Thorlabs .NET, LabVIEW, and MATLAB Camera SDK

Rev. 2.0.2 2022-02-25 ITN002885-D01

Contents

0.1 Thorlabs Camera SDK

0.1.1 Getting Started With .NET

Getting started programming Thorlabs scientific cameras is easy. This guide describes how to program Thorlabs scientific cameras using .NET languages such as C# or VB.NET.

- Create an instance of ITLCameraSDK by calling Thorlabs.TSI.TLCamera.TLCameraSDK.OpenTLCameraSDK().
- Use the Thorlabs.TSI.TLCameraInterfaces.ITLCameraSDK object to discover the serial numbers of any connected cameras.
- Use the Thorlabs.TSI.TLCameraInterfaces.ITLCameraSDK object to open a camera.
- Set camera properties like exposure, ROI, binning, and trigger mode.
- Set the Thorlabs.TSI.TLCameraInterfaces.ITLCamera.OperationMode to Thorlabs.TSI.TLCameraInterfaces.OperationMode.So Thorlabs.TSI.TLCameraInterfaces.OperationMode.HardwareTriggered, or Thorlabs.TSI.TLCameraInterfaces.OperationMode.B
- For the next step, there are two choices for getting image data from the camera: polling or registering a callback.

The polling approach:

- Call Thorlabs.TSI.TLCameraInterfaces.ITLCamera.GetPendingFrameOrNull in a loop to get the latest image data.

The callback approach:

- Register for the Thorlabs.TSI.TLCameraInterfaces.ITLCamera.OnImageFrameAvailable event. In the event handler, call Thorlabs.TSI.TLCameraInterfaces.ITLCamera.GetPendingFrameOrNull. Note that the event will arrive on a worker thread.

- Prepare the camera for software or hardware triggers by calling Thorlabs.TSI.TLCameraInterfaces.ITLCamera.Arm().
- Depending on the OperationMode chosen above, call Thorlabs.TSI.TLCameraInterfaces.ITLCamera.IssueSoftwareTrigger or issue a hardware trigger.
- When finished issuing triggers, call Thorlabs.TSI.TLCameraInterfaces.ITLCamera.Disarm().
- Dispose the Thorlabs.TSI.TLCameraInterfaces.ITLCamera object.
- Dispose the Thorlabs.TSI.TLCameraInterfaces.ITLCameraSDK object.

0.1.2 Getting Started With MATLAB

MATLAB programming of Thorlabs scientific cameras is supported through MATLAB .NET interface. Please refer to "Thorlabs Scientific Camera MATLAB .NET Interface Guide" and included Matlab sample scripts for more details.

0.1.3 Getting Started With LabVIEW

LabVIEW programming of Thorlabs scientific cameras is supported through LabVIEW .NET connectivity. Please refer to "Thorlabs Scientific Camera LabVIEW .NET Interface Guide" and included LabVIEW code samples for more details.

0.2 Namespace Index

0.2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

0.3 Hierarchical Index

0.3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

0.4 Class Index

0.4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

0.5 Namespace Documentation

0.5.1 Thorlabs Namespace Reference

Namespaces

- namespace TSI

0.5.2 Thorlabs.TSI Namespace Reference

Namespaces

- namespace ColorInterfaces
- namespace ColorProcessor
- namespace Demosaicker
- namespace ImageData
- namespace PolarizationInterfaces
- namespace TLCamera
- namespace TLCameraInterfaces

0.5.3 Thorlabs.TSI.ColorInterfaces Namespace Reference

Classes

- interface IColorProcessor
- interface IColorProcessorSDK
- interface IDemosaicker

Enumerations

- enum ColorFilterArrayPhase { ColorFilterArrayPhase.BayerRed = 0, ColorFilterArrayPhase.BayerBlue = 1, ColorFilterArrayPhase.BayerGreenLeftOfRed = 2, ColorFilterArrayPhase.BayerGreenLeftOfBlue = 3 }
- enum ColorFormat { ColorFormat.BGRPlanar = 0, ColorFormat.BGRPixel = 1, ColorFormat.RGBPixel = 2 }
- enum ColorSensorType { ColorSensorType.Bayer }

0.5.3.1 Enumeration Type Documentation

0.5.3.1.1 ColorFilterArrayPhase

`enum Thorlabs.TSI.ColorInterfaces.ColorFilterArrayPhase [strong]`

Thorlabs color scientific cameras use a Bayer filter over the camera sensor to distinguish red, green, and blue intensities. This enumeration indicates how the Bayer filter is aligned over the sensor by indicating which pixel of the two-by-two Bayer pattern is aligned with the top-left pixel of the camera sensor.

Enumerator

BayerRed	bayer pattern - red
BayerBlue	bayer pattern - blue
BayerGreenLeftOfRed	bayer pattern - green left of red
BayerGreenLeftOfBlue	bayer pattern - green left of blue

0.5.3.1.2 ColorFormat

`enum Thorlabs.TSI.ColorInterfaces.ColorFormat [strong]`

The sequence of color channels and pixels in memory.

Enumerator

BGRPlanar	Data in BBBBB...GGGGG...RRRRR... All of the blue values appear first, then all of the green values, then all of the red values.
BGRPixel	Data in BGRBGRBGR... The one-dimensional image data interleaves blue, green, and red values in a repeating pattern.
RGBPixel	Data in RGBRGBRGB... The one-dimensional image data interleaves red, green, and blue values in a repeating pattern.

0.5.3.1.3 ColorSensorType

`enum Thorlabs.TSI.ColorInterfaces.ColorSensorType [strong]`

The physical sensor design for achieving color-imaging results.

Enumerator

Bayer	Thorlabs color scientific cameras use a Bayer filter over the camera sensor to distinguish red, green, and blue intensities.

0.5.4 Thorlabs.TSI.ColorProcessor Namespace Reference

Classes

- class ColorProcessor
- class ColorProcessorSDK

0.5.5 Thorlabs.TSI.Demosaicker Namespace Reference

Classes

- class Demosaicker

0.5.6 Thorlabs.TSI.ImageData Namespace Reference

Classes

- class ImageDataUShort1D

0.5.7 Thorlabs.TSI.PolarizationInterfaces Namespace Reference

Classes

- interface IPolarizationProcessor
- interface IPolarizationProcessorSDK

Enumerations

- enum PolarPhase { PolarPhase.PolarPhase0 = 0, PolarPhase.PolarPhase45 = 1, PolarPhase.PolarPhase90 = 2, PolarPhase.PolarPhase135 = 3 }

0.5.7.1 Enumeration Type Documentation

0.5.7.1.1 PolarPhase

`enum Thorlabs.TSI.PolarizationInterfaces.PolarPhase [strong]`

The possible polarization angle values (in degrees) for a pixel in a polarization sensor. The polarization phase pattern of the sensor is

```
--------------
| + 0 | -45 |
--------------
| +45 | +90 |
--------------
```

The primitive pattern shown above represents the fundamental polarization phase arrangement in a polarization sensor. The basic pattern would extend in the X and Y directions in a real polarization sensor containing millions of pixels. Notice that the phase of the origin (0, 0) pixel logically determines the phase of every other pixel. It is for this reason that the phase of this origin pixel is termed the polarization "phase" because it represents the reference point for the phase determination of all other pixels.

Enumerator

PolarPhase0	0 degrees polarization phase
PolarPhase45	45 degrees polarization phase
PolarPhase90	90 degrees polarization phase
PolarPhase135	135 (-45) degrees polarization phase

0.5.8 Thorlabs.TSI.TLCamera Namespace Reference

Classes

- class TLCameraSDK

0.5.9 Thorlabs.TSI.TLCameraInterfaces Namespace Reference

Classes

- class CameraConnectEventArgs
- class CameraDisconnectEventArgs
- class DiagnosticsMessageEventArgs
- class Frame
- interface ITLCamera
- interface ITLCameraSDK
- struct ROIAndBin

Enumerations

- enum CameraSensorType { CameraSensorType.Monochrome, CameraSensorType.Bayer, CameraSensorType.MonochromePo }
- enum ColorCorrectionMatrixOutputColorSpace { ColorCorrectionMatrixOutputColorSpace.CCIR_709 = 0 }
- enum CommunicationInterface { CommunicationInterface.GigE = 0, CommunicationInterface.CameraLink = 1, CommunicationInterface.USB = 2 }
- enum DataRate { DataRate.ReadoutSpeed20MHz = 0, DataRate.ReadoutSpeed40MHz = 1, DataRate.FPS30 = 2, DataRate.FPS50 = 3 }
- enum EEPStatus { EEPStatus.Disabled, EEPStatus.EnabledActive, EEPStatus.EnabledInactive, EEPStatus.EnabledBulb }
- enum OperationMode { OperationMode.SoftwareTriggered, OperationMode.HardwareTriggered, OperationMode.Bulb, OperationMode.Reserved1, OperationMode.Reserved2 }
- enum Taps { Taps.SingleTap = 0, Taps.DualTap = 1, Taps.QuadTap = 2 }
- enum TriggerPolarity { TriggerPolarity.ActiveHigh = 0, TriggerPolarity.ActiveLow = 1 }
- enum CameraUSBPortType { CameraUSBPortType.USB1_0, CameraUSBPortType.USB2_0, CameraUSBPortType.USB3_0, CameraUSBPortType.Unknown }

Functions

- delegate void OnImageFrameAvailableDelegate (ITLCamera sender, EventArgs eventArgs)
- delegate void OnDiagnosticsMessageDelegate (ITLCamera sender, DiagnosticsMessageEventArgs event↩ Args)
- delegate void OnCameraConnectDelegate (ITLCameraSDK sender, CameraConnectEventArgs eventArgs)
- delegate void OnCameraDisconnectDelegate (ITLCameraSDK sender, CameraDisconnectEventArgs eventArgs)

0.5.9.1 Enumeration Type Documentation

0.5.9.1.1 CameraSensorType

`enum Thorlabs.TSI.TLCameraInterfaces.CameraSensorType [strong]`

This describes the physical capabilities of the camera sensor.

Enumerator

Monochrome	Each pixel of the sensor indicates an intensity.
Bayer	The sensor has a bayer-patterned filter overlaying it, allowing the camera SDK to distinguish red, green, and blue values.
MonochromePolarized	The sensor has a polarization filter overlaying it, allowing the camera to capture polarization information from the incoming light.

0.5.9.1.2 CameraUSBPortType

`enum Thorlabs.TSI.TLCameraInterfaces.CameraUSBPortType [strong]`

The CameraUSBPortType enumeration defines the values the SDK uses for specifying the USB port type.

These values are returned by SDK API functions and callbacks based on the type of physical USB port that the device is connected to.

Enumerator

USB1_0	The device is connected to a USB 1.0/1.1 port (1.5 Mbits/sec or 12 Mbits/sec).
USB2_0	The device is connected to a USB 2.0 port (480 Mbits/sec).
USB3_0	The device is connected to a USB 3.0 port (5000 Mbits/sec).
Unknown	The device is connected to a UNKNOWN port.

0.5.9.1.3 ColorCorrectionMatrixOutputColorSpace

`enum Thorlabs.TSI.TLCameraInterfaces.ColorCorrectionMatrixOutputColorSpace [strong]`

Determines camera color correction matrix output color space value.

Enumerator

CCIR_709	ColorSpace - "CCIR 709".

0.5.9.1.4 CommunicationInterface

enum Thorlabs.TSI.TLCameraInterfaces.CommunicationInterface [strong]

Computer interface type.

Enumerator

GigE	The camera uses the GigE Vision (GigE) interface standard.
CameraLink	The camera uses the CameraLink serial-communication-protocol standard.
USB	The camera uses a USB interface.

0.5.9.1.5 DataRate

enum Thorlabs.TSI.TLCameraInterfaces.DataRate [strong]

The camera-sensor-level data readout rate. Use GetIsDataRateSupported to see if a particular data rate is supported.

Enumerator

ReadoutSpeed20MHz	For scientific-CCD cameras, the 20MHz sensor clock rate.
ReadoutSpeed40MHz	For scientific-CCD cameras, the 40MHz sensor clock rate.
FPS30	For compact-scientific cameras doing full-frame exposures, 30 frames per second (FPS).
FPS50	For compact-scientific cameras doing full-frame exposures, 50 frames per second (FPS).

0.5.9.1.6 EEPStatus

enum Thorlabs.TSI.TLCameraInterfaces.EEPStatus [strong]

Equal Exposure Pulse (EEP) mode is an LVTTL-level signal that is active during the time when all rows have been reset during rolling reset, and the end of the exposure time (and the beginning of rolling readout). The signal can be used to control an external light source that will be on only during the equal exposure period, providing the same amount of exposure for all pixels in the ROI.

When EEP mode is disabled, the status will always be EEPStatus.Off.
EEP mode can be enabled, but, depending on the exposure value, active or inactive.
If EEP is enabled in bulb mode, it will always give a status of Bulb.

Enumerator

Disabled	Equal Exposure Pulse (EEP) mode is disabled.
EnabledActive	Equal Exposure Pulse (EEP) mode is enabled and currently active.
EnabledInactive	Equal Exposure Pulse (EEP) is enabled, but due to an unsupported exposure value, currently inactive.
EnabledBulb	Equal Exposure Pulse (EEP) is enabled in bulb mode.

0.5.9.1.7 OperationMode

enum Thorlabs.TSI.TLCameraInterfaces.OperationMode [strong]

Thorlabs scientific cameras can be software- or hardward-triggered.

To run continuous-video mode, set FramesPerTrigger_zeroForUnlimited to zero and OperationMode to Software↩ Triggered.

To issue individual software triggers, set FramesPerTrigger_zeroForUnlimited to a number greater than zero and OperationMode to SoftwareTriggered.

To trigger frames using the hardware trigger input, set OperationMode mode to HardwareTriggered. In this mode, the Exposure_us property is used to determine the length of the exposure.

To trigger frames using the hardware trigger input and to determine the exposure length with that signal, set OperationMode to Bulb.

Enumerator

SoftwareTriggered	Use software operation mode to generate one or more frames per trigger or to run continuous video mode.
HardwareTriggered	Use hardware triggering to generate one or more frames per trigger by issuing hardware signals.
Bulb	Use bulb-mode triggering to generate one or more frames per trigger by issuing hardware signals. Please refer to the camera manual for signaling details.
Reserved1	Reserved for internal use.
Reserved2	Reserved for internal use.

0.5.9.1.8 Taps

enum Thorlabs.TSI.TLCameraInterfaces.Taps [strong]

Scientific CCD cameras support one or more taps.

After exposure is complete, a CCD pixel array holds the charges corresponding to the amount of light collected at each pixel location. The data is then read out through 1, 2, or 4 channels at a time.

Enumerator

SingleTap	Charges are read out through a single analog-to-digital converter.
DualTap	Charges are read out through two analog-to-digital converters.
QuadTap	Charges are read out through four analog-to-digital converters.

0.5.9.1.9 TriggerPolarity

enum Thorlabs.TSI.TLCameraInterfaces.TriggerPolarity [strong]

Determines when triggered-image exposure begins.

Enumerator

ActiveHigh	Exposure will begin on the rising edge of the trigger-input signal.
ActiveLow	Exposure will begin on the falling edge of the trigger-input signal.

0.5.9.2 Function Documentation

0.5.9.2.1 OnCameraConnectDelegate()

```
delegate void Thorlabs.TSI.TLCameraInterfaces.OnCameraConnectDelegate (
            ITLCameraSDK sender,
            CameraConnectEventArgs eventArgs )
```

The OnCameraConnect event is invoked when a USB camera is plugged in while the application is running.

Registering for the event is optional, but can be useful to refresh the list of attached cameras.

Parameters

sender	The instance of the ITLCameraSDK sending the event. (Normally, an application will have only a single instance of ITLCameraSDK, so this sender may be safely ignored.)
eventArgs	Serial number and USB bus speed information.

0.5.9.2.2 OnCameraDisconnectDelegate()

```
delegate void Thorlabs.TSI.TLCameraInterfaces.OnCameraDisconnectDelegate (
            ITLCameraSDK sender,
            CameraDisconnectEventArgs eventArgs )
```

The OnCameraConnect event is invoked when a USB camera is plugged in while the application is running.

Registering for the event is optional, but can be useful to refresh the list of attached cameras.

Parameters

sender	The instance of the ITLCameraSDK sending the event. (Normally, an application will have only a single instance of ITLCameraSDK, so this sender may be safely ignored.)
eventArgs	The event args include the serial number of the camera that was disconnected.

0.5.9.2.3 OnDiagnosticsMessageDelegate()

```
delegate void Thorlabs.TSI.TLCameraInterfaces.OnDiagnosticsMessageDelegate (
```

```
            ITLCamera sender,
            DiagnosticsMessageEventArgs eventArgs )
```

When possible, ITLCamera properties and methods throw exceptions. However, during image acquisition, error information is available only by registering for the OnDiagnosticsMessage event. The event usually arrives on a worker thread.

The ITLCameraSDK.OpenCamera(string serialNumber, bool isDiagnosticsMessagesEnabled) method takes a serial number and a bool. Passing true to the bool parameter enables verbose diagnostics information such as reporting every camera parameter that is set.

Parameters

sender	The instance of ITLCamera sending the event.
eventArgs	The event arguments include a user friendly error message and sometimes a more detailed diagnostics message.

0.5.9.2.4 OnImageFrameAvailableDelegate()

```
delegate void Thorlabs.TSI.TLCameraInterfaces.OnImageFrameAvailableDelegate (
            ITLCamera sender,
            EventArgs eventArgs )
```

ITLCamera offers two ways to get image data from the camera:

1. Poll for data using GetPendingFrameOrNull()

2. Register for the OnImageFrameAvailable event. The event will arrive on a worker thread. In the event handler, always call GetPendingFrameOrNull() one and only one time to get the data.

   For more details, see the Getting Started section in the help file.

Parameters

sender	The instance of ITLCamera sending the event.
eventArgs	The event arguments do not provide the image data. To get the image data, call GetPendingFrameOrNull().

0.6 Class Documentation

0.6.1 Thorlabs.TSI.TLCameraInterfaces.CameraConnectEventArgs Class Reference

Inherits EventArgs.

Public Member Functions

- CameraConnectEventArgs (string serialNumber, CameraUSBPortType portType)

Properties

- string SerialNumber [get]
- CameraUSBPortType USBPortType [get]

0.6.1.1 Detailed Description

See also

OnCameraConnectDelegate, ITLCameraSDK.OnCameraConnect

0.6.1.2 Constructor & Destructor Documentation

0.6.1.2.1 CameraConnectEventArgs()

```
Thorlabs.TSI.TLCameraInterfaces.CameraConnectEventArgs.CameraConnectEventArgs (
            string serialNumber,
            CameraUSBPortType portType ) [inline]
```

This constructor is for internal use only.

0.6.1.3 Property Documentation

0.6.1.3.1 SerialNumber

```
string Thorlabs.TSI.TLCameraInterfaces.CameraConnectEventArgs.SerialNumber [get]
```

The serial number of a camera that is discovered while the application is running.

0.6.1.3.2 USBPortType

```
CameraUSBPortType Thorlabs.TSI.TLCameraInterfaces.CameraConnectEventArgs.USBPortType [get]
```

This indicates which usb-port into which the camera was connected.

0.6.2 Thorlabs.TSI.TLCameraInterfaces.CameraDisconnectEventArgs Class Reference

Inherits EventArgs.

Public Member Functions

- CameraDisconnectEventArgs (string serialNumber)

Properties

- string SerialNumber [get]

0.6.2.1 Detailed Description

0.6.2.2 Constructor & Destructor Documentation

0.6.2.2.1 CameraDisconnectEventArgs()

```
Thorlabs.TSI.TLCameraInterfaces.CameraDisconnectEventArgs.CameraDisconnectEventArgs (
            string serialNumber )
```

This constructor is for internal use only.

Parameters

serialNumber	

0.6.2.3 Property Documentation

0.6.2.3.1 SerialNumber

```
string Thorlabs.TSI.TLCameraInterfaces.CameraDisconnectEventArgs.SerialNumber [get]
```

The serial number of the camera that was disconnected.

0.6.3 Thorlabs.TSI.ColorProcessor.ColorProcessor Class Reference

Inherits DisposableBase, and Thorlabs.TSI.ColorInterfaces.IColorProcessor.

Public Member Functions

- override unsafe void Dispose ()
- unsafe void Transform48To48 (ushort[ ] threeChannelImageData, ColorFormat inputColorFormat, ushort blue_output_min_value, ushort blue_output_max_value, ushort green_output_min_value, ushort green_↩ output_max_value, ushort red_output_min_value, ushort red_output_max_value, int blue_shift_distance, int green_shift_distance, int red_shift_distance, ushort[ ] outputData, ColorFormat outputColorFormat)

- unsafe void Transform48To32 (ushort[ ] threeChannelImageData, ColorFormat inputColorFormat, ushort blue_output_min_value, ushort blue_output_max_value, ushort green_output_min_value, ushort green_output_max_value, ushort red_output_min_value, ushort red_output_max_value, int blue_shift_distance, int green_shift_distance, int red_shift_distance, byte[ ] outputData, ColorFormat outputColorFormat)
- unsafe void Transform48To24 (ushort[ ] threeChannelImageData, byte[ ] outputData, ColorFormat inputColorFormat, ColorFormat outputColorFormat, ushort blue_output_min_value_before_shift, ushort blue_output_max_value_before_shift, ushort green_output_min_value_before_shift, ushort green_output_max_value_before_shift, ushort red_output_min_value_before_shift, ushort red_output_max_value_before_shift, int blue_shift_distance, int green_shift_distance, int red_shift_distance)
- void ClearColorTransformMatrices ()
- IList< ushort > GetBlueInputTransformLookupTable ()
- IList< ushort > GetBlueOutputTransformLookupTable ()
- float [ ] GetColorTransformMatrix (int index)
- IList< ushort > GetGreenInputTransformLookupTable ()
- IList< ushort > GetGreenOutputTransformLookupTable ()
- IList< ushort > GetRedInputTransformLookupTable ()
- IList< ushort > GetRedOutputTransformLookupTable ()
- void InsertColorTransformMatrix (int index, float[ ] colorTransformMatrix)
- void RemoveColorTransformMatrix (int index)
- void SetBlueInputTransformLookupTable (IList< ushort > blueInputLUTOrNull)
- void SetBlueOutputTransformLookupTable (IList< ushort > blueOutputLUTOrNull)
- void SetGreenInputTransformLookupTable (IList< ushort > greenInputLUTOrNull)
- void SetGreenOutputTransformLookupTable (IList< ushort > greenOutputLUTOrNull)
- void SetRedInputTransformLookupTable (IList< ushort > redInputLUTOrNull)
- void SetRedOutputTransformLookupTable (IList< ushort > redOutputLUTOrNull)
- override string ToString ()

Properties

- int NumberOfColorTransformMatrices `[get]`

0.6.3.1 Member Function Documentation

0.6.3.1.1 ClearColorTransformMatrices()

```
void Thorlabs.TSI.ColorProcessor.ColorProcessor.ClearColorTransformMatrices ( )  [inline]
```

Remove all of the color-matrix transforms from the color pipeline.

RemoveColorTransformMatrix

Implements Thorlabs.TSI.ColorInterfaces.IColorProcessor.

0.6.3.1.2 Dispose()

```
override unsafe void Thorlabs.TSI.ColorProcessor.ColorProcessor.Dispose ( )  [inline]
```

0.6.3.1.3 GetBlueInputTransformLookupTable()

```
IList<ushort> Thorlabs.TSI.ColorProcessor.ColorProcessor.GetBlueInputTransformLookupTable ( )
```

Get a shared reference to the blue input transform lookup table.

Implements Thorlabs.TSI.ColorInterfaces.IColorProcessor.

0.6.3.1.4 GetBlueOutputTransformLookupTable()

```
IList<ushort> Thorlabs.TSI.ColorProcessor.ColorProcessor.GetBlueOutputTransformLookupTable ( )
```

Get a shared reference to the blue output transform lookup table.

Implements Thorlabs.TSI.ColorInterfaces.IColorProcessor.

0.6.3.1.5 GetColorTransformMatrix()

```
float [] Thorlabs.TSI.ColorProcessor.ColorProcessor.GetColorTransformMatrix (
            int index )  [inline]
```

Get the color-transform matrix previously added at the given position.

Parameters

index	Zero-based index at which to retrieve the matrix.

Returns

A one-dimensional array of floats ordered as follows:
0 1 2
3 4 5
6 7 8

Implements Thorlabs.TSI.ColorInterfaces.IColorProcessor.

0.6.3.1.6 GetGreenInputTransformLookupTable()

```
IList<ushort> Thorlabs.TSI.ColorProcessor.ColorProcessor.GetGreenInputTransformLookupTable ( )
```

Get a shared reference to the green input transform lookup table.

Implements Thorlabs.TSI.ColorInterfaces.IColorProcessor.

0.6.3.1.7 GetGreenOutputTransformLookupTable()

```
IList<ushort> Thorlabs.TSI.ColorProcessor.ColorProcessor.GetGreenOutputTransformLookupTable (
)
```

Get a shared reference to the green output transform lookup table.

Implements Thorlabs.TSI.ColorInterfaces.IColorProcessor.

0.6.3.1.8 GetRedInputTransformLookupTable()

```
IList<ushort> Thorlabs.TSI.ColorProcessor.ColorProcessor.GetRedInputTransformLookupTable ( )
```

Get a shared reference to the red input transform lookup table.

Implements Thorlabs.TSI.ColorInterfaces.IColorProcessor.

0.6.3.1.9 GetRedOutputTransformLookupTable()

```
IList<ushort> Thorlabs.TSI.ColorProcessor.ColorProcessor.GetRedOutputTransformLookupTable ( )
```

Get a shared reference to the red output transform lookup table.

Implements Thorlabs.TSI.ColorInterfaces.IColorProcessor.

0.6.3.1.10 InsertColorTransformMatrix()

```
void Thorlabs.TSI.ColorProcessor.ColorProcessor.InsertColorTransformMatrix (
            int index,
            float [] colorTransformMatrix )  [inline]
```

Add a three-by-three color-transform matrix in the specified order. These matrices are multiplied together, and the order is significant.

A common use case is to apply the standard white- balance matrix retrieved with ITLCamera.GetDefaultWhite↩ BalanceMatrix() followed by the color-correction matrix retrieved with ITLCamera.GetCameraColorCorrection↩ Matrix().

Parameters

index	Zero-based index into which position the matrix will be located.
colorTransformMatrix	A one-dimensional array of floats ordered as follows: 0 1 2 3 4 5 6 7 8

Implements Thorlabs.TSI.ColorInterfaces.IColorProcessor.

0.6.3.1.11 RemoveColorTransformMatrix()

```
void Thorlabs.TSI.ColorProcessor.ColorProcessor.RemoveColorTransformMatrix (
            int index )  [inline]
```

Matrices can be added or removed at any time. They are only applied when calling one of the Transform methods.

Parameters

index	Zero-based index at which the matrix will be removed.

Implements Thorlabs.TSI.ColorInterfaces.IColorProcessor.

0.6.3.1.12 SetBlueInputTransformLookupTable()

```
void Thorlabs.TSI.ColorProcessor.ColorProcessor.SetBlueInputTransformLookupTable (
            IList< ushort > blueInputLUTOrNull )  [inline]
```

Provide a shared reference to a blue input transform lookup table.

Parameters

blueInputLUTOrNull	A lookup table of size = 2^n where n is the bit depth of the image. If null, no lookup table will be applied in the transform methods.

Implements Thorlabs.TSI.ColorInterfaces.IColorProcessor.

0.6.3.1.13 SetBlueOutputTransformLookupTable()

```
void Thorlabs.TSI.ColorProcessor.ColorProcessor.SetBlueOutputTransformLookupTable (
            IList< ushort > blueOutputLUTOrNull )  [inline]
```

Provide a shared reference to a blue output transform lookup table.

Parameters

blueOutputLUTOrNull	A lookup table of size = 2^n where n is the bit depth of the image. If null, no lookup table will be applied in the transform methods.

Implements Thorlabs.TSI.ColorInterfaces.IColorProcessor.

0.6.3.1.14 SetGreenInputTransformLookupTable()

```
void Thorlabs.TSI.ColorProcessor.ColorProcessor.SetGreenInputTransformLookupTable (
            IList< ushort > greenInputLUTOrNull )  [inline]
```

Provide a shared reference to a green input transform lookup table.

Parameters

greenInputLUTOrNull	A lookup table of size = 2^n where n is the bit depth of the image. If null, no lookup table will be applied in the transform methods.

Implements Thorlabs.TSI.ColorInterfaces.IColorProcessor.

0.6.3.1.15 SetGreenOutputTransformLookupTable()

```
void Thorlabs.TSI.ColorProcessor.ColorProcessor.SetGreenOutputTransformLookupTable (
            IList< ushort > greenOutputLUTOrNull )  [inline]
```

Provide a shared reference to a green output transform lookup table.

Parameters

greenOutputLUTOrNull	A lookup table of size = 2^n where n is the bit depth of the image. If null, no lookup table will be applied in the transform methods.

Implements Thorlabs.TSI.ColorInterfaces.IColorProcessor.

0.6.3.1.16 SetRedInputTransformLookupTable()

```
void Thorlabs.TSI.ColorProcessor.ColorProcessor.SetRedInputTransformLookupTable (
            IList< ushort > redInputLUTOrNull )  [inline]
```

Provide a shared reference to a red input transform lookup table.

Parameters

redInputLUTOrNull	A lookup table of size = 2^n where n is the bit depth of the image. If null, no lookup table will be applied in the transform methods.

Implements Thorlabs.TSI.ColorInterfaces.IColorProcessor.

0.6.3.1.17 SetRedOutputTransformLookupTable()

```
void Thorlabs.TSI.ColorProcessor.ColorProcessor.SetRedOutputTransformLookupTable (
            IList< ushort > redOutputLUTOrNull )  [inline]
```

Provide a shared reference to a red output transform lookup table.

Parameters

redOutputLUTOrNull	A lookup table of size = 2^n where n is the bit depth of the image. If null, no lookup table will be applied in the transform methods.

Implements Thorlabs.TSI.ColorInterfaces.IColorProcessor.

0.6.3.1.18 ToString()

```
override string Thorlabs.TSI.ColorProcessor.ColorProcessor.ToString ( )
```

0.6.3.1.19 Transform48To24()

```
unsafe void Thorlabs.TSI.ColorProcessor.ColorProcessor.Transform48To24 (
            ushort [] threeChannelImageData,
            byte [] outputData,
            ColorFormat inputColorFormat,
            ColorFormat outputColorFormat,
            ushort blue_output_min_value_before_shift,
            ushort blue_output_max_value_before_shift,
            ushort green_output_min_value_before_shift,
            ushort green_output_max_value_before_shift,
            ushort red_output_min_value_before_shift,
            ushort red_output_max_value_before_shift,
            int blue_shift_distance,
            int green_shift_distance,
            int red_shift_distance )  [inline]
```

Transforms a 48-bit-per-pixel image (two bytes per color channel) in the given input color format to a 32-bit-per-pixel image (one byte per color channel plus a byte of padding) in the given output color format.

First, it applies the input lookup tables (if provided). Then, it applies the matrix transforms in order. Then, it applies the output lookup tables. Finally, it right or left shifts the resulting pixel values by the given number of pixels (usually zero).

For an in-depth analysis of the theory behind IColorProcessor, please see the help file targeted at C and C++ developers called "thorlabs_tsi_color_processing_API.chm"

void Transform48To32(ushort[ ] threeChannelImageData, ColorFormat inputColorFormat, ushort blue_output_min_value, ushort blue_

Transforms a 48-bit-per-pixel image (two bytes per color channel) in the given input color format to a 24-bit-per-pixel image (one byte per color channel) in the given output color format.

First, it applies the input lookup tables (if provided). Then, it applies the matrix transforms in order. Then, it applies the output lookup tables. Finally, it right or left shifts the resulting pixel values by the given number of pixels (usually zero).

For an in-depth analysis of the theory behind IColorProcessor, please see the help file targeted at C and C++ developers called "thorlabs_tsi_color_processing_API.chm"

Implements Thorlabs.TSI.ColorInterfaces.IColorProcessor.

0.6.3.1.20 Transform48To32()

```
unsafe void Thorlabs.TSI.ColorProcessor.ColorProcessor.Transform48To32 (
            ushort [] threeChannelImageData,
            ColorFormat inputColorFormat,
            ushort blue_output_min_value,
            ushort blue_output_max_value,
            ushort green_output_min_value,
            ushort green_output_max_value,
            ushort red_output_min_value,
            ushort red_output_max_value,
            int blue_shift_distance,
            int green_shift_distance,
            int red_shift_distance,
            byte [] outputData,
            ColorFormat outputColorFormat )  [inline]
```

0.6.3.1.21 Transform48To48()

```
unsafe void Thorlabs.TSI.ColorProcessor.ColorProcessor.Transform48To48 (
            ushort [] threeChannelImageData,
            ColorFormat inputColorFormat,
            ushort blue_output_min_value,
            ushort blue_output_max_value,
            ushort green_output_min_value,
            ushort green_output_max_value,
            ushort red_output_min_value,
            ushort red_output_max_value,
            int blue_shift_distance,
            int green_shift_distance,
            int red_shift_distance,
            ushort [] outputData,
            ColorFormat outputColorFormat )  [inline]
```

Transforms a 48-bit-per-pixel image (two bytes per color channel) in the given input color format to a 48-bit-per-pixel image in the given output color format.

First, it applies the input lookup tables (if provided). Then, it applies the matrix transforms in order. Then, it applies the output lookup tables. Finally, it right or left shifts the resulting pixel values by the given number of pixels (usually 0).

For an in-depth analysis of the theory behind IColorProcessor, please see the help file targeted at C and C++ developers called "thorlabs_tsi_color_processing_API.chm"

Implements Thorlabs.TSI.ColorInterfaces.IColorProcessor.

0.6.3.2 Property Documentation

0.6.3.2.1 NumberOfColorTransformMatrices

```
int Thorlabs.TSI.ColorProcessor.ColorProcessor.NumberOfColorTransformMatrices  [get]
```

0.6.4 Thorlabs.TSI.ColorProcessor.ColorProcessorSDK Class Reference

Inherits DisposableBase, and Thorlabs.TSI.ColorInterfaces.IColorProcessorSDK.

Public Member Functions

- ColorProcessorSDK ()
- override void Dispose ()
- IColorProcessor CreateColorProcessor ()
- IColorProcessor CreateLinearRGBColorProcessor (float[ ] whiteBalanceMatrix, float[ ] colorCorrectionMatrix, int bitDepth)
- IColorProcessor CreateStandardRGBColorProcessor (float[ ] whiteBalanceMatrix, float[ ] colorCorrection↩ Matrix, int bitDepth, int blackLevel=0)

0.6.4.1 Constructor & Destructor Documentation

0.6.4.1.1 ColorProcessorSDK()

```
Thorlabs.TSI.ColorProcessor.ColorProcessorSDK.ColorProcessorSDK ( )  [inline]
```

0.6.4.2 Member Function Documentation

0.6.4.2.1 CreateColorProcessor()

```
IColorProcessor Thorlabs.TSI.ColorProcessor.ColorProcessorSDK.CreateColorProcessor ( )
```

Creates an empty color processor. An empty color processor can be configured for a variety of specialized color-processing strategies when color spaces other than sRGB or LinearRGB are needed. sRGB and LinearRGB can be created with CreateStandardRGBColorProcessor or CreateLinearRGBColorProcessor, respectively.

Implements Thorlabs.TSI.ColorInterfaces.IColorProcessorSDK.

0.6.4.2.2 CreateLinearRGBColorProcessor()

```
IColorProcessor Thorlabs.TSI.ColorProcessor.ColorProcessorSDK.CreateLinearRGBColorProcessor (
            float [] whiteBalanceMatrix,
            float [] colorCorrectionMatrix,
            int bitDepth )  [inline]
```

Create a color processor that can transform three-channel data into a standard-RGB (sRGB) color space but without applying any non-linear conversions normally used for sRGB. This would be equivalent to sRGB with a gamma value of 1.

Implements Thorlabs.TSI.ColorInterfaces.IColorProcessorSDK.

0.6.4.2.3 CreateStandardRGBColorProcessor()

```
IColorProcessor Thorlabs.TSI.ColorProcessor.ColorProcessorSDK.CreateStandardRGBColorProcessor
(
            float [] whiteBalanceMatrix,
            float [] colorCorrectionMatrix,
            int bitDepth,
            int blackLevel = 0 )  [inline]
```

Create a color processor that can transform three-channel data into a standard RGB (sRGB) color space.

Implements Thorlabs.TSI.ColorInterfaces.IColorProcessorSDK.

0.6.4.2.4 Dispose()

```
override void Thorlabs.TSI.ColorProcessor.ColorProcessorSDK.Dispose ( )  [inline]
```

Unload the color-processing DLLs from memory. Be sure to dispose all ColorProcessor objects before disposing the ColorProcessorSDK.

0.6.5 Thorlabs.TSI.Demosaicker.Demosaicker Class Reference

Inherits DisposableBase, and Thorlabs.TSI.ColorInterfaces.IDemosaicker.

Public Member Functions

- Demosaicker ()
- override void Dispose ()
- void Demosaic (int width, int height, int xOrigin, int yOrigin, ColorFilterArrayPhase colorFilterArrayPhase, ColorFormat output_color_format, ColorSensorType colorSensorType, int bitDepth, ushort[ ] singleChannel↩ MosaicPatternInputData, ushort[ ] outputBuffer)

0.6.5.1 Detailed Description

See also

T:Thorlabs.TSI.Core.DisposableBase, T:Thorlabs.TSI.ColorInterfaces.IDemosaicker

0.6.5.2 Constructor & Destructor Documentation

0.6.5.2.1 Demosaicker()

```
Thorlabs.TSI.Demosaicker.Demosaicker.Demosaicker ( )  [inline]
```

0.6.5.3 Member Function Documentation

0.6.5.3.1 Demosaic()

```
void Thorlabs.TSI.Demosaicker.Demosaicker.Demosaic (
            int width,
            int height,
            int xOrigin,
            int yOrigin,
            ColorFilterArrayPhase colorFilterArrayPhase,
            ColorFormat output_color_format,
            ColorSensorType colorSensorType,
            int bitDepth,
            ushort [] singleChannelMosaicPatternInputData,
            ushort [] outputBuffer )  [inline]
```

This function takes an input buffer containing monochrome image data and writes a color image into the specified output buffer using a standard demosaic computation.

Implements Thorlabs.TSI.ColorInterfaces.IDemosaicker.

0.6.5.3.2 Dispose()

```
override void Thorlabs.TSI.Demosaicker.Demosaicker.Dispose ( )  [inline]
```

Unloads the demosaic dll from memory.

Exceptions

InvalidOperationException	

0.6.6 Thorlabs.TSI.TLCameraInterfaces.DiagnosticsMessageEventArgs Class Reference

Inherits EventArgs.

Public Member Functions

- DiagnosticsMessageEventArgs (string publicErrorMessageOrNull, string privateErrorMessageOrNull)

Properties

- string PrivateErrorMessageOrNull `[get]`
- string PublicErrorMessageOrNull `[get]`

0.6.6.1 Detailed Description

0.6.6.2 Constructor & Destructor Documentation

0.6.6.2.1 DiagnosticsMessageEventArgs()

```
Thorlabs.TSI.TLCameraInterfaces.DiagnosticsMessageEventArgs.DiagnosticsMessageEventArgs (
            string publicErrorMessageOrNull,
            string privateErrorMessageOrNull )  [inline]
```

The constructor is used internally.

0.6.6.3 Property Documentation

0.6.6.3.1 PrivateErrorMessageOrNull

```
string Thorlabs.TSI.TLCameraInterfaces.DiagnosticsMessageEventArgs.PrivateErrorMessageOrNull
[get]
```

A diagnostics message which may include additional, more detailed information that would normally be logged rather than displayed to the user.

0.6.6.3.2 PublicErrorMessageOrNull

```
string Thorlabs.TSI.TLCameraInterfaces.DiagnosticsMessageEventArgs.PublicErrorMessageOrNull
[get]
```

A user-friendly error message that can be displayed to the user in an error dialog.

0.6.7 Thorlabs.TSI.TLCameraInterfaces.Frame Class Reference

Public Member Functions

- Frame (IImageData imageData, uint frameNumber, uint? exposureTime_us_orNull, uint? gainOrNull, uint? blackLevelOrNull, ulong? timeStampRelative_ns_OrNull=null)

Properties

- ulong? TimeStampRelative_ns_OrNull [get]
- uint? BlackLevelOrNull [get]
- uint? ExposureTime_us_orNull [get]
- uint FrameNumber [get]
- uint? GainOrNull [get]
- IImageData ImageData [get]

0.6.7.1 Detailed Description

0.6.7.2 Constructor & Destructor Documentation

0.6.7.2.1 Frame()

```
Thorlabs.TSI.TLCameraInterfaces.Frame.Frame (
            IImageData imageData,
            uint frameNumber,
            uint?  exposureTime_us_orNull,
            uint?  gainOrNull,
            uint?  blackLevelOrNull,
            ulong?  timeStampRelative_ns_OrNull = null )  [inline]
```

This constructor is normally called by the camera SDK..

Parameters

imageData	IImageData can be downcast to a derived interface, such as IImageDataUShort1D.
frameNumber	The frame number.
exposureTime_us_orNull	The exposure time may be null as some cameras do not have the metadata for each frame
gainOrNull	The gain may be null as some cameras do not have the metadata for each frame
blackLevelOrNull	The black level may be null as some cameras do not have the metadata for each frame
timeStampRelative_ns_OrNull	The time stamp may be null as some cameras do not have the metadata for each frame

0.6.7.3 Property Documentation

0.6.7.3.1 BlackLevelOrNull

```
uint?  Thorlabs.TSI.TLCameraInterfaces.Frame.BlackLevelOrNull  [get]
```

Gets the black level.

0.6.7.3.2 ExposureTime_us_orNull

```
uint?  Thorlabs.TSI.TLCameraInterfaces.Frame.ExposureTime_us_orNull  [get]
```

Gets the exposure time in us.

0.6.7.3.3 FrameNumber

```
uint Thorlabs.TSI.TLCameraInterfaces.Frame.FrameNumber  [get]
```

Gets the frame number.

0.6.7.3.4 GainOrNull

```
uint?  Thorlabs.TSI.TLCameraInterfaces.Frame.GainOrNull  [get]
```

Gets the gain.

0.6.7.3.5 ImageData

```
IImageData Thorlabs.TSI.TLCameraInterfaces.Frame.ImageData  [get]
```

Gets the one-dimensional array of ushort (16-bit unsigned whole number) values along with information for how to interpret that data.

0.6.7.3.6 TimeStampRelative_ns_OrNull

```
ulong?  Thorlabs.TSI.TLCameraInterfaces.Frame.TimeStampRelative_ns_OrNull  [get]
```

Gets the time stamp relative to camera plug in time in nanoseconds for the frame. The timestamp is found by taking the pixel clock count just after exposure is finished and dividing it by a model-specific base clock frequency. Get the time in between frames by subtracting their relative time stamps.

0.6.8 Thorlabs.TSI.ColorInterfaces.IColorProcessor Interface Reference

Inherits IDisposable.

Inherited by Thorlabs.TSI.ColorProcessor.ColorProcessor.

Public Member Functions

- void InsertColorTransformMatrix (int index, float[ ] colorTransformMatrix)
- void RemoveColorTransformMatrix (int index)
- float [ ] GetColorTransformMatrix (int index)
- void ClearColorTransformMatrices ()
- void Transform48To48 (ushort[ ] threeChannelImageData, ColorFormat inputColorFormat, ushort blue_output_min_value, ushort blue_output_max_value, ushort green_output_min_value, ushort green_output_max_value, ushort red_output_min_value, ushort red_output_max_value, int blue_shift_distance, int green_shift_distance, int red_shift_distance, ushort[ ] outputData, ColorFormat outputColorFormat)
- void Transform48To24 (ushort[ ] threeChannelImageData, byte[ ] outputData, ColorFormat inputColorFormat, ColorFormat outputColorFormat, ushort blue_output_min_value_before_shift, ushort blue_output_max_value_before_shift, ushort green_output_min_value_before_shift, ushort green_output_max_value_before_shift, ushort red_output_min_value_before_shift, ushort red_output_max_value_before_shift, int blue_shift_distance, int green_shift_distance, int red_shift_distance)
- IList< ushort > GetRedInputTransformLookupTable ()
- IList< ushort > GetGreenInputTransformLookupTable ()
- IList< ushort > GetBlueInputTransformLookupTable ()
- IList< ushort > GetRedOutputTransformLookupTable ()
- IList< ushort > GetGreenOutputTransformLookupTable ()
- IList< ushort > GetBlueOutputTransformLookupTable ()
- void SetRedInputTransformLookupTable (IList< ushort > redInputLUTOrNull)
- void SetGreenInputTransformLookupTable (IList< ushort > greenInputLUTOrNull)
- void SetBlueInputTransformLookupTable (IList< ushort > blueInputLUTOrNull)
- void SetRedOutputTransformLookupTable (IList< ushort > redOutputLUTOrNull)
- void SetGreenOutputTransformLookupTable (IList< ushort > greenOutputLUTOrNull)
- void SetBlueOutputTransformLookupTable (IList< ushort > blueOutputLUTOrNull)

Properties

- int NumberOfColorTransformMatrices `[get]`

0.6.8.1 Detailed Description

The color processor takes three-channel color data and transforms it to a different color space.

For an in-depth analysis of the theory behind IColorProcessor, please see the help file targeted at C and C++ developers called "thorlabs_tsi_color_processing_API.chm"

First, it applies any input-transform lookup tables (LUTs) that are provided.

Second, it multiplies together any sequence of matrices and applies them to the pixel data.

Third, it applies any output-transform lookup tables (LUTs) that are provided.

See ITLCamera.GetDefaultWhiteBalanceMatrix() and ITLCamera.GetCameraColorCorrectionMatrix().

0.6.8.2 Member Function Documentation

0.6.8.2.1 ClearColorTransformMatrices()

```
void Thorlabs.TSI.ColorInterfaces.IColorProcessor.ClearColorTransformMatrices ( )
```

Remove all of the color-matrix transforms from the color pipeline.

RemoveColorTransformMatrix

Implemented in Thorlabs.TSI.ColorProcessor.ColorProcessor.

0.6.8.2.2 GetBlueInputTransformLookupTable()

```
IList<ushort> Thorlabs.TSI.ColorInterfaces.IColorProcessor.GetBlueInputTransformLookupTable (
)
```

Get a shared reference to the blue input transform lookup table.

Implemented in Thorlabs.TSI.ColorProcessor.ColorProcessor.

0.6.8.2.3 GetBlueOutputTransformLookupTable()

```
IList<ushort> Thorlabs.TSI.ColorInterfaces.IColorProcessor.GetBlueOutputTransformLookupTable (
)
```

Get a shared reference to the blue output transform lookup table.

Implemented in Thorlabs.TSI.ColorProcessor.ColorProcessor.

0.6.8.2.4 GetColorTransformMatrix()

```
float [] Thorlabs.TSI.ColorInterfaces.IColorProcessor.GetColorTransformMatrix (
            int index )
```

Get the color-transform matrix previously added at the given position.

Parameters

index	Zero-based index at which to retrieve the matrix.

Returns

A one-dimensional array of floats ordered as follows:
0 1 2
3 4 5
6 7 8

Implemented in Thorlabs.TSI.ColorProcessor.ColorProcessor.

0.6.8.2.5 GetGreenInputTransformLookupTable()

```
IList<ushort> Thorlabs.TSI.ColorInterfaces.IColorProcessor.GetGreenInputTransformLookupTable (
)
```

Get a shared reference to the green input transform lookup table.

Implemented in Thorlabs.TSI.ColorProcessor.ColorProcessor.

0.6.8.2.6 GetGreenOutputTransformLookupTable()

```
IList<ushort> Thorlabs.TSI.ColorInterfaces.IColorProcessor.GetGreenOutputTransformLookupTable
( )
```

Get a shared reference to the green output transform lookup table.

Implemented in Thorlabs.TSI.ColorProcessor.ColorProcessor.

0.6.8.2.7 GetRedInputTransformLookupTable()

```
IList<ushort> Thorlabs.TSI.ColorInterfaces.IColorProcessor.GetRedInputTransformLookupTable ( )
```

Get a shared reference to the red input transform lookup table.

Implemented in Thorlabs.TSI.ColorProcessor.ColorProcessor.

0.6.8.2.8 GetRedOutputTransformLookupTable()

```
IList<ushort> Thorlabs.TSI.ColorInterfaces.IColorProcessor.GetRedOutputTransformLookupTable (
)
```

Get a shared reference to the red output transform lookup table.

Implemented in Thorlabs.TSI.ColorProcessor.ColorProcessor.

0.6.8.2.9 InsertColorTransformMatrix()

```
void Thorlabs.TSI.ColorInterfaces.IColorProcessor.InsertColorTransformMatrix (
            int index,
            float [] colorTransformMatrix )
```

Add a three-by-three color-transform matrix in the specified order. These matrices are multiplied together, and the order is significant.

A common use case is to apply the standard white- balance matrix retrieved with ITLCamera.GetDefaultWhite↩ BalanceMatrix() followed by the color-correction matrix retrieved with ITLCamera.GetCameraColorCorrection↩ Matrix().

Parameters

index	Zero-based index into which position the matrix will be located.
colorTransformMatrix	A one-dimensional array of floats ordered as follows: 0 1 2 3 4 5 6 7 8

Implemented in Thorlabs.TSI.ColorProcessor.ColorProcessor.

0.6.8.2.10 RemoveColorTransformMatrix()

```
void Thorlabs.TSI.ColorInterfaces.IColorProcessor.RemoveColorTransformMatrix (
            int index )
```

Matrices can be added or removed at any time. They are only applied when calling one of the Transform methods.

Parameters

index	Zero-based index at which the matrix will be removed.

Implemented in Thorlabs.TSI.ColorProcessor.ColorProcessor.

0.6.8.2.11 SetBlueInputTransformLookupTable()

```
void Thorlabs.TSI.ColorInterfaces.IColorProcessor.SetBlueInputTransformLookupTable (
            IList< ushort > blueInputLUTOrNull )
```

Provide a shared reference to a blue input transform lookup table.

Parameters

blueInputLUTOrNull	A lookup table of size = 2^n where n is the bit depth of the image. If null, no lookup table will be applied in the transform methods.

Implemented in Thorlabs.TSI.ColorProcessor.ColorProcessor.

0.6.8.2.12 SetBlueOutputTransformLookupTable()

```
void Thorlabs.TSI.ColorInterfaces.IColorProcessor.SetBlueOutputTransformLookupTable (
            IList< ushort > blueOutputLUTOrNull )
```

Provide a shared reference to a blue output transform lookup table.

Parameters

blueOutputLUTOrNull	A lookup table of size = 2^n where n is the bit depth of the image. If null, no lookup table will be applied in the transform methods.

Implemented in Thorlabs.TSI.ColorProcessor.ColorProcessor.

0.6.8.2.13 SetGreenInputTransformLookupTable()

```
void Thorlabs.TSI.ColorInterfaces.IColorProcessor.SetGreenInputTransformLookupTable (
            IList< ushort > greenInputLUTOrNull )
```

Provide a shared reference to a green input transform lookup table.

Parameters

greenInputLUTOrNull	A lookup table of size = 2^n where n is the bit depth of the image. If null, no lookup table will be applied in the transform methods.

Implemented in Thorlabs.TSI.ColorProcessor.ColorProcessor.

0.6.8.2.14 SetGreenOutputTransformLookupTable()

```
void Thorlabs.TSI.ColorInterfaces.IColorProcessor.SetGreenOutputTransformLookupTable (
            IList< ushort > greenOutputLUTOrNull )
```

Provide a shared reference to a green output transform lookup table.

Parameters

greenOutputLUTOrNull	A lookup table of size = 2^n where n is the bit depth of the image. If null, no lookup table will be applied in the transform methods.

Implemented in Thorlabs.TSI.ColorProcessor.ColorProcessor.

0.6.8.2.15 SetRedInputTransformLookupTable()

```
void Thorlabs.TSI.ColorInterfaces.IColorProcessor.SetRedInputTransformLookupTable (
            IList< ushort > redInputLUTOrNull )
```

Provide a shared reference to a red input transform lookup table.

Parameters

redInputLUTOrNull	A lookup table of size = 2^n where n is the bit depth of the image. If null, no lookup table will be applied in the transform methods.

Implemented in Thorlabs.TSI.ColorProcessor.ColorProcessor.

0.6.8.2.16 SetRedOutputTransformLookupTable()

```
void Thorlabs.TSI.ColorInterfaces.IColorProcessor.SetRedOutputTransformLookupTable (
            IList< ushort > redOutputLUTOrNull )
```

Provide a shared reference to a red output transform lookup table.

Parameters

redOutputLUTOrNull	A lookup table of size = 2^n where n is the bit depth of the image. If null, no lookup table will be applied in the transform methods.

Implemented in Thorlabs.TSI.ColorProcessor.ColorProcessor.

0.6.8.2.17 Transform48To24()

```
void Thorlabs.TSI.ColorInterfaces.IColorProcessor.Transform48To24 (
            ushort [] threeChannelImageData,
            byte [] outputData,
            ColorFormat inputColorFormat,
            ColorFormat outputColorFormat,
            ushort blue_output_min_value_before_shift,
            ushort blue_output_max_value_before_shift,
            ushort green_output_min_value_before_shift,
            ushort green_output_max_value_before_shift,
            ushort red_output_min_value_before_shift,
            ushort red_output_max_value_before_shift,
            int blue_shift_distance,
            int green_shift_distance,
            int red_shift_distance )
```

Transforms a 48-bit-per-pixel image (two bytes per color channel) in the given input color format to a 32-bit-per-pixel image (one byte per color channel plus a byte of padding) in the given output color format.

First, it applies the input lookup tables (if provided). Then, it applies the matrix transforms in order. Then, it applies the output lookup tables. Finally, it right or left shifts the resulting pixel values by the given number of pixels (usually zero).

For an in-depth analysis of the theory behind IColorProcessor, please see the help file targeted at C and C++ developers called "thorlabs_tsi_color_processing_API.chm"

void Transform48To32(ushort[] threeChannelImageData, ColorFormat inputColorFormat, ushort blue_output↩ _min_value, ushort blue_output_max_value, ushort green_output_min_value, ushort green_output_max_value,

ushort red_output_min_value, ushort red_output_max_value, int blue_shift_distance, int green_shift_distance, int red_shift_distance, byte[ ] outputData, ColorFormat outputColorFormat);

Transforms a 48-bit-per-pixel image (two bytes per color channel) in the given input color format to a 24-bit-per-pixel image (one byte per color channel) in the given output color format.

First, it applies the input lookup tables (if provided). Then, it applies the matrix transforms in order. Then, it applies the output lookup tables. Finally, it right or left shifts the resulting pixel values by the given number of pixels (usually zero).

For an in-depth analysis of the theory behind IColorProcessor, please see the help file targeted at C and C++ developers called "thorlabs_tsi_color_processing_API.chm"

Implemented in Thorlabs.TSI.ColorProcessor.ColorProcessor.

0.6.8.2.18 Transform48To48()

```
void Thorlabs.TSI.ColorInterfaces.IColorProcessor.Transform48To48 (
            ushort [ ] threeChannelImageData,
            ColorFormat inputColorFormat,
            ushort blue_output_min_value,
            ushort blue_output_max_value,
            ushort green_output_min_value,
            ushort green_output_max_value,
            ushort red_output_min_value,
            ushort red_output_max_value,
            int blue_shift_distance,
            int green_shift_distance,
            int red_shift_distance,
            ushort [ ] outputData,
            ColorFormat outputColorFormat )
```

Transforms a 48-bit-per-pixel image (two bytes per color channel) in the given input color format to a 48-bit-per-pixel image in the given output color format.

First, it applies the input lookup tables (if provided). Then, it applies the matrix transforms in order. Then, it applies the output lookup tables. Finally, it right or left shifts the resulting pixel values by the given number of pixels (usually 0).

For an in-depth analysis of the theory behind IColorProcessor, please see the help file targeted at C and C++ developers called "thorlabs_tsi_color_processing_API.chm"

Implemented in Thorlabs.TSI.ColorProcessor.ColorProcessor.

0.6.8.3 Property Documentation

0.6.8.3.1 NumberOfColorTransformMatrices

```
int Thorlabs.TSI.ColorInterfaces.IColorProcessor.NumberOfColorTransformMatrices  [get]
```

Returns the number of matrices added to the IColorProcessor object.

InsertColorTransformMatrix RemoveColorTransformMatrix

0.6.9 Thorlabs.TSI.ColorInterfaces.IColorProcessorSDK Interface Reference

Inherits IDisposable.

Inherited by Thorlabs.TSI.ColorProcessor.ColorProcessorSDK.

Public Member Functions

- IColorProcessor CreateColorProcessor ()
- IColorProcessor CreateLinearRGBColorProcessor (float[ ] whiteBalanceMatrix, float[ ] colorCorrectionMatrix, int bitDepth)
- IColorProcessor CreateStandardRGBColorProcessor (float[ ] whiteBalanceMatrix, float[ ] colorCorrection↩ Matrix, int bitDepth, int blackLevel=0)

0.6.9.1 Detailed Description

The color processor SDK loads DLLs into memory and provides three functions to create color-processing objects. It is important to dispose the color-processing objects when they are no longer needed and then disposing the color-processing SDK.

A color-processing object takes three-channel color data and transforms it to a different color space.

Be sure to dispose all ColorProcessor objects and then dispose the ColorProcessorSDK before exiting the application.

See also

IDemosaicker, IColorProcessor, IDisposable

0.6.9.2 Member Function Documentation

0.6.9.2.1 CreateColorProcessor()

```
IColorProcessor Thorlabs.TSI.ColorInterfaces.IColorProcessorSDK.CreateColorProcessor ( )
```

Creates an empty color processor. An empty color processor can be configured for a variety of specialized color-processing strategies when color spaces other than sRGB or LinearRGB are needed. sRGB and LinearRGB can be created with CreateStandardRGBColorProcessor or CreateLinearRGBColorProcessor, respectively.

Returns

An empty color processor object.

See also

CreateStandardRGBColorProcessor, CreateLinearRGBColorProcessor

Implemented in Thorlabs.TSI.ColorProcessor.ColorProcessorSDK.

0.6.9.2.2 CreateLinearRGBColorProcessor()

```
IColorProcessor Thorlabs.TSI.ColorInterfaces.IColorProcessorSDK.CreateLinearRGBColorProcessor
(
            float [] whiteBalanceMatrix,
            float [] colorCorrectionMatrix,
            int bitDepth )
```

Create a color processor that can transform three-channel data into a standard-RGB (sRGB) color space but without applying any non-linear conversions normally used for sRGB. This would be equivalent to sRGB with a gamma value of 1.

The color-correction and white-balance matrices are normally read from the camera because they are specific to a particular model of sensor.

Parameters

whiteBalanceMatrix	The white balance matrix, typically read from the camera. This matrix can be easily customized for specific lighting conditions to achieve a white-balanced image.
colorCorrectionMatrix	The color correction matrix, typically read from the camera. Every model of camera sensor has slightly different color characteristics. This matrix compensates for those differences to help bring the image data into a standard color space such as sRGB or LinearRGB color spaces.
bitDepth	The number of relevant bits in each unsigned short pixel value that are relevant. Please see the specification for a particular camera or read the bit depth from the camera.

Returns

A color processor populated with the given color-correction and white-balances matrices.

See also

CreateColorProcessor, CreateStandardRGBColorProcessor

Implemented in Thorlabs.TSI.ColorProcessor.ColorProcessorSDK.

0.6.9.2.3 CreateStandardRGBColorProcessor()

```
IColorProcessor Thorlabs.TSI.ColorInterfaces.IColorProcessorSDK.CreateStandardRGBColorProcessor
(
            float [] whiteBalanceMatrix,
            float [] colorCorrectionMatrix,
            int bitDepth,
            int blackLevel = 0 )
```

Create a color processor that can transform three-channel data into a standard RGB (sRGB) color space.

The color-correction and white-balance matrices are normally read from the camera because they are specific to a particular model of sensor. Depending on the light source, a customized white-balance matrix may be needed.

Parameters

whiteBalanceMatrix	The white balance matrix, typically read from the camera. This matrix can be easily customized for specific lighting conditions to achieve a white-balanced image.
colorCorrectionMatrix	The color correction matrix, typically read from the camera. Every model of camera sensor has slightly different color characteristics. This matrix compensates for those differences to help bring the image data into a standard color space such as sRGB or LinearRGB color spaces.
bitDepth	The number of relevant bits in each unsigned short pixel value that are relevant. Please see the specification for a particular camera or read the bit depth from the camera.
blackLevel	The camera black level. This is used to shift the lookup table.

Returns

An sRGB color processor populated with the given color-correction and white-balances matrices as well as sRGB look-up tables that apply the non-linear transformation required for the sRGB color space.

See also

CreateColorProcessor, CreateLinearRGBColorProcessor

Implemented in Thorlabs.TSI.ColorProcessor.ColorProcessorSDK.

0.6.10 Thorlabs.TSI.ColorInterfaces.IDemosaicker Interface Reference

Inherits IDisposable.

Inherited by Thorlabs.TSI.Demosaicker.Demosaicker.

Public Member Functions

- void Demosaic (int width, int height, int xOrigin, int yOrigin, ColorFilterArrayPhase colorFilterArrayPhase, ColorFormat output_color_format, ColorSensorType colorSensorType, int bitDepth, ushort[ ] singleChannel↩ MosaicPatternInputData, ushort[ ] outputBuffer)

0.6.10.1 Detailed Description

The demosaic module offers a function which accepts an input buffer containing monochrome pixel data (presumably generated by an imaging device capable of producing Bayer pattern image data) and "explodes" that data into 3 color channels of image data.

The main transform function has been optimized using Intel's AVX2 vector instructions to accelerate the computation for machines which support that instruction set. It also includes a legacy scalar processing path for older generation hardware which lacks support for the new instructions.

Currently, the demosaic module supports color sensors which implement a Bayer pattern color pixel array.The implementation can be extended to support other color pixel arrangements.

See http://en.wikipedia.org/wiki/Bayer_filter for more information on Bayer pattern color sensors.

The Bayer pattern primitive that the demosaic module currently supports is:

```
 R     GR

GB     B
```

where:

R = a red pixel

GR = a green pixel next to a red pixel

B = a blue pixel

GB = a green pixel next to a blue pixel

See also

System.IDisposable

0.6.10.2 Member Function Documentation

0.6.10.2.1 Demosaic()

```
void Thorlabs.TSI.ColorInterfaces.IDemosaicker.Demosaic (
             int width,
             int height,
             int xOrigin,
             int yOrigin,
             ColorFilterArrayPhase colorFilterArrayPhase,
             ColorFormat output_color_format,
             ColorSensorType colorSensorType,
             int bitDepth,
             ushort [] singleChannelMosaicPatternInputData,
             ushort [] outputBuffer )
```

This function takes an input buffer containing monochrome image data and writes a color image into the specified output buffer using a standard demosaic computation.

The transformation can be viewed as "expanding" the single channel monochrome pixel data into three color channels of pixel data.

The implementation uses AVX2 vector instructions to accelerate the computation on machines which support that instruction set.

A legacy scalar implementation is also included to support older generation hardware which do not support the new instructions.

The user does not need to choose between the vector vs.scalar implementation - that is done automatically based on a run time interrogation of the CPU capabilities.

Parameters

width	The width (x-axis) in pixels of the region of interest (ROI) specified in the input buffer.
height	The height (y-axis) in pixels of the ROI specified in the input buffer.
xOrigin	The x coordinate of the origin pixel in the ROI relative to the full frame.
yOrigin	The y coordinate of the origin pixel in the ROI relative to the full frame.
colorFilterArrayPhase	The Bayer pattern color of the origin pixel in the full frame.
output_color_format	The desired pixel order that should be used when the color data is written to the output buffer.
colorSensorType	The color filter type (such as Bayer pattern) of the device which produced the input data.
bitDepth	The pixel bit depth of the input buffer data. The maximum number of bits used to specify a pixel intensity value in the input buffer.
singleChannelMosaicPatternInputData	A pointer to the 16-bit input monochrome data.
outputBuffer	A pointer to the output 16-bit color data buffer.

Implemented in Thorlabs.TSI.Demosaicker.Demosaicker.

0.6.11 Thorlabs.TSI.ImageData.ImageDataUShort1D Class Reference

Inherits IImageDataUShort1D.

Public Member Functions

- ImageDataUShort1D (ushort[ ] imageData_monoOrBGR, int width_pixels, int height_pixels, int bitDepth, ImageDataFormat imageDataFormat)
- Bitmap ToBitmap_Format24bppRgb ()
- void ToTiff (string filePath, int bitsPerPixel)
- BitmapSource ToBitmapSource_FormatBgr24 ()
- BitmapSource ToBitmapSource_FormatRgb48OrGray16 ()
- override string ToString ()

Public Attributes

- ushort this[int row, int column, int channelIndex] => this.ImageData_monoOrBGR[row * this.Width_pixels * this.NumberOfChannels + column * this.NumberOfChannels + channelIndex]
- ushort this[int row, int column]

Properties

- int BitDepth `[get]`
- int Height_pixels `[get]`
- ushort [ ] ImageData_monoOrBGR `[get]`
- ImageDataFormat ImageDataFormat `[get]`
- int NumberOfChannels `[get]`
- int Width_pixels `[get]`

0.6.11.1 Detailed Description

0.6.11.2 Constructor & Destructor Documentation

0.6.11.2.1 ImageDataUShort1D()

```
Thorlabs.TSI.ImageData.ImageDataUShort1D.ImageDataUShort1D (
            ushort [] imageData_monoOrBGR,
            int width_pixels,
            int height_pixels,
            int bitDepth,
            ImageDataFormat imageDataFormat )  [inline]
```

This constructor normally called by the camera SDK.

0.6.11.3 Member Function Documentation

0.6.11.3.1 ToBitmap_Format24bppRgb()

```
Bitmap Thorlabs.TSI.ImageData.ImageDataUShort1D.ToBitmap_Format24bppRgb ( )  [inline]
```

0.6.11.3.2 ToBitmapSource_FormatBgr24()

```
BitmapSource Thorlabs.TSI.ImageData.ImageDataUShort1D.ToBitmapSource_FormatBgr24 ( )
```

0.6.11.3.3 ToBitmapSource_FormatRgb48OrGray16()

```
BitmapSource Thorlabs.TSI.ImageData.ImageDataUShort1D.ToBitmapSource_FormatRgb48OrGray16 ( )
```

0.6.11.3.4 ToString()

```
override string Thorlabs.TSI.ImageData.ImageDataUShort1D.ToString ( )
```

0.6.11.3.5 ToTiff()

```
void Thorlabs.TSI.ImageData.ImageDataUShort1D.ToTiff (
            string filePath,
            int bitsPerPixel )  [inline]
```

0.6.11.4 Member Data Documentation

0.6.11.4.1 this[int row, int column, int channelIndex]

```
ushort Thorlabs.TSI.ImageData.ImageDataUShort1D.this[int row, int column, int channelIndex]
=> this.ImageData_monoOrBGR[row * this.Width_pixels * this.NumberOfChannels + column * this.↩
NumberOfChannels + channelIndex]
```

Internal use only.

0.6.11.4.2 this[int row, int column]

```
ushort Thorlabs.TSI.ImageData.ImageDataUShort1D.this[int row, int column]
```

Initial value:

```
=> (ushort)(0.3 * this.ImageData_monoOrBGR[row * this.Width_pixels * this.NumberOfChannels + column *
       this.NumberOfChannels + 2]
                                                 + 0.6 * this.ImageData_monoOrBGR[row *
       this.Width_pixels * this.NumberOfChannels + column * this.NumberOfChannels + 1]
                                                 + 0.1 * this.ImageData_monoOrBGR[row *
       this.Width_pixels * this.NumberOfChannels + column * this.NumberOfChannels + 0])
```

Internal use only.

0.6.11.5 Property Documentation

0.6.11.5.1 BitDepth

```
int Thorlabs.TSI.ImageData.ImageDataUShort1D.BitDepth  [get]
```

0.6.11.5.2 Height_pixels

```
int Thorlabs.TSI.ImageData.ImageDataUShort1D.Height_pixels  [get]
```

0.6.11.5.3 ImageData_monoOrBGR

```
ushort [] Thorlabs.TSI.ImageData.ImageDataUShort1D.ImageData_monoOrBGR  [get]
```

0.6.11.5.4 ImageDataFormat

```
ImageDataFormat Thorlabs.TSI.ImageData.ImageDataUShort1D.ImageDataFormat  [get]
```

0.6.11.5.5 NumberOfChannels

```
int Thorlabs.TSI.ImageData.ImageDataUShort1D.NumberOfChannels  [get]
```

0.6.11.5.6 Width_pixels

```
int Thorlabs.TSI.ImageData.ImageDataUShort1D.Width_pixels  [get]
```

0.6.12 Thorlabs.TSI.PolarizationInterfaces.IPolarizationProcessor Interface Reference

Inherits IDisposable.

Public Member Functions

- unsafe void TransformToIntensity (PolarPhase sensorPolarPhase, ushort[ ] inputImage, int imageOriginX_pixels, int imageOriginY_pixels, int imageWidth_pixels, int imageHeight_pixels, int inputImageBitDepth, ushort outputMaxValue, ushort[ ] output_TotalOpticalPower)
- unsafe void TransformToDoLP (PolarPhase sensorPolarPhase, ushort[ ] inputImage, int imageOriginX_pixels, int imageOriginY_pixels, int imageWidth_pixels, int imageHeight_pixels, int inputImageBitDepth, ushort outputMaxValue, ushort[ ] output_DoLP)
- unsafe void TransformToAzimuth (PolarPhase sensorPolarPhase, ushort[ ] inputImage, int imageOriginX_pixels, int imageOriginY_pixels, int imageWidth_pixels, int imageHeight_pixels, int inputImageBitDepth, ushort outputMaxValue, ushort[ ] output_Azimuth)

0.6.12.1 Detailed Description

The transform functions return an output buffer with values from 0 to max value for each pixel.
To calculate scaled values for each pixel, you may use the following equation:

- polarization_parameter_value = pixel_integer_value $*$ (max_parameter_value / (2^{bit_depth} - 1)) + min_parameter_value

The suggested min and max for each type are as follows:

- Azimuth: (range from -90° to 90°)
  - min_parameter_value = -90
  - max_parameter_value = 180
- DoLP: (range from 0 to 100%)
  - min_parameter_value = 0
  - max_parameter_value = 100
- Intensity: (range from 0 to 100)
  - min_parameter_value = 0
  - max_parameter_value = 100

0.6.12.2 Member Function Documentation

0.6.12.2.1 TransformToAzimuth()

```
unsafe void Thorlabs.TSI.PolarizationInterfaces.IPolarizationProcessor.TransformToAzimuth (
            PolarPhase sensorPolarPhase,
            ushort [] inputImage,
            int imageOriginX_pixels,
            int imageOriginY_pixels,
            int imageWidth_pixels,
            int imageHeight_pixels,
            int inputImageBitDepth,
            ushort outputMaxValue,
            ushort [] output_Azimuth )
```

Transforms raw-image data into an azimuth-output buffer, which shows the angle of polarized light at each pixel.

Parameters

sensorPolarPhase	The polar phase (in degrees) of the origin (top-left) pixel of the camera sensor.
inputImage	Unprocessed input image delivered by the camera.
imageOriginX_pixels	The X position of the origin (top-left) of the input image on the sensor. Warning: Some camera models may nudge input ROI values. Please read values back from camera. See ITLCamera.ROIAndBin.
imageOriginY_pixels	The Y position of the origin (top-left) of the input image on the sensor. Warning: Some camera models may nudge input ROI values. Please read values back from camera. See ITLCamera.ROIAndBin.
imageWidth_pixels	Width of input image in pixels
imageHeight_pixels	Height of input image in pixels
inputImageBitDepth	Bit depth of input image pixels
outputMaxValue	The maximum possible pixel value in the output images. Must be between 1 and 65535.
output_Azimuth	Buffer for storing the azimuth (polar angle) output. The buffer must be pre-allocated.

0.6.12.2.2 TransformToDoLP()

```
unsafe void Thorlabs.TSI.PolarizationInterfaces.IPolarizationProcessor.TransformToDoLP (
            PolarPhase sensorPolarPhase,
            ushort [] inputImage,
            int imageOriginX_pixels,
            int imageOriginY_pixels,
            int imageWidth_pixels,
            int imageHeight_pixels,
            int inputImageBitDepth,
            ushort outputMaxValue,
            ushort [] output_DoLP )
```

Transforms raw-image data into a DoLP (degree of linear polarization) output buffer, which is a measure of how polarized the light is from none to totally polarized.

Parameters

sensorPolarPhase	The polar phase (in degrees) of the origin (top-left) pixel of the camera sensor.
inputImage	Unprocessed input image delivered by the camera.
imageOriginX_pixels	The X position of the origin (top-left) of the input image on the sensor. Warning: Some camera models may nudge input ROI values. Please read values back from camera. See ITLCamera.ROIAndBin.
imageOriginY_pixels	The Y position of the origin (top-left) of the input image on the sensor. Warning: Some camera models may nudge input ROI values. Please read values back from camera. See ITLCamera.ROIAndBin.
imageWidth_pixels	Width of input image in pixels
imageHeight_pixels	Height of input image in pixels
inputImageBitDepth	Bit depth of input image pixels
outputMaxValue	The maximum possible pixel value in the output images. Must be between 1 and 65535.
output_DoLP	Buffer for storing the DoLP (degree of linear polarization) output. The buffer must be pre-allocated.

0.6.12.2.3 TransformToIntensity()

```
unsafe void Thorlabs.TSI.PolarizationInterfaces.IPolarizationProcessor.TransformToIntensity (
            PolarPhase sensorPolarPhase,
            ushort [] inputImage,
            int imageOriginX_pixels,
            int imageOriginY_pixels,
            int imageWidth_pixels,
            int imageHeight_pixels,
            int inputImageBitDepth,
            ushort outputMaxValue,
            ushort [] output_TotalOpticalPower )
```

Transforms raw-image data into the intensity-output buffer, which shows the brightness of the light at each pixel.

Parameters

sensorPolarPhase	The polar phase (in degrees) of the origin (top-left) pixel of the camera sensor.
inputImage	Unprocessed input image delivered by the camera.
imageOriginX_pixels	The X position of the origin (top-left) of the input image on the sensor. Warning: Some camera models may nudge input ROI values. Please read values back from camera. See ITLCamera.ROIAndBin.
imageOriginY_pixels	The Y position of the origin (top-left) of the input image on the sensor. Warning: Some camera models may nudge input ROI values. Please read values back from camera. See ITLCamera.ROIAndBin.
imageWidth_pixels	Width of input image in pixels
imageHeight_pixels	Height of input image in pixels
inputImageBitDepth	Bit depth of input image pixels
outputMaxValue	The maximum possible pixel value in the output images. Must be between 1 and 65535.
output_TotalOpticalPower	Buffer for storing the total optical power (intensity) output. The buffer must be pre-allocated.

0.6.13 Thorlabs.TSI.PolarizationInterfaces.IPolarizationProcessorSDK Interface Reference

Inherits IDisposable.

Public Member Functions

- IPolarizationProcessor CreatePolarizationProcessor ()

0.6.13.1 Detailed Description

The polarization processor SDK loads DLLs into memory and provides function to creates instances of PolarizationProcessor.

Be sure to dispose all PolarizationProcessor objects and then dispose the PolarizationProcessorSDK before exiting the application.

0.6.13.2 Member Function Documentation

0.6.13.2.1 CreatePolarizationProcessor()

```
IPolarizationProcessor Thorlabs.TSI.PolarizationInterfaces.IPolarizationProcessorSDK.Create↩
PolarizationProcessor ( )
```

Creates a polarization processor.

Returns

0.6.14 Thorlabs.TSI.TLCameraInterfaces.ITLCamera Interface Reference

Inherits IDisposable.

Public Member Functions

- void Arm ()
- void Disarm ()
- void FreeAllButGivenNumberOfFrames (int numberOfFramesToLeaveInTheQueue)
- float [ ] GetCameraColorCorrectionMatrix ()
- float [ ] GetDefaultWhiteBalanceMatrix ()
- bool GetIsDataRateSupported (DataRate dataRate)
- bool GetIsOperationModeSupported (OperationMode operationMode)
- bool GetIsTapsSupported (Taps taps)
- double GetMeasuredFrameRate ()
- double ConvertGainToDecibels (uint index)
- uint ConvertDecibelsToGain (double gain_dB)
- Frame GetPendingFrameOrNull ()
- void GetROIRange (out uint upper_left_x_min, out uint upper_left_x_max, out uint upper_left_y_min, out uint upper_left_y_max, out uint lower_right_x_min, out uint lower_right_x_max, out uint lower_right_y_min, out uint lower_right_y_max)
- void IssueSoftwareTrigger ()
- void SetIsEEPEnabled (bool isEnabled)

Properties

- Range< uint > BinXRange [get]
- Range< uint > BinYRange [get]
- uint BitDepth [get]
- uint BlackLevel [get, set]
- Range< uint > BlackLevelRange [get]
- ColorCorrectionMatrixOutputColorSpace CameraColorCorrectionMatrixOutputColorSpace [get]
- CameraSensorType CameraSensorType [get]
- CameraUSBPortType CameraUSBPortType [get]
- ColorFilterArrayPhase ColorFilterArrayPhase [get]
- CommunicationInterface CommunicationInterface [get]
- DataRate DataRate [get, set]
- EEPStatus EEPStatus [get]
- uint ExposureTime_us [get, set]
- Range< uint > ExposureTimeRange_us [get]
- string FirmwareVersion [get]
- uint FramesPerTrigger_zeroForUnlimited [get, set]
- Range< uint > FramesPerTriggerRange [get]
- float FrameTime_us [get]
- bool IsFrameRateControlEnabled [get, set]
- double FrameRateControlValue_fps [get, set]
- Range< double > FrameRateControlValueRange_fps [get]
- uint Gain [get, set]
- Range< uint > GainRange [get]
- uint HotPixelCorrectionThreshold [get, set]
- Range< uint > HotPixelCorrectionThresholdRange [get]
- int ImageHeight_pixels [get]
- int ImageWidth_pixels [get]
- bool IsArmed [get]
- bool IsCoolingAdjustable [get]
- bool IsCoolingEnabled [get, set]
- bool IsCoolingSupported [get]
- bool IsEEPSupported [get]
- bool IsHotPixelCorrectionEnabled [get, set]
- bool IsLEDOn [get, set]
- bool IsLEDSupported [get]
- bool IsNIRBoostEnabled [get, set]
- bool IsNIRBoostSupported [get]
- bool IsTapBalanceEnabled [get, set]
- int MaximumNumberOfFramesToQueue [get, set]
- string Model [get]
- string Name [get, set]
- int NumberOfQueuedFrames [get]
- OperationMode OperationMode [get, set]
- PolarPhase PolarPhase [get]
- ROIAndBin ROIAndBin [get, set]
- uint SensorHeight_pixels [get]
- float SensorPixelHeight_um [get]
- uint SensorPixelSize_bytes [get]
- float SensorPixelWidth_um [get]
- uint SensorReadoutTime_ns [get]
- uint SensorWidth_pixels [get]
- string SerialNumber [get]
- Taps Taps [get, set]
- TriggerPolarity TriggerPolarity [get, set]

Events

- OnDiagnosticsMessageDelegate OnDiagnosticsMessage
- OnImageFrameAvailableDelegate OnImageFrameAvailable

0.6.14.1 Detailed Description

0.6.14.2 Member Function Documentation

0.6.14.2.1 Arm()

```
void Thorlabs.TSI.TLCameraInterfaces.ITLCamera.Arm ( )
```

Before issuing software or hardware triggers to get images from a camera, prepare it for imaging by calling ITLCamera.Arm().

Depending on the OperationMode, either call ITLCamera.IssueSoftwareTrigger or issue a hardware trigger.

To start a camera in continuous mode, set the OperationMode to SoftwareTriggered, FramesPerTrigger_zeroFor↩ Unlimited to zero, Arm the camera, and then call IssueSoftwareTrigger one time. The camera will then self-trigger frames until Disarm() or Dispose() is called.

To start a camera for hardware triggering, set the OperationMode to either HardwareTriggered or Bulb, Frames↩ PerTrigger_zeroForUnlimited to one, TriggerPolarity to rising- edge or falling-edge triggered, arm the camera, and then issue a triggering signal on the trigger input.

Use the IsArmed property to determine whether Arm() has been called for the camera.

If any images are still in the queue when calling Arm(), they will be considered stale and cleared from the queue.

For more information on the proper procedure for triggering frames and receiving them from the camera, please see the Getting Started section.

See also

IsArmed, OperationMode, Disarm, IssueSoftwareTrigger, TriggerPolarity, FramesPerTrigger_zeroForUnlimited

0.6.14.2.2 ConvertDecibelsToGain()

```
uint Thorlabs.TSI.TLCameraInterfaces.ITLCamera.ConvertDecibelsToGain (
            double gain_dB )
```

The gain value is set with the Gain property. The range of possible gain values varies by camera model. It can be retrieved from the camera with GainRange. This conversion method can be used to convert gain in decibels (found using ConvertGainToDecibels) back to its gain index.

Parameters

gain_dB	A gain value found by converting the gain retrieved from the camera to decibels.

Returns

The gain value as an integer index into the available gain range for specific the camera model.

See also

Gain, GainRange

0.6.14.2.3 ConvertGainToDecibels()

```
double Thorlabs.TSI.TLCameraInterfaces.ITLCamera.ConvertGainToDecibels (
            uint index )
```

The gain value is set with the Gain property. The range of possible gain values varies by camera model. It can be retrieved from the camera with GainRange. The gain value units vary by camera model, but with this conversion function, it can be converted to decibels (dB).

Parameters

index	A gain value retrieved from the camera through the Gain property.

Returns

The gain value converted to decibels (dB).

See also

Gain, GainRange

0.6.14.2.4 Disarm()

```
void Thorlabs.TSI.TLCameraInterfaces.ITLCamera.Disarm ( )
```

When finished issuing software or hardware triggers, call Disarm(). This allows setting parameters that are not available in armed mode such as ROIAndBin or OperationMode.

The camera will automatically disarm when ITLCamera.Dispose is called.

To determine whether the camera is armed, use the IsArmed property.

Disarming the camera does not clear the image queue. Callbacks will continue to be invoked or polling can continue until the queue is empty. When calling Arm() again, the queue will be automatically cleared.

See also

Arm, IsArmed

Exceptions

InvalidOperationException	

0.6.14.2.5 FreeAllButGivenNumberOfFrames()

```
void Thorlabs.TSI.TLCameraInterfaces.ITLCamera.FreeAllButGivenNumberOfFrames (
            int numberOfFramesToLeaveInTheQueue )
```

To control the amount of image buffering, set the MaximumNumberOfFramesToQueue property. By default, there is no buffering.

To discard some or all of the frames in the buffer, call this method, indicating the desired number of frames to be left in the queue. To clear the entire queue, specify zero frames. Otherwise, the oldest frames are cleared, leaving the newest ones in the queue.

To determine the current number of queued frames, use the NumberOfQueuedFrames property.

Note that when calling Arm(), any images still in the queue will be automatically cleared.

Parameters

numberOfFramesToLeaveInTheQueue	Number of frames to leave in the queue. A value of zero will clear the queue.

See also

NumberOfQueuedFrames, MaximumNumberOfFramesToQueue

0.6.14.2.6 GetCameraColorCorrectionMatrix()

```
float [] Thorlabs.TSI.TLCameraInterfaces.ITLCamera.GetCameraColorCorrectionMatrix ( )
```

Each scientific-CCD color camera includes a three-by-three matrix that can be used to achieve consistent color for different camera models.

Returns

A one-dimensional array of doubles ordered as follows:
0 1 2
3 4 5
6 7 8

0.6.14.2.7 GetDefaultWhiteBalanceMatrix()

```
float [] Thorlabs.TSI.TLCameraInterfaces.ITLCamera.GetDefaultWhiteBalanceMatrix ( )
```

Each scientific-CCD color camera includes a three-by-three matrix that corrects white balance for the default color temperature.

Returns

A one-dimensional array of doubles ordered as follows:
0 1 2
3 4 5
6 7 8

0.6.14.2.8 GetIsDataRateSupported()

```
bool Thorlabs.TSI.TLCameraInterfaces.ITLCamera.GetIsDataRateSupported (
            DataRate dataRate )
```

Scientific-CCD cameras and compact-scientific cameras handle sensor- level data-readout speed differently.

Use this method to test whether the connected camera supports a particular data rate.

For more details about the data rate options, see the DataRate property.

Parameters

dataRate	The data rate such as ReadoutSpeed40MHz or FPS50, depending on the camera model.

Returns

True if the given data rate is supported by the connected camera, false if it is not.

See also

DataRate

0.6.14.2.9 GetIsOperationModeSupported()

```
bool Thorlabs.TSI.TLCameraInterfaces.ITLCamera.GetIsOperationModeSupported (
            OperationMode operationMode )
```

Cameras can be operated in several distinct modes. However, not all modes exist on all cameras.

Use this method to test whether a connected camera supports a particular mode.

Parameters

operationMode	The desired operation mode.

Returns

True if the connected camera supports the given operation mode, false if not.

See also

OperationMode

0.6.14.2.10 GetIsTapsSupported()

```
bool Thorlabs.TSI.TLCameraInterfaces.ITLCamera.GetIsTapsSupported (
            Taps taps )
```

All CCD cameras support a single tap. Some also support dual tap or quad tap. Use this method to test whether a connected camera supports a particular Taps value.

For more information on taps and tap balancing, see IsTapBalanceEnabled.

Parameters

taps	The desired taps mode.

Returns

True if the connected camera supports the given taps mode, false if not.

See also

IsTapBalanceEnabled

0.6.14.2.11 GetMeasuredFrameRate()

```
double Thorlabs.TSI.TLCameraInterfaces.ITLCamera.GetMeasuredFrameRate ( )
```

Gets the current rate of frames in frames per second that are delivered to the host computer. The frame rate can be affected by the performance capabilities of the host computer and the communication interface.

This method can be polled for updated values as needed.

Returns

The current frame rate in frames per second as measured by the camera SDK.

0.6.14.2.12 GetPendingFrameOrNull()

```
Frame Thorlabs.TSI.TLCameraInterfaces.ITLCamera.GetPendingFrameOrNull ( )
```

ITLCamera offers two ways to get image data from the camera:

1. Poll for data using GetPendingFrameOrNull() from a single thread, usually the main thread.

2. Register for the OnImageFrameAvailable event. The event will arrive on a worker thread to avoid blocking the main thread. Because of this, it might be necessary to marshal the data to the user-interface thread in order to display it. In doing so, it is important to protect any shared data with proper thread protections.

   In the event handler, GetPendingFrameOrNull() may be used to get the data.

   A 'Frame' object includes image data, meta data, and a frame number.

   To get the image data, downcast the IImageData to the specific type:

   IImageDataUShort1D rawImageData = ((IImageDataUShort1D)frame.ImageData).ImageData_monoOr↩ BGR;

   The data for both color and monochrome cameras is ordered left to right across a row followed by the next row below it.
   For monochrome and color cameras, each pixel requires two bytes.
   For color cameras, it is necessary to demosaic the image in order to get separate blue, red, and green channels for each pixel. A performant demosaic algorithm is provided in Demosaicker.Demosaic(). Once demosaicked, each pixel requires two bytes for blue followed by two bytes for green followed by two bytes for red (BBGGRRBBGGRRBBGGRR...). Therefore, each color pixel requires six bytes.

   For convenience, several conversion functions are provided:

   To get a WinForms 8-bit displayable Bitmap, use IImageData.ToBitmap_Format24bppRgb()
   To get a WPF 8-bit displayable BitmapSource, use IImageData.ToBitmapSource_FormatBgr24
   To get a WPF 16-bit BitmapSource, use ToBitmapSource_FormatRgb48OrGray16

   For more details, see the Getting Started section of this manual.

Returns

The pending frame or null if there is no pending frame.

0.6.14.2.13 GetROIRange()

```
void Thorlabs.TSI.TLCameraInterfaces.ITLCamera.GetROIRange (
            out uint upper_left_x_min,
            out uint upper_left_x_max,
            out uint upper_left_y_min,
            out uint upper_left_y_max,
            out uint lower_right_x_min,
            out uint lower_right_x_max,
```

```
            out uint lower_right_y_min,
            out uint lower_right_y_max )
```

The rules for rectangular regions of interest (ROIs) vary by camera model. Please consult the camera documentation for more details. The ROI height range indicates the smallest height to which an ROI can be set up to a maximum of the sensor's vertical resolution. The ROI width range indicates the smallest width to which an ROI can be set up to a maximum of the sensor's horizontal resolution.

0.6.14.2.14 IssueSoftwareTrigger()

```
void Thorlabs.TSI.TLCameraInterfaces.ITLCamera.IssueSoftwareTrigger ( )
```

If the OperationMode is set to SoftwareTriggered and Arm() is called, then calling IssueSoftwareTrigger will generate a trigger through the camera SDK rather than through the hardware trigger input.

The behavior of a software trigger depends on the FramesPerTrigger_zeroForUnlimited property.

- If FramesPerTrigger_zeroForUnlimited is set to zero, then a single software trigger will start continuous-video mode.
- If FramesPerTrigger_zeroForUnlimited is set to one or higher, then one software trigger will generate a corresponding number of frames.

  In this case, it is important to avoid issuing subsequent software triggers until FrameTime_us multiplied by the number of frames has elapsed. If insufficient time elapses, the trigger is ignored by the camera. FrameTime_us is based on ExposureTime_us, SensorReadoutTime_ns, and FramesPerTrigger_zeroFor↩ Unlimited. NOTE: Some versions of camera firmware exhibit a longer frame time than is ideal. Please check the website for the latest available firmware.

  NOTE:For scientific-CCD cameras, after issuing a software trigger, it is necessary to wait at least 300ms before adjusting the Exposure_us property.

See also

FrameTime_us, ExposureTime_us, SensorReadoutTime_ns, FramesPerTrigger_zeroForUnlimited

0.6.14.2.15 SetIsEEPEnabled()

```
void Thorlabs.TSI.TLCameraInterfaces.ITLCamera.SetIsEEPEnabled (
            bool isEnabled )
```

Equal Exposure Pulse (EEP) mode is an LVTTL-level signal that is active between the time when all rows have been reset during rolling reset, and the end of the exposure time (and the beginning of rolling readout). The signal can be used to control an external light source that will be triggered on only during the equal exposure period, providing the same amount of exposure for all pixels in the ROI.

Please see the camera specification for details on EEP mode.

When enabled, EEP mode will be active or inactive depending on the exposure duration.

Use IsEEPSupported to test whether the connected camera supports this mode.

Use EEPStatus to see whether the mode is active, inactive, in bulb mode, or disabled.

See also

IsEEPSupported, EEPStatus

0.6.14.3 Property Documentation

0.6.14.3.1 BinXRange

```
Range<uint> Thorlabs.TSI.TLCameraInterfaces.ITLCamera.BinXRange [get]
```

The binning ratio in the X direction can be determined with this property. By default, binning is set to one in both X and Y directions. For details on binning, see ROIAndBin.

See also

BinYRange, ROIAndBin

0.6.14.3.2 BinYRange

```
Range<uint> Thorlabs.TSI.TLCameraInterfaces.ITLCamera.BinYRange [get]
```

The binning ratio in the Y direction can be determined with this property. By default, binning is set to one in both X and Y directions. For details on binning, see ROIAndBin.

See also

BinXRange, ROIAndBin

0.6.14.3.3 BitDepth

```
uint Thorlabs.TSI.TLCameraInterfaces.ITLCamera.BitDepth [get]
```

The number of bits to which a pixel value is digitized on a camera.

In the image data that is delivered to the host application, the bit depth indicates how many of the lower bits of each 16-bit ushort value are relevant.

While most cameras operate at a fixed bit depth, some are reduced when data bandwidth limitations would otherwise restrict the frame rate. Please consult the camera manual and specification for details about a specific model.

0.6.14.3.4 BlackLevel

```
uint Thorlabs.TSI.TLCameraInterfaces.ITLCamera.BlackLevel [get], [set]
```

Black level adds an offset to pixel values.

If the connected camera supports BlackLevel, the BlackLevelRange will have a maximum greater than zero.

See also

BlackLevelRange

0.6.14.3.5 BlackLevelRange

```
Range<uint> Thorlabs.TSI.TLCameraInterfaces.ITLCamera.BlackLevelRange [get]
```

Black level adds an offset to pixel values. If BlackLevel is supported by a camera model, then BlackLevelRange will have an upper range higher than zero.

BlackLevelRange indicates the available values that can be used for the BlackLevel property.

See also

BlackLevel

0.6.14.3.6 CameraColorCorrectionMatrixOutputColorSpace

```
ColorCorrectionMatrixOutputColorSpace Thorlabs.TSI.TLCameraInterfaces.ITLCamera.CameraColor↩
CorrectionMatrixOutputColorSpace [get]
```

This describes the camera color correction matrix output color space.

0.6.14.3.7 CameraSensorType

```
CameraSensorType Thorlabs.TSI.TLCameraInterfaces.ITLCamera.CameraSensorType [get]
```

Cameras can be color (bayer), monochrome, or monochrome polarized.

0.6.14.3.8 CameraUSBPortType

```
CameraUSBPortType Thorlabs.TSI.TLCameraInterfaces.ITLCamera.CameraUSBPortType [get]
```

This property describes the computer USB type, such as USB3.0, USB2.0, etc.

0.6.14.3.9 ColorFilterArrayPhase

```
ColorFilterArrayPhase Thorlabs.TSI.TLCameraInterfaces.ITLCamera.ColorFilterArrayPhase [get]
```

This describes the camera color filter phase.

0.6.14.3.10 CommunicationInterface

```
CommunicationInterface Thorlabs.TSI.TLCameraInterfaces.ITLCamera.CommunicationInterface [get]
```

This property describes the computer interface type, such as USB, GigE, or CameraLink.

0.6.14.3.11 DataRate

```
DataRate Thorlabs.TSI.TLCameraInterfaces.ITLCamera.DataRate  [get], [set]
```

This property sets or gets the sensor-level data rate.

Scientific-CCD cameras offer data rates of 20MHz or 40MHz.

Compact-scientific cameras offer FPS30 or FPS50, which are frame rates supported by the camera when doing full-frame readout. The actual rate can vary if a region of interest (ROI) or binning is set or if the host computer cannot keep up with the camera.

To test whether the connected camera supports a particular data rate, use GetIsDataRateSupported(...).

See also

GetIsDataRateSupported

0.6.14.3.12 EEPStatus

```
EEPStatus Thorlabs.TSI.TLCameraInterfaces.ITLCamera.EEPStatus  [get]
```

Equal Exposure Pulse (EEP) mode is an LVTTL-level signal that is active between the time when all rows have been reset during rolling reset, and the end of the exposure time (and the beginning of rolling readout). The signal can be used to control an external light source that will be triggered on only during the equal exposure period, providing the same amount of exposure for all pixels in the ROI.

Even if enabled, EEP mode will be active or inactive depending on the exposure duration.

Please see the camera documentation for details on EEP mode.

To determine whether the connected camera supports EEP mode, check IsEEPSupported.

Note that EEPStatus is different than enabled/disabled because even when the mode is enabled, it can present three different types of status. See the Thorlabs.TSI.TLCameraInterfaces.EEPStatus enumeration for details.

See also

IsEEPSupported

0.6.14.3.13 ExposureTime_us

```
uint Thorlabs.TSI.TLCameraInterfaces.ITLCamera.ExposureTime_us  [get], [set]
```

The time, in microseconds (us), that charge is integrated on the image sensor.

To convert milliseconds to microseconds, multiply the milliseconds by 1,000.
To convert microseconds to milliseconds, divide the microseconds by 1,000.

IMPORTANT: After issuing a software trigger, it is recommended to wait at least 300ms before setting exposure.

See also

ExposureTimeRange_us

0.6.14.3.14 ExposureTimeRange_us

```
Range<uint> Thorlabs.TSI.TLCameraInterfaces.ITLCamera.ExposureTimeRange_us [get]
```

The minimum and maximum exposures, in microseconds (us), supported by the connected camera.

To convert milliseconds to microseconds, multiply the milliseconds by 1,000.
To convert microseconds to milliseconds, divide the microseconds by 1,000.

See also

ExposureTime_us

0.6.14.3.15 FirmwareVersion

```
string Thorlabs.TSI.TLCameraInterfaces.ITLCamera.FirmwareVersion [get]
```

This property return the firmware version.

0.6.14.3.16 FrameRateControlValue_fps

```
double Thorlabs.TSI.TLCameraInterfaces.ITLCamera.FrameRateControlValue_fps [get], [set]
```

The frame rate control value in frames per second for the specified camera. the connected camera supports FrameRateControlValue_fps, the FrameRateControlValueRange_fps will have a maximum greater than zero.

0.6.14.3.17 FrameRateControlValueRange_fps

```
Range<double> Thorlabs.TSI.TLCameraInterfaces.ITLCamera.FrameRateControlValueRange_fps [get]
```

Gets the range of acceptable values for the frame-rate control setting for the specified camera.

If the range maximum is zero, then FrameRateControlValue_fps is not supported for the connected camera.

0.6.14.3.18 FramesPerTrigger_zeroForUnlimited

```
uint Thorlabs.TSI.TLCameraInterfaces.ITLCamera.FramesPerTrigger_zeroForUnlimited [get], [set]
```

The number of frames/images generated per software or hardware trigger can be unlimited or finite.

To capture an unlimited number of frames, set this parameter to zero. This will create a continuous stream of frames with overlapped exposure and readout. This will result in the highest achievable frame rate. The sequence can be terminated with Disarm().

If set to one or higher, a single software or hardware trigger will generate the prescribed number of frames and then stop. In this case, the camera will not operate in overlapped mode. It will complete reading out a frame before starting to expose the next frame. In this case, the frame time will be approximately the ExposureTime_us plus the SensorReadoutTime_ns. See the timing diagrams in the camera user guide for details.

See also

IssueSoftwareTrigger, Disarm, ExposureTime_us, SensorReadoutTime_ns

0.6.14.3.19 FramesPerTriggerRange

```
Range<uint> Thorlabs.TSI.TLCameraInterfaces.ITLCamera.FramesPerTriggerRange [get]
```

The number of frames generated per software or hardware trigger can be unlimited or finite.

If set to zero, the camera will self-trigger indefinitely, allowing a continuous video feed.

If set to one or higher, a single software or hardware trigger will generate the prescribed number of frames and then stop.

This property returns the valid range for FramesPerTrigger_zeroForUnlimited.

0.6.14.3.20 FrameTime_us

```
float Thorlabs.TSI.TLCameraInterfaces.ITLCamera.FrameTime_us [get]
```

The time, in microseconds (us), required for a frame to be exposed and read out from the sensor. When triggering frames, this property may be used to determine when the camera is ready to accept another trigger. Other factors such as the communication speed between the camera and the host computer can affect the maximum trigger rate.

NOTE: This parameter depends on FramesPerTrigger_zeroForUnlimited and ExposureTime_us.

0.6.14.3.21 Gain

```
uint Thorlabs.TSI.TLCameraInterfaces.ITLCamera.Gain [get], [set]
```

Gain refers to the scaling of pixel values up or down for a given amount of light. This scaling is applied prior to digitization.

To determine the valid range of values, use the GainRange property.

If the GainRange maximum is zero, then Gain is not supported for the connected camera.

Query the GainRange property to determine the possible values for this property.

The units of measure for Gain can vary by camera. Please consult the data sheet for the specific camera model or use the ConvertGainToDecibels helper function.

See also

GainRange, ConvertGainToDecibels, ConvertDecibelsToGain

0.6.14.3.22 GainRange

```
Range<uint> Thorlabs.TSI.TLCameraInterfaces.ITLCamera.GainRange [get]
```

Gain refers to the scaling of pixel values up or down for a given amount of light. This scaling is applied prior to digitization.

The gain range provides the values that can be set on the Gain property.

If the range maximum is zero, then Gain is not supported for the connected camera.

The units of measure for Gain can vary by camera. Please consult the data sheet for the specific camera model or use the ConvertGainToDecibels helper function.

See also

Gain, ConvertGainToDecibels, ConvertDecibelsToGain

0.6.14.3.23 HotPixelCorrectionThreshold

```
uint Thorlabs.TSI.TLCameraInterfaces.ITLCamera.HotPixelCorrectionThreshold [get], [set]
```

Due to variability in manufacturing, some pixels have inherently higher dark current which manifests as abnormally bright pixels in images, typically visible with longer exposures. Hot-pixel correction identifies hot pixels and then substitutes a calculated value based on the values of neighboring pixels in place of hot pixels.

This property may be used to get or set the hot-pixel correction threshold within the available range.

To determine the available range, query the HotPixelCorrectionThresholdRange property.

If the threshold range maximum is zero, the connected camera does not support hot-pixel correction.

To enable hot-pixel correction, use IsHotPixelCorrectionEnabled.

See also

IsHotPixelCorrectionEnabled, HotPixelCorrectionThresholdRange

0.6.14.3.24 HotPixelCorrectionThresholdRange

```
Range<uint> Thorlabs.TSI.TLCameraInterfaces.ITLCamera.HotPixelCorrectionThresholdRange [get]
```

Due to variability in manufacturing, some pixels have inherently higher dark current which manifests as abnormally bright pixels in images, typically visible with longer exposures. Hot-pixel correction identifies hot pixels and then substitutes a calculated value based on the values of neighboring pixels in place of hot pixels.

This property may be used to discover the available threshold range for the connected camera.

If the threshold range maximum is zero, the connected camera does not support hot-pixel correction.

See also

IsHotPixelCorrectionEnabled, HotPixelCorrectionThreshold

0.6.14.3.25 ImageHeight_pixels

```
int Thorlabs.TSI.TLCameraInterfaces.ITLCamera.ImageHeight_pixels [get]
```

This property provide the image height in pixels. It is related to ROI height.

0.6.14.3.26 ImageWidth_pixels

```
int Thorlabs.TSI.TLCameraInterfaces.ITLCamera.ImageWidth_pixels [get]
```

This property provide the image width in pixels. It is related to ROI width.

0.6.14.3.27 IsArmed

```
bool Thorlabs.TSI.TLCameraInterfaces.ITLCamera.IsArmed [get]
```

Prior to issuing software or hardware triggers to get images from a camera, call Arm() to prepare it for imaging.

This property indicates whether Arm() has been called.

0.6.14.3.28 IsCoolingAdjustable

```
bool Thorlabs.TSI.TLCameraInterfaces.ITLCamera.IsCoolingAdjustable [get]
```

All Thorlabs scientific cameras are designed to efficiently dissipate heat. Some models additionally provide active cooling.

To determine whether camera has active-cooling hardware, use the IsCoolingSupported property.

To determine whether active cooling can be enabled or disabled through a software setting, use this property. If a camera supports active cooling, but this property returns false, then active cooling is enabled by plugging in the power cable (in addition to the USB cable).

See also

IsCoolingEnabled, IsCoolingSupported

0.6.14.3.29 IsCoolingEnabled

```
bool Thorlabs.TSI.TLCameraInterfaces.ITLCamera.IsCoolingEnabled [get], [set]
```

All Thorlabs scientific cameras are designed to efficiently dissipate heat. Some models additionally provide active cooling.

This property enables or disables active cooling for cameras that allow is to be changed through software.

To determine whether camera has active cooling hardware, use the IsCoolingSupported property.

To determine whether the active cooling can be enabled or disabled through software, use IsCoolingAdjustable.

See also

IsCoolingSupported, IsCoolingAdjustable

0.6.14.3.30 IsCoolingSupported

```
bool Thorlabs.TSI.TLCameraInterfaces.ITLCamera.IsCoolingSupported [get]
```

All Thorlabs scientific cameras are designed to efficiently dissipate heat. Some models additionally provide active cooling.

To determine whether camera has active-cooling hardware, use this property.

To determine whether active cooling can be enabled or disabled through a software setting, use IsCooling↩ Adjustable.

See also

IsCoolingEnabled, IsCoolingAdjustable

0.6.14.3.31 IsEEPSupported

```
bool Thorlabs.TSI.TLCameraInterfaces.ITLCamera.IsEEPSupported [get]
```

Equal Exposure Pulse (EEP) mode is an LVTTL-level signal that is active between the time when all rows have been reset during rolling reset, and the end of the exposure time (and the beginning of rolling readout). The signal can be used to control an external light source that will be triggered on only during the equal exposure period, providing the same amount of exposure for all pixels in the ROI.

This property determines whether the connected camera supports EEP mode.

See also

EEPStatus

0.6.14.3.32 IsFrameRateControlEnabled

```
bool Thorlabs.TSI.TLCameraInterfaces.ITLCamera.IsFrameRateControlEnabled [get], [set]
```

Some scientific cameras allow frame-rate control in the camera.

IsFrameRateControlEnabled determines whether frame-rate control is turned on or off the connected camera if it is supported.

If the connected camera supports FrameRateControlValue_fps, the FrameRateControlValueRange_fps will have a maximum greater than zero.

0.6.14.3.33 IsHotPixelCorrectionEnabled

```
bool Thorlabs.TSI.TLCameraInterfaces.ITLCamera.IsHotPixelCorrectionEnabled [get], [set]
```

Due to variability in manufacturing, some pixels have inherently higher dark current which manifests as abnormally bright pixels in images, typically visible with longer exposures. Hot-pixel correction identifies hot pixels and then substitutes a calculated value based on the values of neighboring pixels in place of hot pixels.

This property enables or disables hot-pixel correction.

If the connected camera supports hot-pixel correction, the threshold-range maximum will be greater than zero.

See also

HotPixelCorrectionThresholdRange, HotPixelCorrectionThreshold

0.6.14.3.34 IsLEDOn

```
bool Thorlabs.TSI.TLCameraInterfaces.ITLCamera.IsLEDOn [get], [set]
```

Some scientific cameras include an LED indicator light on the back panel.

This property gets the LED status or turns the light on or off.

See also

IsLEDSupported

0.6.14.3.35 IsLEDSupported

```
bool Thorlabs.TSI.TLCameraInterfaces.ITLCamera.IsLEDSupported [get]
```

Some scientific cameras include an LED indicator light on the back panel.

IsLEDSupported determines whether the connected camera has an LED indicator.

See also

IsLEDOn

0.6.14.3.36 IsNIRBoostEnabled

```
bool Thorlabs.TSI.TLCameraInterfaces.ITLCamera.IsNIRBoostEnabled [get], [set]
```

Some scientific-CCD camera models offer an enhanced near-infrared imaging mode for wavelengths in the 500 to 1000nm range. The Thorlabs website includes a helpful overview of this camera function on the Camera Basics page: https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=8962

This property enables or disables NIR-boost mode.

See also

IsNIRBoostSupported

0.6.14.3.37 IsNIRBoostSupported

`bool Thorlabs.TSI.TLCameraInterfaces.ITLCamera.IsNIRBoostSupported [get]`

Some scientific-CCD camera models offer an enhanced near-infrared imaging mode for wavelengths in the 500 to 1000nm range. The Thorlabs website includes a helpful overview of this camera function on the Camera Basics page: https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=8962

This property indicates whether the connected camera supports NIR boost.

See also

IsNIRBoostEnabled

0.6.14.3.38 IsTapBalanceEnabled

`bool Thorlabs.TSI.TLCameraInterfaces.ITLCamera.IsTapBalanceEnabled [get], [set]`

Scientific-CCD cameras support one, two, or four charge-readout channels (taps).

After exposure is complete, a CCD pixel array holds the charges corresponding to the amount of light collected at each pixel location. The data is then read out through one, two, or four channels at a time.

A higher number of taps usually results in a faster readout (all else being equal) although a faint "seam" may remain visible in the image even with tap-balance in effect.

Readout may not be perfectly balanced, but enabling on-camera tap balancing corrects the minor differences as much as possible.

To determine if the connected camera supports a particular taps mode, use GetIsTapsSupported().

See also

Taps, GetIsTapsSupported

0.6.14.3.39 MaximumNumberOfFramesToQueue

`int Thorlabs.TSI.TLCameraInterfaces.ITLCamera.MaximumNumberOfFramesToQueue [get], [set]`

By default, frames are not buffered. Thus, MaximumNumberOfFramesToQueue is set to one. If frames cannot be processed as quickly as they are coming in, then it might be necessary to increase the buffer size.

IMPORTANT NOTE: Setting MaximumNumberOfFramesToQueue to one (the default) instructs the camera SDK that the priority is to receive images in as close to realtime as possible. This can cause frames to get dropped if the system cannot keep up with the processing, but it ensures that the frames that are delivered are always the most recent.
If this is set to any value greater than one, the priority becomes avoiding dropping frames. More memory will be consumed, and it may introduce lag between the camera and the host application. Once the maximum number of frames to queue is reached, it will be forced to drop the oldest frame to make room for the new one.

To determine how many frames are currently queued, use the NumberOfQueuedFrames property.

See also

NumberOfQueuedFrames

0.6.14.3.40 Model

```
string Thorlabs.TSI.TLCameraInterfaces.ITLCamera.Model  [get]
```

Gets the camera model number such as 1501M or 8051C.

0.6.14.3.41 Name

```
string Thorlabs.TSI.TLCameraInterfaces.ITLCamera.Name  [get], [set]
```

Cameras can always be distinguished from each other by their serial numbers and/or model. A camera can also be named to distinguish between them. For example, if using a two-camera system, cameras may be named "Left" and "Right."

0.6.14.3.42 NumberOfQueuedFrames

```
int Thorlabs.TSI.TLCameraInterfaces.ITLCamera.NumberOfQueuedFrames  [get]
```

Gets the number of currently queued frames.

See also

MaximumNumberOfFramesToQueue

0.6.14.3.43 OperationMode

```
OperationMode Thorlabs.TSI.TLCameraInterfaces.ITLCamera.OperationMode  [get], [set]
```

The camera can operate in several distinct modes. The behavior of Arm() depends on the OperationMode.

In SoftwareTriggered operation mode, the camera only accepts triggers through IssueSoftwareTrigger().
In HardwareTriggered or Bulb operation modes, the camera only accepts triggers through the trigger input line.

Hardware trigger may be based on the trigger-input signal's rising or falling edge. This is determined by the TriggerPolarity property.

OperationMode can only be set when the camera is not armed.

To determine whether the connected camera supports a particular operation mode, use GetIsOperationMode↩ Supported.

See also

TriggerPolarity, Arm, GetIsOperationModeSupported

0.6.14.3.44 PolarPhase

```
PolarPhase Thorlabs.TSI.TLCameraInterfaces.ITLCamera.PolarPhase [get]
```

This describes how the polarization filter is aligned over the camera sensor. This property is only supported in polarized cameras.

In a polarized camera, each pixel is covered with one of four linear polarizers with orientations of -45°, 0°, 45°, or 90°. The polar phase represents the origin pixel on the sensor. To determine if a camera supports polarization, check the CameraSensorType property.

See also

CameraSensorType

0.6.14.3.45 ROIAndBin

```
ROIAndBin Thorlabs.TSI.TLCameraInterfaces.ITLCamera.ROIAndBin [get], [set]
```

By default, the region of interest (ROI) is the same as the sensor resolution. The region of interest can be reduced to a smaller rectangle in order to focus on an area smaller than a full- frame image. In some cases, reducing the ROI can increase the frame rate since less data needs to be transmitted to the host computer. Binning sums adjacent sensor pixels into "super pixels". It trades off spatial resolution for sensitivity and speed. For example, if a sensor is 1920 by 1080 pixels and binning is set to two in the X direction and two in the Y direction, the resulting image will be 960 by 540 pixels. Since smaller images require less data to be transmitted to the host computer, binning may increase the frame rate. By default, binning is set to one in both horizontal and vertical directions. It can be changed by setting ROIAndBin. It can be different in the X direction than the Y direction, and the available ranges vary by camera model.

To determine the available ROI ranges, use the GetROIRange.

Binning and ROIs must be set together to avoid invalid intermediate states that could result from setting them separately.

NOTE: Some camera models have restrictions on ROI parameters, and inputs may be nudged to the nearest valid values. Please read back ROI from camera after setting the ROI to get the actual camera values.

See also

BinXRange, BinYRange

0.6.14.3.46 SensorHeight_pixels

```
uint Thorlabs.TSI.TLCameraInterfaces.ITLCamera.SensorHeight_pixels [get]
```

This property provides the physical height of the camera sensor in pixels. It is equivalent to the ROI-width-range-maximum value.

See also

SensorWidth_pixels

0.6.14.3.47 SensorPixelHeight_um

```
float Thorlabs.TSI.TLCameraInterfaces.ITLCamera.SensorPixelHeight_um [get]
```

This property provides the physical height of a single light-sensitive photo site on the sensor.

0.6.14.3.48 SensorPixelSize_bytes

```
uint Thorlabs.TSI.TLCameraInterfaces.ITLCamera.SensorPixelSize_bytes [get]
```

This property return the sensor pixel size in bytes.

0.6.14.3.49 SensorPixelWidth_um

```
float Thorlabs.TSI.TLCameraInterfaces.ITLCamera.SensorPixelWidth_um [get]
```

This property provides the physical width of a single light-sensitive photo site on the sensor.

0.6.14.3.50 SensorReadoutTime_ns

```
uint Thorlabs.TSI.TLCameraInterfaces.ITLCamera.SensorReadoutTime_ns [get]
```

The time, in nanoseconds (ns), that readout data from image sensor.

0.6.14.3.51 SensorWidth_pixels

```
uint Thorlabs.TSI.TLCameraInterfaces.ITLCamera.SensorWidth_pixels [get]
```

This property provides the physical width of the camera sensor in pixels. This is equivalent to the ROI-height-range maximum value.

See also

SensorHeight_pixels

0.6.14.3.52 SerialNumber

```
string Thorlabs.TSI.TLCameraInterfaces.ITLCamera.SerialNumber [get]
```

This property gets the unique identifier for a camera.

0.6.14.3.53 Taps

```
Taps Thorlabs.TSI.TLCameraInterfaces.ITLCamera.Taps  [get], [set]
```

After exposure is complete, a CCD pixel array holds the charges corresponding to the amount of light collected at each pixel location. The data is then read out through one, two, or four channels at a time.

The number of taps can be selected only when the camera is disarmed.

0.6.14.3.54 TriggerPolarity

```
TriggerPolarity Thorlabs.TSI.TLCameraInterfaces.ITLCamera.TriggerPolarity  [get], [set]
```

When the OperationMode is set to HardwareTriggered or Bulb and then Arm() is called, the camera will respond to a trigger input as a signal to begin exposure. Setting TriggerPolarity tells the camera to begin exposure on either the rising edge or falling edge of the trigger signal.

0.6.14.4 Event Documentation

0.6.14.4.1 OnDiagnosticsMessage

```
OnDiagnosticsMessageDelegate Thorlabs.TSI.TLCameraInterfaces.ITLCamera.OnDiagnosticsMessage
```

Properties and methods may throw exceptions. However, during image acquisition, error information is available only by registering for the OnDiagnosticsMessage event. The event is usually raised on a worker thread.

The ITLCameraSDK.OpenCamera(string serialNumber, bool isDiagnosticsMessagesEnabled) method offers additional, verbose diagnostics information by passing 'true' as the second parameter. For example, if diagnostics messages are enabled, the SDK will report every camera parameter that is set.

0.6.14.4.2 OnImageFrameAvailable

```
OnImageFrameAvailableDelegate Thorlabs.TSI.TLCameraInterfaces.ITLCamera.OnImageFrameAvailable
```

ITLCamera offers two ways to get image data from the camera:

1. Poll for data using GetPendingFrameOrNull()
2. Register for the OnImageFrameAvailable event. The event is raised on a worker thread. In the event handler, use GetPendingFrameOrNull() to get the data.

   For more details, see the Getting Started section.

0.6.15 Thorlabs.TSI.TLCameraInterfaces.ITLCameraSDK Interface Reference

Inherits IDisposable.

Public Member Functions

- IList< string > DiscoverAvailableCameras ()
- string GetVersionInfo ()
- ITLCamera OpenCamera (string serialNumber, bool isDiagnosticsMessagesEnabled)

Events

- OnCameraConnectDelegate OnCameraConnect
- OnCameraDisconnectDelegate OnCameraDisconnect

0.6.15.1 Detailed Description

0.6.15.2 Member Function Documentation

0.6.15.2.1 DiscoverAvailableCameras()

```
IList<string> Thorlabs.TSI.TLCameraInterfaces.ITLCameraSDK.DiscoverAvailableCameras ( )
```

This property queries connected devices for GigE, CameraLink, and USB cameras, both scientific and compact scientific. This must be called before opening a camera. The operation requires several seconds to complete.

Returns

Serial number for each camera that can be provided to OpenCamera.

0.6.15.2.2 GetVersionInfo()

```
string Thorlabs.TSI.TLCameraInterfaces.ITLCameraSDK.GetVersionInfo ( )
```

Version information describes the camera SDK dynamic link libraries (DLLs). Typically, this information is displayed in an About dialog.

Returns

A multiple-line string of DLL version information.

0.6.15.2.3 OpenCamera()

```
ITLCamera Thorlabs.TSI.TLCameraInterfaces.ITLCameraSDK.OpenCamera (
            string serialNumber,
            bool isDiagnosticsMessagesEnabled )
```

Open the camera that has the given serial number. Cameras should not be opened more than once at the same time. First it is necessary to call DiscoverAvailableCameras before attempting to open them. Is is important to call Dispose() on the camera before exiting an application.

Parameters

serialNumber	A serial number returned by DiscoverAvailableCameras
isDiagnosticsMessagesEnabled	If set to true, the ITLCamera.OnDiagnosticsMessage event will be raised for every camera parameter that is modified.

Returns

Interface to a TSI camera.

0.6.15.3 Event Documentation

0.6.15.3.1 OnCameraConnect

`OnCameraConnectDelegate Thorlabs.TSI.TLCameraInterfaces.ITLCameraSDK.OnCameraConnect`

The OnCameraConnect event is invoked when a USB camera is plugged in while the application is running.

Registering for the event is optional, but can be useful to refresh the list of attached cameras.

0.6.15.3.2 OnCameraDisconnect

`OnCameraDisconnectDelegate Thorlabs.TSI.TLCameraInterfaces.ITLCameraSDK.OnCameraDisconnect`

The OnCameraConnect event is invoked when a USB camera is plugged in while the application is running.

Registering for the event is optional, but can be useful to refresh the list of attached cameras.

0.6.16 Thorlabs.TSI.TLCameraInterfaces.ROIAndBin Struct Reference

Public Attributes

- uint ROIOriginX_pixels
- uint ROIOriginY_pixels
- uint ROIWidth_pixels
- uint ROIHeight_pixels
- uint BinX
- uint BinY

0.6.16.1 Detailed Description

0.6.16.2 Member Data Documentation

0.6.16.2.1 BinX

```
uint Thorlabs.TSI.TLCameraInterfaces.ROIAndBin.BinX
```

0.6.16.2.2 BinY

```
uint Thorlabs.TSI.TLCameraInterfaces.ROIAndBin.BinY
```

0.6.16.2.3 ROIHeight_pixels

```
uint Thorlabs.TSI.TLCameraInterfaces.ROIAndBin.ROIHeight_pixels
```

0.6.16.2.4 ROIOriginX_pixels

```
uint Thorlabs.TSI.TLCameraInterfaces.ROIAndBin.ROIOriginX_pixels
```

A region of interest (ROI) is specified by a top-left X,Y coordinate, and a width and height. Binning is specified as independent X and Y values of one or greater.

0.6.16.2.5 ROIOriginY_pixels

```
uint Thorlabs.TSI.TLCameraInterfaces.ROIAndBin.ROIOriginY_pixels
```

0.6.16.2.6 ROIWidth_pixels

```
uint Thorlabs.TSI.TLCameraInterfaces.ROIAndBin.ROIWidth_pixels
```

0.6.17 Thorlabs.TSI.TLCamera.TLCameraSDK Class Reference

Static Public Member Functions

- static ITLCameraSDK OpenTLCameraSDK ()

0.6.17.1 Detailed Description

0.6.17.2 Member Function Documentation

0.6.17.2.1 OpenTLCameraSDK()

```
static ITLCameraSDK Thorlabs.TSI.TLCamera.TLCameraSDK.OpenTLCameraSDK ( )  [static]
```

Create an instance of ITLCameraSDK. There should only be one instance of a ITLCameraSDK object within an application. When all cameras are disposed, call ITLCameraSDK.Dispose().

Returns

An instance of ITLCameraSDK.

Index