THORLABS

Scientific Camera LabVIEW .NET Interface Guide

For Kiralux, Quantalux, Zelux, and all Scientific CCD Camera Models

Table of Contents

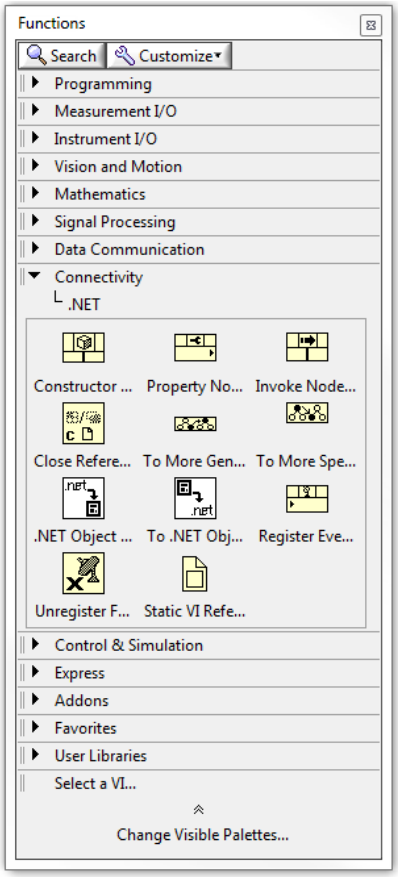
Chapter 1 Introduction

1.1. Overview

The Thorlabs Scientific Camera LabVIEW .NET Manual contains information to help you program your own LabVIEW applications to operate your Thorlabs Scientific Camera. The Thorlabs Scientific Camera .NET interface is part of the Thorlabs Scientific Camera software packages that can be downloaded for free on the Thorlabs website. In addition to drivers and documentations, the software package includes Software Development Kit (SDK) for creating your own applications. Code samples are also provided to make it easier to start programming with your Thorlabs Scientific Camera.

1.2. LabVIEW .NET connectivity

The LabVIEW .NET connectivity functions allows you to create and interact with the Thorlabs Scientific Camera .NET SDK. On the Functions Palette, select “Connectivity » .NET” to access all functions used to interact with .NET objects. For more information on LabVIEW .NET connectivity, please refer to LabVIEW Manuals.



1.3. Thorlabs Scientific Camera .NET Application Programming Interface (API)

For complete information on programming with Thorlabs Scientific Camera .NET SDK, please refer to the *DotNet, LabVIEW, and MATLAB scientific-camera programming guide.chm* help file.

Chapter 2 Getting Started in LabVIEW

2.1. Before you begin

National Instruments strongly recommends that you use .NET objects only in LabVIEW projects.

It is important that you call the “Close Reference” function after finish using a .NET object or reference to prevent memory leaks.

TLCameraSDK and TLCamera objects need to be properly disposed by calling their respective Dispose methods before closing the references or exiting the .vi script.

LabVIEW removes references if the top level VI that opened the reference stops execution.

Please make sure to use the correct 32-bit or 64-bit libraries corresponding to the version of LabVIEW you are using.

If you are migrating VIs based on the version 1.0 of the Thorlabs Scientific Camera .NET interface, please refer to Section 2.10 on accessing image data.

2.2. Reference Thorlabs Scientific Camera .NET assembly

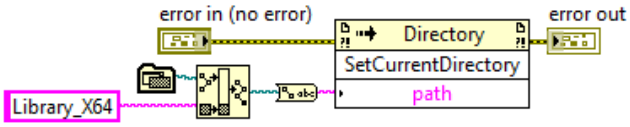
Create a LabVIEW project, and add all .dll libraries into a folder in the project.

Reference “Thorlabs.TSI.TLCamera.dll” with LabVIEW .NET connectivity to access every function and property of the Thorlabs Scientific Camera .NET API.

Alternatively, sub VIs that wraps most functions and properties of the Thorlabs Scientific Camera .NET API are available in the provided LabVIEW code samples.

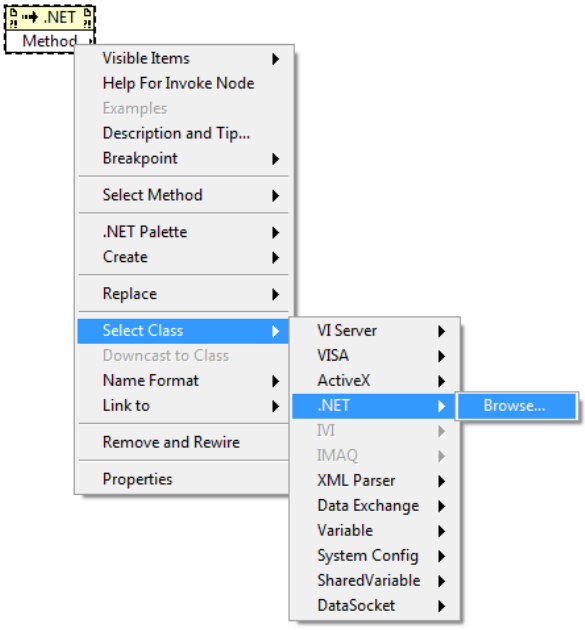
2.3. Set the Current Directory to load libraries.

IMPORTANT: In order to properly load all referenced libraries, the current directory needs to be set to the location where you store the libraries. This can be achieved by calling System.IO.Directory.SetCurrentDirectory(String) method (in mscorlib Assembly). See the code samples for an example of this.

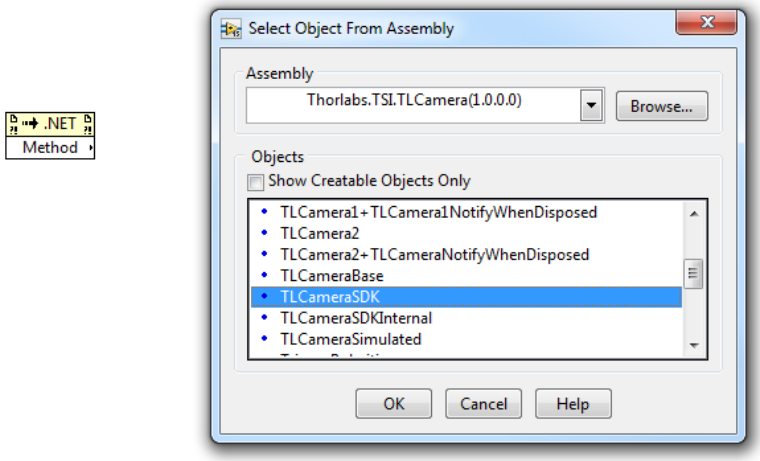


2.4. Create an instance of ITLCameraSDK

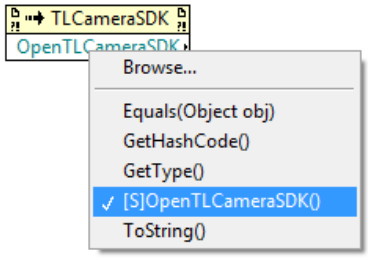
You can create an instance of an “ITLCameraSDK” object by calling the static method “Thorlabs.TSI.TLCamera.TLCameraSDK.OpenTLCameraSDK”. To call the static method, on the block diagram, add a “Connectivity » .NET » Invoke Node (.NET)”. Right click on the node, click “Select Class » .NET » Browse”.



Browse and select “Thorlabs.TSI.TLCamera.dll”, then select “TLCameraSDK”.

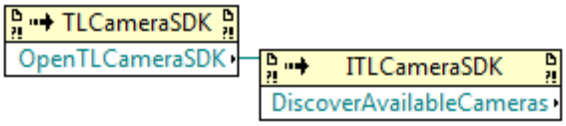


From the “Invoke Node” method drop down menu, select “OpenTLCameraSDK”. The method returns the reference to a new instance of TLCameraSDK.



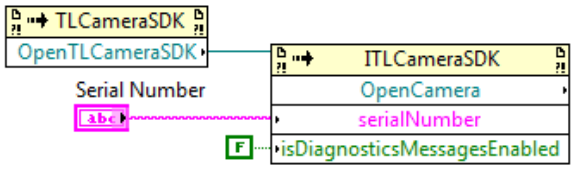
2.5. Discover connected Thorlabs scientific cameras (Optional)

Connect an “Invoke Node (.NET)” to the “TLCameraSDK” reference and select “DiscoverAvailableCameras” method. The method returns a list of serial numbers of connected cameras.



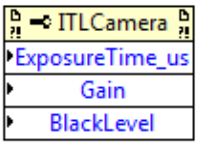
2.6. Open a camera

Connect an “Invoke Node (.NET)” to the “TLCameraSDK” reference and select “OpenCamera” method with a serial number. The method returns a reference to an ITLCamera.



2.7. Set and get camera properties

Connect a “Property Node (.NET)” to the ITLCamera reference. The Property Node shows a list of properties that can be read and written. For detailed descriptions of the properties please refer to the API Help.



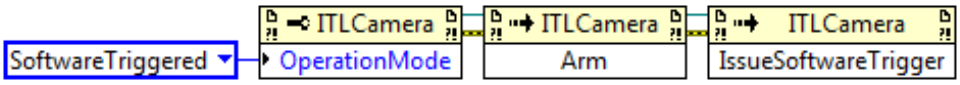
2.8. Start the camera

Connect a “Property Node” to the TLCamera reference to set the “OperationMode” parameter of the camera to “SoftwareTriggered” (default), “HardwareTriggered”, or “Bulb” mode. Then connect an “Invoke Node” to the camera reference and call “Arm”. This prepares the camera to acquire images in response to software or hardware triggers according to “OperationMode”. If the camera is set to “SoftwareTriggered” operation mode, a software trigger can be issued by using an “Invoke Node” and call “TLCamera.IssueSoftwareTrigger”.

Issuing a software trigger can do one of two things:

1. If you have set the FramesPerTrigger_zeroForUnlimited property to zero, it will start continuous-acquisition mode.

2. If you have set the FramesPerTrigger_zeroForUnlimited property to one, it will trigger one frame and then await the next software trigger.



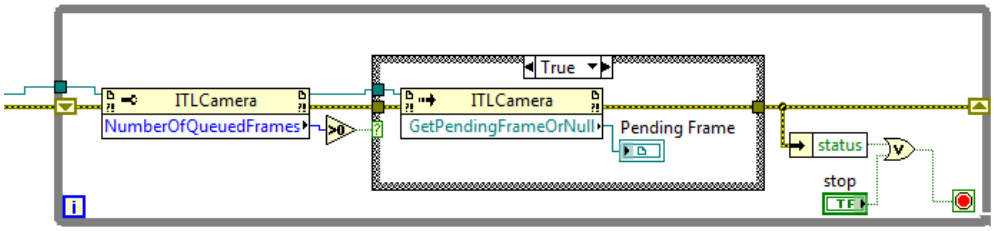
2.9. Get image from camera

There are two choices for getting image data from the camera: polling or registering a callback.

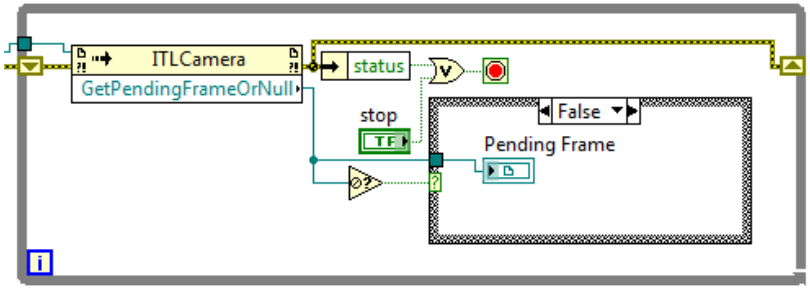
2.9.1. Poll image frame from camera

Query in a loop whether an image is available on the camera, and then retrieve the image. This can be done in two ways:

1. Query “TLCamera.NumberOfQueuedFrames” using a “Property Node”, wait until the number is larger than zero, and call “TLCamera.GetPendingFrameOrNull” with an “Invoke Node” to get the pending image frame;

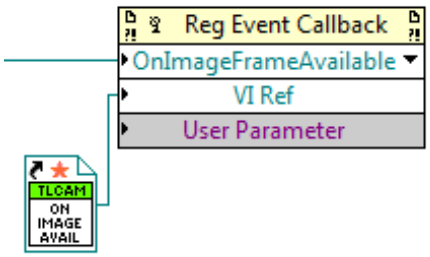


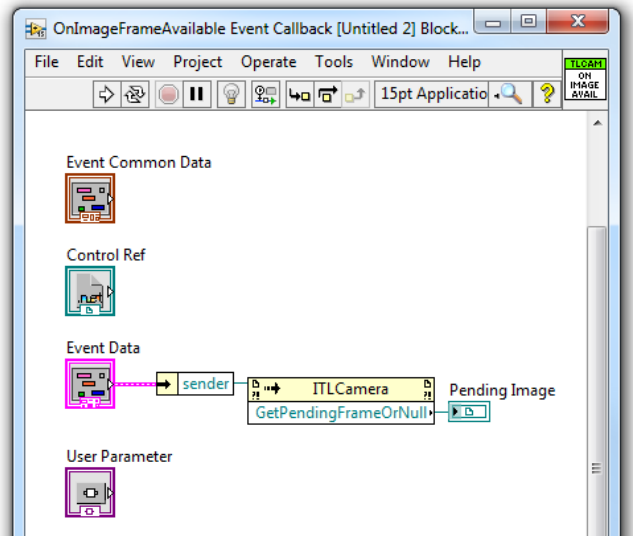
2. Directly call “TLCamera. GetPendingFrameOrNull”, and check if the returned frame is null.



2.9.2. Register camera event callback

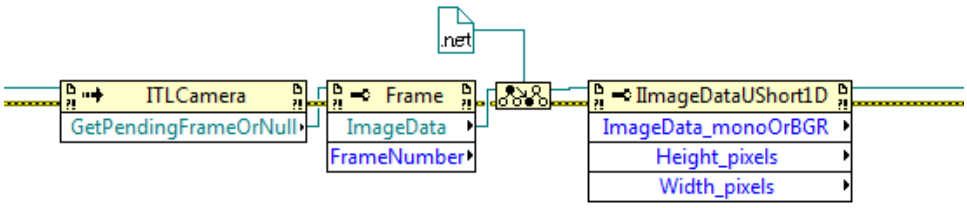
Use the “Register Event Callback” function to register for the “TLCamera.OnImageFrameAvailable” event. When you receive the event, call “TLCamera.GetPendingFrameOrNull” in the event callback vi to retrieve the image frame.



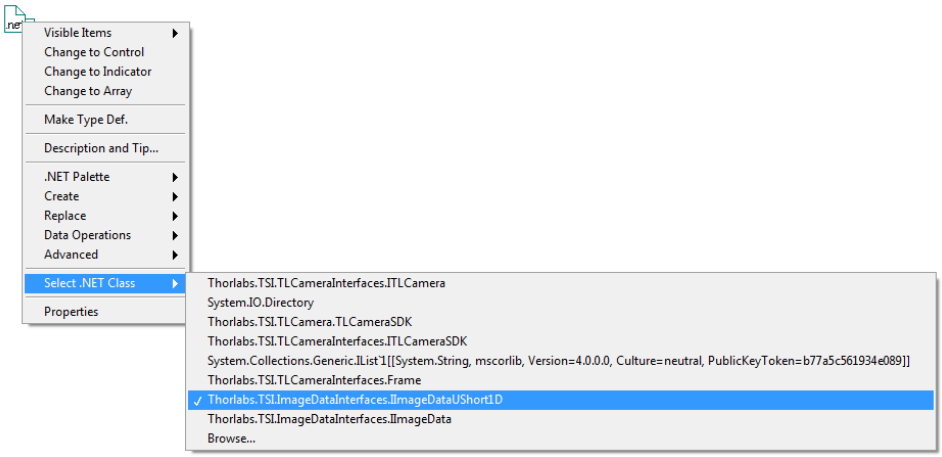


2.10. Accessing image data

The image data is stored in the ImageData property of the frame. In order to access the data as UInt16 array, the ImageData property needs to be converted to the IImageDataUShort1D interface using “Connectivity » .NET » To More Specific Class”. Connect ImageData to the “reference” input, and create a constant .NET object at the “target class” terminal.



Right click on the constant .NET object, and use the “Select .NET Class” menu option to set its class to “Thorlabs.TSI.ImageDataInterfaces.IImageDataUShort1D”. If the option is not immediately available in the list, please use “Browse…” to select “Thorlabs.TSI.ImageDataInterefaces.dll” and then select “IImageDataUShort1D”.



After the conversion, the image data can be accessed as the “ImageData_monoOrBGR” property.

2.11. Color and polarization processing of image data

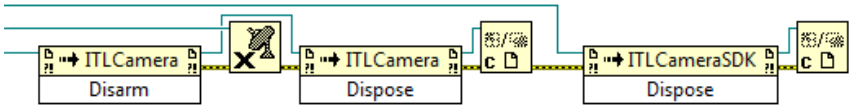
For color or polarization cameras, the unprocessed monochrome Bayer image data from the camera can be processed by stand-alone image processing modules to color or polarization images. Please refer to included code samples for using the image processing modules.

2.12. Stop the camera

When you are finished acquiring images with the camera, use an “Invoke Node” to call “TLCamera.Disarm”.

2.13. Close and dispose the camera

When you are done using a camera, use “Unregister For Events” to unregister any registered camera events, then connect an “Invoke Node” to the TLCamera reference and call “TLCamera.Dispose”, and finally use “Close Reference” to close the refnum associated with the TLCamera object. When you are done using all cameras, connect an “Invoke Node” to the TLCameraSDK reference and call “TLCameraSDK.Dispose”, then use “Close Reference” to close the refnum associated with the TLCameraSDK object.



Chapter 3 LabVIEW code samples

The following LabVIEW sample codes are installed with the Thorlabs Scientific Camera software package. They are arranged in a LabVIEW Project. You can find the sample codes under “Program Files\Thorlabs\Scientific Imaging\Scientific Camera Support\Third party software support\LabVIEW\TLCameraSamples”.

Example	Description
Simple image acquisition mono - polling	Starts the camera and gets images by polling. The sample shows camera initialization, image acquisition and camera closing.
Simple image acquisition mono - events	Starts the camera and gets images by registering for events. The sample shows how to use the .NET events provided by the camera.
Image acquisition color 8-bit RGB - polling	Demonstrates how to use color processing with a color camera. The sample shows how to create color processor, set up white balance correction, and process Bayer patterned image data from the camera. The image data is converted to an 8-bit RGB PixMap for display.
Image acquisition color 16-bit RGB data - polling	Demonstrates how to use a color camera. The acquired color image data is separated into Red, Green and Blue components at original bit depth.
Image acquisition polarization camera - polling	Demonstrates how to use a polarization camera. The sample shows how to convert Bayer patterned raw image data from the camera into Azimuth, Degree of Linear Polarization (DoLP) or Intensity images.
Software triggered acquisition - polling	Demonstrates how to use software triggered acquisition.
Hardware triggered acquisition - polling	Demonstrates how to use hardware triggered acquisition.
TLCamera Main	A comprehensive code sample using Event Structure to demonstrate common camera operations.
\Controls	Type defined controls.
\SubVIs	Sub VIs that perform certain functions or wraps .NET properties or function calls.

THORLABS

www.thorlabs.com