FlyCapture2 Managed

2.13.3.61

Generated by Doxygen 1.7.5

Wed Apr 3 2019 19:09:01

Contents

Chapter 1

Software Licensing Information

Component	License
FlyCapture2	Copyright © 2017 FLIR Integrated Imaging Solutions, 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.
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
GTK	LGPv2.1 License http://www.gnu.-org/licenses/old-licenses/lgpl-2.-1.txt
Libusb	LGPLv2.1 License http://www.gnu.-org/licenses/old-licenses/lgpl-2.-1.txt
Libraw1394	LGPLv2.0 License http://www.gnu.-org/licenses/old-licenses/lgpl-2.-0.txt

Table 1.1: License table

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

Chapter 4

Class Index

4.1 Class Hierarchy

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

Chapter 5

Class Index

5.1 Class List

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

Chapter 6

Module Documentation

6.1 Enumerations

Enumerations

- enum ErrorType { Undefined = -1, Ok, Failed, NotImplemented, FailedBusMasterConnection, NotConnected, InitFailed, NotInitialized, InvalidParameter, InvalidSettings, InvalidBuManager, MemoryAllocationFailed, LowLevelFailure, NotFound, FailedGuid, InvalidPacketSize, InvalidMode, NotInFormat7, × NotSupported, Timeout, BusMasterFailed, InvalidGeneration, LutFailed, × IidcFailed, StrobeFailed, TriggerFailed, PropertyFailed, PropertyNotPresent, RegisterFailed, ReadRegisterFailed, WriteRegisterFailed, IsochFailed, × IsochAlreadyStarted, IsochNotStarted, IsochStartFailed, IsochRetrieveBufferFailed, IsochStopFailed, IsochSyncFailed, IsochBandwidthExceeded, ImageConversionFailed, ImageLibraryFailure, BufferTooSmall, ImageConsistencyError, IncompatibleDriver }

  The error types returned by functions.
- enum ManagedCallbackType { BusReset, Arrival, Removal }

  The type of bus callback to register a callback function for.
- enum GrabMode { DropFrames, BufferFrames, Unspecified = -2 }

  The grab strategy employed during image transfer.
- enum GrabTimeout { None = 0, Infinite = -1, Unspecified = -2 }

  Timeout options for grabbing images.
- enum BandwidthAllocation { Off = 0, On = 1, Unsupported = 2, Unspecified = -2 }

  Bandwidth allocation options for 1394 devices.
- enum InterfaceType { Ieee1394, Usb2, Usb3, GigE, Unknown = -1 }

  Interfaces that a camera may use to communicate with a host.
- enum PropertyType { Brightness, AutoExposure, Sharpness, WhiteBalance, Hue, Saturation, Gamma, Iris, Focus, Zoom, Pan, Tilt, Shutter, Gain, TriggerMode, TriggerDelay, FrameRate, Temperature, Unspecified = -2 }

  Camera properties.

- enum FrameRate { FrameRate1_875, FrameRate3_75, FrameRate7_5, FrameRate15, FrameRate30, FrameRate60, FrameRate120, FrameRate240, FrameRateFormat7, NumberOfFrameRates }

  Frame rates in frames per second.
- enum VideoMode { VideoMode160x120Yuv444, VideoMode320x240Yuv422, VideoMode640x480Yuv411, VideoMode640x480Yuv422, VideoMode640x480Rgb, VideoMode640x480Y8, VideoMode640x480Y16, VideoMode800x600Yuv422, VideoMode800x600Rgb, VideoMode800x600Y8, VideoMode800x600Y16, VideoMode1024x768Yuv422, VideoMode1024x768Rgb, VideoMode1024x768Y8, VideoMode1024x768Y16, VideoMode1280x960Yuv422, VideoMode1280x960Rgb, VideoMode1280x960Y8, VideoMode1280x960Y16, VideoMode1600x1200Yuv422, VideoMode1600x1200Rgb, VideoMode1600x1200Y8, VideoMode1600x1200Y16, VideoModeFormat7, NumberOfVideoModes }

  DCAM video modes.
- enum Mode { Mode0 = 0, Mode1, Mode2, Mode3, Mode4, Mode5, Mode6, Mode7, Mode8, Mode9, Mode10, Mode11, Mode12, Mode13, Mode14, Mode15, Mode16, Mode17, Mode18, Mode19, Mode20, Mode21, Mode22, Mode23, Mode24, Mode25, Mode26, Mode27, Mode28, Mode29, Mode30, Mode31, NumberOfModes }

  Camera modes for DCAM formats as well as Format7.
- enum PixelFormat { PixelFormatMono8 = 0x80000000, PixelFormat411Yuv8 = 0x40000000, PixelFormat422Yuv8 = 0x20000000, PixelFormat444Yuv8 = 0x10000000, PixelFormatRgb8 = 0x08000000, PixelFormatMono16 = 0x04000000, PixelFormatRgb16 = 0x02000000, PixelFormatSignedMono16 = 0x01000000, PixelFormatSignedRgb16 = 0x00800000, PixelFormatRaw8 = 0x00400000, PixelFormatRaw16 = 0x00200000, PixelFormatMono12 = 0x00100000, PixelFormatRaw12 = 0x00080000, PixelFormatBgr = 0x80000008, PixelFormatBgru = 0x40000008, PixelFormatRgb = PixelFormatRgb8, PixelFormatRgbu = 0x40000002, PixelFormatBgr16 = 0x02000001, PixelFormatBgru16 = 0x02000002, PixelFormat422Yuv8Jpeg = 0x40000001, NumberOfPixelFormats = 20 }

  Pixel formats available for Format7 modes.
- enum BusSpeed { S100, S200, S400, S480, S800, S1600, S3200, S5000, GigE_10Base_T, GigE_100Base_T, GigE_1000Base_T, GigE_10000Base_T, Fastest, Any, Unknown = -1 }

  Bus speeds.
- enum PCIeBusSpeed { Speed_2_5, Speed_5_0, Unknown = -1 }

  PCIe Bus Speeds.
- enum DriverType { Ieee1394_Cam, Ieee1394_Pro, Ieee1394_Juju, Ieee1394_Video1394, Ieee1394_Raw1394, Usb_None, Usb_Cam, Usb3_Pro, GigE_None, GigE_Filter, GigE_Pro, GigE_Lwf, Unknown = -1 }

  Types of low level drivers that flycapture uses.
- enum ColorProcessingAlgorithm { Default, NoColorProcessing, NearestNeighbor, EdgeSensing, HQLinear, Rigorous, IPP, Directional, WeightedDirectional }

  Color processing algorithms.

- enum BayerTileFormat { None = 0, RGGB, GRBG, GBRG, BGGR }

  Bayer tile formats.
- enum ImageFileFormat { FromFileExtension = -1, Pgm, Ppm, Bmp, Jpeg, Jpeg2000, Tiff, Png, Raw }

  File formats to be used for saving images to disk.
- enum GigEPropertyType { Heartbeat, HeartbeatTimeout, PacketSize, PacketDelay }

  Possible properties that can be queried from the camera.
- enum StatisticsChannel { Grey, Red, Green, Blue, Hue, Saturation, Lightness, NumberOfStatisticsChannels }

  Channels that allow statistics to be calculated.
- enum OSType { WindowsX86, WindowsX64, LinuxX86, LinuxX64, Mac, UnknownOS }

  Possible operating systems.
- enum ByteOrder { LittleEndian, BigEndian }

  Possible byte orders.

6.1.1 Enumeration Type Documentation

6.1.1.1 enum BandwidthAllocation

Bandwidth allocation options for 1394 devices.

Enumerator:

Off Do not allocate bandwidth.

On Allocate bandwidth. This is the default setting.

Unsupported Bandwidth allocation is not supported by either the camera or operating system.

Unspecified Unspecified grab mode. Unspecified property type.
Not specified.
Unspecified timeout setting.
This leaves the current setting unchanged.

6.1.1.2 enum BayerTileFormat

Bayer tile formats.

Enumerator:

None Non-blocking wait. No bayer tile format.

RGGB Red-Green-Green-Blue.

GRBG Green-Red-Blue-Green.

GBRG Green-Blue-Red-Green.

BGGR Blue-Green-Green-Red.

6.1.1.3 enum BusSpeed

Bus speeds.

Enumerator:

S100 100Mbits/sec.

S200 200Mbits/sec.

S400 400Mbits/sec.

S480 480Mbits/sec. Only for USB2 cameras.

S800 800Mbits/sec.

S1600 1600Mbits/sec.

S3200 3200Mbits/sec.

S5000 5000Mbits/sec. Only for USB3 cameras.

GigE_10Base_T

GigE_100Base_T

GigE_1000Base_T

GigE_10000Base_T

Fastest The fastest speed available.

Any Any speed that is available.

Unknown Unknown interface. Unknown driver type.

5.0 Gb/s

Unknown bus speed.

Speed is unknown

6.1.1.4 enum ByteOrder

Possible byte orders.

Enumerator:

LittleEndian

BigEndian

6.1.1.5 enum ColorProcessingAlgorithm

Color processing algorithms.

Please refer to our knowledge base at article at http://www.ptgrey.-com/support/kb/index.asp?a=4&q=33 for complete details for each algorithm.

Enumerator:

Default Default method.

NoColorProcessing No color processing.

NearestNeighbor Fastest but lowest quality. Equivalent to FLYCAPTURE_NEAREST_NEIGHBOR_FAST in FlyCapture.

EdgeSensing Weights surrounding pixels based on localized edge orientation.

HQLinear Similar quality to rigorous but much faster.

Rigorous Slowest but produces the best results.

IPP Multithreaded with similar results to edge sensing.

Directional Best quality but much faster than rigorous.

WeightedDirectional Weighted pixel average from different directions.

6.1.1.6 enum DriverType

Types of low level drivers that flycapture uses.

Enumerator:

Ieee1394_Cam PGRCam.sys.

Ieee1394_Pro PGR1394.sys.

Ieee1394_Juju firewire_core.

Ieee1394_Video1394 video1394.

Ieee1394_Raw1394 raw1394.

Usb_None No usb driver used just BSD stack. (Linux only)

Usb_Cam PGRUsbCam.sys.

Usb3_Pro PGRXHCI.sys.

GigE_None no gige drivers used,MS/BSD stack.

GigE_Filter PGRGigE.sys.

GigE_Pro PGRGigEPro.sys.

GigE_Lwf PgrLwf.sys.

Unknown Unknown interface. Unknown driver type.
5.0 Gb/s
Unknown bus speed.
Speed is unknown

6.1.1.7 enum ErrorType

The error types returned by functions.

Enumerator:

Undefined Undefined.

Ok Function returned with no errors.

Failed General failure.

NotImplemented Function has not been implemented.

FailedBusMasterConnection Could not connect to Bus Master.

NotConnected Camera has not been connected.

InitFailed Initialization failed.

NotInitialized Camera has not been initialized.

InvalidParameter Invalid parameter passed to function.

InvalidSettings Setting set to camera is invalid.

InvalidBuManager Invalid Bus Manager object.

MemoryAllocationFailed Could not allocate memory.

LowLevelFailure Low level error.

NotFound Device not found.

FailedGuid GUID failure.

InvalidPacketSize Packet size set to camera is invalid.

InvalidMode Invalid mode has been passed to function.

NotInFormat7 Error due to not being in Format7.

NotSupported This feature is unsupported.

Timeout Timeout error.

BusMasterFailed Bus Master Failure.

InvalidGeneration Generation Count Mismatch.

LutFailed Look Up Table failure.

IidcFailed IIDC failure.

StrobeFailed Strobe failure.

TriggerFailed Trigger failure.

PropertyFailed Property failure.

PropertyNotPresent Property is not present.

RegisterFailed Register access failed.

ReadRegisterFailed Register read failed.

WriteRegisterFailed Register write failed.

IsochFailed Isochronous failure.

IsochAlreadyStarted Isochronous transfer has already been started.

IsochNotStarted Isochronous transfer has not been started.

IsochStartFailed Isochronous start failed.

IsochRetrieveBufferFailed Isochronous retrieve buffer failed.

IsochStopFailed Isochronous stop failed.

IsochSyncFailed Isochronous image synchronization failed.

IsochBandwidthExceeded Isochronous bandwidth exceeded.

ImageConversionFailed Image conversion failed.

ImageLibraryFailure Image library failure.

BufferTooSmall Buffer is too small.

ImageConsistencyError There is an image consistency error.

IncompatibleDriver IncompatibleDriver error.

6.1.1.8 enum FrameRate

Frame rates in frames per second.

Enumerator:

FrameRate1_875 1.875 fps.

FrameRate3_75 3.75 fps.

FrameRate7_5 7.5 fps.

FrameRate15 15 fps.

FrameRate30 30 fps.

FrameRate60 60 fps.

FrameRate120 120 fps.

FrameRate240 240 fps.

FrameRateFormat7 Custom frame rate for Format7 functionality.

NumberOfFrameRates Number of possible camera frame rates.

6.1.1.9 enum GigEPropertyType

Possible properties that can be queried from the camera.

Enumerator:

Heartbeat

HeartbeatTimeout

PacketSize

PacketDelay

6.1.1.10 enum GrabMode

The grab strategy employed during image transfer.

This type controls how images that stream off the camera accumulate in a user buffer for handling. Unlike earlier versions of the FlyCapture SDK, it is no longer necessary to explicitly start the image grabbing process before specifying an image grabbing mode.

Enumerator:

DropFrames Grabs the newest image in the user buffer each time the RetrieveBuffer() function is called. Older images are dropped instead of accumulating in the user buffer. Grabbing blocks if the camera has not finished transmitting the next available image. If the camera is transmitting images faster than the application can grab them, images may be dropped and only the most recent image is stored for grabbing. Note that this mode is the equivalent of flycaptureLockLatest in earlier versions of the FlyCapture SDK.

BufferFrames Images accumulate in the user buffer, and the oldest image is grabbed for handling before being discarded. This member can be used to guarantee that each image is seen. However, image processing time must not exceed transmission time from the camera to the buffer. Grabbing blocks if the camera has not finished transmitting the next available image. The buffer size is controlled by the numBuffers parameter in the FC2Config struct. Note that this mode is the equivalent of flycaptureLockNext in earlier versions of the FlyCapture SDK.

Unspecified Unspecified grab mode. Unspecified property type.

Not specified.

Unspecified timeout setting.

This leaves the current setting unchanged.

6.1.1.11 enum GrabTimeout

Timeout options for grabbing images.

Enumerator:

None Non-blocking wait. No bayer tile format.

Infinite Wait indefinitely.

Unspecified Unspecified grab mode. Unspecified property type.

Not specified.

Unspecified timeout setting.

This leaves the current setting unchanged.

6.1.1.12 enum ImageFileFormat

File formats to be used for saving images to disk.

Enumerator:

FromFileExtension Determine file format from file extension.

Pgm Portable gray map.

Ppm Portable pixmap.

Bmp Bitmap.

Jpeg JPEG.

Jpeg2000 JPEG 2000.

Tiff Tagged image file format.

Png Portable network graphics.

Raw Raw data.

6.1.1.13 enum InterfaceType

Interfaces that a camera may use to communicate with a host.

Enumerator:

Ieee1394 IEEE-1394 (Includes 1394a and 1394b).

Usb2 USB 2.0.

Usb3 USB 3.0.

GigE GigE.

Unknown Unknown interface. Unknown driver type.

5.0 Gb/s

Unknown bus speed.

Speed is unknown

6.1.1.14 enum ManagedCallbackType

The type of bus callback to register a callback function for.

Enumerator:

BusReset Register for all bus events.

Arrival Register for arrivals only.

Removal Register for removals only.

6.1.1.15 enum Mode

Camera modes for DCAM formats as well as Format7.

Enumerator:

Mode0

Mode1

Mode2

Mode3

Mode4

Mode5
Mode6
Mode7
Mode8
Mode9
Mode10
Mode11
Mode12
Mode13
Mode14
Mode15
Mode16
Mode17
Mode18
Mode19
Mode20
Mode21
Mode22
Mode23
Mode24
Mode25
Mode26
Mode27
Mode28
Mode29
Mode30
Mode31
NumberOfModes

6.1.1.16 enum OSType

Possible operating systems.

Enumerator:

WindowsX86 All Windows 32-bit variants.
WindowsX64 All Windows 64-bit variants.
LinuxX86 All Linux 32-bit variants.
LinuxX64 All Linux 32-bit variants.
Mac Mac OSX.
UnknownOS Unknown operating system.

6.1.1.17 enum PCIeBusSpeed

PCIe Bus Speeds.

Enumerator:

Speed_2_5

Speed_5_0 2.5 Gb/s

Unknown Unknown interface. Unknown driver type.

5.0 Gb/s

Unknown bus speed.

Speed is unknown

6.1.1.18 enum PixelFormat

Pixel formats available for Format7 modes.

Enumerator:

PixelFormatMono8 8 bits of mono information.

PixelFormat411Yuv8 YUV 4:1:1.

PixelFormat422Yuv8 YUV 4:2:2.

PixelFormat444Yuv8 YUV 4:4:4.

PixelFormatRgb8 R = G = B = 8 bits.

PixelFormatMono16 16 bits of mono information.

PixelFormatRgb16 R = G = B = 16 bits.

PixelFormatSignedMono16 16 bits of signed mono information.

PixelFormatSignedRgb16 R = G = B = 16 bits signed.

PixelFormatRaw8 8 bit raw data output of sensor.

PixelFormatRaw16 16 bit raw data output of sensor.

PixelFormatMono12 12 bits of mono information.

PixelFormatRaw12 12 bit raw data output of sensor.

PixelFormatBgr 24 bit BGR.

PixelFormatBgru 32 bit BGRU.

PixelFormatRgb 24 bit RGB.

PixelFormatRgbu 32 bit RGBU.

PixelFormatBgr16 R = G = B = 16 bits.

PixelFormatBgru16 64 bit BGRU.

PixelFormat422Yuv8Jpeg JPEG compressed stream.

NumberOfPixelFormats Number of pixel formats.

6.1.1.19 enum PropertyType

Camera properties.

Not all properties may be supported, depending on the camera model.

Enumerator:

Brightness Brightness.
AutoExposure Auto exposure.
Sharpness Sharpness.
WhiteBalance White balance.
Hue Hue.
Saturation Saturation.
Gamma Gamma.
Iris Iris.
Focus Focus.
Zoom Zoom.
Pan Pan.
Tilt Tilt.
Shutter Shutter.
Gain Gain.
TriggerMode Trigger mode.
TriggerDelay Trigger delay.
FrameRate Frame rate.
Temperature Temperature.
Unspecified Unspecified grab mode. Unspecified property type.
Not specified.
Unspecified timeout setting.
This leaves the current setting unchanged.

6.1.1.20 enum StatisticsChannel

Channels that allow statistics to be calculated.

Enumerator:

Grey
Red
Green
Blue
Hue Hue.
Saturation Saturation.
Lightness
NumberOfStatisticsChannels

6.1.1.21 enum VideoMode

DCAM video modes.

Enumerator:

VideoMode160x120Yuv444 160x120 YUV444.

VideoMode320x240Yuv422 320x240 YUV422.

VideoMode640x480Yuv411 640x480 YUV411.

VideoMode640x480Yuv422 640x480 YUV422.

VideoMode640x480Rgb 640x480 24-bit RGB.

VideoMode640x480Y8 640x480 8-bit.

VideoMode640x480Y16 640x480 16-bit.

VideoMode800x600Yuv422 800x600 YUV422.

VideoMode800x600Rgb 800x600 RGB.

VideoMode800x600Y8 800x600 8-bit.

VideoMode800x600Y16 800x600 16-bit.

VideoMode1024x768Yuv422 1024x768 YUV422.

VideoMode1024x768Rgb 1024x768 RGB.

VideoMode1024x768Y8 1024x768 8-bit.

VideoMode1024x768Y16 1024x768 16-bit.

VideoMode1280x960Yuv422 1280x960 YUV422.

VideoMode1280x960Rgb 1280x960 RGB.

VideoMode1280x960Y8 1280x960 8-bit.

VideoMode1280x960Y16 1280x960 16-bit.

VideoMode1600x1200Yuv422 1600x1200 YUV422.

VideoMode1600x1200Rgb 1600x1200 RGB.

VideoMode1600x1200Y8 1600x1200 8-bit.

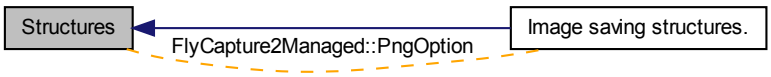
VideoMode1600x1200Y16 1600x1200 16-bit.

VideoModeFormat7 Custom video mode for Format7 functionality.

NumberOfVideoModes Number of possible video modes.

6.2 Structures

Collaboration diagram for Structures:



Classes

- struct FC2Version

  The current version of the library.
- struct GigEProperty

  A GigE property.
- struct GigEStreamChannel

  Information about a single GigE stream channel.
- struct GigEConfig

  Configuration for a GigE camera.
- struct GigEImageSettingsInfo

  Format 7 information for a single mode.
- struct GigEImageSettings

  Image settings for a GigE camera.
- struct FC2Config

  Configuration for a camera.
- struct CameraPropertyInfo

  Information about a specific camera property.
- struct CameraProperty

  A specific camera property.
- struct TriggerModeInfo

  Information about a camera trigger property.
- struct TriggerMode

  A camera trigger.
- struct StrobeInfo

  A camera strobe property.
- struct StrobeControl

  A camera strobe.
- struct Format7ImageSettings

  Format 7 image settings.
- struct Format7Info

Format 7 information for a single mode.

- struct Format7PacketInfo

  Format 7 packet information.

- struct TimeStamp

  Timestamp information.

- struct ConfigROM

  Camera configuration ROM.

- struct CameraInfo

  Camera information.

- struct EmbeddedImageInfoProperty

  Properties of a single embedded image info property.

- struct EmbeddedImageInfo

  Properties of the possible embedded image information.

- struct ImageMetadata

  Metadata related to an image.

- struct LutData

  Information about the camera's look up table.

- struct CameraStats

  Camera diagnostic information.

- struct PngOption

  Options for saving PNG images.

Modules

- Image saving structures.

  These structures define various parameters used for saving images.

6.3 Image saving structures.

These structures define various parameters used for saving images.

Collaboration diagram for Image saving structures.:

Classes

- struct PngOption

  Options for saving PNG images.
- struct PpmOption

  Options for saving PPM images.
- struct PgmOption

  Options for saving PGM images.
- struct TiffOption

  Options for saving TIFF images.
- struct JpegOption

  Options for saving JPEG image.
- struct Jpg2Option

  Options for saving JPEG2000 image.
- struct BMPOption

  Options for saving Bitmap image.
- struct SystemInfo

  Description of the system.

6.3.1 Detailed Description

These structures define various parameters used for saving images.

Chapter 7

Namespace Documentation

7.1 FlyCapture2 Namespace Reference

7.2 FlyCapture2Managed Namespace Reference

Namespaces

- namespace Gui

Classes

- class FC2Exception

  Exception that is thrown when an error is encountered.
- class ManagedBusManager

  ManagedBusManager provides the functionality for the user to get an PGRGuid for a desired camera or device easily.
- class ManagedCamera

  ManagedCamera represents a physical camera that uses the IIDC register set.
- struct ManagedEventCallbackData
- struct ManagedEventOptions

  Options for enabling device event registration.
- struct NativeEventStruct
- class ManagedCameraBase

  Abstract base class that represents a generic camera that defines a general interface to a camera.
- struct FC2Version

  The current version of the library.
- struct GigEProperty

  A GigE property.
- struct GigEStreamChannel

Information about a single GigE stream channel.

- struct GigEConfig

  Configuration for a GigE camera.
- struct GigEImageSettingsInfo

  Format 7 information for a single mode.
- struct GigEImageSettings

  Image settings for a GigE camera.
- struct FC2Config

  Configuration for a camera.
- struct CameraPropertyInfo

  Information about a specific camera property.
- struct CameraProperty

  A specific camera property.
- struct TriggerModeInfo

  Information about a camera trigger property.
- struct TriggerMode

  A camera trigger.
- struct StrobeInfo

  A camera strobe property.
- struct StrobeControl

  A camera strobe.
- struct Format7ImageSettings

  Format 7 image settings.
- struct Format7Info

  Format 7 information for a single mode.
- struct Format7PacketInfo

  Format 7 packet information.
- struct TimeStamp

  Timestamp information.
- struct ConfigROM

  Camera configuration ROM.
- struct CameraInfo

  Camera information.
- struct EmbeddedImageInfoProperty

  Properties of a single embedded image info property.
- struct EmbeddedImageInfo

  Properties of the possible embedded image information.
- struct ImageMetadata

  Metadata related to an image.
- struct LutData

  Information about the camera's look up table.
- struct CameraStats

  Camera diagnostic information.

- struct PngOption

  Options for saving PNG images.
- struct PpmOption

  Options for saving PPM images.
- struct PgmOption

  Options for saving PGM images.
- struct TiffOption

  Options for saving TIFF images.
- struct JpegOption

  Options for saving JPEG image.
- struct Jpg2Option

  Options for saving JPEG2000 image.
- struct BMPOption

  Options for saving Bitmap image.
- struct SystemInfo

  Description of the system.
- class ManagedGCCamera
- class ManagedGCPort
- class ManagedGigECamera

  The GigECamera object represents a physical Gigabit Ethernet camera.
- class ManagedImage

  The ManagedImageImage class is used to retrieve images from a camera, convert between multiple pixel formats and save images to disk.
- class ManagedImageStatistics
- class ManagedPGRGuid

  Managed version of a PGRGuid.
- class ManagedTopologyNode

  The ManagedTopologyNode class contains topology information that can be used to generate a tree structure of all cameras and devices connected to a computer.
- class ManagedUtilities
- class Translate

Enumerations

- enum ErrorType { Undefined = -1, Ok, Failed, NotImplemented, FailedBusMasterConnection, NotConnected, InitFailed, NotInitialized, InvalidParameter, InvalidSettings, InvalidBuManager, MemoryAllocationFailed, LowLevelFailure, NotFound, FailedGuid, InvalidPacketSize, InvalidMode, NotInFormat7, NotSupported, Timeout, BusMasterFailed, InvalidGeneration, LutFailed, IidcFailed, StrobeFailed, TriggerFailed, PropertyFailed, PropertyNotPresent, RegisterFailed, ReadRegisterFailed, WriteRegisterFailed, IsochFailed, IsochAlreadyStarted, IsochNotStarted, IsochStartFailed, IsochRetrieveBufferFailed, IsochStopFailed, IsochSyncFailed, IsochBandwidthExceeded, ImageConversionFailed, ImageLibraryFailure, BufferTooSmall, ImageConsistencyError, IncompatibleDriver }

The error types returned by functions.

- enum ManagedCallbackType { BusReset, Arrival, Removal }

  The type of bus callback to register a callback function for.

- enum GrabMode { DropFrames, BufferFrames, Unspecified = -2 }

  The grab strategy employed during image transfer.

- enum GrabTimeout { None = 0, Infinite = -1, Unspecified = -2 }

  Timeout options for grabbing images.

- enum BandwidthAllocation { Off = 0, On = 1, Unsupported = 2, Unspecified = -2 }

  Bandwidth allocation options for 1394 devices.

- enum InterfaceType { Ieee1394, Usb2, Usb3, GigE, Unknown = -1 }

  Interfaces that a camera may use to communicate with a host.

- enum PropertyType { Brightness, AutoExposure, Sharpness, WhiteBalance, Hue, Saturation, Gamma, Iris, Focus, Zoom, Pan, Tilt, Shutter, Gain, TriggerMode, TriggerDelay, FrameRate, Temperature, Unspecified = -2 }

  Camera properties.

- enum FrameRate { FrameRate1_875, FrameRate3_75, FrameRate7_5, FrameRate15, FrameRate30, FrameRate60, FrameRate120, FrameRate240, FrameRateFormat7, NumberOfFrameRates }

  Frame rates in frames per second.

- enum VideoMode { VideoMode160x120Yuv444, VideoMode320x240Yuv422, VideoMode640x480Yuv411, VideoMode640x480Yuv422, VideoMode640x480Rgb, VideoMode640x480Y8, VideoMode640x480Y16, VideoMode800x600Yuv422, VideoMode800x600Rgb, VideoMode800x600Y8, VideoMode800x600Y16, VideoMode1024x768Yuv422, VideoMode1024x768Rgb, VideoMode1024x768Y8, VideoMode1024x768Y16, VideoMode1280x960Yuv422, VideoMode1280x960Rgb, VideoMode1280x960Y8, VideoMode1280x960Y16, VideoMode1600x1200Yuv422, VideoMode1600x1200Rgb, VideoMode1600x1200Y8, VideoMode1600x1200Y16, VideoModeFormat7, NumberOfVideoModes }

  DCAM video modes.

- enum Mode { Mode0 = 0, Mode1, Mode2, Mode3, Mode4, Mode5, Mode6, Mode7, Mode8, Mode9, Mode10, Mode11, Mode12, Mode13, Mode14, Mode15, Mode16, Mode17, Mode18, Mode19, Mode20, Mode21, Mode22, Mode23, Mode24, Mode25, Mode26, Mode27, Mode28, Mode29, Mode30, Mode31, NumberOfModes }

  Camera modes for DCAM formats as well as Format7.

- enum PixelFormat { PixelFormatMono8 = 0x80000000, PixelFormat411Yuv8 = 0x40000000, PixelFormat422Yuv8 = 0x20000000, PixelFormat444Yuv8 = 0x10000000, PixelFormatRgb8 = 0x08000000, PixelFormatMono16 = 0x04000000, PixelFormatRgb16 = 0x02000000, PixelFormatSignedMono16 = 0x01000000, PixelFormatSignedRgb16 = 0x00800000, PixelFormatRaw8 = 0x00400000, PixelFormatRaw16 = 0x00200000, PixelFormatMono12 = 0x00100000, PixelFormatRaw12 = 0x00080000, PixelFormatBgr = 0x80000008, PixelFormatBgru = 0x40000008, PixelFormatRgb = PixelFormatRgb8, PixelFormatRgbu = 0x40000002, PixelFormatBgr16 = 0x02000001, PixelFormatBgru16 = 0x02000002, PixelFormat422Yuv8Jpeg = 0x40000001, NumberOfPixelFormats = 20 }

Pixel formats available for Format7 modes.

- enum BusSpeed { S100, S200, S400, S480, S800, S1600, S3200, S5000, GigE_10Base_T, GigE_100Base_T, GigE_1000Base_T, GigE_10000Base_T, Fastest, Any, Unknown = -1 }

  Bus speeds.
- enum PCIeBusSpeed { Speed_2_5, Speed_5_0, Unknown = -1 }

  PCIe Bus Speeds.
- enum DriverType { Ieee1394_Cam, Ieee1394_Pro, Ieee1394_Juju, Ieee1394_Video1394, Ieee1394_Raw1394, Usb_None, Usb_Cam, Usb3_Pro, GigE_None, GigE_Filter, GigE_Pro, GigE_Lwf, Unknown = -1 }

  Types of low level drivers that flycapture uses.
- enum ColorProcessingAlgorithm { Default, NoColorProcessing, NearestNeighbor, EdgeSensing, HQLinear, Rigorous, IPP, Directional, WeightedDirectional }

  Color processing algorithms.
- enum BayerTileFormat { None = 0, RGGB, GRBG, GBRG, BGGR }

  Bayer tile formats.
- enum ImageFileFormat { FromFileExtension = -1, Pgm, Ppm, Bmp, Jpeg, Jpeg2000, Tiff, Png, Raw }

  File formats to be used for saving images to disk.
- enum GigEPropertyType { Heartbeat, HeartbeatTimeout, PacketSize, PacketDelay }

  Possible properties that can be queried from the camera.
- enum StatisticsChannel { Grey, Red, Green, Blue, Hue, Saturation, Lightness, NumberOfStatisticsChannels }

  Channels that allow statistics to be calculated.
- enum OSType { WindowsX86, WindowsX64, LinuxX86, LinuxX64, Mac, UnknownOS }

  Possible operating systems.
- enum ByteOrder { LittleEndian, BigEndian }

  Possible byte orders.

Functions

- public delegate void EnumCallback (System::IntPtr parameter, unsigned int serialNumber)

  Bus event callback function prototype.
- public delegate void ImageEventCallback (ManagedImage^ image)

  The external callback that will be used by managed consumers.
- protected delegate void ImageCallbackDelegate (FlyCapture2::Image *image, void *data)

  Internal callback that we use internally so we can create the proper external callback for users.
- public delegate void ManagedCameraEventCallback (ManagedEventCallbackData^ data)

The external callback that will be used by managed consumers.

- protected delegate void ManagedCameraEventCallbackDelegate (void ∗data)

  Internal callback that we use internally so we can create the proper proper external callback for users.
- unsigned long htonl (unsigned long data)
- public delegate void AsyncCommandCallback (bool retError)

  The external callback that will be used by managed consumers.
- protected delegate void CommandCallbackDelegate (FlyCapture2::Error retError, void ∗pUserData)

  Internal callback that we use internally so we can create the proper external callback for users.

7.2.1 Function Documentation

7.2.1.1 public delegate void FlyCapture2Managed::AsyncCommandCallback (bool *retError*)

The external callback that will be used by managed consumers.

7.2.1.2 protected delegate void FlyCapture2Managed::CommandCallbackDelegate (FlyCapture2::Error *retError,* void ∗ *pUserData*)

Internal callback that we use internally so we can create the proper external callback for users.

7.2.1.3 public delegate void FlyCapture2Managed::EnumCallback (System::IntPtr *parameter,* unsigned int *serialNumber*)

Bus event callback function prototype.

Defines the syntax of the callback function that is passed into RegisterCallback() and UnregisterCallback().

7.2.1.4 unsigned long FlyCapture2Managed::htonl (unsigned long *data*)

7.2.1.5 protected delegate void FlyCapture2Managed::ImageCallbackDelegate (FlyCapture2::Image ∗ *image,* void ∗ *data*)

Internal callback that we use internally so we can create the proper external callback for users.

7.2.1.6 public delegate void FlyCapture2Managed::ImageEventCallback (ManagedImage^ *image*)

The external callback that will be used by managed consumers.

7.2.1.7 public delegate void FlyCapture2Managed::ManagedCameraEventCallback (ManagedEventCallbackData^ *data*)

The external callback that will be used by managed consumers.

7.2.1.8 protected delegate void FlyCapture2Managed::ManagedCameraEventCallbackDelegate (void * *data*)

Internal callback that we use internally so we can create the proper proper external callback for users.

7.3 FlyCapture2Managed::Gui Namespace Reference

Classes

- class CameraControlDialog

  CameraControlDialog: managed wrapper of FlyCapture2::CameraControlDialog (see for details)
- class CameraSelectionDialog

  CameraControlDialog: managed wrapper of FlyCapture2::CameraSelectionDialog (see for details)

Chapter 8

Class Documentation

8.1 BMPOption Struct Reference

Options for saving Bitmap image.

Public Member Functions

- BMPOption ()

Properties

- bool indexedColor_8bit

8.1.1 Detailed Description

Options for saving Bitmap image.

8.1.2 Constructor & Destructor Documentation

8.1.2.1 BMPOption ( ) [inline]

8.1.3 Property Documentation

8.1.3.1 bool indexedColor_8bit

8.2 CameraControlDialog Class Reference

CameraControlDialog: managed wrapper of FlyCapture2::CameraControlDialog (see for details)

Public Member Functions

- CameraControlDialog ()
- ~CameraControlDialog ()
- void Connect (ManagedCameraBase^ camera)
- void Disconnect ()
- void Show ()
- void Hide ()
- bool IsVisible ()
- void SetTitle (System::String^ title)

8.2.1 Detailed Description

CameraControlDialog: managed wrapper of FlyCapture2::CameraControlDialog (see for details)

8.2.2 Constructor & Destructor Documentation

8.2.2.1 CameraControlDialog ( )

8.2.2.2 ~CameraControlDialog ( )

8.2.3 Member Function Documentation

8.2.3.1 void Connect (FlyCapture2Managed::ManagedCameraBase^ *camera*)

8.2.3.2 void Disconnect (void)

8.2.3.3 void Hide ( )

8.2.3.4 bool IsVisible ( )

8.2.3.5 void SetTitle (System::String^ *title*)

8.2.3.6 void Show ( )

8.3 CameraInfo Struct Reference

Camera information.

Properties

- unsigned int serialNumber

  Device serial number.
- InterfaceType interfaceType

Interface type.

- DriverType driverType

  Driver type.

- bool isColorCamera

  Flag indicating if this is a color camera.

- System::String^ modelName

  Device model name.

- System::String^ vendorName

  Device vendor name.

- System::String^ sensorInfo

  String detailing the sensor information.

- System::String^ sensorResolution

  String providing the sensor resolution.

- System::String^ driverName

  Driver name of driver being used.

- System::String^ firmwareVersion

  Firmware version of camera.

- System::String^ firmwareBuildTime

  Firmware build time.

- BusSpeed maximumBusSpeed

  Maximum bus speed.

- PCIeBusSpeed pcieBusSpeed

  Maximum PCIe bus speed.

- BayerTileFormat bayerTileFormat

  Bayer tile format.

- unsigned short busNumber

  Bus Number, set to 0 for USB and GigE.

- unsigned short nodeNumber

  Node Number, set to 0 for USB and GigE.

IIDC specific information

- unsigned int iidcVersion

  DCAM version.

- ConfigROM configROM

  Configuration ROM data.

GigE specific information

- unsigned int gigEMajorVersion

  GigE Vision version.

- unsigned int gigEMinorVersion

  GigE Vision minor version.

- System::String^ userDefinedName

  User defined name.

- System::String^ xmlURL1

XML URL 1.

- System::String^ xmlURL2

  XML URL 2.
- System::Net::NetworkInformation::PhysicalAddress^ macAddress

  MAC address.
- System::Net::IPAddress^ ipAddress

  IP address.
- System::Net::IPAddress^ subnetMask

  Subnet mask.
- System::Net::IPAddress^ defaultGateway

  Default gateway.
- unsigned int ccpStatus

  Status/Content of CCP register.
- unsigned int applicationIPAddress

  Local Application IP Address.
- unsigned int applicationPort

  Local Application port.

8.3.1 Detailed Description

Camera information.

8.3.2 Property Documentation

8.3.2.1 unsigned int applicationIPAddress

Local Application IP Address.

8.3.2.2 unsigned int applicationPort

Local Application port.

8.3.2.3 BayerTileFormat bayerTileFormat

Bayer tile format.

8.3.2.4 unsigned short busNumber

Bus Number, set to 0 for USB and GigE.

8.3.2.5 unsigned int ccpStatus

Status/Content of CCP register.

8.3.2.6 ConfigROM configROM

Configuration ROM data.

8.3.2.7 System:: Net:: IPAddress^ defaultGateway

Default gateway.

8.3.2.8 System:: String^ driverName

Driver name of driver being used.

8.3.2.9 DriverType driverType

Driver type.

8.3.2.10 System:: String^ firmwareBuildTime

Firmware build time.

8.3.2.11 System:: String^ firmwareVersion

Firmware version of camera.

8.3.2.12 unsigned int gigEMajorVersion

GigE Vision version.

8.3.2.13 unsigned int gigEMinorVersion

GigE Vision minor version.

8.3.2.14 unsigned int iidcVersion

DCAM version.

8.3.2.15 InterfaceType interfaceType

Interface type.

8.3.2.16 System:: Net:: IPAddress^ ipAddress

IP address.

8.3.2.17 bool isColorCamera

Flag indicating if this is a color camera.

8.3.2.18 System:: Net:: NetworkInformation:: PhysicalAddress^ macAddress

MAC address.

8.3.2.19 BusSpeed maximumBusSpeed

Maximum bus speed.

8.3.2.20 System:: String^ modelName

Device model name.

8.3.2.21 unsigned short nodeNumber

Node Number, set to 0 for USB and GigE.

8.3.2.22 PCIeBusSpeed pcieBusSpeed

Maximum PCIe bus speed.

8.3.2.23 System:: String^ sensorInfo

String detailing the sensor information.

8.3.2.24 System:: String^ sensorResolution

String providing the sensor resolution.

8.3.2.25 unsigned int serialNumber

Device serial number.

8.3.2.26 System:: Net:: IPAddress^ subnetMask

Subnet mask.

8.3.2.27 System:: String^ userDefinedName

User defined name.

8.3.2.28 System:: String^ vendorName

Device vendor name.

8.3.2.29 System:: String^ xmlURL1

XML URL 1.

8.3.2.30 System:: String^ xmlURL2

XML URL 2.

8.4 CameraProperty Struct Reference

A specific camera property.

Public Member Functions

- CameraProperty ()
- CameraProperty (PropertyType type)

Properties

- PropertyType type

  Property info type.
- bool present

  Flag indicating if the property is present.
- bool absControl

  Flag controlling absolute mode.
- bool onePush

  Flag controlling one push.
- bool onOff

  Flag controlling on/off.

- bool autoManualMode

  Flag controlling auto.
- unsigned int valueA

  Value A (integer).
- unsigned int valueB

  Value B (integer).
- float absValue

  Floating point value.

8.4.1 Detailed Description

A specific camera property.

8.4.2 Constructor & Destructor Documentation

8.4.2.1 CameraProperty ( ) [inline]

8.4.2.2 CameraProperty (PropertyType *type*) [inline]

8.4.3 Property Documentation

8.4.3.1 bool absControl

Flag controlling absolute mode.

8.4.3.2 float absValue

Floating point value.

8.4.3.3 bool autoManualMode

Flag controlling auto.

8.4.3.4 bool onePush

Flag controlling one push.

8.4.3.5 bool onOff

Flag controlling on/off.

8.4.3.6 bool present

Flag indicating if the property is present.

8.4.3.7 PropertyType type

Property info type.

8.4.3.8 unsigned int valueA

Value A (integer).

8.4.3.9 unsigned int valueB

Value B (integer).

Applies only to the white balance blue value. Use Value A for the red value.

8.5 CameraPropertyInfo Struct Reference

Information about a specific camera property.

Public Member Functions

- CameraPropertyInfo ()
- CameraPropertyInfo (PropertyType type)

Properties

- PropertyType type
  Property info type.
- bool present
  Flag indicating if the property is present.
- bool autoSupported
  Flag indicating if auto is supported.
- bool manualSupported
  Flag indicating if manual is supported.
- bool onOffSupported
  Flag indicating if on/off is supported.
- bool onePushSupported
  Flag indicating if one push is supported.
- bool absValSupported

Flag indicating if absolute mode is supported.

- bool readOutSupported

  Flag indicating if property value can be read out.

- unsigned int min

  Minimum value (as an integer).

- unsigned int max

  Maximum value (as an integer).

- float absMin

  Minimum value (as a floating point value).

- float absMax

  Maximum value (as a floating point value).

- System::String^ units

  Textual description of units.

- System::String^ unitAbbr

  Abbreviated textual description of units.

8.5.1 Detailed Description

Information about a specific camera property.

This structure is also also used as the TriggerDelayInfo structure.

8.5.2 Constructor & Destructor Documentation

8.5.2.1 CameraPropertyInfo ( ) [inline]

8.5.2.2 CameraPropertyInfo (PropertyType *type*) [inline]

8.5.3 Property Documentation

8.5.3.1 float absMax

Maximum value (as a floating point value).

8.5.3.2 float absMin

Minimum value (as a floating point value).

8.5.3.3 bool absValSupported

Flag indicating if absolute mode is supported.

8.5.3.4 bool autoSupported

Flag indicating if auto is supported.

8.5.3.5 bool manualSupported

Flag indicating if manual is supported.

8.5.3.6 unsigned int max

Maximum value (as an integer).

8.5.3.7 unsigned int min

Minimum value (as an integer).

8.5.3.8 bool onePushSupported

Flag indicating if one push is supported.

8.5.3.9 bool onOffSupported

Flag indicating if on/off is supported.

8.5.3.10 bool present

Flag indicating if the property is present.

8.5.3.11 bool readOutSupported

Flag indicating if property value can be read out.

8.5.3.12 PropertyType type

Property info type.

8.5.3.13 System:: String^ unitAbbr

Abbreviated textual description of units.

8.5.3.14 System:: String^ units

Textual description of units.

8.6 CameraSelectionDialog Class Reference

CameraControlDialog: managed wrapper of FlyCapture2::CameraSelectionDialog (see for details)

Public Member Functions

- CameraSelectionDialog ()
- ~CameraSelectionDialog ()
- bool ShowModal ()

  Show the CameraSelectionDlg.
- array< ManagedPGRGuid^ >^ GetSelectedCameraGuids ()

  Returns the list of camera guids selected by the user while in ShowModal()
- void SetTitle (System::String^ title)

  Set the window title.

8.6.1 Detailed Description

CameraControlDialog: managed wrapper of FlyCapture2::CameraSelectionDialog (see for details)

8.6.2 Constructor & Destructor Documentation

8.6.2.1 CameraSelectionDialog ( )

8.6.2.2 ~CameraSelectionDialog ( )

8.6.3 Member Function Documentation

8.6.3.1 array< ManagedPGRGuid^ > GetSelectedCameraGuids ( )

Returns the list of camera guids selected by the user while in ShowModal()

Returns

Array of PGRGuids identifying the selected cameras.

8.6.3.2 void SetTitle (System::String^ *title*)

Set the window title.

Parameters

pTitle	Null-terminated string representing the title.

8.6.3.3 bool ShowModal ( )

Show the CameraSelectionDlg.

Returns

Whether Ok (true) or Cancel (false) was clicked.

8.7 CameraStats Struct Reference

Camera diagnostic information.

Public Member Functions

- CameraStats ()

Properties

- unsigned int imageDropped
- unsigned int imageCorrupt
- unsigned int imageXmitFailed
- unsigned int imageDriverDropped
- unsigned int regReadFailed
- unsigned int regWriteFailed
- unsigned int portErrors
- bool cameraPowerUp
- List< float >^ cameraVoltages
- unsigned int numVoltages

  The number of voltage registers available.
- List< float >^ cameraCurrents
- unsigned int numCurrents

  The number of current registers available.
- unsigned int temperature
- unsigned int timeSinceInitialization
- unsigned int timeSinceBusReset
- TimeStamp^ timeStamp
- unsigned int numResendPacketsRequested
- unsigned int numResendPacketsReceived

8.7.1 Detailed Description

Camera diagnostic information.

8.7.2 Constructor & Destructor Documentation

8.7.2.1 CameraStats ( ) [inline]

8.7.3 Property Documentation

8.7.3.1 List< float>^ cameraCurrents

8.7.3.2 bool cameraPowerUp

8.7.3.3 List< float>^ cameraVoltages

8.7.3.4 unsigned int imageCorrupt

8.7.3.5 unsigned int imageDriverDropped

8.7.3.6 unsigned int imageDropped

8.7.3.7 unsigned int imageXmitFailed

8.7.3.8 unsigned int numCurrents

The number of current registers available.

0: the values in cameraCurrents[] are invalid.

8.7.3.9 unsigned int numResendPacketsReceived

8.7.3.10 unsigned int numResendPacketsRequested

8.7.3.11 unsigned int numVoltages

The number of voltage registers available.

0: the values in cameraVoltages[] are invalid.

8.7.3.12 unsigned int portErrors

8.7.3.13 unsigned int regReadFailed

8.7.3.14 unsigned int regWriteFailed

8.7.3.15 unsigned int temperature

8.7.3.16 unsigned int timeSinceBusReset

8.7.3.17 unsigned int timeSinceInitialization

8.7.3.18 TimeStamp^ timeStamp

8.8 ConfigROM Struct Reference

Camera configuration ROM.

Properties

- unsigned int nodeVendorId

  Vendor ID of a node.
- unsigned int chipIdHi

  Chip ID (high part).
- unsigned int chipIdLo

  Chip ID (low part).
- unsigned int unitSpecId

  Unit Spec ID, usually 0xa02d.
- unsigned int unitSWVer

  Unit software version.
- unsigned int unitSubSWVer

  Unit sub software version.
- unsigned int vendorUniqueInfo0

  Vendor unique info 0.
- unsigned int vendorUniqueInfo1

  Vendor unique info 1.
- unsigned int vendorUniqueInfo2

  Vendor unique info 2.
- unsigned int vendorUniqueInfo3

  Vendor unique info 3.
- System::String^ keyword

  Keyword.

8.8.1 Detailed Description

Camera configuration ROM.

8.8.2 Property Documentation

8.8.2.1 unsigned int chipIdHi

Chip ID (high part).

8.8.2.2 unsigned int chipIdLo

Chip ID (low part).

8.8.2.3 System:: String^ keyword

Keyword.

8.8.2.4 unsigned int nodeVendorId

Vendor ID of a node.

8.8.2.5 unsigned int unitSpecId

Unit Spec ID, usually 0xa02d.

8.8.2.6 unsigned int unitSubSWVer

Unit sub software version.

8.8.2.7 unsigned int unitSWVer

Unit software version.

8.8.2.8 unsigned int vendorUniqueInfo0

Vendor unique info 0.

8.8.2.9 unsigned int vendorUniqueInfo1

Vendor unique info 1.

8.8.2.10 unsigned int vendorUniqueInfo2

Vendor unique info 2.

8.8.2.11 unsigned int vendorUniqueInfo3

Vendor unique info 3.

8.9 EmbeddedImageInfo Struct Reference

Properties of the possible embedded image information.

Public Member Functions

- EmbeddedImageInfo ()

Properties

- EmbeddedImageInfoProperty^ timestamp
- EmbeddedImageInfoProperty^ gain
- EmbeddedImageInfoProperty^ shutter
- EmbeddedImageInfoProperty^ brightness
- EmbeddedImageInfoProperty^ exposure
- EmbeddedImageInfoProperty^ whiteBalance
- EmbeddedImageInfoProperty^ frameCounter
- EmbeddedImageInfoProperty^ strobePattern
- EmbeddedImageInfoProperty^ GPIOPinState
- EmbeddedImageInfoProperty^ ROIPosition

8.9.1 Detailed Description

Properties of the possible embedded image information.

8.9.2 Constructor & Destructor Documentation

8.9.2.1 EmbeddedImageInfo ( ) [inline]

8.9.3 Property Documentation

8.9.3.1 EmbeddedImageInfoProperty^ brightness

8.9.3.2 EmbeddedImageInfoProperty^ exposure

8.9.3.3 EmbeddedImageInfoProperty^ frameCounter

8.9.3.4 EmbeddedImageInfoProperty^ gain

8.9.3.5 EmbeddedImageInfoProperty^ GPIOPinState

8.9.3.6 EmbeddedImageInfoProperty^ ROIPosition

8.9.3.7 EmbeddedImageInfoProperty^ shutter

8.9.3.8 EmbeddedImageInfoProperty^ strobePattern

8.9.3.9 EmbeddedImageInfoProperty^ timestamp

8.9.3.10 EmbeddedImageInfoProperty^ whiteBalance

8.10 EmbeddedImageInfoProperty Struct Reference

Properties of a single embedded image info property.

Properties

- bool available

  Whether this property is available.
- bool onOff

  Whether this property is on or off.

8.10.1 Detailed Description

Properties of a single embedded image info property.

8.10.2 Property Documentation

8.10.2.1 bool available

Whether this property is available.

8.10.2.2 bool onOff

Whether this property is on or off.

8.11 FC2Config Struct Reference

Configuration for a camera.

Public Member Functions

- FC2Config ()

Properties

- unsigned int numBuffers

  Number of buffers used by the FlyCapture2 library to grab images.
- unsigned int numImageNotifications

  Number of notifications per image.
- unsigned int minNumImageNotifications

  Minimum number of notifications needed for the current image settings on the camera.
- int grabTimeout

  Time in milliseconds that RetrieveBuffer() and WaitForBufferEvent() will wait for an image before timing out and returning.
- GrabMode grabMode

  Grab mode for the camera.
- bool highPerformanceRetrieveBuffer

  This parameter enables RetrieveBuffer to run in high performance mode.
- BusSpeed isochBusSpeed

  Isochronous bus speed.
- BusSpeed asyncBusSpeed

  Asynchronous bus speed.
- BandwidthAllocation bandwidthAllocation

  Bandwidth allocation flag that tells the camera the bandwidth allocation strategy to employ.
- unsigned int registerTimeoutRetries

  Number of retries to perform when a register read/write timeout is received by the library.
- unsigned int registerTimeout

  Register read/write timeout value, in microseconds.

8.11.1 Detailed Description

Configuration for a camera.

These options are options that are generally should be set before starting isochronous transfer.

8.11.2 Constructor & Destructor Documentation

8.11.2.1 FC2Config ( ) [inline]

8.11.3 Property Documentation

8.11.3.1 BusSpeed asyncBusSpeed

Asynchronous bus speed.

8.11.3.2 BandwidthAllocation bandwidthAllocation

Bandwidth allocation flag that tells the camera the bandwidth allocation strategy to employ.

8.11.3.3 GrabMode grabMode

Grab mode for the camera.

The default is DROP_FRAMES.

8.11.3.4 int grabTimeout

Time in milliseconds that RetrieveBuffer() and WaitForBufferEvent() will wait for an image before timing out and returning.

8.11.3.5 bool highPerformanceRetrieveBuffer

This parameter enables RetrieveBuffer to run in high performance mode.

This means that any interaction with the camera, other then grabbing the image is disabled. Currently Retrieve buffer reads registers on the camera to determine which embedded image information settings have been enabled, and it reads what the bayer tile is currently set to. When High Performance mode is on, these reads are disabled. - This means that any changes to the Bayer Tile or to the Embedded image info after StartCapture() will not be tracked when made using direct register writes. If the corresponding SetEmbededImageInfo() and GetEmbededImageInfo() calls are used then the changes will be appropriately reflected. This also means that changes to embedded image info from other processes will not be updated either.

8.11.3.6 BusSpeed isochBusSpeed

Isochronous bus speed.

8.11.3.7 unsigned int minNumImageNotifications

Minimum number of notifications needed for the current image settings on the camera.

Read-only value.

8.11.3.8 unsigned int numBuffers

Number of buffers used by the FlyCapture2 library to grab images.

8.11.3.9 unsigned int numImageNotifications

Number of notifications per image.

This value should only be set after the image settings to be used is set to the camera. The default number of notifications is 1.

There are 4 general scenarios:

- 1 notification - End of image
- 2 notifications - After first packet and end of image
- 3 notifications - After first packet, middle of image, end of image
- x notifications - After first packet, (x -2) spread evenly, end of image

8.11.3.10 unsigned int registerTimeout

Register read/write timeout value, in microseconds.

The default value is dependent on the interface type.

8.11.3.11 unsigned int registerTimeoutRetries

Number of retries to perform when a register read/write timeout is received by the library.

The default value is 0.

8.12 FC2Exception Class Reference

Exception that is thrown when an error is encountered.

Public Member Functions

- FC2Exception ()
- FC2Exception (String^ string)
- FC2Exception (String^ string, Exception^ exception)
- ~FC2Exception ()

Protected Member Functions

- FC2Exception (Runtime::Serialization::SerializationInfo^ serializationInfo, - Runtime::Serialization::StreamingContext context)

Package Functions

- FC2Exception (FlyCapture2::Error *error)

Properties

- ErrorType Type [get]
- ErrorType CauseType [get]
- String^ NativeErrorTrace [get]

8.12.1 Detailed Description

Exception that is thrown when an error is encountered.

This is used instead of returning an Error object as used in the C++ interface.

8.12.2 Constructor & Destructor Documentation

8.12.2.1 FC2Exception ( )

8.12.2.2 FC2Exception (String^ *string*)

8.12.2.3 FC2Exception (String^ *string,* Exception^ *exception*)

8.12.2.4 ~FC2Exception ( )

8.12.2.5 FC2Exception (Runtime::Serialization::SerializationInfo^ *serializationInfo,* Runtime::Serialization::StreamingContext *context*) [protected]

8.12.2.6 FC2Exception (FlyCapture2::Error * *error*) [package]

8.12.3 Property Documentation

8.12.3.1 ErrorType CauseType [get]

8.12.3.2 String^ NativeErrorTrace [get]

8.12.3.3 ErrorType Type [get]

8.13 FC2Version Struct Reference

The current version of the library.

Properties

- unsigned int major

  Major version number.
- unsigned int minor

  Minor version number.
- unsigned int type

  Type version number.
- unsigned int build

  Build version number.

8.13.1 Detailed Description

The current version of the library.

8.13.2 Property Documentation

8.13.2.1 unsigned int build

Build version number.

8.13.2.2 unsigned int major

Major version number.

8.13.2.3 unsigned int minor

Minor version number.

8.13.2.4 unsigned int type

Type version number.

8.14 Format7ImageSettings Struct Reference

Format 7 image settings.

Properties

- Mode mode

  Format 7 mode.
- unsigned int offsetX

  Horizontal image offset.
- unsigned int offsetY

  Vertical image offset.
- unsigned int width

  Width of image.
- unsigned int height

  Height of image.
- PixelFormat pixelFormat

  Pixel format of image.

8.14.1 Detailed Description

Format 7 image settings.

8.14.2 Property Documentation

8.14.2.1 unsigned int height

Height of image.

8.14.2.2 Mode mode

Format 7 mode.

8.14.2.3 unsigned int offsetX

Horizontal image offset.

8.14.2.4 unsigned int offsetY

Vertical image offset.

8.14.2.5 PixelFormat pixelFormat

Pixel format of image.

8.14.2.6 unsigned int width

Width of image.

8.15 Format7Info Struct Reference

Format 7 information for a single mode.

Properties

- Mode mode

  Format 7 mode.
- unsigned int maxWidth

  Maximum image width.
- unsigned int maxHeight

  Maximum image height.
- unsigned int offsetHStepSize

  Horizontal step size for the offset.
- unsigned int offsetVStepSize

  Vertical step size for the offset.
- unsigned int imageHStepSize

  Horizontal step size for the image.
- unsigned int imageVStepSize

  Vertical step size for the image.
- unsigned int pixelFormatBitField

  Supported pixel formats in a bit field.
- unsigned int vendorPixelFormatBitField

  Vendor unique pixel formats in a bit field.
- unsigned int packetSize

  Current packet size in bytes.
- unsigned int minPacketSize

  Minimum packet size in bytes for current mode.
- unsigned int maxPacketSize

  Maximum packet size in bytes for current mode.
- float percentage

  Current packet size as a percentage of maximum packet size.

8.15.1 Detailed Description

Format 7 information for a single mode.

8.15.2 Property Documentation

8.15.2.1 unsigned int imageHStepSize

Horizontal step size for the image.

8.15.2.2 unsigned int imageVStepSize

Vertical step size for the image.

8.15.2.3 unsigned int maxHeight

Maximum image height.

8.15.2.4 unsigned int maxPacketSize

Maximum packet size in bytes for current mode.

8.15.2.5 unsigned int maxWidth

Maximum image width.

8.15.2.6 unsigned int minPacketSize

Minimum packet size in bytes for current mode.

8.15.2.7 Mode mode

Format 7 mode.

8.15.2.8 unsigned int offsetHStepSize

Horizontal step size for the offset.

8.15.2.9 unsigned int offsetVStepSize

Vertical step size for the offset.

8.15.2.10 unsigned int packetSize

Current packet size in bytes.

8.15.2.11 float percentage

Current packet size as a percentage of maximum packet size.

8.15.2.12 unsigned int pixelFormatBitField

Supported pixel formats in a bit field.

8.15.2.13 unsigned int vendorPixelFormatBitField

Vendor unique pixel formats in a bit field.

8.16 Format7PacketInfo Struct Reference

Format 7 packet information.

Properties

- unsigned int recommendedBytesPerPacket
  Recommended bytes per packet.
- unsigned int maxBytesPerPacket
  Maximum bytes per packet.
- unsigned int unitBytesPerPacket
  Minimum bytes per packet.

8.16.1 Detailed Description

Format 7 packet information.

8.16.2 Property Documentation

8.16.2.1 unsigned int maxBytesPerPacket

Maximum bytes per packet.

8.16.2.2 unsigned int recommendedBytesPerPacket

Recommended bytes per packet.

8.16.2.3 unsigned int unitBytesPerPacket

Minimum bytes per packet.

8.17 GigEConfig Struct Reference

Configuration for a GigE camera.

Properties

- bool enablePacketResend

  Turn on/off packet resend functionality.

8.17.1 Detailed Description

Configuration for a GigE camera.

These options are options that are generally should be set before starting isochronous transfer.

8.17.2 Property Documentation

8.17.2.1 bool enablePacketResend

Turn on/off packet resend functionality.

8.18 GigEImageSettings Struct Reference

Image settings for a GigE camera.

Properties

- unsigned int offsetX

  Horizontal image offset.
- unsigned int offsetY

  Vertical image offset.
- unsigned int width

  Width of image.
- unsigned int height

  Height of image.
- PixelFormat pixelFormat

  Pixel format of image.

8.18.1 Detailed Description

Image settings for a GigE camera.

8.18.2 Property Documentation

8.18.2.1 unsigned int height

Height of image.

8.18.2.2 unsigned int offsetX

Horizontal image offset.

8.18.2.3 unsigned int offsetY

Vertical image offset.

8.18.2.4 PixelFormat pixelFormat

Pixel format of image.

8.18.2.5 unsigned int width

Width of image.

8.19 GigEImageSettingsInfo Struct Reference

Format 7 information for a single mode.

Properties

- unsigned int maxWidth
  Maximum image width.
- unsigned int maxHeight
  Maximum image height.
- unsigned int offsetHStepSize
  Horizontal step size for the offset.
- unsigned int offsetVStepSize
  Vertical step size for the offset.
- unsigned int imageHStepSize
  Horizontal step size for the image.
- unsigned int imageVStepSize
  Vertical step size for the image.
- unsigned int pixelFormatBitField
  Supported pixel formats in a bit field.

- unsigned int vendorPixelFormatBitField

  Vendor unique pixel formats in a bit field.

8.19.1 Detailed Description

Format 7 information for a single mode.

8.19.2 Property Documentation

8.19.2.1 unsigned int imageHStepSize

Horizontal step size for the image.

8.19.2.2 unsigned int imageVStepSize

Vertical step size for the image.

8.19.2.3 unsigned int maxHeight

Maximum image height.

8.19.2.4 unsigned int maxWidth

Maximum image width.

8.19.2.5 unsigned int offsetHStepSize

Horizontal step size for the offset.

8.19.2.6 unsigned int offsetVStepSize

Vertical step size for the offset.

8.19.2.7 unsigned int pixelFormatBitField

Supported pixel formats in a bit field.

8.19.2.8 unsigned int vendorPixelFormatBitField

Vendor unique pixel formats in a bit field.

8.20 GigEProperty Struct Reference

A GigE property.

Properties

- GigEPropertyType propType

  The type of property.
- bool isReadable

  Whether the property is readable.
- bool isWritable

  Whether the property is writable.
- unsigned int min

  Minimum value.
- unsigned int max

  Maximum value.
- unsigned int value

  Current value.

8.20.1 Detailed Description

A GigE property.

8.20.2 Property Documentation

8.20.2.1 bool isReadable

Whether the property is readable.

If this is false, then no other value in this structure is valid.

8.20.2.2 bool isWritable

Whether the property is writable.

8.20.2.3 unsigned int max

Maximum value.

8.20.2.4 unsigned int min

Minimum value.

8.20.2.5 GigEPropertyType propType

The type of property.

8.20.2.6 unsigned int value

Current value.

8.21 GigEStreamChannel Struct Reference

Information about a single GigE stream channel.

Properties

- unsigned int networkInterfaceIndex

  Network interface index used (or to use).
- unsigned int hostPort

  Host port on the PC where the camera will send the data stream.
- bool doNotFragment

  Disable IP fragmentation of packets.
- unsigned int packetSize

  Packet size, in bytes.
- unsigned int interPacketDelay

  Inter packet delay, in timestamp counter units.
- System::Net::IPAddress^ destinationIpAddress

  Destination IP address.
- unsigned int sourcePort

  Source UDP port of the stream channel.

8.21.1 Detailed Description

Information about a single GigE stream channel.

8.21.2 Property Documentation

8.21.2.1 System:: Net:: IPAddress^ destinationIpAddress

Destination IP address.

It can be a multicast or unicast address.

8.21.2.2 bool doNotFragment

Disable IP fragmentation of packets.

8.21.2.3 unsigned int hostPort

Host port on the PC where the camera will send the data stream.

8.21.2.4 unsigned int interPacketDelay

Inter packet delay, in timestamp counter units.

8.21.2.5 unsigned int networkInterfaceIndex

Network interface index used (or to use).

8.21.2.6 unsigned int packetSize

Packet size, in bytes.

8.21.2.7 unsigned int sourcePort

Source UDP port of the stream channel.

Read only.

8.22 ImageMetadata Struct Reference

Metadata related to an image.

Properties

- unsigned int embeddedTimeStamp

  Embedded timestamp.
- unsigned int embeddedGain

  Embedded gain.
- unsigned int embeddedShutter

  Embedded shutter.
- unsigned int embeddedBrightness

  Embedded brightness.
- unsigned int embeddedExposure

  Embedded exposure.

- unsigned int embeddedWhiteBalance

  Embedded white balance.
- unsigned int embeddedFrameCounter

  Embedded frame counter.
- unsigned int embeddedStrobePattern

  Embedded strobe pattern.
- unsigned int embeddedGPIOPinState

  Embedded GPIO pin state.
- unsigned int embeddedROIPosition

  Embedded ROI position.

8.22.1 Detailed Description

Metadata related to an image.

8.22.2 Property Documentation

8.22.2.1 unsigned int embeddedBrightness

Embedded brightness.

8.22.2.2 unsigned int embeddedExposure

Embedded exposure.

8.22.2.3 unsigned int embeddedFrameCounter

Embedded frame counter.

8.22.2.4 unsigned int embeddedGain

Embedded gain.

8.22.2.5 unsigned int embeddedGPIOPinState

Embedded GPIO pin state.

8.22.2.6 unsigned int embeddedROIPosition

Embedded ROI position.

8.22.2.7 unsigned int embeddedShutter

Embedded shutter.

8.22.2.8 unsigned int embeddedStrobePattern

Embedded strobe pattern.

8.22.2.9 unsigned int embeddedTimeStamp

Embedded timestamp.

8.22.2.10 unsigned int embeddedWhiteBalance

Embedded white balance.

8.23 JpegOption Struct Reference

Options for saving JPEG image.

Public Member Functions

- JpegOption ()

Properties

- bool progressive

  Whether to save as a progressive JPEG file.
- unsigned int quality

  JPEG image quality in range (0-100).

8.23.1 Detailed Description

Options for saving JPEG image.

8.23.2 Constructor & Destructor Documentation

8.23.2.1 JpegOption ( ) [inline]

8.23.3 Property Documentation

8.23.3.1 bool progressive

Whether to save as a progressive JPEG file.

8.23.3.2 unsigned int quality

JPEG image quality in range (0-100).

- 100 - Superb quality.
- 75 - Good quality.
- 50 - Normal quality.
- 10 - Poor quality.

8.24 Jpg2Option Struct Reference

Options for saving JPEG2000 image.

Public Member Functions

- Jpg2Option ()

Properties

- unsigned int quality

  JPEG saving quality in range (1-512).

8.24.1 Detailed Description

Options for saving JPEG2000 image.

8.24.2 Constructor & Destructor Documentation

8.24.2.1 Jpg2Option ( ) [inline]

8.24.3 Property Documentation

8.24.3.1 unsigned int quality

JPEG saving quality in range (1-512).

8.25 LutData Struct Reference

Information about the camera's look up table.

Properties

- bool supported

  Flag indicating if LUT is supported.
- bool enabled

  Flag indicating if LUT is enabled.
- unsigned int numBanks

  The number of LUT banks available (Always 1 for PGR LUT).
- unsigned int numChannels

  The number of LUT channels per bank available.
- unsigned int inputBitDepth

  The input bit depth of the LUT.
- unsigned int outputBitDepth

  The output bit depth of the LUT.
- unsigned int numEntries

  The number of entries in the LUT.

8.25.1 Detailed Description

Information about the camera's look up table.

8.25.2 Property Documentation

8.25.2.1 bool enabled

Flag indicating if LUT is enabled.

8.25.2.2 unsigned int inputBitDepth

The input bit depth of the LUT.

8.25.2.3 unsigned int numBanks

The number of LUT banks available (Always 1 for PGR LUT).

8.25.2.4 unsigned int numChannels

The number of LUT channels per bank available.

8.25.2.5 unsigned int numEntries

The number of entries in the LUT.

8.25.2.6 unsigned int outputBitDepth

The output bit depth of the LUT.

8.25.2.7 bool supported

Flag indicating if LUT is supported.

8.26 ManagedBusManager Class Reference

ManagedBusManager provides the functionality for the user to get an PGRGuid for a desired camera or device easily.

Public Member Functions

- ManagedBusManager ()
- ~ManagedBusManager ()
- void FireBusReset (ManagedPGRGuid^ guid)

  Fire a bus reset.
- unsigned int GetNumOfCameras ()

  Gets the number of cameras attached to the PC.
- ManagedPGRGuid^ GetCameraFromIPAddress (System::Net::IPAddress^ ipAddress)

  Gets the ManagedPGRGuid for a camera with the specified IPv4 address.
- ManagedPGRGuid^ GetCameraFromIndex (unsigned int index)

  Gets the ManagedPGRGuid for a camera on the PC.
- ManagedPGRGuid^ GetCameraFromSerialNumber (unsigned int serialNumber)

  Gets the ManagedPGRGuid for a camera on the PC.
- unsigned int GetCameraSerialNumberFromIndex (unsigned int index)

  Gets the serial number of the camera with the specified index.
- InterfaceType GetInterfaceTypeFromGuid (ManagedPGRGuid^ guid)

  Gets the interface type associated with a ManagedPGRGuid.
- unsigned int GetNumOfDevices ()

  Gets the number of devices.
- ManagedPGRGuid^ GetDeviceFromIndex (unsigned int index)

  Gets the ManagedPGRGuid for a device.
- unsigned int ReadPhyRegister (ManagedPGRGuid^ guid, unsigned int page, unsigned int port, unsigned int address)

Read a phy register on the specified device.

- void WritePhyRegister (ManagedPGRGuid^ guid, unsigned int page, unsigned int port, unsigned int address, unsigned int regVal)

  Write a phy register on the specified device.

- unsigned int GetUsbLinkInfo (ManagedPGRGuid^ guid)

  Read usb link info for the port that the specified device is connected to.

- unsigned int GetUsbPortStatus (ManagedPGRGuid^ guid)

  Read usb port status for the port that the specified device is connected to.

- ManagedTopologyNode^ GetTopology ()

  Gets the topology information for the PC.

- System::IntPtr RegisterCallback (EnumCallback^ hCallbackDelegate, ManagedCallbackType callbackType, System::IntPtr parameter)

  Register a callback function that will be called when the specified callback event occurs.

- void UnregisterCallback (System::IntPtr callbackHandle)

  Unregister a callback function.

- void RescanBus ()

  Force a rescan of the buses.

- bool IsCameraControlable (ManagedPGRGuid^ guid)

  Query CCP status on camera with corresponding PGRGuid.

Static Public Member Functions

- static void ForceIPAddressToCamera (System::Net::NetworkInformation::PhysicalAddress^ macAddress, System::Net::IPAddress^ ipAddress, System::Net::IPAddress^ subnetMask, System::Net::IPAddress^ defaultGateway)

  Force the camera with the specific MAC address to the specified IP address, subnet mask and default gateway.

- static void ForceAllIPAddressesAutomatically ()

  Force all cameras on the network to be assigned sequential IP addresses on the same subnet as the netowrk adapters that they are connected to.

- static void ForceAllIPAddressesAutomatically (unsigned int serialNumber)

  Force a specific cameras on the network to be assigned sequential IP address on the same subnet as the netowrk adapters that it is connected to.

- static array< CameraInfo^ >^ DiscoverGigECameras ()

  Discover all cameras connected to the network even if they reside on a different subnet.

Protected Member Functions

- !ManagedBusManager ()

Static Package Functions

- static void ConvertToNativeGuid (ManagedPGRGuid^ mgdPGRGuid, FlyCapture2::PGRGuid *pgrGuid)

  Convert a ManagedPGRGuid to a native PGRGuid.
- static void ConvertToManagedGuid (FlyCapture2::PGRGuid *pgrGuid, ManagedPGRGuid^ mgdPGRGuid)

  Convert a native PGRGuid to a ManagedPGRGuid.

8.26.1 Detailed Description

ManagedBusManager provides the functionality for the user to get an PGRGuid for a desired camera or device easily.

Once the camera or device token is found, it can then be used to connect to the camera or device through the camera class or device class. In addition, the BusManager class provides the ability to be notified when a camera or device is added or removed or some event occurs on the PC.

8.26.2 Constructor & Destructor Documentation

8.26.2.1 ManagedBusManager ( )

8.26.2.2 ~ManagedBusManager ( )

8.26.2.3 !ManagedBusManager ( ) [protected]

8.26.3 Member Function Documentation

8.26.3.1 void ConvertToManagedGuid (FlyCapture2::PGRGuid * *pgrGuid*, ManagedPGRGuid^ *mgdPGRGuid*) [inline, static, package]

Convert a native PGRGuid to a ManagedPGRGuid.

Parameters

pgrGuid	The native PGRGuid.
mgdPGR-Guid	The ManagedPGRGuid.

8.26.3.2 void ConvertToNativeGuid (ManagedPGRGuid^ *mgdPGRGuid*, FlyCapture2::PGRGuid * *pgrGuid*) [inline, static, package]

Convert a ManagedPGRGuid to a native PGRGuid.

Parameters

mgdPGR-Guid	The ManagedPGRGuid.
pgrGuid	The native PGRGuid.

8.26.3.3 array< CameraInfo^ > DiscoverGigECameras ( ) [static]

Discover all cameras connected to the network even if they reside on a different subnet.

This is useful in situations where a GigE camera is using Persistent IP and the application's subnet is different from the device subnet. After discovering the camera, it is easy to use ForceIPAddressToCamera() to set a different IP configuration.

Returns

Array of CameraInfo structures containing information about discovered cameras.

8.26.3.4 void FireBusReset (ManagedPGRGuid^ *guid*)

Fire a bus reset.

The actual bus reset is only fired for the specified 1394 bus, but it will effectively cause a global bus reset for the library.

Parameters

guid	ManagedPGRGuid of the camera or the device to cause bus reset.

8.26.3.5 void ForceAllIPAddressesAutomatically ( ) [static]

Force all cameras on the network to be assigned sequential IP addresses on the same subnet as the netowrk adapters that they are connected to.

This is useful in situations where a GigE Vision cameras are using Persistent IP addresses and the application's subnet is different from the devices.

8.26.3.6 void ForceAllIPAddressesAutomatically (unsigned int *serialNumber*) [static]

Force a specific cameras on the network to be assigned sequential IP address on the same subnet as the netowrk adapters that it is connected to.

This is useful in situations where a GigE Vision camera is using Persistent IP addresses and the application's subnet is different from the device.

8.26.3.7 void ForceIPAddressToCamera (System::Net::NetworkInformation::PhysicalAddress^ *macAddress*, System::Net::IPAddress^ *ipAddress*, System::Net::IPAddress^ *subnetMask*, System::Net::IPAddress^ *defaultGateway*) [static]

Force the camera with the specific MAC address to the specified IP address, subnet mask and default gateway.

This is useful in situations where a GigE Vision camera is using Persistent IP and the application's subnet is different from the device subnet.

Parameters

macAddress	MAC address of the camera.
ipAddress	IP address to set on the camera.
subnetMask	Subnet mask to set on the camera.
default-Gateway	Default gateway to set on the camera.

8.26.3.8 ManagedPGRGuid GetCameraFromIndex (unsigned int *index*)

Gets the ManagedPGRGuid for a camera on the PC.

It uniquely identifies the camera specified by the index and is used to identify the camera during a ManagedCamera::Connect() call.

Parameters

index	Zero based index of camera.

Returns

Unique ManagedPGRGuid for the camera.

8.26.3.9 ManagedPGRGuid GetCameraFromIPAddress (System::Net::IPAddress^ *ipAddress*)

Gets the ManagedPGRGuid for a camera with the specified IPv4 address.

Parameters

ipAddress	IP address to get ManagedPGRGuid for.

Returns

Unique ManagedPGRGuid for the camera.

8.26.3.10 ManagedPGRGuid GetCameraFromSerialNumber (unsigned int *serialNumber*)

Gets the ManagedPGRGuid for a camera on the PC.

It uniquely identifies the camera specified by the serial number and is used to identify the camera during a ManagedCamera::Connect() call.

Parameters

serial-Number	Serial number of camera.

See also

GetCameraFromIndex()

Returns

Unique ManagedPGRGuid for the camera.

8.26.3.11 unsigned int GetCameraSerialNumberFromIndex (unsigned int *index*)

Gets the serial number of the camera with the specified index.

Parameters

index	Zero based index of desired camera.

Returns

Serial number of camera.

8.26.3.12 ManagedPGRGuid GetDeviceFromIndex (unsigned int *index*)

Gets the ManagedPGRGuid for a device.

It uniquely identifies the device specified by the index.

Parameters

index	Zero based index of device.

See also

GetNumOfDevices()

Returns

Unique ManagedPGRGuid for the device.

8.26.3.13 InterfaceType GetInterfaceTypeFromGuid (ManagedPGRGuid^ *guid*)

Gets the interface type associated with a ManagedPGRGuid.

This is useful in situations where there is a need to enumerate all cameras for a particular interface.

Parameters

guid	The ManagedPGRGuid to get the interface for.

Returns

The interface type of the PGRGuid.

8.26.3.14 unsigned int GetNumOfCameras ( )

Gets the number of cameras attached to the PC.

Returns

The number of cameras attached.

8.26.3.15 unsigned int GetNumOfDevices ( )

Gets the number of devices.

This may include hubs, host controllers and other hardware devices (including cameras).

Returns

The number of devices found.

8.26.3.16 ManagedTopologyNode GetTopology ( )

Gets the topology information for the PC.

Returns

ManagedTopologyNode object that will contain the topology

8.26.3.17 unsigned int GetUsbLinkInfo (ManagedPGRGuid^ *guid*)

Read usb link info for the port that the specified device is connected to.

Parameters

guid	PGRGuid of the device to read from.

Returns

Value read from the card register.

8.26.3.18 unsigned int GetUsbPortStatus (ManagedPGRGuid^ *guid*)

Read usb port status for the port that the specified device is connected to.

Parameters

guid	PGRGuid of the device to read from.

Returns

Value read from the card register.

8.26.3.19 bool IsCameraControlable (ManagedPGRGuid^ *guid*)

Query CCP status on camera with corresponding PGRGuid.

This is useful to determine if a GigE camera can be controlled.

Parameters

pGuid	PGRGuid of the camera

Returns

True means camera is controlable.

8.26.3.20 unsigned int ReadPhyRegister (ManagedPGRGuid^ *guid,* unsigned int *page,* unsigned int *port,* unsigned int *address*)

Read a phy register on the specified device.

The full address to be read from is determined by the page, port and address.

Parameters

guid	ManagedPGRGuid of the device to read from.
page	Page to read from.
port	Port to read from.
address	Address to read from.

Returns

Value read from the phy register.

8.26.3.21 System::IntPtr RegisterCallback (EnumCallback^ *hCallbackDelegate*, ManagedCallbackType *callbackType*, System::IntPtr *parameter*)

Register a callback function that will be called when the specified callback event occurs.

Parameters

hCallback-Delegate	Handle to EnumCallback function to receive the callback.
callbackType	Type of callback to register for.
parameter	Callback parameter to be passed to callback.

See also

UnregisterCallback()

Returns

Unique callback handle used for unregistering callback.

8.26.3.22 void RescanBus ( )

Force a rescan of the buses.

This does not trigger a bus reset. The camera objects will be invalidated only if the camera network topology is changed (ie. a camera is disconnected or added)

8.26.3.23 void UnregisterCallback (System::IntPtr *callbackHandle*)

Unregister a callback function.

Parameters

callback-Handle	Unique callback handle.

See also

RegisterCallback()

8.26.3.24 void WritePhyRegister (ManagedPGRGuid^ *guid*, unsigned int *page*, unsigned int *port*, unsigned int *address*, unsigned int *regVal*)

Write a phy register on the specified device.

The full address to be written to is determined by the page, port and address.

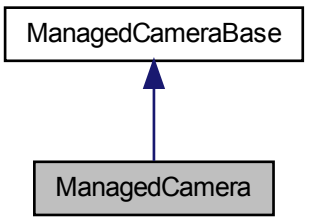
Parameters

guid	ManagedPGRGuid of the device to write to.
page	Page to write to.
port	Port to write to.
address	Address to write to.
regVal	Value to write to phy register.

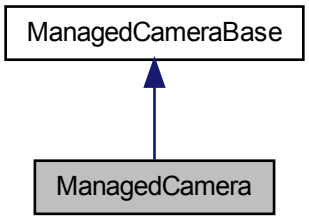
8.27 ManagedCamera Class Reference

ManagedCamera represents a physical camera that uses the IIDC register set.

Inheritance diagram for ManagedCamera:



Collaboration diagram for ManagedCamera:



Public Member Functions

- ManagedCamera ()
- ~ManagedCamera ()
- virtual void Connect (ManagedPGRGuid^ mgdPGRGuid) override

  Connects the ManagedCamera object to the camera specified by the GUID.

Protected Member Functions

- !ManagedCamera ()

DCAM Formats

These functions deal with DCAM video mode and frame rate on the camera.

They are only used for firewire and Usb2 cameras.

- bool GetVideoModeAndFrameRateInfo (VideoMode videoMode, FrameRate frameRate)

  Query the camera to determine if the specified video mode and frame rate is supported.
- void GetVideoModeAndFrameRate (VideoMode% videoMode, FrameRate% frameRate)

  Get the current video mode and frame rate from the camera.
- void SetVideoModeAndFrameRate (VideoMode videoMode, FrameRate frameRate)

  Set the specified video mode and frame rate to the camera.

Format7

These functions deal with Format7 custom image control on the camera.

- Format7Info^ GetFormat7Info (Mode mode, bool% supported)

  Retrieve the availability of Format7 custom image mode and the camera capabilities for the specified Format7 mode.
- Format7PacketInfo^ ValidateFormat7Settings (Format7ImageSettings^ imageSettings, bool% settingsAreValid)

  Validates Format7ImageSettings structure and returns valid packet size information if the image settings are valid.
- void GetFormat7Configuration (Format7ImageSettings^ imageSettings, unsigned int% packetSize, float% percentSpeed)

  Get the current Format7 configuration from the camera.
- void SetFormat7Configuration (Format7ImageSettings^ imageSettings, unsigned int recommendedPacketSize)

  Set the current Format7 configuration to the camera.
- void SetFormat7Configuration (Format7ImageSettings^ imageSettings, float recommendedPercentSpeed)

  Set the current Format7 configuration to the camera.
- static void StartSyncCapture (unsigned int numCameras, array< ManagedCamera^ >^ppCameras)

  Start multiple cameras in synchronization.
- static void StartSyncCapture (unsigned int numCameras, array< ManagedCamera^ >^ppCameras, array< ImageEventCallback^ >^pCallbackFns, array< IntPtr^ >^pCallbackDataArray)

  Start multiple cameras in synchronization using callbacks.

8.27.1 Detailed Description

ManagedCamera represents a physical camera that uses the IIDC register set.

The object must first be connected to using Connect() before any other operations can proceed.

It is possible for more than 1 Camera object to connect to a single physical camera. However, isochronous transmission to more than 1 Camera object is not supported.

8.27.2 Constructor & Destructor Documentation

8.27.2.1 ManagedCamera ( )

8.27.2.2 ~ManagedCamera ( )

8.27.2.3 !ManagedCamera ( ) [protected]

8.27.3 Member Function Documentation

8.27.3.1 void Connect (ManagedPGRGuid^ *mgdPGRGuid*) [override, virtual]

Connects the ManagedCamera object to the camera specified by the GUID.

Parameters

mgdPGR-Guid	The unique identifier for a specific camera on the PC.

See also

ManagedBusManager::GetCameraFromIndex()
ManagedBusManager::GetCameraFromSerialNumber()

Reimplemented from ManagedCameraBase.

8.27.3.2 void GetFormat7Configuration (Format7ImageSettings^ *imageSettings*, unsigned int% *packetSize*, float% *percentSpeed*)

Get the current Format7 configuration from the camera.

This call will only succeed if the camera is already in Format7.

Parameters

image-Settings	Current image settings.
packetSize	Current packet size.
percent-Speed	Current packet size as a percentage.

See also

GetFormat7Info()
ValidateFormat7Settings()
SetFormat7Configuration()
GetVideoModeAndFrameRate()

8.27.3.3 Format7Info GetFormat7Info (Mode *mode*, bool% *supported*)

Retrieve the availability of Format7 custom image mode and the camera capabilities for the specified Format7 mode.

The mode must be specified in the Format7Info structure in order for the function to succeed.

Parameters

mode	Format7 mode to query.
supported	Whether the specified mode is supported.

See also

ValidateFormat7Settings()
GetFormat7Configuration()
SetFormat7Configuration()

Returns

Format7Info structure filled with the capabilities of the specified mode and the current state in the specified mode.

8.27.3.4 void GetVideoModeAndFrameRate (VideoMode% *videoMode*, FrameRate% *frameRate*)

Get the current video mode and frame rate from the camera.

If the camera is in Format7, the video mode will be VIDEOMODE_FORMAT7 and the frame rate will be FRAMERATE_FORMAT7.

Parameters

videoMode	Current video mode.
frameRate	Current frame rate.

See also

GetVideoModeAndFrameRateInfo()
SetVideoModeAndFrameRate()

8.27.3.5 bool GetVideoModeAndFrameRateInfo (VideoMode *videoMode*, FrameRate *frameRate*)

Query the camera to determine if the specified video mode and frame rate is supported.

Parameters

videoMode	Video mode to check.
frameRate	Frame rate to check.

See also

GetVideoModeAndFrameRate()
SetVideoModeAndFrameRate()

Returns

Whether the video mode and frame rate is supported.

8.27.3.6 void SetFormat7Configuration (Format7ImageSettings^ *imageSettings*, unsigned int *recommendedPacketSize*)

Set the current Format7 configuration to the camera.

Parameters

imageSettings	Image settings to be written to the camera.
recommendedPacketSize	Packet size to be written to the camera.

See also

GetFormat7Info()
ValidateFormat7Settings()
GetFormat7Configuration()

8.27.3.7 void SetFormat7Configuration (Format7ImageSettings^ *imageSettings*, float *recommendedPercentSpeed*)

Set the current Format7 configuration to the camera.

Parameters

imageSettings	Image settings to be written to the camera.
recommendedPercentSpeed	Percentage of packet size to be written to the camera.

See also

GetFormat7Info()
ValidateFormat7Settings()
GetFormat7Configuration()

8.27.3.8 void SetVideoModeAndFrameRate (VideoMode *videoMode*, FrameRate *frameRate*)

Set the specified video mode and frame rate to the camera.

It is not possible to set the camera to VIDEOMODE_FORMAT7 or FRAMERATE_FORMAT7. Use the Format7 functions to set the camera into Format7.

Parameters

videoMode	Video mode to set to camera.
frameRate	Frame rate to set to camera.

See also

GetVideoModeAndFrameRateInfo()
GetVideoModeAndFrameRate()

8.27.3.9 void StartSyncCapture (unsigned int *numCameras,* array< ManagedCamera^ >^ *ppCameras*) [static]

Start multiple cameras in synchronization.

This function is only used for firewire cameras.

Parameters

numCameras	Number of cameras to start.
ppCameras	An array of ManagedCamera objects to be started.

See also

StartCapture()

8.27.3.10 void StartSyncCapture (unsigned int *numCameras,* array< ManagedCamera^ >^ *ppCameras,* array< ImageEventCallback^ >^ *pCallbackFns,* array< IntPtr^ >^ *pCallbackDataArray*) [static]

Start multiple cameras in synchronization using callbacks.

This function is only used for firewire cameras.

Parameters

numCameras	Number of cameras to start.
ppCameras	An array of ManagedCamera objects to be started
pCallbackFns	An array of callback functions
pCallbackDataArray	An array of ManagedImage objects to be populated during callback

See also

StartCapture()

8.27.3.11 Format7PacketInfo ValidateFormat7Settings (Format7ImageSettings^ *imageSettings*, bool% *settingsAreValid*)

Validates Format7ImageSettings structure and returns valid packet size information if the image settings are valid.

The current image settings are cached while validation is taking place. The cached settings are restored when validation is complete.

Parameters

imageSettings	Structure containing the image settings.
settingsAreValid	Whether the settings are valid.

See also

GetFormat7Info()
GetFormat7Configuration()
SetFormat7Configuration()

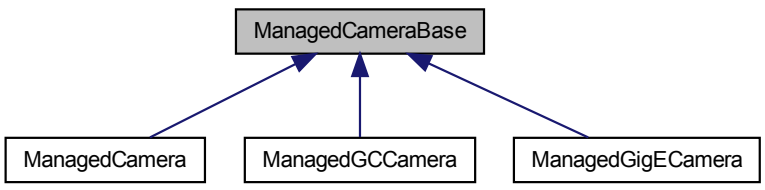
Returns

Packet size information that can be used to determine a valid packet size.

8.28 ManagedCameraBase Class Reference

Abstract base class that represents a generic camera that defines a general interface to a camera.

Inheritance diagram for ManagedCameraBase:



Public Member Functions

- virtual ~ManagedCameraBase ()
- void SetCamera (System::IntPtr otherCamera)

  Set camera from a integer pointer camera.
- virtual TimeStamp^ GetCycleTime ()

  Returns a Timestamp struct containing 1394 CYCLE_TIME information.
- virtual CameraStats^ GetStats ()
- virtual void ResetStats ()
- virtual void RegisterEvent (ManagedEventOptions^ hOpts)
- virtual void DeregisterEvent (ManagedEventOptions^ hOpts)
- virtual void RegisterAllEvents (ManagedEventOptions^ hOpts)
- virtual void DeregisterAllEvents ()

Connection and Image Retrieval

These functions deal with connections and image retrieval from the camera.

- virtual void Connect (ManagedPGRGuid^ mgdPGRGuid)

  Connects the ManagedCamera object to the camera specified by the GUID.
- virtual void Disconnect ()

  Disconnects the ManagedCamera object from the camera.
- virtual bool IsConnected ()

  Checks if the ManagedCamera object is connected to a physical camera specified by a GUID.
- virtual void SetCallback (ImageEventCallback^ hCallbackDelegate)

  Sets the callback data to be used on completion of image transfer.
- virtual void StartCapture ()

  Starts isochronous image capture.
- virtual void StartCapture (ImageEventCallback^ hCallbackDelegate)

  Starts isochronous image capture.
- virtual void StopCapture ()

  Stops isochronous image transfer and cleans up all associated resources.
- virtual void RetrieveBuffer (ManagedImage^ image)

  Retrieves the the next image object containing the next image.
- virtual void WaitForBufferEvent (ManagedImage^ image, unsigned int eventNumber)

  Retrieves the next image event containing the next part of the image.
- virtual void SetUserBuffers (IntPtr pMemBuffers, int size, int numBuffers)

  Specify user allocated buffers to use as image data buffers.
- virtual FC2Config^ GetConfiguration ()

  Get the configuration associated with the camera object.
- virtual void SetConfiguration (FC2Config^ config)

  Set the configuration associated with the camera object.

Information and Properties

These functions deal with information and properties can be retrieved from the camera.

- virtual CameraInfo^ GetCameraInfo ()
  Retrieves information from the camera such as serial number, model name and other camera information.
- virtual CameraPropertyInfo^ GetPropertyInfo (PropertyType type)
  Retrieves information about the specified camera property.
- virtual CameraProperty^ GetProperty (PropertyType type)
  Reads the settings for the specified property from the camera.
- virtual void SetProperty (CameraProperty^ camProperty)
  Writes the settings for the specified property to the camera.
- virtual void SetProperty (CameraProperty^ camProperty, bool broadcast)
  Writes the settings for the specified property to the camera.

General Purpose Input / Output

These functions deal with general GPIO pin control on the camera.

- virtual unsigned int GetGPIOPinDirection (unsigned int pin)
  Get the GPIO pin direction for the specified pin.
- virtual void SetGPIOPinDirection (unsigned int pin, unsigned int direction)
  Set the GPIO pin direction for the specified pin.
- virtual void SetGPIOPinDirection (unsigned int pin, unsigned int direction, bool broadcast)
  Set the GPIO pin direction for the specified pin.

Trigger

These functions deal with trigger control on the camera.

- virtual TriggerModeInfo^ GetTriggerModeInfo ()
  Retrieve trigger information from the camera.
- virtual TriggerMode^ GetTriggerMode ()
  Retrieve current trigger settings from the camera.
- virtual void SetTriggerMode (TriggerMode^ triggerMode)
  Set the specified trigger settings to the camera.
- virtual void FireSoftwareTrigger (bool broadcast)
  Fire the software trigger according to the DCAM specifications.
- virtual CameraPropertyInfo^ GetTriggerDelayInfo ()
  Retrieve trigger delay information from the camera.
- virtual CameraProperty^ GetTriggerDelay ()
  Retrieve current trigger delay settings from the camera.
- virtual void SetTriggerDelay (CameraProperty^ triggerDelay)
  Set the specified trigger delay settings to the camera.
- virtual void SetTriggerDelay (CameraProperty^ triggerDelay, bool broadcast)
  Set the specified trigger delay settings to the camera.

Strobe

These functions deal with strobe control on the camera.

- virtual StrobeInfo^ GetStrobeInfo (unsigned int source)
  Retrieve strobe information from the camera.
- virtual StrobeControl^ GetStrobe (unsigned int source)
  Retrieve current strobe settings from the camera.
- virtual void SetStrobe (StrobeControl^ strobeControl)
  Set current strobe settings to the camera.

Look Up Table

These functions deal with Look Up Table control on the camera.

Note that some cameras may report support for the LUT and return an inputBitDepth of 0. In these cases use log2(numEntries) for the inputBitDepth.

- virtual LutData^ GetLUTInfo ()
  Query if LUT support is available on the camera.
- virtual void GetLUTBankInfo (unsigned int bank, bool% readSupported, bool% writeSupported)
  Query the read/write status of a single LUT bank.
- virtual unsigned int GetActiveLUTBank ()
  Get the LUT bank that is currently being used.
- virtual void SetActiveLUTBank (unsigned int activeBank)
  Set the LUT bank that will be used.
- virtual void EnableLUT (bool on)
  Enable or disable LUT functionality on the camera.
- virtual void GetLUTChannel (unsigned int bank, unsigned int channel, unsigned int sizeEntries, array< unsigned int >^entries)
  Get the LUT channel settings from the camera.
- virtual void SetLUTChannel (unsigned int bank, unsigned int channel, unsigned int sizeEntries, array< unsigned int >^entries)
  Set the LUT channel settings to the camera.

Memory Channels

These functions deal with memory channel control on the camera.

- virtual unsigned int GetMemoryChannel ()
  Retrieve the current memory channel from the camera.
- virtual void SaveToMemoryChannel (unsigned int channel)
  Save the current settings to the specfied current memory channel.
- virtual void RestoreFromMemoryChannel (unsigned int channel)
  Restore the specfied current memory channel.
- virtual unsigned int GetMemoryChannelInfo ()
  Query the camera for memory channel support.

Embedded Image Information

These functions deal with embedded image information control on the camera.

- virtual EmbeddedImageInfo^ GetEmbeddedImageInfo ()
  Get the current status of the embedded image information register, as well as the availability of each embedded property.
- virtual void SetEmbeddedImageInfo (EmbeddedImageInfo^ info)
  Sets the on/off values of the embedded image information structure to the camera.

Register Operation

These functions deal with register operation on the camera.

- virtual void WriteRegister (unsigned int address, unsigned int value)
  Write to the specified register on the camera.
- virtual void WriteRegister (unsigned int address, unsigned int value, bool broadcast)
  Write to the specified register on the camera.
- virtual unsigned int ReadRegister (unsigned int address)
  Read the specified register from the camera.
- virtual void WriteRegisterBlock (unsigned short addressHigh, unsigned int addressLow, array< unsigned int >^buffer)
  Write to the specified register block on the camera.
- virtual void ReadRegisterBlock (unsigned short addressHigh, unsigned int addressLow, array< unsigned int >^buffer)
  Read from the specified register block on the camera.

Static Public Member Functions

- static System::String^ GetRegisterString (unsigned int registerVal)
  Returns a text representation of the register value.

Protected Member Functions

- ManagedCameraBase ()
- void OnNativeCallback (FlyCapture2::Image *pImage, void *pCallbackData)
- void OnNativeCameraEventCallback (void *pCallbackData)

Protected Attributes

- FlyCapture2::CameraBase * m_pNativeCamBase
- bool m_isLocal
- ImageEventCallback^ m_externalDelegate
- ImageCallbackDelegate^ m_internalDelegate
- ManagedCameraEventCallbackDelegate^ m_internalCameraEventDelegate
- IntPtr m_p

- Dictionary < ManagedEventOptions^ , NativeEventStruct >^ m_specificInternalCameraEvents
- Dictionary < ManagedEventOptions^ , NativeEventStruct >^ m_allInternalCameraEvents

Package Functions

- FlyCapture2::CameraBase * GetNativeCamera ()

8.28.1 Detailed Description

Abstract base class that represents a generic camera that defines a general interface to a camera.

8.28.2 Constructor & Destructor Documentation

8.28.2.1 virtual ~ManagedCameraBase ( ) [inline, virtual]

8.28.2.2 ManagedCameraBase ( ) [inline, protected]

8.28.3 Member Function Documentation

8.28.3.1 void Connect (ManagedPGRGuid^ *mgdPGRGuid*) [virtual]

Connects the ManagedCamera object to the camera specified by the GUID.

Parameters

mgdPGRGuid	The unique identifier for a specific camera on the PC.

See also

ManagedBusManager::GetCameraFromIndex()
ManagedBusManager::GetCameraFromSerialNumber()

Reimplemented in ManagedCamera, ManagedGigECamera, and ManagedGCCamera.

8.28.3.2 void DeregisterAllEvents ( ) [virtual]

8.28.3.3 void DeregisterEvent (ManagedEventOptions^ *hOpts*) [virtual]

8.28.3.4 void Disconnect (void) [virtual]

Disconnects the ManagedCamera object from the camera.

This allows another physical camera specified by a GUID to be connected to the -ManagedCamera object.

See also

Connect()

Reimplemented in ManagedGCCamera.

8.28.3.5 void EnableLUT (bool *on*) [virtual]

Enable or disable LUT functionality on the camera.

Parameters

on	Whether to enable or disable LUT.

See also

GetLUTInfo()
GetLUTChannel()
SetLUTChannel()

8.28.3.6 void FireSoftwareTrigger (bool *broadcast*) [virtual]

Fire the software trigger according to the DCAM specifications.

Parameters

broadcast	Whether the action should be broadcast.

8.28.3.7 unsigned int GetActiveLUTBank ( ) [virtual]

Get the LUT bank that is currently being used.

For cameras with PGR LUT, the active bank is always 0.

Returns

The currently active bank.

8.28.3.8 CameraInfo GetCameraInfo ( ) [virtual]

Retrieves information from the camera such as serial number, model name and other camera information.

Returns

Structure containing camera information.

8.28.3.9 FC2Config GetConfiguration ( ) [virtual]

Get the configuration associated with the camera object.

See also

SetConfiguration()

Returns

Current configuration.

8.28.3.10 TimeStamp GetCycleTime ( ) [virtual]

Returns a Timestamp struct containing 1394 CYCLE_TIME information.

Parameters

registerVal	The register value to query.

Returns

An Error indicating the success or failure of the function.

8.28.3.11 EmbeddedImageInfo GetEmbeddedImageInfo ( ) [virtual]

Get the current status of the embedded image information register, as well as the availability of each embedded property.

See also

SetEmbeddedImageInfo()

Returns

EmbeddedImageInfo structure containing embedded image information.

8.28.3.12 unsigned int GetGPIOPinDirection (unsigned int *pin*) [virtual]

Get the GPIO pin direction for the specified pin.

This is not a required call when using the trigger or strobe functions as the pin direction is set automatically internally.

Parameters

pin	Pin to get the direction for.

See also

SetGPIOPinDirection()

Returns

Direction of the pin. 0 for input, 1 for output.

8.28.3.13 void GetLUTBankInfo (unsigned int *bank*, bool% *readSupported*, bool% *writeSupported*) [virtual]

Query the read/write status of a single LUT bank.

Parameters

bank	The bank to query.
read-Supported	Whether reading from the bank is supported.
write-Supported	Whether writing to the bank is supported.

8.28.3.14 void GetLUTChannel (unsigned int *bank*, unsigned int *channel*, unsigned int *sizeEntries*, array< unsigned int >^ *entries*) [virtual]

Get the LUT channel settings from the camera.

Parameters

bank	Bank to retrieve.
channel	Channel to retrieve.
sizeEntries	Number of entries in LUT table to read.
entries	Array to store LUT entries in.

See also

GetLUTInfo()
EnableLUT()
SetLUTChannel()

8.28.3.15 LutData GetLUTInfo ( ) [virtual]

Query if LUT support is available on the camera.

See also

EnableLUT()
GetLUTChannel()
SetLUTChannel()

Returns

LutData structure containing the LUT information.

8.28.3.16 unsigned int GetMemoryChannel ( ) [virtual]

Retrieve the current memory channel from the camera.

See also

SaveToMemoryChannel()
RestoreFromMemoryChannel()
GetMemoryChannelInfo()

Returns

Currently selected memory channel.

8.28.3.17 unsigned int GetMemoryChannelInfo ( ) [virtual]

Query the camera for memory channel support.

If the number of channels is 0, then memory channel support is not available.

See also

GetMemoryChannel()
SaveToMemoryChannel()
RestoreFromMemoryChannel()

Returns

Number of memory channels supported.

8.28.3.18 FlyCapture2::CameraBase * GetNativeCamera ( ) [package]

8.28.3.19 CameraProperty GetProperty (PropertyType *type*) [virtual]

Reads the settings for the specified property from the camera.

If auto is on, the integer and abs values returned may not be consistent with each other.

Parameters

type	The PropertyType to retrieve information about.

See also

GetPropertyInfo()
SetProperty()

Returns

Property structure containing property information.

8.28.3.20 CameraPropertyInfo GetPropertyInfo (PropertyType *type*) [virtual]

Retrieves information about the specified camera property.

Parameters

type	The PropertyType to retrieve information about.

See also

GetProperty()
SetProperty()

Returns

PropertyInfo structure containing property information.

8.28.3.21 System::String GetRegisterString (unsigned int *registerVal*) [static]

Returns a text representation of the register value.

Parameters

registerVal	The register value to query.

Returns

The text representation of the register.

8.28.3.22 CameraStats GetStats ( ) [virtual]

8.28.3.23 StrobeControl GetStrobe (unsigned int *source*) [virtual]

Retrieve current strobe settings from the camera.

Parameters

source	Source pin for strobe information.

See also

GetStrobeInfo()
SetStrobe()

Returns

StrobeControl structure containing strobe information.

8.28.3.24 StrobeInfo GetStrobeInfo (unsigned int *source*) [virtual]

Retrieve strobe information from the camera.

Parameters

source	Source pin for strobe information.

See also

GetStrobe()
SetStrobe()

Returns

StrobeInfo structure containing strobe information.

8.28.3.25 CameraProperty GetTriggerDelay ( ) [virtual]

Retrieve current trigger delay settings from the camera.

See also

GetTriggerModeInfo()
GetTriggerMode()
SetTriggerMode()
GetTriggerDelayInfo()
SetTriggerDelay()

Returns

Structure to receive trigger delay settings.

8.28.3.26 CameraPropertyInfo GetTriggerDelayInfo ( ) [virtual]

Retrieve trigger delay information from the camera.

See also

GetTriggerModeInfo()
GetTriggerMode()
SetTriggerMode()
GetTriggerDelay()
SetTriggerDelay()

Returns

Structure to receive trigger delay information.

8.28.3.27 TriggerMode GetTriggerMode ( ) [virtual]

Retrieve current trigger settings from the camera.

See also

GetTriggerModeInfo()
SetTriggerMode()

Returns

TriggerMode structure containing trigger mode settings.

8.28.3.28 TriggerModeInfo GetTriggerModeInfo ( ) [virtual]

Retrieve trigger information from the camera.

See also

GetTriggerMode()
SetTriggerMode()

Returns

TriggerModeInfo structure containing receive trigger information.

8.28.3.29 bool IsConnected ( ) [virtual]

Checks if the ManagedCamera object is connected to a physical camera specified by a GUID.

See also

Connect()
Disconnect()

Returns

Whether Connect() was called on the ManagedCamera object.

8.28.3.30 void OnNativeCallback (FlyCapture2::Image * *pImage*, void * *pCallbackData*) [protected]

8.28.3.31 void OnNativeCameraEventCallback (void * *pCallbackData*) [protected]

8.28.3.32 unsigned int ReadRegister (unsigned int *address*) [virtual]

Read the specified register from the camera.

Parameters

address	DCAM address to be read from.

See also

WriteRegister()

Returns

The register value that is read.

8.28.3.33 void ReadRegisterBlock (unsigned short *addressHigh*, unsigned int *addressLow*, array< unsigned int >^ *buffer*) [virtual]

Read from the specified register block on the camera.

Parameters

addressHigh	Top 16 bits of the 48 bit absolute address to read from.
addressLow	Bottom 32 bits of the 48 bits absolute address to read from.
buffer	Array to store read data.

See also

WriteRegisterBlock()

8.28.3.34 void RegisterAllEvents (ManagedEventOptions^ *hOpts*) [virtual]

8.28.3.35 void RegisterEvent (ManagedEventOptions^ *hOpts*) [virtual]

8.28.3.36 void ResetStats ( ) [virtual]

8.28.3.37 void RestoreFromMemoryChannel (unsigned int *channel*) [virtual]

Restore the specfied current memory channel.

Parameters

channel	Memory channel to restore from.

See also

GetMemoryChannel()
SaveToMemoryChannel()
GetMemoryChannelInfo()

8.28.3.38 void RetrieveBuffer (ManagedImage^ *image*) [virtual]

Retrieves the the next image object containing the next image.

If the grab mode has not been set, or has been set to DROP_FRAMES the default behavior is to requeue images for DMA if they have not been retrieved by the time the next image transfer completes. If BUFFER_FRAMES is specified, the next image in the sequence will be retrieved. Note that for the BUFFER_FRAMES case, if retrieval does not keep up with the DMA process, images will be lost. The default behavior is to perform DROP_FRAMES image retrieval.

Parameters

image	ManagedImage object to store image data.

See also

StartCapture()
StopCapture()
WaitForBufferEvent()

8.28.3.39 void SaveToMemoryChannel (unsigned int *channel*) [virtual]

Save the current settings to the specfied current memory channel.

Parameters

channel	Memory channel to save to.

See also

GetMemoryChannel()
RestoreFromMemoryChannel()
GetMemoryChannelInfo()

8.28.3.40 void SetActiveLUTBank (unsigned int *activeBank*) [virtual]

Set the LUT bank that will be used.

Parameters

activeBank	The bank to be set as active.

8.28.3.41 void SetCallback (ImageEventCallback^ *hCallbackDelegate*) [virtual]

Sets the callback data to be used on completion of image transfer.

To clear the current stored callback data, pass in NULL as the argument.

Parameters

hCallback-Delegate	A function to be called when a new image is received.

See also

StartCapture()

Returns

An Error indicating the success or failure of the function.

8.28.3.42 void SetCamera (System::IntPtr *otherCamera*)

Set camera from a integer pointer camera.

8.28.3.43 void SetConfiguration (FC2Config^ *config*) [virtual]

Set the configuration associated with the camera object.

Parameters

config	Configuration structure to be used.

See also

GetConfiguration()

8.28.3.44 void SetEmbeddedImageInfo (EmbeddedImageInfo^ *info*) [virtual]

Sets the on/off values of the embedded image information structure to the camera.

Parameters

info	Structure to be used.

See also

GetEmbeddedImageInfo()

8.28.3.45 void SetGPIOPinDirection (unsigned int *pin,* unsigned int *direction*) [virtual]

Set the GPIO pin direction for the specified pin.

This is useful if there is a need to set the pin into an input pin (i.e. to read the voltage) off the pin without setting it as a trigger source. This is not a required call when using the trigger or strobe functions as the pin direction is set automatically internally.

Parameters

pin	Pin to get the direction for.
direction	Direction of the pin. 0 for input, 1 for output.

See also

GetGPIOPinDirection()

8.28.3.46 void SetGPIOPinDirection (unsigned int *pin,* unsigned int *direction,* bool *broadcast*) [virtual]

Set the GPIO pin direction for the specified pin.

This is useful if there is a need to set the pin into an input pin (i.e. to read the voltage) off the pin without setting it as a trigger source. This is not a required call when using the trigger or strobe functions as the pin direction is set automatically internally.

Parameters

pin	Pin to get the direction for.
direction	Direction of the pin. 0 for input, 1 for output.
broadcast	Whether the action should be broadcast.

See also

GetGPIOPinDirection()

8.28.3.47 void SetLUTChannel (unsigned int *bank,* unsigned int *channel,* unsigned int *sizeEntries,* array< unsigned int >^ *entries*) [virtual]

Set the LUT channel settings to the camera.

Parameters

bank	Bank to set.
channel	Channel to set.
sizeEntries	Number of entries in LUT table to write. This must be the same size as numEntries returned by GetLutInfo().
entries	Array containing LUT entries to write.

See also

GetLUTInfo()
EnableLUT()
GetLUTChannel()

8.28.3.48 void SetProperty (CameraProperty^ *camProperty*) [virtual]

Writes the settings for the specified property to the camera.

The property type must be specified in the Property structure passed into the function in order for the function to succeed. The absControl flag controls whether the absolute or integer value is written to the camera.

Parameters

camProperty	CameraProperty structure to be used.

See also

GetPropertyInfo()
GetProperty()

8.28.3.49 void SetProperty (CameraProperty^ *camProperty,* bool *broadcast*) [virtual]

Writes the settings for the specified property to the camera.

The property type must be specified in the Property structure passed into the function in order for the function to succeed. The absControl flag controls whether the absolute or integer value is written to the camera.

Parameters

camProperty	CameraProperty structure to be used.
broadcast	Whether the action should be broadcast.

See also

GetPropertyInfo()
GetProperty()

8.28.3.50 void SetStrobe (StrobeControl^ *strobeControl*) [virtual]

Set current strobe settings to the camera.

The strobe pin must be specified in the structure before being passed in to the function.

Parameters

strobe-Control	Structure providing strobe settings.

See also

GetStrobeInfo()
GetStrobe()

8.28.3.51 void SetTriggerDelay (CameraProperty^ *triggerDelay*) [virtual]

Set the specified trigger delay settings to the camera.

Parameters

triggerDelay	Structure providing trigger delay settings.

See also

GetTriggerModeInfo()
GetTriggerMode()
SetTriggerMode()
GetTriggerDelayInfo()
GetTriggerDelay()

Returns

An Error indicating the success or failure of the function.

8.28.3.52 void SetTriggerDelay (CameraProperty^ *triggerDelay*, bool *broadcast*) [virtual]

Set the specified trigger delay settings to the camera.

Parameters

triggerDelay	Structure providing trigger delay settings.
broadcast	Whether the action should be broadcast.

See also

GetTriggerModeInfo()
GetTriggerMode()
SetTriggerMode()
GetTriggerDelayInfo()
GetTriggerDelay()

Returns

An Error indicating the success or failure of the function.

8.28.3.53 void SetTriggerMode (TriggerMode^ *triggerMode*) [virtual]

Set the specified trigger settings to the camera.

Parameters

triggerMode	Structure providing trigger mode settings.

See also

GetTriggerModeInfo()
GetTriggerMode()

8.28.3.54 void SetUserBuffers (IntPtr *pMemBuffers*, int *size*, int *numBuffers*) [virtual]

Specify user allocated buffers to use as image data buffers.

To prevent image tearing, the size of each buffer should be equal to ((unsigned int)(bufferSize + packetSize - 1)/packetSize) * packetSize. The total size should be (size * numBuffers) or larger. The packet Size that should be used differs between interfaces: Firewire: Use the Format7 packet size. Usb2: First round to Format7 packet size then round to 512 bytes. Usb3: Use a packet size of 1024 bytes. GigE: No need to do any rounding on GigE

Parameters

pMemBuffers	Pointer to memory buffers to be written to.
size	The size of each buffer (in bytes).
numBuffers	Number of buffers in the array.

See also

StartCapture()
RetrieveBuffer()
StopCapture()

Returns

An Error indicating the success or failure of the function.

8.28.3.55 void StartCapture ( ) [virtual]

Starts isochronous image capture.

It will use either the current video mode or the most recently set video mode of the camera. RetrieveBuffer() can be called to get the image data.

See also

RetrieveBuffer()
StopCapture()

8.28.3.56 void StartCapture (ImageEventCallback^ *hCallbackDelegate*) [virtual]

Starts isochronous image capture.

It will use either the current video mode or the most recently set video mode of the camera. The callback function parameter is called on completion of image transfer.

Parameters

hCallback-Delegate	A function to be called when a new image is received.

See also

RetrieveBuffer()
StopCapture()

8.28.3.57 void StopCapture ( ) [virtual]

Stops isochronous image transfer and cleans up all associated resources.

If an image callback function (specified in the StartCapture() call) is currently executing, StopCapture() will not return until after the callback has completed.

See also

StartCapture()
RetrieveBuffer()

8.28.3.58 void WaitForBufferEvent (ManagedImage^ *image*, unsigned int *eventNumber*) [virtual]

Retrieves the next image event containing the next part of the image.

Parameters

image	ManagedImage object to store image data.
event-Number	The event number to wait for.

See also

StartCapture()
RetrieveBuffer()
StopCapture()

8.28.3.59 void WriteRegister (unsigned int *address*, unsigned int *value*) [virtual]

Write to the specified register on the camera.

Parameters

address	DCAM address to be written to.
value	The value to be written.

See also

ReadRegister()

8.28.3.60 void WriteRegister (unsigned int *address,* unsigned int *value,* bool *broadcast*) [virtual]

Write to the specified register on the camera.

Parameters

address	DCAM address to be written to.
value	The value to be written.
broadcast	Whether the action should be broadcast.

See also

ReadRegister()

8.28.3.61 void WriteRegisterBlock (unsigned short *addressHigh,* unsigned int *addressLow,* array< unsigned int >^ *buffer*) [virtual]

Write to the specified register block on the camera.

Parameters

addressHigh	Top 16 bits of the 48 bit absolute address to write to.
addressLow	Bottom 32 bits of the 48 bits absolute address to write to.
buffer	Array containing data to be written.

See also

ReadRegisterBlock()

8.28.4 Member Data Documentation

8.28.4.1 Dictionary<ManagedEventOptions^, NativeEventStruct> ^ m_allInternalCameraEvents [protected]

8.28.4.2 ImageEventCallback ^ m_externalDelegate [protected]

8.28.4.3 ManagedCameraEventCallbackDelegate ^ m_internalCameraEventDelegate [protected]

8.28.4.4 ImageCallbackDelegate ^ m_internalDelegate [protected]

8.28.4.5 bool m_isLocal [protected]

8.28.4.6 IntPtr m_p [protected]

8.28.4.7 FlyCapture2::CameraBase* m_pNativeCamBase [protected]

8.28.4.8 Dictionary<ManagedEventOptions^, NativeEventStruct>^ m_specificInternalCameraEvents [protected]

8.29 ManagedEventCallbackData Struct Reference

Public Attributes

- System::String^ EventName

  The event name used to register the event.
- UInt64 EventID

  The device register which EventName maps to.
- UInt64 EventTimestamp

  Timestamp indicated the time (as reported by the camera) at which the camera exposure operation completed.

8.29.1 Member Data Documentation

8.29.1.1 UInt64 EventID

The device register which EventName maps to.

Provides an alternate means of indexing into different event types.

8.29.1.2 System::String ^ EventName

The event name used to register the event.

Provided so the user knows which event triggered the callback.

8.29.1.3 UInt64 EventTimestamp

Timestamp indicated the time (as reported by the camera) at which the camera exposure operation completed.

This can be compared with image stimestamps if there is a need to map event timestamps to specific images, if applicable.

8.30 ManagedEventOptions Struct Reference

Options for enabling device event registration.

Public Attributes

- ManagedCameraEventCallback^ EventCallbackFcn

  Callback function pointer.
- System::String^ EventName

  Event name to register.

8.30.1 Detailed Description

Options for enabling device event registration.

8.30.2 Member Data Documentation

8.30.2.1 ManagedCameraEventCallback ^ EventCallbackFcn

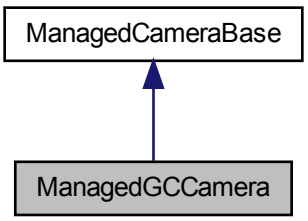
Callback function pointer.

8.30.2.2 System::String ^ EventName

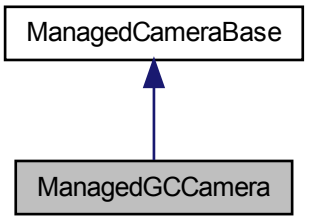
Event name to register.

8.31 ManagedGCCamera Class Reference

Inheritance diagram for ManagedGCCamera:



Collaboration diagram for ManagedGCCamera:



Public Member Functions

- ManagedGCCamera (void)
- virtual ~ManagedGCCamera (void)
- virtual void Connect (ManagedPGRGuid^ mgdPGRGuid) override

  Connects the ManagedCamera object to the camera specified by the GUID.
- virtual void Connect (ManagedPGRGuid^ mgdPGRGuid, String^ xmlPath) override
- virtual void Disconnect (void) override

  Disconnects the ManagedCamera object from the camera.
- void SetCamera (ManagedCameraBase^ cameraBase)
- void SetCamera (ManagedCameraBase^ cameraBase, String^ xmlPath)
- GenICam::GenApi::NodeMap^ GetNodeMap ()

Protected Member Functions

- !ManagedGCCamera ()

8.31.1 Constructor & Destructor Documentation

8.31.1.1 ManagedGCCamera (void)

8.31.1.2 ~ManagedGCCamera (void) [virtual]

8.31.1.3 !ManagedGCCamera ( ) [protected]

8.31.2 Member Function Documentation

8.31.2.1 void Connect (ManagedPGRGuid^ *mgdPGRGuid*) [override, virtual]

Connects the ManagedCamera object to the camera specified by the GUID.

Parameters

mgdPGR-Guid	The unique identifier for a specific camera on the PC.

See also

ManagedBusManager::GetCameraFromIndex()
ManagedBusManager::GetCameraFromSerialNumber()

Reimplemented from ManagedCameraBase.

8.31.2.2 void Connect (ManagedPGRGuid^ *mgdPGRGuid,* String^ *xmlPath*) [override, virtual]

8.31.2.3 void Disconnect (void) [override, virtual]

Disconnects the ManagedCamera object from the camera.

This allows another physical camera specified by a GUID to be connected to the -ManagedCamera object.

See also

Connect()

Reimplemented from ManagedCameraBase.

8.31.2.4 GenICam::GenApi::NodeMap GetNodeMap ( )

8.31.2.5 void SetCamera (ManagedCameraBase^ *cameraBase*)

8.31.2.6 void SetCamera (ManagedCameraBase^ *cameraBase,* String^ *xmlPath*)

8.32 ManagedGCPort Class Reference

Public Member Functions

- ManagedGCPort (GCCamera ∗camera)
- virtual ∼ManagedGCPort (void)
- virtual void Read (IntPtr buffer, __int64 address, __int64 length) override
- virtual void Write (IntPtr buffer, __int64 address, __int64 length) override

8.32.1 Constructor & Destructor Documentation

8.32.1.1 ManagedGCPort (GCCamera * *camera*)

8.32.1.2 ~ManagedGCPort (void) [virtual]

8.32.2 Member Function Documentation

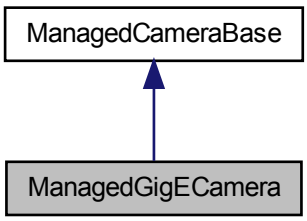
8.32.2.1 void Read (IntPtr *buffer*, __int64 *address*, __int64 *length*) [override, virtual]

8.32.2.2 void Write (IntPtr *buffer*, __int64 *address*, __int64 *length*) [override, virtual]

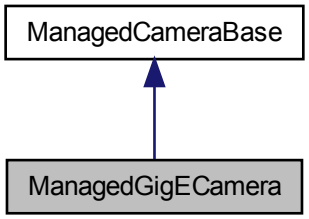
8.33 ManagedGigECamera Class Reference

The GigECamera object represents a physical Gigabit Ethernet camera.

Inheritance diagram for ManagedGigECamera:



Collaboration diagram for ManagedGigECamera:



Public Member Functions

- ManagedGigECamera ()
- ~ManagedGigECamera ()
- virtual void Connect (ManagedPGRGuid^ mgdPGRGuid) override

  Connects the ManagedCamera object to the camera specified by the GUID.

Protected Member Functions

- !ManagedGigECamera ()

GVCP Register Operation

These functions deal with GVCP register operation on the camera.

- void WriteGVCPRegister (unsigned int address, unsigned int value)

  Write a GVCP register.
- void WriteGVCPRegister (unsigned int address, unsigned int value, bool broadcast)

  Write a GVCP register.
- unsigned int ReadGVCPRegister (unsigned int address)

  Read a GVCP register.
- void WriteGVCPRegisterBlock (unsigned int address, array< unsigned int >^buffer)

  Write a GVCP register block.
- void ReadGVCPRegisterBlock (unsigned int address, array< unsigned int >^buffer)

  Read a GVCP register block.

- void WriteGVCPMemory (unsigned int address, array< unsigned char >^buffer)

  Write a GVCP memory block.
- void ReadGVCPMemory (unsigned int address, array< unsigned char >^buffer)

  Read a GVCP memory block.

GigE property manipulation

These functions deal with GigE properties.

- GigEProperty^ GetGigEProperty (GigEPropertyType propType)

  Get the specified GigEProperty.
- void SetGigEProperty (GigEProperty^ prop)

  Set the specified GigEProperty.
- unsigned int DiscoverGigEPacketSize ()

  Discover the largest packet size that works for the network link between the PC and the camera.

GigE image settings

These functions deal with GigE image setting.

- bool QueryGigEImagingMode (Mode mode)

  Check if the particular imaging mode is supported by the camera.
- Mode GetGigEImagingMode ()

  Get the current imaging mode on the camera.
- void SetGigEImagingMode (Mode mode)

  Set the current imaging mode to the camera.
- GigEImageSettingsInfo^ GetGigEImageSettingsInfo ()

  Get information about the image settings possible on the camera.
- GigEImageSettings^ GetGigEImageSettings ()

  Get the current image settings on the camera.
- void SetGigEImageSettings (GigEImageSettings^ settings)

  Set the image settings specified to the camera.

GigE image binning settings

These functions deal with GigE image binning setting.

- void GetGigEImageBinningSettings (unsigned int% horzBinningValue, unsigned int% vertBinningValue)

  Get the current binning settings on the camera.

- void SetGigEImageBinningSettings (unsigned int horzBinnningValue, unsigned int vertBinnningValue)

  Set the specified binning values to the camera.

GigE image stream configuration

These functions deal with GigE image stream configuration.

- unsigned int GetNumStreamChannels ()

  Get the number of stream channels present on the camera.
- GigEStreamChannel^ GetGigEStreamChannelInfo (unsigned int channel)

  Get the stream channel information for the specified channel.
- void SetGigEStreamChannelInfo (unsigned int channel, GigEStreamChannel^ channelInfo)

  Set the stream channel information for the specified channel.

GigE Configuration

These functions deal with configuring camera.

- GigEConfig^ GetGigEConfig ()

  Get the current configuration on the camera.
- void SetGigEConfig (GigEConfig^ config)

  Set the configuration specified to the camera.

8.33.1 Detailed Description

The GigECamera object represents a physical Gigabit Ethernet camera.

The object must first be connected to using Connect() before any other operations can proceed.

Please see ManagedCameraBase for basic functions that this class inherits from.

8.33.2 Constructor & Destructor Documentation

8.33.2.1 ManagedGigECamera ( )

8.33.2.2 ~ManagedGigECamera ( )

8.33.2.3 !ManagedGigECamera ( ) [protected]

8.33.3 Member Function Documentation

8.33.3.1 void Connect (ManagedPGRGuid^ *mgdPGRGuid*) [override, virtual]

Connects the ManagedCamera object to the camera specified by the GUID.

Parameters

mgdPGR-Guid	The unique identifier for a specific camera on the PC.

See also

ManagedBusManager::GetCameraFromIndex()
ManagedBusManager::GetCameraFromSerialNumber()

Reimplemented from ManagedCameraBase.

8.33.3.2 unsigned int DiscoverGigEPacketSize ( )

Discover the largest packet size that works for the network link between the PC and the camera.

This is useful in cases where there may be multiple links between the PC and the camera and there is a possiblity of a component not supporting the recommended jumbo frame packet size of 9000.

Returns

The maximum packet size supported by the link.

8.33.3.3 GigEConfig GetGigEConfig ( )

Get the current configuration on the camera.

Returns

Current configuration on camera.

8.33.3.4 void GetGigEImageBinningSettings (unsigned int% *horzBinningValue*, unsigned int% *vertBinningValue*)

Get the current binning settings on the camera.

Parameters

horzBinning-Value	Current horizontal binning value.
vertBinning-Value	Current vertical binning value.

8.33.3.5 GigEImageSettings GetGigEImageSettings ( )

Get the current image settings on the camera.

Returns

Current image settings on camera.

8.33.3.6 GigEImageSettingsInfo GetGigEImageSettingsInfo ( )

Get information about the image settings possible on the camera.

Returns

Image settings information.

8.33.3.7 Mode GetGigEImagingMode ( )

Get the current imaging mode on the camera.

Returns

Current imaging mode on the camera.

8.33.3.8 GigEProperty GetGigEProperty (GigEPropertyType *propType*)

Get the specified GigEProperty.

Returns

The GigE property to get.

8.33.3.9 GigEStreamChannel GetGigEStreamChannelInfo (unsigned int *channel*)

Get the stream channel information for the specified channel.

Parameters

channel	Channel number to use.

Returns

Stream channel information for the specified channel.

8.33.3.10 unsigned int GetNumStreamChannels ( )

Get the number of stream channels present on the camera.

Returns

Number of stream channels present.

8.33.3.11 bool QueryGigEImagingMode (Mode *mode*)

Check if the particular imaging mode is supported by the camera.

Parameters

mode	The mode to check.

Returns

Whether the mode is supported.

8.33.3.12 void ReadGVCPMemory (unsigned int *address,* array< unsigned char >^ *buffer*)

Read a GVCP memory block.

Parameters

address	GVCP address to be read from.
buffer	Array for data to be read into.

8.33.3.13 unsigned int ReadGVCPRegister (unsigned int *address*)

Read a GVCP register.

Parameters

address	GVCP address to be read from.

Returns

The value that is read.

8.33.3.14 void ReadGVCPRegisterBlock (unsigned int *address,* array< unsigned int >^ *buffer*)

Read a GVCP register block.

Parameters

address	GVCP address to be read from.
buffer	Array for data to be read into.

8.33.3.15 void SetGigEConfig (GigEConfig^ *config*)

Set the configuration specified to the camera.

Parameters

config	Configuration to set to camera.

8.33.3.16 void SetGigEImageBinningSettings (unsigned int *horzBinnningValue*, unsigned int *vertBinnningValue*)

Set the specified binning values to the camera.

It is recommended that GetGigEImageSettingsInfo() be called after this function succeeds to retrieve the new image settings information for the new binning mode.

Parameters

horzBinnningValue	Horizontal binning value.
vertBinnningValue	Vertical binning value.

8.33.3.17 void SetGigEImageSettings (GigEImageSettings^ *settings*)

Set the image settings specified to the camera.

Parameters

settings	Image settings to set to camera.

8.33.3.18 void SetGigEImagingMode (Mode *mode*)

Set the current imaging mode to the camera.

This should only be done when the camera is not streaming images.

Parameters

mode	Imaging mode to set to the camera.

8.33.3.19 void SetGigEProperty (GigEProperty^ *prop*)

Set the specified GigEProperty.

The GigEPropertyType field must be set in order for this function to succeed.

Parameters

prop	The GigE property to set.

8.33.3.20 void SetGigEStreamChannelInfo (unsigned int *channel*, GigEStreamChannel^ *channelInfo*)

Set the stream channel information for the specified channel.

Parameters

channel	Channel number to use.
channelInfo	Stream channel information to use for the specified channel.

8.33.3.21 void WriteGVCPMemory (unsigned int *address*, array< unsigned char >^ *buffer*)

Write a GVCP memory block.

Parameters

address	GVCP address to be write to.
buffer	Array containing data to be written.

8.33.3.22 void WriteGVCPRegister (unsigned int *address*, unsigned int *value*)

Write a GVCP register.

Parameters

address	GVCP address to be written to.
value	The value to be written.

8.33.3.23 void WriteGVCPRegister (unsigned int *address*, unsigned int *value*, bool *broadcast*)

Write a GVCP register.

Parameters

address	GVCP address to be written to.
value	The value to be written.
broadcast	Whether the action should be broadcast.

8.33.3.24 void WriteGVCPRegisterBlock (unsigned int *address,* array< unsigned int >^ *buffer*)

Write a GVCP register block.

Parameters

address	GVCP address to be write to.
buffer	Array containing data to be written.

8.34 ManagedImage Class Reference

The ManagedImageImage class is used to retrieve images from a camera, convert between multiple pixel formats and save images to disk.

Public Member Functions

- ManagedImage ()
- ManagedImage (unsigned int rows, unsigned int cols, unsigned int stride, unsigned char *pData, unsigned int dataSize, PixelFormat format)
- ManagedImage (unsigned int rows, unsigned int cols, unsigned int stride, unsigned char *pData, unsigned int dataSize, unsigned int receivedActualSize, PixelFormat format)
- ManagedImage (unsigned int rows, unsigned int cols, unsigned int stride, unsigned char *pData, unsigned int dataSize, PixelFormat format, BayerTileFormat bayerFormat)
- ManagedImage (unsigned int rows, unsigned int cols, unsigned int stride, unsigned char *pData, unsigned int dataSize, unsigned int receivedActualSize, PixelFormat format, BayerTileFormat bayerFormat)
- ManagedImage (unsigned char *pData, unsigned int dataSize)
- ManagedImage (unsigned int rows, unsigned int cols, PixelFormat format)
- ManagedImage (unsigned int rows, unsigned int cols, PixelFormat format, BayerTileFormat bayerFormat)
- ManagedImage (ManagedImage^ image)
- ~ManagedImage ()
- void SetDimensions (unsigned int rows, unsigned int cols, unsigned int stride, PixelFormat pixelFormat, BayerTileFormat bayerFormat)

  Sets the dimensions of the ManagedImage object.
- void GetDimensions (unsigned int *pRows, unsigned int *pCols, unsigned int *pStride, PixelFormat^ pPixelFormat, BayerTileFormat^ pBayerFormat)

Get the image dimensions associated with the ManagedImage object.

- void SetData (unsigned char *pData, unsigned int dataSize)

  Set the data of the ManagedImage object.
- void CalculateStatistics (ManagedImageStatistics^ statistics)

  Calculate statistics associated with the image.
- void Save (System::String^ fileName)

  Save the image to the specified file name.
- void Save (System::String^ fileName, ImageFileFormat format)

  Save the image to the specified file name with the file format specified.
- void Save (System::String^ fileName, PngOption^ option)

  Save the image to the specified file name with the options specified.
- void Save (System::String^ fileName, PpmOption^ option)

  Save the image to the specified file name with the options specified.
- void Save (System::String^ fileName, PgmOption^ option)

  Save the image to the specified file name with the options specified.
- void Save (System::String^ fileName, TiffOption^ option)

  Save the image to the specified file name with the options specified.
- void Save (System::String^ fileName, JpegOption^ option)

  Save the image to the specified file name with the options specified.
- void Save (System::String^ fileName, Jpg2Option^ option)

  Save the image to the specified file name with the options specified.
- void Save (System::String^ fileName, BMPOption^ option)

  Save the image to the specified file name with the options specified.
- void Convert (ManagedImage^ destImage)

  Converts the current image buffer and stores the result in the specified image.
- void Convert (PixelFormat format, ManagedImage^ destImage)

  Converts the current image buffer to the specified output format and stores the result in the specified image.
- void ReleaseBuffer ()

  Release the buffer associated with the ManagedImage.
- void * GetRawNativeImagePointer ()

Static Public Member Functions

- static unsigned int DetermineBitsPerPixel (PixelFormat format)

  Calculate the bits per pixel for the specified pixel format.

Protected Member Functions

- !ManagedImage ()

Package Functions

- ManagedImage (FlyCapture2::Image &image)
- bool IsNativeImageValid ()
- FlyCapture2::Image * GetNativeImage ()

Properties

- static ColorProcessingAlgorithm defaultColorProcessingAlgorithm [get, set]

  The default color processing algorithm to be used.
- static PixelFormat defaultOutputPixelFormat [get, set]

  The default output pixel format to be used.
- ColorProcessingAlgorithm colorProcessingAlgorithm [get, set]

  Color processing algorithm to be used.
- PixelFormat pixelFormat [get]

  Pixel format of the image.
- BayerTileFormat bayerTileFormat [get]

  Bayer tile format of the image.
- unsigned int blockId [get, set]

  Block id of the image.
- unsigned int cols [get]

  Number of columns in the image.
- unsigned int rows [get]

  Number of rows in the image.
- unsigned int stride [get]

  Number of bytes between rows in the image.
- unsigned int bitsPerPixel [get]

  Number of bits per pixel in the image.
- unsigned char * data [get]

  Raw pointer to image data.
- unsigned int dataSize [get]

  Size of the buffer associated with the image, in bytes.
- unsigned int receivedDataSize [get]

  Get the size of the compressed data, in bytes.
- ImageMetadata^ imageMetadata [get]

  Get the metadata associated with the image.
- TimeStamp^ timeStamp [get]

  Get the timestamp data associated with the image.
- System::Drawing::Bitmap^ bitmap [get]

  Get the internal bitmap representation associated with the image.

8.34.1 Detailed Description

The ManagedImageImage class is used to retrieve images from a camera, convert between multiple pixel formats and save images to disk.

Operations on Image objects are not guaranteed to be thread safe. It is recommended that operations on Image objects be protected by thread synchronization constructs such as mutexes.

8.34.2 Constructor & Destructor Documentation

8.34.2.1 ManagedImage ( )

8.34.2.2 ManagedImage (unsigned int *rows,* unsigned int *cols,* unsigned int *stride,* unsigned char * *pData,* unsigned int *dataSize,* PixelFormat *format*)

8.34.2.3 ManagedImage (unsigned int *rows,* unsigned int *cols,* unsigned int *stride,* unsigned char * *pData,* unsigned int *dataSize,* unsigned int *receivedActualSize,* PixelFormat *format*)

8.34.2.4 ManagedImage (unsigned int *rows,* unsigned int *cols,* unsigned int *stride,* unsigned char * *pData,* unsigned int *dataSize,* PixelFormat *format,* BayerTileFormat *bayerFormat*)

8.34.2.5 ManagedImage (unsigned int *rows,* unsigned int *cols,* unsigned int *stride,* unsigned char * *pData,* unsigned int *dataSize,* unsigned int *receivedActualSize,* PixelFormat *format,* BayerTileFormat *bayerFormat*)

8.34.2.6 ManagedImage (unsigned char * *pData,* unsigned int *dataSize*)

8.34.2.7 ManagedImage (unsigned int *rows,* unsigned int *cols,* PixelFormat *format*)

8.34.2.8 ManagedImage (unsigned int *rows,* unsigned int *cols,* PixelFormat *format,* BayerTileFormat *bayerFormat*)

8.34.2.9 ManagedImage (ManagedImage^ *image*)

8.34.2.10 ~ManagedImage ( )

8.34.2.11 ManagedImage (FlyCapture2::Image & *image*) [package]

8.34.2.12 !ManagedImage ( ) [protected]

8.34.3 Member Function Documentation

8.34.3.1 void CalculateStatistics (ManagedImageStatistics^ *statistics*)

Calculate statistics associated with the image.

In order to collect statistics for a particular channel, the enabled flag for the channel must be set to true. Statistics can only be collected for images in Mono8, Mono16, RGB, RGBU, BGR and BGRU.

Parameters

statistics	The ManagedImageStatistics object to hold the statistics.

8.34.3.2 void Convert (ManagedImage^ *destImage*)

Converts the current image buffer and stores the result in the specified image.

The destination image does not need to be configured in any way before the call is made.

Parameters

destImage	Destination image.

8.34.3.3 void Convert (PixelFormat *format,* ManagedImage^ *destImage*)

Converts the current image buffer to the specified output format and stores the result in the specified image.

The destination image does not need to be configured in any way before the call is made.

Parameters

format	Output format of the converted image.
destImage	Destination image.

8.34.3.4 unsigned int DetermineBitsPerPixel (PixelFormat *format*) [static]

Calculate the bits per pixel for the specified pixel format.

Parameters

format	The pixel format.

Returns

The bits per pixel.

8.34.3.5 void GetDimensions (unsigned int * *pRows*, unsigned int * *pCols*, unsigned int * *pStride*, PixelFormat^ *pPixelFormat*, BayerTileFormat^ *pBayerFormat*)

Get the image dimensions associated with the ManagedImage object.

Parameters

pRows	Number of rows.
pCols	Number of columns.
pStride	The stride.
pPixelFormat	Pixel format.
pBayerFormat	Bayer tile format.

8.34.3.6 FlyCapture2::Image * GetNativeImage ( ) [package]

8.34.3.7 void * GetRawNativeImagePointer ( )

8.34.3.8 bool IsNativeImageValid ( ) [package]

8.34.3.9 void ReleaseBuffer ( )

Release the buffer associated with the ManagedImage.

If no buffer is associated, the function does nothing.

8.34.3.10 void Save (System::String^ *fileName*)

Save the image to the specified file name.

Parameters

fileName	Filename to save image with.

8.34.3.11 void Save (System::String^ *fileName*, ImageFileFormat *format*)

Save the image to the specified file name with the file format specified.

Parameters

fileName	Filename to save image with.
format	File format to save in.

8.34.3.12 void Save (System::String^ *fileName,* PngOption^ *option*)

Save the image to the specified file name with the options specified.

Parameters

fileName	Filename to save image with.
option	Options to use while saving image.

8.34.3.13 void Save (System::String^ *fileName,* PpmOption^ *option*)

Save the image to the specified file name with the options specified.

Parameters

fileName	Filename to save image with.
option	Options to use while saving image.

8.34.3.14 void Save (System::String^ *fileName,* PgmOption^ *option*)

Save the image to the specified file name with the options specified.

Parameters

fileName	Filename to save image with.
option	Options to use while saving image.

8.34.3.15 void Save (System::String^ *fileName,* TiffOption^ *option*)

Save the image to the specified file name with the options specified.

Parameters

fileName	Filename to save image with.
option	Options to use while saving image.

8.34.3.16 void Save (System::String^ *fileName,* JpegOption^ *option*)

Save the image to the specified file name with the options specified.

Parameters

fileName	Filename to save image with.
option	Options to use while saving image.

8.34.3.17 void Save (System::String^ *fileName*, Jpg2Option^ *option*)

Save the image to the specified file name with the options specified.

Parameters

fileName	Filename to save image with.
option	Options to use while saving image.

8.34.3.18 void Save (System::String^ *fileName*, BMPOption^ *option*)

Save the image to the specified file name with the options specified.

Parameters

pFilename	Filename to save image with.
pOption	Options to use while saving image.

8.34.3.19 void SetData (unsigned char * *pData*, unsigned int *dataSize*)

Set the data of the ManagedImage object.

Ownership of the image buffer is not transferred to the ManagedImage object. It is the user's responsibility to delete the buffer when it is no longer in use.

Parameters

pData	Pointer to the image buffer.
dataSize	Size of the image buffer.

8.34.3.20 void SetDimensions (unsigned int *rows*, unsigned int *cols*, unsigned int *stride*, PixelFormat *pixelFormat*, BayerTileFormat *bayerFormat*)

Sets the dimensions of the ManagedImage object.

Parameters

rows	Number of rows to set.
cols	Number of cols to set.
stride	Stride to set.
pixelFormat	Pixel format to set.
bayerFormat	Bayer tile format to set.

8.34.4 Property Documentation

8.34.4.1 BayerTileFormat bayerTileFormat [get]

Bayer tile format of the image.

8.34.4.2 System:: Drawing:: Bitmap^ bitmap [get]

Get the internal bitmap representation associated with the image.

Returns

A System::Drawing::Bitmap containing the image data.

8.34.4.3 unsigned int bitsPerPixel [get]

Number of bits per pixel in the image.

8.34.4.4 unsigned int blockId [get, set]

Block id of the image.

8.34.4.5 ColorProcessingAlgorithm colorProcessingAlgorithm [get, set]

Color processing algorithm to be used.

8.34.4.6 unsigned int cols [get]

Number of columns in the image.

8.34.4.7 unsigned char* data [get]

Raw pointer to image data.

8.34.4.8 unsigned int dataSize [get]

Size of the buffer associated with the image, in bytes.

8.34.4.9 ColorProcessingAlgorithm defaultColorProcessingAlgorithm [static, get, set]

The default color processing algorithm to be used.

8.34.4.10 PixelFormat defaultOutputPixelFormat [static, get, set]

The default output pixel format to be used.

8.34.4.11 ImageMetadata^ imageMetadata [get]

Get the metadata associated with the image.

This includes embedded image information.

Returns

Metadata associated with the image.

8.34.4.12 PixelFormat pixelFormat [get]

Pixel format of the image.

8.34.4.13 unsigned int receivedDataSize [get]

Get the size of the compressed data, in bytes.

A compressed image will have a maximum size equal to GetDataSize(), but may actually contain less data, depending on the compression level. For uncompressed images, a value smaller than the data size may indicate lost data.

8.34.4.14 unsigned int rows [get]

Number of rows in the image.

8.34.4.15 unsigned int stride [get]

Number of bytes between rows in the image.

8.34.4.16 TimeStamp^ timeStamp [get]

Get the timestamp data associated with the image.

Returns

Timestamp data associated with the image.

8.35 ManagedImageStatistics Class Reference

Public Member Functions

- ManagedImageStatistics ()
- ~ManagedImageStatistics ()
- void EnableAll ()
- void DisableAll ()
- void EnableGreyOnly ()
- void EnableRGBOnly ()
- void EnableHSLOnly ()
- bool GetChannelStatus (StatisticsChannel channel)
- void SetChannelStatus (StatisticsChannel channel, bool enabled)
- void GetRange (StatisticsChannel channel, unsigned int% min, unsigned int% max)
- void GetPixelValueRange (StatisticsChannel channel, unsigned int% pixelValueMin, unsigned int% pixelValueMax)
- void GetNumPixelValues (StatisticsChannel channel, unsigned int% numPixelValues)
- void GetMean (StatisticsChannel channel, float% mean)
- void GetHistogram (StatisticsChannel channel, array< int >^histogram)
- void GetStatistics (StatisticsChannel channel, unsigned int% rangeMin, unsigned int% rangeMax, unsigned int% pixelValueMin, unsigned int% pixelValueMax, unsigned int% numPixelValues, float% mean, array< int >^histogram)

Package Functions

- FlyCapture2::ImageStatistics * GetNativeImageStatistics ()

8.35.1 Constructor & Destructor Documentation

8.35.1.1 ManagedImageStatistics ( )

8.35.1.2 ~ManagedImageStatistics ( )

8.35.2 Member Function Documentation

8.35.2.1 void DisableAll ( )

8.35.2.2 void EnableAll ( )

8.35.2.3 void EnableGreyOnly ( )

8.35.2.4 void EnableHSLOnly ( )

8.35.2.5 void EnableRGBOnly ( )

8.35.2.6 bool GetChannelStatus (StatisticsChannel *channel*)

8.35.2.7 void GetHistogram (StatisticsChannel *channel,* array< int >^ *histogram*)

8.35.2.8 void GetMean (StatisticsChannel *channel,* float% *mean*)

8.35.2.9 FlyCapture2::ImageStatistics * GetNativeImageStatistics ( ) [package]

8.35.2.10 void GetNumPixelValues (StatisticsChannel *channel,* unsigned int% *numPixelValues*)

8.35.2.11 void GetPixelValueRange (StatisticsChannel *channel,* unsigned int% *pixelValueMin,* unsigned int% *pixelValueMax*)

8.35.2.12 void GetRange (StatisticsChannel *channel,* unsigned int% *min,* unsigned int% *max*)

8.35.2.13 void GetStatistics (StatisticsChannel *channel,* unsigned int% *rangeMin,* unsigned int% *rangeMax,* unsigned int% *pixelValueMin,* unsigned int% *pixelValueMax,* unsigned int% *numPixelValues,* float% *mean,* array< int >^ *histogram*)

8.35.2.14 void SetChannelStatus (StatisticsChannel *channel,* bool *enabled*)

8.36 ManagedPGRGuid Class Reference

Managed version of a PGRGuid.

Public Member Functions

- ManagedPGRGuid ()

  Constructor.
- ManagedPGRGuid (ManagedPGRGuid^ managedGuid)

  Copy constructor.
- ManagedPGRGuid (ManagedPGRGuid% managedGuid)

  Copy constructor.
- ManagedPGRGuid% operator= (ManagedPGRGuid% managedGuid)

  Assignment operator.
- virtual bool Equals (Object^ obj) override
- virtual int GetHashCode () override

Static Public Member Functions

- static bool operator== (ManagedPGRGuid% left, ManagedPGRGuid% right)

Equality operator.

- static bool operator!= (ManagedPGRGuid% left, ManagedPGRGuid% right)

  Inequality operator.

Public Attributes

- unsigned int value0
- unsigned int value1
- unsigned int value2
- unsigned int value3

8.36.1 Detailed Description

Managed version of a PGRGuid.

It is used to uniquely identify a camera.

8.36.2 Constructor & Destructor Documentation

8.36.2.1 ManagedPGRGuid ( ) [inline]

Constructor.

8.36.2.2 ManagedPGRGuid (ManagedPGRGuid^ *managedGuid*) [inline]

Copy constructor.

8.36.2.3 ManagedPGRGuid (ManagedPGRGuid% *managedGuid*) [inline]

Copy constructor.

8.36.3 Member Function Documentation

8.36.3.1 virtual bool Equals (Object^ *obj*) [inline, override, virtual]

8.36.3.2 virtual int GetHashCode ( ) [inline, override, virtual]

8.36.3.3 static bool operator!= (ManagedPGRGuid% *left*, ManagedPGRGuid% *right*) [inline, static]

Inequality operator.

8.36.3.4 ManagedPGRGuid % operator= (ManagedPGRGuid% *managedGuid*)

`[inline]`

Assignment operator.

8.36.3.5 static bool operator== (ManagedPGRGuid% *left,* ManagedPGRGuid% *right*)

`[inline, static]`

Equality operator.

8.36.4 Member Data Documentation

8.36.4.1 unsigned int value0

8.36.4.2 unsigned int value1

8.36.4.3 unsigned int value2

8.36.4.4 unsigned int value3

8.37 ManagedTopologyNode Class Reference

The ManagedTopologyNode class contains topology information that can be used to generate a tree structure of all cameras and devices connected to a computer.

Public Types

- enum PortType { NotConnected = 1, ConnectedToParent, ConnectedToChild }

  Possible states of a port on a node.
- enum NodeType { Computer, Bus, Camera, Node }

  Type of node.

Public Member Functions

- virtual ~ManagedTopologyNode ()
- ManagedTopologyNode (ManagedTopologyNode^ other)
- ManagedTopologyNode (ManagedTopologyNode% other)
- ManagedPGRGuid^ GetGuid ()

  Get the PGRGuid associated with the node.
- int GetDeviceId ()

  Get the device ID associated with the node.
- NodeType GetNodeType ()

Get the node type associated with the node.

- InterfaceType GetInterfaceType ()

  Get the interface type associated with the node.

- unsigned int GetNumChildren ()

  Get the number of child nodes.

- ManagedTopologyNode^ GetChild (unsigned int position)

  Get child node located at the specified position.

- unsigned int GetNumPorts ()

  Get the number of ports.

- PortType GetPortType (unsigned int position)

  Get type of port located at the specified position.

Package Functions

- ManagedTopologyNode (FlyCapture2::TopologyNode *pNode)

Static Package Functions

- static ManagedTopologyNode::PortType TranslatePortType (FlyCapture2::TopologyNode::PortType portType)
- static FlyCapture2::TopologyNode::PortType TranslatePortType (ManagedTopologyNode::PortType portType)
- static ManagedTopologyNode::NodeType TranslateNodeType (FlyCapture2::TopologyNode::NodeType portType)
- static FlyCapture2::TopologyNode::NodeType TranslateNodeType (ManagedTopologyNode::NodeType portType)

8.37.1 Detailed Description

The ManagedTopologyNode class contains topology information that can be used to generate a tree structure of all cameras and devices connected to a computer.

8.37.2 Member Enumeration Documentation

8.37.2.1 enum NodeType

Type of node.

Enumerator:

Computer

Bus

Camera

Node

8.37.2.2 enum PortType

Possible states of a port on a node.

Enumerator:

NotConnected

ConnectedToParent

ConnectedToChild

8.37.3 Constructor & Destructor Documentation

8.37.3.1 virtual ~ManagedTopologyNode ( ) [inline, virtual]

8.37.3.2 ManagedTopologyNode (ManagedTopologyNode^ *other*)

8.37.3.3 ManagedTopologyNode (ManagedTopologyNode% *other*)

8.37.3.4 ManagedTopologyNode (FlyCapture2::TopologyNode * *pNode*) [package]

8.37.4 Member Function Documentation

8.37.4.1 ManagedTopologyNode GetChild (unsigned int *position*)

Get child node located at the specified position.

Parameters

position	Position of the node.

Returns

ManagedTopologyNode at the specified position.

8.37.4.2 int GetDeviceId ( )

Get the device ID associated with the node.

Returns

Device ID of the node.

8.37.4.3 ManagedPGRGuid GetGuid ( )

Get the PGRGuid associated with the node.

Returns

PGRGuid of the node.

8.37.4.4 InterfaceType GetInterfaceType ( )

Get the interface type associated with the node.

Returns

Interface type of the node.

8.37.4.5 ManagedTopologyNode::NodeType GetNodeType ( )

Get the node type associated with the node.

Returns

Node type of the node.

8.37.4.6 unsigned int GetNumChildren ( )

Get the number of child nodes.

Returns

Number of child nodes.

8.37.4.7 unsigned int GetNumPorts ( )

Get the number of ports.

Returns

Number of ports.

8.37.4.8 ManagedTopologyNode::PortType GetPortType (unsigned int *position*)

Get type of port located at the specified position.

Parameters

position	Position of the port.

Returns

PortType at the specified position.

8.37.4.9 ManagedTopologyNode::NodeType TranslateNodeType (FlyCapture2::TopologyNode::NodeType *portType*) [static, package]

8.37.4.10 FlyCapture2::TopologyNode::NodeType TranslateNodeType (ManagedTopologyNode::NodeType *portType*) [static, package]

8.37.4.11 ManagedTopologyNode::PortType TranslatePortType (FlyCapture2::TopologyNode::PortType *portType*) [static, package]

8.37.4.12 FlyCapture2::TopologyNode::PortType TranslatePortType (ManagedTopologyNode::PortType *portType*) [static, package]

8.38 ManagedUtilities Class Reference

Static Public Member Functions

- static void CheckDriver (ManagedPGRGuid^ guid)
- static System::String^ GetDriverDeviceName (ManagedPGRGuid^ mgdPGRGuid)
- static void LaunchBrowser (System::String^ address)
- static void LaunchHelp (System::String^ fileName)
- static void LaunchCommand (System::String^ command)
- static void LaunchCommandAsync (System::String^ command, AsyncCommandCallback^ hCallbackDelegate)

Static Package Functions

- static void OnNativeCallback (FlyCapture2::Error retError, void *pUserData)

Static Package Attributes

- static AsyncCommandCallback^ m_externalDelegate
- static CommandCallbackDelegate^ m_internalDelegate

Properties

- static SystemInfo^ systemInfo [get]
- static FC2Version^ libraryVersion [get]

8.38.1 Member Function Documentation

8.38.1.1 void CheckDriver (ManagedPGRGuid^ *guid*) [static]

8.38.1.2 System::String GetDriverDeviceName (ManagedPGRGuid^ *mgdPGRGuid*) [static]

8.38.1.3 void LaunchBrowser (System::String^ *address*) [static]

8.38.1.4 void LaunchCommand (System::String^ *command*) [static]

8.38.1.5 void LaunchCommandAsync (System::String^ *command,* AsyncCommandCallback^ *hCallbackDelegate*) [static]

8.38.1.6 void LaunchHelp (System::String^ *fileName*) [static]

8.38.1.7 void OnNativeCallback (FlyCapture2::Error *retError,* void * *pUserData*) [static, package]

8.38.2 Member Data Documentation

8.38.2.1 AsyncCommandCallback ^ m_externalDelegate [static, package]

8.38.2.2 CommandCallbackDelegate ^ m_internalDelegate [static, package]

8.38.3 Property Documentation

8.38.3.1 FC2Version^ libraryVersion [static, get]

8.38.3.2 SystemInfo^ systemInfo [static, get]

8.39 NativeEventStruct Struct Reference

Public Attributes

- FlyCapture2::EventOptions * ptr

8.39.1 Member Data Documentation

8.39.1.1 FlyCapture2::EventOptions* ptr

8.40 PgmOption Struct Reference

Options for saving PGM images.

Public Member Functions

- PgmOption ()

Properties

- bool binaryFile

  Whether to save the PPM as a binary file.

8.40.1 Detailed Description

Options for saving PGM images.

8.40.2 Constructor & Destructor Documentation

8.40.2.1 PgmOption ( ) [inline]

8.40.3 Property Documentation

8.40.3.1 bool binaryFile

Whether to save the PPM as a binary file.

8.41 PngOption Struct Reference

Options for saving PNG images.

Public Member Functions

- PngOption ()

Properties

- bool interlaced

  Whether to save the PNG as interlaced.
- unsigned int compressionLevel

  Compression level (0-9).

8.41.1 Detailed Description

Options for saving PNG images.

8.41.2 Constructor & Destructor Documentation

8.41.2.1 PngOption ( ) [inline]

8.41.3 Property Documentation

8.41.3.1 unsigned int compressionLevel

Compression level (0-9).

0 is no compression, 9 is best compression.

8.41.3.2 bool interlaced

Whether to save the PNG as interlaced.

8.42 PpmOption Struct Reference

Options for saving PPM images.

Public Member Functions

- PpmOption ()

Properties

- bool binaryFile

  Whether to save the PPM as a binary file.

8.42.1 Detailed Description

Options for saving PPM images.

8.42.2 Constructor & Destructor Documentation

8.42.2.1 PpmOption ( ) [inline]

8.42.3 Property Documentation

8.42.3.1 bool binaryFile

Whether to save the PPM as a binary file.

8.43 StrobeControl Struct Reference

A camera strobe.

Properties

- unsigned int source

  Source value.
- bool onOff

  Flag controlling on/off.
- unsigned int polarity

  Signal polarity.
- float delay

  Signal delay (in ms).
- float duration

  Signal duration (in ms).

8.43.1 Detailed Description

A camera strobe.

8.43.2 Property Documentation

8.43.2.1 float delay

Signal delay (in ms).

8.43.2.2 float duration

Signal duration (in ms).

8.43.2.3 bool onOff

Flag controlling on/off.

8.43.2.4 unsigned int polarity

Signal polarity.

8.43.2.5 unsigned int source

Source value.

8.44 StrobeInfo Struct Reference

A camera strobe property.

Properties

- unsigned int source

  Source value.
- bool present

  Presence of strobe.
- bool readOutSupported

  Flag indicating if strobe value can be read out.
- bool onOffSupported

  Flag indicating if on/off is supported.
- bool polaritySupported

  Flag indicating if polarity is supported.
- float minValue

  Minimum value.
- float maxValue

  Maximum value.

8.44.1 Detailed Description

A camera strobe property.

8.44.2 Property Documentation

8.44.2.1 float maxValue

Maximum value.

8.44.2.2 float minValue

Minimum value.

8.44.2.3 bool onOffSupported

Flag indicating if on/off is supported.

8.44.2.4 bool polaritySupported

Flag indicating if polarity is supported.

8.44.2.5 bool present

Presence of strobe.

8.44.2.6 bool readOutSupported

Flag indicating if strobe value can be read out.

8.44.2.7 unsigned int source

Source value.

8.45 SystemInfo Struct Reference

Description of the system.

Properties

- OSType osType

  Operating system type as described by OSType.
- System::String^ osDescription

  Detailed description of the operating system.
- ByteOrder byteOrder

  Byte order of the system.
- unsigned int systemMemorySize

  Amount of memory available on the system.
- System::String^ cpuDescription

  Detailed description of the CPU.
- unsigned int numCpuCores

  Number of cores on all CPUs on the system.
- System::String^ driverList

  List of drivers used.
- System::String^ libraryList

  List of libraries used.
- System::String^ gpuDescription

  Detailed description of the GPU.
- unsigned int screenWidth

  Screen resolution width in pixels.
- unsigned int screenHeight

  Screen resolution height in pixels.

8.45.1 Detailed Description

Description of the system.

8.45.2 Property Documentation

8.45.2.1 ByteOrder byteOrder

Byte order of the system.

8.45.2.2 System:: String^ cpuDescription

Detailed description of the CPU.

8.45.2.3 System:: String^ driverList

List of drivers used.

8.45.2.4 System:: String^ gpuDescription

Detailed description of the GPU.

8.45.2.5 System:: String^ libraryList

List of libraries used.

8.45.2.6 unsigned int numCpuCores

Number of cores on all CPUs on the system.

8.45.2.7 System:: String^ osDescription

Detailed description of the operating system.

8.45.2.8 OSType osType

Operating system type as described by OSType.

8.45.2.9 unsigned int screenHeight

Screen resolution height in pixels.

8.45.2.10 unsigned int screenWidth

Screen resolution width in pixels.

8.45.2.11 unsigned int systemMemorySize

Amount of memory available on the system.

8.46 TiffOption Struct Reference

Options for saving TIFF images.

Public Types

- enum CompressionMethod { None = 1, PackBits, Deflate, AdobeDeflate, CcittFax3, CcittFax4, Lzw, Jpeg }

Public Member Functions

- TiffOption ()

Properties

- CompressionMethod compression

  Compression method to use for encoding TIFF images.

8.46.1 Detailed Description

Options for saving TIFF images.

8.46.2 Member Enumeration Documentation

8.46.2.1 enum CompressionMethod

Enumerator:

None Save without any compression.

PackBits Save using PACKBITS compression.

Deflate Save using DEFLATE compression (ZLIB compression).

AdobeDeflate Save using ADOBE DEFLATE compression.

CcittFax3 Save using CCITT Group 3 fax encoding. This is only valid for 1-bit images only. Default to LZW for other bit depths.

CcittFax4 Save using CCITT Group 4 fax encoding. This is only valid for 1-bit images only. Default to LZW for other bit depths.

Lzw Save using LZW compression.

Jpeg Save using JPEG compression. This is only valid for 8-bit greyscale and 24-bit only. Default to LZW for other bit depths.

8.46.3 Constructor & Destructor Documentation

8.46.3.1 TiffOption ( ) [inline]

8.46.4 Property Documentation

8.46.4.1 CompressionMethod compression

Compression method to use for encoding TIFF images.

8.47 TimeStamp Struct Reference

Timestamp information.

Properties

- long long seconds

  Seconds.
- unsigned int microSeconds

  Microseconds.
- unsigned int cycleSeconds

  1394 cycle time seconds.
- unsigned int cycleCount

  1394 cycle time count.
- unsigned int cycleOffset

  1394 cycle time offset.

8.47.1 Detailed Description

Timestamp information.

8.47.2 Property Documentation

8.47.2.1 unsigned int cycleCount

1394 cycle time count.

8.47.2.2 unsigned int cycleOffset

1394 cycle time offset.

8.47.2.3 unsigned int cycleSeconds

1394 cycle time seconds.

8.47.2.4 unsigned int microSeconds

Microseconds.

8.47.2.5 long long seconds

Seconds.

8.48 Translate Class Reference

Static Package Functions

- static ErrorType translate (FlyCapture2::ErrorType errorType)
- static FlyCapture2::ErrorType translate (ErrorType errorType)
- static GrabMode translate (FlyCapture2::GrabMode grabMode)
- static FlyCapture2::GrabMode translate (GrabMode grabMode)
- static BandwidthAllocation translate (FlyCapture2::BandwidthAllocation bandwidthAllocation)
- static FlyCapture2::BandwidthAllocation translate (BandwidthAllocation bandwidthAllocation)
- static InterfaceType translate (FlyCapture2::InterfaceType interfaceType)
- static FlyCapture2::InterfaceType translate (InterfaceType interfaceType)
- static DriverType translate (FlyCapture2::DriverType driverType)
- static FlyCapture2::DriverType translate (DriverType driverType)
- static PropertyType translate (FlyCapture2::PropertyType propertyType)
- static FlyCapture2::PropertyType translate (PropertyType propertyType)
- static FrameRate translate (FlyCapture2::FrameRate frmRate)
- static FlyCapture2::FrameRate translate (FrameRate frmRate)
- static VideoMode translate (FlyCapture2::VideoMode videoMode)
- static FlyCapture2::VideoMode translate (VideoMode videoMode)
- static PixelFormat translate (FlyCapture2::PixelFormat pixelFormat)
- static FlyCapture2::PixelFormat translate (PixelFormat pixelFormat)
- static BayerTileFormat translate (FlyCapture2::BayerTileFormat bayerFormat)
- static FlyCapture2::BayerTileFormat translate (BayerTileFormat bayerFormat)
- static Mode translate (FlyCapture2::Mode mode)
- static FlyCapture2::Mode translate (Mode mode)

- static BusSpeed translate (FlyCapture2::BusSpeed busSpeed)
- static FlyCapture2::BusSpeed translate (BusSpeed busSpeed)
- static PCIeBusSpeed translate (FlyCapture2::PCIeBusSpeed pcieBusSpeed)
- static FlyCapture2::PCIeBusSpeed translate (PCIeBusSpeed pcieBusSpeed)
- static ColorProcessingAlgorithm translate (FlyCapture2::ColorProcessing-Algorithm algorithm)
- static FlyCapture2::ColorProcessingAlgorithm translate (ColorProcessing-Algorithm algorithm)
- static ImageFileFormat translate (FlyCapture2::ImageFileFormat fileFmt)
- static FlyCapture2::ImageFileFormat translate (ImageFileFormat fileFmt)
- static TiffOption::CompressionMethod translate (FlyCapture2::TIFFOption::-CompressionMethod method)
- static FlyCapture2::TIFFOption::CompressionMethod translate (TiffOption::-CompressionMethod method)
- static StatisticsChannel translate (FlyCapture2::ImageStatistics::Statistics-Channel channel)
- static FlyCapture2::ImageStatistics::StatisticsChannel translate (Statistics-Channel channel)
- static OSType translate (FlyCapture2::OSType osType)
- static FlyCapture2::OSType translate (OSType osType)
- static ByteOrder translate (FlyCapture2::ByteOrder byteOrder)
- static FlyCapture2::ByteOrder translate (ByteOrder byteOrder)
- static GigEPropertyType translate (FlyCapture2::GigEPropertyType propType)
- static FlyCapture2::GigEPropertyType translate (GigEPropertyType propType)
- static void ToMgd (FlyCapture2::FC2Config *pNative, FC2Config^ mgd)
- static void ToNative (FC2Config^ mgd, FlyCapture2::FC2Config *pNative)
- static void ToMgd (FlyCapture2::PropertyInfo *pNative, CameraPropertyInfo^ mgd)
- static void ToNative (CameraPropertyInfo^ mgd, FlyCapture2::PropertyInfo *p-Native)
- static void ToMgd (FlyCapture2::Property *pNative, CameraProperty^ mgd)
- static void ToNative (CameraProperty^ mgd, FlyCapture2::Property *pNative)
- static void ToMgd (FlyCapture2::TriggerModeInfo *pNative, TriggerModeInfo^ mgd)
- static void ToNative (TriggerModeInfo^ mgd, FlyCapture2::TriggerModeInfo *p-Native)
- static void ToMgd (FlyCapture2::TriggerMode *pNative, TriggerMode^ mgd)
- static void ToNative (TriggerMode^ mgd, FlyCapture2::TriggerMode *pNative)
- static void ToMgd (FlyCapture2::StrobeInfo *pNative, StrobeInfo^ mgd)
- static void ToNative (StrobeInfo^ mgd, FlyCapture2::StrobeInfo *pNative)
- static void ToMgd (FlyCapture2::StrobeControl *pNative, StrobeControl^ mgd)
- static void ToNative (StrobeControl^ mgd, FlyCapture2::StrobeControl *pNative)
- static void ToMgd (FlyCapture2::Format7ImageSettings *pNative, Format7-ImageSettings^ mgd)
- static void ToNative (Format7ImageSettings^ mgd, FlyCapture2::Format7Image-Settings *pNative)
- static void ToMgd (FlyCapture2::Format7Info *pNative, Format7Info^ mgd)

- static void ToNative (Format7Info^ mgd, FlyCapture2::Format7Info *pNative)
- static void ToMgd (FlyCapture2::Format7PacketInfo *pNative, Format7PacketInfo^ mgd)
- static void ToNative (Format7PacketInfo^ mgd, FlyCapture2::Format7PacketInfo *pNative)
- static void ToMgd (FlyCapture2::TimeStamp *pNative, TimeStamp^ mgd)
- static void ToNative (TimeStamp^ mgd, FlyCapture2::TimeStamp *pNative)
- static void ToMgd (FlyCapture2::ConfigROM *pNative, ConfigROM^ mgd)
- static void ToNative (ConfigROM^ mgd, FlyCapture2::ConfigROM *pNative)
- static void ToMgd (FlyCapture2::CameraInfo *pNative, CameraInfo^ mgd)
- static void ToNative (CameraInfo^ mgd, FlyCapture2::CameraInfo *pNative)
- static void ToMgd (FlyCapture2::ImageMetadata *pNative, ImageMetadata^ mgd)
- static void ToNative (ImageMetadata^ mgd, FlyCapture2::ImageMetadata *pNative)
- static void ToMgd (FlyCapture2::LUTData *pNative, LutData^ mgd)
- static void ToNative (LutData^ mgd, FlyCapture2::LUTData *pNative)
- static void ToMgd (FlyCapture2::EmbeddedImageInfoProperty *pNative, EmbeddedImageInfoProperty^ mgd)
- static void ToNative (EmbeddedImageInfoProperty^ mgd, FlyCapture2::EmbeddedImageInfoProperty *pNative)
- static void ToMgd (FlyCapture2::EmbeddedImageInfo *pNative, EmbeddedImageInfo^ mgd)
- static void ToNative (EmbeddedImageInfo^ mgd, FlyCapture2::EmbeddedImageInfo *pNative)
- static void ToMgd (FlyCapture2::PNGOption *pNative, PngOption^ mgd)
- static void ToNative (PngOption^ mgd, FlyCapture2::PNGOption *pNative)
- static void ToMgd (FlyCapture2::PPMOption *pNative, PpmOption^ mgd)
- static void ToNative (PpmOption^ mgd, FlyCapture2::PPMOption *pNative)
- static void ToMgd (FlyCapture2::PGMOption *pNative, PgmOption^ mgd)
- static void ToNative (PgmOption^ mgd, FlyCapture2::PGMOption *pNative)
- static void ToMgd (FlyCapture2::TIFFOption *pNative, TiffOption^ mgd)
- static void ToNative (TiffOption^ mgd, FlyCapture2::TIFFOption *pNative)
- static void ToMgd (FlyCapture2::JPEGOption *pNative, JpegOption^ mgd)
- static void ToNative (JpegOption^ mgd, FlyCapture2::JPEGOption *pNative)
- static void ToMgd (FlyCapture2::JPG2Option *pNative, Jpg2Option^ mgd)
- static void ToNative (Jpg2Option^ mgd, FlyCapture2::JPG2Option *pNative)
- static void ToMgd (FlyCapture2::BMPOption *pNative, BMPOption^ mgd)
- static void ToNative (BMPOption^ mgd, FlyCapture2::BMPOption *pNative)
- static void ToMgd (FlyCapture2::SystemInfo *pNative, SystemInfo^ mgd)
- static void ToMgd (FlyCapture2::FC2Version *pNative, FC2Version^ mgd)
- static void ToMgd (FlyCapture2::IPAddress *pNative, System::Net::IPAddress^ %mgd)
- static void ToNative (System::Net::IPAddress^ mgd, FlyCapture2::IPAddress *pNative)
- static void ToMgd (FlyCapture2::MACAddress *pNative, System::Net::NetworkInformation::PhysicalAddress^ %mgd)

- static void ToNative (System::Net::NetworkInformation::PhysicalAddress^ mgd, -FlyCapture2::MACAddress *pNative)
- static void ToMgd (FlyCapture2::GigEProperty *pNative, GigEProperty^ mgd)
- static void ToNative (GigEProperty^ mgd, FlyCapture2::GigEProperty *pNative)
- static void ToMgd (FlyCapture2::GigEImageSettingsInfo *pNative, GigEImageSettingsInfo^ mgd)
- static void ToNative (GigEImageSettingsInfo^ mgd, FlyCapture2::GigEImageSettingsInfo *pNative)
- static void ToMgd (FlyCapture2::GigEImageSettings *pNative, GigEImageSettings^ mgd)
- static void ToNative (GigEImageSettings^ mgd, FlyCapture2::GigEImageSettings *pNative)
- static void Translate::ToMgd (FlyCapture2::GigEConfig *pNative, GigEConfig^ mgd)
- static void Translate::ToNative (GigEConfig^ mgd, FlyCapture2::GigEConfig *pNative)
- static void ToMgd (FlyCapture2::GigEStreamChannel *pNative, GigEStreamChannel^ mgd)
- static void ToNative (GigEStreamChannel^ mgd, FlyCapture2::GigEStreamChannel *pNative)
- static void ToMgd (FlyCapture2::CameraStats *pNative, CameraStats^ mgd)

8.48.1 Member Function Documentation

8.48.1.1 void ToMgd (FlyCapture2::FC2Config * *pNative*, FC2Config^ *mgd*) [static, package]

8.48.1.2 void ToMgd (FlyCapture2::PropertyInfo * *pNative*, CameraPropertyInfo^ *mgd*) [static, package]

8.48.1.3 void ToMgd (FlyCapture2::Property * *pNative*, CameraProperty^ *mgd*) [static, package]

8.48.1.4 void ToMgd (FlyCapture2::TriggerModeInfo * *pNative*, TriggerModeInfo^ *mgd*) [static, package]

8.48.1.5 void ToMgd (FlyCapture2::TriggerMode * *pNative*, TriggerMode^ *mgd*) [static, package]

8.48.1.6 void ToMgd (FlyCapture2::StrobeInfo * *pNative*, StrobeInfo^ *mgd*) [static, package]

8.48.1.7 void ToMgd (FlyCapture2::StrobeControl * *pNative*, StrobeControl^ *mgd*) [static, package]

8.48.1.8 void ToMgd (FlyCapture2::Format7ImageSettings * *pNative*, Format7ImageSettings^ *mgd*) [static, package]

8.48.1.9 void ToMgd (FlyCapture2::Format7Info * *pNative*, **Format7Info**^ *mgd*) [static, package]

8.48.1.10 void ToMgd (FlyCapture2::Format7PacketInfo * *pNative*, **Format7PacketInfo**^ *mgd*) [static, package]

8.48.1.11 void ToMgd (FlyCapture2::TimeStamp * *pNative*, **TimeStamp**^ *mgd*) [static, package]

8.48.1.12 void ToMgd (FlyCapture2::ConfigROM * *pNative*, **ConfigROM**^ *mgd*) [static, package]

8.48.1.13 void ToMgd (FlyCapture2::CameraInfo * *pNative*, **CameraInfo**^ *mgd*) [static, package]

8.48.1.14 void ToMgd (FlyCapture2::ImageMetadata * *pNative*, **ImageMetadata**^ *mgd*) [static, package]

8.48.1.15 void ToMgd (FlyCapture2::LUTData * *pNative*, **LutData**^ *mgd*) [static, package]

8.48.1.16 void ToMgd (FlyCapture2::EmbeddedImageInfoProperty * *pNative*, **EmbeddedImageInfoProperty**^ *mgd*) [static, package]

8.48.1.17 void ToMgd (FlyCapture2::EmbeddedImageInfo * *pNative*, **EmbeddedImageInfo**^ *mgd*) [static, package]

8.48.1.18 void ToMgd (FlyCapture2::PNGOption * *pNative*, **PngOption**^ *mgd*) [static, package]

8.48.1.19 void ToMgd (FlyCapture2::PPMOption * *pNative*, **PpmOption**^ *mgd*) [static, package]

8.48.1.20 void ToMgd (FlyCapture2::PGMOption * *pNative*, **PgmOption**^ *mgd*) [static, package]

8.48.1.21 void ToMgd (FlyCapture2::TIFFOption * *pNative*, **TiffOption**^ *mgd*) [static, package]

8.48.1.22 void ToMgd (FlyCapture2::JPEGOption * *pNative*, **JpegOption**^ *mgd*) [static, package]

8.48.1.23 void ToMgd (FlyCapture2::JPG2Option * *pNative*, **Jpg2Option**^ *mgd*) [static, package]

8.48.1.24 void ToMgd (FlyCapture2::BMPOption * *pNative*, **BMPOption**^ *mgd*) [static, package]

8.48.1.25 void ToMgd (FlyCapture2::SystemInfo * *pNative*, SystemInfo^ *mgd*) [static, package]

8.48.1.26 void ToMgd (FlyCapture2::FC2Version * *pNative*, FC2Version^ *mgd*) [static, package]

8.48.1.27 void ToMgd (FlyCapture2::IPAddress * *pNative*, System::Net::IPAddress^ % *mgd*) [static, package]

8.48.1.28 void ToMgd (FlyCapture2::MACAddress * *pNative*, System::Net::-NetworkInformation::PhysicalAddress^ % *mgd*) [static, package]

8.48.1.29 void ToMgd (FlyCapture2::GigEProperty * *pNative*, GigEProperty^ *mgd*) [static, package]

8.48.1.30 void ToMgd (FlyCapture2::GigEImageSettingsInfo * *pNative*, GigEImageSettingsInfo^ *mgd*) [static, package]

8.48.1.31 void ToMgd (FlyCapture2::GigEImageSettings * *pNative*, GigEImageSettings^ *mgd*) [static, package]

8.48.1.32 void ToMgd (FlyCapture2::GigEStreamChannel * *pNative*, GigEStreamChannel^ *mgd*) [static, package]

8.48.1.33 void ToMgd (FlyCapture2::CameraStats * *pNative*, CameraStats^ *mgd*) [static, package]

8.48.1.34 void ToNative (FC2Config^ *mgd*, FlyCapture2::FC2Config * *pNative*) [static, package]

8.48.1.35 void ToNative (CameraPropertyInfo^ *mgd*, FlyCapture2::PropertyInfo * *pNative*) [static, package]

8.48.1.36 void ToNative (CameraProperty^ *mgd*, FlyCapture2::Property * *pNative*) [static, package]

8.48.1.37 void ToNative (TriggerModeInfo^ *mgd*, FlyCapture2::TriggerModeInfo * *pNative*) [static, package]

8.48.1.38 void ToNative (TriggerMode^ *mgd*, FlyCapture2::TriggerMode * *pNative*) [static, package]

8.48.1.39 void ToNative (StrobeInfo^ *mgd*, FlyCapture2::StrobeInfo * *pNative*) [static, package]

8.48.1.40 void ToNative (StrobeControl^ *mgd*, FlyCapture2::StrobeControl * *pNative*) [static, package]

8.48.1.41 void ToNative (**Format7ImageSettings**^ *mgd,* FlyCapture2::Format7ImageSettings * *pNative*) [static, package]

8.48.1.42 void ToNative (**Format7Info**^ *mgd,* FlyCapture2::Format7Info * *pNative*) [static, package]

8.48.1.43 void ToNative (**Format7PacketInfo**^ *mgd,* FlyCapture2::Format7PacketInfo * *pNative*) [static, package]

8.48.1.44 void ToNative (**TimeStamp**^ *mgd,* FlyCapture2::TimeStamp * *pNative*) [static, package]

8.48.1.45 void ToNative (**ConfigROM**^ *mgd,* FlyCapture2::ConfigROM * *pNative*) [static, package]

8.48.1.46 void ToNative (**CameraInfo**^ *mgd,* FlyCapture2::CameraInfo * *pNative*) [static, package]

8.48.1.47 void ToNative (**ImageMetadata**^ *mgd,* FlyCapture2::ImageMetadata * *pNative*) [static, package]

8.48.1.48 void ToNative (**LutData**^ *mgd,* FlyCapture2::LUTData * *pNative*) [static, package]

8.48.1.49 void ToNative (**EmbeddedImageInfoProperty**^ *mgd,* FlyCapture2::EmbeddedImageInfoProperty * *pNative*) [static, package]

8.48.1.50 void ToNative (**EmbeddedImageInfo**^ *mgd,* FlyCapture2::EmbeddedImageInfo * *pNative*) [static, package]

8.48.1.51 void ToNative (**PngOption**^ *mgd,* FlyCapture2::PNGOption * *pNative*) [static, package]

8.48.1.52 void ToNative (**PpmOption**^ *mgd,* FlyCapture2::PPMOption * *pNative*) [static, package]

8.48.1.53 void ToNative (**PgmOption**^ *mgd,* FlyCapture2::PGMOption * *pNative*) [static, package]

8.48.1.54 void ToNative (**TiffOption**^ *mgd,* FlyCapture2::TIFFOption * *pNative*) [static, package]

8.48.1.55 void ToNative (**JpegOption**^ *mgd,* FlyCapture2::JPEGOption * *pNative*) [static, package]

8.48.1.56 void ToNative (**Jpg2Option**^ *mgd,* FlyCapture2::JPG2Option * *pNative*) [static, package]

8.48.1.57 void ToNative (**BMPOption**^ *mgd*, FlyCapture2::BMPOption * *pNative*) [static, package]

8.48.1.58 void ToNative (System::Net::IPAddress^ *mgd*, FlyCapture2::IPAddress * *pNative*) [static, package]

8.48.1.59 void ToNative (System::Net::NetworkInformation::PhysicalAddress^ *mgd*, FlyCapture2::MACAddress * *pNative*) [static, package]

8.48.1.60 void ToNative (**GigEProperty**^ *mgd*, FlyCapture2::GigEProperty * *pNative*) [static, package]

8.48.1.61 void ToNative (**GigEImageSettingsInfo**^ *mgd*, FlyCapture2::GigEImageSettingsInfo * *pNative*) [static, package]

8.48.1.62 void ToNative (**GigEImageSettings**^ *mgd*, FlyCapture2::GigEImageSettings * *pNative*) [static, package]

8.48.1.63 void ToNative (**GigEStreamChannel**^ *mgd*, FlyCapture2::GigEStreamChannel * *pNative*) [static, package]

8.48.1.64 **ErrorType** translate (FlyCapture2::ErrorType *errorType*) [static, package]

8.48.1.65 FlyCapture2::ErrorType translate (**ErrorType** *errorType*) [static, package]

8.48.1.66 **GrabMode** translate (FlyCapture2::GrabMode *grabMode*) [static, package]

8.48.1.67 FlyCapture2::GrabMode translate (**GrabMode** *grabMode*) [static, package]

8.48.1.68 **BandwidthAllocation** translate (FlyCapture2::BandwidthAllocation *bandwidthAllocation*) [static, package]

8.48.1.69 FlyCapture2::BandwidthAllocation translate (**BandwidthAllocation** *bandwidthAllocation*) [static, package]

8.48.1.70 **InterfaceType** translate (FlyCapture2::InterfaceType *interfaceType*) [static, package]

8.48.1.71 FlyCapture2::InterfaceType translate (**InterfaceType** *interfaceType*) [static, package]

8.48.1.72 **DriverType** translate (FlyCapture2::DriverType *driverType*) [static, package]

8.48.1.73 FlyCapture2::DriverType translate (**DriverType** *driverType*) [static, package]

8.48.1.74 **PropertyType** translate (FlyCapture2::PropertyType *propertyType*) [static, package]

8.48.1.75 FlyCapture2::PropertyType translate (**PropertyType** *propertyType*) [static, package]

8.48.1.76 **FrameRate** translate (FlyCapture2::FrameRate *frmRate*) [static, package]

8.48.1.77 FlyCapture2::FrameRate translate (**FrameRate** *frmRate*) [static, package]

8.48.1.78 **VideoMode** translate (FlyCapture2::VideoMode *videoMode*) [static, package]

8.48.1.79 FlyCapture2::VideoMode translate (**VideoMode** *videoMode*) [static, package]

8.48.1.80 **PixelFormat** translate (FlyCapture2::PixelFormat *pixelFormat*) [static, package]

8.48.1.81 FlyCapture2::PixelFormat translate (**PixelFormat** *pixelFormat*) [static, package]

8.48.1.82 **BayerTileFormat** translate (FlyCapture2::BayerTileFormat *bayerFormat*) [static, package]

8.48.1.83 FlyCapture2::BayerTileFormat translate (**BayerTileFormat** *bayerFormat*) [static, package]

8.48.1.84 **Mode** translate (FlyCapture2::Mode *mode*) [static, package]

8.48.1.85 FlyCapture2::Mode translate (**Mode** *mode*) [static, package]

8.48.1.86 **BusSpeed** translate (FlyCapture2::BusSpeed *busSpeed*) [static, package]

8.48.1.87 FlyCapture2::BusSpeed translate (**BusSpeed** *busSpeed*) [static, package]

8.48.1.88 **PCIeBusSpeed** translate (FlyCapture2::PCIeBusSpeed *pcieBusSpeed*) [static, package]

8.48.1.89 FlyCapture2::PCIeBusSpeed translate (**PCIeBusSpeed** *pcieBusSpeed*) [static, package]

8.48.1.90 **ColorProcessingAlgorithm** translate (FlyCapture2::ColorProcessingAlgorithm *algorithm*) [static, package]

8.48.1.91 FlyCapture2::ColorProcessingAlgorithm translate (**ColorProcessingAlgorithm** *algorithm*) [static, package]

8.48.1.92 **ImageFileFormat** translate (FlyCapture2::ImageFileFormat *fileFmt*) [static, package]

8.48.1.93 FlyCapture2::ImageFileFormat translate (**ImageFileFormat** *fileFmt*) [static, package]

8.48.1.94 **TiffOption::CompressionMethod** translate (FlyCapture2-::TIFFOption::CompressionMethod *method*) [static, package]

8.48.1.95 FlyCapture2::TIFFOption::CompressionMethod translate (**TiffOption::CompressionMethod** *method*) [static, package]

8.48.1.96 **StatisticsChannel** translate (FlyCapture2::ImageStatistics::StatisticsChannel *channel*) [static, package]

8.48.1.97 FlyCapture2::ImageStatistics::StatisticsChannel translate (**StatisticsChannel** *channel*) [static, package]

8.48.1.98 **OSType** translate (FlyCapture2::OSType *osType*) [static, package]

8.48.1.99 FlyCapture2::OSType translate (**OSType** *osType*) [static, package]

8.48.1.100 **ByteOrder** translate (FlyCapture2::ByteOrder *byteOrder*) [static, package]

8.48.1.101 FlyCapture2::ByteOrder translate (**ByteOrder** *byteOrder*) [static, package]

8.48.1.102 **GigEPropertyType** translate (FlyCapture2::GigEPropertyType *propType*) [static, package]

8.48.1.103 FlyCapture2::GigEPropertyType translate (**GigEPropertyType** *propType*) [static, package]

8.48.1.104 static void Translate::ToMgd (FlyCapture2::GigEConfig * *pNative*, **GigEConfig**^ *mgd*) [static, package]

8.48.1.105 static void Translate::ToNative (**GigEConfig**^ *mgd*, FlyCapture2::GigEConfig * *pNative*) [static, package]

8.49 TriggerMode Struct Reference

A camera trigger.

Properties

- bool onOff

  Flag controlling on/off.
- unsigned int polarity

  Polarity value.
- unsigned int source

  Source value.
- unsigned int mode

  Mode value.
- unsigned int parameter

  Parameter value.

8.49.1 Detailed Description

A camera trigger.

8.49.2 Property Documentation

8.49.2.1 unsigned int mode

Mode value.

8.49.2.2 bool onOff

Flag controlling on/off.

8.49.2.3 unsigned int parameter

Parameter value.

8.49.2.4 unsigned int polarity

Polarity value.

8.49.2.5 unsigned int source

Source value.

8.50 TriggerModeInfo Struct Reference

Information about a camera trigger property.

Properties

- bool present

  Presence of trigger mode.
- bool readOutSupported

  Flag indicating if trigger value can be read out.
- bool onOffSupported

  Flag indicating if on/off is supported.
- bool polaritySupported

  Flag indicating if polarity is supported.
- bool valueReadable

  Flag indicating if the value is readable.
- unsigned int sourceMask

  Source mask.
- bool softwareTriggerSupported

  Flag indicating if software trigger is supported.
- unsigned int modeMask

  Mode mask.

8.50.1 Detailed Description

Information about a camera trigger property.

8.50.2 Property Documentation

8.50.2.1 unsigned int modeMask

Mode mask.

8.50.2.2 bool onOffSupported

Flag indicating if on/off is supported.

8.50.2.3 bool polaritySupported

Flag indicating if polarity is supported.

8.50.2.4 bool present

Presence of trigger mode.

8.50.2.5 bool readOutSupported

Flag indicating if trigger value can be read out.

8.50.2.6 bool softwareTriggerSupported

Flag indicating if software trigger is supported.

8.50.2.7 unsigned int sourceMask

Source mask.

8.50.2.8 bool valueReadable

Flag indicating if the value is readable.

Index