PySpin API Reference

Release 3.1

FLIR Integrated Imaging Solutions, Inc

Apr 05, 2023

CONTENTS:

CHAPTER

ONE

INTRODUCTION

PySpin is a wrapper for FLIR Integrated Imaging Solutions' Spinnaker library.

FLIR Integrated Imaging Solutions' website is located at https://www.flir.com/iis/machine-vision.

The PySpin Python extension provides a common software interface to control and acquire images from FLIR USB 3.0, GigE, and USB 2.0 cameras using the same API.

CHAPTER

TWO

SOFTWARE LICENSING INFORMATION

Component	License
PySpin	Copyright (c) 2001-2023 FLIR Systems, Inc. All Rights Reserved. This software is the confidential and proprietary information of FLIR Integrated Imaging Solutions, Inc. (“Confidential Information”). You shall not disclose such Confidential Information and shall use it only in accordance with the terms of the license agreement you entered into with FLIR Integrated Imaging Solutions, Inc. (FLIR). FLIR MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THE SOFTWARE, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. FLIR SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.
GenICam	GenICam License http://www.emva.org/wp-content/uploads/GenICam_License_20140921.pdf
AdapterList	The Code Project Open License (CPOL) http://www.codeproject.com/info/cpol10.aspx
Boost	Boost Software License http://www.boost.org/users/license.html
FFMPEG	LGPv2.1 License https://www.ffmpeg.org/legal.html
FreeImage	FreeImage public license http://freeimage.sourceforge.net/freeimage-license.txt
Libusb	LGPLv2. License http://www.gnu.org/licenses/old-licenses/lgpl-2.1.txt
Libraw394	LGPLv2.0 License http://www.gnu.org/licenses/old-licenses/lgpl-2.0.txt
log4Net	Apache license 2.0 https://logging.apache.org/log4net/license.html
log4Cpp	LGPL License http://log4cpp.sourceforge.net/#license
Work with Bitmaps Faster in C#	The Code Project Open License (CPOL) 1.02 http://www.codeproject.com/info/cpol10.aspx
GUI ListView Improvements	WP:CC_BY-SA License https://goo.gl/a9l9yA

CHAPTER

THREE

EVENT CLASSES

3.1 PySpin.DeviceArrivalEventHandler

class PySpin.**DeviceArrivalEventHandler**

Proxy of C++ Spinnaker::DeviceArrivalEventHandler class.

OnDeviceArrival(*self*, *pCamera*)

Parameters
pCamera (*Spinnaker::CameraPtr*) –

property thisown

The membership flag

3.2 PySpin.DeviceEventHandler

class PySpin.**DeviceEventHandler**

Proxy of C++ Spinnaker::DeviceEventHandler class.

GetDeviceEventId(*self*) → uint64_t

GetDeviceEventName(*self*) → gcstring

OnDeviceEvent(*self*, *eventName*)

Parameters
eventName (*Spinnaker::GenICam::gcstring*) –

property thisown

The membership flag

3.3 PySpin.DeviceRemovalEventHandler

class PySpin.**DeviceRemovalEventHandler**

Proxy of C++ Spinnaker::DeviceRemovalEventHandler class.

OnDeviceRemoval(*self*, *pCamera*)

Parameters
pCamera (*Spinnaker::CameraPtr*) –

property thisown

The membership flag

3.4 PySpin.EventHandler

class PySpin.**EventHandler**(**args*, ***kwargs*)

Proxy of C++ Spinnaker::EventHandler class.

GetEventPayloadData(*self*) → PyObject *

GetEventPayloadDataSize(*self*) → size_t const

GetEventType(*self*) → Spinnaker::EventType

SetEventType(*self*, *eventType*)

Parameters
eventType (*enum Spinnaker::EventType*) –

property thisown

The membership flag

3.5 PySpin.ImageEventHandler

class PySpin.**ImageEventHandler**

Proxy of C++ Spinnaker::ImageEventHandler class.

OnImageEvent(*self*, *image*)

Parameters
image (*Spinnaker::ImagePtr*) –

property thisown

The membership flag

3.6 PySpin.ImageListEventHandler

class PySpin.**ImageListEventHandler**

Proxy of C++ Spinnaker::ImageListEventHandler class.

OnImageListEvent(*self*, *imageList*)

Parameters
imageList (*Spinnaker::ImageList*) –

property thisown

The membership flag

3.7 PySpin.InterfaceArrivalEventHandler

class PySpin.**InterfaceArrivalEventHandler**

Proxy of C++ Spinnaker::InterfaceArrivalEventHandler class.

OnInterfaceArrival(*self*, *pInterface*)

Parameters
pInterface (*Spinnaker::InterfacePtr*) –

property thisown

The membership flag

3.8 PySpin.InterfaceEventHandler

class PySpin.**InterfaceEventHandler**

Proxy of C++ Spinnaker::InterfaceEventHandler class.

OnDeviceArrival(*self*, *pCamera*)

Parameters
pCamera (*Spinnaker::CameraPtr*) –

OnDeviceRemoval(*self*, *pCamera*)

Parameters
pCamera (*Spinnaker::CameraPtr*) –

property thisown

The membership flag

3.9 PySpin.InterfaceRemovalEventHandler

class PySpin.**InterfaceRemovalEventHandler**

Proxy of C++ Spinnaker::InterfaceRemovalEventHandler class.

OnInterfaceRemoval(*self*, *pInterface*)

Parameters
pInterface (*Spinnaker::InterfacePtr*) –

property thisown

The membership flag

3.10 PySpin.LoggingEventHandler

class PySpin.**LoggingEventHandler**

Proxy of C++ Spinnaker::LoggingEventHandler class.

OnLogEvent(*self*, *eventPtr*)

Parameters
eventPtr (*Spinnaker::LoggingEventDataPtr*) –

property thisown

The membership flag

3.11 PySpin.LoggingEventDataPtr

class PySpin.**LoggingEventDataPtr**(**args*)

A reference tracked pointer to the LoggingEvent object.

C++ includes: LoggingEventDataPtr.h

property thisown

The membership flag

3.12 PySpin.SystemEventHandler

class PySpin.**SystemEventHandler**

Proxy of C++ Spinnaker::SystemEventHandler class.

OnInterfaceArrival(*self*, *pInterface*)

Parameters
pInterface (*Spinnaker::InterfacePtr*) –

OnInterfaceRemoval(*self*, *pInterface*)

Parameters
pInterface (*Spinnaker::InterfacePtr*) –

property thisown

The membership flag

CHAPTER

FOUR

PYSPIN CLASSES

4.1 PySpin.CBasePtr

class PySpin.**CBasePtr**(**args*)

Encapsulates a GenApi pointer dealing with the dynamic_cast automatically.

C++ includes: Pointer.h

GetAccessMode(*self*) → Spinnaker::GenApi::EAccessMode

IsValid(*self*) → bool

bool Spinnaker::GenApi::CPointer< T, B >::IsValid() const throw () true if the pointer is valid

property thisown

The membership flag

4.2 PySpin.Camera

class PySpin.Camera(*args, **kwargs)

The camera object class.

C++ includes: Camera.h

property AasRoiEnable

property AasRoiHeight

property AasRoiOffsetX

property AasRoiOffsetY

property AasRoiWidth

property AcquisitionAbort

property AcquisitionArm

property AcquisitionBurstFrameCount

property AcquisitionFrameCount

property AcquisitionFrameRate

property AcquisitionFrameRateEnable

property AcquisitionLineRate

property AcquisitionMode

property AcquisitionResultingFrameRate

property AcquisitionStart

property AcquisitionStatus

property AcquisitionStatusSelector

property AcquisitionStop

property ActionDeviceKey

property ActionGroupKey

property ActionGroupMask

property ActionQueueSize

property ActionSelector

property ActionUnconditionalMode

property AdaptiveCompressionEnable

property AdcBitDepth

property AutoAlgorithmSelector

property AutoExposureControlLoopDamping

property AutoExposureControlPriority

property AutoExposureEVCompensation

property AutoExposureExposureTimeLowerLimit

property AutoExposureExposureTimeUpperLimit

property AutoExposureGainLowerLimit

property AutoExposureGainUpperLimit

property AutoExposureGreyValueLowerLimit

property AutoExposureGreyValueUpperLimit

property AutoExposureLightingMode

property AutoExposureMeteringMode

property AutoExposureTargetGreyValue

property AutoExposureTargetGreyValueAuto

property BalanceRatio

property BalanceRatioSelector

property BalanceWhiteAuto

property BalanceWhiteAutoDamping

property BalanceWhiteAutoLowerLimit

property BalanceWhiteAutoProfile

property BalanceWhiteAutoUpperLimit

property BinningHorizontal

property BinningHorizontalMode

property BinningSelector

property BinningVertical

property BinningVerticalMode

property BlackLevel

property BlackLevelAuto

property BlackLevelAutoBalance

property BlackLevelClampingEnable

property BlackLevelRaw

property BlackLevelSelector

property ChunkBlackLevel

property ChunkBlackLevelSelector

property ChunkCRC

property ChunkCompressionMode

property ChunkCompressionRatio

property ChunkCounterSelector

property ChunkCounterValue

property ChunkEnable

property ChunkEncoderSelector

property ChunkEncoderStatus

property ChunkEncoderValue

property ChunkExposureEndLineStatusAll

property ChunkExposureTime

property ChunkExposureTimeSelector

property ChunkFrameID

property ChunkGain

property ChunkGainSelector

property ChunkHeight

property ChunkImage

property ChunkImageComponent

property ChunkInferenceBoundingBoxResult

property ChunkInferenceConfidence

property ChunkInferenceFrameId

property ChunkInferenceResult

property ChunkLinePitch

property ChunkLineStatusAll

property ChunkModeActive

property ChunkOffsetX

property ChunkOffsetY

property ChunkPartSelector

property ChunkPixelDynamicRangeMax

property ChunkPixelDynamicRangeMin

property ChunkPixelFormat

property ChunkRegionID

property ChunkScan3dAxisMax

property ChunkScan3dAxisMin

property ChunkScan3dCoordinateOffset

property ChunkScan3dCoordinateReferenceSelector

property ChunkScan3dCoordinateReferenceValue

property ChunkScan3dCoordinateScale

property ChunkScan3dCoordinateSelector

property ChunkScan3dCoordinateSystem

property ChunkScan3dCoordinateSystemReference

property ChunkScan3dCoordinateTransformSelector

property ChunkScan3dDistanceUnit

property ChunkScan3dInvalidDataFlag

property ChunkScan3dInvalidDataValue

property ChunkScan3dOutputMode

property ChunkScan3dTransformValue

property ChunkScanLineSelector

property ChunkSelector

property ChunkSequencerSetActive

property ChunkSerialData

property ChunkSerialDataLength

property ChunkSerialReceiveOverflow

property ChunkSourceID

property ChunkStreamChannelID

property ChunkTimerSelector

property ChunkTimerValue

property ChunkTimestamp

property ChunkTimestampLatchValue

property ChunkTransferBlockID

property ChunkTransferQueueCurrentBlockCount

property ChunkTransferStreamID

property ChunkWidth

property ClConfiguration

property ClTimeSlotsCount

property ColorTransformationEnable

property ColorTransformationSelector

property ColorTransformationValue

property ColorTransformationValueSelector

property CompressionRatio

property CompressionSaturationPriority

property CounterDelay

property CounterDuration

property CounterEventActivation

property CounterEventSource

property CounterReset

property CounterResetActivation

property CounterResetSource

property CounterSelector

property CounterStatus

property CounterTriggerActivation

property CounterTriggerSource

property CounterValue

property CounterValueAtReset

property CxpConnectionSelector

property CxpConnectionTestErrorCount

property CxpConnectionTestMode

property CxpConnectionTestPacketCount

property CxpLinkConfiguration

property CxpLinkConfigurationPreferred

property CxpLinkConfigurationStatus

property CxpPoCxpAuto

property CxpPoCxpStatus

property CxpPoCxpTripReset

property CxpPoCxpTurnOff

property DecimationHorizontal

property DecimationHorizontalMode

property DecimationSelector

property DecimationVertical

property DecimationVerticalMode

property DefectCorrectStaticEnable

property DefectCorrectionMode

property DefectTableApply

property DefectTableCoordinateX

property DefectTableCoordinateY

property DefectTableFactoryRestore

property DefectTableIndex

property DefectTablePixelCount

property DefectTableSave

property Deinterlacing

property DeviceCharacterSet

property DeviceClockFrequency

property DeviceClockSelector

property DeviceConnectionSelector

property DeviceConnectionSpeed

property DeviceConnectionStatus

property DeviceEventChannelCount

property DeviceFamilyName

property DeviceFeaturePersistenceEnd

property DeviceFeaturePersistenceStart

property DeviceFirmwareVersion

property DeviceGenCPVersionMajor

property DeviceGenCPVersionMinor

property DeviceID

property DeviceIndicatorMode

property DeviceLinkBandwidthReserve

property DeviceLinkCommandTimeout

property DeviceLinkConnectionCount

property DeviceLinkCurrentThroughput

property DeviceLinkHeartbeatMode

property DeviceLinkHeartbeatTimeout

property DeviceLinkSelector

property DeviceLinkSpeed

property DeviceLinkThroughputLimit

property DeviceLinkThroughputLimitMode

property DeviceManifestEntrySelector

property DeviceManifestPrimaryURL

property DeviceManifestSchemaMajorVersion

property DeviceManifestSchemaMinorVersion

property DeviceManifestSecondaryURL

property DeviceManifestXMLMajorVersion

property DeviceManifestXMLMinorVersion

property DeviceManifestXMLSubMinorVersion

property DeviceManufacturerInfo

property DeviceMaxThroughput

property DeviceModelName

property DevicePowerSupplySelector

property DeviceRegistersCheck

property DeviceRegistersEndianness

property DeviceRegistersStreamingEnd

property DeviceRegistersStreamingStart

property DeviceRegistersValid

property DeviceReset

property DeviceSFNCVersionMajor

property DeviceSFNCVersionMinor

property DeviceSFNCVersionSubMinor

property DeviceScanType

property DeviceSerialNumber

property DeviceSerialPortBaudRate

property DeviceSerialPortSelector

property DeviceStreamChannelCount

property DeviceStreamChannelEndianness

property DeviceStreamChannelLink

property DeviceStreamChannelPacketSize

property DeviceStreamChannelSelector

property DeviceStreamChannelType

property DeviceTLType

property DeviceTLVersionMajor

property DeviceTLVersionMinor

property DeviceTLVersionSubMinor

property DeviceTapGeometry

property DeviceTemperature

property DeviceTemperatureSelector

property DeviceType

property DeviceUptime

property DeviceUserID

property DeviceVendorName

property DeviceVersion

property EncoderDivider

property EncoderMode

property EncoderOutputMode

property EncoderReset

property EncoderResetActivation

property EncoderResetSource

property EncoderSelector

property EncoderSourceA

property EncoderSourceB

property EncoderStatus

property EncoderTimeout

property EncoderValue

property EncoderValueAtReset

property EnumerationCount

property EventAcquisitionEnd

property EventAcquisitionEndFrameID

property EventAcquisitionEndTimestamp

property EventAcquisitionError

property EventAcquisitionErrorFrameID

property EventAcquisitionErrorTimestamp

property EventAcquisitionStart

property EventAcquisitionStartFrameID

property EventAcquisitionStartTimestamp

property EventAcquisitionTransferEnd

property EventAcquisitionTransferEndFrameID

property EventAcquisitionTransferEndTimestamp

property EventAcquisitionTransferStart

property EventAcquisitionTransferStartFrameID

property EventAcquisitionTransferStartTimestamp

property EventAcquisitionTrigger

property EventAcquisitionTriggerFrameID

property EventAcquisitionTriggerTimestamp

property EventActionLate

property EventActionLateFrameID

property EventActionLateTimestamp

property EventCounter0End

property EventCounter0EndFrameID

property EventCounter0EndTimestamp

property EventCounter0Start

property EventCounter0StartFrameID

property EventCounter0StartTimestamp

property EventCounter1End

property EventCounter1EndFrameID

property EventCounter1EndTimestamp

property EventCounter1Start

property EventCounter1StartFrameID

property EventCounter1StartTimestamp

property EventEncoder0Restarted

property EventEncoder0RestartedFrameID

property EventEncoder0RestartedTimestamp

property EventEncoder0Stopped

property EventEncoder0StoppedFrameID

property EventEncoder0StoppedTimestamp

property EventEncoder1Restarted

property EventEncoder1RestartedFrameID

property EventEncoder1RestartedTimestamp

property EventEncoder1Stopped

property EventEncoder1StoppedFrameID

property EventEncoder1StoppedTimestamp

property EventError

property EventErrorCode

property EventErrorFrameID

property EventErrorTimestamp

property EventExposureEnd

property EventExposureEndFrameID

property EventExposureEndTimestamp

property EventExposureStart

property EventExposureStartFrameID

property EventExposureStartTimestamp

property EventFrameBurstEnd

property EventFrameBurstEndFrameID

property EventFrameBurstEndTimestamp

property EventFrameBurstStart

property EventFrameBurstStartFrameID

property EventFrameBurstStartTimestamp

property EventFrameEnd

property EventFrameEndFrameID

property EventFrameEndTimestamp

property EventFrameStart

property EventFrameStartFrameID

property EventFrameStartTimestamp

property EventFrameTransferEnd

property EventFrameTransferEndFrameID

property EventFrameTransferEndTimestamp

property EventFrameTransferStart

property EventFrameTransferStartFrameID

property EventFrameTransferStartTimestamp

property EventFrameTrigger

property EventFrameTriggerFrameID

property EventFrameTriggerTimestamp

property EventLine0AnyEdge

property EventLine0AnyEdgeFrameID

property EventLine0AnyEdgeTimestamp

property EventLine0FallingEdge

property EventLine0FallingEdgeFrameID

property EventLine0FallingEdgeTimestamp

property EventLine0RisingEdge

property EventLine0RisingEdgeFrameID

property EventLine0RisingEdgeTimestamp

property EventLine1AnyEdge

property EventLine1AnyEdgeFrameID

property EventLine1AnyEdgeTimestamp

property EventLine1FallingEdge

property EventLine1FallingEdgeFrameID

property EventLine1FallingEdgeTimestamp

property EventLine1RisingEdge

property EventLine1RisingEdgeFrameID

property EventLine1RisingEdgeTimestamp

property EventLinkSpeedChange

property EventLinkSpeedChangeFrameID

property EventLinkSpeedChangeTimestamp

property EventLinkTrigger0

property EventLinkTrigger0FrameID

property EventLinkTrigger0Timestamp

property EventLinkTrigger1

property EventLinkTrigger1FrameID

property EventLinkTrigger1Timestamp

property EventNotification

property EventSelector

property EventSequencerSetChange

property EventSequencerSetChangeFrameID

property EventSequencerSetChangeTimestamp

property EventSerialData

property EventSerialDataLength

property EventSerialPortReceive

property EventSerialPortReceiveTimestamp

property EventSerialReceiveOverflow

property EventStream0TransferBlockEnd

property EventStream0TransferBlockEndFrameID

property EventStream0TransferBlockEndTimestamp

property EventStream0TransferBlockStart

property EventStream0TransferBlockStartFrameID

property EventStream0TransferBlockStartTimestamp

property EventStream0TransferBlockTrigger

property EventStream0TransferBlockTriggerFrameID

property EventStream0TransferBlockTriggerTimestamp

property EventStream0TransferBurstEnd

property EventStream0TransferBurstEndFrameID

property EventStream0TransferBurstEndTimestamp

property EventStream0TransferBurstStart

property EventStream0TransferBurstStartFrameID

property EventStream0TransferBurstStartTimestamp

property EventStream0TransferEnd

property EventStream0TransferEndFrameID

property EventStream0TransferEndTimestamp

property EventStream0TransferOverflow

property EventStream0TransferOverflowFrameID

property EventStream0TransferOverflowTimestamp

property EventStream0TransferPause

property EventStream0TransferPauseFrameID

property EventStream0TransferPauseTimestamp

property EventStream0TransferResume

property EventStream0TransferResumeFrameID

property EventStream0TransferResumeTimestamp

property EventStream0TransferStart

property EventStream0TransferStartFrameID

property EventStream0TransferStartTimestamp

property EventTest

property EventTestTimestamp

property EventTimer0End

property EventTimer0EndFrameID

property EventTimer0EndTimestamp

property EventTimer0Start

property EventTimer0StartFrameID

property EventTimer0StartTimestamp

property EventTimer1End

property EventTimer1EndFrameID

property EventTimer1EndTimestamp

property EventTimer1Start

property EventTimer1StartFrameID

property EventTimer1StartTimestamp

property ExposureActiveMode

property ExposureAuto

property ExposureMode

property ExposureTime

property ExposureTimeMode

property ExposureTimeSelector

property FactoryReset

property FileAccessBuffer

property FileAccessLength

property FileAccessOffset

property FileOpenMode

property FileOperationExecute

property FileOperationResult

property FileOperationSelector

property FileOperationStatus

property FileSelector

property FileSize

property Gain

property GainAuto

property GainAutoBalance

property GainSelector

property Gamma

property GammaEnable

property GevActiveLinkCount

property GevCCP

property GevCurrentDefaultGateway

property GevCurrentIPAddress

property GevCurrentIPConfigurationDHCP

property GevCurrentIPConfigurationLLA

property GevCurrentIPConfigurationPersistentIP

property GevCurrentPhysicalLinkConfiguration

property GevCurrentSubnetMask

property GevDiscoveryAckDelay

property GevFirstURL

property GevGVCPExtendedStatusCodes

property GevGVCPExtendedStatusCodesSelector

property GevGVCPHeartbeatDisable

property GevGVCPPendingAck

property GevGVCPPendingTimeout

property GevGVSPExtendedIDMode

property GevHeartbeatTimeout

property GevIEEE1588

property GevIEEE1588ClockAccuracy

property GevIEEE1588Mode

property GevIEEE1588Status

property GevIPConfigurationStatus

property GevInterfaceSelector

property GevMACAddress

property GevMCDA

property GevMCPHostPort

property GevMCRC

property GevMCSP

property GevMCTT

property GevNumberOfInterfaces

property GevPAUSEFrameReception

property GevPAUSEFrameTransmission

property GevPersistentDefaultGateway

property GevPersistentIPAddress

property GevPersistentSubnetMask

property GevPhysicalLinkConfiguration

property GevPrimaryApplicationIPAddress

property GevPrimaryApplicationSocket

property GevPrimaryApplicationSwitchoverKey

property GevSCCFGAllInTransmission

property GevSCCFGExtendedChunkData

property GevSCCFGPacketResendDestination

property GevSCCFGUnconditionalStreaming

property GevSCDA

property GevSCPD

property GevSCPDirection

property GevSCPHostPort

property GevSCPInterfaceIndex

property GevSCPSBigEndian

property GevSCPSDoNotFragment

property GevSCPSFireTestPacket

property GevSCPSPacketSize

property GevSCSP

property GevSCZoneConfigurationLock

property GevSCZoneCount

property GevSCZoneDirectionAll

property GevSecondURL

property GevStreamChannelSelector

property GevSupportedOption

property GevSupportedOptionSelector

property GevTimestampTickFrequency

property GuiXmlManifestAddress

property Height

property HeightMax

property ImageComponentEnable

property ImageComponentSelector

property ImageCompressionBitrate

property ImageCompressionJPEGFormatOption

property ImageCompressionMode

property ImageCompressionQuality

property ImageCompressionRateOption

Init(*self*)

void Spinnaker::Camera::Init()

property IspEnable

property LUTEnable

property LUTIndex

property LUTSelector

property LUTValue

property LUTValueAll

property LineFilterWidth

property LineFormat

property LineInputFilterSelector

property LineInverter

property LineMode

property LinePitch

property LineSelector

property LineSource

property LineStatus

property LineStatusAll

property LinkErrorCount

property LinkUptime

property LogicBlockLUTInputActivation

property LogicBlockLUTInputSelector

property LogicBlockLUTInputSource

property LogicBlockLUTOutputValue

property LogicBlockLUTOutputValueAll

property LogicBlockLUTRowIndex

property LogicBlockLUTSelector

property LogicBlockSelector

property MaxDeviceResetTime

property OffsetX

property OffsetY

property PacketResendRequestCount

property PayloadSize

property PixelColorFilter

property PixelDynamicRangeMax

property PixelDynamicRangeMin

property PixelFormat

property PixelFormatInfoID

property PixelFormatInfoSelector

property PixelSize

property PowerSupplyCurrent

property PowerSupplyVoltage

property RegionDestination

property RegionMode

property RegionSelector

property ReverseX

property ReverseY

property RgbTransformLightSource

property Saturation

property SaturationEnable

property Scan3dAxisMax

property Scan3dAxisMin

property Scan3dCoordinateOffset

property Scan3dCoordinateReferenceSelector

property Scan3dCoordinateReferenceValue

property Scan3dCoordinateScale

property Scan3dCoordinateSelector

property Scan3dCoordinateSystem

property Scan3dCoordinateSystemReference

property Scan3dCoordinateTransformSelector

property Scan3dDistanceUnit

property Scan3dInvalidDataFlag

property Scan3dInvalidDataValue

property Scan3dOutputMode

property Scan3dTransformValue

property SensorDescription

property SensorDigitizationTaps

property SensorHeight

property SensorShutterMode

property SensorTaps

property SensorWidth

property SequencerConfigurationMode

property SequencerConfigurationValid

property SequencerFeatureEnable

property SequencerMode

property SequencerPathSelector

property SequencerSetActive

property SequencerSetLoad

property SequencerSetNext

property SequencerSetSave

property SequencerSetSelector

property SequencerSetStart

property SequencerSetValid

property SequencerTriggerActivation

property SequencerTriggerSource

property SerialPortBaudRate

property SerialPortDataBits

property SerialPortParity

property SerialPortSelector

property SerialPortSource

property SerialPortStopBits

property SerialReceiveFramingErrorCount

property SerialReceiveParityErrorCount

property SerialReceiveQueueClear

property SerialReceiveQueueCurrentCharacterCount

property SerialReceiveQueueMaxCharacterCount

property SerialTransmitQueueCurrentCharacterCount

property SerialTransmitQueueMaxCharacterCount

property Sharpening

property SharpeningAuto

property SharpeningEnable

property SharpeningThreshold

property SoftwareSignalPulse

property SoftwareSignalSelector

property SourceCount

property SourceSelector

property TLParamsLocked

property Test0001

property TestEventGenerate

property TestPattern

property TestPatternGeneratorSelector

property TestPendingAck

property TimerDelay

property TimerDuration

property TimerReset

property TimerSelector

property TimerStatus

property TimerTriggerActivation

property TimerTriggerSource

property TimerValue

property Timestamp

property TimestampLatch

property TimestampLatchValue

property TimestampReset

property TransferAbort

property TransferBlockCount

property TransferBurstCount

property TransferComponentSelector

property TransferControlMode

property TransferOperationMode

property TransferPause

property TransferQueueCurrentBlockCount

property TransferQueueMaxBlockCount

property TransferQueueMode

property TransferQueueOverflowCount

property TransferResume

property TransferSelector

property TransferStart

property TransferStatus

property TransferStatusSelector

property TransferStop

property TransferStreamChannel

property TransferTriggerActivation

property TransferTriggerMode

property TransferTriggerSelector

property TransferTriggerSource

property TriggerActivation

property TriggerDelay

property TriggerDivider

property TriggerEventTest

property TriggerMode

property TriggerMultiplier

property TriggerOverlap

property TriggerSelector

property TriggerSoftware

property TriggerSource

property UserOutputSelector

property UserOutputValue

property UserOutputValueAll

property UserOutputValueAllMask

property UserSetDefault

property UserSetFeatureEnable

property UserSetLoad

property UserSetSave

property UserSetSelector

property V3_3Enable

property WhiteClip

property WhiteClipSelector

property Width

property WidthMax

property aPAUSEMACCtrlFramesReceived

property aPAUSEMACCtrlFramesTransmitted

property thisown

The membership flag

4.3 PySpin.CameraBase

class PySpin.**CameraBase**(**args*, ***kwargs*)

The base class for the camera object.

C++ includes: CameraBase.h

BeginAcquisition(*self*)

void Spinnaker::CameraBase::BeginAcquisition()

Starts the image acquisition engine. The camera must be initialized via a call to Init() before starting an acquisition.

See: Init()

DeInit(*self*)

void Spinnaker::CameraBase::DeInit()

Disconnect camera port and free GenICam node map and GUI XML. Do not call more functions that access the remote device such as WritePort/ReadPort after calling DeInit(); Events should also be unregistered before calling camera DeInit(). Otherwise an exception will be thrown in the DeInit() call and require the user to unregister events before the camera can be re-initialized again.

See: Init()

See: UnregisterEvent(Event & evtToUnregister)

DiscoverMaxPacketSize(*self*) → unsigned int

unsigned int Spinnaker::CameraBase::DiscoverMaxPacketSize()

Returns the largest packet size that can be safely used on the interface that device is connected to

The maximum packet size returned.

EndAcquisition(*self*)

void Spinnaker::CameraBase::EndAcquisition()

Stops the image acquisition engine. If EndAcquisition() is called without a prior call to BeginAcquisition() an error message "Camera is not started" will be thrown. All Images that were acquired using GetNextImage() need to be released first using image->Release() before calling EndAcquisition(). All buffers in the input pool and output queue will be discarded when EndAcquisition() is called.

See: Init()

See: BeginAcquisition()

See: GetNextImage(grabTimeout)

See: Image::Release()

ForceIP(*self*)

GetAccessMode(*self*) → Spinnaker::GenApi::EAccessMode

GenApi::EAccessMode Spinnaker::CameraBase::GetAccessMode() const

Returns the access mode that the software has on the Camera. The camera does not need to be initialized before calling this function.

See: Init()

An enumeration value indicating the access mode

GetBufferOwnership(*self*) → Spinnaker::BufferOwnership

GetGuiXml(*self*) → gcstring

GenICam::gcstring Spinnaker::CameraBase::GetGuiXml() const

Returns the GUI XML that can be passed into the Spinnaker GUI framework

GenICam::gcstring that represents the uncompressed GUI XML file

GetNextImage(*self*, *grabTimeout=EVENT_TIMEOUT_INFINITE*, *streamIndex=0*) → *ImagePtr*

Parameters

- **grabTimeout** (*a 64bit value that represents a timeout in milliseconds*) –
- **streamIndex** (*uint64_t*) –
- **ImagePtr** –
- **Spinnaker::CameraBase::GetNextImage(uint64_t** –
- **grabTimeout=EVENT_TIMEOUT_INFINITE** –
- **streamID=0)** (*uint64_t*) –
- **This** (*Gets the next image that was received by the transport layer.*) –
- **cameras** (*function will block indefinitely until an image arrives. Most*) –
- **camera** (*support one stream so the default streamID is 0 but if a*) –
- **select** (*supports multiple streams the user can input the streamID to*) –
- **images** (*from which stream to grab*) –
- **See** (*EndAcquisition()*) –
- **See** –
- **See** –
- **Parameters** –
- **----------** –
- **grabTimeout** –
- **streamID** (*The stream to grab the image.*) –
- **object** (*pointer to an Image*) –

GetNextImageSync(*self*, *grabTimeout=EVENT_TIMEOUT_INFINITE*) → *ImageList*

Parameters
grabTimeout (*uint64_t*) –

GetNodeMap(*self*) → INodeMap

GenApi::INodeMap& Spinnaker::CameraBase::GetNodeMap() const

Gets a reference to the node map that is generated from a GenICam XML file. The camera must be initialized by a call to Init() first before a node map reference can be successfully acquired.

See: Init()

A reference to the INodeMap.

GetNumDataStreams(*self*) → unsigned int

unsigned int Spinnaker::CameraBase::GetNumDataStreams()

Returns the number of streams that a device supports.

The number of data streams

GetNumImagesInUse(*self*) → unsigned int

unsigned int Spinnaker::CameraBase::GetNumImagesInUse()

Returns the number of images that are currently in use. Each of the images that are currently in use must be cleaned up with a call to image->Release() before calling system->ReleaseInstance().

The number of images that needs to be cleaned up.

GetTLDeviceNodeMap(*self*) → INodeMap

GenApi::INodeMap& Spinnaker::CameraBase::GetTLDeviceNodeMap() const

Gets a reference to the node map that is generated from a GenICam XML file for the GenTL Device module. The camera does not need to be initialized before acquiring this node map.

A reference to the INodeMap.

GetTLStreamNodeMap(*self*, *streamIndex=0*) → INodeMap

Parameters

- **streamIndex** (*uint64_t*) –
- **const** (*GenApi::INodeMap& Spinnaker::CameraBase::GetTLStreamNodeMap()*) –
- **XML** (*Gets a reference to the node map that is generated from a GenICam*) –
- **be** (*file for the GenTL Stream module. The camera does not need to*) –
- **map.** (*initialized before acquiring this node*) –
- **INodeMap.** (*A reference to the*) –

GetUniqueID(*self*) → gcstring

GenICam::gcstring Spinnaker::CameraBase::GetUniqueID()

This returns a unique id string that identifies the camera. This is the camera serial number.

string that uniquely identifies the camera (serial number)

GetUserBufferCount(*self*) → uint64_t

GetUserBufferSize(*self*) → uint64_t

GetUserBufferTotalSize(*self*) → uint64_t

Init(*self*)

void Spinnaker::CameraBase::Init()

Connect to camera, retrieve XML and generate node map. This function needs to be called before any camera related API calls such as BeginAcquisition(), EndAcquisition(), GetNodeMap(), GetNextImage().

See: BeginAcquisition()

See: EndAcquisition()

See: GetNodeMap()

See: GetNextImage()

IsInitialized(*self*) → bool

bool Spinnaker::CameraBase::IsInitialized()

Checks if camera is initialized. This function needs to return true in order to retrieve a valid NodeMap from the GetNodeMap() call.

See: GetNodeMap()

If camera is initialized or not

IsStreaming(*self*) → bool

bool Spinnaker::CameraBase::IsStreaming() const

Returns true if the camera is currently streaming or false if it is not.

See: Init()

returns true if camera is streaming and false otherwise.

IsValid(*self*) → bool

bool Spinnaker::CameraBase::IsValid()

Checks a flag to determine if camera is still valid for use.

If camera is valid or not

Note that CameraPtr and CameraBase both define an IsValid() function. In order to determine the validity of the camera using a CameraPtr, user must first call get() to retrieve the CameraBase object.

RegisterEventHandler(*self*, *evtHandlerToRegister*)

Parameters

- **evtHandlerToRegister** (*Spinnaker::ImageEventHandler &*) –
- **RegisterEventHandler(self** –
- **evtHandlerToRegister** –
- **eventName)** –
- **evtHandlerToRegister** –
- **eventName** (*Spinnaker::GenICam::gcstring const &*) –
- **RegisterEventHandler(self** –
- **evtHandlerToRegister** –

- **streamIndex)** –
- **evtHandlerToRegister** –
- **streamIndex** (*uint64_t*) –

SetBufferOwnership(*self*, *mode*)

Parameters
mode (*enum Spinnaker::BufferOwnership const*) –

SetUserBuffers(*self*, *pMemBuffers*, *totalSize*)

Parameters

- **pMemBuffers** (void *const) –
- **totalSize** (*uint64_t*) –
- **SetUserBuffers(self** –
- **ppMemBuffers** (void **const) –
- **bufferCount** (*uint64_t const*) –
- **bufferSize)** –
- **ppMemBuffers** –
- **bufferCount** –
- **bufferSize** (*uint64_t const*) –

UnregisterEventHandler(*self*, *evtHandlerToUnregister*)

Parameters
evtHandlerToUnregister (*Spinnaker::EventHandler &*) –

property thisown

The membership flag

4.4 PySpin.CameraList

class PySpin.**CameraList**(**args*)

Used to hold a list of camera objects.

C++ includes: CameraList.h

Add(*self*, *camera*)

Parameters
camera (*Spinnaker::CameraPtr*) –

Append(*self*, *list*)

Parameters

- **list** (*Spinnaker::CameraList const &*) –
- **void** –
- **&otherList)** (*Spinnaker::CameraList::Append(*CameraList) –
- **list.** (*Appends a camera list to the current*) –

- **Parameters** –
- ---------- –
- **otherList** (*The other list to append to this list*) –

Clear(*self*)

void Spinnaker::CameraList::Clear()

Clears the list of cameras and destroys their corresponding reference counted objects. This is necessary in order to clean up the parent interface. It is important that the camera list is destroyed or is cleared before calling system->ReleaseInstance() or else the call to system->ReleaseInstance() will result in an error message thrown that a reference to the camera is still held.

See: System:ReleaseInstance()

GetByDeviceID(*self*, *deviceID*) → *CameraPtr*

Parameters
deviceID (*std::string*) –

GetByIndex(*self*, *index*) → *CameraPtr*

Parameters

- **index** (*The index at which to retrieve the camera object*) –
- **CameraPtr** –
- **const** (*Spinnaker::CameraList::GetByIndex(int index)*) –
- **"index".** (*Returns a pointer to a camera object at the*) –
- **Parameters** –
- ---------- –
- **index** –
- **object.** (*A pointer to an camera*) –

GetBySerial(*self*, *serialNumber*) → *CameraPtr*

Parameters

- **serialNumber** (*The serial number of the camera object to retrieve*) –
- **CameraPtr** –
- **const** (*Spinnaker::CameraList::GetBySerial(std::string serialNumber)*) –
- **number.** (*Returns a pointer to a camera object with the specified serial*) –
- **Parameters** –
- ---------- –
- **serialNumber** –
- **object.** (*A pointer to an camera*) –

GetSize(*self*) → unsigned int

int Spinnaker::CameraList::GetSize() const

Returns the size of the camera list. The size is the number of Camera objects stored in the list.

An integer that represents the list size.

Remove(*self*, *camera*)

Parameters
camera (*Spinnaker::CameraPtr*) –

RemoveByDeviceID(*self*, *deviceID*)

Parameters
deviceID (*std::string*) –

RemoveByIndex(*self*, *index*)

Parameters

- **index** (*The index at which to remove the Camera object*) –
- **void** –
- **index)** (*Spinnaker::CameraList::RemoveByIndex(int*) –
- **reference** (*Removes a camera at "index" and destroys its corresponding*) –
- **object.** (*counted*) –
- **Parameters** –
- **----------** –
- **index** –

RemoveBySerial(*self*, *serialNumber*)

Parameters

- **serialNumber** (*The serial number of the Camera object to remove*) –
- **void** –
- **serialNumber)** (*Spinnaker::CameraList::RemoveBySerial(std::string*) –
- **its** (*Removes a camera using its serial number and destroys*) –
- **object.** (*corresponding reference counted*) –
- **Parameters** –
- **----------** –
- **serialNumber** –

property thisown
The membership flag

4.5 PySpin.CameraPtr

class PySpin.**CameraPtr**(**args*)
A reference tracked pointer to a camera object.

C++ includes: CameraPtr.h

property thisown
The membership flag

4.6 PySpin.ChannelStatistics

class PySpin.**ChannelStatistics**(*image*, *channel*)

Class used to store statistics (as properties) for one channel of an image. Properties:

- channel: The image channel that the statistics are based on (as an int).
- range_min: The smallest possible pixel value.
- range_max: The largest possible pixel value.
- pixel_value_min: The smallest pixel value in the current channel.
- pixel_value_max: The largest pixel value in the current channel.
- num_pixel_values: The total number of pixel values in the current channel.
- pixel_value_mean: The average pixel value in the current channel.
- histogram: NumPy array representing the histogram of the current channel.

property channel

property histogram

property num_pixel_values

property pixel_value_max

property pixel_value_mean

property pixel_value_min

property range_max

property range_min

property thisown

The membership flag

4.7 PySpin.ChunkData

class PySpin.**ChunkData**(**args*)

The chunk data which contains additional information about an image.

C++ includes: ChunkData.h

GetBlackLevel(*self*) → float64_t

float64_t Spinnaker::ChunkData::GetBlackLevel() const

Description: Returns the black level used to capture the image included in the payload. Visibility: Expert

GetCRC(*self*) → int64_t

GetCompressionMode(*self*) → int64_t

GetCompressionRatio(*self*) → float64_t

GetCounterValue(*self*) → int64_t

int64_t Spinnaker::ChunkData::GetCounterValue() const

Description: Returns the value of the selected Chunk counter at the time of the FrameStart event. Visibility: Expert

GetEncoderValue(*self*) → int64_t

int64_t Spinnaker::ChunkData::GetEncoderValue() const

Description: Returns the counter's value of the selected Encoder at the time of the FrameStart in area scan mode or the counter's value at the time of the LineStart selected by ChunkScanLineSelector in LineScan mode. Visibility: Expert

GetExposureEndLineStatusAll(*self*) → int64_t

GetExposureTime(*self*) → float64_t

float64_t Spinnaker::ChunkData::GetExposureTime() const

Description: Returns the exposure time used to capture the image. Visibility: Expert

GetFrameID(*self*) → int64_t

int64_t Spinnaker::ChunkData::GetFrameID() const

Description: Returns the unique Identifier of the frame (or image) included in the payload. Visibility: Expert

GetGain(*self*) → float64_t

float64_t Spinnaker::ChunkData::GetGain() const

Description: Returns the gain used to capture the image. Visibility: Expert

GetHeight(*self*) → int64_t

int64_t Spinnaker::ChunkData::GetHeight() const

Description: Returns the Height of the image included in the payload. Visibility: Expert

GetImage(*self*) → int64_t

GetInferenceBoundingBoxResult(*self*) → InferenceBoundingBoxResult

GetInferenceConfidence(*self*) → float64_t

GetInferenceFrameId(*self*) → int64_t

GetInferenceResult(*self*) → int64_t

GetLinePitch(*self*) → int64_t

int64_t Spinnaker::ChunkData::GetLinePitch() const

Description: Returns the LinePitch of the image included in the payload. Visibility: Expert

GetLineStatusAll(*self*) → int64_t

int64_t Spinnaker::ChunkData::GetLineStatusAll() const

Description: Returns the status of all the I/O lines at the time of the FrameStart internal event. Visibility: Expert

GetOffsetX(*self*) → int64_t

int64_t Spinnaker::ChunkData::GetOffsetX() const

Description: Returns the OffsetX of the image included in the payload. Visibility: Expert

GetOffsetY(*self*) → int64_t

int64_t Spinnaker::ChunkData::GetOffsetY() const

Description: Returns the OffsetY of the image included in the payload. Visibility: Expert

GetPartSelector(*self*) → int64_t

int64_t Spinnaker::ChunkData::GetPartSelector() const

Description: Selects the part to access in chunk data in a multipart transmission. Visibility: Expert

GetPixelDynamicRangeMax(*self*) → int64_t

int64_t Spinnaker::ChunkData::GetPixelDynamicRangeMax() const

Description: Returns the maximum value of dynamic range of the image included in the payload. Visibility: Expert

GetPixelDynamicRangeMin(*self*) → int64_t

int64_t Spinnaker::ChunkData::GetPixelDynamicRangeMin() const

Description: Returns the minimum value of dynamic range of the image included in the payload. Visibility: Expert

GetScan3dAxisMax(*self*) → float64_t

float64_t Spinnaker::ChunkData::GetScan3dAxisMax() const

Description: Returns the Maximum Axis value for the selected coordinate axis of the image included in the payload. Visibility: Expert

GetScan3dAxisMin(*self*) → float64_t

float64_t Spinnaker::ChunkData::GetScan3dAxisMin() const

Description: Returns the Minimum Axis value for the selected coordinate axis of the image included in the payload. Visibility: Expert

GetScan3dCoordinateOffset(*self*) → float64_t

float64_t Spinnaker::ChunkData::GetScan3dCoordinateOffset() const

Description: Returns the Offset for the selected coordinate axis of the image included in the payload. Visibility: Expert

GetScan3dCoordinateReferenceValue(*self*) → float64_t

float64_t Spinnaker::ChunkData::GetScan3dCoordinateReferenceValue() const

Description: Reads the value of a position or pose coordinate for the anchor or transformed coordinate systems relative to the reference point. Visibility: Expert

GetScan3dCoordinateScale(*self*) → float64_t

float64_t Spinnaker::ChunkData::GetScan3dCoordinateScale() const

Description: Returns the Scale for the selected coordinate axis of the image included in the payload. Visibility: Expert

GetScan3dInvalidDataValue(*self*) → float64_t

float64_t Spinnaker::ChunkData::GetScan3dInvalidDataValue() const

Description: Returns the Invalid Data Value used for the image included in the payload. Visibility: Expert

GetScan3dTransformValue(*self*) → float64_t

float64_t Spinnaker::ChunkData::GetScan3dTransformValue() const

Description: Returns the transform value. Visibility: Expert

GetScanLineSelector(*self*) → int64_t

int64_t Spinnaker::ChunkData::GetScanLineSelector() const

Description: Index for vector representation of one chunk value per line in an image. Visibility: Expert

GetSequencerSetActive(*self*) → int64_t

int64_t Spinnaker::ChunkData::GetSequencerSetActive() const

Description: Return the index of the active set of the running sequencer included in the payload. Visibility: Expert

GetSerialDataLength(*self*) → int64_t

GetStreamChannelID(*self*) → int64_t

int64_t Spinnaker::ChunkData::GetStreamChannelID() const

Description: Returns identifier of the stream channel used to carry the block. Visibility: Expert

GetTimerValue(*self*) → float64_t

float64_t Spinnaker::ChunkData::GetTimerValue() const

Description: Returns the value of the selected Timer at the time of the FrameStart internal event. Visibility: Expert

GetTimestamp(*self*) → int64_t

int64_t Spinnaker::ChunkData::GetTimestamp() const

Description: Returns the Timestamp of the image included in the payload at the time of the FrameStart internal event. Visibility: Expert

GetTimestampLatchValue(*self*) → int64_t

int64_t Spinnaker::ChunkData::GetTimestampLatchValue() const

Description: Returns the last Timestamp latched with the TimestampLatch command. Visibility: Expert

GetTransferBlockID(*self*) → int64_t

int64_t Spinnaker::ChunkData::GetTransferBlockID() const

Description: Returns the unique identifier of the transfer block used to transport the payload. Visibility: Expert

GetTransferQueueCurrentBlockCount(*self*) → int64_t

int64_t Spinnaker::ChunkData::GetTransferQueueCurrentBlockCount() const

Description: Returns the current number of blocks in the transfer queue. Visibility: Expert

GetWidth(*self*) → int64_t

int64_t Spinnaker::ChunkData::GetWidth() const

Description: Returns the Width of the image included in the payload. Visibility: Expert

SetChunks(*self*, *pNodeMap*)

Parameters

- **pNodeMap** (*Spinnaker::GenApi::INodeMap &*) –
- **void** –
- **&pNodeMap)** (*Spinnaker::ChunkData::SetChunks(GenApi::INodeMap*) –

property thisown

The membership flag

4.8 PySpin.Image

class PySpin.**Image**(**args*, ***kwargs*)

The image object class.

C++ includes: Image.h

CheckCRC(*self*) → bool

bool Spinnaker::Image::CheckCRC() const

Checks if the computed checksum matches with chunk data's ImageCRC

Returns true if computed checksum matches with the chunk data's CRC and false otherwise.

static Create() → *ImagePtr*

static Create(*image*) → *ImagePtr*

Parameters

- **image** (*Spinnaker::ImagePtr const*) –
- **Create(width** –
- **height** (*size_t*) –
- **offsetX** (*size_t*) –
- **offsetY** (*size_t*) –
- **pixelFormat** (*enum Spinnaker::PixelFormatEnums*) –
- **ImagePtr** (*copied from another*) –
- **width** (*or using*) –
- **height** –
- **offsetX** –
- **offsetY** –
- **pixelFormat** –
- **pData** (*void **) –
- **Create(width** –
- **height** –
- **offsetX** –
- **offsetY** –
- **pixelFormat** –
- **pData** –
- **dataPayloadType** (*enum Spinnaker::TLPayloadType*) –
- **ImagePtr** –
- **width** –
- **height** –
- **offsetX** –
- **offsetY** –

- **pixelFormat** –
- **pData** –
- **dataPayloadType** –
- **dataSize** (*size_t*) –
- **object** (*Creates a new Image*) –
- **constructor** (*either using a default*) –
- **ImagePtr** –
- **width** –
- **height** –

:param : :param offset_x: :param offset_y: :param pixel format: :param and a NumPy array containing 8-bit unsigned ints representing the image data: :param (replaces the void* pData argument).:

DeepCopy(*self*, *pSrcImage*)

Parameters

- **pSrcImage** (*The Image to copy the data from.*) –
- **void** –
- **pSrcImage)** (*Spinnaker::Image::DeepCopy(const ImagePtr*) –
- **operation** (*Performs a deep copy of the Image. After this*) –
- **image** (*the*) –
- **not** (*contents and member variables will be the same. The Images will*) –
- **released.** (*share a buffer. The Image's current buffer will not be*) –
- **Parameters** –
- **----------** –
- **pSrcImage** –

GetBitsPerPixel(*self*) → size_t

size_t Spinnaker::Image::GetBitsPerPixel() const

Gets the number of bits used per pixel in the image. This information is retrieved from the Transport Layer Image format headers. It is retrieved on a per image basis.

The number of bits used per pixel.

GetBufferSize(*self*) → size_t

size_t Spinnaker::Image::GetBufferSize() const

Gets the size of the buffer associated with the image in bytes.

The size of the buffer, in bytes.

GetChunkData(*self*) → *ChunkData*

const ChunkData& Spinnaker::Image::GetChunkData() const

Returns a pointer to a chunk data interface. No ownership is transfered, the chunk data interface reference is valid until Image::Release() is called on this image.

ChunkData interface that provides access to image chunks.

GetChunkLayoutId(*self*) → uint64_t

uint64_t Spinnaker::Image::GetChunkLayoutId() const

Returns the id of the chunk data layout.

uint64_t value representing the id of the chunk data layout.

GetColorProcessing(*self*) → Spinnaker::ColorProcessingAlgorithm

ColorProcessingAlgorithm Spinnaker::Image::GetColorProcessing() const

Gets the algorithm used to produce the image.

See: Convert()

The color processing algorithm used to produce the image.

GetDataAbsoluteMax(*self*) → float

GetDataAbsoluteMin(*self*) → float

GetFrameID(*self*) → uint64_t

uint64_t Spinnaker::Image::GetFrameID() const

Gets the frame ID for this image.

The frame ID.

GetHeight(*self*) → size_t

size_t Spinnaker::Image::GetHeight() const

Gets the height of the image in pixels. This information is retrieved from the Transport Layer Image format headers. It is retrieved on a per image basis.

The height in pixels.

GetID(*self*) → uint64_t

uint64_t Spinnaker::Image::GetID() const

Gets a unique ID for this image. Each image in a steam will have a unique ID to help identify it.

The 64 bit unique id for this image.

GetImageSize(*self*) → size_t

size_t Spinnaker::Image::GetImageSize() const

Returns the size of the image

The image size in bytes.

GetImageStatus(*self*) → Spinnaker::ImageStatus

ImageStatus Spinnaker::Image::GetImageStatus() const

Returns data integrity status of the image returned from GetNextImage()

Returns whether image has any data integrity issues.

static GetImageStatusDescription(*status*) → char const *

Parameters
status (*enum Spinnaker::ImageStatus*) –

GetNumChannels(*self*) → size_t

GetPayloadType(*self*) → size_t

size_t Spinnaker::Image::GetPayloadType() const

Gets the payload type that was transmitted. This is a device types specific value that identifies how the image was transmitted. This information is retrieved from the Transport Layer Image format headers. It is retrieved on a per image basis.

Device types specific payload type.

GetPixelFormat(*self*) → Spinnaker::PixelFormatEnums

Spinnaker::PixelFormatEnums Spinnaker::Image::GetPixelFormat() const

Returns an enum value that represents the pixel format of this image. The enum can be used with the easy access GenICam features available through the Camera.h header file. This easy access enum can also be used in the Convert() function.

See: Convert()

enum value representing the PixelFormat.

GetPixelFormatIntType(*self*) → Spinnaker::PixelFormatIntType

GetPixelFormatName(*self*) → gcstring

GenICam::gcstring Spinnaker::Image::GetPixelFormatName() const

Returns a string value that represents this image's pixel format. The string is a valid SFNC name that maps to the underlying TL specific pixel format. This is the most generic way to identify the pixel format of the image.

string value representing the PixelFormat.

GetPrivateData(*self*) → void *

void* Spinnaker::Image::GetPrivateData() const

Gets a pointer to the user passed data associated with the image. This function is considered unsafe. The pointer returned could be invalidated if the buffer is released. The pointer may also be invalidated if the Image object is passed to Image::Release().

TODO: no way to set private data for image yet.

A pointer to the user passed data pointer.

GetStreamIndex(*self*) → uint64_t

GetStride(*self*) → size_t

size_t Spinnaker::Image::GetStride() const

Gets the stride of the image in bytes. The stride of an image is how many bytes are in each row. This information is retrieved from the Transport Layer Image format headers. It is retrieved on a per image basis.

The stride in bytes.

GetTLPayloadType(*self*) → Spinnaker::TLPayloadType

PayloadTypeInfoIDs Spinnaker::Image::GetTLPayloadType() const

Gets the GenTL specific payload type that was transmitted. This is a Transport Layer specific value that identifies how the image was transmitted. This information is retrieved from the Transport Layer Image format headers. It is retrieved on a per image basis.

Transport Layer specific payload type.

GetTLPixelFormat(*self*) → uint64_t

uint64_t Spinnaker::Image::GetTLPixelFormat() const

Gets the pixel format of the image. This is a Transport Layer specific pixel format that identifies how the pixels in the image should be interpreted. To understand how to interpret this value it is necessary to know what the transport layer namespace is. This can be retrieved through a call to GetTLPixelFormatNamespace(). This information is retrieved from the Transport Layer Image format headers. It is retrieved on a per image basis.

See: GetTLPixelFormatNamespace()

Transport Layer specific pixel format.

GetTLPixelFormatNamespace(*self*) → Spinnaker::TLPixelFormatNamespace

PixelFormatNamespaceID Spinnaker::Image::GetTLPixelFormatNamespace() const

Returns an enum value that represents the namespace in which this image's TL specific pixel format resides. This information is important to properly interpret the value returned by GetTLPixelFormat()

See: GetTLPixelFormat()

enum value representing the PixelFormatNamespace.

GetTimeStamp(*self*) → uint64_t

uint64_t Spinnaker::Image::GetTimeStamp() const

Gets the time stamp for the image in nanoseconds.

The time stamp of the image.

GetValidPayloadSize(*self*) → size_t

size_t Spinnaker::Image::GetValidPayloadSize() const

Returns the size of valid data in the image payload. This is the actual amount of data read from the device. A user created image has a payload size of zero. GetBufferSize() returns the total size of bytes allocated for the image.

See: GetBufferSize()

size_t value representing valid payload.

GetWidth(*self*) → size_t

size_t Spinnaker::Image::GetWidth() const

Gets the width of the image in pixels. This information is retrieved from the Transport Layer image format headers. It is retrieved on a per image basis.

The width in pixels.

GetXOffset(*self*) → size_t

size_t Spinnaker::Image::GetXOffset() const

Gets the ROI x offset in pixels for this image. This information is retrieved from the Transport Layer Image format headers. It is retrieved on a per image basis.

The x offset in pixels.

GetXPadding(*self*) → size_t

size_t Spinnaker::Image::GetXPadding() const

Gets the x padding in bytes for this image. This is the number of bytes at the end of each line to facilitate alignment in buffers. This information is retrieved from the Transport Layer Image format headers. It is retrieved on a per image basis.

The x padding in bytes.

GetYOffset(*self*) → size_t

size_t Spinnaker::Image::GetYOffset() const

Gets the ROI y offset in pixels for this image. This information is retrieved from the Transport Layer Image format headers. It is retrieved on a per image basis.

The y offset in pixels.

GetYPadding(*self*) → size_t

size_t Spinnaker::Image::GetYPadding() const

Gets the y padding in bytes for this image. This is the number of bytes at the end of each image to facilitate alignment in buffers. This information is retrieved from the Transport Layer Image format headers. It is retrieved on a per image basis.

The y padding in bytes.

HasCRC(*self*) → bool

bool Spinnaker::Image::HasCRC() const

Checks if the image contains ImageCRC checksum from chunk data

Returns true if image contains ImageCRC checksum from chunk data and false otherwise.

HasChunkData(*self*) → bool

IsCompressed(*self*) → bool

IsInUse(*self*) → bool

bool Spinnaker::Image::IsInUse()

Returns true if the image is still in use by the stream

Returns true if the image is in use and false otherwise.

IsIncomplete(*self*) → bool

bool Spinnaker::Image::IsIncomplete() const

Returns a boolean value indicating if this image was incomplete. An image is marked as incomplete if the transport layer received less data then it requested.

Returns true if image is incomplete, false otherwise.

static Load(*pFilename*, *format=SPINNAKER_IMAGE_FILE_FORMAT_FROM_FILE_EXT*) → *ImagePtr*

Parameters

- **pFilename** (*char const **) –
- **format** (*enum Spinnaker::ImageFileFormat*) –

Release(*self*)

void Spinnaker::Image::Release()

ResetImage(*self*, *width*, *height*, *offsetX*, *offsetY*, *pixelFormat*)

Parameters

- **width** (*The width of image in pixels to set.*) –
- **height** (*The height of image in pixels to set.*) –
- **offsetX** (*The x offset in pixels to set.*) –

- **offsetY** (*The y offset in pixels to set.*) –
- **pixelFormat** (*Pixel format to set.*) –
- **ResetImage(self** –
- **width** –
- **height** –
- **offsetX** –
- **offsetY** –
- **pixelFormat** –
- **pData)** –
- **width** –
- **height** –
- **offsetX** –
- **offsetY** –
- **pixelFormat** –
- **pData** (*Pointer to the image buffer.*) –
- **ResetImage(self** –
- **width** –
- **height** –
- **offsetX** –
- **offsetY** –
- **pixelFormat** –
- **pData** –
- **dataPayloadType** (*enum Spinnaker::TLPayloadType*) –
- **dataSize)** –
- **width** –
- **height** –
- **offsetX** –
- **offsetY** –
- **pixelFormat** –
- **pData** –
- **dataPayloadType** –
- **dataSize** (*size_t*) –
- **void** –
- **width** –
- **height** –
- **size_t** –

- offsetX –
- offsetY –
- pixelFormat –
- void –
- *pData) –
- object. (*Sets new dimensions of the image*) –
- Parameters –
- ---------- –
- width –
- height –
- offsetX –
- offsetY –
- pixelFormat –
- pData –

Save(*self*, *pFilename*, *format=SPINNAKER_IMAGE_FILE_FORMAT_FROM_FILE_EXT*)

Parameters

- pFilename (*Filename to save image with.*) –
- format (*enum Spinnaker::ImageFileFormat*) –
- Save(self –
- pFilename –
- pOption) –
- pFilename –
- pOption (*Options to use while saving image.*) –
- Save(self –
- pFilename –
- pOption) –
- pFilename –
- pOption –
- Save(self –
- pFilename –
- pOption) –
- pFilename –
- pOption –
- Save(self –
- pFilename –
- pOption) –

- **pFilename** –
- **pOption** –
- **Save(self** –
- **pFilename** –
- **pOption)** –
- **pFilename** –
- **pOption** –
- **Save(self** –
- **pFilename** –
- **pOption)** –
- **pFilename** –
- **pOption** –
- **Save(self** –
- **pFilename** –
- **pOption)** –
- **pFilename** –
- **pOption** –
- **Save(self** –
- **pFilename** –
- **option)** –
- **pFilename** –
- **option** (*Spinnaker::SIOption &*) –
- **void** –
- ***pFilename** (*Spinnaker::Image::Save(const char)* –
- **&pOption)** (*BMPOption*) –
- **specified.** (*Saves the image to the specified file name with the options*) –
- **Parameters** –
- **----------** –
- **pFilename** –
- **pOption** –

property thisown

The membership flag

4.9 PySpin.ImageList

class PySpin.**ImageList**(**args*)

Proxy of C++ Spinnaker::ImageList class.

Add(*self*, *image*)

Parameters
image (*Spinnaker::ImagePtr*) –

Append(*self*, *list*)

Parameters
list (*Spinnaker::ImageList const &*) –

Clear(*self*)

GetByIndex(*self*, *index*) → *ImagePtr*

Parameters
index (*unsigned int*) –

GetByPixelFormat(*self*, *pixelFormat*) → *ImagePtr*

Parameters
pixelFormat (*enum Spinnaker::PixelFormatEnums*) –

GetSize(*self*) → unsigned int

static Load(*filename*) → *ImageList*

Parameters
filename (*char const **) –

Release(*self*)

RemoveByIndex(*self*, *index*)

Parameters
index (*unsigned int*) –

RemoveByPixelFormat(*self*, *pixelFormat*)

Parameters
pixelFormat (*enum Spinnaker::PixelFormatEnums*) –

Save(*self*, *filename*)

Parameters
filename (*char const **) –

property thisown

The membership flag

4.10 PySpin.ImageProcessor

class PySpin.**ImageProcessor**(**args*)

Proxy of C++ Spinnaker::ImageProcessor class.

ApplyGamma(*self*, *srcImage*, *gamma*, *applyGammaInverse=False*) → *ImagePtr*

Parameters

- **srcImage** (*Spinnaker::ImagePtr const &*) –
- **gamma** (*float*) –
- **applyGammaInverse** (*bool*) –
- **ApplyGamma(self** –
- **srcImage** –
- **destImage** (*Spinnaker::ImagePtr &*) –
- **gamma** –
- **applyGammaInverse=False)** –
- **srcImage** –
- **destImage** –
- **gamma** –
- **applyGammaInverse** –

Convert(*self*, *srcImage*, *destFormat*) → *ImagePtr*

Parameters

- **srcImage** (*Spinnaker::ImagePtr const &*) –
- **destFormat** (*enum Spinnaker::PixelFormatEnums*) –
- **Convert(self** –
- **srcImage** –
- **destImage** (*Spinnaker::ImagePtr &*) –
- **destFormat)** –
- **srcImage** –
- **destImage** –
- **destFormat** –
- **Convert(self** –
- **srcImageList** (*Spinnaker::ImageList const &*) –
- **ImagePtr** (*destFormat) ->*) –
- **srcImageList** –
- **destFormat** –
- **Convert(self** –
- **srcImageList** –

- **destImage** –
- **destFormat)** –
- **srcImageList** –
- **destImage** –
- **destFormat** –

GetColorProcessing(*self*) → Spinnaker::ColorProcessingAlgorithm

GetNumDecompressionThreads(*self*) → unsigned int

SetColorProcessing(*self*, *colorAlgorithm*)

Parameters
colorAlgorithm (*enum Spinnaker::ColorProcessingAlgorithm*) –

SetNumDecompressionThreads(*self*, *numThreads*)

Parameters
numThreads (*unsigned int*) –

property thisown

The membership flag

4.11 PySpin.ImagePtr

class PySpin.**ImagePtr**(**args*)

A reference tracked pointer to an image object. When the ImagePtr goes out of scope, it will trigger an auto release of the image from the stream.

C++ includes: ImagePtr.h

property thisown

The membership flag

4.12 PySpin.IInterface

class PySpin.**IInterface**(**args*, ***kwargs*)

Proxy of C++ Spinnaker::IInterface class.

GetCameras(*self*, *updateCameras=True*) → *CameraList*

Parameters
updateCameras (*bool*) –

GetTLNodeMap(*self*) → INodeMap

IsInUse(*self*) → bool

IsValid(*self*) → bool

RegisterEventHandler(*self*, *evtHandlerToRegister*)

Parameters
evtHandlerToRegister (*Spinnaker::EventHandler &*) –

SendActionCommand(*self*, *deviceKey*, *groupKey*, *groupMask*, *actionTime=0*, *pResultSize=None*, *results=0*)

Parameters

- **deviceKey** (*unsigned int*) –
- **groupKey** (*unsigned int*) –
- **groupMask** (*unsigned int*) –
- **actionTime** (*unsigned long long*) –
- **pResultSize** (*unsigned int **) –
- **results** (*Spinnaker::ActionCommandResult []*) –

property TLInterface

UnregisterEventHandler(*self*, *evtHandlerToUnregister*)

Parameters
evtHandlerToUnregister (*Spinnaker::EventHandler &*) –

UpdateCameras(*self*) → bool

property thisown
The membership flag

4.13 PySpin.InterfaceList

class PySpin.**InterfaceList**(**args*)
A list of the available interfaces on the system.

C++ includes: InterfaceList.h

Add(*self*, *iface*)

Parameters
iface (*Spinnaker::InterfacePtr*) –

Append(*self*, *list*)

Parameters
list (*Spinnaker::InterfaceList const **) –

Clear(*self*)
void Spinnaker::InterfaceList::Clear()

Clears the list of interfaces and destroys their corresponding objects. It is important to first make sure there are no referenced cameras still in use before calling Clear(). If a camera on any of the interfaces is still in use this function will throw an exception.

GetByIndex(*self*, *index*) → *InterfacePtr*

Parameters

- **index** (*The index at which to retrieve the Interface object*) –
- **const** (*InterfacePtr Spinnaker::InterfaceList::GetByIndex(int index)*) –
- **"index".** (*Returns a pointer to an Interface object at the*) –
- **Parameters** –
- **----------** –
- **index** –
- **object.** (*A pointer to an Interface*) –

GetByInterfaceID(*self*, *interfaceID*) → *InterfacePtr*

Parameters
interfaceID (*std::string*) –

GetSize(*self*) → unsigned int

int Spinnaker::InterfaceList::GetSize() const

Returns the size of the interface list. The size is the number of Interface objects stored in the list.

An integer that represents the list size.

Remove(*self*, *iface*)

Parameters
iface (*Spinnaker::InterfacePtr*) –

property thisown

The membership flag

4.14 PySpin.InterfacePtr

class PySpin.**InterfacePtr**(**args*)

A reference tracked pointer to the interface object.

C++ includes: InterfacePtr.h

property thisown

The membership flag

4.15 PySpin.SpinnakerException

class PySpin.**SpinnakerException**

Exception class for the PySpin module. This class has these attributes: message, errorcode, fullmessage

errorcode = 0

fullmessage = ''

message = ''

4.16 PySpin.SpinVideo

class PySpin.**SpinVideo**

Provides the functionality for the user to record images to an AVI file.

C++ includes: SpinVideo.h

Append(*self*, *pImage*)

Parameters

- **pImage** (*The image to append.*) –
- **virtual** –
- **pImage)** (*void Spinnaker::Video::SpinVideo::Append(ImagePtr*) –
- **file.** (*Append an image to the AVI/MP4*) –
- **Parameters** –
- **----------** –
- **pImage** –

Close(*self*)

virtual void Spinnaker::Video::SpinVideo::Close()

Close the AVI/MP4 file.

See: Open()

Open(*self*, *pFileName*, *pOption*)

Parameters

- **pFileName** (*The filename of the MP4 file.*) –
- **pOption** (*H264 options to apply to the MP4 file.*) –
- **Open(self** –
- **pFileName** –
- **pOption)** –
- **pFileName** –
- **pOption** –
- **Open(self** –
- **pFileName** –
- **pOption)** –
- **pFileName** –
- **pOption** –
- **void** (*virtual*) –
- ***pFileName** (*Spinnaker::Video::SpinVideo::Open(const char*) –
- **Video::H264Option** –
- **&pOption)** –

- **The** (*Open an H264 MP4 file in preparation for writing Images to disk.*) –
- **automatically** (*size of MP4 files is limited to 2GB. The filenames are*) –
- **specified.** (*generated using the filename*) –
- **Parameters** –
- **----------** –
- **pFileName** –
- **pOption** –
- **See** (*H264Option*) –
- **See** –

SetMaximumFileSize(*self*, *size*)

Parameters
size (*unsigned int*) –

property thisown

The membership flag

4.17 PySpin.System

class PySpin.**System**(**args*, ***kwargs*)

The system object is used to retrieve the list of interfaces and cameras available.

C++ includes: System.h

GetCameras(*self*, *updateInterfaces=True*, *updateCameras=True*) → *CameraList*

Parameters

- **updateInterfaces** (*Determines whether or not updateInterfaceList() is*) –
- **updateCameras** (*Determines whether or not UpdateCameras() is called*) –
- **CameraList** –
- **updateInterfaces=true** (*Spinnaker::System::GetCameras(bool)*) –
- **bool** –
- **updateCameras=true)** –
- **call** (*Returns a list of cameras that are available on the system. This*) –
- **interfaces.** (*returns both GigE Vision and Usb3 Vision cameras from all*) –
- **It** (*The camera list object will reference count the cameras it returns.*) –
- **before** (*is important that the camera list is destroyed or is cleared*) –

- **system->** (*calling system-> ReleaseInstance() or else the call to*) –
- **a** (*ReleaseInstance() will result in an error message thrown that*) –
- **held.** (*reference to the camera is still*) –
- **See** (*CameraList::Clear()*) –
- **See** –
- **Parameters** –
- **----------** –
- **updateInterfaces** –
- **system** (*before getting cameras from available interfaces on the*) –
- **updateCameras** –
- **system** –
- **cameras.** (*An CameraList object that contains a list of all*) –

static GetInstance() → *SystemPtr*

GetInterfaces(*self*, *updateInterface=True*) → *InterfaceList*

Parameters

- **updateInterface** (*Determines whether or not UpdateInterfaceList() is*) –
- **Spinnaker::System::GetInterfaces(bool** (InterfaceList) –
- **updateInterface=true)** –
- **call** (*Returns a list of interfaces available on the system. This*) –
- **interfaces.** (*An InterfaceList object that contains a list of all*) –
- **Parameters** –
- **----------** –
- **updateInterface** –
- **interfaces** (*called before getting available*) –
- **interfaces.** –

GetLibraryVersion(*self*) → LibraryVersion

GetLoggingEventPriorityLevel(*self*) → Spinnaker::SpinnakerLogLevel

SpinnakerLogLevel Spinnaker::System::GetLoggingEventPriorityLevel()

Retrieves the current logging event priority level.

Spinnaker uses five levels of logging: Error - failures that are non- recoverable without user intervention.

Warning - failures that are recoverable without user intervention.

Notice - information about events such as camera arrival and removal, initialization and deinitialization, starting and stopping image acquisition, and feature modification.

Info - information about recurring events that are generated regularly such as information on individual images.

Debug - information that can be used to troubleshoot the system.

See: SpinnakerLogLevel

Level The threshold level

GetTLNodeMap(*self*) → INodeMap

IsInUse(*self*) → bool

bool Spinnaker::System::IsInUse()

Checks if the system is in use by any interface or camera objects.

Returns true if the system is in use and false otherwise.

RegisterEventHandler(*self*, *evtHandlerToRegister*, *updateInterface=False*)

Parameters

- **evtHandlerToRegister** (*Spinnaker::EventHandler &*) –
- **updateInterface** (*bool*) –

RegisterLoggingEventHandler(*self*, *handler*)

Parameters
handler (*Spinnaker::LoggingEventHandler &*) –

ReleaseInstance(*self*)

void Spinnaker::System::ReleaseInstance()

This call releases the instance of the System Singleton for this process. After successfully releasing the System instance the pointer returned by GetInstance() will be invalid. Calling ReleaseInstance while a camera reference is still held will throw an error of type SPINNAKER_ERR_RESOURCE_IN_USE.

See: Error

See: GetInstance()

SendActionCommand(*self*, *deviceKey*, *groupKey*, *groupMask*, *actionTime=0*, *pResultSize=None*, *results=0*)

Parameters

- **deviceKey** (*Spinnaker::System::SendActionCommand(unsigned int*) –
- **groupKey** (*int*) –
- **groupMask** (*unsigned int*) –
- **actionTime** (*unsigned long long*) –
- **pResultSize** (*unsigned int **) –
- **results** (*Spinnaker::ActionCommandResult []*) –
- **void** –
- **deviceKey** –
- **unsigned** –
- **groupKey** –
- **groupMask** –
- **actionTime=0** (*unsigned long long*) –

:param : :param unsigned int *pResultSize=0: :param ActionCommandResult results[]=NULL): :param Broadcast an Action Command to all devices on system: :param Parameters: :param ————: :param deviceKey: :type deviceKey: The Action Command's device key :param groupKey: :type groupKey: The Action Command's group key :param groupMask: :type groupMask: The Action Command's group mask :param actionTime: :type actionTime: (Optional) Time when to assert a future action. Zero :param means immediate action.: :param pResultSize: :type pResultSize: (Optional) The number of results in the results array. :param The value passed should be equal to the expected number of devices: :param that acknowledge the command. Returns the number of received results.: :param results: :type results: (Optional) An Array with *pResultSize elements to hold the :param action command result status. The buffer is filled starting from index: :param 0. If received results are less than expected number of devices that: :param acknowledge the command: :param remaining results are not changed. If: :param received results are more than expected number of devices that: :param acknowledge the command: :param extra results are ignored and not appended to: :param array. This parameter is ignored if pResultSize is 0. Thus this: :param parameter can be NULL if pResultSize is 0 or NULL.:

SetLoggingEventPriorityLevel(*self*, *level*)

Parameters

- **level** (*enum Spinnaker::SpinnakerLogLevel*) –
- **void** –
- **Spinnaker::System::SetLoggingEventPriorityLevel(SpinnakerLogLevel** –
- **level)** –
- **events** (*Sets a threshold priority level for logging event. Logging*) –
- **callbacks.** (*below such level will not trigger*) –
- **logging** (*Spinnaker uses five levels of*) –
- **intervention.** (*Warning - failures that are recoverable without user*) –
- **intervention.** –
- **removal** (*Notice - information about events such as camera arrival and*) –

:param : :param initialization and deinitialization: :param starting and stopping image: :param acquisition: :param and feature modification.: :param Info - information about recurring events that are generated regularly: :param such as information on individual images.: :param Debug - information that can be used to troubleshoot the system.: :param See: :type See: SpinnakerLogLevel :param Parameters: :param ————: :param level: :type level: The threshold level

UnregisterAllLoggingEventHandlers(*self*)

UnregisterEventHandler(*self*, *evtHandlerToUnregister*)

Parameters

evtHandlerToUnregister (*Spinnaker::EventHandler &*) –

UnregisterLoggingEventHandler(*self*, *handler*)

Parameters

handler (*Spinnaker::LoggingEventHandler &*) –

UpdateCameras(*self*, *updateInterfaces=True*) → bool

Parameters

- **updateInterfaces** (*bool*) –

- **bool** –
- **updateInterfaces=true)** (*Spinnaker::System::UpdateCameras(bool)* –
- **that** (*Updates the list of cameras on the system. Note*) –
- **each** (*System::GetCameras() internally calls UpdateCameras() for*) –
- **the** (*interface it enumerates. If the list changed between this call and*) –
- **true** (*last time UpdateCameras was called then the return value will be*) –

:param : :param otherwise it is false.: :param See: :type See: GetCameras() :param Parameters: :param ————: :param updateInterfaces: :type updateInterfaces: Determines whether or not UpdateInterfaceList() is :param called before updating cameras for available interfaces on the system: :param True if cameras changed on interface and false otherwise.:

UpdateInterfaceList(*self*)

property thisown

The membership flag

4.18 PySpin.SystemPtr

class PySpin.**SystemPtr**(**args*)

A reference tracked pointer to a system object.

C++ includes: SystemPtr.h

property thisown

The membership flag

CHAPTER

FIVE

QUICKSPIN CLASSES

5.1 PySpin.TransportLayerDevice

class PySpin.**TransportLayerDevice**(*nodeMapTLDevice*)

Part of the QuickSpin API to provide access to camera information without having to first initialize the camera.

C++ includes: TransportLayerDevice.h

property DeviceAccessStatus

property DeviceBootloaderVersion

property DeviceCurrentSpeed

property DeviceDisplayName

property DeviceDriverVersion

property DeviceEndianessMechanism

property DeviceID

property DeviceInstanceId

property DeviceIsUpdater

property DeviceLinkSpeed

property DeviceLocation

property DeviceModelName

property DeviceMulticastMonitorMode

property DevicePortId

property DeviceReset

property DeviceSerialNumber

property DeviceType

property DeviceU3VProtocol

property DeviceUserID

property DeviceVendorName

property DeviceVersion

property GUIXMLLocation

property GUIXMLPath

property GenICamXMLLocation

property GenICamXMLPath

property GevCCP

property GevDeviceAutoForceIP

property GevDeviceDiscoverMaximumPacketSize

property GevDeviceForceGateway

property GevDeviceForceIP

property GevDeviceForceIPAddress

property GevDeviceForceSubnetMask

property GevDeviceGateway

property GevDeviceIPAddress

property GevDeviceIsWrongSubnet

property GevDeviceMACAddress

property GevDeviceMaximumPacketSize

property GevDeviceMaximumRetryCount

property GevDeviceModeIsBigEndian

property GevDevicePort

property GevDeviceReadAndWriteTimeout

property GevDeviceSubnetMask

property GevVersionMajor

property GevVersionMinor

property thisown

The membership flag

5.2 PySpin.TransportLayerInterface

class PySpin.**TransportLayerInterface**(*nodeMapTLDevice*)

Part of the QuickSpin API to provide access to camera information without having to first initialize the camera.

C++ includes: TransportLayerInterface.h

property ActionCommand

property DeviceAccessStatus

property DeviceCount

property DeviceID

property DeviceModelName

property DeviceSelector

property DeviceSerialNumber

property DeviceUnlock

property DeviceUpdateList

property DeviceVendorName

property FilterDriverStatus

property GevActionDeviceKey

property GevActionGroupKey

property GevActionGroupMask

property GevActionTime

property GevDeviceAutoForceIP

property GevDeviceForceGateway

property GevDeviceForceIP

property GevDeviceForceIPAddress

property GevDeviceForceSubnetMask

property GevDeviceGateway

property GevDeviceIPAddress

property GevDeviceMACAddress

property GevDeviceSubnetMask

property GevInterfaceGateway

property GevInterfaceGatewaySelector

property GevInterfaceMACAddress

property GevInterfaceMTU

property GevInterfaceReceiveLinkSpeed

property GevInterfaceSubnetIPAddress

property GevInterfaceSubnetMask

property GevInterfaceSubnetSelector

property GevInterfaceTransmitLinkSpeed

property HostAdapterDriverVersion

property HostAdapterName

property HostAdapterVendor

property IncompatibleDeviceCount

property IncompatibleDeviceID

property IncompatibleDeviceModelName

property IncompatibleDeviceSelector

property IncompatibleDeviceVendorName

property IncompatibleGevDeviceIPAddress

property IncompatibleGevDeviceMACAddress

property IncompatibleGevDeviceSubnetMask

property InterfaceDisplayName

property InterfaceID

property InterfaceType

property POEStatus

property thisown

The membership flag

5.3 PySpin.TransportLayerStream

class PySpin.**TransportLayerStream**(*nodeMapTLDevice*)

Part of the QuickSpin API to provide access to camera information without having to first initialize the camera.

C++ includes: TransportLayerStream.h

property StreamAnnounceBufferMinimum

property StreamAnnouncedBufferCount

property StreamBlockTransferSize

property StreamBufferAlignment

property StreamBufferCountManual

property StreamBufferCountMax

property StreamBufferCountMode

property StreamBufferCountResult

property StreamBufferHandlingMode

property StreamCRCCheckEnable

property StreamChunkCountMaximum

property StreamDeliveredFrameCount

property StreamDroppedFrameCount

property StreamID

property StreamIncompleteFrameCount

property StreamInputBufferCount

property StreamIsGrabbing

property StreamLostFrameCount

property StreamMissedPacketCount

property StreamMode

property StreamOutputBufferCount

property StreamPacketResendEnable

property StreamPacketResendMaxRequests

property StreamPacketResendReceivedPacketCount

property StreamPacketResendRequestCount

property StreamPacketResendRequestSuccessCount

property StreamPacketResendRequestedPacketCount

property StreamPacketResendTimeout

property StreamReceivedFrameCount

property StreamReceivedPacketCount

property StreamStartedFrameCount

property StreamType

property thisown

The membership flag

CHAPTER

SIX

PYSPIN MODULE

PYTHON MODULE INDEX

INDEX

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W