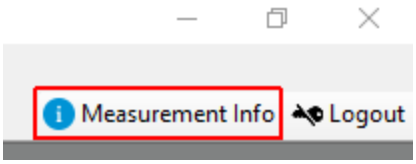
See also:

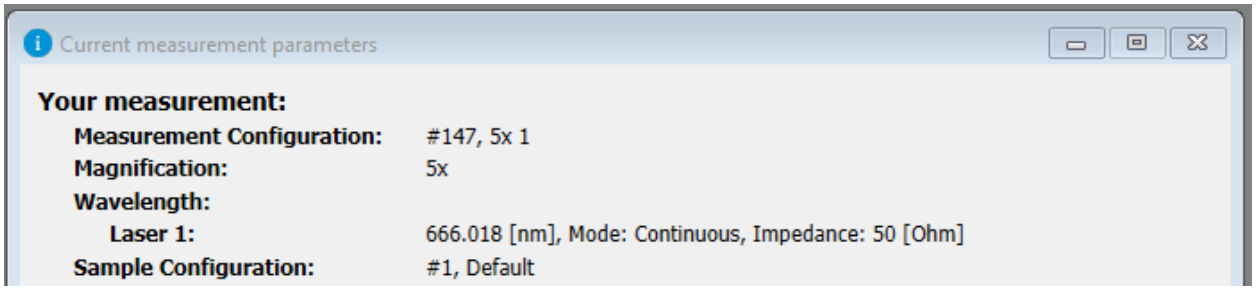
- -

GetMeasurementInformation

Saves the current measurement information.

Koala UI equivalent:
The window Measurement Information can be obtained via the top right button **Measurement Info**.





Parameters:

- Path (string): path to save the data.
- Filename (string): filename to save the content of the measurement information.

Return: void (nothing).

Requirements:

- A user must be logged in
- A configuration must be loaded.
- A Reconstruction must be available.

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

See also:

- -

GetOPLPos

Gets the non-corrected position of the OPL1 motor in [qc].

Koala UI equivalent:
(not applicable)

(No parameters)

Return: Int32: The position of the OPL1 motor, in [qc]

Requirements:

- The OPL1 option must be enabled
- The OPL1 motor must be available and initialized
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

See also:

- *MoveOPL*
- *MoveChosenOPLToPosition*
- *GetChosenOPLPosition*

GetPhase32fImage

Copies the current phase image in the given memory location, as a floating point image (32 bits)
Note: Contrary to the GetIntensity32fImage function, if a mask is active, the returned image contains the mask information (masked pixels are set to NaN).

Koala UI equivalent:
(not applicable)

Parameters:

- buffer (*Single[]*): Array of singles of dimension `stride*height` where the data will be copied

Return: void (nothing)

Example: (See example for *GetPhaseImage*)

Requirements:

- A configuration must be loaded
- The phase window must be opened and must contain an image
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *GetIntensity32fImage*
- *GetPhaseImage*
- *OpenPhaseWin*

GetPhaseHeight

Gets the height of the phase image ROI, according to the current configuration.
Note that the intensity (amplitude) image has the same dimensions.

Koala UI equivalent:
(not applicable)

(No parameters)

Return: Int32

Requirements:

- A configuration must be loaded
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *GetPhaseWidth*
- *OpenConfig*

GetPhaseImage

Copies the current phase image in the given memory location, as a grayscale image (8 bits)

Koala UI equivalent:
(not applicable)

Parameters:

- buffer (*Byte[]*): Array of bytes of dimension `stride*height` where the data will be copied

Return: void (nothing)

Example:

```
int width = host.GetPhaseWidth();
int height = host.GetPhaseHeight();
int stride = (width / 4)*4;
if (width %  4 != 0)
stride += 4;
byte[] buffer = new byte[stride*height];
host.GetPhaseImage(buffer);
```

Requirements:

- A configuration must be loaded
- The phase window must be opened and must contain an image
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *GetPhase32fImage*
- *OpenPhaseWin*

GetPhaseProfile

Copies the current phase profile data in the given memory location

Note: the length of the array to receive the data is given by the function *GetPhaseProfileLength*

Koala UI equivalent:
(not applicable)

Parameters:

- buffer (Double[]): Array of doubles where the data will be copied

Return: void (nothing)

Example (C#):

```
host.SetPhaseProfileState(true);
host.ExtractPhaseProfile(100, 100, 200, 200);
int profileLength = host.GetPhaseProfileLength();
double[] profileData = new double[profileLength];
host.GetPhaseProfile(profileData);
```

Requirements:

- A configuration must be loaded
- A reconstruction object must be created
- A phase profile must be selected
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *ExtractPhaseProfile*
- *GetPhaseProfileLength*
- *SetPhaseProfileState*

GetPhaseProfileLength

Gets the length of the current phase profile array.

Note: this function is needed to know the length of the array to give to *GetPhaseProfile*.

Koala UI equivalent:
(not applicable)

(No parameters)

Return: Int32

Example: (See example for *GetPhaseProfile*)

Requirements:

- A configuration must be loaded
- A reconstruction object must be created
- A phase profile must be selected
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *ExtractPhaseProfile*
- *GetPhaseProfile*
- *SetPhaseProfileState*

GetPhaseWidth

Gets the width of the phase image ROI, according to the current configuration.
Note that the intensity (amplitude) image has the same dimensions.

Koala UI equivalent:
(not applicable)

(No parameters)

Return: Int32

Requirements:

- A configuration must be loaded
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *GetPhaseHeight*

GetPxSizeUm

Returns the calibrated size, in [um], represented by a pixel in the image.
Note that this value is calibrated and configuration-dependent, because it depends on the objective used for this configuration.

Koala UI equivalent:
(not applicable)

(No parameters)

Return: Single: the calibrated size of a pixel, averaged between X and Y, in [um]

Requirements:

- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- -

GetRecDistCM

Gets the current reconstruction distance, in [cm], of the active user processing configuration.

Koala UI equivalent:
(Not applicable, but the reconstruction distance is displayed in the *Focus* field of the *Reconstruction settings window*, which can be opened via the *Settings* ⟶ *Reconstruction* menu.)

(No parameters)

Return: Single: The reconstruction distance, in [cm]

Example: (See example for *OnDistanceChange*)

Requirements:

- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

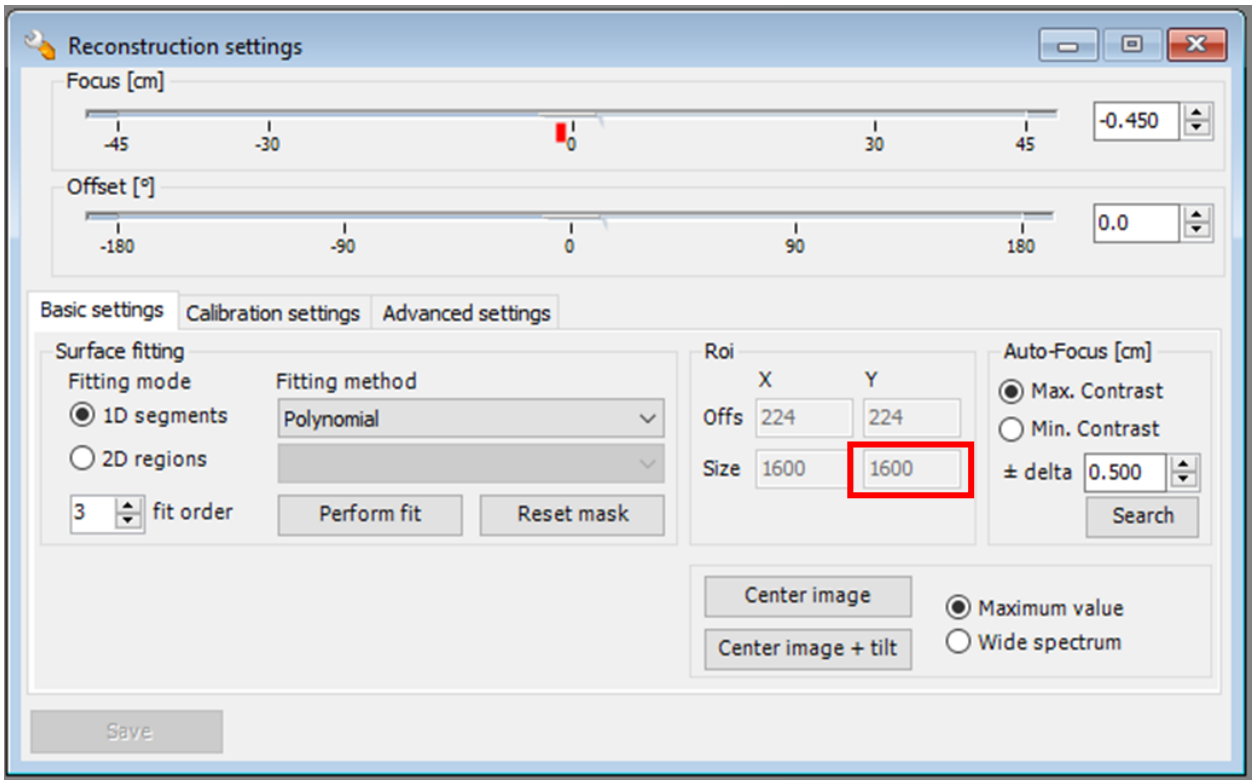
- *OnDistanceChange*
- *SetRecDistCM*

GetReconstructionRoiHeight

Gets the height of the reconstruction ROI, according to the current configuration.

Koala UI equivalent:

The height is visible in the *Reconstruction settings* window, but it is not editable.



(No parameters)

Return: Int32

Requirements:

- A configuration must be loaded
- A reconstruction object must be created
- A reconstruction object must contain a ROI.
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *GetReconstructionRoiLeft*

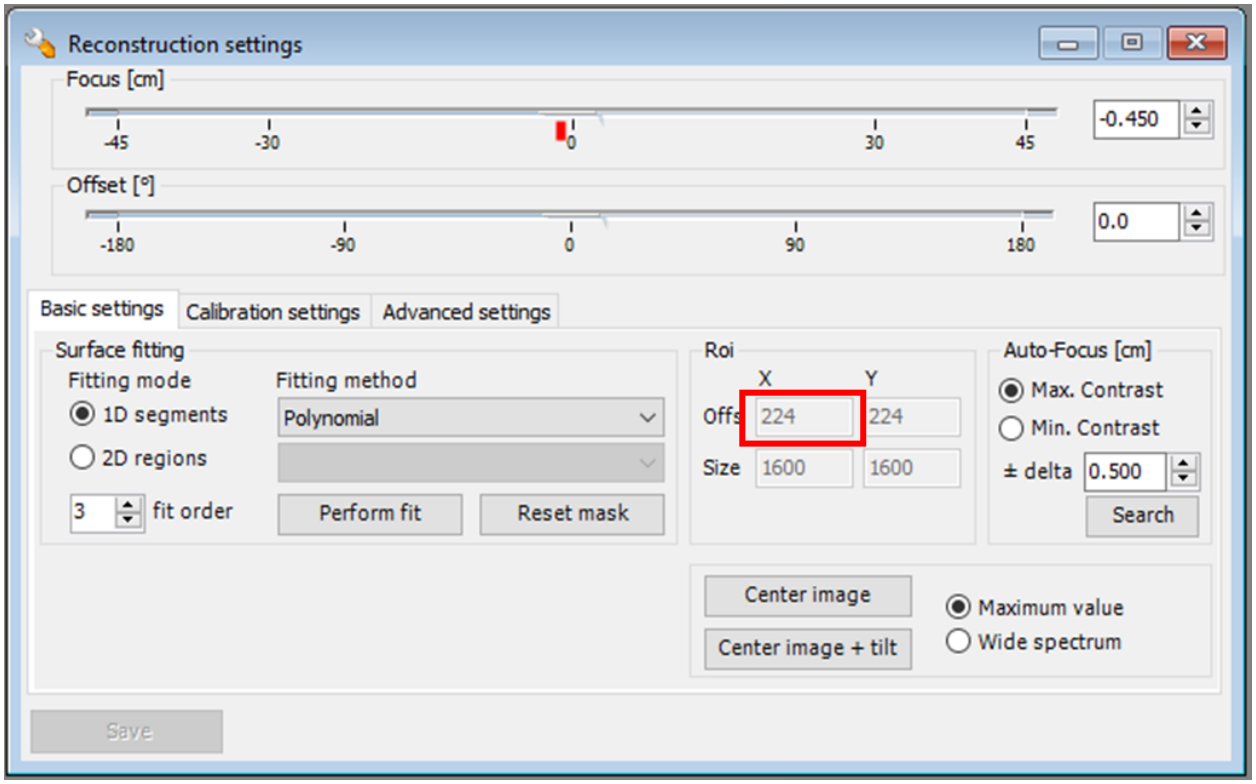
- *GetReconstructionRoiWidth*
- *GetReconstructionRoiTop*
- *ResetReconstructionRoi*
- *SetReconstructionRoi*

GetReconstructionRoiLeft

Gets the left coordinate of the reconstruction ROI, according to the current configuration.

Koala UI equivalent:

The left coordinate is visible in the *Reconstruction settings* window, but it is not editable.



(No parameters)

Return: Int32

Requirements:

- A configuration must be loaded
- A reconstruction object must be created
- A reconstruction object must contain a ROI.
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *GetReconstructionRoiHeight*
- *GetReconstructionRoiWidth*

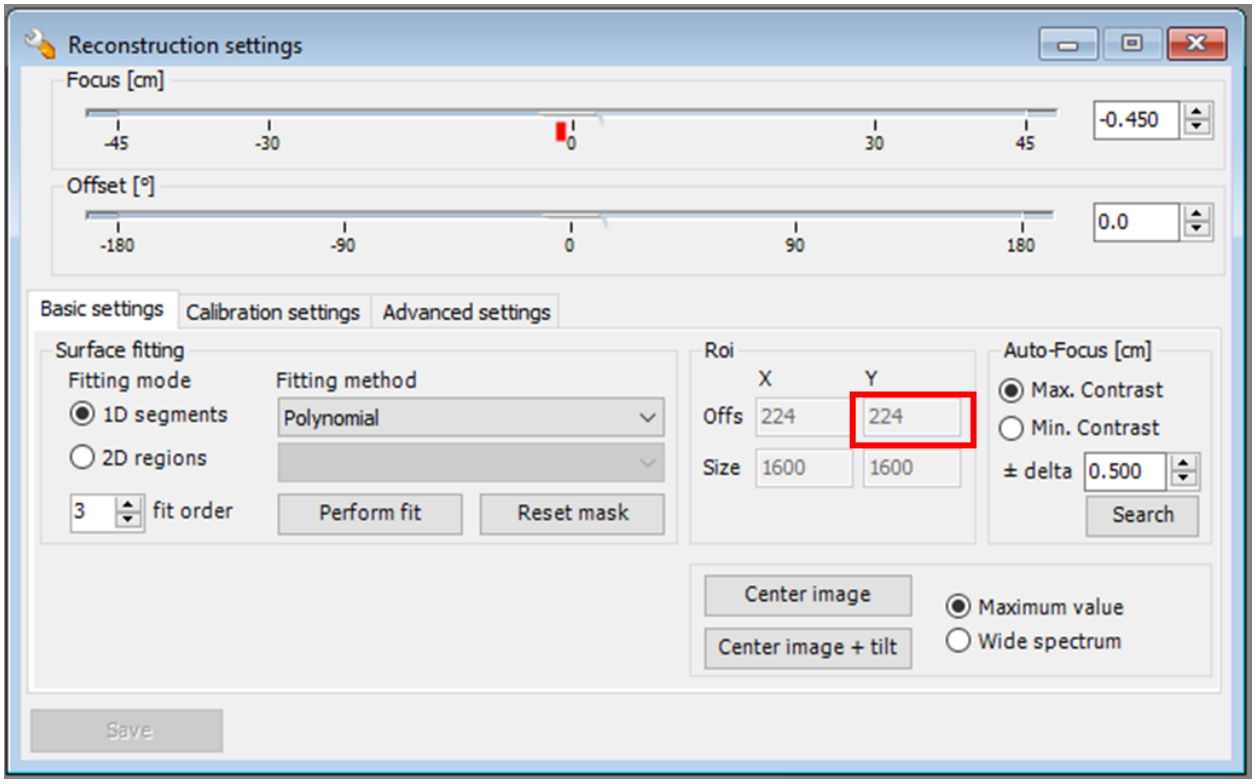
- *GetReconstructionRoiTop*
- *ResetReconstructionRoi*
- *SetReconstructionRoi*

GetReconstructionRoiTop

Gets the top coordinate of the reconstruction ROI, according to the current configuration.

Koala UI equivalent:

The top coordinate is visible in the *Reconstruction settings* window, but it is not editable.



(No parameters)

Return: Int32

Requirements:

- A configuration must be loaded
- A reconstruction object must be created
- A reconstruction object must contain a ROI.
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *GetReconstructionRoiHeight*
- *GetReconstructionRoiLeft*

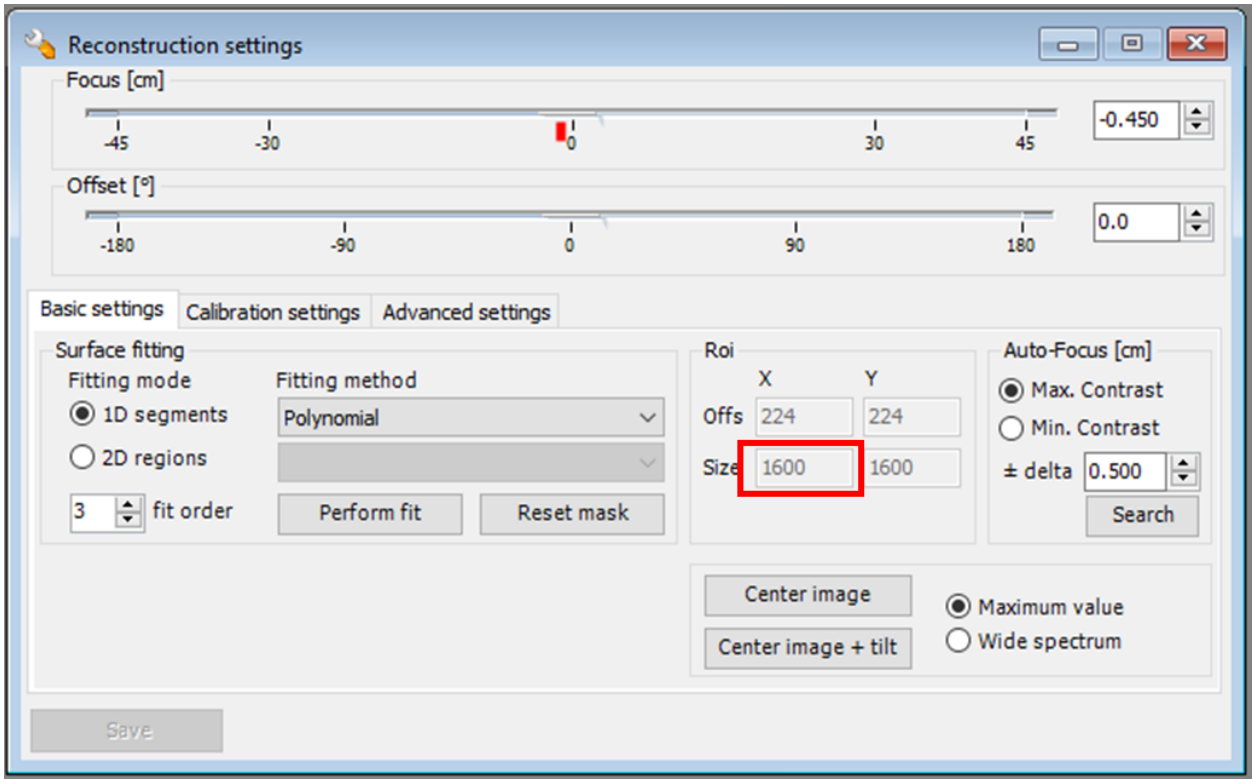
- *GetReconstructionRoiWidth*
- *ResetReconstructionRoi*
- *SetReconstructionRoi*

GetReconstructionRoiWidth

Gets the width of the reconstruction ROI, according to the current configuration.

Koala UI equivalent:

The width is visible in the *Reconstruction settings* window, but it is not editable.



(No parameters)

Return: Int32

Requirements:

- A configuration must be loaded
- A reconstruction object must be created
- A reconstruction object must contain a ROI.
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *GetReconstructionRoiHeight*
- *GetReconstructionRoiLeft*

- *GetReconstructionRoiTop*
- *ResetReconstructionRoi*
- *SetReconstructionRoi*

GetUnwrap2DState

Gets a value indicating if the unwrapping of the phase image is enabled or not.

Koala UI equivalent:

(Not applicable, but the unwrapping is enabled if the *Unwrap*  button is activated in the *Phase Image window*.)

(No parameters)

Return: Boolean

Requirements:

- A configuration must be loaded
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *SetUnwrap2DState*
- *SetUnwrap2DMethod*

InitXYZStage

Moves the axes of the XYZ stage to their minimal position.
Please use with caution as moving to the minimal positions (X=0, Y=0, Z=0) may damage your stage if it is not correctly calibrated.

Koala UI equivalent:
(not applicable)

Parameters:

- withProgressBar (Boolean): If set to `true`, the progress bar will be displayed. (Note that progress tracking might not be implemented depending of your type of stage. In this case the progress bar will simply remain on 0.) Optional, default value is `false`.
- moveToCenter (Boolean): If set to `true`, will move the stage to the center of each axis range afterwards. Optional, default value is `false`.

Return: Boolean: returns `true` if the operation was successful.

Requirements:

- A stage must be available and initialized
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

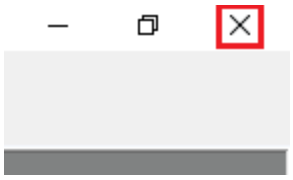
See also:

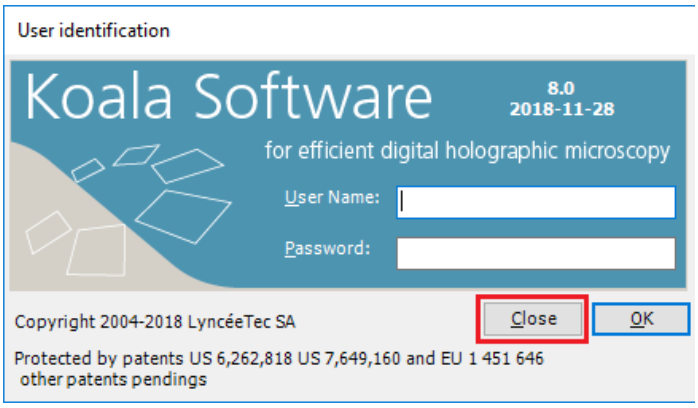
- -

KoalaShutDown

Closes Koala. This function is not blocking and will return before execution is finished.

Koala UI equivalent:
Click on the closing button at the top right of the Koala main window or clicking on the *Close* button in the login window of Koala.





Parameters:

- confirm (Boolean): `true` to display the dialog to ask for confirmation, `false` to close without the need for user input. Optional, default value is `false`.

Return: void (nothing)

Requirements: (nothing)

Possible errors: (nothing)

See also:

- -

LoadHolo

Loads an hologram from the disk. In order to get a correct phase and intensity reconstruction of the hologram, it should be loaded with the configuration that was used to record it. If the hologram window is not opened yet, it will be opened automatically.

Note that this function **does not support multi-holograms files anymore**. (Those files were created from alternate dual wavelengths configurations, which are deprecated.)

Koala UI equivalent:

Clicking on the *Open multi-hologram*  button or *Open hologram*  button (depending of your configuration) in the main toolbar and opening an hologram file. Or via the *Mode* ⟶ *Hologram 2-lambdas* or *Mode* ⟶ *Hologram* menu (depending of your configuration).

Parameters:

- Path (String): Full path of the hologram file. It is recommended to only work with files in .tif format.
- numLambda (Int32): The number of wavelengths of the configuration with which the hologram was taken.

Return: void (nothing)

Requirements:

- A user must be logged in
- A configuration must be loaded
- The hologram parameters such as height, width and number of bits per pixel must correspond to the parameters of the current configuration

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *SaveImageFloatToFile*
- *SaveImageToFile*

Login

Login for the remote client.
Note that connection and login is handled automatically in the *Koala Remote Test App*. It is only needed if you implement your own client application.

Koala UI equivalent:
(not applicable)

Parameters:

- password (String): The password for the current Koala user

Return: `true` if the login was successful.

Requirements:

- A user must be logged in Koala
- The remote client application must be connected

Possible errors:

- The *Remote Log Message window* is not opened or needs to be reopened because of a previous login failure

See also:

- *Connect*
- *Logout*

Logout

Logout for the remote client and disconnects the TCP client without waiting for an answer before returning.
This function should be called instead of *ExtUIGoodBye*, which is deprecated
After a *Logout*, it is possible to call *Connect* and then *Login* again, without the need to restart the *Remote Message Log window* in Koala.
Note that disconnection is handled automatically in the *Koala Remote Test App*. It is only needed if you implement your own client application.

Koala UI equivalent:
(not applicable)

(No parameters)

Return: void (nothing)

Requirements:

- A user must be logged in Koala
- The remote client application must be connected
- The remote client application must have logged in

Possible errors:

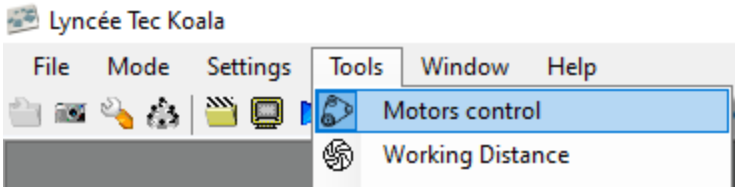
- The *Remote Log Message window* is not opened

See also:

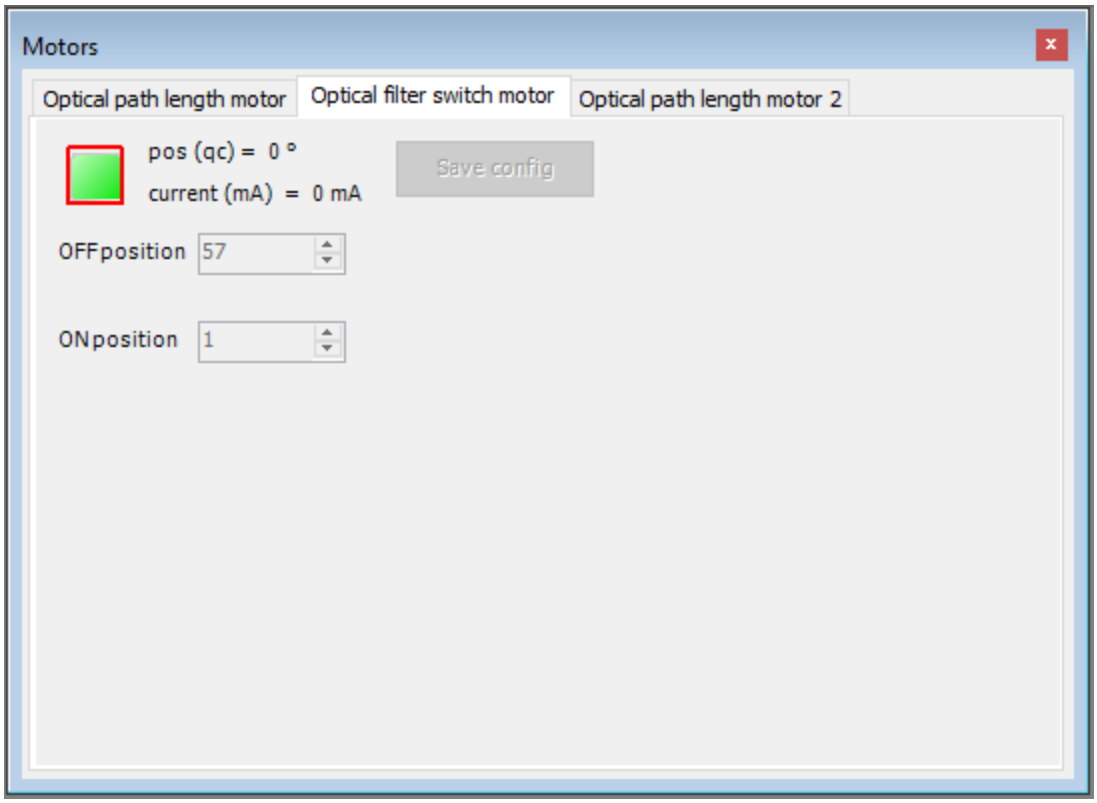
- *Connect*
- *ExtUIGoodBye*
- *Login*

ModifyFilterSwitchStatus

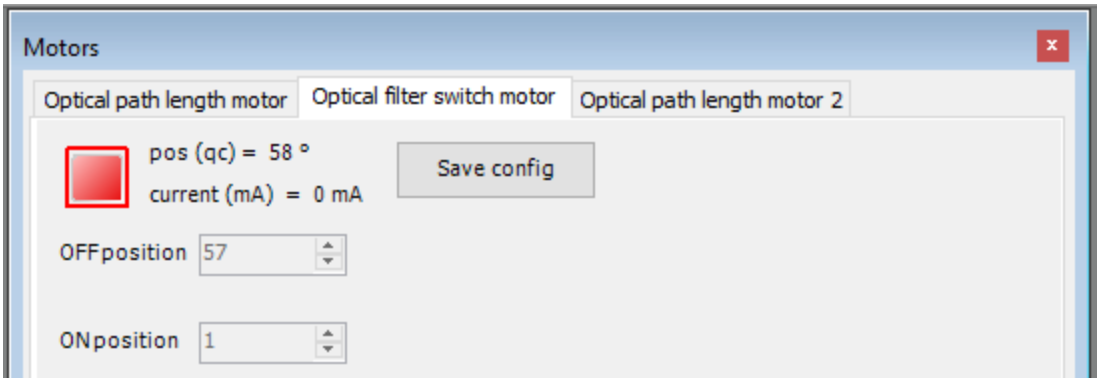
Enables/Disables the optical filter switch motor, accessible in the **Motors control** panel from the **Tools** menu.



Koala UI equivalent:
Corresponds to clicking on the **Green/Red** button for the optical filter switch motor of the *Motors* window.



Filter Switch is on position ON.



Filter Switch is on position OFF.

Parameters:

- state (Boolean): `true` to enable the filter switch motor, `false` to disable it

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in
- The Filter Switch option must be enabled in the Motors window.
- The Filter Switch motor must be available and initialized

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- -

MoveAxes

Moves several axes of the stage simultaneously.

Koala UI equivalent:
Moving the axes via the *Stage XYZ window*, or moving the XYZ stage axes via the joystick.

Parameters:

- absMove (Boolean): Set to `true` for an absolute move, set to `false` for a relative move.
- mvX (Boolean): Set to `true` to move the X axis.
- mvY (Boolean): Set to `true` to move the Y axis.
- mvZ (Boolean): Set to `true` to move the Z axis.
- mvTh (Boolean): Set to `true` to move the Theta axis.
- distX (Double): Absolute position or distance to move for the X axis. In [um].
- distY (Double): Absolute position or distance to move for the Y axis. In [um].
- distZ (Double): Absolute position or distance to move for the Z axis. In [um].
- distTh (Double): Absolute position or distance to move for the Theta axis. In [um].
- accX (Double): Accuracy of the move for the X axis, in [um].
- accY (Double): Accuracy of the move for the Y axis, in [um].
- accZ (Double): Accuracy of the move for the Z axis, in [um].
- accTh (Double): Accuracy of the move for the Theta axis, in [um].
- waitEnd (Boolean): If set to `true`, the function will only return after the move is completed. Optional, default value is `true`.

Return: Boolean: `true` if the operation completed successfully.

Requirements:

- A stage must be available and initialized
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *GetAxesPosMu*
- *MoveAxis*
- *MoveAxesArr*

MoveAxesArr

Moves several axes of the stage simultaneously.

Koala UI equivalent:
Moving the axes via the *Stage XYZ window*, or moving the XYZ stage axes via the joystick.

Parameters:

- axes (Boolean[]): Array of 4 booleans for the X, Y, Z and Theta axes respectively, to indicate if the axis must be moved or not (set to `true` to move).
- absMove (Boolean): Set to `true` for an absolute move, set to *false* for a relative move.
- dist (Double[]): Array of 4 double for the X, Y, Z and Theta axes respectively, for the absolute position or distance to move for each axis. In [um].
- acc (Double[]): Array of 4 double for the X, Y, Z and Theta axes respectively, for the accuracy of the move for each axis. In [um].
- waitEnd (Boolean): If set to `true`, the function will only return after the move is completed. Optional, default value is `true`.

Return: Boolean: `true` if the operation completed successfully.

Requirements:

- A stage must be available and initialized
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error
- One or several of the argument array does not have the correct dimension ⇒ *Execution Failed* error.

See also:

- *GetAxesPosMu*
- *MoveAxis*
- *MoveAxes*

MoveAxis

Moves a single axis of the stage.

Koala UI equivalent:
Moving a single axis via the *Stage XYZ window*, or moving a XYZ stage axis via the joystick.

Parameters:

- axisId (Int32): Id of the axis to move. Possible values are
  - 0: X axis
  - 1: Y axis
  - 2: Z axis
  - 3: Theta axis (rotation, system-dependent)
- absMove (Boolean): Set to `true` for an absolute move, set to `false` for a relative move.
- distUM (Double): Absolute position or distance to move. In [um].
- accuracyUM (Double): Accuracy of the move, in [um]. This parameter is ignored and will be removed in a future version.
- waitEnd (Boolean): If set to `true`, the function will only return after the move is completed. Optional, default value is `true`.

Return: Boolean: `true` if the operation completed successfully.

Requirements:

- A stage must be available and initialized
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *GetAxesPosMu*
- *MoveAxes*
- *MoveAxesArr*

MoveChosenOPLToPosition

Moves the OPL1 or OPL2 motor (depending on the **OPL ID**) to a specific position in [qc].
Note that this function does not require a configuration to be loaded, but it doesn't make much sense to move the OPL before loading a configuration.

Koala UI equivalent:
(not applicable for normal users)

Parameters:

- posQc (Int32): The position in qc where to move the OPL1 or OPL2 . This position corresponds to the position returned by the *GetChosenOPLPosition* function. The OPL will move to the absolute position posQc while taking into account the OPL compensation (factor in database, weighted by the difference between maximum position and minimum position, then divided by physical dimensions of the OPL)
- oplId (Int32): The OPL ID (1 or 2).

Return: void (nothing)

Requirements:

- The OPL1 or OPL2 option must be enabled
- The OPL1 or OPL2 motor must be available and initialized
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error
- The target OPLx position is outside the range of possible values ⇒ *Execution Failed* error

See also:

- *GetOPLPos*
- *GetChosenOPLPosition*
- *MoveOPL*

MoveOPL

Moves the OPL1 motor to a specific position, in [qc].
Note that this function does not require a configuration to be loaded, but it doesn't make much sense to move the OPL1 before loading a configuration.

Koala UI equivalent:
(not applicable for normal users)

Parameters:

- position (Int32): The position in qc where to move the OPL1. This position corresponds to the position returned by the *GetOPLPos* function.

Return: void (nothing)

Requirements:

- The OPL1 option must be enabled
- The OPL1 motor must be available and initialized
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error
- The target OPL1 position is outside the range of possible values ⇒ *Execution Failed* error

See also:

- *GetChosenOPLPosition*
- *GetOPLPos*
- *MoveChosenOPLToPosition*

OnDistanceChange

Applies the last modification of the reconstruction distance, which will result in the recomputation of the intensity and phase images if they are available.

Koala UI equivalent:
(not applicable, done automatically)

(No parameters)

Return: void (nothing)

Example:

```
host.SetRecDistCM(reconstructionDistance);
host.OnDistanceChange();
```

Requirements:

- A configuration must be loaded
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *GetRecDistCM*
- *SetRecDistCM*

OpenConfig

Opens the selected configuration.

Koala UI equivalent:
Corresponds to *File → Open Configuration* and then selecting a configuration and clicking on *OK*, or to clicking on the *Open configuration* button  on the toolbar and then selecting a configuration and clicking on OK.

Parameters:

- configNumber (*Int32*): ID of the Measurement Configuration to open

Return: void (nothing)

Requirements:

- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error
- The configuration does not exist ⇒ *Execution Failed* error

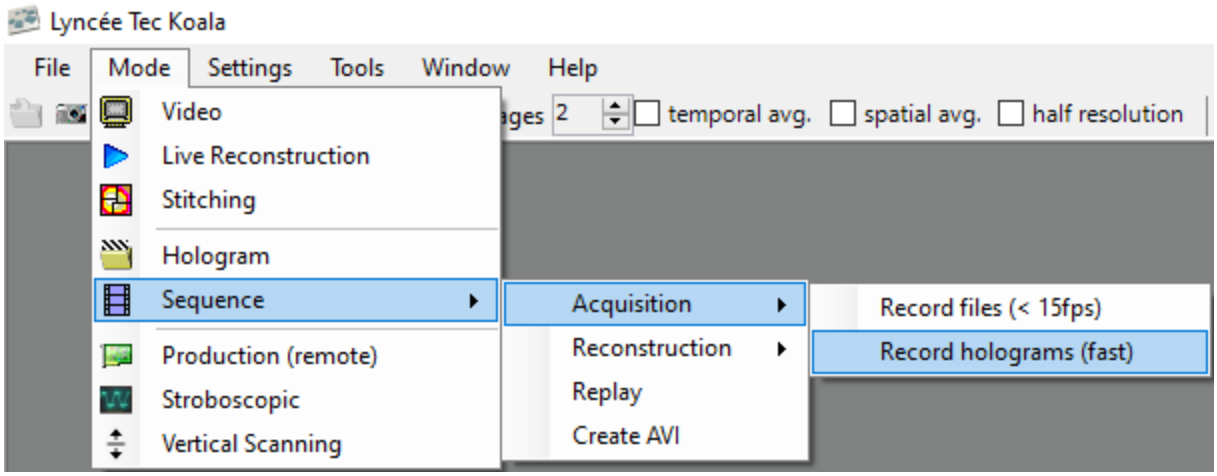
See also:

- -

OpenFastHologramsRecordWin

Opens the fast holograms acquisition window

Koala UI equivalent:
Corresponds to clicking on the **Sequence → Acquisition → Record holograms (fast)** option in the ***Mode*** menu.



(No parameters)

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in
- The window must not be already opened.

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *CloseFastHologramsRecordWin*
- *OpenHoloWin*
- *OpenIntensityWin*
- *OpenPhaseWin*
- *OpenStroboWin*

OpenFrmTopography

Opens the topography (roughness) window

Koala UI equivalent:
(No exact equivalent, but the *Roughness phase window* (*Topography window*) is opened when a topography zone is drawn on the *Phase Image window* when the *Draw a roughness area*  button is activated.)

(No parameters)

Return: void (nothing)

Requirements:

- The phase window must be opened and must contain an image
- The topography option must be enabled in the phase window
- The last selected item in the phase window must be a topography zone or topography profile
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *SelectTopoZone*

OpenHoloWin

Opens the hologram window

Koala UI equivalent:

Corresponds to clicking on the *Open Holo window* button  on the toolbar.

(No parameters)

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *OpenIntensityWin*
- *OpenMaskSettingsWin*
- *OpenPhaseWin*
- *OpenReconstructionSettingsWin*
- *OpenStroboWin*

OpenIntensityWin

Opens the intensity (amplitude) window

Koala UI equivalent:

Corresponds to clicking on the *Open Intensity window* button  in the toolbar.

Parameters:

- updateXYScale (*Boolean*): If set to `true`, the scale displayed in the *Intensity Image window* will be updated. Optional, default value is `true`.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *OpenHoloWin*
- *OpenMaskSettingsWin*
- *OpenPhaseWin*
- *OpenReconstructionSettingsWin*
- *OpenStroboWin*
- *CloseIntensityWin*

OpenMaskSettingsWin

Opens the mask settings window

Koala UI equivalent:
Corresponds to clicking on the *Mask* option in the *Settings* menu.

(No parameters)

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in
- The Mask settings window must be closed

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *OpenHoloWin*
- *OpenIntensityWin*
- *OpenPhaseWin*
- *OpenReconstructionSettingsWin*
- *OpenStroboWin*
- *CloseMaskSettingsWin*

OpenPhaseWin

Opens the phase window

Koala UI equivalent:

Corresponds to clicking on the *Open Phase window* button  in the toolbar

Parameters:

- withoutColorbar (*Boolean*): If set to `true`, the phase window will not have a color bar. Optional, default value is `false`.
- doReconstruction (*Boolean*): If set to `true`, a full reconstruction will be performed after the phase image is opened, which will update its content. Optional, default value is `true` (recommended).
- updateXYScale (*Boolean*): If set to `true`, the scale displayed in the *Phase Image window* will be updated. Optional, default value is `true`.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *OpenHoloWin*
- *OpenIntensityWin*
- *OpenMaskSettingsWin*
- *OpenReconstructionSettingsWin*
- *OpenStroboWin*
- *ClosePhaseWin*

OpenReconstructionSettingsWin

Opens the reconstruction settings window

Koala UI equivalent:
Corresponds to clicking on the *Reconstruction* option in the *Settings* menu.

(No parameters)

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

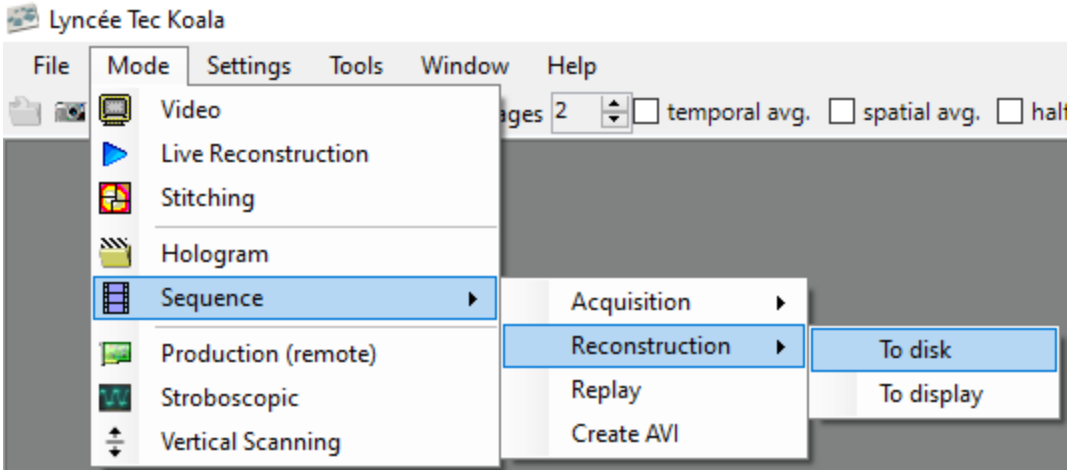
See also:

- *OpenHoloWin*
- *OpenIntensityWin*
- *OpenPhaseWin*
- *OpenStroboWin*
- *CloseReconstructionSettingsWin*

OpenReconstructionToDiskSequenceWin

Opens the reconstruction to disk settings window

Koala UI equivalent:
Corresponds to clicking on the ***Sequence → Reconstruction → To disk*** option in the ***Mode*** menu.



(No parameters)

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in
- The reconstruction to disk window must not be already opened

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *OpenHoloWin*
- *OpenIntensityWin*
- *OpenPhaseWin*
- *OpenStroboWin*
- *CloseReconstructionToDiskSequenceWin*

ResetCorrSegment

Reset the phase correction segments.

Koala UI equivalent:
In the *Phase Image window*, clicking on the *Reset phase correction segments* context menu (which appears when the *Horizontal and Vertical profile for phase mask adjustment* button is activated, one or more phase correction segments have been drawn and the button was the last activated button).

(No parameters)

Return: void (nothing)

Requirements:

- A configuration must be loaded
- The phase window must be opened and must contain an image
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *AddCorrSegment*
- *AddCorrZone*
- *ResetCorrZone*
- *AlgoResetPhaseMask*
- *ComputePhaseCorrection*

ResetCorrZone

Reset the phase correction zones.

Koala UI equivalent:
In the *Phase Image window*, clicking on the *Reset phase correction zones* context menu (which appears when the *Horizontal and Vertical profile for phase mask adjustment*  button is activated, one or more phase correction zones have been drawn and the button was the last activated button).

(No parameters)

Return: void (nothing)

Requirements:

- A configuration must be loaded
- The phase window must be opened and must contain an image
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *AddCorrSegment*
- *AddCorrZone*
- *ResetCorrSegment*
- *AlgoResetPhaseMask*
- *ComputePhaseCorrection*

ResetGrab

Stops the continuous acquisition of the camera if it was ON.

Koala UI equivalent:
(Not applicable)

(No parameters)

Return: void (nothing)

Requirements:

- The intern camera must be available
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *DigitizerAcquiring*

ResetPhaseOffsetAdjustmentZone

Reset the phase offset adjustment zones.

Koala UI equivalent:
In the *Phase Image window*, clicking on the *Reset phase correction zones* context menu (which appears when the *Phase offset adjustment zone*  button is activated, one or more phase correction zones have been drawn and the button was the last activated button).

(No parameters)

Return: void (nothing)

Requirements:

- A configuration must be loaded
- The phase window must be opened and must contain an image
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *AddEllipticalUserDefinedMaskZoneToPhase*
- *AddRectangularUserDefinedMaskZoneToPhase*
- *CloseMaskSettingsWin*
- *OpenMaskSettingsWin*
- *SetInteractionModeWhenAddingMaskZone*
- *SetUserDefinedMaskState*

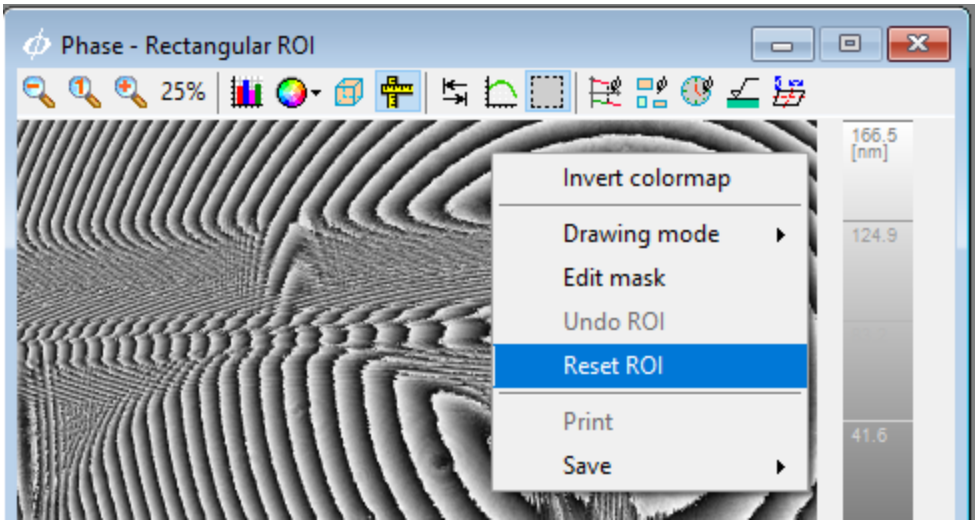
ResetReconstructionRoi

Removes the ROI on the reconstruction (Phase and Intensity). The phase and intensity size corresponds to the whole hologram image size.

! IMPORTANT ! Do not save the configuration in this state, as Koala will not be able to reload it.

! IMPORTANT ! Do not call this function twice in a row: Koala forbids it.

Koala UI equivalent:
Resets the user defined ROI in the *Phase Image window*, when the button **Define a new Roi** is activated (right-click to access the drop-down menu), and a user defined ROI has already been added.



Parameters:
- None

Return: void (nothing)

Requirements:
- A configuration must be loaded
- A reconstruction object must be created
- A reconstruction object must contain a ROI.
- A user must be logged in

Possible errors:
- Current Koala state does not meet the requirements ⇒ *Invalid Operation*
- Function is called twice ⇒ *Invalid Operation*

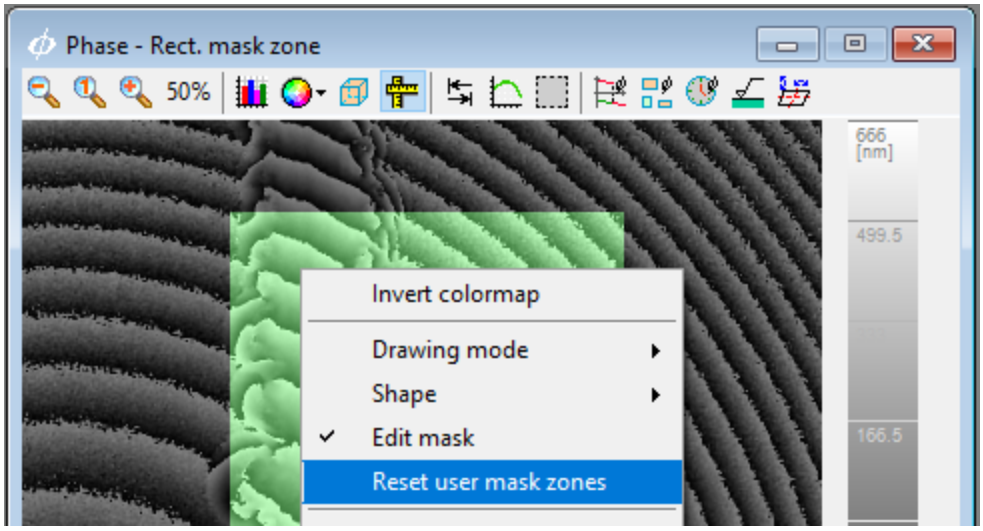
See also:
- *SetReconstructionRoi*

- *GetReconstructionRoiHeight*
- *GetReconstructionRoiLeft*
- *GetReconstructionRoiWidth*
- *GetReconstructionRoiTop*

ResetUserDefinedMaskZone

Remove all user defined mask zones on the Phase.

Koala UI equivalent:
Resets the user defined mask zones in the *Phase Image window*, when the **Edit mask** is activated (right-click to access the drop-down menu), and a user defined zone has already been added.



Parameters:
- None

Return: void (nothing)

Requirements:
- A configuration must be loaded
- The phase window must be opened and must contain an image
- A reconstruction object must be created
- A user must be logged in
- The mask window must be opened

Possible errors:
- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

See also:
- *AddEllipticalUserDefinedMaskZoneToPhase*
- *AddRectangularUserDefinedMaskZoneToPhase*
- *SetUserDefinedMaskState*

SaveImageFloatToFile

Saves a 32 bits image on the disk, as .bin or .txt.
Note that the function does not return an error if the low-level saving operation failed.

Koala UI equivalent:
Clicking on the *Save* ⟶ *Save as float* context menu of an image

Parameters:

- winId (Int32): Id of the image to save. Possible values are:
  - 1: hologram
  - 2: amplitude image
  - 4: phase image
  - 8: Fourier image

  For amplitude, phase and Fourier images, what exact type of image is saved (Lambda 1 only, Lambda 2 only, short or long synthetic wavelength or wavelength mapping) depends on the selected image in the corresponding window.
  Values cannot be combined. To save several images, call the function several times.
- fileName (String): Full path of the destination file. The extension has no influence on the file format, which is determined by the *useBinFormat* parameter. However it is recommended to use the correct extension to avoid confusion when working with the file later on.
- useBinFormat (Boolean): Set to `true` to save as .bin file, `false` to save as .txt. Optional, default value is `false`.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- The corresponding window must be opened and must contain an image
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error
- Invalid winId ⇒ *Execution Failed* error

See also:

- *SaveImageToFile*
- *SelectDisplayWL*

SaveImageToFile

Saves an 8 bits image on the disk, as tiff, png or jpeg. If the directory does not exist, it will be created.
Note that the function does not return an error if the low-level saving operation failed.

Koala UI equivalent:
Clicking on the *Save* ⟶ *Save as TIF* context menu of an image

Parameters:

- winId (Int32): Id of the image to save. Possible values are:
  - 1: hologram
  - 2 : amplitude image
  - 4 : phase image
  - 8 : Fourier image

  For amplitude, phase and Fourier images, what exact type of image is saved (Lambda 1 only, Lambda 2 only, short or long synthetic wavelength or wavelength mapping) depends on the selected image in the corresponding window.
  Values cannot be combined. To save several images, call the function several times.
- fileName (String): Full path of the destination file. The file format will be defined according to the extension. If no extension is given, the file will be recorded in tiff format. It is recommended to save holograms as .tif if you intend to load them again for further processing.
  Possible extension values are
  - .png
  - .jpg
  - .tiff or .tif

Return: void (nothing)

Requirements:

- A configuration must be loaded
- The corresponding window must be opened and must contain an image
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*
- Invalid winId ⇒ *Execution Failed*

See also:

- *SaveImageFloatToFile*
- *SelectDisplayWL*

SaveReconstructionSettings

Opens the reconstruction settings window

Koala UI equivalent:
Corresponds to clicking on the *Save* button (if available) in the *Reconstruction Settings* window. The configuration loaded must be of the same user level as the user logged in.

(No parameters)

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in
- The *Reconstruction Settings* window must be opened.

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *CloseReconstructionSettingsWin*
- *OpenReconstructionSettingsWin*

SelectDisplayWL

Select the specific type of wavelength (WL) to display in the phase, amplitude or Fourier window.

Koala UI equivalent:
Clicking on one of the following buttons in an image:

- *Long synthetic WL*  button (in the *Phase Image window*)
- *Short synthetic WL*  button (in the *Phase Image window*)
- *1st WL*  button (in the *Phase / Intensity / Fourier Image windows*)
- *2nd WL*  button (in the *Phase / Intensity / Fourier Image windows*)

Parameters:

- winId (Int32): Id of the image to display. Possible values are:
    - 8192: phase lambda 1 image
    - 16384: phase lambda 2 image
    - 32768: phase long synthetic wavelength image
    - 65536: phase short synthetic wavelength image
    - 2048: amplitude (intensity) lambda 1 image
    - 4096: amplitude (intensity) lambda 2 image
    - 512: Fourier lambda 1 image
    - 1024: Fourier lambda 2 image

  Values cannot be combined.To display several images, call the function several times.

Return: void (nothing)

Requirements:

- The corresponding window must be opened and the corresponding image type must be available
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*
- Invalid winId ⇒ *Execution Failed*

See also:

- *OpenIntensityWin*
- *OpenPhaseWin*

SelectTopoZone

Selects an area for topographic (roughness) measurement.

Koala UI equivalent:

Tracing a zone in the *Phase Image window*, when the *Draw a roughness area* button  is activated.

Parameters:

- top (Int32): Y coordinate of the top left point of the zone, in pixel
- left (Int32): X coordinate of the top left point of the zone, in pixel
- width (Int32): width of the zone, in pixel
- height (Int32): height of the zone, in pixel

Return: void (nothing)

Requirements:

- A configuration must be loaded
- The phase window must be opened and must contain an image
- The topography option  must be selected in the phase window
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*
- The given zone is not fully inside the phase ROI ⇒ *Execution Failed*

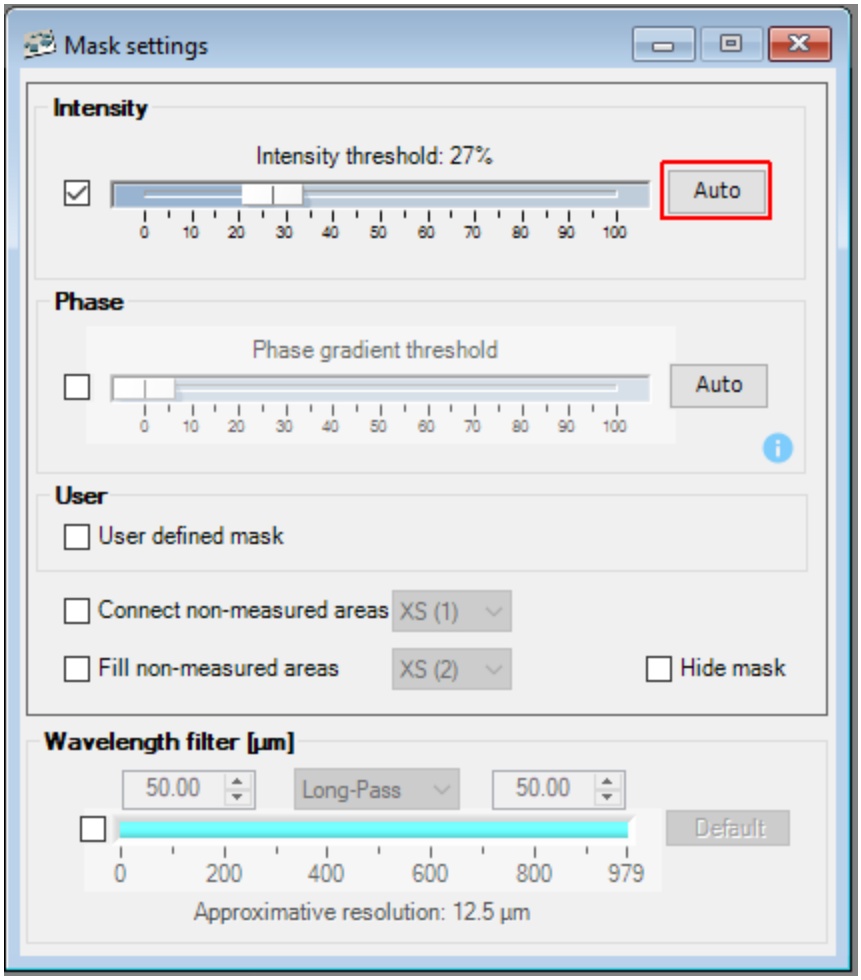
See also:

- *OpenFrmTopography*

SetAutomaticIntensityThresholdFilterToEnabledState

Enables the intensity threshold filter on the intensity, and sets the intensity threshold filter value automatically..

Koala UI equivalent:
Corresponds to clicking on the **Auto** button for the intensity threshold filter of the *Mask Settings Window*.



Parameters:

- None.

Return: void (nothing)

Requirements:

- A configuration must be loaded

- A user must be logged in
- The mask window must be opened

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

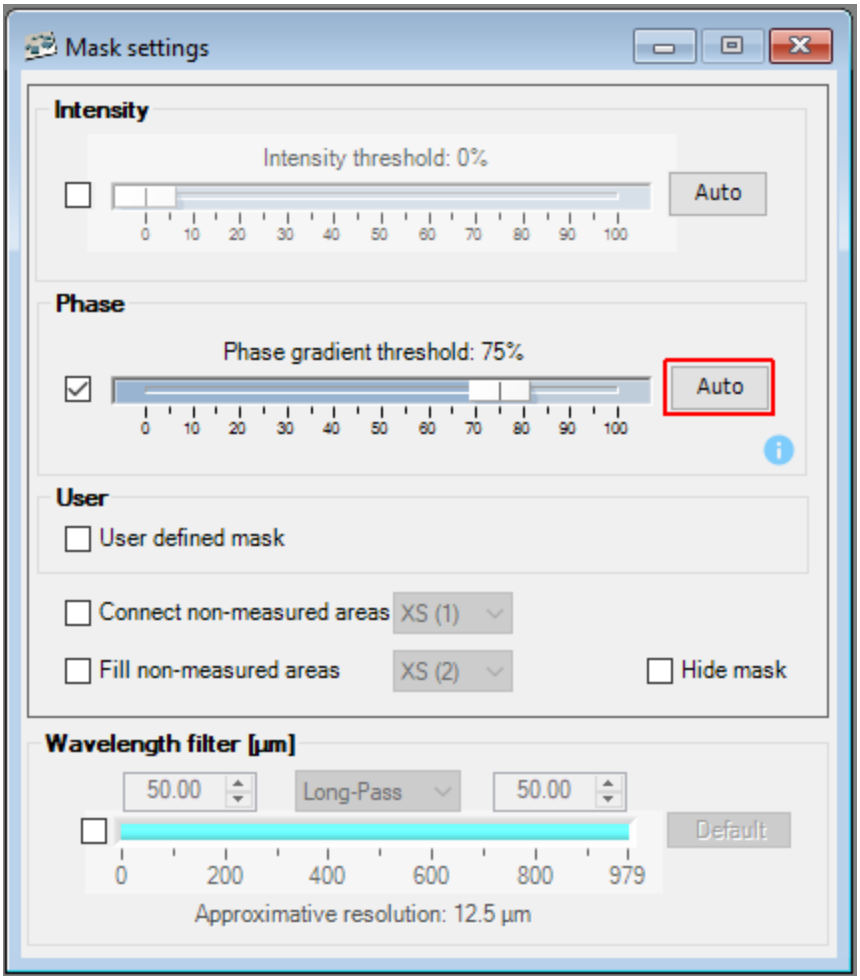
See also:

- *CloseMaskSettingsWin*
- *OpenMaskSettingsWin*
- *SetIntensityThresholdFilterState*
- *SetIntensityThresholdFilterValueInPercent*

SetAutomaticPhaseGradientThresholdFilterToEnabledState

Enables the phase gradient threshold filter on the phase, and sets the phase gradient threshold filter value automatically..

Koala UI equivalent:
Corresponds to clicking on the **Auto** button for the phase gradient threshold filter of the *Mask Settings Window*.



Parameters:

- None.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in

- The mask window must be opened

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *CloseMaskSettingsWin*
- *OpenMaskSettingsWin*
- *SetPhaseGradientThresholdFilterState*
- *SetPhaseGradientThresholdFilterValueInPercent*

SetCameraShutterUs

Sets the shutter value of the camera, in [us].
Note that this function does not require a configuration to be loaded, but it doesn't make much sense to set the shutter value before loading a configuration.

Koala UI equivalent:
Setting the *Shutter* value in the *Camera settings window*.

Parameters:

- shutterUs (Int32): The shutter value for the camera, in [us]

Return: void (nothing)

Requirements:

- The intern camera must be available
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

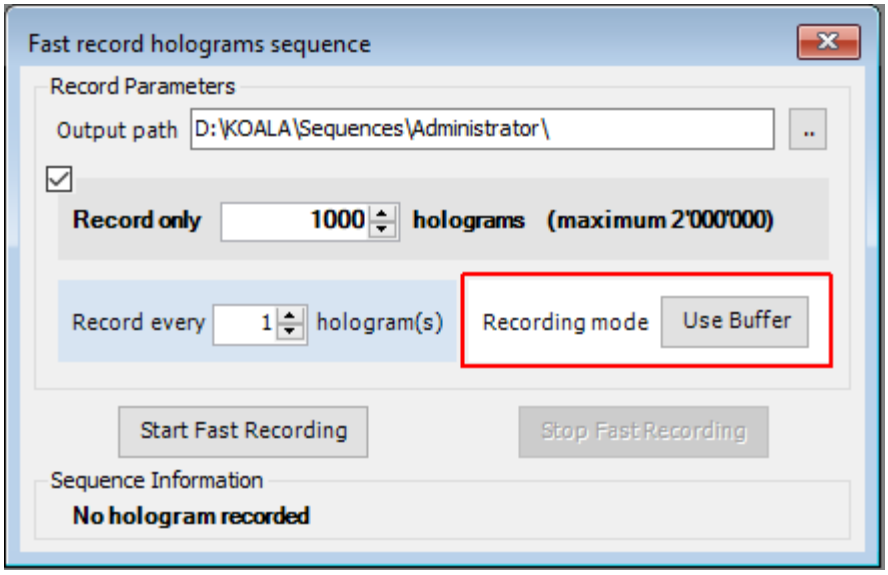
See also:

- *GetCameraShutterUs*

SetFastHologramsSequenceRecordingModeBuffer

Enables or disables the buffer mode using the *Fast record holograms sequence* window.

Koala UI equivalent:
Corresponds to enabling the button **Use Buffer** in the **Fast record holograms sequence** window.



Parameters:

- state (*Boolean*): `true` to enable the intensity threshold filter, `false` to disable it.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in
- The Fast Record Holograms sequence window must be opened.

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

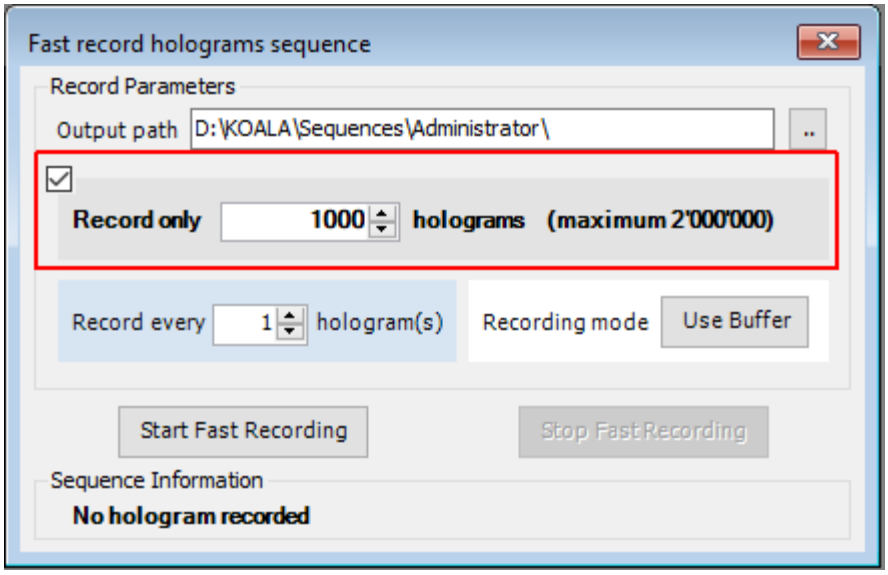
See also:

- *CloseFastHologramsRecordWin*
- *OpenFastHologramsRecordWin*
- *SetFastHologramsSequenceRecordNumberOfHolograms*
- *SetFastHologramsSequenceRecordPath*
- *SetFastHologramsSequenceRecordPeriodicallyEveryXHologram*
- *StartFastHologramsSequenceRecord*
- *StopFastHologramsSequenceRecord*

SetFastHologramsSequenceRecordNumberOfHolograms

Set the number of holograms to be recorded using the *Fast record holograms sequence* window.

Koala UI equivalent:
Corresponds to enabling the checkbox and defining the number of holograms in the **Fast record holograms sequence** window.



Parameters:

- numberOfHolograms (Int32): number of holograms to be recorded in the Hologram.raw sequence.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in
- The Fast Record Holograms sequence window must be opened.

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

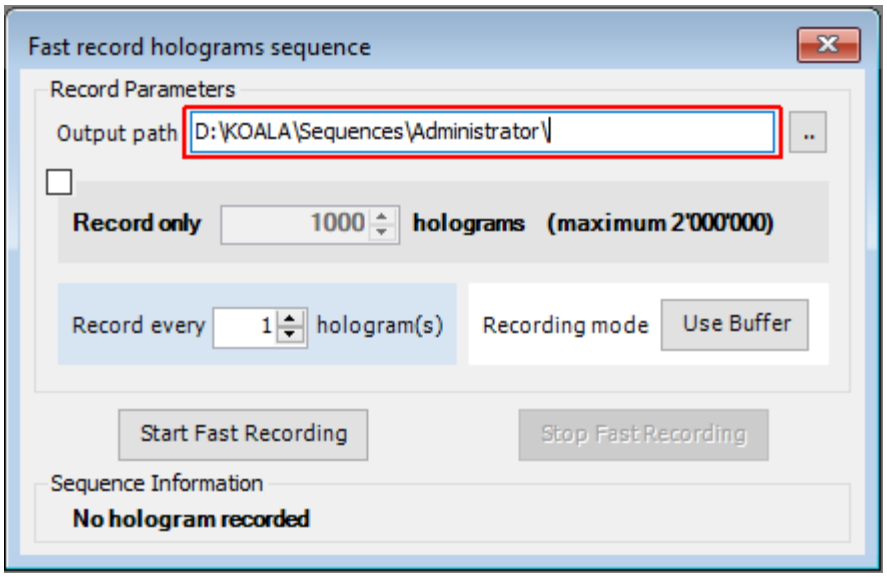
- *CloseFastHologramsRecordWin*
- *OpenFastHologramsRecordWin*
- *SetFastHologramsSequenceRecordingModeBuffer*
- *SetFastHologramsSequenceRecordPath*
- *SetFastHologramsSequenceRecordPeriodicallyEveryXHologram*

- *StartFastHologramsSequenceRecord*
- *StopFastHologramsSequenceRecord*

SetFastHologramsSequenceRecordPath

Set the path to record the hologram sequence using the *Fast record holograms sequence* window.

Koala UI equivalent:
Corresponds to defining the output path in the **Fast record holograms sequence** window.



Parameters:

- Path (string): Full path to the folder where to store Hologram.raw sequence.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in
- The window must be opened.

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

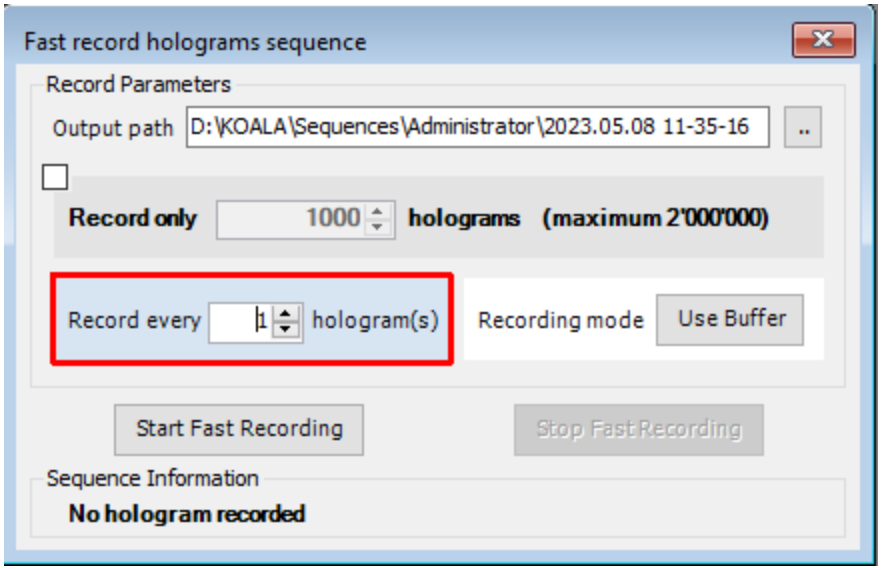
See also:

- *CloseFastHologramsRecordWin*
- *OpenFastHologramsRecordWin*
- *SetFastHologramsSequenceRecordingModeBuffer*
- *SetFastHologramsSequenceRecordNumberOfHolograms*
- *SetFastHologramsSequenceRecordPeriodicallyEveryXHologram*
- *StartFastHologramsSequenceRecord*
- *StopFastHologramsSequenceRecord*

SetFastHologramsSequenceRecordPeriodicallyEveryXHologram

Set the number of holograms period when recording the hologram sequence using the *Fast record holograms sequence* window. We record 1 hologram periodically every X holograms, (X-1) being the number of holograms not recorded.

Koala UI equivalent:
Corresponds to defining the number of holograms skipped in the **Fast record holograms sequence** window.



Parameters:

- numberOfHolograms (Int32): we record **1 hologram every numberOfHolograms** in the Hologram.raw sequence.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in
- The window must be opened.

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *CloseFastHologramsRecordWin*
- *OpenFastHologramsRecordWin*
- *SetFastHologramsSequenceRecordingModeBuffer*
- *SetFastHologramsSequenceRecordNumberOfHolograms*

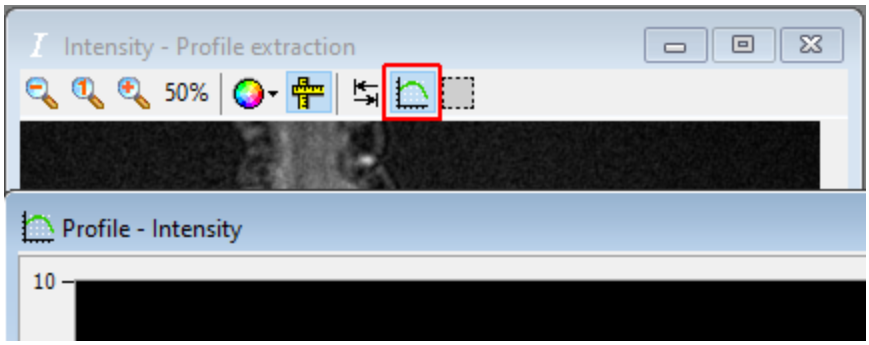
- *SetFastHologramsSequenceRecordPath*
- *StartFastHologramsSequenceRecord*
- *StopFastHologramsSequenceRecord*

SetIntensityProfileState

Opens or closes the intensity profile window. The profile window must be opened to be able to extract the intensity profile.

Koala UI equivalent:

Opening or closing the *Intensity Profile* window by pressing the button 



Parameters:

- state (Boolean): `true` to open the intensity profile window, `false` to close it.

Return: void (nothing)

Example: (See example for *GetPhaseProfile*)

Requirements:

- A configuration must be loaded
- The intensity window must be opened and must contain an image
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

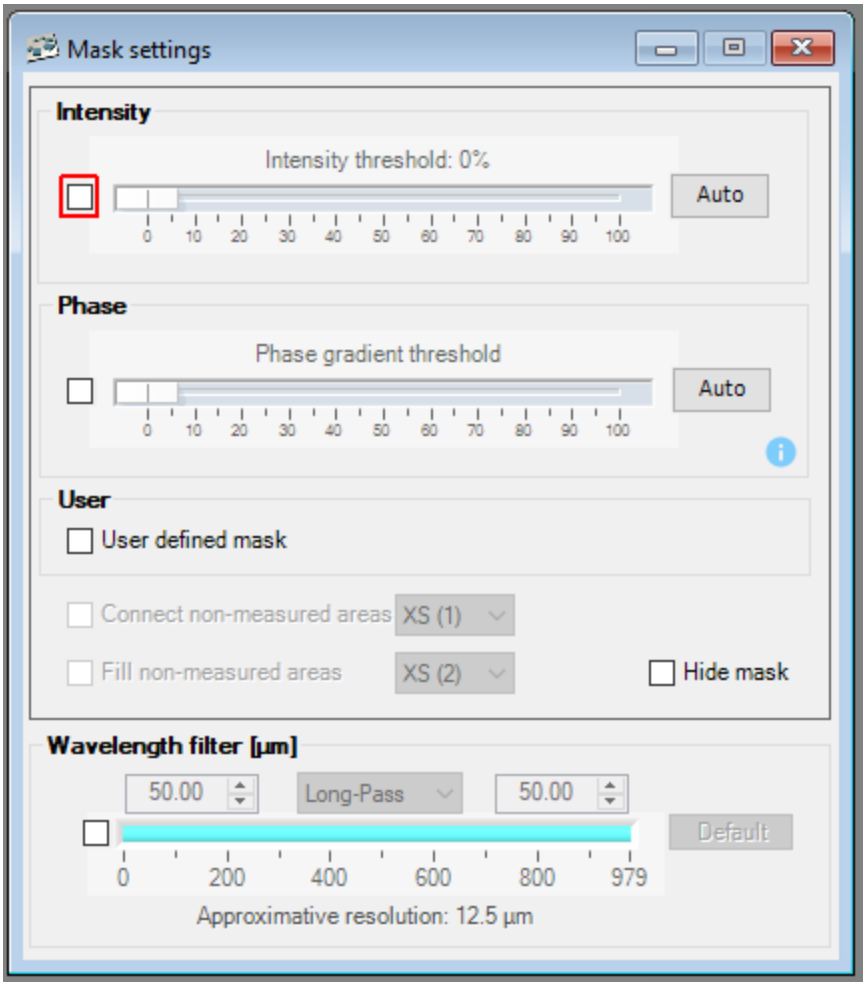
See also:

- *ExtractIntensityProfile*
- *GetIntensityProfile*
- *GetIntensityProfileLength*

SetIntensityThresholdFilterState

Enables or disables the intensity threshold filter on the intensity.

Koala UI equivalent:
Corresponds to clicking on the check button of the *Mask Settings Window*.



Parameters:

- state (*Boolean*): `true` to enable the intensity threshold filter, `false` to disable it.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in
- The Mask settings window must be opened

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

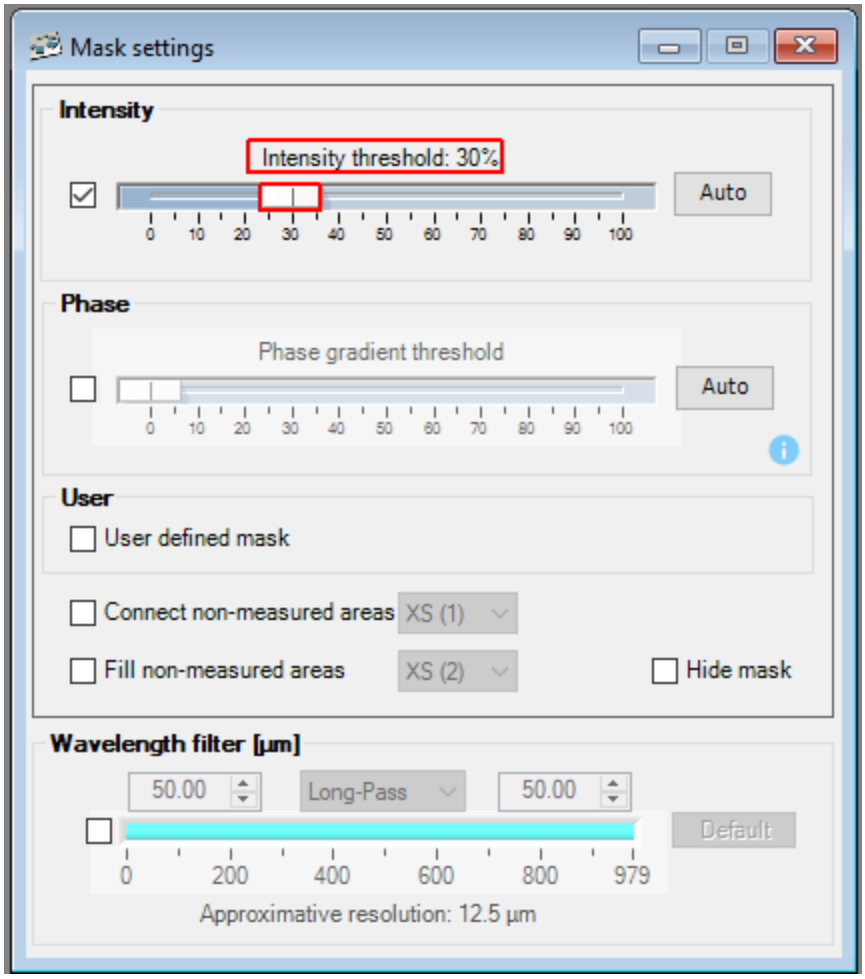
See also:

- *CloseMaskSettingsWin*
- *OpenMaskSettingsWin*
- *SetIntensityThresholdFilterValueInPercent*
- *SetAutomaticIntensityThresholdFilterToEnabledState*

SetIntensityThresholdFilterValueInPercent

Sets the intensity threshold filter value in percent.

Koala UI equivalent:
Corresponds to setting the intensity threshold value fixed with the intensity slider of the *Mask Settings Window*.



Parameters:

- Value (*float*): value of the intensity filter in percent.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in

- The mask window must be opened

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

Example:

```
//Set the intensity filter to 43%
host.SetIntensityThresholdFilterValueInPercent(43);
```

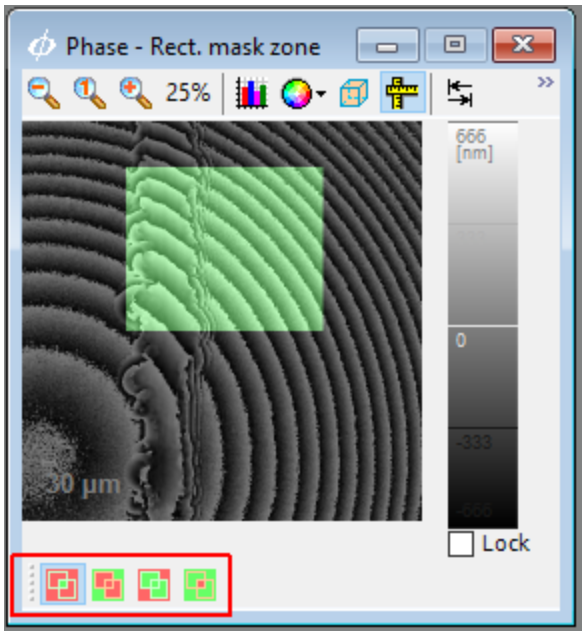
See also:

- *CloseMaskSettingsWin*
- *OpenMaskSettingsWin*
- *SetIntensityThresholdFilterState*
- *SetAutomaticIntensityThresholdFilterToEnabledState*

SetInteractionModeWhenAddingMaskZone

Set the interaction mode between user defined zones when adding a new mask zone on the Phase.

Koala UI equivalent:
Selecting the interaction mode at the bottom of the *Phase Image window*.



Parameters:

- mode (Int32): chose mode of interaction, with possible values [1,2,3,4].
    - 1: keep inside intersection 
    - 2: cut inside intersection 
    - 3: keep inside union 
    - 4: cut inside union 

Return: void (nothing)

Requirements:

- A configuration must be loaded
- The phase window must be opened and must contain an image
- A reconstruction object must be created
- A user must be logged in
- The mask window must be opened

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

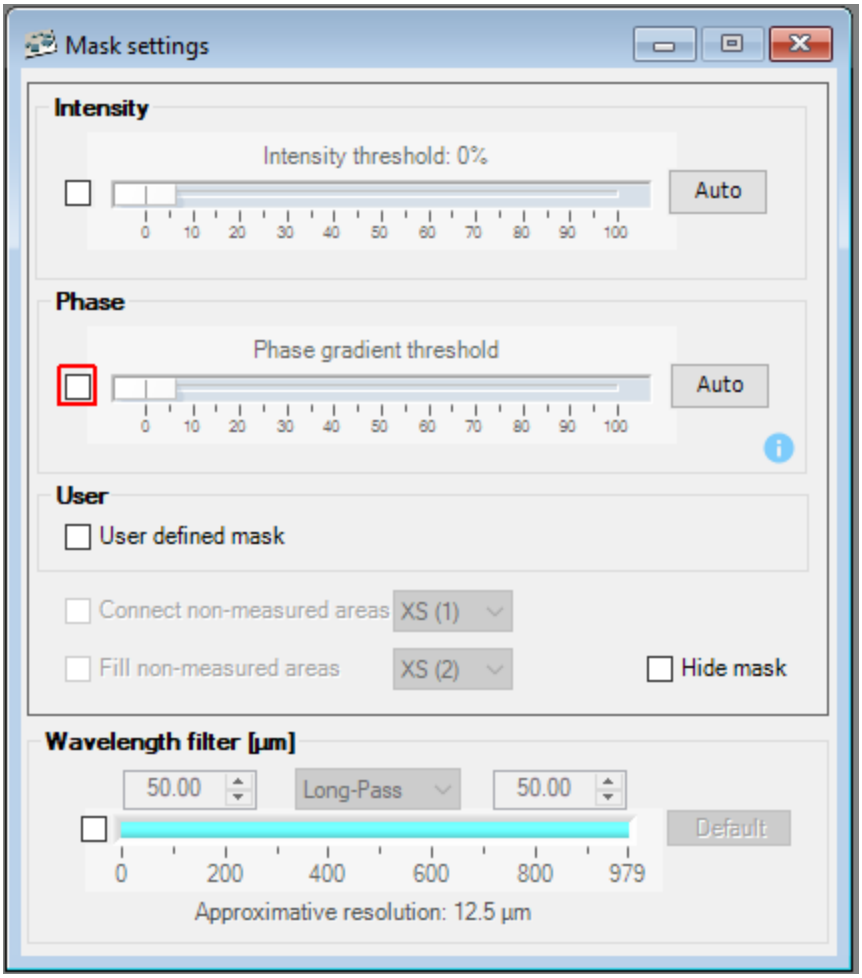
See also:

- *AddEllipticalUserDefinedMaskZoneToPhase*
- *AddRectangularUserDefinedMaskZoneToPhase*
- *CloseMaskSettingsWin*
- *OpenMaskSettingsWin*
- *ResetUserDefinedMaskZone*
- *SetUserDefinedMaskState*

SetPhaseGradientThresholdFilterState

Enables or disables the phase gradient threshold filter on the phase.

Koala UI equivalent:
Corresponds to clicking on the check button of the Phase gradient filter in the *Mask Settings Window*.



Parameters:

- state (*Boolean*): `true` to enable the intensity threshold filter, `false` to disable it.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in
- The Mask settings window must be opened

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

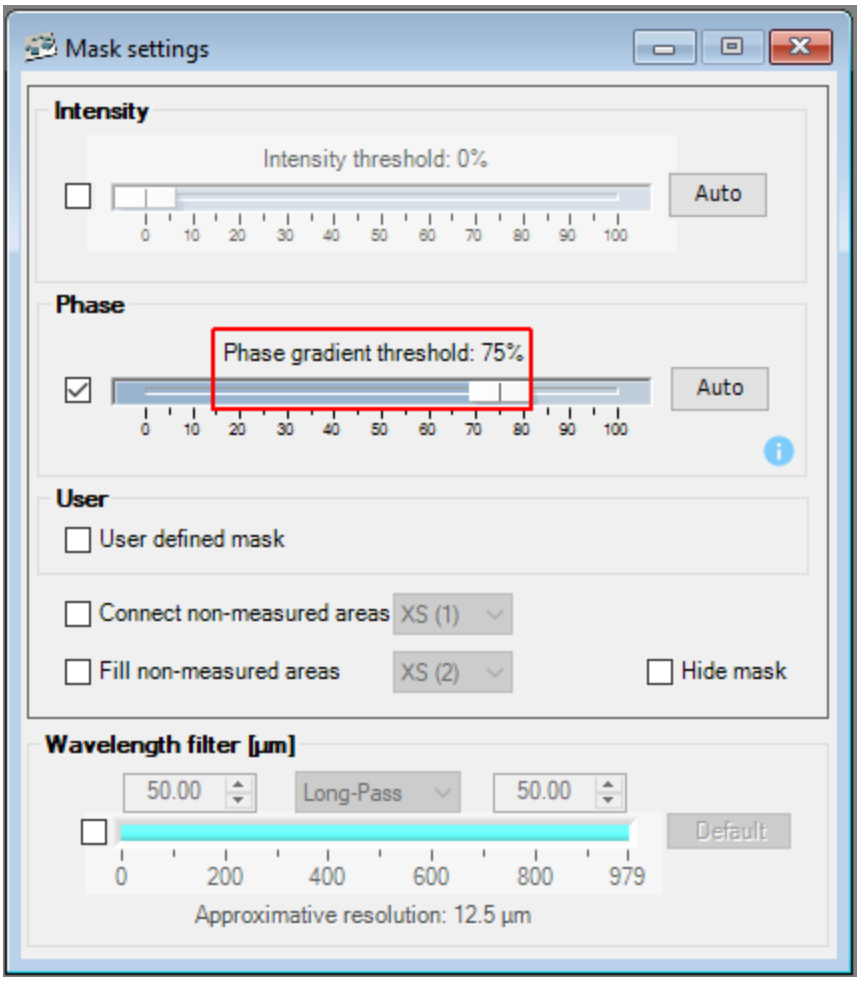
See also:

- *CloseMaskSettingsWin*
- *OpenMaskSettingsWin*
- *SetPhaseGradientThresholdFilterValueInPercent*
- *SetAutomaticPhaseGradientThresholdFilterToEnabledState*

SetPhaseGradientThresholdFilterValueInPercent

Sets the phase gradient threshold filter value in percent.

Koala UI equivalent:
Corresponds to setting the phase gradient threshold value defined with the phase gradient slider of the *Mask Settings Window*.



Parameters:

- Value (*float*): value of the phase gradient filter in percent.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in
- The mask window must be opened

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *CloseMaskSettingsWin*
- *OpenMaskSettingsWin*
- *SetPhaseGradientThresholdFilterState*
- *SetAutomaticPhaseGradientThresholdFilterToEnabledState*

SetPhaseProfileState

Opens or closes the phase profile window. The profile window must be opened to be able to extract the phase profile.

Koala UI equivalent:
Opening or closing the *Phase Profile window*

Parameters:

- state (Boolean): `true` to open the phase profile window, `false` to close it. Optional, default value is `false`.

Return: void (nothing)

Example: (See example for *GetPhaseProfile*)

Requirements:

- A configuration must be loaded
- The phase window must be opened and must contain an image
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

See also:

- *ExtractPhaseProfile*
- *GetPhaseProfile*
- *GetPhaseProfileLength*

SetRecDistCM

Sets the reconstruction distance, in [cm], for the active user processing configuration.
Note that the value is not saved in the database, the database value will be loaded again the next time the configuration is loaded.

Koala UI equivalent:
Setting the *Focus* value in the *Reconstruction settings window*

Parameters:

- distCM (Single): The reconstruction distance, in [cm]

Return: void (nothing)

Example: (See example for *OnDistanceChange*)

Requirements:

- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

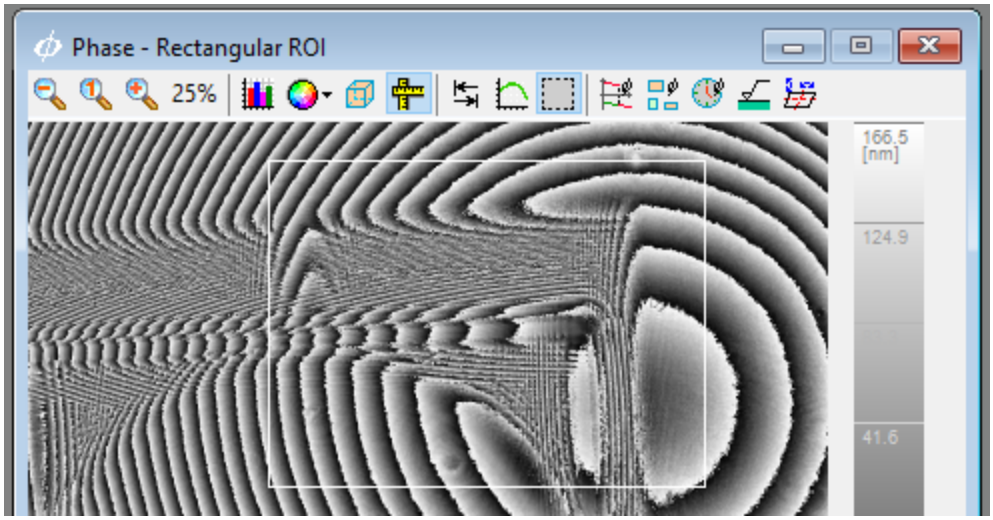
See also:

- *GetRecDistCM*
- *OnDistanceChange*

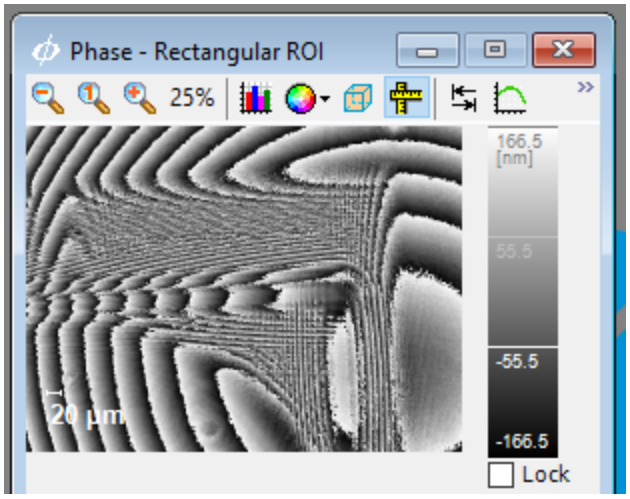
SetReconstructionRoi

Set the ROI of the reconstruction.

Koala UI equivalent:
Sets the user defined ROI in the *Phase Image window* and *Intensity Image Window*, when the **Define a new Roi** is activated (left-click to select the area, and middle-click to validate it). Here, we show the process with the Phase Image window.



Select the area in the Phase Image window



Validate it

Parameters:

- Left (Int32): The left coordinate of the ROI, in [px], in the hologram image.
- Top (Int32): The top coordinate of the ROI, in [px], in the hologram image.
- Width (Int32): The width of the ROI, in [px] (Left+Width < Width(hologram))
- Height (Int32): The height of the ROI, in [px] (Top+Height < Height(hologram))

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*
- If the width or height is too large ⇒ *Execution failed (Exception thrown)*

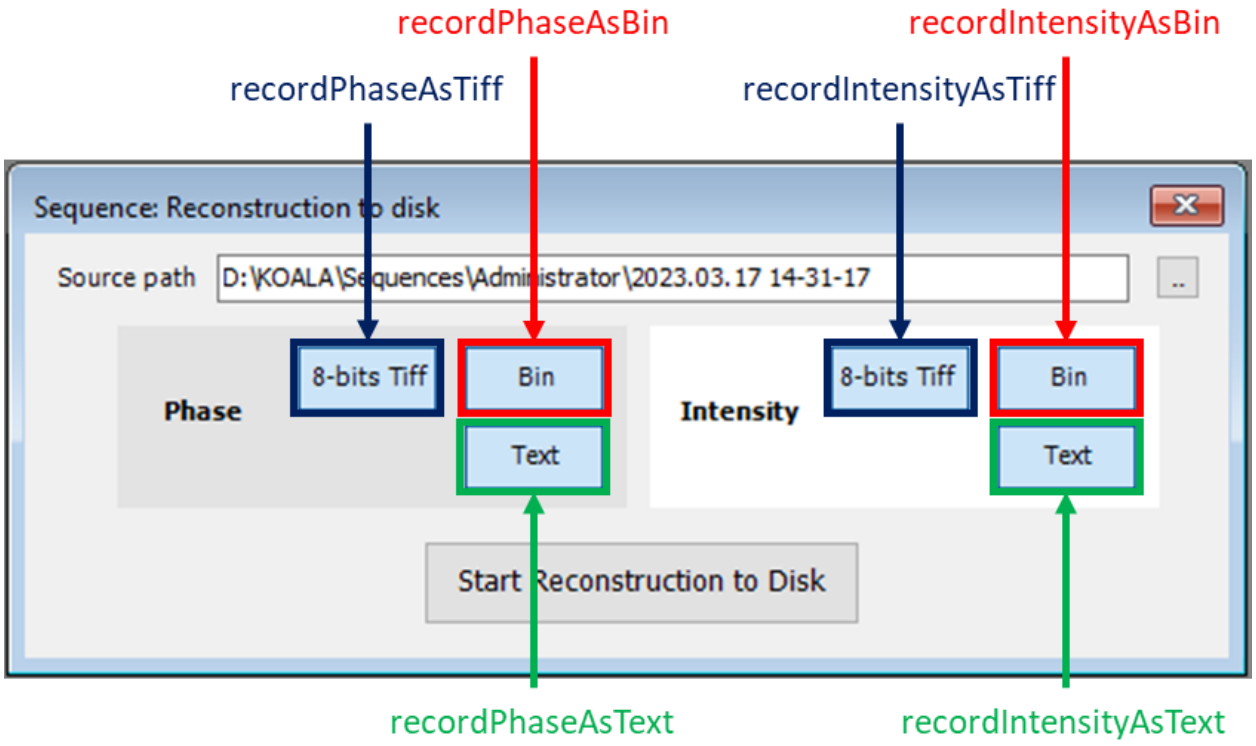
See also:

- *GetReconstructionRoiHeight*
- *GetReconstructionRoiLeft*
- *GetReconstructionRoiWidth*
- *GetReconstructionRoiTop*
- *ResetReconstructionRoi*

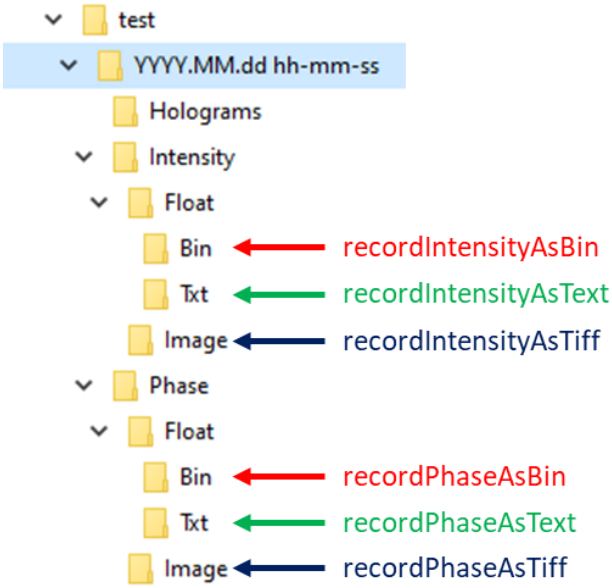
SetReconstructionToDiskDataType

Choose what kind of files are reconstructed and saved or not using the ***Sequence: Reconstruction to disk*** window. If the path is valid (contains a Holograms folder and a timestamp file), the phase and intensity buttons will be enabled.

Koala UI equivalent:
Corresponds to the Phase **8.bits Tiff, Bin, Text** and Intensity **8.bits Tiff, Bin, Text** buttons in the ***Sequence: Reconstruction to disk*** window.



Example: In the figure above, the Phase and Intensity **Bin, 8-bits Tiff and Text** buttons are activated in blue (as compared to e.g. the **Start Reconstruction to Disk** button that is not activated), e.g. we will record phase and intensity in .bin, .tiff and .txt formats.



Parameters:

- recordPhaseAsBin (*Boolean*): `true` to start the phase reconstruction in **.bin** format in the folder **RECORD_PATH\YYYY.MM.dd hh-mm-ss\Phase\Float\Bin**, `false` otherwise.
- recordPhaseAsText (*Boolean*): `true` to start the phase reconstruction in **.txt** format in the folder **RECORD_PATH\YYYY.MM.dd hh-mm-ss\Phase\Float\Txt**, `false` otherwise.
- recordPhaseAsTiff (*Boolean*): `true` to start the phase reconstruction in **.tiff** format in the folder **RECORD_PATH\YYYY.MM.dd hh-mm-ss\Phase\Image**, `false` otherwise.
- recordIntensityAsBin (*Boolean*): `true` to start the intensity reconstruction in **.bin** format in the folder **RECORD_PATH\YYYY.MM.dd hh-mm-ss\Intensity\Float\Bin**, `false` otherwise.
- recordIntensityAsText (*Boolean*): `true` to start the intensity reconstruction in **.txt** format in the folder **RECORD_PATH\YYYY.MM.dd hh-mm-ss\Intensity\Float\Txt**, `false` otherwise.
- recordIntensityAsTiff (*Boolean*): `true` to start the intensity reconstruction in **.tiff** format in the folder **RECORD_PATH\YYYY.MM.dd hh-mm-ss\Intensity\Image**, `false` otherwise.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in
- The sequence reconstruction to disk window must be opened.

- Path cannot be left empty.

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

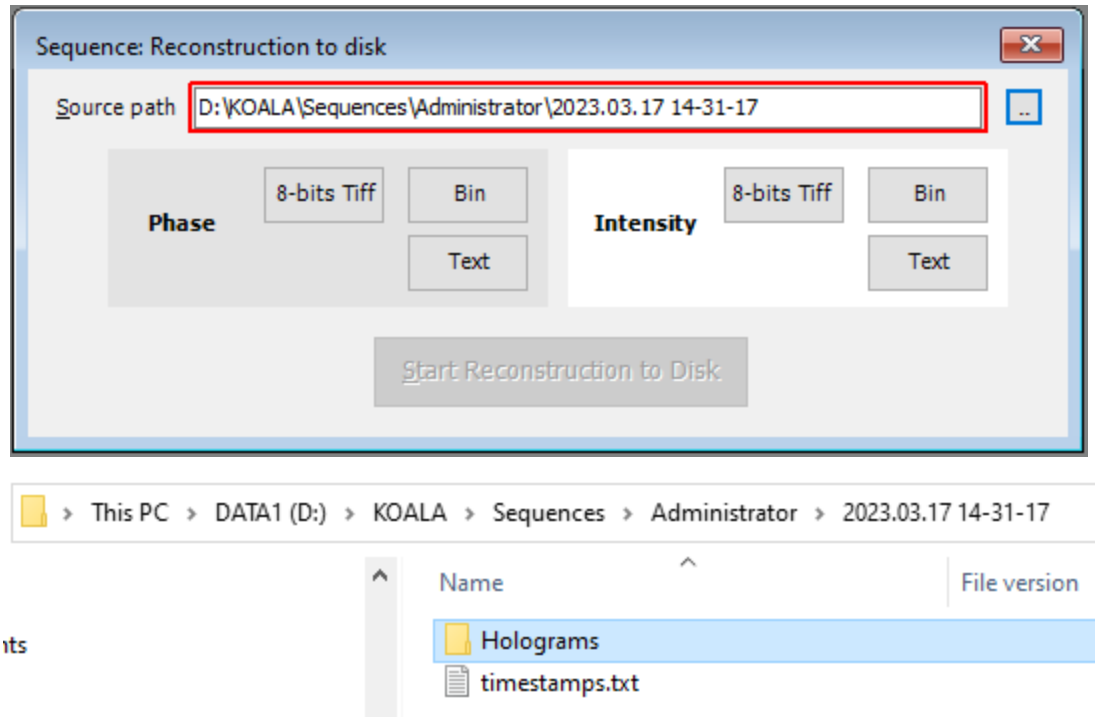
See also:

- *CloseReconstructionToDiskSequenceWin*
- *OpenReconstructionToDiskSequenceWin*
- *SetReconstructionToDiskSequencePath*
- *StartReconstructionToDisk*

SetReconstructionToDiskSequencePath

Set the path to the sequence to be reconstructed using the ***Sequence: Reconstruction to disk*** window. If the path is valid (contains a Holograms folder and a timestamp file), the phase and intensity buttons will be enabled.

Koala UI equivalent:
Corresponds to defining the source path in the ***Sequence: Reconstruction to disk*** window.



Parameters:

- Path (string): Full path to the folder where to read the holograms sequence.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in
- The window must be opened.
- The path must be valid (it cannot be left empty, the drive must exist and no invalid characters must be present).

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *CloseReconstructionToDiskSequenceWin*

- *OpenReconstructionToDiskSequenceWin*
- *SetReconstructionToDiskDataType*
- *StartReconstructionToDisk*

SetSourceState

Sets a source ON or OFF.

Koala UI equivalent:
Clicking on one of the *Sources* buttons in the main menu, or on one of the *ON* or *OFF* buttons in the *Laser Sources window*.

Parameters:

- srcId (Int32): The id of the source (logical or physical according to *useLogicalId*)
- state (Boolean): Set to `true` to switch the source ON, `false` to switch it OFF
- useLogicalId (Boolean): Set to `true` to use logical id, to `false` to use physical id. Optional, default value is `true`.

Return: void (nothing)

Requirements:

- The laser source controller must be available
- If logical ids are used, a configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

See also:

- -

SetUnwrap2DMethod

Sets the method used for unwrapping.
Note that unwrapping will not take place until it has been enabled with *SetUnwrap2DState*.

Koala UI equivalent:
Selecting one of the menu option in the *Settings* ⟶ *Phase unwrapping* menu

Parameters:

- method (Int32): Id of the method to use for unwrapping. Possible values:
  - 0: Discrete Cosine Transform (DCT)
  - 1: Path-following (also known as Quality Path)
  - 2: Preconditioned Conjugate Gradient (PCG). Do not use, deprecated.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *GetUnwrap2DState*
- *SetUnwrap2DState*

SetUnwrap2DState

Enables or disables the unwrapping of the phase.

Koala UI equivalent:

Corresponds to clicking on the Unwrap  button of the *Phase Image window*.

Parameters:

- state (*Boolean*): `true` to enable the unwrapping, `false` to disable it. Optional, default value is `false`.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- The phase window must be opened and must contain an image
- A reconstruction object must be created
- A sequence must not be playing
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

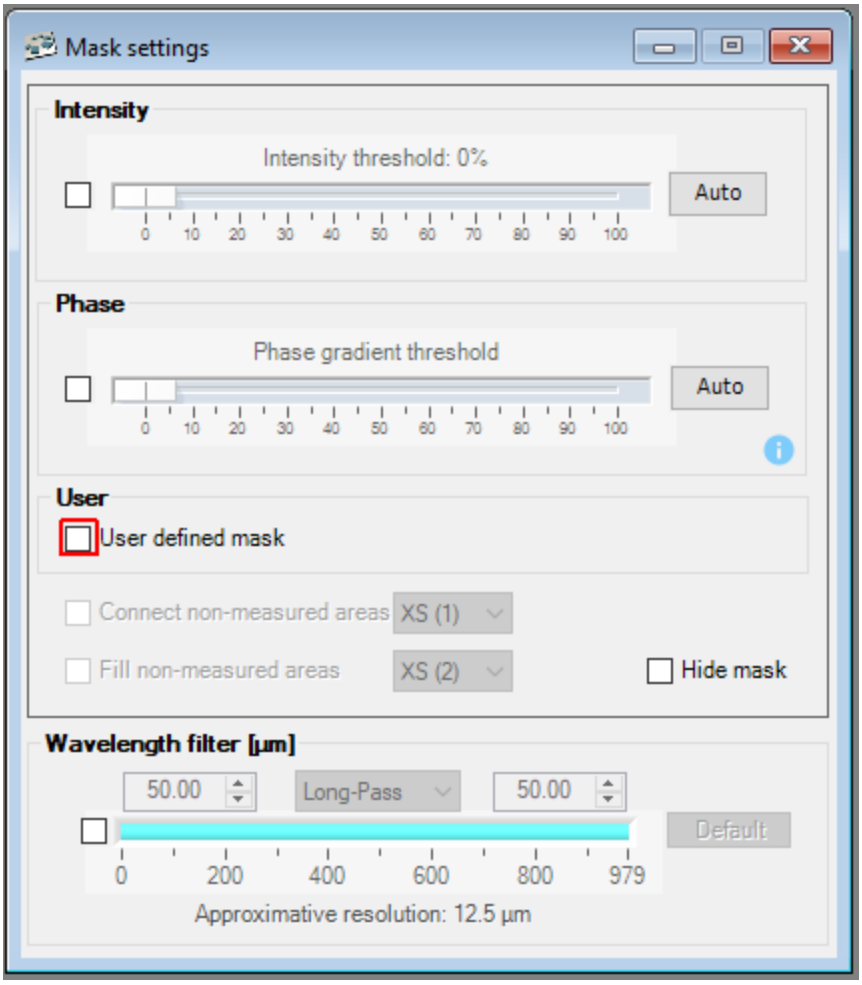
See also:

- *GetUnwrap2DState*
- *SetUnwrap2DMethod*

SetUserDefinedMaskState

Enables or disables the user defined mask filter on the phase.

Koala UI equivalent:
Corresponds to clicking on the check button **User Defined Mask** of the *Mask Settings Window*.



Parameters:

- state (*Boolean*): `true` to enable the user defined mask filter, `false` to disable it.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in
- The Mask settings window must be opened

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

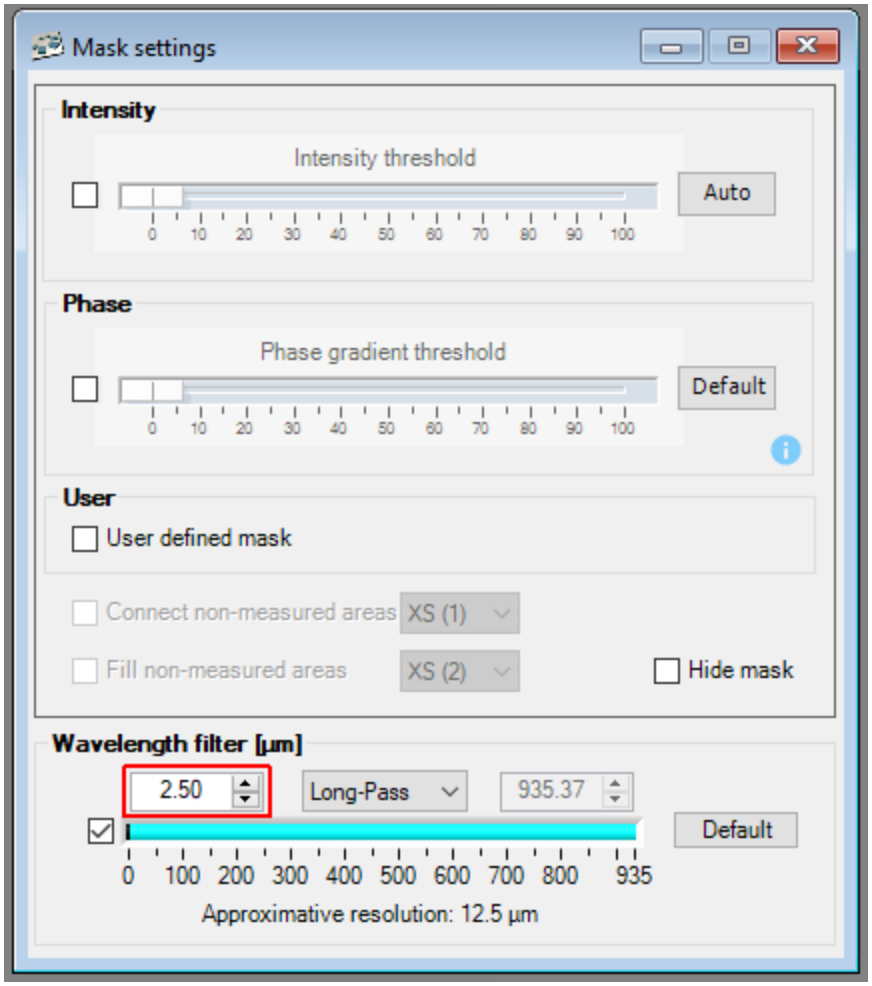
See also:

- *AddEllipticalUserDefinedMaskZoneToPhase*
- *AddRectangularUserDefinedMaskZoneToPhase*
- *CloseMaskSettingsWin*
- *OpenMaskSettingsWin*
- *ResetUserDefinedMaskZone*
- *SetInteractionModeWhenAddingMaskZone*

SetWavelengthFilterMinimalCutOffValue

Sets the wavelength filter minimal cutoff value. The minimum cutoff value depends on the pixel size: it is 2 pixels (Nyquist).

Koala UI equivalent:
Corresponds to setting the minimal cutoff value of the *Mask Settings Window*.



Parameters:

- minimal (*float*): value of the minimal cutoff in um.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in

- The mask window must be opened

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

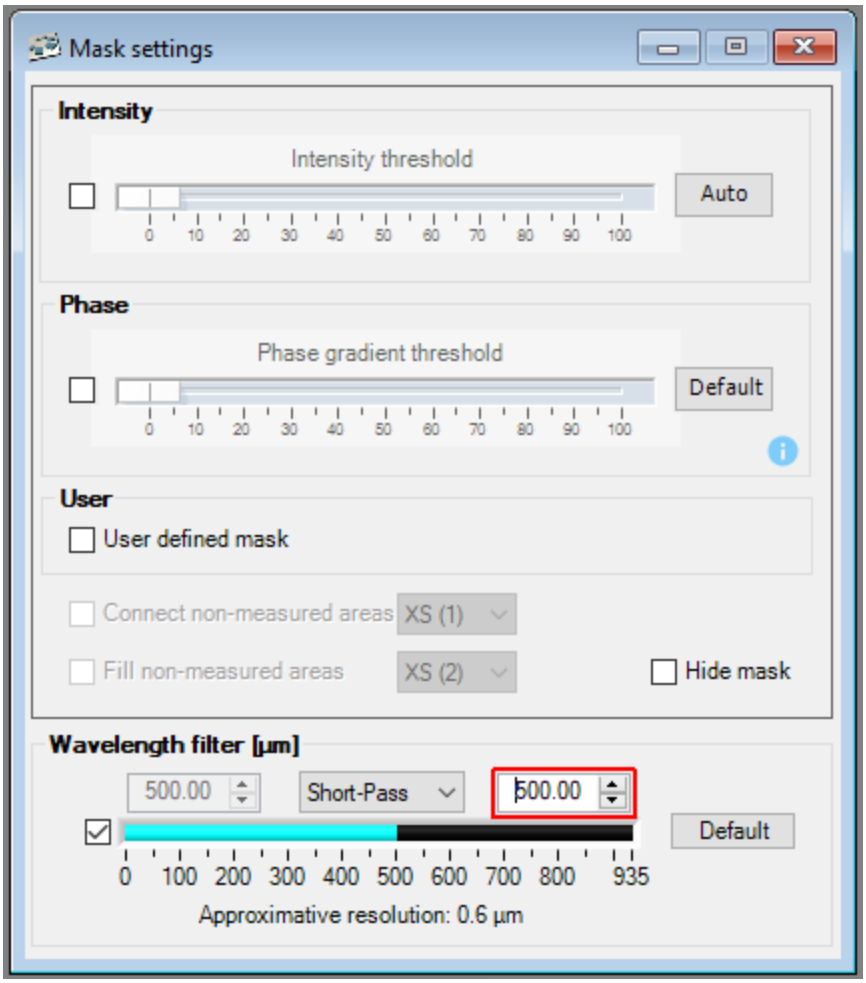
See also:

- *CloseMaskSettingsWin*
- *OpenMaskSettingsWin*
- *SetWavelengthFilterMaximalCutOffValue*
- *SetWavelengthFilterState*
- *SetWavelengthFilterType*

SetWavelengthFilterMaximalCutOffValue

Sets the wavelength filter maximal cutoff value. The maximum cutoff value depends on the image size.

Koala UI equivalent:
Corresponds to setting the maximal cutoff value of the *Mask Settings Window*.



Parameters:

- maximal (*float*): value of the maximal cutoff in um.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in
- The mask window must be opened

- The Wavelength filter must be enabled

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

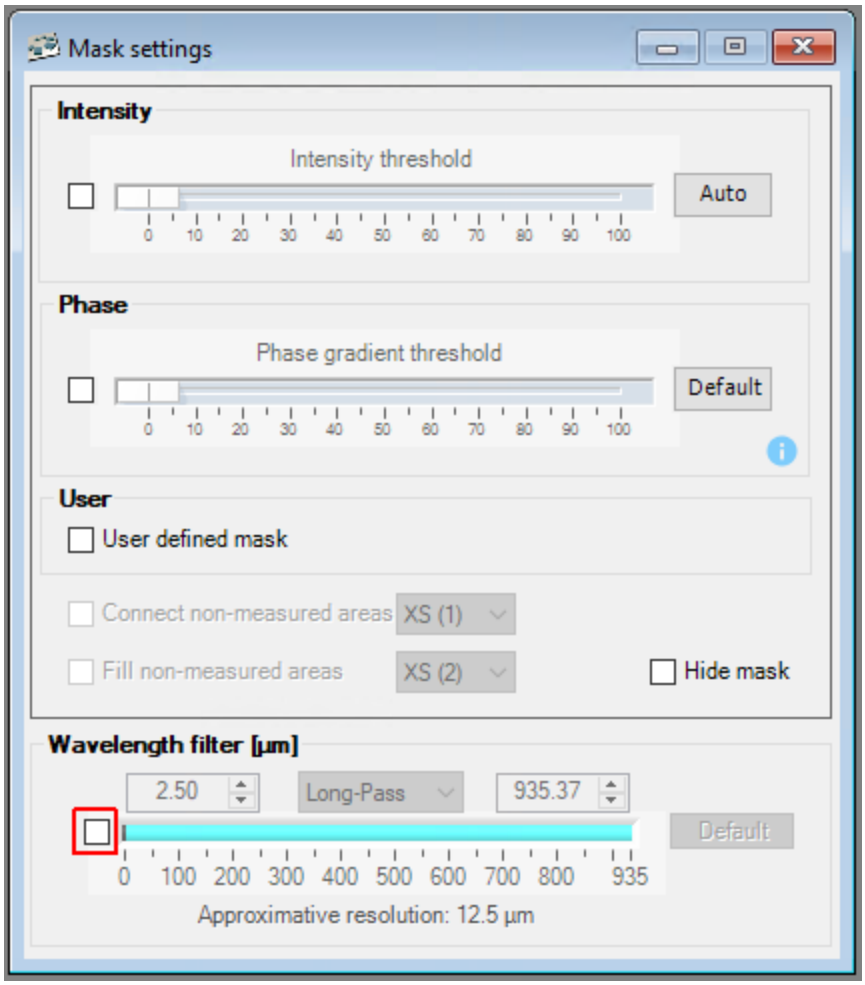
See also:

- *CloseMaskSettingsWin*
- *OpenMaskSettingsWin*
- *SetWavelengthFilterMinimalCutOffValue*
- *SetWavelengthFilterState*
- *SetWavelengthFilterType*

SetWavelengthFilterState

Enables or disables the wavelength filter on the phase.

Koala UI equivalent:
Corresponds to clicking on the checkbox of the *Mask Settings Window*.



Parameters:

- state (*Boolean*): `true` to enable the wavelength filter, `false` to disable it. Optional, default value is `false`.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

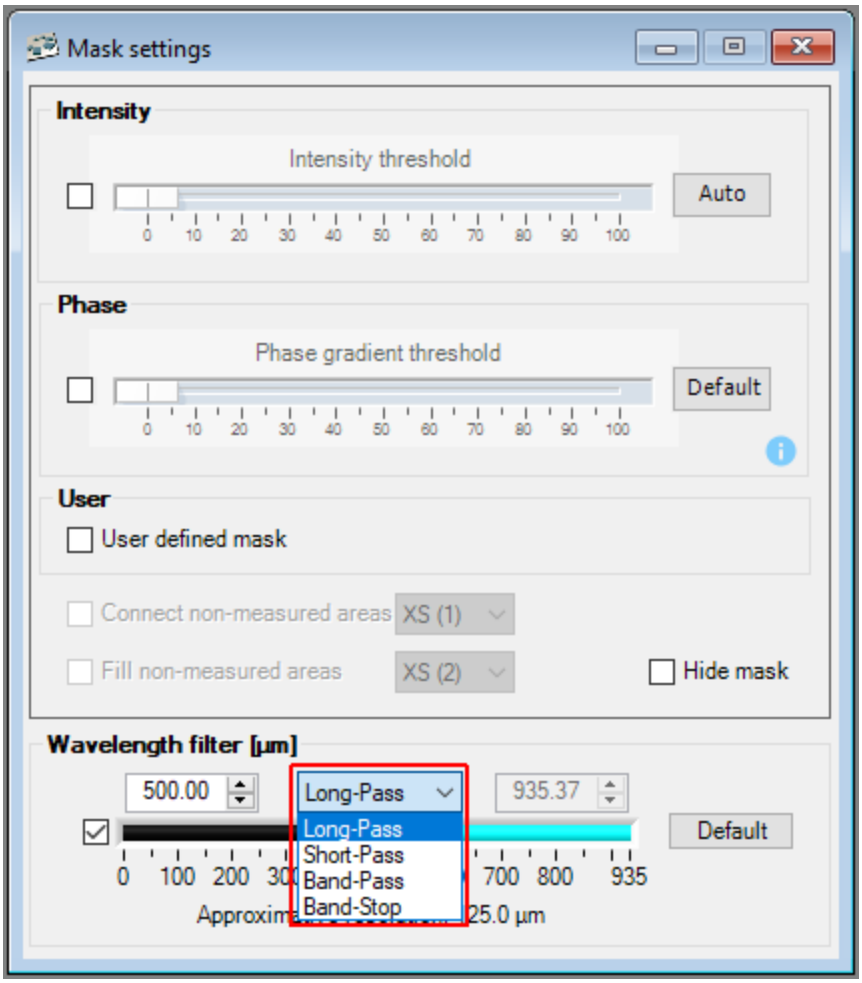
See also:

- *CloseMaskSettingsWin*
- *OpenMaskSettingsWin*
- *SetWavelengthFilterMinimalCutOffValue*
- *SetWavelengthFilterMaximalCutOffValue*
- *SetWavelengthFilterType*

SetWavelengthFilterType

Set the type of wavelength filter on the phase.

Koala UI equivalent:
Corresponds to selecting the filter type in the combobox of the *Mask Settings Window*.



Parameters:

- type (*Int32*): can be values [1,2,3,4]. Select:
  - **1** for Long-pass type filter,
  - **2** for Short-pass type filter,
  - **3** for Band-pass type filter,
  - **4** for Band-stop type filter.

Return: void (nothing)

Requirements:

- A configuration must be loaded

- A user must be logged in
- The mask window must be opened
- The Wavelength filter must be enabled

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error
- Type entered is not in the valid values [1,2,3,4] ⇒ *Invalid Operation* error

See also:

- *CloseMaskSettingsWin*
- *OpenMaskSettingsWin*
- *SetWavelengthFilterMinimalCutOffValue*
- *SetWavelengthFilterMaximalCutOffValue*
- *SetWavelengthFilterState*

SingleReconstruction

Acquires an image (or several if temporal averaging is on) and reconstruct it.

Koala UI equivalent:
(not applicable)

(No parameters)

Return: void (nothing)

Requirements:

- A configuration must be loaded
- Live mode must be available
- The intern camera must be available
- Video mode must not be running
- A strobo measurement must not be running
- The sequence reconstruction to display window must not be opened
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation*

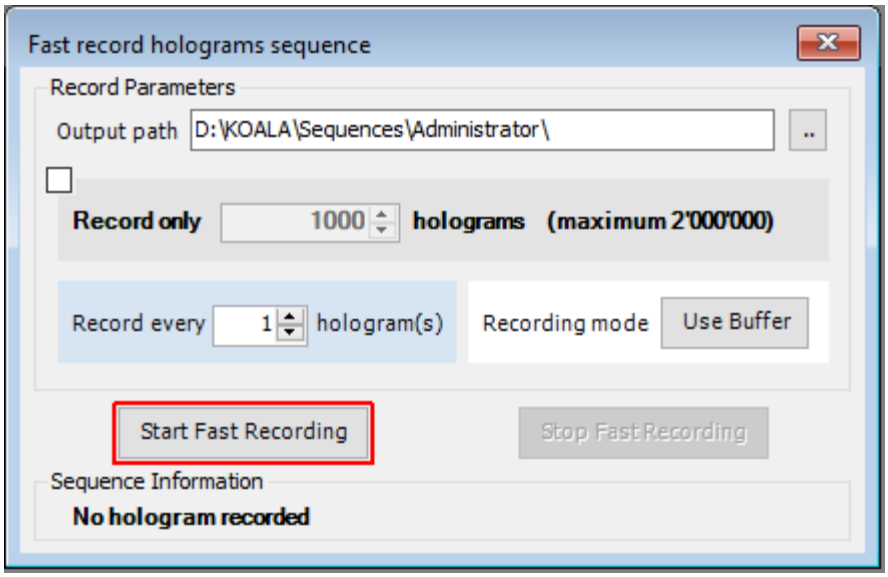
See also:

- *Acquisition2L*

StartFastHologramsSequenceRecord

Start the sequence recording using the *Fast record holograms sequence* window.

Koala UI equivalent:
Corresponds to clicking the button **Start Fast Recording** in the **Fast record holograms sequence** window.



Parameters:

- None.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in
- The Fast Record Holograms sequence window must be opened.

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

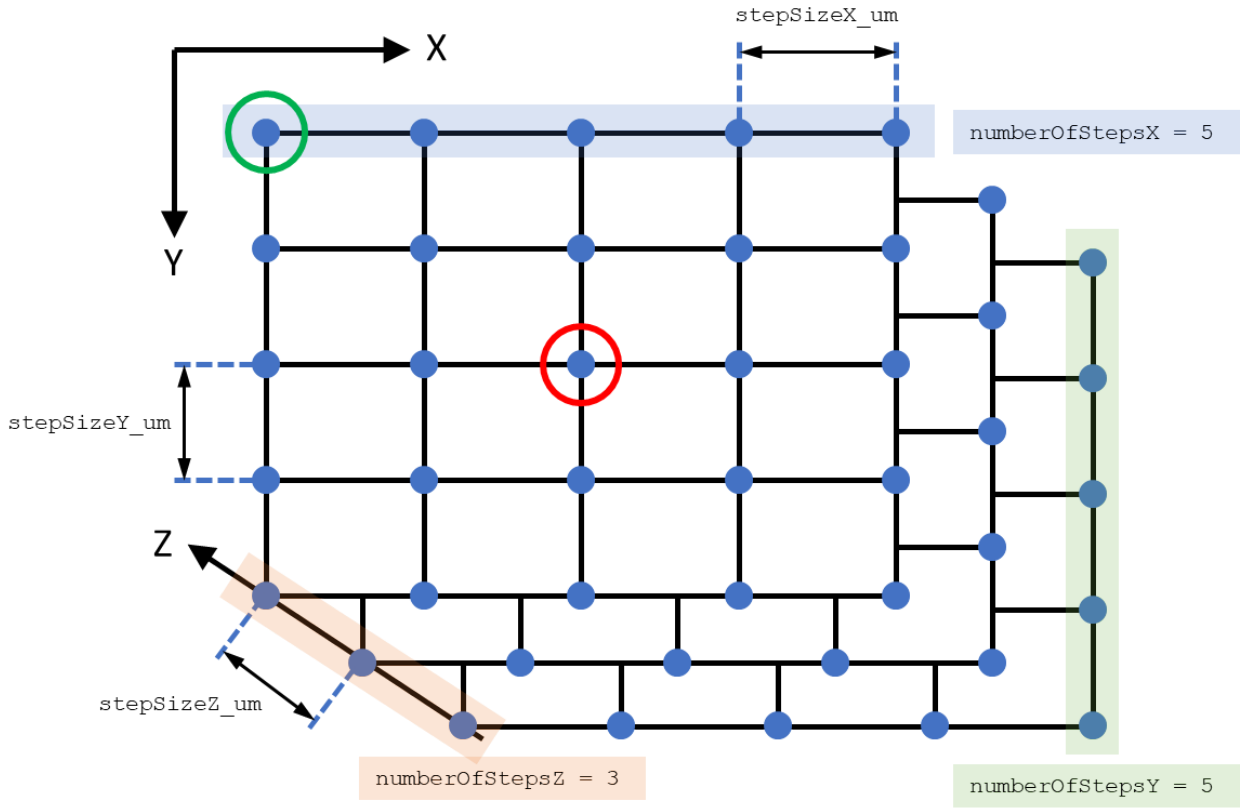
- *CloseFastHologramsRecordWin*
- *OpenFastHologramsRecordWin*
- *SetFastHologramsSequenceRecordingModeBuffer*
- *SetFastHologramsSequenceRecordNumberOfHolograms*
- *SetFastHologramsSequenceRecordPath*
- *StopFastHologramsSequenceRecord*

StartLateralScanning

Starts a recording of data to do a lateral scanning acquisition.

! IMPORTANT ! The acquisition must be stopped before starting the recording. (See ResetGrab)

The user describes a displacement grid for the stage using all the parameters for positions and step size in the X,Y and Z directions. The minimal number of positions in each direction is 1, i.e if you only move in X and Y direction, the number of positions in the Z direction must be set to 1. The stage goes back to its starting position at the end of the scan.

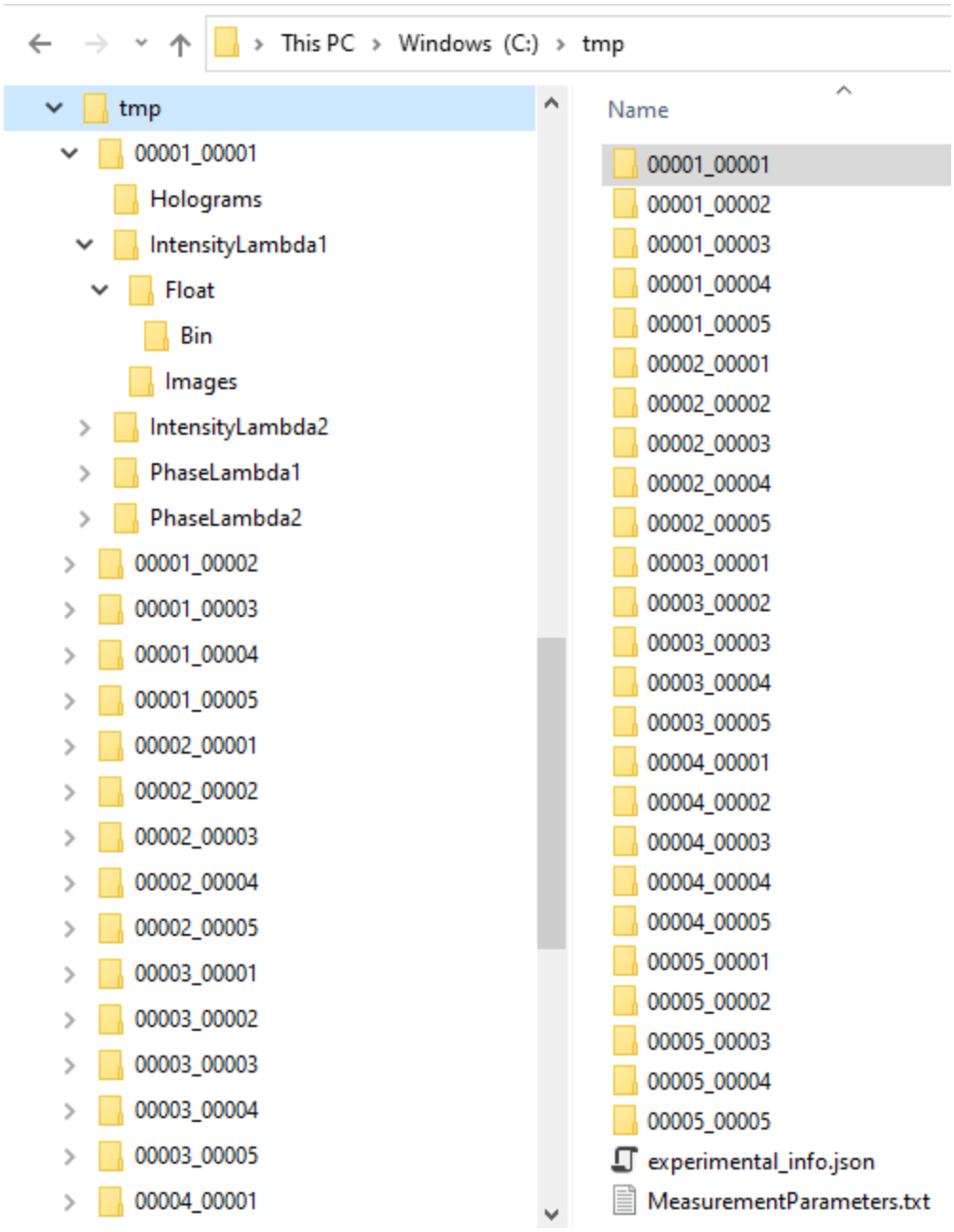


Example of a grid for 5 positions in X and Y direction and 3 positions in Z direction

The **red circle** indicates the starting position if the user chooses to perform an acquisition around the starting position. The **green circle** indicates the starting position with respect to the grid if the user chooses to perform an acquisition from this position to the right in X direction and bottom in Y direction.
For the acquisition, we only use the laser sources selected for the configuration. The user has then the option to supersede that setting and activate or not the available laser sources using the parameters *isLambdaXLaserUsed*.

As for a Koala cycle, the *experimental_info.json* file describing the experimental setup and the *MeasurementParameters.txt* file describing the DHM status at the time of the acquisition are saved in the saving folder.

An example of the saved folder structure is shown below:



Koala UI equivalent:
-

Parameters:

- *startPositionX_um* (Double): absolute starting position for X axis in [um] as shown in the XYZ stage window.
- *startPositionY_um* (Double): absolute starting position for Y axis in [um] as shown in the XYZ stage window.
- *startPositionZ_um* (Double): absolute starting position for Z axis in [um] as shown in the XYZ stage window.
- *numberOfPositionsX* (Int32): number of positions in the grid in the X axis direction. The minimum number of positions is 1.
- *numberOfPositionsY* (Int32): number of positions in the grid in the Y axis direction. The minimum number of positions is 1.
- *numberOfPositionsZ* (Int32): number of positions in the grid in the Z axis direction. The minimum number of positions is 1.
- *stepSizeX_um* (Double): size of one step in the X axis direction in [um].
- *stepSizeY_um* (Double): size of one step in the Y axis direction in [um].
- *stepSizeZ_um* (Double): size of one step in the Z axis direction in [um].
- *saveHolograms* (Boolean): if **true** save holograms during acquisition.
- *saveIntensity* (Boolean): if **true** save intensity during acquisition.
- *savePhase* (Boolean): if **true** save phase during acquisition.
- *saveAsBin* (Boolean): if **true** save phase and intensity in **.bin** format during acquisition.
- *saveAsTxt* (Boolean): if **true** save phase and intensity in **.txt** format during acquisition.
- *saveAsTif* (Boolean): if **true** save phase and intensity in **.tif** format during acquisition.
- *saveLambda1* (Boolean): if **true** save phase and intensity for laser source lambda 1 separately.
- *saveLambda2* (Boolean): if **true** save phase and intensity for laser source lambda 2 separately.
- *saveLambda3* (Boolean): if **true** save phase and intensity for laser source lambda 3 separately.
- *isLambda1LaserUsed* (Boolean): if **true** use laser source lambda 1 for acquisition.
- *isLambda2LaserUsed* (Boolean): if **true** use laser source lambda 2 for acquisition.
- *isLambda3LaserUsed* (Boolean): if **true** use laser source lambda 3 for acquisition.
- *centerScanOnStartPosition* (Boolean): if **true**, the displacement grid is centered on the current starting position (**red circle** in the figure above). If **false**, the starting position is the top left corner of the grid (**green circle** in the figure above).
- *savePath* (String): absolute path where the data will be saved. If left empty, the data will be saved in a subfolder with its current date and time in the **My Documents** folder.
- numberOfAcquisitionsPerPosition (Int32): number of acquisitions to be done at each position.
- *isAcquisitionAlternate* (Boolean): if **true**, the hologram acquisition is done alternately for 2-wavelengths configuration, so we obtain a hologram for lambda 1 (folder HologramsLambda1) and a hologram for lambda 2 (folder HologramsLambda2). If **false**, we obtain only one hologram for lambda 1-2 (folder Holograms).

Return: void (nothing)

Requirements:

- A user must be logged in
- A configuration must be loaded
- The Hologram window must be opened.
- The Phase window must be opened.
- The Intensity window must be opened.
- The acquisition must be stopped before starting the recording.

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

C# Example:

```
//css_reference LynceeTec.KoalaRemote.Client.dll;
using System;
using System.IO;
using LynceeTec.KoalaRemote.Client;

//*****************************************************************************
// NOTES:
// - This script is designed to work with a 2 wavelengths configuration.
//   If you use a single wavelength configuration, you will have to comment
//   some of the lines for the script to run.
//
// - The script demonstrate the procedure to record a set of data to produce
//   a lateral scanning.
//
// - You might need to adapt some of the parameters (such as the save/load
//   location or the configuration number) for the script to work on your
//   system with your database
//*****************************************************************************

class TestSampleScriptForLateralScanning : IKoalaScript
{
   public void Run(KoalaRemoteClient host)
   {

      //Open a 2 wavelengths configuration.
      host.OpenConfig(186);

      //Open main display windows
      host.OpenHoloWin();
      host.OpenPhaseWin();
      host.OpenIntensityWin();
```

```
//Do an acquisition for test
host.Acquisition2L();

//Get the original stage position
double[] startPosition = new double[4];
host.GetAxesPosMu(startPosition);

double startPositionX_um = startPosition[0];
double startPositionY_um = startPosition[1];
double startPositionZ_um = startPosition[2];

//Define the displacement grid size
int numberOfPositionsX = 5;
int numberOfPositionsY = 5;
int numberOfPositionsZ = 1;

double stepSizeX_um = 1000;
double stepSizeY_um = 1000;
double stepSizeZ_um = 0.0;

//Define the saving options
bool saveHolograms = true;
bool saveIntensity = true;
bool savePhase = true;
bool saveAsBin = true;
bool saveAsTxt = false;
bool saveAsTif = true;
bool saveLambda1 = true;
bool saveLambda2 = true;
bool saveLambda3 = false;

//Define which wavelength can be used
bool isLambda1LaserUsed = true;
bool isLambda2LaserUsed = true;
bool isLambda3LaserUsed = true;

//Is the lateral scanning centered on the start position or not ?
bool centerScanOnStartPosition = true;

//Indicate where you save your acquisition data
string savePath = @"C:\tmp";

//Give the number of acquisitions to be realized at each position
int numberOfAcquisitionsPerPosition = 2;

//In case of 2-wavelengths configuration, it indicates if we save
//a hologram for each wavelength separately or not
```

```
        bool isAcquisitionAlternate = false;

        //Start the lateral scanning acquisition
        host.StartLateralScanning(startPositionX_um, startPositionY_um, startPositionZ_um,
            numberOfPositionsX, numberOfPositionsY, numberOfPositionsZ,
            stepSizeX_um, stepSizeY_um, stepSizeZ_um,
            saveHolograms, saveIntensity, savePhase, saveAsBin, saveAsTxt, saveAsTif,
saveLambda1, saveLambda2, saveLambda3, isLambda1LaserUsed, isLambda2LaserUsed,
isLambda3LaserUsed,
            centerScanOnStartPosition,
            savePath,
            numberOfAcquisitionsPerPosition,
            isAcquisitionAlternate);
    }
}
```

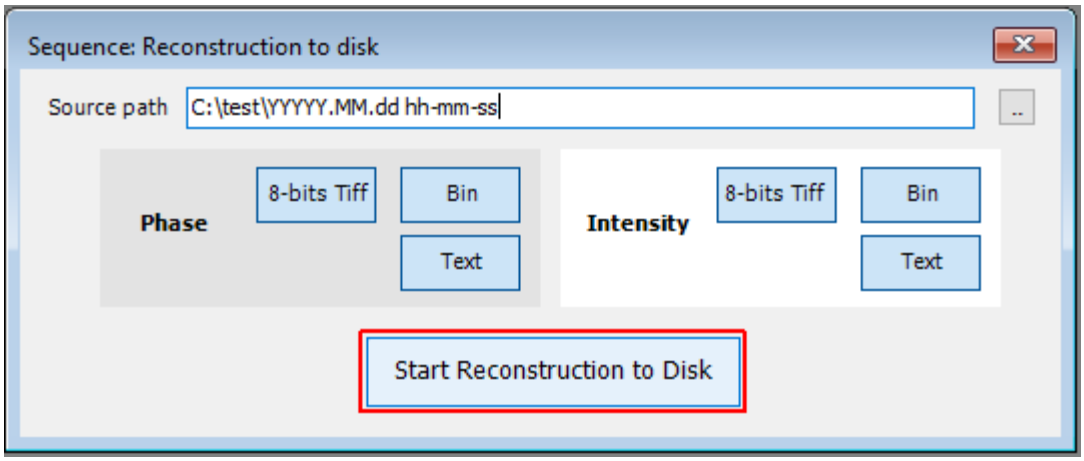
See also:

- *ResetGrab*
- *StartScanningFromPoints*
- *StartSpiralScanning*

StartReconstructionToDisk

Start the sequence reconstruction using the *Sequence: Reconstruction to disk* window.

Koala UI equivalent:
Corresponds to clicking the button **Start Reconstruction to Disk** in the **Sequence: Reconstruction to disk** window.



Parameters:

- None.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in
- The *Sequence: Reconstruction to disk* window must be opened.
- The path must be valid (it cannot be left empty, the drive must exist and no invalid characters must be present).
- At least one type of Phase or Intensity must be selected.

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *CloseReconstructionToDiskSequenceWin*
- *OpenReconstructionToDiskSequenceWin*
- *SetReconstructionToDiskDataType*
- *SetReconstructionToDiskSequencePath*

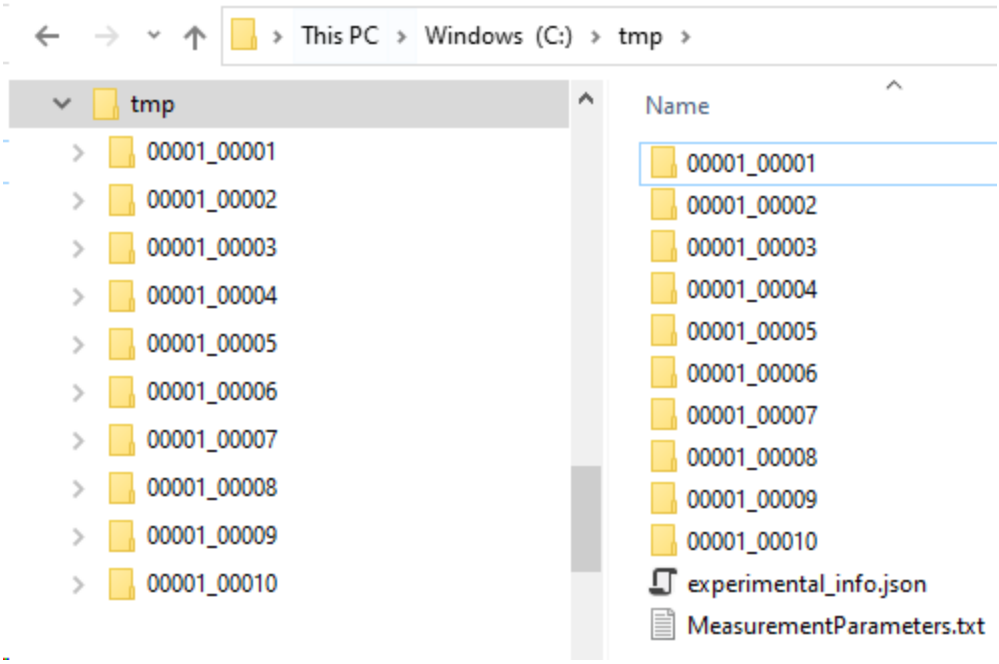
StartScanningFromPoints

Starts a recording of data to do a scanning acquisition from points coordinates. The coordinates are given in pixels relative to the starting position.

! IMPORTANT ! The acquisition must be stopped before starting the recording. (See ResetGrab)

The user gives a list of positions with 3 separated inputs for each axis (X, Y, Z) coordinates. The format is described hereafter.
The stage goes back to its starting position at the end of the scan.
For the acquisition, we only use the laser sources selected for the configuration. The user has then the option to supersede that setting and activate or not the available laser sources using the parameters *isLambdaXLaserUsed*.
As for a Koala cycle, the *experimental_info.json* file describing the experimental setup and the *MeasurementParameters.txt* file describing the DHM status at the time of the acquisition are saved in the saving folder.

An example of the saved folder structure for 10 positions is shown below:



Koala UI equivalent:
-

Parameters:

- *startPositionX_um* (Double): absolute starting position for X axis in [um] as shown in the XYZ stage window.

- *startPositionY_um* (Double): absolute starting position for Y axis in [um] as shown in the XYZ stage window.
- *startPositionZ_um* (Double): absolute starting position for Z axis in [um] as shown in the XYZ stage window.
- *totalNumberOfPositions* (Int32): total number of positions. The minimum number of positions is 1. The total number of positions must correspond to the number of relative positions indicated hereafter, when the corresponding relativePositions fields are parsed.
- *relativePositionsXList_um* (string): list of coordinates in the X axis direction in [um], relative to the starting position. Example: "[0, 1, 2]" or "0, 1, 2" for a total number of positions set to 3.
- *relativePositionsYList_um* (string): list of coordinates in the Y axis direction in [um], relative to the starting position. Example: "[0, 1, 2]" or "0, 1, 2" for a total number of positions set to 3.
- *relativePositionsZList_um* (string): list of coordinates in the Z axis direction in [um], relative to the starting position. Example: "[0, 1, 2]" or "0, 1, 2" for a total number of positions set to 3.
- *saveHolograms* (Boolean): if **true** save holograms during acquisition.
- *saveIntensity* (Boolean): if **true** save intensity during acquisition.
- *savePhase* (Boolean): if **true** save phase during acquisition.
- *saveAsBin* (Boolean): if **true** save phase and intensity in **.bin** format during acquisition.
- *saveAsTxt* (Boolean): if **true** save phase and intensity in **.txt** format during acquisition.
- *saveAsTif* (Boolean): if **true** save phase and intensity in **.tif** format during acquisition.
- *saveLambda1* (Boolean): if **true** save phase and intensity for laser source lambda 1 separately.
- *saveLambda2* (Boolean): if **true** save phase and intensity for laser source lambda 2 separately.
- *saveLambda3* (Boolean): if **true** save phase and intensity for laser source lambda 3 separately.
- *isLambda1LaserUsed* (Boolean): if **true** use laser source lambda 1 for acquisition.
- *isLambda2LaserUsed* (Boolean): if **true** use laser source lambda 2 for acquisition.
- *isLambda3LaserUsed* (Boolean): if **true** use laser source lambda 3 for acquisition.
- *savePath* (String): absolute path where the data will be saved. If left empty, the data will be saved in a subfolder with its current date and time in the **My Documents** folder.
- numberOfAcquisitionsPerPosition (Int32): number of acquisitions to be done at each position.
- *isAcquisitionAlternate* (Boolean): if **true**, the hologram acquisition is done alternately for 2-wavelengths configuration, so we obtain a hologram for lambda 1 (folder HologramsLambda1) and a hologram for lambda 2 (folder HologramsLambda2). If **false**, we obtain only one hologram for lambda 1-2 (folder Holograms).

Return: void (nothing)

Requirements:

- A user must be logged in
- A configuration must be loaded
- The Hologram window must be opened.
- The Phase window must be opened.
- The Intensity window must be opened.
- The acquisition must be stopped before starting the recording.

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

C# Example:

```
//css_reference LynceeTec.KoalaRemote.Client.dll;
using System;
using System.IO;
using LynceeTec.KoalaRemote.Client;

//*****************************************************************************
// NOTES:
// - This script is designed to work with a 2 wavelengths configuration.
//   If you use a single wavelength configuration, you will have to comment
//   some of the lines for the script to run.
//
// - The script demonstrate the procedure to record a set of data to produce
//   a scanning from a list of points.
//
// - You might need to adapt some of the parameters (such as the save/load
//   location or the configuration number) for the script to work on your
//   system with your database
//*****************************************************************************

class TestSampleScriptForScanningFromPoints : IKoalaScript
{
   public void Run(KoalaRemoteClient host)
   {

      //Open a 2 wavelengths configuration.
      host.OpenConfig(141);

      //Open main display windows
      host.OpenHoloWin();
      host.OpenPhaseWin();
      host.OpenIntensityWin();
```

```
        //Do an acquisition for test
        host.Acquisition2L();

        //Get the original stage position
        double[] startPosition = new double[4];
        host.GetAxesPosMu(startPosition);

        double startPositionX_um = startPosition[0];
        double startPositionY_um = startPosition[1];
        double startPositionZ_um = startPosition[2];

        //Define the scanning parameters
        int totalNumberOfPositions = 11;
        string relativePositionsXList_um = "0 ,1, 2, 3, 4, 5, 6, 7, 8, 9, 10";
        string relativePositionsYList_um = "0 ,1, 2, 3, 4, 5, 6, 7, 8, 9, 10";
        string relativePositionsZList_um = string.Empty;

        //Define the saving options
        bool saveHolograms = true;
        bool saveIntensity = true;
        bool savePhase = true;
        bool saveAsBin = true;
        bool saveAsTxt = false;
        bool saveAsTif = true;
        bool saveLambda1 = true;
        bool saveLambda2 = false;
        bool saveLambda3 = false;

        //Define which wavelength will be used
        bool isLambda1LaserUsed = true;
        bool isLambda2LaserUsed = true;
        bool isLambda3LaserUsed = true;

        //Indicate where you save your acquisition data
        string savePath = @"C:\tmp";

        //Give the number of acquisitions to be realized at each position
        int numberOfAcquisitionsPerPosition = 1;

        //In case of 2-wavelengths configuration, it indicates if we save
        //a hologram for each wavelength separately or not
        bool isAcquisitionAlternate = false;

        //Start the scanning acquisition
        host.StartScanningFromPoints(startPositionX_um, startPositionY_um,
startPositionZ_um,
```

```
            totalNumberOfPositions,
            relativePositionsXList_um, relativePositionsYList_um, relativePositionsZList_um,
            saveHolograms, saveIntensity, savePhase, saveAsBin, saveAsTxt, saveAsTif,
            saveLambda1, saveLambda2, saveLambda3,
            isLambda1LaserUsed, isLambda2LaserUsed, isLambda3LaserUsed,
            savePath, numberOfAcquisitionsPerPosition, isAcquisitionAlternate);
    }
}
```

See also:

- *ResetGrab*
- *StartLateralScanning*
- *StartSpiralScanning*

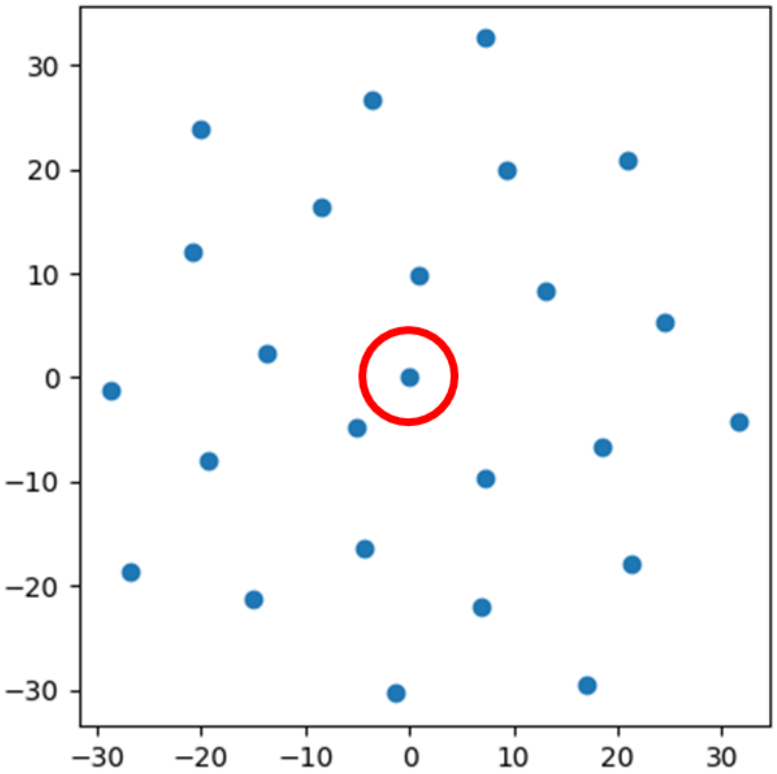
StartSpiralScanning

Starts a recording of data to do a scanning acquisition following a Fermat spiral (see https://en.wikipedia.org/wiki/Fermat%27s_spiral for more details and https://www.flyingpudding.com/projects/florets/applet/ for an example to see the stage positions calculated).

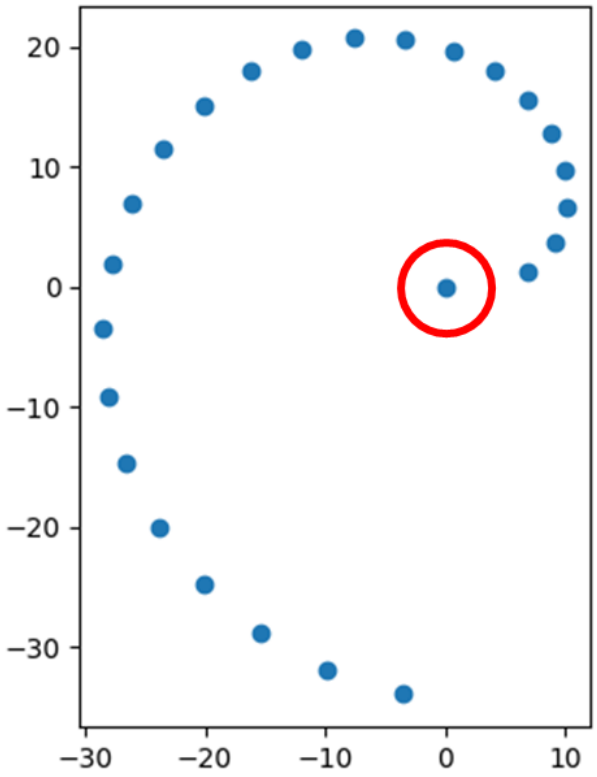
! IMPORTANT ! The acquisition must be stopped before starting the recording. (See ResetGrab)

The user describes stage positions using all the parameters to characterize a Fermat spiral, i.e. its basis angle, and the radius applied between 2 positions. The internal basis angle (which can be selected as an option, see below) is the one corresponding to the golden angle, in order to give a distributed pattern for acquisition.
The stage goes back to its starting position at the end of the scan.

Example of the 25 positions for the golden angle with a radius equal to 10 (the starting point is indicated in the **red circle**, axes are in um):

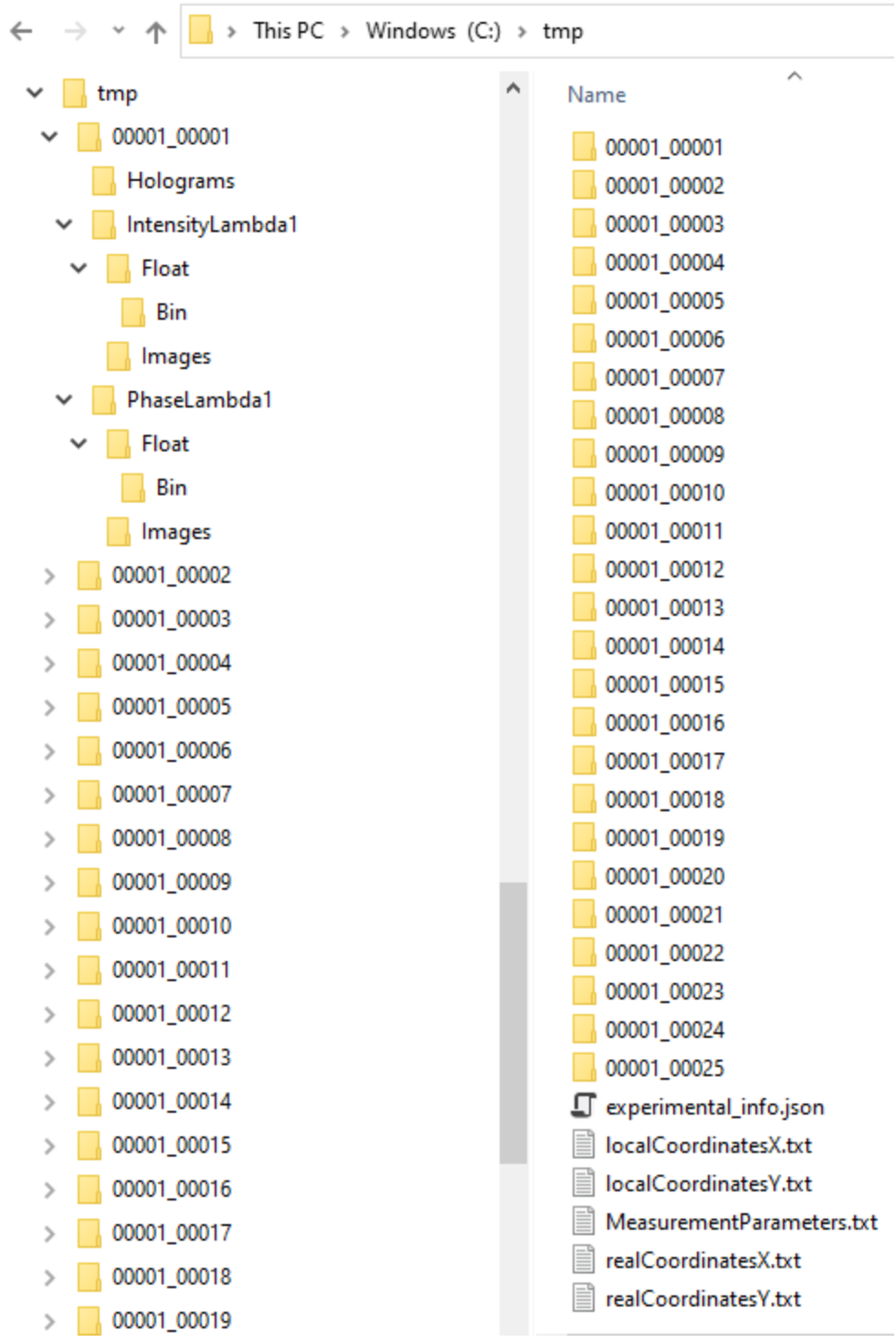


Example of the 25 positions for the angle equal to 11 degrees with a radius equal to 10 (the starting point is indicated in the **red circle**, axes are in um):



For the acquisition, we only use the laser sources selected for the configuration. The user has then the option to supersede that setting and activate or not the available laser sources using the parameters *isLambdaXLaserUsed*.
As for a Koala cycle, the *experimental_info.json* file describing the experimental setup and the *MeasurementParameters.txt* file describing the DHM status at the time of the acquisition are saved in the saving folder.
Here, we also save the coordinates for X and Y axes in 2 separated files to be able to verify the distribution of the points as shown above. *localCoordinatesX.txt* and *localCoordinatesY.txt* contain the coordinates relative to the stage starting point and are in [um]. *realCoordinatesX.txt* and *realCoordinatesY.txt* contain the absolute coordinates of the stage in [um].

An example of the saved folder structure for 25 positions and only lambda1 is shown below:



Koala UI equivalent:

-

Parameters:

- *startPositionX_um* (Double): absolute starting position for X axis in [um] as shown in the XYZ stage window.
- *startPositionY_um* (Double): absolute starting position for Y axis in [um] as shown in the XYZ stage window.
- *startPositionZ_um* (Double): absolute starting position for Z axis in [um] as shown in the XYZ stage window.
- *totalNumberOfPositions* (Int32): total number of positions. The minimum number of positions is 1. The total number of positions must correspond to the number of relative positions indicated hereafter, when the corresponding relativePositions fields are parsed.
- *useInternalBasisAngle* (bool): if **true**, uses the golden angle to calculate the stage positions, if **false**, uses the *basisAngle_deg* parameter.
- *basisAngle_deg* (Double): angle in degrees used to calculate the spiral positions if the *useInternalBasisAngle* is set to **false**.
- *constantRadius_px* (Double): factor used between consecutive points, in pixels.
- *saveHolograms* (Boolean): if **true** save holograms during acquisition.
- *saveIntensity* (Boolean): if **true** save intensity during acquisition.
- *savePhase* (Boolean): if **true** save phase during acquisition.
- *saveAsBin* (Boolean): if **true** save phase and intensity in **.bin** format during acquisition.
- *saveAsTxt* (Boolean): if **true** save phase and intensity in **.txt** format during acquisition.
- *saveAsTif* (Boolean): if **true** save phase and intensity in **.tif** format during acquisition.
- *saveLambda1* (Boolean): if **true** save phase and intensity for laser source lambda 1 separately.
- *saveLambda2* (Boolean): if **true** save phase and intensity for laser source lambda 2 separately.
- *saveLambda3* (Boolean): if **true** save phase and intensity for laser source lambda 3 separately.
- *isLambda1LaserUsed* (Boolean): if **true** use laser source lambda 1 for acquisition.
- *isLambda2LaserUsed* (Boolean): if **true** use laser source lambda 2 for acquisition.
- *isLambda3LaserUsed* (Boolean): if **true** use laser source lambda 3 for acquisition.
- *savePath* (String): absolute path where the data will be saved. If left empty, the data will be saved in a subfolder with its current date and time in the **My Documents** folder.
- numberOfAcquisitionsPerPosition (Int32): number of acquisitions to be done at each position.
- *isAcquisitionAlternate* (Boolean): if **true**, the hologram acquisition is done alternately for 2-wavelengths configuration, so we obtain a hologram for lambda 1 (folder HologramsLambda1) and a hologram for lambda 2 (folder HologramsLambda2). If **false**, we obtain only one hologram for lambda 1-2 (folder Holograms).

Return: void (nothing)

Requirements:

- A user must be logged in
- A configuration must be loaded
- The Hologram window must be opened.
- The Phase window must be opened.
- The Intensity window must be opened.
- The acquisition must be stopped before starting the recording.

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

C# Example:

```
//css_reference LynceeTec.KoalaRemote.Client.dll;
using System;
using System.IO;
using LynceeTec.KoalaRemote.Client;

//*****************************************************************************
// NOTES:
// - This script is designed to work with a 2 wavelengths configuration.
//   If you use a single wavelength configuration, you will have to comment
//   some of the lines for the script to run.
//
// - The script demonstrate the procedure to record a set of data to produce
//   a lateral scanning.
//
// - You might need to adapt some of the parameters (such as the save/load
//   location or the configuration number) for the script to work on your
//   system with your database
//*****************************************************************************

class TestSampleScriptForSpiralScanning : IKoalaScript
{
   public void Run(KoalaRemoteClient host)
   {

      //Open a 2 wavelengths configuration.
      host.OpenConfig(186);

      //Open main display windows
      host.OpenHoloWin();
      host.OpenPhaseWin();
      host.OpenIntensityWin();
```

```
//Do an acquisition for test
host.Acquisition2L();

//Get the original stage position
double[] startPosition = new double[4];
host.GetAxesPosMu(startPosition);

double startPositionX_um = startPosition[0];
double startPositionY_um = startPosition[1];
double startPositionZ_um = startPosition[2];

//Define the spiral parameters
int totalNumberOfPositions = 25;
bool useInternalBasisAngle = true;
double basisAngle_degrees = 1;
double constantRadius_px = 10.0;

//Define the saving options
bool saveHolograms = true;
bool saveIntensity = true;
bool savePhase = true;
bool saveAsBin = true;
bool saveAsTxt = false;
bool saveAsTif = true;
bool saveLambda1 = true;
bool saveLambda2 = false;
bool saveLambda3 = false;

//Define which wavelength will be used
bool isLambda1LaserUsed = true;
bool isLambda2LaserUsed = true;
bool isLambda3LaserUsed = true;

//Indicate where you save your acquisition data
string savePath = @"C:\tmp";

//Give the number of acquisitions to be realized at each position
int numberOfAcquisitionsPerPosition = 2;

//In case of 2-wavelengths configuration, it indicates if we save
//a hologram for each wavelength separately or not
bool isAcquisitionAlternate = false;

//Start the spiral scanning acquisition
host.StartSpiralScanning(startPositionX_um, startPositionY_um, startPositionZ_um,
    totalNumberOfPositions,
```

```
        useInternalBasisAngle, basisAngle_degrees, constantRadius_px,
        saveHolograms, saveIntensity, savePhase, saveAsBin, saveAsTxt, saveAsTif,
        saveLambda1, saveLambda2, saveLambda3,
        isLambda1LaserUsed, isLambda2LaserUsed, isLambda3LaserUsed,
        savePath, numberOfAcquisitionsPerPosition, isAcquisitionAlternate);
    }
}
```

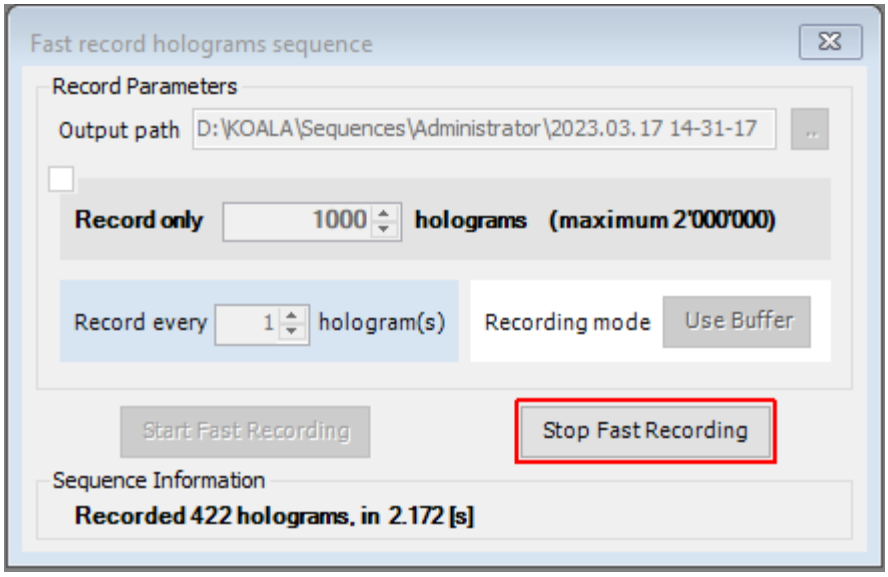
See also:

- *ResetGrab*
- *StartLateralScanning*
- *StartScanningFromPoints*

StopFastHologramsSequenceRecord

Stop the sequence recording using the *Fast record holograms sequence* window. It should not be used in conjunction with a given number of holograms to record, as the recording session will automatically stop when a number or holograms is given.

Koala UI equivalent:
Corresponds to clicking the button **Stop Fast Recording** in the **Fast record holograms sequence** window.



Parameters:

- None.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in
- The Fast Record Holograms sequence window must be opened.
- A Fast recording must be running.

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *CloseFastHologramsRecordWin*
- *OpenFastHologramsRecordWin*
- *SetFastHologramsSequenceRecordingModeBuffer*
- *SetFastHologramsSequenceRecordNumberOfHolograms*

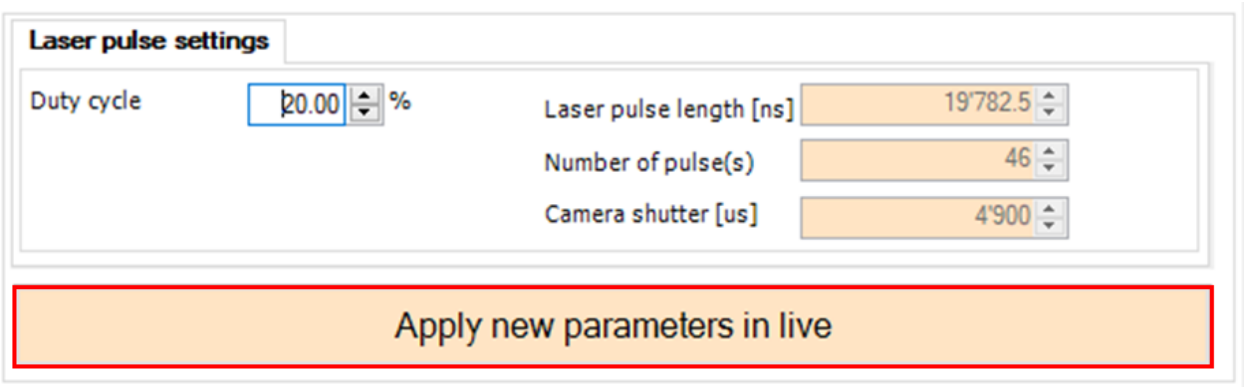
- *SetFastHologramsSequenceRecordPath*
- *SetFastHologramsSequenceRecordPeriodicallyEveryXHologram*
- *StartFastHologramsSequenceRecord*

The stroboscope remote functions

ApplyNewDutyCycleInLiveMode

Apply the modified laser duty cycle when the stroboscope is already started, i.e. in Live mode.

Koala UI equivalent:
Corresponds to clicking on the **Apply new parameters in live** button when it is enabled after a laser duty cycle modification.



(No parameters)

Return: void (nothing)

Requirements:

- The stroboscopic module must be enabled and correctly initialized
- A configuration must be loaded
- The camera must be available
- The stroboscopic tool must be available
- The stroboscopic window must be opened
- A user must be logged in
- The stroboscopic tool must be started (live mode)
- The laser duty cycle must have been modified

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

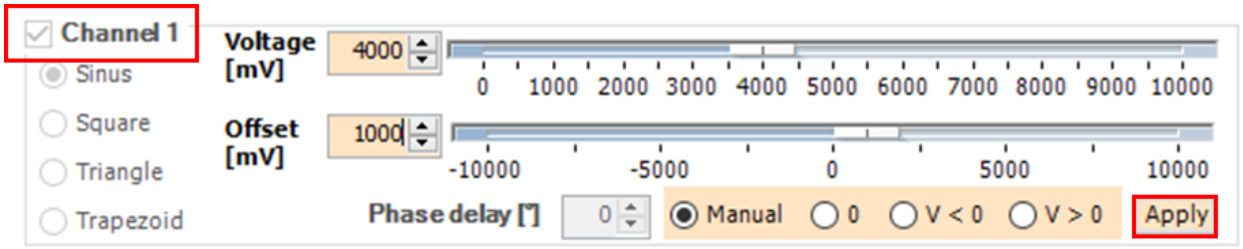
See also:

- *SetStroboscopeLaserPulseDutyCycle*

ApplyNewVoltageAndOffsetInLiveModeForChannelNumber

Apply the modified voltage, offset and offset type, for a given channel number when the stroboscope is already started, i.e. in Live mode.

Koala UI equivalent:
Corresponds to clicking on the **Apply** button when it is enabled after a voltage, offset or offset type modification in the channel panel.



Parameters:

- channelNumber (Int32): gives the channel number in the range [1,2,3,4].

Return: void (nothing)

Requirements:

- The stroboscopic module must be enabled and correctly initialized
- A configuration must be loaded
- The camera must be available
- The stroboscopic tool must be available
- The stroboscopic window must be opened
- A user must be logged in
- The stroboscopic tool must be started (live mode)
- The voltage, offset or offset type must have been modified

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *SetStroboscopeChannel1Parameters*
- *SetStroboscopeChannel2Parameters*
- *SetStroboscopeChannel3Parameters*
- *SetStroboscopeChannel4Parameters*

CloseStroboWin

Closes the stroboscope window

Koala UI equivalent:

Corresponds to clicking on the stroboscope window button  on the toolbar when it is already checked.

(No parameters)

Return: void (nothing)

Requirements:

- The stroboscopic module must be enabled and correctly initialized
- A configuration must be loaded
- The camera must be available
- The stroboscopic tool must be available
- The stroboscopic window must be opened
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

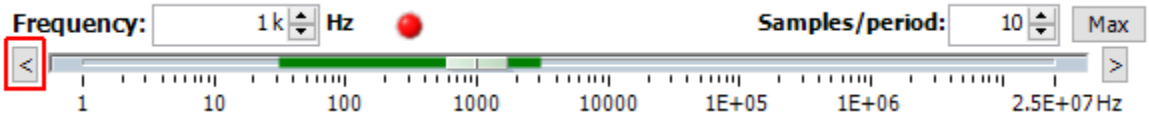
See also:

- *OpenStroboWin*

DecreaseStroboscopeAngleStep

Modify the current frequency operational range (indicated in green on the frequency slider) by moving it to lower frequencies. Reducing the angle step can lead to having more samples per period available.

Koala UI equivalent:
Corresponds to clicking the left arrow button from the frequency slider in the stroboscope window.



(No parameters)

Return: void (nothing)

Requirements:

- The stroboscopic module must be enabled and correctly initialized
- A configuration must be loaded
- The camera must be available
- The stroboscopic tool must be available
- The stroboscopic window must be opened
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

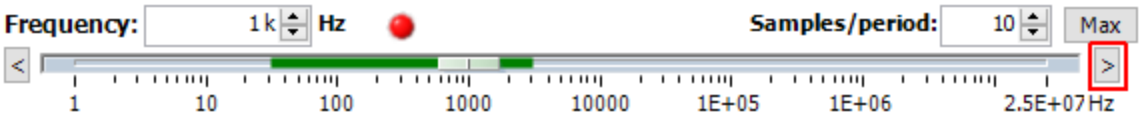
See also:

- *IncreaseStroboscopeAngleStep*

IncreaseStroboscopeAngleStep

Modify the current frequency operational range (indicated in green on the frequency slider) by moving it to higher frequencies.

Koala UI equivalent:
Corresponds to clicking the right arrow button from the frequency slider in the stroboscope window.



(No parameters)

Return: void (nothing)

Requirements:

- The stroboscopic module must be enabled and correctly initialized
- A configuration must be loaded
- The camera must be available
- The stroboscopic tool must be available
- The stroboscopic window must be opened
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

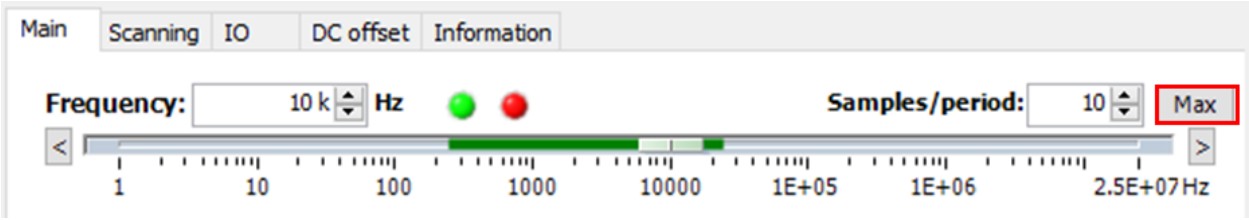
See also:

- *DecreaseStroboscopeAngleStep*

MaximizeStroboscopeNumberOfSamples

Calculate and set the actual maximal number of samples for the current frequency range.

Koala UI equivalent:
Corresponds to the button **Max** besides the number of samples per period in the stroboscopic UI tool.



(No Parameters)

Return: void (nothing)

Requirements:

- The stroboscopic module must be enabled and correctly initialized
- A configuration must be loaded
- The camera must be available
- The stroboscopic tool must be available
- The stroboscopic window must be opened
- A user must be logged in

Possible errors:

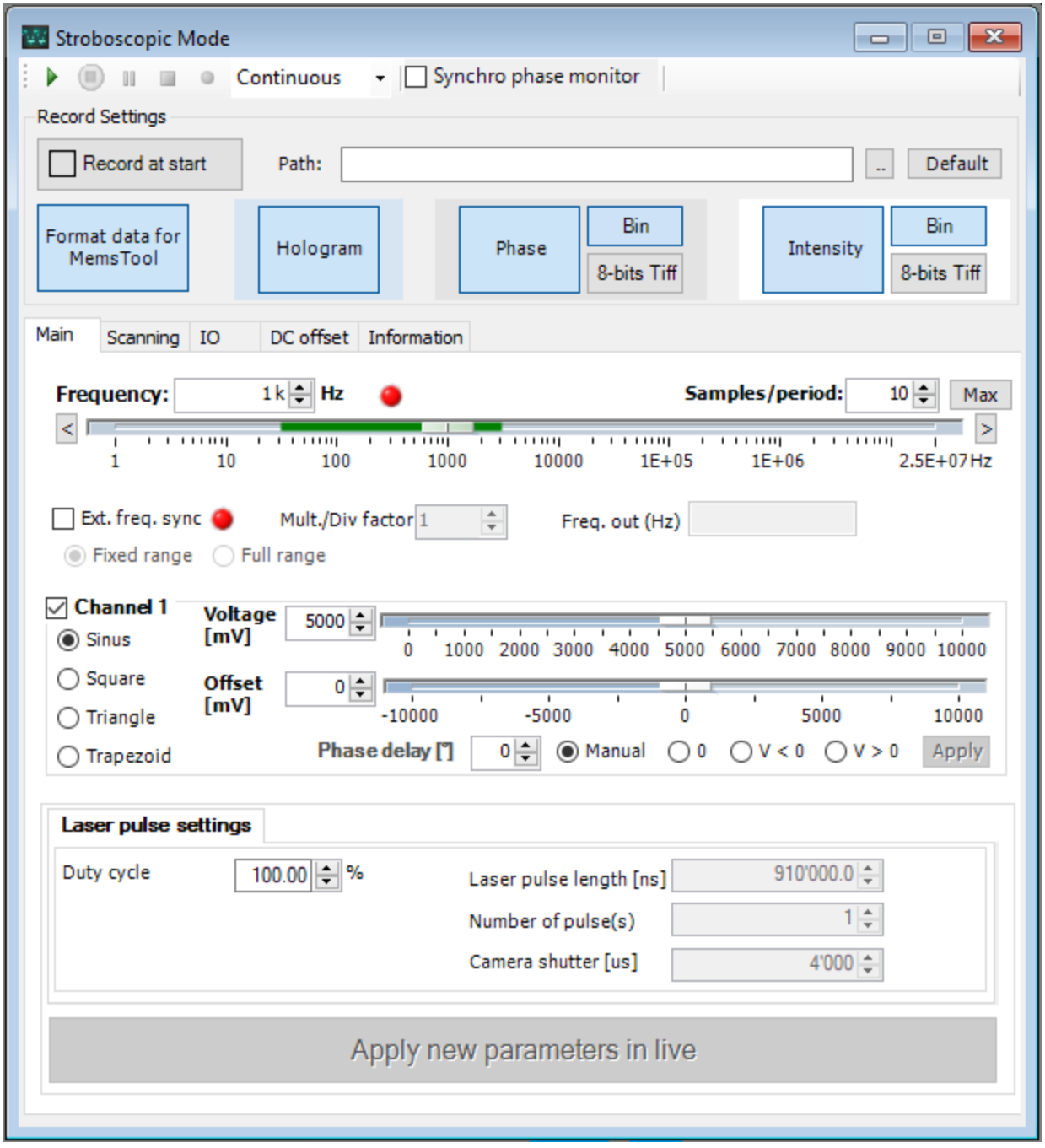
- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *SetStroboscopeNumberOfSamplesPerPeriod*

OpenStroboWin

Opens the stroboscope window



Stroboscope window example for one channel

Koala UI equivalent:

Corresponds to clicking on the Open stroboscope window button  on the toolbar.

(No parameters)

Return: void (nothing)

Requirements:

- The stroboscopic module must be enabled and correctly initialized
- A configuration must be loaded
- The camera must be available
- The stroboscopic tool must be available
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

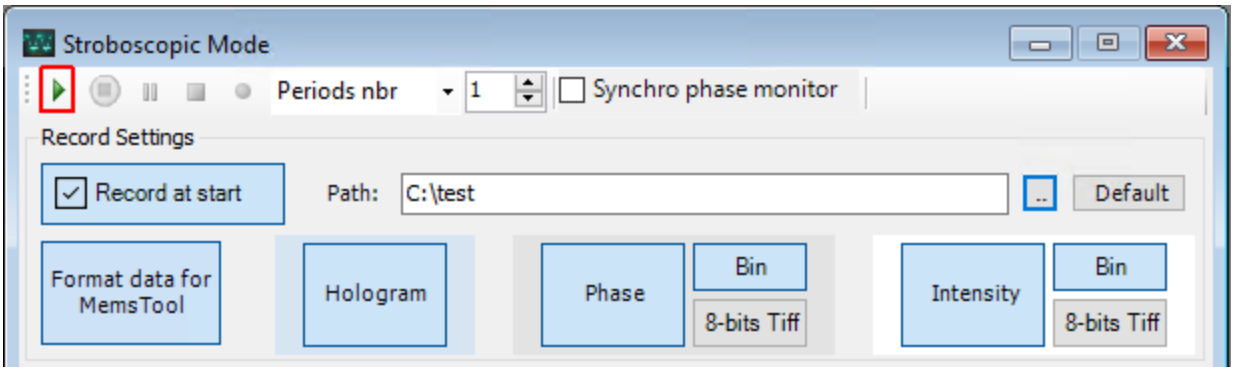
- *OpenHoloWin*
- *OpenIntensityWin*
- *OpenPhaseWin*

RecordStroboscopeFixedFrequency

Starts a fixed frequency **recording** with the stroboscopic tool, for a given number of periods. A classic stroboscope fixed frequency sequence where recording is not done can be started with *StartStroboscopeFixedFrequency*.

The stroboscopic tool must have been previously configured by hand in Koala, or using remote commands. The stroboscopic window must be opened on the *Main* tab, and all options except the number of periods must have been set manually.

Koala UI equivalent:
Corresponds to enabling *Record at start* and then clicking on the *Start stroboscopic* mode button  in the stroboscopic window.



Stroboscope window with **Record at start** checked

Parameters:

- numberOfPeriods (Int32): The number of periods of the signal generated by the stroboscopic tool to record

Return: String: the path where the result has been saved

Requirements:

- The stroboscopic module must be enabled and correctly initialized
- A configuration must be loaded
- The camera must be available
- The stroboscopic tool must be available
- The stroboscopic window must be opened
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error
- The stroboscopic window is not configured correctly ⇒ *Execution failed* error

See also:

- *RecordStroboscopeFrequencyScan*
- *StartStroboscopeFixedFrequency*

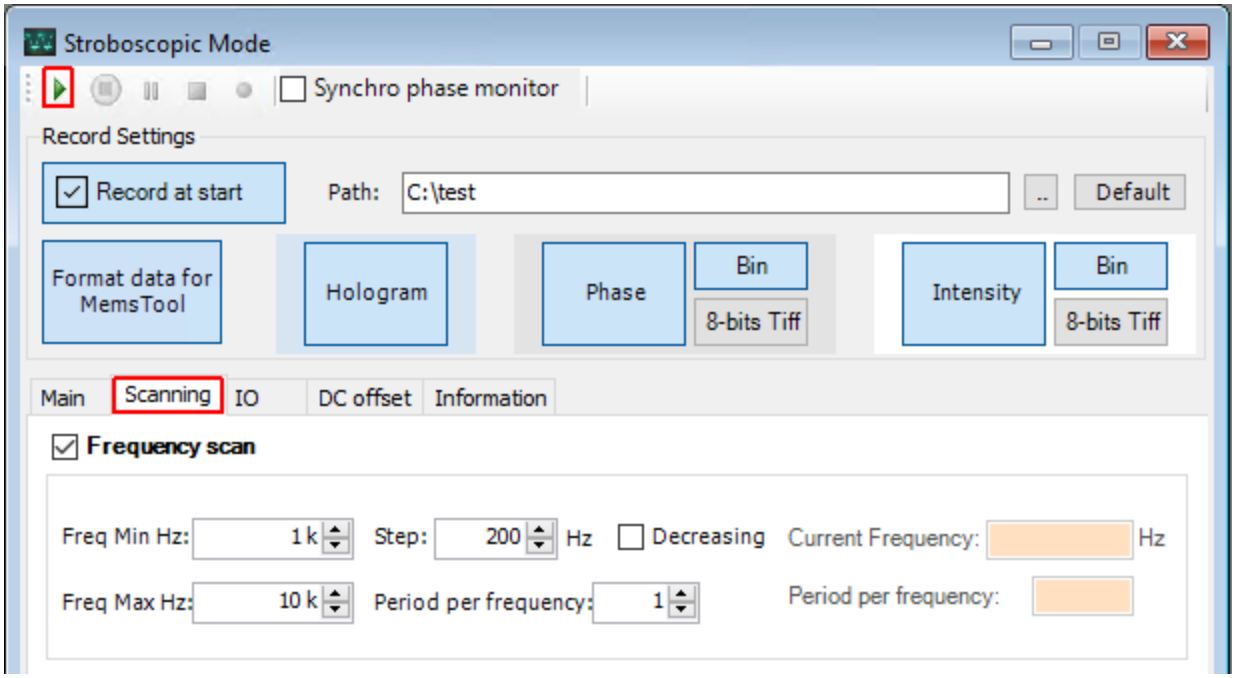
Modifications:

- See RecordStroboFixFrequency ⇒ RecordStroboscopeFixedFrequency

RecordStroboscopeFrequencyScan

Starts a frequency scan recording with the stroboscopic tool. The stroboscopic tool must have been previously configured by hand in Koala , or using remote commands. The stroboscopic window must be opened on the *Scanning* tab, the *Frequency scan* option must have been selected and the desired options (minimal and maximal frequency, step size, number of periods per frequency, etc…) must have been set before starting the recording).

Koala UI equivalent:
Corresponds to enabling *Record at start* and then clicking on the *Start stroboscopic* mode button  in the stroboscopic window.



(No parameters)

Return: String: the path where the result has been saved

Requirements:

- The stroboscopic module must be enabled and correctly initialized
- A configuration must be loaded
- The camera must be available
- The stroboscopic tool must be available
- The stroboscopic window must be opened
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error
- The stroboscopic window is not configured correctly ⇒ *Execution failed* error

See also:

- *RecordStroboscopeFixedFrequency*
- *SetStroboscopeFrequencyScanEnabled*
- *SetStroboscopeFrequencyScanParameters*

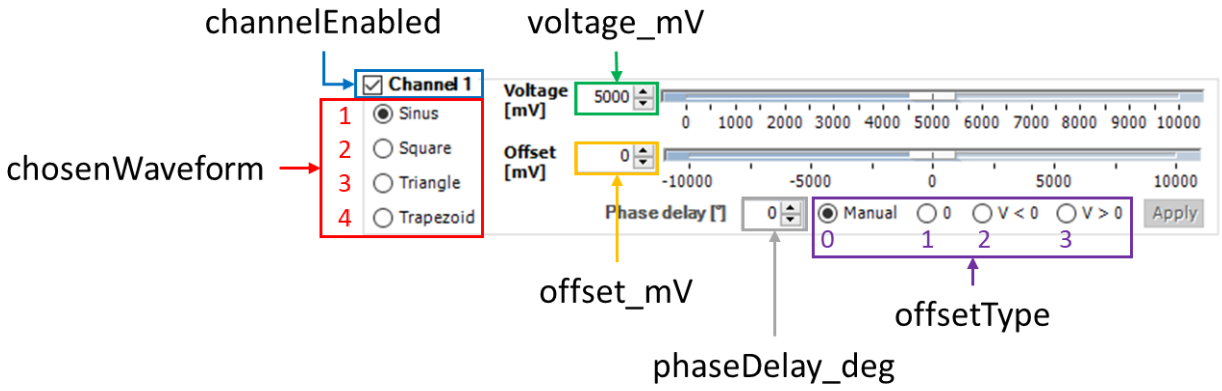
Modifications:

- See RecordStroboFrequencyScan ⇒ RecordStroboscopeFrequencyScan

SetStroboscopeChannel1Parameters

Set the stroboscopic tool parameters for channel 1.

Koala UI equivalent:
Corresponds to the channel 1 settings in the stroboscopic UI tool.



Parameters (indicated in color on the diagram above):

- channelEnabled (*Boolean*): `true` to enable the channel, `false` to disable it.
- chosenWaveform (Int32): waveform chosen in the array [1,2,3,4]
- voltage_mV (Int32): Voltage value in [mV] in the range [0,10000]
- offset_mV (Int32): Offset value in [mV] in the range [-10000,10000]
- phaseDelay_deg (Int32): Phase in [degrees] in the range [0,360]
- offsetType (Int32): Offset type [0 for "Manual", 1 for "0", 2 for "V<0", 3 for "V>0"]

Return: void (nothing)

Requirements:

- The stroboscopic module must be enabled and correctly initialized
- A configuration must be loaded
- The camera must be available
- The stroboscopic tool must be available
- The stroboscopic window must be opened
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error
- The stroboscopic window is not configured correctly ⇒ *Execution failed* error

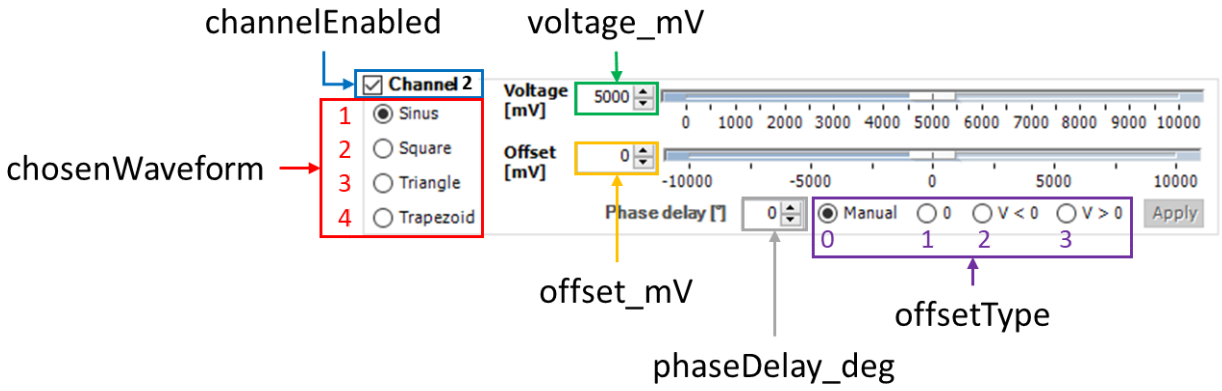
See also:

- *SetStroboscopeChannel2Parameters*
- *SetStroboscopeChannel3Parameters*
- *SetStroboscopeChannel4Parameters*

SetStroboscopeChannel2Parameters

Set the stroboscopic tool parameters for channel 2.

Koala UI equivalent:
Corresponds to the channel 2 settings in the stroboscopic UI tool.



Parameters (indicated in color on the diagram above):

- channelEnabled (*Boolean*): `true` to enable the channel, `false` to disable it.
- chosenWaveform (Int32): waveform chosen in the array [1,2,3,4]
- voltage_mV (Int32): Voltage value in [mV] in the range [0,10000]
- offset_mV (Int32): Offset value in [mV] in the range [-10000,10000]
- phaseDelay_deg (Int32): Phase in [degrees] in the range [0,360]
- offsetType (Int32): Offset type [0 for "Manual", 1 for "0", 2 for "V<0", 3 for "V>0"]

Return: void (nothing)

Requirements:

- The stroboscopic module must be enabled and correctly initialized
- A configuration must be loaded
- The camera must be available
- The stroboscopic tool must be available
- The stroboscopic window must be opened
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error
- The stroboscopic window is not configured correctly ⇒ *Execution failed* error

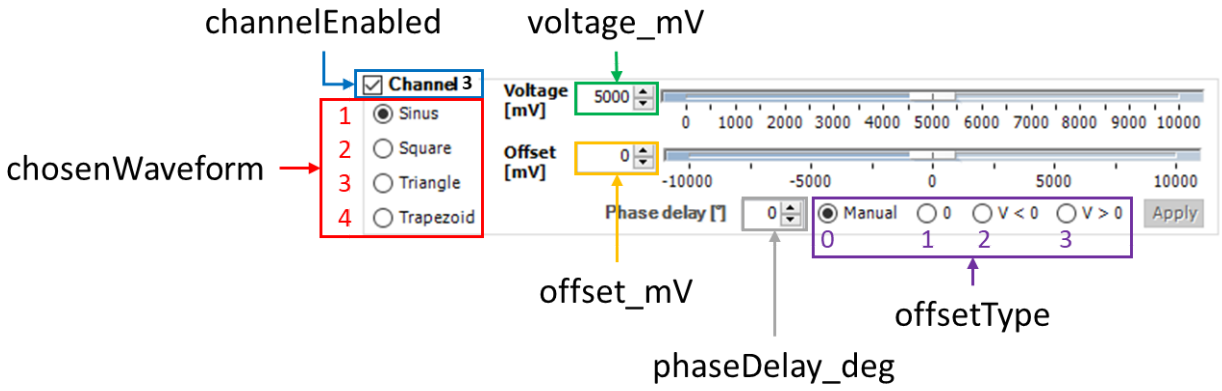
See also:

- *SetStroboscopeChannel1Parameters*
- *SetStroboscopeChannel3Parameters*
- *SetStroboscopeChannel4Parameters*

SetStroboscopeChannel3Parameters

Set the stroboscopic tool parameters for channel 3.

Koala UI equivalent:
Corresponds to the channel 3 settings in the stroboscopic UI tool.



Parameters (indicated in color on the diagram above):

- channelEnabled (*Boolean*): `true` to enable the channel, `false` to disable it.
- chosenWaveform (Int32): waveform chosen in the array [1,2,3,4]
- voltage_mV (Int32): Voltage value in [mV] in the range [0,10000]
- offset_mV (Int32): Offset value in [mV] in the range [-10000,10000]
- phaseDelay_deg (Int32): Phase in [degrees] in the range [0,360]
- offsetType (Int32): Offset type [0 for "Manual", 1 for "0", 2 for "V<0", 3 for "V>0"]

Return: void (nothing)

Requirements:

- The stroboscopic module must be enabled and correctly initialized
- A configuration must be loaded
- The camera must be available
- The stroboscopic tool must be available
- The stroboscopic window must be opened
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error
- The stroboscopic window is not configured correctly ⇒ *Execution failed* error

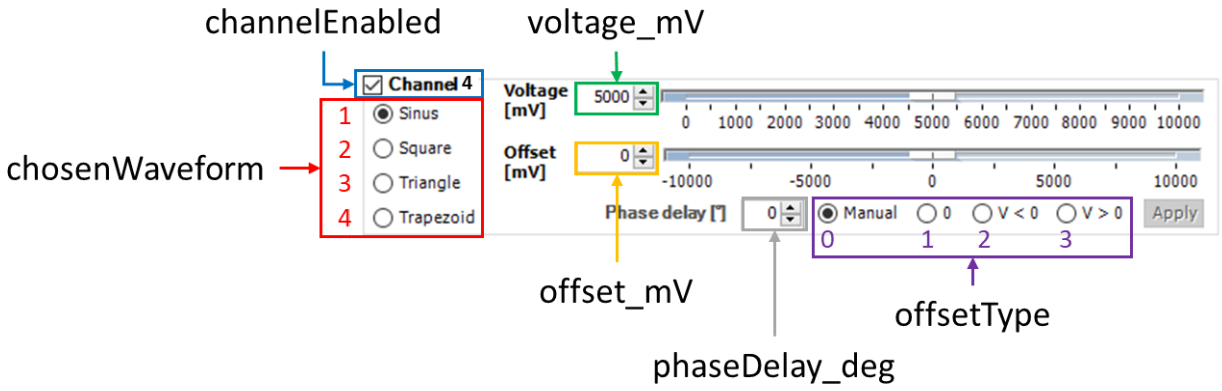
See also:

- *SetStroboscopeChannel1Parameters*
- *SetStroboscopeChannel2Parameters*
- *SetStroboscopeChannel4Parameters*

SetStroboscopeChannel4Parameters

Set the stroboscopic tool parameters for channel 4.

Koala UI equivalent:
Corresponds to the channel 4 settings in the stroboscopic UI tool.



Parameters (indicated in color on the diagram above):

- channelEnabled (*Boolean*): `true` to enable the channel, `false` to disable it.
- chosenWaveform (Int32): waveform chosen in the array [1,2,3,4]
- voltage_mV (Int32): Voltage value in [mV] in the range [0,10000]
- offset_mV (Int32): Offset value in [mV] in the range [-10000,10000]
- phaseDelay_deg (Int32): Phase in [degrees] in the range [0,360]
- offsetType (Int32): Offset type [0 for "Manual", 1 for "0", 2 for "V<0", 3 for "V>0"]

Return: void (nothing)

Requirements:

- The stroboscopic module must be enabled and correctly initialized
- A configuration must be loaded
- The camera must be available
- The stroboscopic tool must be available
- The stroboscopic window must be opened
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error
- The stroboscopic window is not configured correctly ⇒ *Execution failed* error

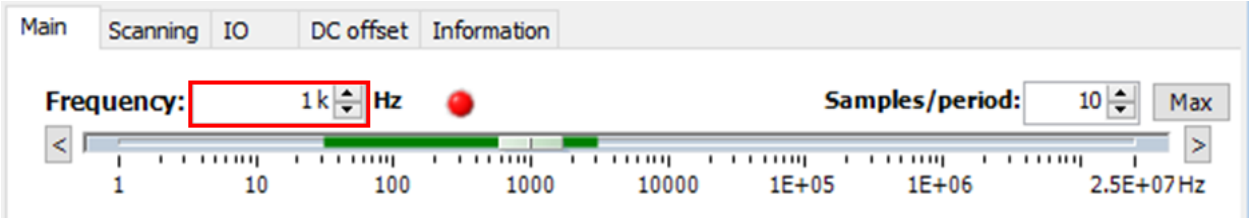
See also:

- *SetStroboscopeChannel1Parameters*
- *SetStroboscopeChannel2Parameters*
- *SetStroboscopeChannel3Parameters*

SetStroboscopeFixedFrequency

Set the stroboscopic tool *Fixed Frequency* mode.

Koala UI equivalent:
Corresponds to the frequency settings in the stroboscopic UI tool.



Parameters:

- frequency (*Double*): The frequency must be between 1[Hz] and 25[MHz].

Return: void (nothing)

Requirements:

- The stroboscopic module must be enabled and correctly initialized
- A configuration must be loaded
- The camera must be available
- The stroboscopic tool must be available
- The stroboscopic window must be opened
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error
- The stroboscopic window is not configured correctly ⇒ *Execution failed* error

See also:

- *RecordStroboscopeFixedFrequency*

SetStroboscopeFrequencyApproachEnabled

Enable/Disable the stroboscopic tool frequency approach. The main panel will have its input for fixed frequency blocked when the frequency approach is selected, as shown below.

Koala UI equivalent:
Corresponds to the **frequency approach** checkbox status in the stroboscopic UI tool.

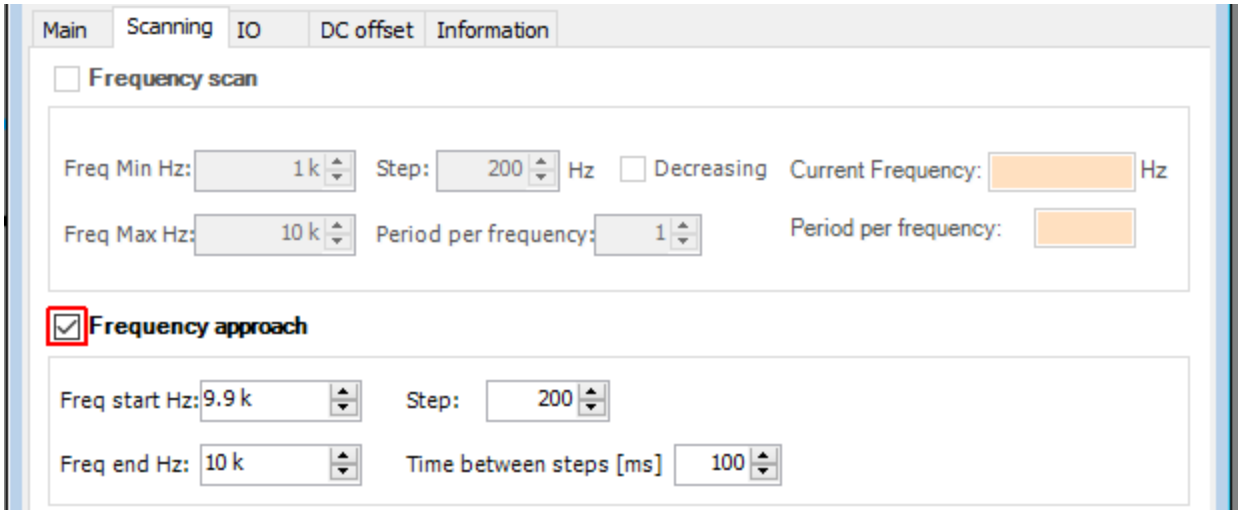


Figure: Frequency approach panel

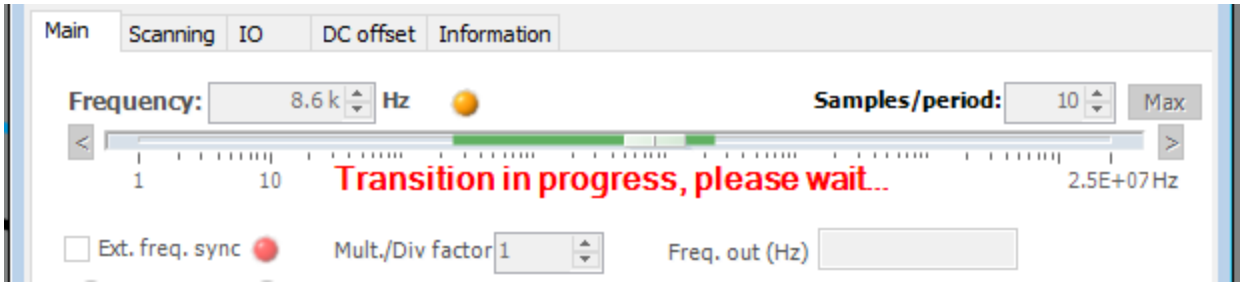


Figure: Main panel

Parameters:

- status (*Boolean*): `true` to enable the frequency approach, `false` to disable it.

Return: void (nothing)

Requirements:

- The stroboscopic module must be enabled and correctly initialized
- A configuration must be loaded
- The camera must be available
- The stroboscopic tool must be available
- The stroboscopic window must be opened

- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error
- The stroboscopic window is not configured correctly ⇒ *Execution failed* error

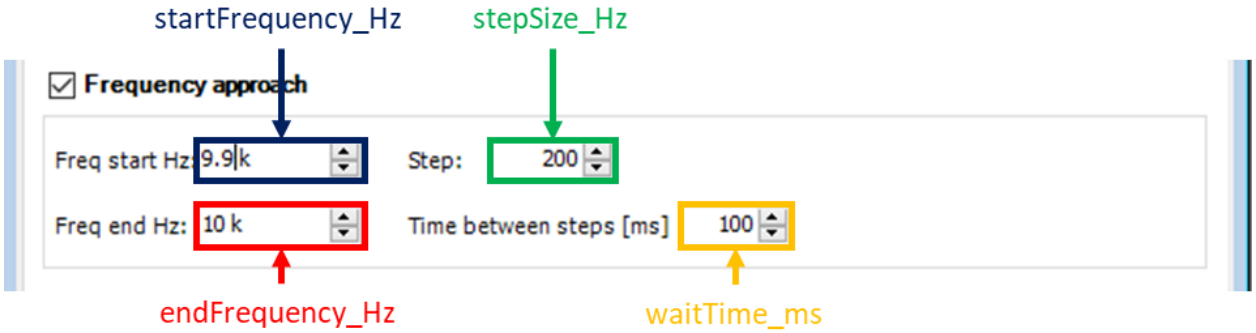
See also:

- *SetStroboscopeFrequencyApproachParameters*

SetStroboscopeFrequencyApproachParameters

Set the stroboscopic tool frequency approach parameters. The iterations of the frequency approach are applied between the starting and ending (working) frequencies, with each time a frequency difference indicated by the step size. We apply the iterations until the current frequency added to the step size reaches the desired frequency.

Koala UI equivalent:
Corresponds to the frequency approach parameters in the stroboscopic UI tool indicated below.



Parameters:

- startFrequency_Hz (Double): starting frequency in [Hz] for the frequency approach.
- endFrequency_Hz (Double): end frequency in [Hz] (working frequency) for the frequency approach.
- stepSize_Hz (Double): frequency difference in [Hz] between 2 iterations (e.g with a frequency start at 1[kHz], and a step size of 200[Hz], the next frequency will be at 1.2[kHz], if we are increasing the frequency.
- waitTime_ms (Int32): number of milliseconds waited between each frequency iteration.

Return: void (nothing)

Requirements:

- The stroboscopic module must be enabled and correctly initialized
- A configuration must be loaded
- The camera must be available
- The stroboscopic tool must be available
- The stroboscopic window must be opened
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error
- The stroboscopic window is not configured correctly ⇒ *Execution failed* error

See also:

- *SetStroboscopeFrequencyApproachEnabled*

SetStroboscopeFrequencyScanEnabled

Enable/Disable the stroboscopic tool frequency scan. The main panel will have its input for fixed frequency blocked when the frequency scan is selected, as shown below.

Koala UI equivalent:
Corresponds to the frequency scan checkbox status in the stroboscopic UI tool.

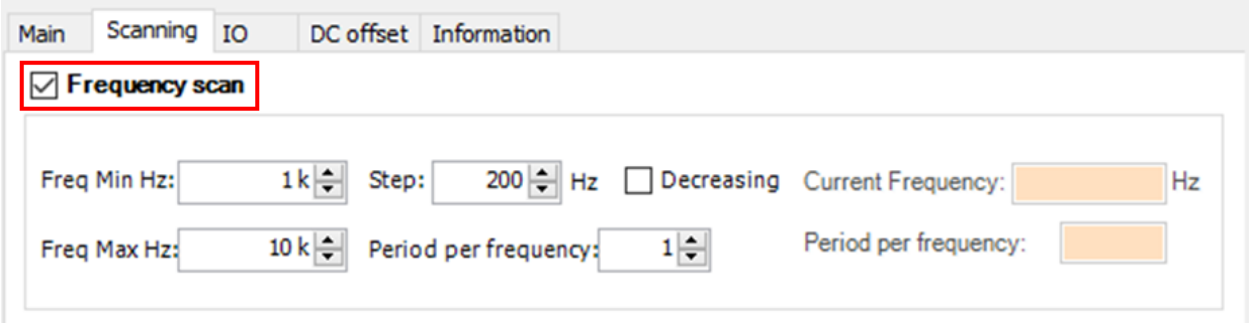


Figure: Frequency scan panel

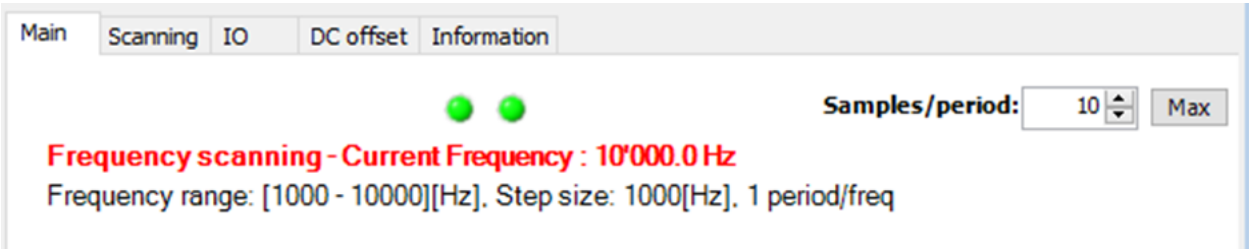


Figure: Main panel

Parameters:

- status (*Boolean*): `true` to enable the frequency scan, `false` to disable it.

Return: void (nothing)

Requirements:

- The stroboscopic module must be enabled and correctly initialized
- A configuration must be loaded
- The camera must be available
- The stroboscopic tool must be available
- The stroboscopic window must be opened
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error
- The stroboscopic window is not configured correctly ⇒ *Execution failed* error

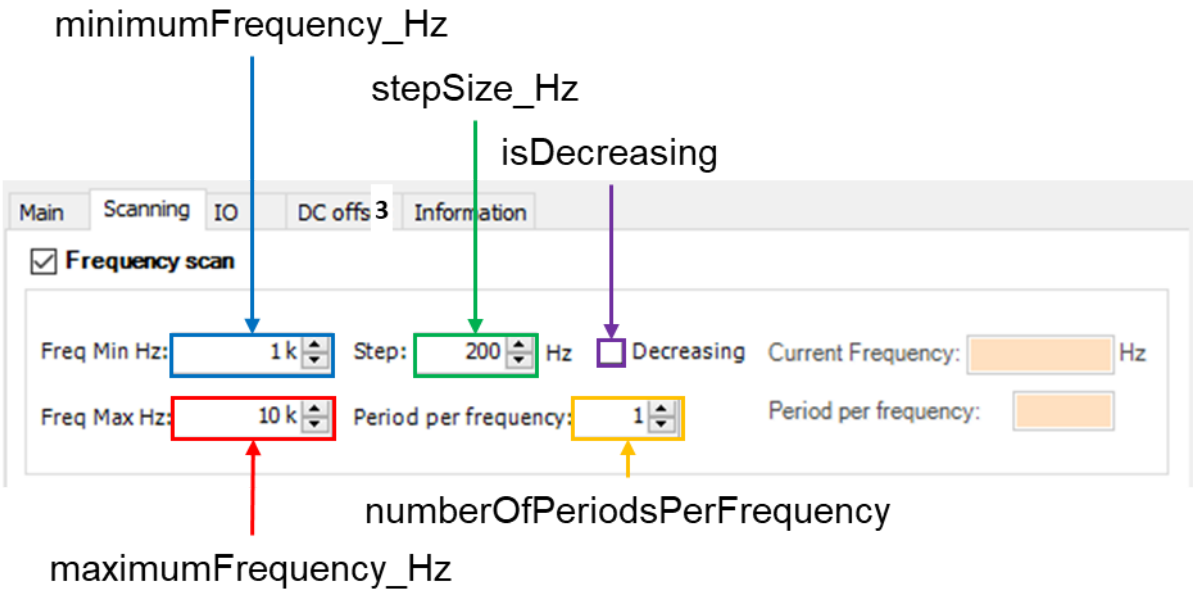
See also:

- *SetStroboscopeFrequencyScanParameters*

SetStroboscopeFrequencyScanParameters

Set the stroboscopic tool frequency scan parameters. The iterations of the frequency scan are applied between the minimum and maximum frequencies, with each time a frequency difference indicated by the step size. We apply the iterations until the current frequency added to the step size reaches the maximal frequency. If the step applied produces a frequency value greater than the maximal frequency chosen, the scan stops.

Koala UI equivalent:
Corresponds to the frequency scan parameters in the stroboscopic UI tool indicated below.



Parameters:

- minimumFrequency_Hz (Double): minimal frequency in [Hz] for the frequency scan.
- maximumFrequency_Hz (Double): maximal frequency in [Hz] for the frequency scan.
- stepSize_Hz (Double): frequency difference in [Hz] between 2 iterations (e.g with a frequency start at 1[kHz], and a step size of 200[Hz], the next frequency will be at 1.2[kHz], if we are increasing the frequency.
- numberOfPeriodsPerFrequency (Int32): number of periods applied on each frequency iteration.
- isDecreasing (*Boolean*):
    - `true` to decrease frequencies during the frequency scan (the scan will start at *maximumFrequency_Hz* and end at *minimumFrequency_Hz* while taking into account the *stepSize_Hz*),
    - `false` to increase frequencies during the frequency scan (the scan will start at *minimumFrequency_Hz* and end at *maximumFrequency_Hz* while taking into account the *stepSize_Hz*).

Return: void (nothing)

Requirements:

- The stroboscopic module must be enabled and correctly initialized
- A configuration must be loaded
- The camera must be available
- The stroboscopic tool must be available
- The stroboscopic window must be opened
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error
- The stroboscopic window is not configured correctly ⇒ *Execution failed* error

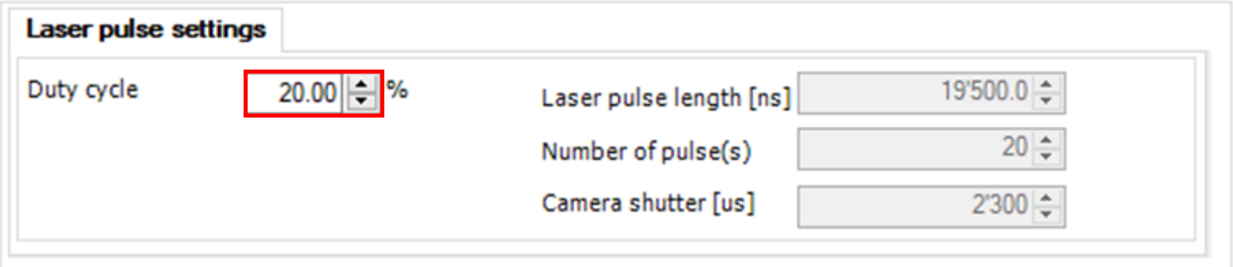
See also:

- *SetStroboscopeFrequencyScanEnabled*

SetStroboscopeLaserPulseDutyCycle

Set a value for the laser duty cycle when using the stroboscope.

Koala UI equivalent:
Corresponds to the stroboscope laser duty cycle in the stroboscopic UI tool.



Parameters:

- dutyCycle (*Double*): gives the laser duty cycle value in percent between 0.01% and 100% .

Return: void (nothing)

Requirements:

- The stroboscopic module must be enabled and correctly initialized
- A configuration must be loaded
- The camera must be available
- The stroboscopic tool must be available
- The stroboscopic window must be opened
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

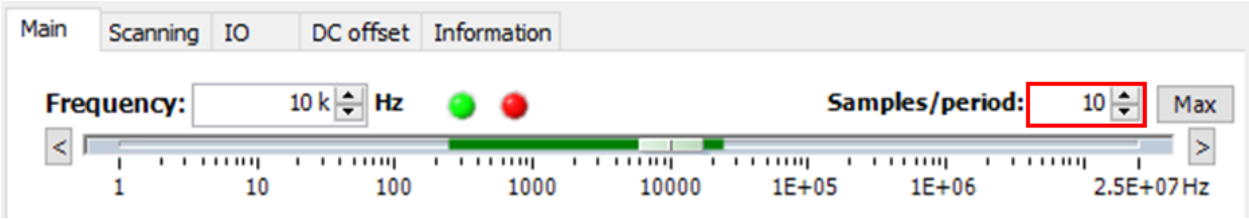
See also:

- *SetStroboscopeFixedFrequency*
- *ApplyNewDutyCycleInLiveMode*

SetStroboscopeNumberOfSamplesPerPeriod

Set the number of samples used for one period when using the stroboscope.

Koala UI equivalent:
Corresponds to the stroboscope **Samples/period** in the stroboscopic UI tool.



Parameters:

- samplesPerPeriod (*Int32*): gives the number of samples per period.

Return: void (nothing)

Requirements:

- The stroboscopic module must be enabled and correctly initialized
- A configuration must be loaded
- The camera must be available
- The stroboscopic tool must be available
- The stroboscopic window must be opened
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

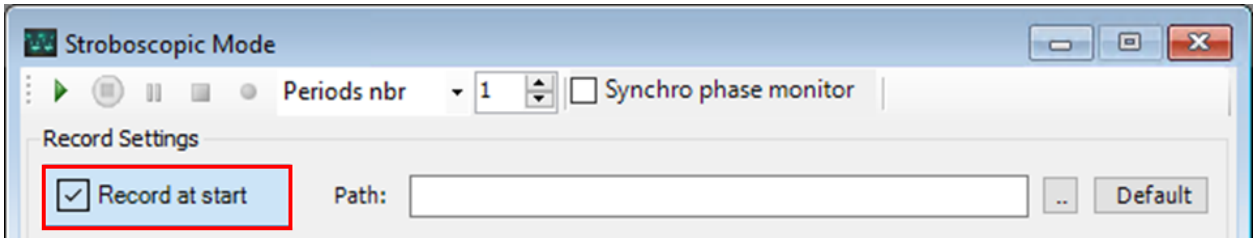
See also:

- *SetStroboscopeFixedFrequency*
- *SetStroboscopeMaxNumberOfSamples*

SetStroboscopeRecordAtStartStatus

Choose if we record or not when we begin a stroboscopic sequence.

Koala UI equivalent:
Corresponds to the stroboscope **Record At Start** button in the stroboscopic UI tool.



Here the Record at start button is activated, e.g. the record will start automatically when we start the stroboscopic tool.

Parameters:

- status (*Boolean*): `true` to start the recording when we start the stroboscope sequence, `false` will not trigger any recording session.

Return: void (nothing)

Requirements:

- The stroboscopic module must be enabled and correctly initialized
- A configuration must be loaded
- The camera must be available
- The stroboscopic tool must be available
- The stroboscopic window must be opened
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

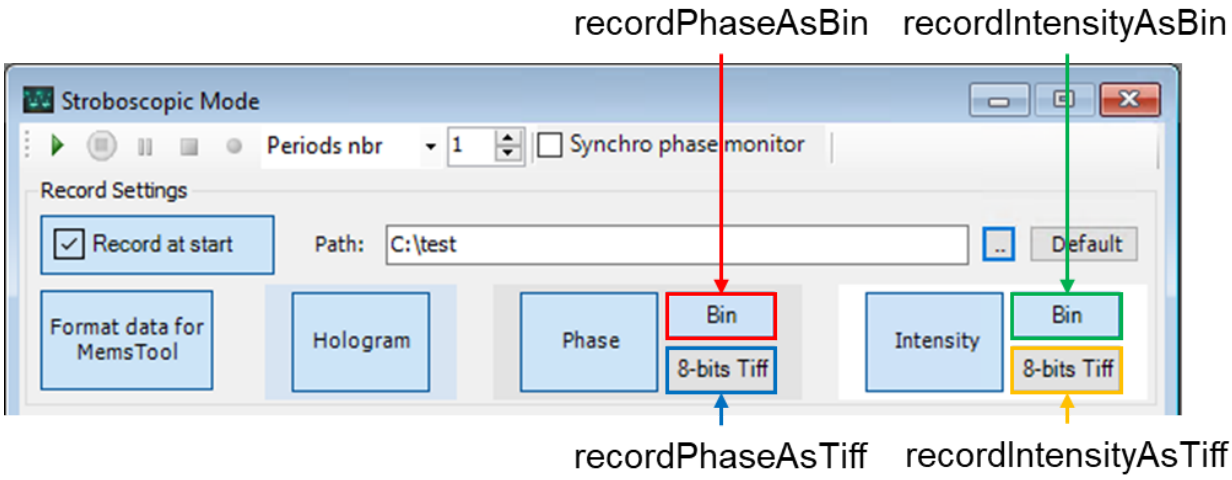
See also:

- *RecordStroboscopeFixedFrequency*
- *SetStroboscopeRecordDataType*
- *SetStroboscopeRecordPath*
- *StartStroboscopeFixedFrequency*

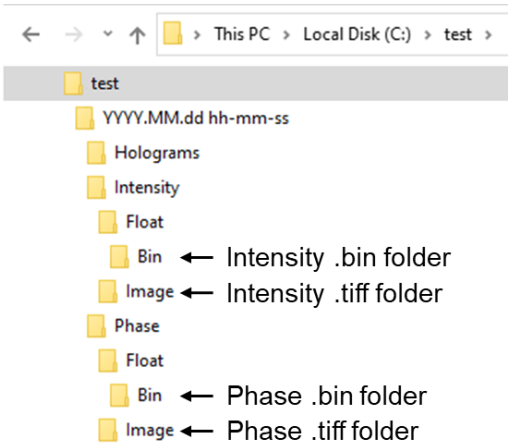
SetStroboscopeRecordDataType

Choose what kind of files are recorded or not when we begin a stroboscopic sequence.

Koala UI equivalent:
Corresponds to the stroboscope Phase **Bin, 8.bits Tiff** and Intensity **Bin, 8.bits Tiff** buttons in the stroboscopic UI tool.



Here the Phase and Intensity **Bin** buttons are activated, e.g. we will record phase and intensity in .bin formats.
See *SetStroboscopeRecordPath* for details about the main recording folder. The folders are created automatically when the record starts, if the corresponding option is checked. The **Holograms** folder is mandatory.



Parameters:

- recordPhaseAsBin (*Boolean*): `true` to start recording phase in **.bin** format in the folder **RECORD_PATH\YYYY.MM.dd hh-mm-ss\Phase\Float\Bin**, `false` otherwise.
- recordPhaseAsTiff (*Boolean*): `true` to start recording phase in **.tiff** format in the folder **RECORD_PATH\YYYY.MM.dd hh-mm-ss\Phase\Image**, `false` otherwise.
- recordIntensityAsBin (*Boolean*): `true` to start recording intensity in **.bin** format in the folder **RECORD_PATH\YYYY.MM.dd hh-mm-ss\Intensity\Float\Bin**, `false` otherwise.
- recordIntensityAsTiff (*Boolean*): `true` to start recording intensity in **.tiff** format in the folder **RECORD_PATH\YYYY.MM.dd hh-mm-ss\Intensity\Image**, `false` otherwise.

Return: void (nothing)

Requirements:

- The stroboscopic module must be enabled and correctly initialized
- A configuration must be loaded
- The camera must be available
- The stroboscopic tool must be available
- The stroboscopic window must be opened
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

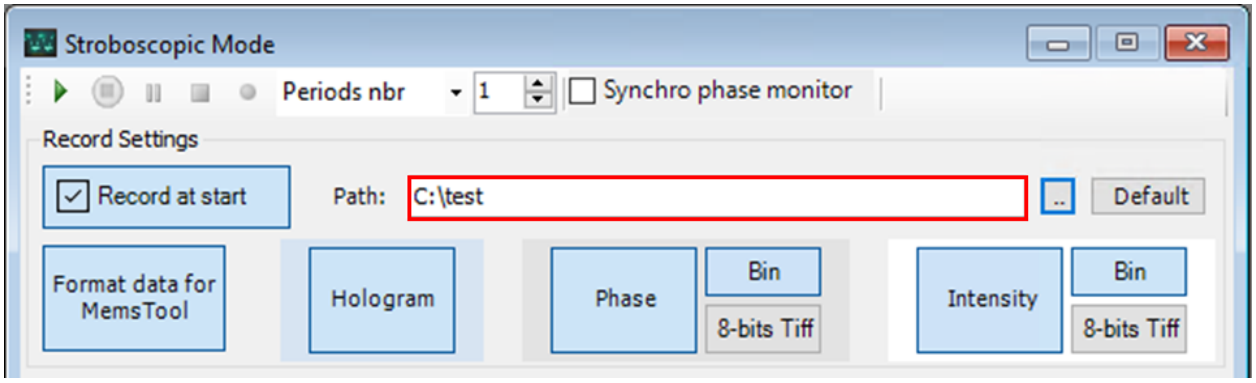
See also:

- *RecordStroboscopeFixedFrequency*
- *SetStroboscopeRecordAtStartStatus*
- *SetStroboscopeRecordPath*
- *StartStroboscopeFixedFrequency*

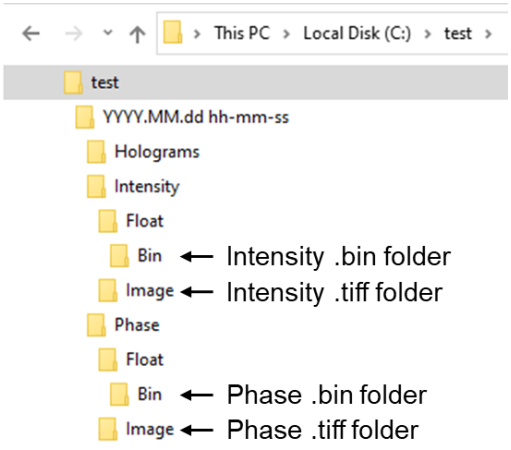
SetStroboscopeRecordPath

Choose the path where saved data is stored.

Koala UI equivalent:
Corresponds to the path indicated in the stroboscopic tool below.



The target folder **RECORD_PATH** where the records will be stored is the path concatenated with the date and time: for example, here it will be **RECORD_PATH\YYYY.MM.dd hh-mm-ss**, where YYYY is for the year, MM for the month, dd for the day, hh for the hour, mm for the minutes, and ss for the seconds (e.g. **C:\test\2022-01-13 10-22-55**). These folders are created automatically when the record starts, if the corresponding option is checked. The **Holograms** folder is mandatory.



Parameters:

- path (*String*): set the folder path **RECORD_PATH**.

Return: void (nothing)

Requirements:

- The stroboscopic module must be enabled and correctly initialized
- A configuration must be loaded
- The camera must be available
- The stroboscopic tool must be available
- The stroboscopic window must be opened
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

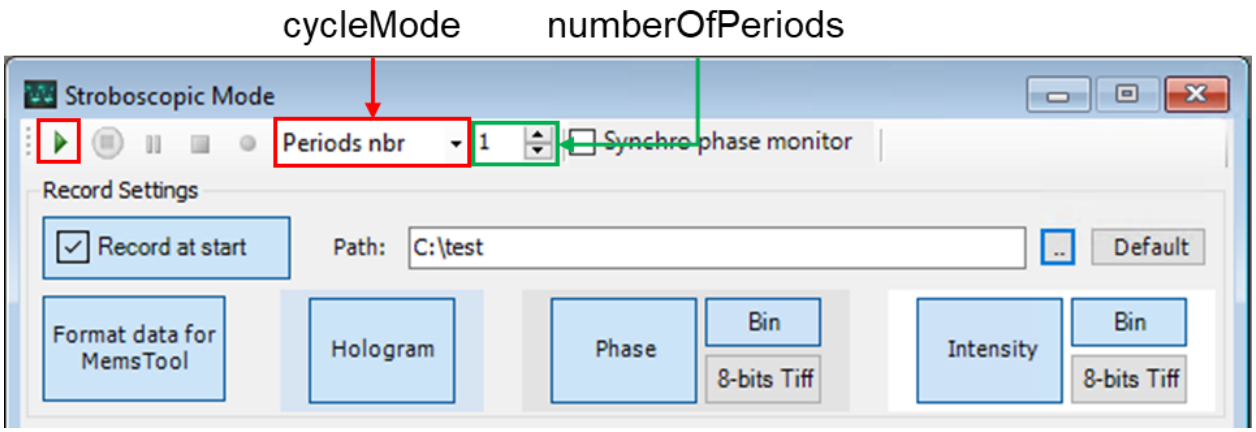
- *RecordStroboscopeFixedFrequency*
- *SetStroboscopeRecordAtStartStatus*
- *SetStroboscopeRecordDataType*
- *StartStroboscopeFixedFrequency*

StartStroboscopeFixedFrequency

Start the stroboscopic sequence with the given cycle mode (continuous or number of periods).
A stroboscope fixed frequency sequence **recording** can be started with *RecordStroboscopeFixedFrequency*.

Koala UI equivalent:

Corresponds to the start button and other fields indicated in the stroboscopic tool below.



Parameters:

- cycleMode (*Int32*): choose the mode to do the stroboscope sequence [0 for continuous, 1 for number of periods].
- numberOfPeriods (*Int32*): sets the number of periods when the mode chosen before is set to 1.

Return: void (nothing)

Requirements:

- The stroboscopic module must be enabled and correctly initialized
- A configuration must be loaded
- The camera must be available
- The stroboscopic tool must be available
- The stroboscopic window must be opened
- A user must be logged in

Possible errors:

- cycleMode is different from 0 or 1 ⇒ *Invalid Operation* error
- cycleMode is 1 and numberOfPeriods < 1 ⇒ *Invalid Operation* error
- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error
- Start fails ⇒ *Execution fails* error

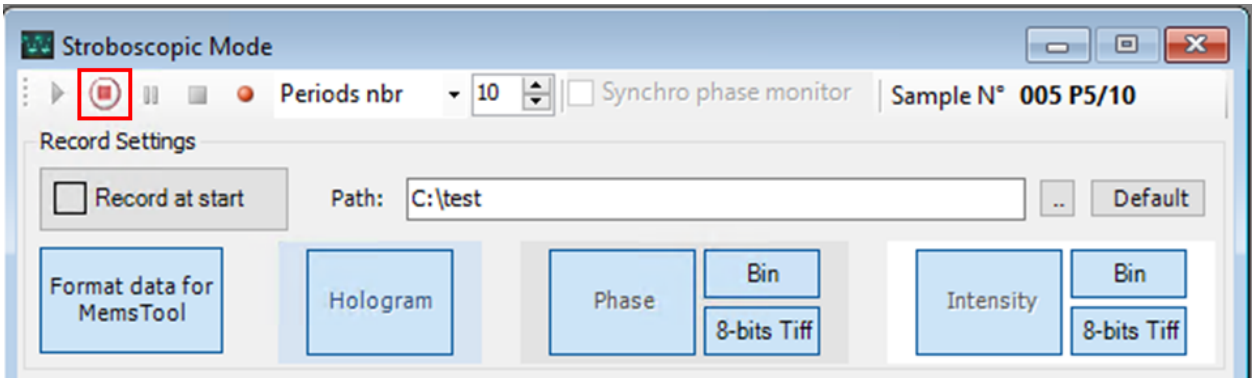
See also:

- *StopStroboscope*
- *RecordStroboscopeFixedFrequency*

StopStroboscope

Stop the current stroboscopic sequence (fixed frequency or frequency scan).

Koala UI equivalent:
Corresponds to the stop button indicated in the stroboscopic tool below.



(No Parameters)

Return: void (nothing)

Requirements:

- The stroboscopic module must be started
- The stroboscopic module must be enabled and correctly initialized
- A configuration must be loaded
- The camera must be available
- The stroboscopic tool must be available
- The stroboscopic window must be opened
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error
- Stop fails ⇒ *Execution fails* error

See also:

- *StartStroboscopeFixedFrequency*

Former functions which were removed

CloseConfig

Closes the current measurement configuration

Koala UI equivalent:
Corresponds to *File → Close Configuration*

Parameters:

- confirm (*Boolean*): If set to `true`, the confirmation window will be displayed to the user

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *OpenConfig*

GetHoloImage (deprecated)

Copies the current hologram image **for a given lamdaId**.

Koala UI equivalent:
(not applicable)

Parameters:

- buffer (*Byte[]*): Array of bytes of dimension `stride*height` where the data will be copied. (See the *GetPhaseImage* function for an example of how to compute the stride)
- lambdaId (*Int32*): Logical ID of the source from which the hologram must be saved (for alternate dual wavelength configurations). Optional, default value is 0.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- The hologram window must be opened and must contain an image
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *OpenHoloWin*

GetMaxHoloContrast

Gets the maximum value of the contrast in the hologram image.

Koala UI equivalent:
(not applicable)

(No parameters)

Return: Double

Requirements:

- A configuration must be loaded
- An hologram must have been grabed since program start
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *GetHoloContrast*

GetRaNm

Gets the Ra (arithmetic mean roughness) roughness measurement in the selected topography zone.

Koala UI equivalent:
(Not applicable, but roughness measurements are displayed in the *Surface topography Phase window.*)

(No parameters)

Return: Single: The Ra, in [nm]

Requirements:

- A configuration must be loaded
- The phase window must be opened and must contain an image
- A roughness measurement zone must be selected
- The topography window must be opened
- A reconstruction object must be created
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *OpenFrmTopography*
- *SelectTopoZone*

SaveHolo2L

Saves a combined hologram recorded in alternate mode at different wavelengths

Koala UI equivalent:
(not applicable because this mode is not supported anymore)

Parameters:

- path (String): Full path and name of the destination file.

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *SaveImageToFile*
- *SaveImageFloatToFile*

SetSurfLambdaCNm

Sets lambda C (the limit frequency between wavelengths considered as the rugosity and the wavelengths considered as the waviness of a topography), for topography measurements.

Koala UI equivalent:
(Not applicable, but roughness measurement parameters can be loaded via the *Surface topography Phase window.*)

Parameters:

- lambdaC (Single): Wavelength of lambda C, in [nm]

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *SetSurfLambdaSNm*

SetSurfLambdaSNm

Sets lambda S (the limit frequency between wavelengths considered as noise and the wavelengths considered as the rugosity of a topography), for topography measurements.

Koala UI equivalent:
(Not applicable, but roughness measurement parameters can be loaded via the *Surface topography Phase window.*)

Parameters:

- lambdaS (Single): Wavelength of lambda S, in [nm]

Return: void (nothing)

Requirements:

- A configuration must be loaded
- A user must be logged in

Possible errors:

- Current Koala state does not meet the requirements ⇒ *Invalid Operation* error

See also:

- *SetSurfLambdaCNm*

Appendix

Functions Modifications

GetHoloImage

GetHoloImage had a parameter corresponding to the wavelength used in Koala version 8.2. This parameter has been removed from Koala 8.4 as the hologram contains all the information needed (no alternate mode).
See *GetHoloImage* and *GetHoloImage (deprecated)* for more information.

RecordStroboFixFrequency ⇒ RecordStroboscopeFixedFrequency

The function **RecordStroboFixFrequency** used in Koala version 8.2 has been renamed to **RecordStroboscopeFixedFrequency** since Koala version 8.4. Usage remains the same.
See *RecordStroboscopeFixedFrequency* for more information.

RecordStroboFrequencyScan ⇒ RecordStroboscopeFrequencyScan

The function **RecordStroboFrequencyScan** used in Koala version 8.2 has been renamed to **RecordStroboscopeFrequencyScan** since Koala version 8.4. Usage remains the same.
See *RecordStroboscopeFrequencyScan* for more information.

ResetCorrSegment

The function **ResetCorrSegment** had a parameter corresponding to the dimension (1d or 2D) used in Koala version 8.2. Now, it only applies to 1D and takes no parameter. Usage remains the same.
See *ResetCorrSegment* for more information.

Python remote client application example

Note: this example is copied on your computer when you install Koala with the *Remote* option. You can find it in: *C:\Program Files\LynceeTec\Koala\Remote\Remote Examples\Python\PythonRemoteExample.py*

```
# -*- coding: utf-8 -*-

#*****************************************************************************
# NOTES:
# - This example is designed to work with a 2 wavelengths configuration.
#   If you use a single wavelength configuration, you will have to comment
#   some of the lines for the script to run.
#
# - The example grabs an hologram, saves it, and then loads it to illustrate
#   those functionalities. If you comment the grab and save part, you can
#   run the example offline.
#
# - You might need to adapt some of the parameters (such as the save/load
#   location or the default configuration number) for the example to work
#   on your system with your database
#*****************************************************************************

import sys
import clr #Install pythonnet package to get clr
from pathlib import Path
import numpy as np
import matplotlib.pyplot as plt
import ctypes

#Add Koala remote librairies to Path
sys.path.append(r'C:\Program Files\LynceeTec\Koala\Remote Libraries\x64')
#load x86 for 32 bits applications
#Import KoalaRemoteClient
clr.AddReference("LynceeTec.KoalaRemote.Client")
from LynceeTec.KoalaRemote.Client import KoalaRemoteClient

#get input form console, if input is empty us default value
def get_input(question,default) :
    answer = input(question+'['+default+'] :')
    return answer or default

#Define KoalaRemoteClient host
host=KoalaRemoteClient()

#Ask IP address
```

```
IP = get_input('Enter host IP address','localhost')
#Log on Koala
username = ''
[ret,username] = host.Connect(IP,username,True);
password = get_input('Enter password for '+username+' account', username)
host.Login(password)

#Open a 2 wavelengths configuration.
config = get_input('Enter configuration number', default='137')
host.OpenConfig(config);

#Open main display windows
host.OpenPhaseWin();
host.OpenIntensityWin();
host.OpenHoloWin();

#This block records an hologram so that you can later work offline with the
rest of the script.
#Set logical source 0 (the 1st source of the current configuration) to ON
host.SetSourceState(0, True, True)
#Set logical source 1 (the 2nd source of the current configuration) to ON
host.SetSourceState(1, True, True)
#Acquire on hologram
host.Acquisition2L()
host.ResetGrab()
#Save holo to file
path = Path(r'c:\tmp')
host.SaveImageToFile(1, str(path/'holo.tiff'))

#Load previously recorded hologram
host.LoadHolo(str(path/'holo.tiff'), 2);

#Display lambda 1 image in phase window
host.SelectDisplayWL(8192);

#Save hologram image
host.SaveImageToFile(1, str(path/'holo.tiff'));
#Save intensity image
host.SaveImageToFile(2, str(path/'intensity.tiff'));
#Save phase image (as displayed, which means phase lambda 1)
host.SaveImageToFile(4, str(path/'phase.tif'));
#//Save intensity float in bin
host.SaveImageFloatToFile(2, str(path/'intensity.bin'), True);
#//Save phase float in bin
host.SaveImageFloatToFile(4, str(path/'phase.bin'), True);

#//This block only works for 2 wavelengths configurations
#//Display lambda 2 image in intensity window
```

```
host.SelectDisplayWL(4096);
#//Display lambda 2 image in phase window
host.SelectDisplayWL(16384);
#//Save intensity image (as displayed, which means intensity lambda 2)
host.SaveImageToFile(2, str(path/'intensity2.tiff'));
#//Save phase image (as displayed, which means phase lambda 2)
host.SaveImageToFile(4, str(path/'phase2.tif'));
host.SaveImageFloatToFile(2, str(path/'intensity2.bin'), True);
host.SaveImageFloatToFile(4, str(path/'phase2.bin'), True);

#//Gets the current reconstruction distance
recDist = host.GetRecDistCM();
#//Set a new reconstruction distance
host.SetRecDistCM(recDist * 1.1);
#Do a reconstruction with this new distance
host.OnDistanceChange();

#//Get phase image for computation
roiWidth = host.GetPhaseWidth();
roiHeight = host.GetPhaseHeight();
roiStride = (roiWidth / 4) * 4;
size = int(roiStride * roiHeight)
if (roiWidth % 4 != 0) :
    roiStride += 4;
clr.AddReference("System")
import System
from System import Array
bufPhase=Array.CreateInstance(System.Single,size)
host.GetPhase32fImage(bufPhase);
#Convert DotNET Array to numpy Array
phase = np.zeros(size, dtype=float)
for x in range(0,(int)(size-1)):
    phase[x]=float(bufPhase[x])
phase = np.reshape(phase,(roiHeight,roiWidth))
#plot phase numpy
plt.imshow(phase,cmap="Greys")

#//Extract a profile
host.SetPhaseProfileState(True)
host.ExtractPhaseProfile(100, 100, 200, 200)
profileLength = host.GetPhaseProfileLength()
bufProfile = Array.CreateInstance(System.Double,profileLength)
host.GetPhaseProfile(bufProfile)
#Convert DotNET Array to numpy Array
profile = np.zeros(profileLength, dtype=np.double)
for x in range(0,(int)(profileLength-1)):
    profile[x]=float(bufProfile[x])
#Get xy values to plot calibrated profile
```

```
distance = np.arange(profileLength) * host.GetPxSizeUm()
plt.figure()
plt.plot(distance,profile)
plt.xlabel('dist [um]')
plt.ylabel('OPL [nm]')
plt.show()

#//Reset phase correction segments
host.ResetCorrSegment();
#Add new phase profile correction
host.AddCorrSegment(100, 100, 500, 1);
host.AddCorrSegment(200, 200, 600, 0);
#//Compute 1D phase correction using tilt method
host.ComputePhaseCorrection(0, 1);

#Logout
host.Logout()
```

Python remote application example C#

Note: this example is copied on your computer when you install **Koala Remote Test App**.

Script Examples

You can find them in:
C:\Program Files\LynceeTec\Koala Remote Test App\Scripts Examples CS.

List of scripts:

- *FastHologramsRecordScript.cs*
- *FilterSwitchScript.cs*
- *IntensityFilterScript.cs*
- *KoalaVersionScript.cs*
- *MeasurementInformationScript.cs*
- *PhaseGradientFilterScript.cs*
- *PhaseOffsetAdjustmentZoneScript.cs*
- *ReconstructionToDiskScript.cs*
- *SampleScript.cs*
- *SampleScriptForStroboscopeFixedFrequency.cs*
- *SampleScriptForStroboscopeFrequencyApproach.cs*
- *SampleScriptForStroboscopeFrequencyScan.cs*
- *SampleScriptForWindowsOpenClose.cs*
- *UserDefinedFilterScript.cs*
- *WavelengthFilterScript.cs*

SampleScript.cs

```
//css_reference LynceeTec.KoalaRemote.Client.dll;
using System;
using System.IO;
using LynceeTec.KoalaRemote.Client;

//*************************************************************************
// NOTES:
// - This script is designed to work with a 2 wavelengths configuration.
//   If you use a single wavelength configuration, you will have to comment
//   some of the lines for the script to run.
//
// - The script grabs an hologram, saves it, and then loads it to illustrate
//   these functionalities. If you comment the grab and save part, you can
//   run the script offline.
//
// - You might need to adapt some of the parameters (such as the save/load
//   location or the configuration number) for the script to work on your
//   system with your database
//*************************************************************************

class Test : IKoalaScript
{
   public void Run(KoalaRemoteClient host)
   {
      string pathToSaveData = @"C:\tmp";

      //Open a 2 wavelenghts configuration.
      host.OpenConfig(137);

      //Open main display windows
      host.OpenPhaseWin();
      host.OpenIntensityWin();
      host.OpenHoloWin();

      //This block records an hologram so that you can later work offline
      //with the rest of the script.
      //Set logical source 0 (the 1st source of the current configuration) to ON
      host.SetSourceState(0, true, true);
      //Set logical source 1 (the 2nd source of the current configuration) to ON
      host.SetSourceState(1, true, true);
```

```
host.Acquisition2L();
host.ResetGrab();
host.SaveImageToFile(1, Path.Combine(pathToSaveData, "holo.tif"));

//Load previously recorded hologram
host.LoadHolo(Path.Combine(pathToSaveData, "holo.tif"), 2);

//Display lambda 1 image in phase window
host.SelectDisplayWL(8192);

//Save hologram image
host.SaveImageToFile(1, Path.Combine(pathToSaveData, "holo.tif"));
//Save intensity image
host.SaveImageToFile(2, Path.Combine(pathToSaveData, "intensity.tif"));
//Save phase image (as displayed, which means phase lamabda 1)
host.SaveImageToFile(4, Path.Combine(pathToSaveData, "phase.tif"));
//Save intensity float in bin
host.SaveImageFloatToFile(2, Path.Combine(pathToSaveData, "intensity.bin"), true);
//Save phase float in bin
host.SaveImageFloatToFile(4, Path.Combine(pathToSaveData, "phase.bin"), true);

//This block only works for 2 wavelengths configurations
//Display lambda 2 image in intensity window
host.SelectDisplayWL(4096);
//Display lambda 2 image in phase window
host.SelectDisplayWL(16384);
//Save intensity image (as displayed, which means intensity lamabda 2)
host.SaveImageToFile(2, Path.Combine(pathToSaveData, "intensity2.tif"));
//Save phase image (as displayed, which means phase lamabda 2)
host.SaveImageToFile(4, Path.Combine(pathToSaveData, "phase2.tif"));
host.SaveImageFloatToFile(2, Path.Combine(pathToSaveData, "intensity2.bin"), true);
host.SaveImageFloatToFile(4, Path.Combine(pathToSaveData, "phase2.bin"), true);

//Gets the current reconstruction distance
float recDist = host.GetRecDistCM();
//Set a new reconstruction distance
host.SetRecDistCM(recDist * 1.1f);
//Do a reconstruction with this new distance
host.OnDistanceChange();

//Get phase image for computation
int roiWidth = host.GetPhaseWidth();
int roiHeight = host.GetPhaseHeight();
```

```
        int roiStride = (roiWidth / 4) * 4;
        if (roiWidth % 4 != 0)
            roiStride += 4;
        float[] phaseBuffer = new float[roiStride * roiHeight];
        host.GetPhase32fImage(phaseBuffer);

        //Extract a phase profile
        host.SetPhaseProfileState(true);
        host.ExtractPhaseProfile(100, 100, 200, 200);
        int phaseProfileLength = host.GetPhaseProfileLength();
        double[] phaseProfile = new double[phaseProfileLength];
        host.GetPhaseProfile(phaseProfile);

        using (StreamWriter swPh = new StreamWriter(new
FileStream(Path.Combine(pathToSaveData, "phaseProfile.txt"), FileMode.Create)))
        {
            for (int i = 0; i < phaseProfile.Length; i++)
            {
                swPh.Write(phaseProfile[i]);
                swPh.Write("\n");
            }
        }


        //Extract an intensity profile
        host.SetIntensityProfileState(true);
        host.ExtractIntensityProfile(100, 100, 200, 200);
        int intensityProfileLength = host.GetIntensityProfileLength();
        double[] intensityProfile = new double[intensityProfileLength];
        host.GetIntensityProfile(intensityProfile);

        using (StreamWriter swInt = new StreamWriter(new
FileStream(Path.Combine(pathToSaveData, "intensityProfile.txt"), FileMode.Create)))
        {
            for (int i = 0; i < intensityProfile.Length; i++)
            {
                swInt.Write(intensityProfile[i]);
                swInt.Write("\n");
            }
        }

        ////Enable code if you want to reset the phase correction segments
        ////It has an influence on the above profiles
```

```
        ////Reset phase correction segments
        //host.ResetCorrSegment();
        //host.AddCorrSegment(100, 100, 500, 1);
        //host.AddCorrSegment(200, 200, 600, 0);
        ////Compute 1D phase correction using tilt method
        //host.ComputePhaseCorrection(0, 1);
    }
}
```

SampleScriptForStroboscopeFixedFrequency.cs

```
//css_reference LynceeTec.KoalaRemote.Client.dll;
using System.IO;
using System.Threading;
using System.Threading.Tasks;
using LynceeTec.KoalaRemote.Client;

//*****************************************************************************
// NOTES:
// - This script is designed to work with a 1 or 2 wavelength(s) configuration.
//   If you use a single wavelength configuration, you will have to comment
//   some of the lines for the script to run.
//
// - The script demonstrates the principal functionalities on how to use the
//   LyncéeTec strobocopic unit remotely.
//
// - You will need to adapt the parameters to work with your specific case
//
//*****************************************************************************

class TestStroboscopeFixedFrequency : IKoalaScript
{
   public void Run(KoalaRemoteClient host)
   {
      //Choose which configuration you want to use
      //The configuration number is shown in the Open configuration window in Koala.

      //Open a 2 wavelengths configuration.
      //host.OpenConfig(137);

      //Open a 1 wavelength configuration.
      host.OpenConfig(147);

      //Open main display windows
      //Phase window
      host.OpenPhaseWin();
      //Intensity window
      host.OpenIntensityWin();
      //Hologram window
      host.OpenHoloWin();
      //Stroboscopic window
```

```
host.OpenStroboWin();

//Example on how to set record parameters in stroboscopic window
//Here, we choose to record the phase and intensity data in bin and tiff format
//SetStroboscopeRecordDataType(recordPhaseAsBin, recordPhaseAsTiff,
recordIntensityAsBin, recordIntensityAsTiff)
host.SetStroboscopeRecordDataType(true, true, true, true);
//Gives the root path to record the data
host.SetStroboscopeRecordPath(@"C:\test");
//Sets the stroboscopic tool to record immediately on start.
host.SetStroboscopeRecordAtStartStatus(true);

////////////////////////////////////////////////////////////////
//Example on how to give a fixed frequency to the stroboscope (between 1Hz and 25MHz)
//Frequency = 15kHz
//host.SetStroboscopeFixedFrequency(15000);
//Frequency = 7MHz
host.SetStroboscopeFixedFrequency(7000000);
////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////
//Sets the number of samples per period
host.SetStroboscopeNumberOfSamplesPerPeriod(16);
////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////
//ANGLE STEP
//Modifying the angle step will also modify the valid frequency range where the stroboscope
can work
//and the maximal number of samples per period

//Increase the angle step (move the green bar from the frequency slider towards higher
frequencies)
host.IncreaseStroboscopeAngleStep();

//Decrease the angle step (move the green bar from the frequency slider towards lower
frequencies)
host.DecreaseStroboscopeAngleStep();
////////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////////
//Modify the number of samples per period to their maximum admissible value for a given
frequency range
```

```
        //indicated by the green bar on the frequency slider.
        host.MaximizeStroboscopeNumberOfSamples();
        ////////////////////////////////////////////////////////////

        ////////////////////////////////////////////////////////////
        //Modify the channel 1 parameters
        //SetStroboscopeChannel1Parameters(channelEnabled, chosenWaveform, voltage_in_mV,
offset_in_mV, phaseDelay_deg, offsetType)
        //Example: Activate channel 1, select wavelength 1, put 5.5V as voltage, -2V as offset,
phase delay 57 degrees, and manual offset
        host.SetStroboscopeChannel1Parameters(true, 1, 5500, -2000, 57, 0);

        //Example: Activate channel 1, select wavelength 2, put 5.5V as voltage, -2V as offset,
phase delay 57 degrees, and offset as 0
        //host.SetStroboscopeChannel1Parameters(true, 2, 5500, -2000, 57, 1);

        //Example: Activate channel 1, select wavelength 3, put 5.5V as voltage, -2V as offset,
phase delay 57 degrees, and offset as (V<0): this offset type will induce a modification of the
real offset applied.
        //host.SetStroboscopeChannel1Parameters(true, 3, 5500, -2000, 57, 2);

        //Example: Activate channel 1, select wavelength 4, put 5.5V as voltage, -2V as offset,
phase delay 57 degrees, and offset as (V>0): this offset type will induce a modification of the
real offset applied.
        //host.SetStroboscopeChannel1Parameters(true, 4, 5500, -2000, 57, 3);
        ////////////////////////////////////////////////////////////

        ////////////////////////////////////////////////////////////
        //Set the laser pulse duty cycle in percent
        host.SetStroboscopeLaserPulseDutyCycle(20);
        ////////////////////////////////////////////////////////////

        ////////////////////////////////////////////////////////////
        //Start the stroboscope at a fixed frequency in continuous mode
        host.StartStroboscopeFixedFrequency(0, -1);
        //Wait 10s
        Thread.Sleep(10000);
        ////////////////////////////////////////////////////////////

        ////////////////////////////////////////////////////////////
        //Modify the laser pulse duty cycle from 20% to 10%
        host.SetStroboscopeLaserPulseDutyCycle(10);
        //Apply the new duty cycle
```

```
        host.ApplyNewDutyCycleInLiveMode();
        ////////////////////////////////////////////////////////////

        ////////////////////////////////////////////////////////////
        //Modify channel 1, select wavelength 1, put 6.5V as voltage, 0V as offset, phase delay 90
degrees, and manual offset
        host.SetStroboscopeChannel1Parameters(true, 1, 6500, 0, 90, 0);
        //Apply the new channel 1 parameters
        host.ApplyNewVoltageAndOffsetInLiveModeForChannelNumber(1);
        //Wait 10s
        Thread.Sleep(10000);
        ////////////////////////////////////////////////////////////

        ////////////////////////////////////////////////////////////
        //Stop the stroboscope
        host.StopStroboscope();
        ////////////////////////////////////////////////////////////

        Thread.Sleep(2000);

        ////////////////////////////////////////////////////////////
        //Start the stroboscope at a fixed frequency for 2 periods
        host.StartStroboscopeFixedFrequency(1, 2);

        Thread.Sleep(4000);
        //OR
        ////////////////////////////////////////////////////////////
        //Record the stroboscope at a fixed frequency for 2 periods
        //host.RecordStroboscopeFixedFrequency(2);

        ////////////////////////////////////////////////////////////
        //Close the stroboscopic window
        host.CloseStroboWin();
        ////////////////////////////////////////////////////////////
    }
}
```

SampleScriptForStroboscopeFrequencyScan.cs

```
//css_reference LynceeTec.KoalaRemote.Client.dll;
using System.IO;
using System.Threading;
using System.Threading.Tasks;
using LynceeTec.KoalaRemote.Client;

//*****************************************************************************
// NOTES:
// - This script is designed to work with a 1 or 2 wavelength(s) configuration.
//   If you use a single wavelength configuration, you will have to comment
//   some of the lines for the script to run.
//
// - The script demonstrates the principal functionalities on how to use the
//   LyncéeTec strobocopic unit remotely.
//
// - You will need to adapt the parameters to work with your specific case
//
//*****************************************************************************

class TestStroboscopeFrequencyScan : IKoalaScript
{
   public void Run(KoalaRemoteClient host)
   {
      //Choose which configuration you want to use
      //The configuration number is shown in the Open configuration window in Koala.

      //Open a 2 wavelengths configuration.
      //host.OpenConfig(137);

      //Open a 1 wavelength configuration.
      host.OpenConfig(147);

      //Open main display windows
      //Phase window
      host.OpenPhaseWin();
      //Intensity window
      host.OpenIntensityWin();
      //Hologram window
      host.OpenHoloWin();
      //Stroboscopic window
```

```
host.OpenStroboWin();

////////////////////////////////////////////////////////////
//Example on how to set record parameters in stroboscopic window
//Here, we choose to record the phase and intensity data in bin and tiff format
//SetStroboscopeRecordDataType(recordPhaseAsBin, recordPhaseAsTiff,
recordIntensityAsBin, recordIntensityAsTiff)
host.SetStroboscopeRecordDataType(true, true, true, true);
//Gives the root path to record the data
host.SetStroboscopeRecordPath(@"C:\test");
////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////
//Modify the channel 1 parameters
//SetStroboscopeChannel1Parameters(channelEnabled, chosenWaveform, voltage_in_mV,
offset_in_mV, phaseDelay_deg, offsetType)
//Example: Activate channel 1, select wavelength 1, put 8V as voltage, -2V as offset, phase
delay 57 degrees, and manual offset
host.SetStroboscopeChannel1Parameters(true, 1, 8000, -2000, 90, 0);
////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////
//Set the laser pulse duty cycle in percent
host.SetStroboscopeLaserPulseDutyCycle(20);
////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////
//For the frequency scan to work correctly, we need to set the fixed frequency to the target
frequency to ensure the stroboscope is in the correct frequency range.
//Frequency = 7MHz
host.SetStroboscopeFixedFrequency(7000000);
////////////////////////////////////////////////////////////

////////////////////////////////////////////////////////////
//Increase the angle step (move the green bar from the frequency slider towards higher
frequencies
host.IncreaseStroboscopeAngleStep();

//Decrease the angle step (move the green bar from the frequency slider towards lower
frequencies
host.DecreaseStroboscopeAngleStep();

//host.SetStroboscopeNumberOfSamplesPerPeriod(16);
```

```
        //Modify the number of samples per period to their maximum admissible value for a given
frequency range
        //indicated by the green bar on the frequency slider.
        host.MaximizeStroboscopeNumberOfSamples();
        ////////////////////////////////////////////////////////////

        ////////////////////////////////////////////////////////////
        //Frequency Scan
        //Enable the frequency scan mode
        host.SetStroboscopeFrequencyScanEnabled(true);

        //Set the frequency scan from 6MHz and 7MHz by increasing the frequency by steps of
100kHz, for 2 periods per frequency
        host.SetStroboscopeFrequencyScanParameters(6000000, 7000000, 100000, 2, false);

        //Start the frequency scan recording
        host.RecordStroboscopeFrequencyScan();

        //Stop the frequency scan mode
        host.SetStroboscopeFrequencyScanEnabled(false);
        ////////////////////////////////////////////////////////////

        //Close the stroboscopic window
        //host.CloseStroboWin();
    }
}
```

PhaseOffsetAdjustmentZoneScript.cs

```
//css_reference LynceeTec.KoalaRemote.Client.dll;
using System.IO;
using LynceeTec.KoalaRemote.Client;

//*************************************************************************
// NOTES:
// - This script is designed to work with a 1 wavelength configuration.
//
// - The script opens an hologram, and then add 2 phase offset adjustment
//   zones.
//*************************************************************************

class TestPhaseOffsetAdjustmentZone : IKoalaScript
{
   public void Run(KoalaRemoteClient host)
   {
      //Open a 1 wavelength configuration.
      host.OpenConfig(147);

      host.LoadHolo(@"C:\ProgramData\LynceeTec\Koala\Holograms\Scratch_holo.tif", 1);

      //Open main display windows
      host.OpenHoloWin();
      host.OpenPhaseWin();
      host.OpenIntensityWin();

      //To add a phase offset adjustment zone
      host.AddPhaseOffsetAdjustmentZone(100, 200, 20, 100);
      host.AddPhaseOffsetAdjustmentZone(100, 300, 20, 100);

      //To remove all phase offset adjustment zones, uncomment the next line
      //host.ResetPhaseOffsetAdjustmentZone();
   }
}
```

WavelengthFilterScript.cs

```
//css_reference LynceeTec.KoalaRemote.Client.dll;
using System.IO;
using System.Threading;
using System.Threading.Tasks;
using LynceeTec.KoalaRemote.Client;

//*****************************************************************************
// NOTES:
// - This script is designed to work with a 1 wavelength configuration.
//
// - The script opens an hologram, and then apply given mask settings for the
//   wavelength filter.
//
//*****************************************************************************

class TestWavelengthFilter : IKoalaScript
{
   public void Run(KoalaRemoteClient host)
   {
      //Open a 1 wavelength configuration.
      host.OpenConfig(147);

      host.LoadHolo(@"C:\ProgramData\LynceeTec\Koala\Holograms\Scratch_holo.tif", 1);

      //Open main display windows
      host.OpenPhaseWin();
      host.OpenIntensityWin();
      host.OpenHoloWin();

      //To open the mask settings window.
      host.OpenMaskSettingsWin();

      //Enable the wavelength filter
      host.SetWavelengthFilterState(true);
      //Set the wavelength filter to Long-Pass
      host.SetWavelengthFilterType(1);
      //Set the wavelength filter minimal cutoff value to 30.
      host.SetWavelengthFilterMinimalCutOffValue(30.0f);

      Thread.Sleep(5000);
```

```
        //Set the wavelength filter maximal cutoff value to 100 applied for Short-Pass filter
        host.SetWavelengthFilterType(2);
        host.SetWavelengthFilterMaximalCutOffValue(100.0f);

        Thread.Sleep(5000);
        //Set the wavelength filter to Band-Pass filter
        host.SetWavelengthFilterType(3);
        host.SetWavelengthFilterMinimalCutOffValue(10.0f);
        host.SetWavelengthFilterMaximalCutOffValue(50.0f);

        Thread.Sleep(5000);
        //Set the wavelength filter to Band-Stop filter
        host.SetWavelengthFilterType(4);
        host.SetWavelengthFilterMinimalCutOffValue(40.0f);
        host.SetWavelengthFilterMaximalCutOffValue(80.0f);


        //To close the mask settings window. Mask settings continue to be applied as they were
defined.
        host.CloseMaskSettingsWin();

        //Open the mask settings window. You can verify that the wavelength filter settings remain
identical.
        host.OpenMaskSettingsWin();
    }
}
```