
PyCapture2 API Reference

Release 2.11

FLIR Integrated Imaging Solutions, Inc

Feb 22, 2017

CONTENTS:

1	Introduction	1
2	Software Licensing Information	3
3	Enumerated Values	5
3.1	PyCapture2.IMAGE_FILE_FORMAT	6
3.2	PyCapture2.PIXEL_FORMAT	6
3.3	PyCapture2.INTERFACE_TYPE	7
3.4	PyCapture2.DRIVER_TYPE	8
3.5	PyCapture2.BANDWIDTH_ALLOCATION	8
3.6	PyCapture2.GRAB_MODE	9
3.7	PyCapture2.BUS_SPEED	9
3.8	PyCapture2.PROPERTY_TYPE	10
3.9	PyCapture2.VIDEO_MODE	11
3.10	PyCapture2.FRAMERATE	12
3.11	PyCapture2.MODE	12
3.12	PyCapture2.GIGE_PROPERTY_TYPE	13
3.13	PyCapture2.PCIE_BUS_SPEED	14
3.14	PyCapture2.BAYER_FORMAT	14
3.15	PyCapture2.COLOR_PROCESSING	14
3.16	PyCapture2.STATISTICS_CHANNEL	15
3.17	PyCapture2.TIFF_COMPRESSION	15
3.18	PyCapture2.OS_TYPE	16
3.19	PyCapture2.BYTE_ORDER	16
3.20	PyCapture2.NODE_TYPE	16
3.21	PyCapture2.PORT_TYPE	17
3.22	PyCapture2.BUS_CALLBACK_TYPE	17
4	Data Storage Classes	19
4.1	PyCapture2.Fc2error	20
4.2	PyCapture2.ConfigROM	20
4.3	PyCapture2.CameraInfo	21
4.4	PyCapture2.AvailableImageInfo	21
4.5	PyCapture2.EmbeddedImageInfo	22
4.6	PyCapture2.Config	23
4.7	PyCapture2.TriggerMode	23
4.8	PyCapture2.PropertyInfo	24
4.9	PyCapture2.Property	25
4.10	PyCapture2.StrobeInfo	25
4.11	PyCapture2.StrobeControl	26

4.12	PyCapture2.LUTData	26
4.13	PyCapture2.TimeStamp	26
4.14	PyCapture2.CameraStats	27
4.15	PyCapture2.Format7Info	28
4.16	PyCapture2.Format7ImageSettings	28
4.17	PyCapture2.Format7PacketInfo	29
4.18	PyCapture2.GigEProperty	29
4.19	PyCapture2.GigEImageSettingsInfo	30
4.20	PyCapture2.GigEImageSettings	30
4.21	PyCapture2.GigEStreamChannel	30
4.22	PyCapture2.GigEConfig	31
4.23	PyCapture2.SystemInfo	31
4.24	PyCapture2.CallbackData	32
4.25	PyCapture2.PNGOption	32
4.26	PyCapture2.PPMOption	32
4.27	PyCapture2.PGMOption	33
4.28	PyCapture2.TIFFOption	33
4.29	PyCapture2.JPEGOption	33
4.30	PyCapture2.JPG2Option	33
4.31	PyCapture2.BMPOption	33
5	Bus Manager Class	35
6	Camera Classes	39
6.1	PyCapture2.BaseCamera	39
6.2	PyCapture2.Camera	47
6.3	PyCapture2.GigECamera	50
7	AVI Recorder Class	55
8	Image Statistics Class	57
9	Topology Class	61
10	Utility Functions	63
10.1	PyCapture2.getLibraryVersion	63
10.2	PyCapture2.startSyncCapture	63
10.3	PyCapture2.getRegisterString	63
10.4	PyCapture2.setDefaultColorProcessing	64
10.5	PyCapture2.getDefaultColorProcessing	64
10.6	PyCapture2.getDefaultOutputFormat	64
10.7	PyCapture2.setDefaultOutputFormat	64
10.8	PyCapture2.determineBitsPerPixel	65
10.9	PyCapture2.checkDriver	65
10.10	PyCapture2.getDriverDeviceName	65
10.11	PyCapture2.getSystemInfo	65
10.12	PyCapture2.launchBrowser	65
10.13	PyCapture2.launchHelp	66
10.14	PyCapture2.launchCommand	66
10.15	PyCapture2.launchCommandAsync	66
11	PyCapture2 Module	67
	Python Module Index	121

INTRODUCTION

A wrapper for FLIR Integrated Imaging Solutions' FlyCapture 2 library.

This module wraps the FlyCapture 2 C library. It is available on our website <https://www.ptgrey.com/support/downloads>, but is only necessary if this is built from source (not a wheel).

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	LGPLv2.1 License https://www.ffmpeg.org/legal.html
FreeImage	FreeImage public license http://freeimage.sourceforge.net/freeimage-license.txt

ENUMERATED VALUES

A python implementation of enums. These are used by several functions to set their values correctly.

- *PyCapture2.IMAGE_FILE_FORMAT*
- *PyCapture2.PIXEL_FORMAT*
- *PyCapture2.INTERFACE_TYPE*
- *PyCapture2.DRIVER_TYPE*
- *PyCapture2.BANDWIDTH_ALLOCATION*
- *PyCapture2.GRAB_MODE*
- *PyCapture2.BUS_SPEED*
- *PyCapture2.PROPERTY_TYPE*
- *PyCapture2.VIDEO_MODE*
- *PyCapture2.FRAMERATE*
- *PyCapture2.MODE*
- *PyCapture2.GIGE_PROPERTY_TYPE*
- *PyCapture2.PCIE_BUS_SPEED*
- *PyCapture2.BAYER_FORMAT*
- *PyCapture2.COLOR_PROCESSING*
- *PyCapture2.STATISTICS_CHANNEL*
- *PyCapture2.TIFF_COMPRESSION*
- *PyCapture2.OS_TYPE*
- *PyCapture2.BYTE_ORDER*
- *PyCapture2.NODE_TYPE*
- *PyCapture2.PORT_TYPE*
- *PyCapture2.BUS_CALLBACK_TYPE*

3.1 PyCapture2.IMAGE_FILE_FORMAT

class `PyCapture2.IMAGE_FILE_FORMAT`
File formats to be used for saving images to disk.

FROM_FILE_EXT
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.

3.2 PyCapture2.PIXEL_FORMAT

class `PyCapture2.PIXEL_FORMAT`
Pixel formats available for Format7 modes.

MONO8
8 bits of mono information.

YUV8_411
YUV 4:1:1.

YUV8_422
YUV 4:2:2.

444YUV8_444
YUV 4:4:4.

RGB8
R = G = B = 8 bits.

MONO16
16 bits of mono information.

RGB16
R = G = B = 16 bits.

S_MONO16
16 bits of signed mono information.

S_RGB16

R = G = B = 16 bits signed.

RAW8

8 bit raw data output of sensor.

RAW16

16 bit raw data output of sensor.

MONO12

12 bits of mono information.

RAW12

12 bit raw data output of sensor.

BGR

24 bit BGR.

BGRU

32 bit BGRU.

RGB

24 bit RGB.

RGBU

32 bit RGBU.

BGR16

R = G = B = 16 bits.

BGRU16

64 bit BGRU.

YUV8_JPEG_422

JPEG compressed stream.

NUM_PIXEL_FORMATS

Number of pixel formats.

UNSPECIFIED_PIXEL_FORMAT

Unspecified pixel format.

3.3 PyCapture2.INTERFACE_TYPE

class `PyCapture2.INTERFACE_TYPE`

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

IEEE1394

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

USB_2

USB 2.0.

USB_3

USB 3.0.

GIGE

GigE.

UNKNOWN

Unknown interface.

3.4 PyCapture2.DRIVER_TYPE

class `PyCapture2.DRIVER_TYPE`

Types of low level drivers that FlyCapture uses.

CAM_1394

PGRCam.sys.

PRO_1394

PGR1394.sys.

JUJU_1394

firewire_core.

VIDEO1394

video1394.

RAW1394

raw1394.

USB_NONE

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

USB_CAM

PGRUsbCam.sys.

USB3_PRO

PGRXHCl.sys.

GIGE_NONE

no GigE drivers used, MS/BSD stack.

GIGE_FILTER

PGRGigE.sys.

GIGE_PRO

PGRGigEPro.sys.

GIGE_LWF

PgrLwf.sys.

UNKNOWN

Unknown driver type.

3.5 PyCapture2.BANDWIDTH_ALLOCATION

class `PyCapture2.BANDWIDTH_ALLOCATION`

Bandwidth allocation options for 1394 devices.

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

Not specified. This leaves the current setting unchanged.

3.6 PyCapture2.GRAB_MODE

class `PyCapture2.GRAB_MODE`

The grab strategy employed during image transfer.

This type controls how images that stream off the camera accumulate in a user buffer for handling.

DROP_FRAMES

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.

BUFFER_FRAMES

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_GRAB_MODE `Unspecified grab mode.`

3.7 PyCapture2.BUS_SPEED

class `PyCapture2.BUS_SPEED`

Bus speeds.

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.

BASE_T_10

10Base-T. Only for GigE cameras.

BASE_T_100

100Base-T. Only for GigE cameras.

BASE_T_1000

1000Base-T (Gigabit Ethernet). Only for GigE cameras.

BASE_T_10000

10000Base-T. Only for GigE cameras.

S_FASTEST

The fastest speed available.

ANY

Any speed that is available.

SPEED_UNKNOWN

Unknown bus speed.

3.8 PyCapture2.PROPERTY_TYPE

class `PyCapture2.PROPERTY_TYPE`

Camera properties.

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

BRIGHTNESS

AUTO_EXPOSURE

SHARPNESS

WHITE_BALANCE

HUE

SATURATION

GAMMA

IRIS

FOCUS

ZOOM

PAN

TILT

SHUTTER

GAIN

TRIGGER_MODE

TRIGGER_DELAY

FRAME_RATE

TEMPERATURE

UNSPECIFIED_PROPERTY_TYPE

3.9 PyCapture2.VIDEO_MODE

class `PyCapture2.VIDEO_MODE`

DCAM video modes.

VM_160x120YUV444

160x120 YUV444.

VM_320x240YUV422

320x240 YUV422.

VM_640x480YUV411

640x480 YUV411.

VM_640x480YUV422

640x480 YUV422.

VM_640x480RGB

640x480 24-bit RGB.

VM_640x480Y8

640x480 8-bit.

VM_640x480Y16

640x480 16-bit

VM_800x600YUV422

800x600 YUV422.

VM_800x600RGB

800x600 RGB.

VM_800x600Y8

800x600 8-bit.

VM_800x600Y16

800x600 16-bit.

VM_1024x768YUV422

1024x768 YUV422.

VM_1024x768RGB

1024x768 RGB.

VM_1024x768Y8

1024x768 8-bit.

VM_1024x768Y16

1024x768 16-bit.

VM_1280x960YUV422

1280x960 YUV422.

VM_1280x960RGB

1280x960 RGB.

VM_1280x960Y8

1280x960 8-bit.

VM_1280x960Y16

1280x960 16-bit.

VM_1600x1200YUV422
1600x1200 YUV422.

VM_1600x1200RGB
1600x1200 RGB.

VM_VM_1600x1200Y8
1600x1200 8-bit.

VM_1600x1200Y16
1600x1200 16-bit.

FORMAT7
Custom video mode for Format7 functionality.

NUM_VIDEOMODES
Number of possible video modes

3.10 PyCapture2.FRAME_RATE

class `PyCapture2.FRAME_RATE`
Frame rates in frames per second.

FR_1_875
1.875 fps.

FR_3_75
3.75 fps.

FR_7_5
7.5 fps.

FR_15
15 fps.

FR_30
30 fps.

FR_60
60 fps.

FR_120
120 fps.

FR_240
240 fps.

FORMAT7
Custom frame rate for Format7 functionality.

3.11 PyCapture2.MODE

class `PyCapture2.MODE`
Camera modes for DCAM formats as well as Format7.

FC2_MODE_0

FC2_MODE_1

FC2_MODE_2

FC2_MODE_3

FC2_MODE_4

FC2_MODE_5

FC2_MODE_6

FC2_MODE_7

FC2_MODE_8

FC2_MODE_9

FC2_MODE_10

FC2_MODE_11

FC2_MODE_12

FC2_MODE_13

FC2_MODE_14

FC2_MODE_15

FC2_MODE_16

FC2_MODE_17

FC2_MODE_18

FC2_MODE_19

FC2_MODE_20

FC2_MODE_21

FC2_MODE_22

FC2_MODE_23

FC2_MODE_24

FC2_MODE_25

FC2_MODE_26

FC2_MODE_27

FC2_MODE_28

FC2_MODE_29

FC2_MODE_30

FC2_MODE_31

FC2_NUM_MODES

Number of modes.

3.12 PyCapture2.GIGE_PROPERTY_TYPE

class `PyCapture2.GIGE_PROPERTY_TYPE`

Possible properties that can be queried from the camera.

GIGE_HEARTBEAT
GIGE_HEARTBEAT_TIMEOUT
GIGE_PACKET_SIZE
GIGE_PACKET_DELAY

3.13 PyCapture2.PCIE_BUS_SPEED

class PyCapture2.**PCIE_BUS_SPEED**
Speed of PCIE busses.

FC2_PCIE_BUSSPEED_2_5
2.5 Gb/s

FC2_PCIE_BUSSPEED_5_0
5.0 Gb/s

FC2_PCIE_BUSSPEED_UNKNOWN
Speed is unknown

3.14 PyCapture2.BAYER_FORMAT

class PyCapture2.**BAYER_FORMAT**
Bayer tile formats.

NONE
No bayer tile format.

RGGB
Red-Green-Green-Blue.

GRBG
Green-Red-Blue-Green.

GBRG
Green-Blue-Red-Green.

BGGR; Blue-Green-Green-Red.

3.15 PyCapture2.COLOR_PROCESSING

class PyCapture2.**COLOR_PROCESSING**
Color processing algorithms.

Please refer to our knowledge base at article at <http://www.ptgrey.com/KB/10141> for complete details for each algorithm.

DEFAULT
Default method.

NO_COLOR_PROCESSING
No color processing.

NEAREST_NEIGHBOR_FAST

Fastest but lowest quality. Equivalent to FLYCAPTURE_NEAREST_NEIGHBOR_FAST in FlyCapture.

EDGE_SENSING

Weights surrounding pixels based on localized edge orientation.

HQ_LINEAR

Well-balanced speed and quality.

RIGOROUS

Slowest but produces good results.

IPP

Multithreaded with similar results to edge sensing.

DIRECTIONAL

Best quality but much faster than rigorous

3.16 PyCapture2.STATISTICS_CHANNEL

class `PyCapture2.STATISTICS_CHANNEL`

Channels that allow statistics to be calculated.

GREY**RED****GREEN****BLUE****HUE****SATURATION****LIGHTNESS**

3.17 PyCapture2.TIFF_COMPRESSION

class `PyCapture2.TIFF_COMPRESSION`

TIFF compression method.

NONE

Save without any compression.

PACKBITS

Save using PACKBITS compression.

DEFLATE

Save using DEFLATE compression (ZLIB compression).

ADOBE_DEFLATE

Save using ADOBE DEFLATE compression.

CCITT_FAX3

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.

3.18 PyCapture2.OS_TYPE

class `PyCapture2.OS_TYPE`

Possible operating systems.

WINDOWS_X86

All Windows 32-bit variants.

WINDOWS_X64

All Windows 64-bit variants.

LINUX_X86

All Linux 32-bit variants.

LINUX_X64

All Linux 32-bit variants.

MAC

Mac OSX.

UNKNOWN_OS

Unknown operating system

3.19 PyCapture2.BYTE_ORDER

class `PyCapture2.BYTE_ORDER`

Possible byte order.

LITTLE_ENDIAN

BIG_ENDIAN

3.20 PyCapture2.NODE_TYPE

class `PyCapture2.NODE_TYPE`

Type of node.

NODE_COMPUTER

The node is a computer.

NODE_BUS

The node is a bus.

NODE_CAMERA

The node is a camera.

NODE_NODE

Unknown node type.

3.21 PyCapture2.PORT_TYPE

class `PyCapture2.PORT_TYPE`

Possible states of a port on a node.

PORT_NOT_CONNECTED**PORT_CONNECTED_TO_PARENT****PORT_CONNECTED_TO_CHILD**

3.22 PyCapture2.BUS_CALLBACK_TYPE

class `PyCapture2.BUS_CALLBACK_TYPE`

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

BUS_RESET

Register for all bus events.

ARRIVAL

Register for arrivals only.

REMOVAL

Register for removals only.

DATA STORAGE CLASSES

The classes used solely for data storage.

- *PyCapture2.Fc2error*
- *PyCapture2.ConfigROM*
- *PyCapture2.CameraInfo*
- *PyCapture2.AvailableImageInfo*
- *PyCapture2.EmbeddedImageInfo*
- *PyCapture2.Config*
- *PyCapture2.TriggerMode*
- *PyCapture2.PropertyInfo*
- *PyCapture2.Property*
- *PyCapture2.StrobeInfo*
- *PyCapture2.StrobeControl*
- *PyCapture2.LUTData*
- *PyCapture2.TimeStamp*
- *PyCapture2.CameraStats*
- *PyCapture2.Format7Info*
- *PyCapture2.Format7ImageSettings*
- *PyCapture2.Format7PacketInfo*
- *PyCapture2.GigEProperty*
- *PyCapture2.GigEImageSettingsInfo*
- *PyCapture2.GigEImageSettings*
- *PyCapture2.GigEStreamChannel*
- *PyCapture2.GigEConfig*
- *PyCapture2.SystemInfo*
- *PyCapture2.CallbackData*
- *PyCapture2.PNGOption*

- *PyCapture2.PPMOption*
- *PyCapture2.PGMOption*
- *PyCapture2.TIFFOption*
- *PyCapture2.JPEGOption*
- *PyCapture2.JPG2Option*
- *PyCapture2.BMPOption*

4.1 PyCapture2.Fc2error

class `PyCapture2.Fc2error` (*Exception*)
An error raised by the PyCapture2 library.

4.2 PyCapture2.ConfigROM

class `PyCapture2.ConfigROM`
camera configuration ROM class.

nodeVendorId
int – Vendor ID of a node.

chipIdHi
int – Chip ID (high part).

chipIdLo
int – Chip ID (low part).

unitSpecID
int – Unit Specification ID, usually 0xa02d

unitSWVer
int – Unit software version.

unitSubSWVer
int – Unit sub software version.

vendorUniqueInfo_0
int – first vendor unique info.

vendorUniqueInfo_1
int – second vendor unique info.

vendorUniqueInfo_2
int – third vendor unique info.

vendorUniqueInfo_3
int – last vendor unique info.

pszKeyword
str – keyword.

4.3 PyCapture2.CameraInfo

class `PyCapture2.CameraInfo`

camera information class.

serialNumber

int – The serial number of the device.

interfaceType

int – The device’s interface type. Use `PyCapture2.INTERFACE_TYPE` to set correctly.

driverType

int – The device’s driver type. Use `PyCapture2.DRIVER_TYPE` to set correctly.

isColorCamera

bool – Indicate whether the Camera can capture color.

modelName

str – The name of the device’s model name.

vendorName

str – The name of the device’s vendor.

sensorInfo

str – Details about the sensor.

sensorResolution

str – String providing the sensor resolution.

driverName

str – Name of the driver being used.

firmwareVersion

str – The firmware version of the camera.

firmwareBuildTime

str – The time the firmware was built.

maximumBusSpeed

int – Maximum bus speed. Use `PyCapture2.BUS_SPEED` to set correctly.

bayerTileFormat

int – The bayer tile format. Use `PyCapture2.BAYER_FORMAT` to set correctly.

pcieBusSpeed

int – The bus number. Use `PyCapture2.PCIE_BUS_SPEED` to set correctly. Set to 0 for GigE and USB cameras.

nodeNumber

int – ieee1394 Node number. Set to 0 for GigE and USB cameras.

busNumber

int – PCIe Bus Speed. Use `PyCapture2.PCIE_BUS_SPEED` to set correctly. Set to “UNKNOWN” for unsupported drivers.

4.4 PyCapture2.AvailableImageInfo

class `PyCapture2.AvailableImageInfo`

Structure containing the availability of image properties, for use in `EmbeddedImageInfo` class.

timestamp
bool – Whether timestamp is supported by the camera.

gain
bool – Whether gain is supported by the camera.

shutter
bool – Whether setting the shutter is supported by the camera.

brightness
bool – Whether modifying the brightness is supported by the camera.

exposure
bool – Whether modifying the exposure is supported by the camera.

whiteBalance
bool – Whether modifying the white balance of the image is supported by the camera.

frameCounter
bool – Whether the camera supports frame counting functionality.

strobePattern
bool – Whether the camera supports strobe functionality.

GPIOPinState
bool – Whether setting the GPIO pin state is supported by the camera.

ROIPosition
bool – Whether specifying the region of interest is specified by the camera.

4.5 PyCapture2.EmbeddedImageInfo

class `PyCapture2.EmbeddedImageInfo`

Structure containing the current status and availability (via the “available” attribute) of image properties.

timestamp
bool – Whether timestamping is currently active.

gain
bool – Whether gain is currently active.

shutter
bool – Whether the shutter is currently set.

brightness
bool – Whether the brightness is currently available for adjustment.

exposure
bool – Whether the exposure is currently available for adjustment.

whiteBalance
bool – Whether the white balance is currently available for adjustment.

frameCounter
bool – Whether the camera’s frame counter is active.

strobePattern
bool – Whether the strobe is currently available for adjustment.

GPIOPinState
bool – Whether the GPIOPinState is settable and gettable.

ROIPosition

bool – Whether the region of interest is available for adjustment.

available

PyCapture2.availableImageInfo – Whether these properties are available for modification. See *AvailableImageInfo* for reference.

4.6 PyCapture2.Config

class `PyCapture2.Config`

Structure containing the configuration for a camera.

numBuffers

int – The number of buffers used by the FlyCapture2 library to grab images.

numImageNotifications

int – The number of notifications per image.

minNumImageNotifications

int – The minimum number of notifications needed for the current image settings on the camera.

grabTimeout

int – Time (in milliseconds) that `camera.retrieveBuffer()` and `camera.waitForBufferEvent()` will wait for an image before timing out and returning.

grabMode

int – Grab mode for the camera. Use `PyCapture2.GRAB_MODE` to set correctly.

highPerformanceRetrieveBuffer

bool – This attribute enables `retrieveBuffer` to run in high performance mode.

isochBusSpeed

int – Isynchronous bus speed. Use `PyCapture2.BUS_SPEED` to set correctly.

asyncBusSpeed

int – Asynchronous bus speed. Use `PyCapture2.BUS_SPEED` to set correctly.

bandwidthAllocation

Bandwidth allocation flag that tells the camera the bandwidth allocation strategy to employ. Use `PyCapture2.BANDWIDTH_ALLOCATION` to set correctly.

registerTimeoutRetries

int – The number of retries to perform when a register read/write timeout is received by the library.

registerTimeout

int – Register read/write timeout value (in microseconds).

4.7 PyCapture2.TriggerMode

class `PyCapture2.TriggerMode`

Properties of a camera trigger.

Used with `Camera.setTriggerMode()`.

onOff

bool – The flag controlling activation of the trigger.

polarity
int – The polarity value.

source
int – The source value.

mode
int – The mode value.

parameter
int – The parameter value.

4.8 PyCapture2.PropertyInfo

class `PyCapture2.PropertyInfo`

Information about a specific camera property.

type
int – The type of property that is described with the following values. Use `PyCapture2.PROPERTY_TYPE` to set correctly.

present
bool – The flag indicating if the property is present.

autoSupported
bool – The flag indicating if auto is supported.

manualSupported
bool – The flag indicating if manual is supported.

onOffSupported
bool – The flag indicating if activation is supported.

onePushSupported
bool – The flag indicating if one push is supported.

absValSupported
bool – The flag indicating if absolute mode is supported.

readOutSupported
bool – The flag indicating if a property value can be read out.

min
int – Minimum value.

max
int – Maximum value.

absMin
float – Minimum value (as a float).

absMax
float – Maximum value (as a float).

units
str – Textual description of units.

unitAbbr
str – Abbreviated textual description of units.

4.9 PyCapture2.Property

class `PyCapture2.Property`

Data from a specific camera property.

Used to get and set property information.

type

PyCapture2.PROPERTY_TYPE – The type of property that is described with the following values.

present

bool – The flag controlling if the property is present on the camera.

absControl

bool – The flag controlling absolute mode (real world units) or non-absolute mode (camera internal units)

onePush

bool – The flag controlling one push.

onOff

bool – Flag controlling activation of property.

autoManualMode

bool – Flag controlling auto/manual.

ValueA

int – Value A

ValueB

int – Value B

absValue

float – Floating point value.

4.10 PyCapture2.StrobeInfo

class `PyCapture2.StrobeInfo`

Information about the camera's strobe settings and capabilities.

source

int – The source value.

present

bool – Whether strobe is present.

readOutSupported

bool – Flag indicating if the strobe value can be read.

onOffSupported

bool – Flag indicating if activating/deactivating strobe is supported.

polaritySupported

bool – Flag indicating if strobe polarity is supported.

minValue

float – Minimum value.

maxValue

float – Maximum value.

4.11 PyCapture2.StrobeControl

class `PyCapture2.StrobeControl`

Data about the camera strobe.

Used to get/set strobe settings.

source

int – Source value.

onOff

bool – Flag controlling state of strobe.

polarity

int – Signal polarity.

delay

float – Signal delay (in ms)

duration

float – Signal duration (in ms)

4.12 PyCapture2.LUTData

class `PyCapture2.LUTData`

Information about the camera's lookup table.

Used to get/set lookup table settings.

supported

bool – Flag indicating if LUT is supported.

enabled

bool – Flag indicating if LUT is enabled.

numBanks

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

numChannels

int – The number of channels per bank available.

inputBitDepth

int – The input bit depth of the LUT.

outputBitDepth

int – The output bit depth of the LUT.

numEntries

int – The number of entries in the LUT.

4.13 PyCapture2.TimeStamp

class `PyCapture2.TimeStamp`

Information detailing the time an image was taken.

seconds

long – Seconds.

microSeconds*int* – Microseconds.**cycleSeconds***int* – 1394 cycle time, in seconds.**cycleCount***int* – 1394 cycle time count.**cycleOffset***int* – 1394 cycle time offset.

4.14 PyCapture2.CameraStats

class `PyCapture2.CameraStats`

Diagnostic information about the camera.

imageDropped*int* – Number of images dropped.**imageCorrupt***int* – Number of corrupted images.**imageXmitFailed***int* – Number of times XMIT failed.**imageDriverDropped***int* – Number of times the driver dropped an image.**regReadFailed***int* – Number of times a register read failed.**regWriteFailed***int* – Number of times a register write failed.**portErrors***int* – Number of port errors.**cameraPowerUp***bool* – Whether the camera is currently on.**cameraVoltages***[float, float, float, ...]* – The values of the camera's voltage registers. Maximum length is 8.**cameraCurrents***[float, float, float, ...]* – The values of the camera's current registers. Maximum length is 8.**temperature***int* – The temperature of the camera.**timeSinceInitialization***int* – The time since the camera was initialized.**timeSinceBusReset***int* – The time since the last bus reset event.**timeStamp***PyCapture2.TimeStamp* – The timestamp of when this data was gathered.**numResendPacketsRequested***int* – The number of packets requested to be resent.

numResendPacketsReceived

int – The number of resent packets recieved.

4.15 PyCapture2.Format7Info

class `PyCapture2.Format7Info`

Format 7 information for a single mode.

Used to get a mode's format 7 capabilities and settings.

mode

int – Format 7 mode to query. Use `PyCapture2.MODE` to set correctly.

maxWidth

int – The maximum image width.

maxHeight

int – The maximum image height.

offsetHStepSize

int – Horizontal step size for the offset.

offsetVStepSize

int – Vertical stem size for the offset.

imageHStepSize

int – Horizontal step size for the image.

imageVStepSize

int – Vertical step size for the image.

pixelFormatBitField

int – Supported pixel formats in a bit field.

vendorPixelFormatBitField

int – Vendor-unique pixel formats in a bit field.

packetSize

int – Current packet size (in bytes).

minPacketSize

int – Minimum packet size (in bytes) for the current mode.

maxPacketSize

int – Maximum packet size (in bytes) for the current mode.

percentage

float – Current packet size as a percentage of maximum.

4.16 PyCapture2.Format7ImageSettings

class `PyCapture2.Format7ImageSettings`

Format 7 image settings.

Used to get/set a mode's format 7 settings.

mode

int – Format 7 mode to get/set. Use `PyCapture2.MODE` to set correctly.

offsetX

int – Horizontal image offset.

offsetY

int – Vertical image offset.

width

int – Width of image.

height

int – Height of image.

PixelFormat

int – Pixel format of image. Use `PIXEL_FORMAT` to set correctly.

4.17 PyCapture2.Format7PacketInfo

class `PyCapture2.Format7PacketInfo`

Format 7 packet information.

Used to determine the possible number of bytes per packet, as well as the recommended.

recommendedBytesPerPacket

int – Recommended bytes per packet.

maxBytesPerPacket

int – Maximum bytes per packet.

minBytesPerPacket

int – Minimum bytes per packet.

4.18 PyCapture2.GigEProperty

class `PyCapture2.GigEProperty`

A property specific to GigE cameras.

Used to get/set GigE properties.

propType

int – The type of property to get/set. Use `GIGE_PROPERTY_TYPE` to set correctly.

isReadable

bool – Whether the property is readable.

isWritable

bool – Whether the property is writable.

min

int – Minimum value of the property.

max

int – Maximum value of the property.

value

int – Current value of the property.

4.19 PyCapture2.GigEImageSettingsInfo

class `PyCapture2.GigEImageSettingsInfo`

Format 7 information for a single mode.

Used to get a mode's format 7 capabilities and settings.

maxWidth

int – Maximum image width.

maxHeight

int – Maximum image height.

offsetHStepSize

int – Horizontal step size for the offset.

offsetVStepSize

int – Vertical step size for the offset.

imageHStepSize

int – Horizontal step size for the image.

imageVStepSize

int – Vertical step size for the image.

pixelFormatBitField

int – Supported pixel formats in a bit field.

vendorPixelFormatBitField

int – Vendor unique pixel formats in a bit field.

4.20 PyCapture2.GigEImageSettings

class `PyCapture2.GigEImageSettings`

Image settings for a GigE camera.

Used to get/set a GigE Camera's settings.

offsetX

int – Horizontal image offset.

offsetY

int – Vertical image offset.

width

int – Width of image.

height

int – Height of image.

pixelFormat

int – Pixel format of image. Use `PyCapture2.PIXEL_FORMAT` to set correctly.

4.21 PyCapture2.GigEStreamChannel

class `PyCapture2.GigEStreamChannel`

Information about a single GigE stream channel.

Used to get/set stream channel info.

networkInterfaceIndex

int – Network interface index used/to use.

hostPort

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

doNotFragment

bool – Disable IP fragmentation of packets.

packetSize

int – Size of a packet (in bytes).

interPacketDelay

int – Inter-Packet delay, in timestamp counter units.

destinationIpAddress

int, int, int, int – IP address of packet destination.

sourcePort

int – Source UDP port of the stream channel.

4.22 PyCapture2.GigEConfig

class `PyCapture2.GigEConfig`

Configuration for a GigE camera.

These options should generally be set before starting isochronous transfer.

enablePacketResend

bool – Turn on/off packet resend functionality.

registerTimeoutRetries

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

registerTimeout

int – Register read/write timeout value (in microseconds).

4.23 PyCapture2.SystemInfo

class `PyCapture2.SystemInfo`

Description of the system connected to the camera.

osType

int – Operating system type. Use `OS_TYPE` to get/set correctly.

osDescription

str – Detailed description of the operating system.

byteOrder

int – Byte order of the system. Use `BYTE_ORDER` to set correctly.

sysMemSize

int – Amount of memory available on the system.

cpuDescription

str – Detailed description of the CPU.

numCpuCores
int – Number of cores on all CPUs on the system.

driverList
str – List of drivers used.

libraryList
str – List of libraries used.

gpuDescription
str – Detailed description of the GPU.

screenWidth
int – Screen resolution width, in pixels.

screenHeight
int – Screen resolution height, in pixels.

4.24 PyCapture2.CallbackData

class `PyCapture2.CallbackData`

Data returned from an event callback.

This must be the first argument for any event callback functions.

eventName
str – The event name used to register the event.

eventID
long – The device register that ‘eventName’ maps to.

eventTimestamp
long – Time as reported by the camera at which the exposure operation completed. Please note this is NOT a `PyCapture2.TimeStamp` object!

4.25 PyCapture2.PNGOption

class `PyCapture2.PNGOption`

Structure containing options for saving PNG images.

interlaced
bool – Whether to save the PNG as interlaced.

compressionLevel
int – Level of compression for the image, on the range (0-9)

4.26 PyCapture2.PPMOption

class `PyCapture2.PPMOption`

Structure containing options for saving PPM images.

binaryFile
bool – Whether to save the PPM as a binary file.

4.27 PyCapture2.PGMOption

class `PyCapture2.PGMOption`

Structure containing options for saving PGM images.

binaryFile

bool – Whether to save the PGM as a binary file.

4.28 PyCapture2.TIFFOption

class `PyCapture2.TIFFOption`

Structure containing options for saving TIFF images.

compression

int – Compression method to use for encoding. Use `TIFF_COMPRESSION` to set correctly.

4.29 PyCapture2.JPEGOption

class `PyCapture2.JPEGOption`

Structure containing options for saving JPEG images.

progressive

bool – Whether to save as a progressive JPEG file.

quality

int – JPEG image quality in range (0-100)

4.30 PyCapture2.JPG2Option

class `PyCapture2.JPG2Option`

Structure containing options for saving JPEG2000 images.

quality

int – JPEG saving quality in range (1-512)

4.31 PyCapture2.BMPOption

class `PyCapture2.BMPOption`

Structure containing options for saving Bitmap images.

indexedColor_8bit

bool – Whether to save with 8bit indexed color.

BUS MANAGER CLASS

class `PyCapture2.BusManager`

A FlyCapture 2 Bus Manager Class.

This class can be used to find GUIDs of cameras easily.

discoverGigECameras (*numCams* = 10) → *cameraInfos*

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

This is useful in situations where GigE Vision cameras are using IP addresses in a subnet different from the host's subnet. After discovering the camera, it is easy to use `ForceIPAddressToCamera()` to set a different IP configuration.

Parameters *numCams* (*int*) – The maximum number of cameras to read. Default is 10.

Returns A list containing *cameraInfo* objects.

Return type *cameraInfos* ([*PyCapture2.CameraInfo*, *PyCapture2.CameraInfo*, ..])

fireBusReset (*guid*) → None

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* (*int*, *int*, *int*, *int*) – the guid to fire a reset for.

forceAllIPAddressesAutomatically () → None

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

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

forceIPAddressToCamera (*macAddress*, *ipAddress*, *subnetMask*, *gateway*) → None

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

Parameters

- **macAddress** (*int*, *int*, *int*, *int*, *int*, *int*) – The MAC address of the camera to force IP to.
- **ipAddress** (*int*, *int*, *int*, *int*) – The IP address to force to the camera.
- **subnetMask** (*int*, *int*, *int*, *int*) – The subnet mask to force to the camera.
- **gateway** (*int*, *int*, *int*, *int*) – The default gateway to force to the camera.

getCameraFromIPAddress (*ip*) → *guid*

Return the guid of a camera, determined by its IP address.

Parameters **ip** (*int*, *int*, *int*, *int*) – The ip to get the GUID from.

Returns The guid of the specified camera.

Return type *guid* (int, int, int, int)

getCameraFromIndex (*index*) → *guid*

Return the guid of a camera, specified by its index.

Parameters **index** (*int*) – The index of the camera to get the guid for.

Returns The guid of the specified camera.

Return type *guid* (int, int, int, int)

getCameraFromSerialNumber ()

BusManager.getCameraFromIndex(serial number) -> *guid*

Return the guid of a camera, specified by its serial number.

Parameters **serialNumber** (*int*) – The serial number to get the camera from.

Returns The guid of the specified camera.

Return type *guid* (int, int, int, int)

getCameraSerialNumberFromIndex (*index*) → *serialNumber*

Return the serial number of a camera, specified by its index.

Parameters **index** (*int*) – The index of the camera to get the guid for.

Returns The serial number of the specified camera.

Return type *serialNumber* (int)

getDeviceFromIndex (*index*) → *guid*

Return the guid of a device, specified by its index.

Parameters **index** (*int*) – The index of the device to get the guid for.

Returns The guid of the specified device.

Return type *guid* (int, int, int, int)

getInterfaceTypeFromGuid (*guid*) → *interfaceType*

Return *interfaceType* of the camera associated with the guid.

Parameters **guid** (*int*, *int*, *int*, *int*) – The guid to get the interface type from.

Returns The interface type of the specified camera. Use PyCapture2.INTERFACE_TYPE to read correctly.

Return type *interfaceType* (int)

getNumOfCameras () → *count*

Return the number of cameras attached to the PC.

Returns The number of cameras attached.

Return type *count* (int)

getNumOfDevices () → *numDevices*

Return the number of devices attached to the PC.

Returns The number of attached devices.

Return type *numDevices* (int)

getTopology () → topologyNode

Get the topology information for the PC.

Returns PyCapture2.TopologyNode object that contains the topology information.

Return type topologyNode (PyCapture2.topologyNode)

getUsbLinkInfo (guid) → value

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

Parameters **guid** (int, int, int, int) – The guid device to read from.

Returns The value read from the card register.

Return type value (int)

getUsbPortStatus (guid) → value

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

Parameters **guid** (int, int, int, int) – The guid device to read from.

Returns The value read from the card register.

Return type value (int)

isCameraControllable (guid) → isControllable

Query CCP status on camera with corresponding guid.

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

Parameters **guid** (int, int, int, int) – The guid device to query.

Returns Whether the camera is controllable.

Return type isControllable (bool)

readPhyRegister (guid, page, port, address) → value

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** (int, int, int, int) – The guid device to read from.
- **page** (int) – The page to read from.
- **port** (int) – The port to read from.
- **address** (int) – The address to read from.

Returns The value of the phy register.

Return type value (int)

registerCallback (callbackType, function, *arguments) → None

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

Parameters

- **callbackType** (int) – The type of callback to register an event to. Use PyCapture2.BUS_CALLBACK_TYPE to set correctly.
- **function** (function) – The function to call when the specified event occurs.
- **arguments** – Any additional arguments will be passed to the function when it is called.

rescanBus () → None

Force a rescan of the buses.

This does not trigger a bus reset. However, any current connections to a Camera object will be invalidated.

unregisterCallback (*callbackType*) → None

Unregister a callback function.

Parameters **callbackType** (*int*) – The type of callback event to deregister. Use PyCapture2.BUS_CALLBACK_TYPE to set correctly.

writePhyRegister (*guid, page, port, address, value*) → None

Write to a phy register on the specified device.

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

Parameters

- **guid** (*int, int, int, int*) – The guid device to write to.
- **page** (*int*) – The page to write to.
- **port** (*int*) – The port to write to.
- **address** (*int*) – The address to write to.
- **value** (*int*) – The value to write.

CAMERA CLASSES

- *PyCapture2.BaseCamera*
- *PyCapture2.Camera*
- *PyCapture2.GigECamera*

6.1 PyCapture2.BaseCamera

class `PyCapture2.BaseCamera`

Base camera class. This class exists only for camera and GigECamera classes to inherit from it.

connect (*guid*) → None

Connect the camera object to a physical camera.

Parameters **guid** (*int, int, int, int*) – The GUID of the camera to connect to.

The BusManager class has several functions that return a valid GUID.

deregisterAllEvents () → None

De-register all registered events.

deregisterEvent (*eventName*) → None

De-register an event previously registered with the camera.

Parameters **eventName** (*str*) – The name of the event to deregister.

This function currently only works with EventExposureEnd events.

disconnect () → None

Disconnect the camera object from a physical camera.

enableLUT (*enable*) → None

Enable or disable LUT functionality, based on the passed bool.

Parameters **enable** (*bool*) – Whether to activate or deactivate the LUT.

fireSoftwareTrigger () → None

Fire the software trigger, according to the DCAM specifications.

getActiveLUTBank () → activeBank

Return the LUT bank that is currently being used.

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

Returns The active LUT bank.

Return type activeBank (int)

getCameraInfo () → cameraInfo

Return detailed information about the camera.

Returns The camera information.

Return type cameraInfo (*PyCapture2.CameraInfo*)

getConfiguration () → config

Return configuration properties of the camera.

Returns The configuration information.

Return type config (*PyCapture2.Config*)

getCycleTime () → timeStamp

Return a timestamp object containing 1394 CYCLE_TIME information.

Returns The TimeStamp object with time data.

Return type *timeStamp* (*PyCapture2.TimeStamp*)

getEmbeddedImageInfo ()

camera.getEmbeddedImageInfo -> embeddedImageInfo

Return the current status of each embedded image property, as well as their availability.

Returns a PyCapture2.EmbeddedImageInfo object containing the properties.

Return type embeddedImageInfo (*PyCapture2.EmbeddedImageInfo*)

getGPIOPinDirection (pin) → direction

Get GPIO pin direction for the specified pin.

Both the pin and the direction are represented by an unsigned int.

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

Parameters pin (*int*) – the pin to query.

Returns the direction of the pin.

Return type direction (int)

getLUTBankInfo (bank) → readSupport, writeSupport

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

Parameters is an int representing the bank to query. (bank) –

Returns indicates whether the bank can be read. writeSupport (bool): indicates whether the bank can be written to.

Return type readSupport (bool)

getLUTChannel (bank, channel, numEntries) → entries

Get the LUT channel settings from the camera.

Entries contains the LUT entries. It is a list of sizeEntries ints.

Parameters

- **bank** (*int*) – specifies which bank to retrieve.
- **channel** (*int*) – specifies which channel to retrieve.
- **sizeEntries** (*int*) – specifies the number of entries to read.

Returns The returned entries.

Return type entries ([int, int, ...])

getLUTInfo () → LUTData

Query if LUT support is available on the camera.

Returns A LUTData object containing the queried data.

Return type *LUTData* (*PyCapture2.LUTData*)

Note that some cameras may report support for the LUT and return an inputBitDepth of 0. In these cases use $\log_2(\text{numEntries})$ for the inputBitDepth.

getMemoryChannel () → currentChannel

Return the current memory channel from the camera.

Returns The current memory channel.

Return type currentChannel (int)

getMemoryChannelInfo () → numChannels

Query the camera for memory channel support.

If numChannels is 0, then memory channel support is not available.

Returns The number of memory channels available.

Return type *numChannels* (int)

getProperty (propertyType) → property

Return the settings for the specified property from the camera.

Parameters **propertyType** (*int*) – The type of property to return. use PyCapture2.PROPERTY_TYPE to set correctly.

Returns a Property object with the specified property information.

Return type property (*PyCapture2.Property*)

getPropertyInfo (propertyType) → propertyInfo

Retrieves information about the specified camera property.

Parameters **propertyType** (*int*) – The type of property to return. Use PyCapture2.PROPERTY_TYPE to set correctly.

Returns a PropertyInfo object with the specified property information.

Return type propertyInfo (*PyCapture2.PropertyInfo*)

getStats () → cameraStats

Return camera diagnostic information.

Returns A CameraStats object with diagnostic information.

Return type cameraStats (*PyCapture2.CameraStats*)

getStrobe () → strobeControl

Return current strobe settings of the camera.

Returns a StrobeControl object.

Return type strobeControl (*PyCapture2.StrobeControl*)

getStrobeInfo () → strobeInfo

Return strobe information from the camera.

Returns a StrobeInfo object containing the strobe information.

Return type `strobeInfo` (*PyCapture2.StrobeInfo*)

getTriggerDelay () → property

Return the current trigger delay settings of the camera.

Returns A Property object containing the trigger delay settings.

Return type property (*PyCapture2.Property*)

getTriggerDelayInfo () → PropertyInfo

Return trigger delay information from the camera.

Returns A PropertyInfo object containing the trigger delay information.

Return type propertyInfo (*PyCapture2.PropertyInfo*)

getTriggerMode () → triggerMode

Retrieve current trigger settings from the camera.

Returns (*PyCapture2.TriggerMode*): A TriggerMode object containing the camera's image settings.

Return type triggerMode

isConnected

readRegister (*address*) → value

Read the specified register from the camera.

Parameters **address** (*int*) – The address of the register to read.

Returns The value read from the register.

Return type *value* (*int*)

readRegisterBlock (*address, length*) → values

Read from the specified register block on the camera.

Parameters

- **address** (*int*) – The 48bit address to write to.
- **length** (*int*) – Amount of data to read.

registerAllEvents (*function, arguments*) → None

Register the camera to issue a custom callback on all available events.

The function must take a CallbackData object as its first argument.

Parameters

- **function** (*function*) – The function to call on event completion. It must take a CallbackData object as its first argument.
- ***arguments** – The rest of the arguments are given to the callback function when called.

registerEvent (*eventName, function, *arguments*) → None

Register the camera to issue a custom callback function for a specific device event.

The function must take a CallbackData object as its first argument.

Parameters

- **eventName** (*str*) – The type of event to register a callback for.
- **function** (*function*) – The function to call on event completion. It must take a CallbackData object as its first argument.

- ***arguments** – The rest of the arguments are given to the callback function when called.

restoreFromMemoryChannel (*channel*) → None

Restore the specified memory channel.

Parameters **channel** (*int*) – memory channel to restore from

retrieveBuffer () → image

Retrieve an image from the buffer and return it.

The method to grab with (the newest image or the oldest) is specified via the `grabMode` attribute of the `Config` class.

Returns The image object grabbed from the camera.

Return type image (*PyCapture2.Image*)

saveToMemoryChannel (*channel*) → None

Save the current settings to the specified memory channel.

Parameters **channel** (*int*) – The memory channel to save the settings to.

setActiveLUTBank (*activeBank*) → None

Take an integer and set that LUT bank as the active one.

Parameters **activeBank** (*int*) – an integer representing the LUT bank to set.

setCallback (*function*, **args*) → None

Sets a callback function to be called on the completion of image transfer.

The function **MUST** take an image object as its first argument, and any extra arguments given afterwards.

To clear the stored callback data, call `unsetCallback()`.

Parameters

- **function** (*function*) – The function that is called in a callback. This function must take an image object as its first argument.
- **arguments** – The rest of the arguments are the arguments for the callback function.

setConfiguration (*config = None*, ***kwargs*) → None

Take config properties and update camera configuration.

There are two ways to call this function: The first is with a `Config` object that has all the values to be updated.

The second is specifying each property to update in the arguments. The arguments that can be specified are:

```
numBuffers (int)
numImageNotifications (int)
minNumImageNotifications (int)
grabTimeout (int)
grabMode (int)
isochBusSpeed (int)
asyncBusSpeed (int)
bandwidthAllocation (int)
registerTimeoutRetries (int)
registerTimeout (int)
```

Information about these properties can be found in the `Config` documentation.

ex. `camera.setConfiguration(isochBusSpeed = PyCapture2.BUS_SPEED.S400)` will update the camera's isochronous bus speed.

Parameters

- **config** (`PyCapture2.Config`) – The configuration object to get all properties from.
- **kwargs** (`{...}`) – The second method of calling: specify the properties to change as keywords.

setEmbeddedImageInfo (*embeddedImageInfo = None, **kwargs*) → None

Take `embeddedImageInfo` properties and update the camera's register to match.

There are two ways to call this function: The first is with an `EmbeddedImageInfo` object that has all the values to be updated.

The second is specifying each property to update in the arguments. The arguments that can be specified are:

timestamp (bool)
gain (bool)
shutter (bool)
brightness (bool)
exposure (bool)
whiteBalance (bool)
frameCounter (bool)
strobePattern (bool)
ROIPosition (bool)

Information about these properties can be found in the `EmbeddedImageInfo` documentation.

ex. `camera.setEmbeddedImageInfo(timestamp = True)` will update the camera's timestamp setting.

Parameters

- **embeddedImageInfo** (`PyCapture2.EmbeddedImageInfo`) – The embedded image info object to get all properties from.
- **kwargs** (`{...}`) – The second method of calling: specify the properties to change as keywords.

setGPIOPinDirection (*pin, direction*) → None

Set 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** (*int*) – The pin number to set.
- **direction** (*int*) – The direction to set.

setLUTChannel (*bank, channel, data*) → None

Set the LUT channel settings to the camera.

Parameters

- **bank** (*int*) – the bank to set

- **channel** (*int*) – the channel to set
- **data** (*[int, int, ...]*) – a list containing the integers to write to the LUT channel.

setProperty (*property = None, **kwargs*) → None

Take strobe properties and update the camera's strobe.

There are two ways to call this function: The first is with a Property object containing the updated properties.

The second is by specifying each property to update in the arguments. The arguments that can be specified are:

type (*int*)
 present (*bool*)
 absControl (*bool*)
 onePush (*bool*)
 onOff (*bool*)
 autoManualMode (*bool*)
 valueA (*int*)
 valueB (*int*)
 absValue (*float*)

Note that type **MUST** be specified if this method of calling is used.

Information about these properties can be found in the Property documentation.

Ex. camera.setProperty(type = PyCapture2.PROPERTY_TYPE.ZOOM, absValue = 2.0) Sets zoom to 2.

Parameters

- **property** (*PyCapture2.Property*) – The property object to get all properties from.
- **kwargs** (*{...}*) – The second method of calling: specify the properties to change as keywords.

setStrobe (*strobeControl = None, **kwargs*) → None

Take strobe properties and update the camera's strobe.

There are two ways to call this function: The first is with a StrobeControl object containing the updated properties.

The second is by specifying each property to update in the arguments. The arguments that can be specified are:

source (*int*)
 onOff (*bool*)
 polarity (*int*)
 delay (*float*)
 duration (*float*)

Information about these properties can be found in the StrobeControl documentation.

Ex. camera.setStrobe(onOff = True) Activates the strobe.

Parameters

- **strobeControl** (`PyCapture2.StrobeControl`) – The strobe control object to get all properties from.
- **kwargs** (`{...}`) – The second method of calling: specify the properties to change as keywords.

setTriggerDelay (*property = None, **kwargs*) → None

Take trigger delay properties and update the camera.

There are two ways to call this function: The first is with a Property object that has all the triggerDelay values to be updated.

The second is by specifying each property to update the arguments. The arguments that can be specified are:

propType (int)
present (bool)
absControl (bool)
onePush (bool)
onOff (bool)
autoManualMode (bool)
valueA (int)
valueB (int)
absValue (float)

Information about these properties can be found in the Property documentation.

Ex. camera.setTriggerDelay(onOff = True) Will activate the camera trigger delay.

Parameters

- **property** (`PyCapture2.Property`) – The Property object to get all properties from.
- **kwargs** (`{...}`) – The second method of calling: specify the properties to change as keywords.

setTriggerMode (*TriggerMode = None, **kwargs*) → None

Take triggerMode properties and update the camera's trigger.

There are two ways to call this function: The first is with a TriggerMode object that has all the values to be updated.

The second is specifying each property to update in the arguments. The arguments that can be specified are:

onOff (bool)
polarity (int)
source (int)
mode (int)
parameter (int)

Information about these properties can be found in the TriggerMode documentation.

Ex. camera.setTriggerMode(onOff = True) will allow the camera to be triggered.

Parameters

- **triggerMode** (`PyCapture2.TriggerMode`) – The trigger mode object to get all properties from.
- **kwargs** (`{...}`) – The second method of calling: specify the properties to change as keywords.

startCapture (*function = None, *arguments*) → None

Start capturing data from camera to image buffers, and optionally set a callback to be run when an image is retrieved.

To start capture and retrieve images manually, call without any arguments.

To set a callback, call startCapture with a function and the function's optional arguments.

The function MUST take an image object as its first argument, and any extra arguments given afterwards.

To clear the stored callback data, call unsetCallback().

Parameters

- **function** (*function*) – The optional function that is called in a callback. This function must take an image object as its first argument.
- **arguments** – The rest of the arguments are the arguments for the callback function.

stopCapture () → None

Stop capturing data from camera.

unsetCallback () → None

Clears stored callback data.

If there is currently no stored callback data, this function does nothing.

waitForBufferEvent (*eventNumber*) → image

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

Parameters **eventNumber** (*int*) – The event number to wait for.

Returns Image object with the latest image buffer.

Return type image (*PyCapture2.Image*)

writeRegister (*address, value*) → None

Write the specified value to the specified register on the camera.

Parameters

- **address** (*int*) – The address of the register to write.
- **value** (*int*) – The value to write to the register.

writeRegisterBlock (*address, values*) → None

Write to the specified register block on the camera.

Parameters

- **address** (*int*) – The 48bit address to write to.
- **values** (*[int, int, ...]*) – List containing data to be written.

6.2 PyCapture2.Camera

class `PyCapture2.Camera` (*BaseCamera*)

A camera object from FlyCapture2. This is specifically for USB cameras.

getFormat7Configuration () → imageSettings, packetSize, percentage

Get the current Format7 configuration from the camera.

This call will raise an Fc2error if the camera is not already in Format7.

Returns

tuple containing: imageSettings (PyCapture2.Format7ImageSettings): Current image settings. packetSize (int): Current packet size. percentage (float): current packet size as a percentage of maximum packet size.

Return type (tuple)

getFormat7Info (mode) → format7Info, supported

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

Parameters *mode* (int) – Format7 mode to query. Use PyCapture2.MODE to set correctly.

Returns

tuple containing: format7Info (PyCapture2.Format7Info): A Format7Info object with capabilities and the current state of the specified mode. supported (bool): Whether the specified mode is supported.

Return type (tuple)

getVideoModeAndFrameRate () → videoMode, frameRate

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

If the camera is in Format7, the video mode will be VIDEO_MODE.FORMAT7 and the framerate will be FRAME_RATE.FORMAT7.

Returns

tuple containing: videoMode (int): Current video mode of the camera. Use VIDEO_MODE to check its value. frameRate (int): Current frame rate of the camera. Use FRAME_RATE to check its value.

Return type (tuple)

getVideoModeAndFrameRateInfo (videoMode, frameRate) → supported

Query the camera to determine if the specified video mode and framerate are supported.

Parameters

- **videoMode** (int) – Video mode to check. Use PyCapture2.VIDEO_MODE to set correctly.
- **frameRate** (int) – Frame rate to check. Use PyCapture2.FRAME_RATE to set correctly.

Returns Whether or not the specified video mode and frame rate are supported.

Return type *supported* (bool)

setFormat7Configuration (percentSpeed, imageSettings = None, **kwargs) → None

Set the given Format7 configuration to the camera.

There are two ways to call this function: The first is with a Format7ImageSettings object that has all the values to be updated.

The second is by specifying each property to update the arguments. The arguments that can be specified are:

mode (int)

offsetX (int)
 offsetY (int)
 width (int)
 height (int)
 pixelFormat (int)

Information about these properties can be found in the Format7ImageSettings documentation.

Ex. camera.setFormat7Settings(50.0, pixelFormat = PyCapture2.PIXEL_FORMAT.MONO8)
 Will set the current mode to 8bit greyscale image capture, with 50% of max packet size.

Other Arguments: percentSpeed (float): packet size, in percentage of max packet size.

imageSettings (PyCapture2.Format7ImageSettings): The image settings object to get all properties from. kwargs ({ ... }): The second method of calling: specify the properties to change as keywords.

setFormat7ConfigurationPacket ()

camera.setFormat7Configuration(packetSize, imageSettings = None, ****kwargs**) -> None

Set the given Format7 configuration to the camera.

There are two ways to call this function: The first is with a Format7ImageSettings object that has all the values to be updated.

The second is by specifying each property to update the arguments. The arguments that can be specified are:

mode (int)
 offsetX (int)
 offsetY (int)
 width (int)
 height (int)
 pixelFormat (int)

Information about these properties can be found in the Format7ImageSettings documentation.

Ex. camera.setFormat7Settings(50.0, pixelFormat = PyCapture2.PIXEL_FORMAT.MONO8)
 Will set the current mode to 8bit greyscale image capture, with 50% of max packet size.

Other Arguments: packetSize (int): packet size, in bytes.

imageSettings (PyCapture2.Format7ImageSettings): The image settings object to get all properties from. kwargs ({ ... }): The second method of calling: specify the properties to change as keywords.

setVideoModeAndFrameRate (videoMode, frameRate) → None

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

It is not possible to set the camera to VIDEO_MODE.FORMAT7 or FRAME_RATE.FORMAT7. Use the Format7 functions to use Format7 functionality.

Parameters

- **videoMode** (*int*) – Video mode to set. Use PyCapture2.VIDEO_MODE to set correctly.
- **frameRate** (*int*) – Frame rate to set. Use PyCapture2.FRAME_RATE to set correctly.

validateFormat7Settings (imageSettings = None, **kwargs**) → packetInfo, isValid**

Check whether given Format7ImageSettings properties are valid, and return packet information and whether the settings are valid.

There are two ways to call this function: The first is with a Format7ImageSettings object that has all the values to be updated.

The second is by specifying each property to update the arguments. The arguments that can be specified are:

mode (int)
offsetX (int)
offsetY (int)
width (int)
height (int)
pixelFormat (int)

Information about these properties can be found in the `Format7ImageSettings` documentation.

Ex. `camera.validateFormat7Settings(pixelFormat = PyCapture2.PIXEL_FORMAT.MONO8)`
Will check if the current mode supports 8bit greyscale image capture.

Parameters

- **imageSettings** (`PyCapture2.Format7ImageSettings`) – The property object to get all properties from.
- **kwargs** (`{...}`) – The second method of calling: specify the properties to change as keywords.

Returns

tuple containing: `packetInfo` (`PyCapture2.Format7PacketInfo`): Packet size information used to determine a valid packet size. `isValid` (bool): Whether the settings are valid.

Return type (tuple)

6.3 PyCapture2.GigECamera

class `PyCapture2.GigECamera` (*BaseCamera*)

A camera object from FlyCapture2. This is specifically for GigE and ethernet cameras.

discoverGigEPacketSize () → `packetSize`

Return the largest packet size possible for the link between the PC and the camera.

Returns The maximum packet size supported by the link.

Return type *packetSize* (int)

getGigEConfig () → `gigEConfig`

Return the camera's current GigE configuration.

Returns `GigEConfig` object containing configuration information.

Return type `gigEConfig` (*PyCapture2.GigEConfig*)

getGigEImageBinningSettings () -> (*horzValue, vertValue*)

Return the current binning settings on the camera.

Returns

tuple containing: `horzValue` (int): The current horizontal binning value. `vertValue` (int): The current vertical binning value.

Return type (tuple)

getGigEImageSettings () → `gigEImageSettings`

Return the current image settings on the camera.

Returns The current image settings on the camera.

Return type `gigEImageSettings` (*GigEImageSettings*)

getGigEImageSettingsInfo () → `gigEImageSettingsInfo`

Get information about the camera's possible image settings.

Returns `GigEImageSettingsInfo` object holding image settings information.

Return type `GigEImageSettings` (*PyCapture2.GigEImageSettingsInfo*)

getGigEImagingMode () → `mode`

Return the current imaging mode on the camera.

Returns Current imaging mode on the camera. Use `PyCapture2.MODE` to read correctly.

Return type `mode` (`int`)

getGigEProperty (*propertyType*) → `gigEProperty`

Return information on the specified GigE property.

Parameters **propertyType** (`int`) – The type of property to retrieve information about. Use `PyCapture2.PROPERTY_TYPE` to set correctly.

Returns A `GigEProperty` object containing information about the specified property.

Return type `gigEProperty` (*PyCapture2.GigEProperty*)

getGigEStreamChannelInfo (*channel*) → `gigEStreamChannel`

Return information about the specified stream channel.

Parameters **channel** (`int`) – The channel to retrieve information about.

Returns `GigEStreamChannel` object containing information about the channel.

Return type `gigEStreamChannel` (*PyCapture2.GigEStreamChannel*)

getNumStreamChannels () → `numChannels`

Return the number of stream channels present on the camera.

Returns the number of channels present.

Return type `numChannels` (`int`)

queryGigEImagingMode (*mode*) → `isSupported`

Return whether the given imaging mode is supported by the camera.

Parameters **mode** (`int`) – The mode to check. Use `PyCapture2.MODE` to set correctly.

Returns Whether the given mode is supported.

Return type `isSupported` (`bool`)

readGVCPMemory (*address*, *length*) → `values`

Read a GVCP memory block on the camera.

Parameters

- **address** (`int`) – The GVCP address to read from.
- **length** (`int`) – The size of the memory to read.

Returns The list of integers containing read data.

Return type `values` (`[int, int, ..]`)

readGVCPRegister (*address*) → `value`

Read a GVCP register's value.

Parameters **address** (*int*) – The address of the register to read.

Returns The value of the register.

Return type *value* (*int*)

readGVCPRegisterBlock (*address*, *length*) → values

Read a block of GVCP registers.

Parameters

- **address** (*int*) – The address to read from.
- **length** (*int*) – The number of registers to read.

Returns The list of values read from the registers.

Return type values ([int, int, ..])

setGigEConfig (*gigEConfig* = *None*, ***kwargs*) → None

Set the camera's GigE configuration.

There are two ways to call this function: The first is with a GigEConfig object that has all the values to be updated.

The second is by specifying each property to update in the arguments. The arguments that can be specified are:

enablePacketResend
registerTimeoutRetries
registerTimeout

Information about these properties can be found in the GigEConfig documentation.

Parameters

- **gigEConfig** (`PyCapture2.GigEConfig`) – The configuration object to get all properties from.
- **kwargs** (*{...}*) – The second method of calling: specify the properties to change as keywords.

setGigEImageBinningSettings (*horzValue*, *vertValue*) → None

Set the binning settings on the camera.

Parameters

- **horzValue** (*int*) – Horizontal binning value to set.
- **vertValue** (*int*) – Vertical binning value to set.

setGigEImageSettings (*gigEImageSettings* = *None*, ***kwargs*) → None

Set the specified image settings to the camera.

There are two ways to call this function: The first is with a GigEImageSettings object that has all the values to be updated.

The second is by specifying each property to update in the arguments. The arguments that can be specified are:

offsetX (*int*)
offsetY (*int*)
width (*int*)
height (*int*)

pixelFormat (int)

Information about these properties can be found in the GigEImageSettings documentation.

Parameters

- **imageSettings** (`PyCapture2.GigEImageSettings`) – The image settings object to get all properties from.
- **kwargs** (`{...}`) – The second method of calling: specify the properties to change as keywords.

setGigEImagingMode (*mode*) → None

Set the camera's imaging mode to the specified mode.

Parameters *mode* (*int*) – Mode to set the camera to. Use `PyCapture2.MODE` to set correctly.

setGigEProperty (*gigEProperty* = None, ****kwargs**) → None

There are two ways to call this function: The first is with a `GigEProperty` object that has all the values to be updated.

The second is by specifying each property to update in the arguments. The arguments that can be specified are:

propType (int)

isReadable (bool)

isWritable (bool)

min (int)

max (int)

value (int)

propType MUST be specified for this method of calling `setGigEProperty` to work correctly.

Information about these properties can be found in the `GigEProperty` documentation.

Parameters

- **gigEProperty** (`PyCapture2.GigEProperty`) – The property object to get all properties from.
- **kwargs** (`{...}`) – The second method of calling: specify the properties to change as keywords.

writeGVCPMemory (*address*, *values*) → None

Write a GVCP memory block on the camera.

Parameters

- **address** (*int*) – The GVCP address to write to.
- **values** (*[int, int, ...]*) – The list of integer values to write to the memory.

writeGVCPRegister (*address*, *value*) → None

Write to a GVCP (GigE Vision Control Protocol) register.

Parameters

- **address** (*int*) – GVCP address to be written to.
- **value** (*int*) – The value to be written.

writeGVCPRegisterBlock (*address*, *values*) → None

Write to a block of GVCP registers.

Parameters

- **address** (*int*) – The address to write to
- **values** (*[int, int, ...]*) – The list of integer values to write to register.

AVI RECORDER CLASS

class `PyCapture2.AVIRecorder`

A class from FlyCapture 2 used to create animated AVI and MP4 files.

AVIOpen (*filename*, *framerate*) → None

Open an AVI file for writing images to disk.

The size of AVI files is limited to 2GB. The filenames are automatically generated using the filename specified.

Parameters

- **filename** (*str*) – The name of the file to save to.
- **framerate** (*float*) – The framerate of the AVI.

H264Open (*filename*, *framerate*, *width*, *height*, *bitrate*)

Open an H264 MP4 file for writing images to disk.

The size of MP4 files is limited to 2GB. The filenames are automatically generated using the filename specified.

Parameters

- **filename** (*str*) – The name of the file to save to.
- **framerate** (*float*) – The framerate of the MP4.
- **width** (*int*) – The width of the source image.
- **height** (*int*) – The height of the source image.
- **bitrate** (*int*) – The bitrate to encode at.

MJPEGOpen (*filename*, *framerate*, *quality*) → None

Open an MJPG AVI file for writing images to disk.

The size of AVI files is limited to 2GB. The filenames are automatically generated using the filename specified.

Parameters

- **filename** (*str*) – The name of the file to save to.
- **framerate** (*float*) – The framerate of the AVI.
- **quality** (*int*) – Image quality on range (1 - 100)

append (*image*) → None

Append an image to the AVI/MP4 file.

Parameters **image** (`PyCapture2.Image`) – Image object to append to file.

close () → None

Close the AVI/MP4 file and save it to disk.

IMAGE STATISTICS CLASS

class PyCapture2.ImageStatistics

calculateStatistics (*image*) → None

Calculate image statistics from the given image.

Channels to be used must be enabled before this function call.

Parameters **image** (PyCapture2.Image) – The image to calculate statistics from.

disableAllChannels () → None

Disable all channels for image statistics analysis.

enableAllChannels () → None

Enable all channels for image statistics analysis.

enableGreyChannel () → None

Enable only the grey channel for image statistics analysis.

enableHSLChannel () → None

Enable only the hue, saturation, and lightness channels for image statistics analysis.

enableRGBChannel () → None

Enable only the red, green, and blue channels for image statistics analysis.

getChannelStatus (*channel*) → enabled

Return the status for the given channel.

Parameters **channel** (*int*) – The channel to retrieve status for. Use PyCapture2.STATISTICS_CHANNEL to set correctly.

Returns Whether the channel is currently enabled.

Return type *enabled* (bool)

getHistogram (*channel*) → histogram

Return a histogram of the values in the image.

Parameters **channel** (*int*) – The channel to create the histogram from. Use PyCapture2.STATISTICS_CHANNEL to set correctly.

Returns A list containing the histogram values. Its length is equal to the number of possible values of a pixel in the image.

Return type histogram ([int, int, ..])

getMean (*channel*) → mean

Return the mean value of a channel of the image.

Parameters **channel** (*int*) – The channel to determine the mean from. Use PyCapture2.STATISTICS_CHANNEL to set correctly.

Returns The mean value of the image.

Return type mean (float)

getNumPixelValues (*channel*) → numValues

Return the number of unique pixel values in the image.

Parameters **channel** (*int*) – The channel to determine the number of values from. Use PyCapture2.STATISTICS_CHANNEL to set correctly.

Returns The number of unique pixel values.

Return type numValues (int)

getPixelValueRange (*channel*) -> (*min*, *max*)

Get the actual range of the pixel values in a statistics channel.

Parameters **channel** (*int*) – The channel to determine the range of. Use PyCapture2.STATISTICS_CHANNEL to set correctly.

Returns

tuple containing: min (int): The minimum pixel value in the image. max (int): The maximum pixel value in the image.

Return type (tuple)

getRange (*channel*) -> (*min*, *max*)

Return the maximum and minimum possible values for any given pixel in the image.

This is generally 0-255 for 8bit images, and 0-65535 for 16bit images.

Parameters **channel** (*int*) – The channel to determine the range of. Use PyCapture2.STATISTICS_CHANNEL to set correctly.

Returns

tuple containing: min (int): The minimum value a pixel can have. max (int): The maximum value a pixel can have.

Return type (tuple)

getStatistics (*channel*) → (rangeMin, rangeMax), (pixValMin, pixValMax), numPixVals, pixValMean, histogram

Return all statistics for the given channel.

Parameters **channel** (*int*) – The channel to create the statistics from. Use PyCapture2.STATISTICS_CHANNEL to set correctly.

Returns

tuple containing: rangeMin (int): The minimum value a pixel can have. rangeMax (int): The maximum value a pixel can have. pixValMin (int): The minimum pixel value in the image. pixValMax (int): The maximum pixel value in the image. numPixVals (int): The number of unique pixel values. pixValMean (float): The mean value of the image. histogram ([int, int, ...]): A list containing the histogram values. Its length is equal to the number of possible values of a pixel in the image.

Return type (tuple)

setChannelStatus (*channel*, *enabled*) → None

Set the status of a single statistics channel.

Parameters

- **channel** (*int*) – The channel whose status to set. Use `PyCapture2.STATISTICS_CHANNEL` to set correctly.
- **enabled** (*bool*) – Whether the channel will be enabled or disabled.

TOPOLOGY CLASS

class `PyCapture2.TopologyNode`

addChild (*child*) → None

Add the specified topologyNode as a child of the node.

Parameters **child** (*PyCapture2.topologyNode*) – The child node to add.

addPortType (*type*) → None

Add the specified port type as a port of the node.

Parameters **type** (*int*) – The type of port to add. Use `PyCapture2.PORT_TYPE` to read correctly.

assignGuid (*guid, deviceID, nodeType = None*) → None

Assign a guid, device ID, and optionally a node type to the node.

Parameters

- **guid** (*int, int, int, int*) – The GUID to be assigned.
- **deviceID** (*int*) – The device ID to be assigned.
- **nodeType** (*int*) – The optional node type to be assigned. Use `PyCapture2.NODE_TYPE` to set correctly.

getChild (*index*) → *childNode*

Get child node located at the specified position.

Parameters **index** (*int*) – The position of the node.

Returns The topologyNode of the child.

Return type *childNode* (*PyCapture2.TopologyNode*)

getDeviceID () → *deviceID*

Get the device ID associated with the node.

Returns The ID of the object represented by the node.

Return type *deviceID* (*int*)

getGuid () → *guid*

Get the guid associated with the node.

Returns The guid of the object represented by the node.

Return type *guid* (*int, int, int, int*)

getInterfaceType () → *interfaceType*

Get the interface type associated with the node.

Returns The interface type of the node. Use PyCapture2.INTERFACE_TYPE to read correctly.

Return type *interfaceType* (int)

getNodeType () → nodeType

Get the type of the node.

Returns The type of the node. Use PyCapture2.NODE_TYPE to read correctly.

Return type nodeType (int)

getNumChildren () → children

Get the number of children associated with the node.

Returns The number of child nodes.

Return type children (int)

getNumPorts () → numPorts

Get the number of ports of the node.

Returns The number of ports.

Return type numPorts (int)

getPortType (*position*) → portType

Get the type of port located at the specified position.

Parameters **position** (*int*) – The position of the port.

Returns The type of port. Use PyCapture2.PORT_TYPE to read correctly.

Return type portType (int)

UTILITY FUNCTIONS

PyCapture2 Utility functions. The utility class is generally used to query for general system information such as operating system, available memory, etc.

10.1 PyCapture2.getLibraryVersion

`PyCapture2.getLibraryVersion()`
`getLibraryVersion()` -> version

Get the FlyCapture 2 library version.

Returns The library version. It's format is (major, minor, type, build)

Return type version (int, int, int, int)

10.2 PyCapture2.startSyncCapture

`PyCapture2.startSyncCapture(cameras)`
`startSyncCapture(cameras)` -> None

Start synchronized isochronous image capture on multiple cameras.

Parameters `cameras` ([`PyCapture2.Camera`, `PyCapture2.Camera`, ...]) – A list of cameras to start isochronous capture.

10.3 PyCapture2.getRegisterString

`PyCapture2.getRegisterString(unsigned int registerVal)`
`getRegisterString(registerValue)` -> registerString

Return a text representation of the register value.

Parameters `registerValue` (*int*) – The register value to query.

Returns The textual representation of the register value.

Return type registerString (str)

10.4 PyCapture2.setDefaultColorProcessing

`PyCapture2.setDefaultColorProcessing(defaultMethod)`
`setDefaultColorProcessing(defaultMethod) -> None`

Set the default color processing algorithm.

This method will be used for any image with the DEFAULT algorithm set. The method used is determined at the time of the Convert() call, therefore the most recent execution of this function will take precedence. The default setting is shared within the current process.

Parameters `defaultMethod` (*int*) – The method to set as default. Use PyCapture2.COLOR_PROCESSING to set correctly.

10.5 PyCapture2.getDefaultColorProcessing

`PyCapture2.getDefaultColorProcessing()`
`getDefaultColorProcessing() -> defaultMethod`

Get the default color processing algorithm.

Returns The default method. Use PyCapture2.COLOR_PROCESSING to read correctly.

Return type `defaultMethod` (*int*)

10.6 PyCapture2.getDefaultOutputFormat

`PyCapture2.getDefaultOutputFormat()`
`getDefaultOutputFormat() -> format`

Get the default output format.

Returns The default pixel format. Use PyCapture2.PIXEL_FORMAT to read correctly.

Return type `format` (*int*)

10.7 PyCapture2.setDefaultOutputFormat

`PyCapture2.setDefaultOutputFormat(format)`
`setDefaultOutputFormat(format) -> None`

Set the default output pixel format.

This format will be used for any call to Convert() that does not specify an output format. The format used will be determined at the time of the Convert() call, therefore the most recent execution of this function will take precedence. The default is shared within the current process.

Parameters `format` (*int*) – The pixel format to set as default. Use PyCapture2.PIXEL_FORMAT to set correctly.

10.8 PyCapture2.determineBitsPerPixel

PyCapture2.**determineBitsPerPixel** (*format*)

determineBitsPerPixel(format) -> bitsPerPixel

Calculate the bits per pixel for the specified pixel format.

Parameters **format** (*int*) – The pixel format. Use PyCapture2.PIXEL_FORMAT to set correctly.

Returns The bits per pixel.

Return type bitsPerPixel (int)

10.9 PyCapture2.checkDriver

PyCapture2.**checkDriver** (*tuple id*)

checkDriver(guid) -> None

Check for driver compatibility for the given camera guid.

Parameters **guid** (*int, int, int, int*) – The guid of the device to check.

10.10 PyCapture2.getDriverDeviceName

PyCapture2.**getDriverDeviceName** (*tuple id*)

getDriverDeviceName(guid) -> name

Get the driver's name for a device.

Parameters **guid** (*int, int, int, int*) – The guid of the device to check.

Returns The name of the device.

Return type name (str)

10.11 PyCapture2.getSystemInfo

PyCapture2.**getSystemInfo** ()

getSystemInfo() -> systemInfo

Get system information.

Returns A SystemInfo object containing the system information.

Return type systemInfo (*PyCapture2.SystemInfo*)

10.12 PyCapture2.launchBrowser

PyCapture2.**launchBrowser** (*const char* url*)

launchBrowser(url) -> None

Launch a URL in the system's default browser.

Parameters **url** (*str*) – The URL to open.

10.13 PyCapture2.launchHelp

PyCapture2.**launchHelp** (*const char* fileName*)
launchHelp(fileName) -> None

Open a CHM file in the system default CHM viewer.

Parameters **fileName** (*str*) – Filename of the CHM file to open.

10.14 PyCapture2.launchCommand

PyCapture2.**launchCommand** (*const char* command*)
launchCommand(command) -> None

Execute a command in the terminal.

This is a blocking call that will return when the command completes.

Parameters **command** (*str*) – The command to execute.

10.15 PyCapture2.launchCommandAsync

PyCapture2.**launchCommandAsync** (*const char* command, func, *funcArgs*)
launchCommandAsync(command, function, **arguments*) -> None

Execute a command in the terminal.

This is a non-blocking call that will return immediately. The return value of the command can be retrieved in the callback.

Parameters

- **command** (*str*) – The command to execute.
- **function** (*function*) – The function to call when the command is complete.
- **arguments** – Any additional arguments will be passed to the callback function when it is called.

PYCAPTURE2 MODULE

A wrapper for FLIR Integrated Imaging Solutions' FlyCapture 2 library.

This module wraps the FlyCapture 2 C library. It is available on our website <https://www.ptgrey.com/support/downloads>, but is only necessary if this is built from source (not a wheel).

class `PyCapture2.AVIREcorder`

Bases: `object`

A class from FlyCapture 2 used to create animated AVI and MP4 files.

AVIOpen (*filename*, *framerate*) → `None`

Open an AVI file for writing images to disk.

The size of AVI files is limited to 2GB. The filenames are automatically generated using the filename specified.

Parameters

- **filename** (*str*) – The name of the file to save to.
- **framerate** (*float*) – The framerate of the AVI.

H264Open (*filename*, *framerate*, *width*, *height*, *bitrate*)

Open an H264 MP4 file for writing images to disk.

The size of MP4 files is limited to 2GB. The filenames are automatically generated using the filename specified.

Parameters

- **filename** (*str*) – The name of the file to save to.
- **framerate** (*float*) – The framerate of the MP4.
- **width** (*int*) – The width of the source image.
- **height** (*int*) – The height of the source image.
- **bitrate** (*int*) – The bitrate to encode at.

MJPEGOpen (*filename*, *framerate*, *quality*) → `None`

Open an MJPG AVI file for writing images to disk.

The size of AVI files is limited to 2GB. The filenames are automatically generated using the filename specified.

Parameters

- **filename** (*str*) – The name of the file to save to.
- **framerate** (*float*) – The framerate of the AVI.

- **quality** (*int*) – Image quality on range (1 - 100)

append (*image*) → None

Append an image to the AVI/MP4 file.

Parameters **image** (`PyCapture2.Image`) – Image object to append to file.

close () → None

Close the AVI/MP4 file and save it to disk.

class `PyCapture2.AvailableImageInfo`

Structure containing the availability of image properties, for use in `EmbeddedImageInfo` class.

timestamp

bool – Whether timestamp is supported by the camera.

gain

bool – Whether gain is supported by the camera.

shutter

bool – Whether setting the shutter is supported by the camera.

brightness

bool – Whether modifying the brightness is supported by the camera.

exposure

bool – Whether modifying the exposure is supported by the camera.

whiteBalance

bool – Whether modifying the white balance of the image is supported by the camera.

frameCounter

bool – Whether the camera supports frame counting functionality.

strobePattern

bool – Whether the camera supports strobe functionality.

GPIOPinState

bool – Whether setting the GPIO pin state is supported by the camera.

ROIPosition

bool – Whether specifying the region of interest is specified by the camera.

class `PyCapture2.BANDWIDTH_ALLOCATION`

Bases: `object`

Bandwidth allocation options for 1394 devices.

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

Not specified. This leaves the current setting unchanged.

OFF = 0

ON = 1

UNSPECIFIED = 3

UNSUPPORTED = 2

class PyCapture2.**BAYER_FORMAT**

Bases: object

Bayer tile formats.

NONE

No bayer tile format.

RGGGB

Red-Green-Green-Blue.

GRBG

Green-Red-Blue-Green.

GBRG

Green-Blue-Red-Green.

BGGR; Blue-Green-Green-Red.

BGGR = 4

GBRG = 3

GRBG = 2

NONE = 0

RGGGB = 1

class PyCapture2.**BMPOption**

Structure containing options for saving Bitmap images.

indexedColor_8bit

bool – Whether to save with 8bit indexed color.

class PyCapture2.**BUS_CALLBACK_TYPE**

Bases: object

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

BUS_RESET

Register for all bus events.

ARRIVAL

Register for arrivals only.

REMOVAL

Register for removals only.

ARRIVAL = 1

BUS_RESET = 0

REMOVAL = 2

class PyCapture2.**BUS_SPEED**

Bases: object

Bus speeds.

S100

100Mbps/sec.

S200

200Mbps/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.

BASE_T_10
10Base-T. Only for GigE cameras.

BASE_T_100
100Base-T. Only for GigE cameras.

BASE_T_1000
1000Base-T (Gigabit Ethernet). Only for GigE cameras.

BASE_T_10000
10000Base-T. Only for GigE cameras.

S_FASTEST
The fastest speed available.

ANY
Any speed that is available.

SPEED_UNKNOWN
Unknown bus speed.

ANY = 13

BASE_T_10 = 8

BASE_T_100 = 9

BASE_T_1000 = 10

BASE_T_10000 = 11

S100 = 0

S1600 = 5

S200 = 1

S3200 = 6

S400 = 2

S480 = 3

S5000 = 7

S800 = 4

SPEED_UNKNOWN = -1

S_FASTEST = 12

class `PyCapture2.BYTE_ORDER`

Bases: `object`

Possible byte order.

LITTLE_ENDIAN

BIG_ENDIAN

BIG_ENDIAN = 1

LITTLE_ENDIAN = 0

class `PyCapture2.BaseCamera`

Bases: `object`

Base camera class. This class exists only for camera and GigECamera classes to inherit from it.

connect (*guid*) → `None`

Connect the camera object to a physical camera.

Parameters **guid** (*int, int, int, int*) – The GUID of the camera to connect to.

The BusManager class has several functions that return a valid GUID.

deregisterAllEvents () → `None`

De-register all registered events.

deregisterEvent (*eventName*) → `None`

De-register an event previously registered with the camera.

Parameters **eventName** (*str*) – The name of the event to deregister.

This function currently only works with EventExposureEnd events.

disconnect () → `None`

Disconnect the camera object from a physical camera.

enableLUT (*enable*) → `None`

Enable or disable LUT functionality, based on the passed bool.

Parameters **enable** (*bool*) – Whether to activate or deactivate the LUT.

fireSoftwareTrigger () → `None`

Fire the software trigger, according to the DCAM specifications.

getActiveLUTBank () → `activeBank`

Return the LUT bank that is currently being used.

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

Returns The active LUT bank.

Return type `activeBank` (*int*)

getCameraInfo () → `cameraInfo`

Return detailed information about the camera.

Returns The camera information.

Return type `cameraInfo` (*PyCapture2.CameraInfo*)

getConfiguration () → `config`

Return configuration properties of the camera.

Returns The configuration information.

Return type `config` (*PyCapture2.Config*)

getCycleTime () → `timeStamp`

Return a timestamp object containing 1394 CYCLE_TIME information.

Returns The TimeStamp object with time data.

Return type *timeStamp* (*PyCapture2.TimeStamp*)

getEmbeddedImageInfo ()

`camera.getEmbeddedImageInfo` -> `embeddedImageInfo`

Return the current status of each embedded image property, as well as their availability.

Returns a `PyCapture2.EmbeddedImageInfo` object containing the properties.

Return type `embeddedImageInfo` (*PyCapture2.EmbeddedImageInfo*)

getGPIOPinDirection (*pin*) → `direction`

Get GPIO pin direction for the specified pin.

Both the pin and the direction are represented by an unsigned int.

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

Parameters **pin** (*int*) – the pin to query.

Returns the direction of the pin.

Return type `direction` (*int*)

getLUTBankInfo (*bank*) → `readSupport`, `writeSupport`

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

Parameters **is an int representing the bank to query.** (*bank*) –

Returns indicates whether the bank can be read. `writeSupport` (*bool*): indicates whether the bank can be written to.

Return type `readSupport` (*bool*)

getLUTChannel (*bank*, *channel*, *numEntries*) → `entries`

Get the LUT channel settings from the camera.

Entries contains the LUT entries. It is a list of `sizeEntries` ints.

Parameters

- **bank** (*int*) – specifies which bank to retrieve.
- **channel** (*int*) – specifies which channel to retrieve.
- **sizeEntries** (*int*) – specifies the number of entries to read.

Returns The returned entries.

Return type `entries` (*[int, int, ...]*)

getLUTInfo () → `LUTData`

Query if LUT support is available on the camera.

Returns A `LUTData` object containing the queried data.

Return type *LUTData* (*PyCapture2.LUTData*)

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`.

getMemoryChannel () → currentChannel

Return the current memory channel from the camera.

Returns The current memory channel.

Return type currentChannel (int)

getMemoryChannelInfo () → numChannels

Query the camera for memory channel support.

If numChannels is 0, then memory channel support is not available.

Returns The number of memory channels available.

Return type numChannels (int)

getProperty (propertyType) → property

Return the settings for the specified property from the camera.

Parameters propertyType (int) – The type of property to return. use PyCapture2.PROPERTY_TYPE to set correctly.

Returns a Property object with the specified property information.

Return type property (PyCapture2.Property)

getPropertyInfo (propertyType) → propertyInfo

Retrieves information about the specified camera property.

Parameters propertyType (int) – The type of property to return. Use PyCapture2.PROPERTY_TYPE to set correctly.

Returns a PropertyInfo object with the specified property information.

Return type propertyInfo (PyCapture2.PropertyInfo)

getStats () → cameraStats

Return camera diagnostic information.

Returns A CameraStats object with diagnostic information.

Return type cameraStats (PyCapture2.CameraStats)

getStrobe () → strobeControl

Return current strobe settings of the camera.

Returns a StrobeControl object.

Return type strobeControl (PyCapture2.StrobeControl)

getStrobeInfo () → strobeInfo

Return strobe information from the camera.

Returns a StrobeInfo object containing the strobe information.

Return type strobeInfo (PyCapture2.StrobeInfo)

getTriggerDelay () → property

Return the current trigger delay settings of the camera.

Returns A Property object containing the trigger delay settings.

Return type property (PyCapture2.Property)

getTriggerDelayInfo () → PropertyInfo

Return trigger delay information from the camera.

Returns A PropertyInfo object containing the trigger delay information.

Return type `propertyInfo` (*PyCapture2.PropertyInfo*)

getTriggerMode () → `triggerMode`

Retrieve current trigger settings from the camera.

Returns (*PyCapture2.TriggerMode*): A `TriggerMode` object containing the camera's image settings.

Return type `triggerMode`

isConnected

readRegister (*address*) → `value`

Read the specified register from the camera.

Parameters **address** (*int*) – The address of the register to read.

Returns The value read from the register.

Return type *value* (*int*)

readRegisterBlock (*address, length*) → `values`

Read from the specified register block on the camera.

Parameters

- **address** (*int*) – The 48bit address to write to.
- **length** (*int*) – Amount of data to read.

registerAllEvents (*function, arguments*) → `None`

Register the camera to issue a custom callback on all available events.

The function must take a `CallbackData` object as its first argument.

Parameters

- **function** (*function*) – The function to call on event completion. It must take a `CallbackData` object as its first argument.
- ***arguments** – The rest of the arguments are given to the callback function when called.

registerEvent (*eventName, function, *arguments*) → `None`

Register the camera to issue a custom callback function for a specific device event.

The function must take a `CallbackData` object as its first argument.

Parameters

- **eventName** (*str*) – The type of event to register a callback for.
- **function** (*function*) – The function to call on event completion. It must take a `CallbackData` object as its first argument.
- ***arguments** – The rest of the arguments are given to the callback function when called.

restoreFromMemoryChannel (*channel*) → `None`

Restore the specified memory channel.

Parameters **channel** (*int*) – memory channel to restore from

retrieveBuffer () → `image`

Retrieve an image from the buffer and return it.

The method to grab with (the newest image or the oldest) is specified via the `grabMode` attribute of the `Config` class.

Returns The image object grabbed from the camera.

Return type image (*PyCapture2.Image*)

saveToMemoryChannel (*channel*) → None

Save the current settings to the specified memory channel.

Parameters **channel** (*int*) – The memory channel to save the settings to.

setActiveLUTBank (*activeBank*) → None

Take an integer and set that LUT bank as the active one.

Parameters **activeBank** (*int*) – an integer representing the LUT bank to set.

setCallback (*function*, **args*) → None

Sets a callback function to be called on the completion of image transfer.

The function **MUST** take an image object as its first argument, and any extra arguments given afterwards.

To clear the stored callback data, call unsetCallback().

Parameters

- **function** (*function*) – The function that is called in a callback. This function must take an image object as its first argument.
- **arguments** – The rest of the arguments are the arguments for the callback function.

setConfiguration (*config = None*, ***kwargs*) → None

Take config properties and update camera configuration.

There are two ways to call this function: The first is with a Config object that has all the values to be updated.

The second is specifying each property to update in the arguments. The arguments that can be specified are:

numBuffers (int)
numImageNotifications (int)
minNumImageNotifications (int)
grabTimeout (int)
grabMode (int)
isochBusSpeed (int)
asyncBusSpeed (int)
bandwidthAllocation (int)
registerTimeoutRetries (int)
registerTimeout (int)

Information about these properties can be found in the Config documentation.

ex. camera.setConfiguration(isochBusSpeed = PyCapture2.BUS_SPEED.S400) will update the camera's isochronous bus speed.

Parameters

- **config** (*PyCapture2.Config*) – The configuration object to get all properties from.
- **kwargs** (*{...}*) – The second method of calling: specify the properties to change as keywords.

setEmbeddedImageInfo (*embeddedImageInfo = None*, ***kwargs*) → None

Take embeddedImageInfo properties and update the camera's register to match.

There are two ways to call this function: The first is with an `EmbeddedImageInfo` object that has all the values to be updated.

The second is specifying each property to update in the arguments. The arguments that can be specified are:

timestamp (bool)
gain (bool)
shutter (bool)
brightness (bool)
exposure (bool)
whiteBalance (bool)
frameCounter (bool)
strobePattern (bool)
ROIPosition (bool)

Information about these properties can be found in the `EmbeddedImageInfo` documentation.

ex. `camera.setEmbeddedImageInfo(timestamp = True)` will update the camera's timestamp setting.

Parameters

- **embeddedImageInfo** (`PyCapture2.EmbeddedImageInfo`) – The embedded image info object to get all properties from.
- **kwargs** (`{...}`) – The second method of calling: specify the properties to change as keywords.

setGPIOPinDirection (*pin, direction*) → None

Set 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** (*int*) – The pin number to set.
- **direction** (*int*) – The direction to set.

setLUTChannel (*bank, channel, data*) → None

Set the LUT channel settings to the camera.

Parameters

- **bank** (*int*) – the bank to set
- **channel** (*int*) – the channel to set
- **data** (*[int, int, ...]*) – a list containing the integers to write to the LUT channel.

setProperty (*property = None, **kwargs*) → None

Take strobe properties and update the camera's strobe.

There are two ways to call this function: The first is with a `Property` object containing the updated properties.

The second is by specifying each property to update in the arguments. The arguments that can be specified are:

type (int)

present (bool)
absControl (bool)
onePush (bool)
onOff (bool)
autoManualMode (bool)
valueA (int)
valueB (int)
absValue (float)

Note that type MUST be specified if this method of calling is used.

Information about these properties can be found in the Property documentation.

Ex. camera.setProperty(type = PyCapture2.PROPERTY_TYPE.ZOOM, absValue = 2.0) Sets zoom to 2.

Parameters

- **property** (`PyCapture2.Property`) – The property object to get all properties from.
- **kwargs** (`{...}`) – The second method of calling: specify the properties to change as keywords.

setStrobe (*strobeControl = None, **kwargs*) → None

Take strobe properties and update the camera's strobe.

There are two ways to call this function: The first is with a StrobeControl object containing the updated properties.

The second is by specifying each property to update in the arguments. The arguments that can be specified are:

source (int)
onOff (bool)
polarity (int)
delay (float)
duration (float)

Information about these properties can be found in the StrobeControl documentation.

Ex. camera.setStrobe(onOff = True) Activates the strobe.

Parameters

- **strobeControl** (`PyCapture2.StrobeControl`) – The strobe control object to get all properties from.
- **kwargs** (`{...}`) – The second method of calling: specify the properties to change as keywords.

setTriggerDelay (*property = None, **kwargs*) → None

Take trigger delay properties and update the camera.

There are two ways to call this function: The first is with a Property object that has all the triggerDelay values to be updated.

The second is by specifying each property to update the arguments. The arguments that can be specified are:

propType (int)
present (bool)
absControl (bool)
onePush (bool)
onOff (bool)
autoManualMode (bool)
valueA (int)
valueB (int)
absValue (float)

Information about these properties can be found in the Property documentation.

Ex. camera.setTriggerDelay(onOff = True) Will activate the camera trigger delay.

Parameters

- **property** (`PyCapture2.Property`) – The Property object to get all properties from.
- **kwargs** (`{...}`) – The second method of calling: specify the properties to change as keywords.

setTriggerMode (*TriggerMode = None, **kwargs*) → None

Take triggerMode properties and update the camera's trigger.

There are two ways to call this function: The first is with a TriggerMode object that has all the values to be updated.

The second is specifying each property to update in the arguments. The arguments that can be specified are:

onOff (bool)
polarity (int)
source (int)
mode (int)
parameter (int)

Information about these properties can be found in the TriggerMode documentation.

Ex. camera.setTriggerMode(onOff = True) will allow the camera to be triggered.

Parameters

- **triggerMode** (`PyCapture2.TriggerMode`) – The trigger mode object to get all properties from.
- **kwargs** (`{...}`) – The second method of calling: specify the properties to change as keywords.

startCapture (*function = None, *arguments*) → None

Start capturing data from camera to image buffers, and optionally set a callback to be run when an image is retrieved.

To start capture and retrieve images manually, call without any arguments.

To set a callback, call startCapture with a function and the function's optional arguments.

The function **MUST** take an image object as its first argument, and any extra arguments given afterwards.

To clear the stored callback data, call unsetCallback().

Parameters

- **function** (*function*) – The optional function that is called in a callback. This function must take an image object as its first argument.
- **arguments** – The rest of the arguments are the arguments for the callback function.

stopCapture () → None

Stop capturing data from camera.

unsetCallback () → None

Clears stored callback data.

If there is currently no stored callback data, this function does nothing.

waitForBufferEvent (*eventNumber*) → image

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

Parameters **eventNumber** (*int*) – The event number to wait for.

Returns Image object with the latest image buffer.

Return type image (*PyCapture2.Image*)

writeRegister (*address, value*) → None

Write the specified value to the specified register on the camera.

Parameters

- **address** (*int*) – The address of the register to write.
- **value** (*int*) – The value to write to the register.

writeRegisterBlock (*address, values*) → None

Write to the specified register block on the camera.

Parameters

- **address** (*int*) – The 48bit address to write to.
- **values** (*[int, int, ...]*) – List containing data to be written.

class `PyCapture2.BusManager`

Bases: `object`

A FlyCapture 2 Bus Manager Class.

This class can be used to find GUIDs of cameras easily.

discoverGigECameras (*numCams = 10*) → `cameraInfos`

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

This is useful in situations where GigE Vision cameras are using IP addresses in a subnet different from the host's subnet. After discovering the camera, it is easy to use `ForceIPAddressToCamera()` to set a different IP configuration.

Parameters **numCams** (*int*) – The maximum number of cameras to read. Default is 10.

Returns A list containing `cameraInfo` objects.

Return type `cameraInfos` (*[PyCapture2.CameraInfo, PyCapture2.CameraInfo, ...]*)

fireBusReset (*guid*) → None

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(int, int, int, int)` – the guid to fire a reset for.

forceAllIPAddressesAutomatically() → None

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

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

forceIPAddressToCamera(macAddress, ipAddress, subnetMask, gateway) → None

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

Parameters

- **macAddress** (*int, int, int, int, int, int*) – The MAC address of the camera to force IP to.
- **ipAddress** (*int, int, int, int*) – The IP address to force to the camera.
- **subnetMask** (*int, int, int, int*) – The subnet mask to force to the camera.
- **gateway** (*int, int, int, int*) – The default gateway to force to the camera.

getCameraFromIPAddress(ip) → guid

Return the guid of a camera, determined by its IP address.

Parameters `ip(int, int, int, int)` – The ip to get the GUID from.

Returns The guid of the specified camera.

Return type guid (int, int, int, int)

getCameraFromIndex(index) → guid

Return the guid of a camera, specified by its index.

Parameters `index(int)` – The index of the camera to get the guid for.

Returns The guid of the specified camera.

Return type guid (int, int, int, int)

getCameraFromSerialNumber()

BusManager.getCameraFromIndex(serial number) -> guid

Return the guid of a camera, specified by its serial number.

Parameters `serialNumber(int)` – The serial number to get the camera from.

Returns The guid of the specified camera.

Return type guid (int, int, int, int)

getCameraSerialNumberFromIndex(index) → serialNumber

Return the serial number of a camera, specified by its index.

Parameters `index(int)` – The index of the camera to get the guid for.

Returns The serial number of the specified camera.

Return type *serialNumber* (int)

getDeviceFromIndex(index) → guid

Return the guid of a device, specified by its index.

Parameters `index(int)` – The index of the device to get the guid for.

Returns The guid of the specified device.

Return type `guid (int, int, int, int)`

getInterfaceTypeFromGuid (*guid*) → `interfaceType`

Return interfaceType of the camera associated with the guid.

Parameters **guid** (*int, int, int, int*) – The guid to get the interface type from.

Returns The interface type of the specified camera. Use `PyCapture2.INTERFACE_TYPE` to read correctly.

Return type `interfaceType (int)`

getNumOfCameras () → `count`

Return the number of cameras attached to the PC.

Returns The number of cameras attached.

Return type `count (int)`

getNumOfDevices () → `numDevices`

Return the number of devices attached to the PC.

Returns The number of attached devices.

Return type `numDevices (int)`

getTopology () → `topologyNode`

Get the topology information for the PC.

Returns `PyCapture2.TopologyNode` object that contains the topology information.

Return type `topologyNode (PyCapture2.topologyNode)`

getUsbLinkInfo (*guid*) → `value`

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

Parameters **guid** (*int, int, int, int*) – The guid device to read from.

Returns The value read from the card register.

Return type `value (int)`

getUsbPortStatus (*guid*) → `value`

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

Parameters **guid** (*int, int, int, int*) – The guid device to read from.

Returns The value read from the card register.

Return type `value (int)`

isCameraControllable (*guid*) → `isControllable`

Query CCP status on camera with corresponding guid.

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

Parameters **guid** (*int, int, int, int*) – The guid device to query.

Returns Whether the camera is controllable.

Return type `isControllable (bool)`

readPhyRegister (*guid, page, port, address*) → `value`

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** (*int*, *int*, *int*, *int*) – The guid device to read from.
- **page** (*int*) – The page to read from.
- **port** (*int*) – The port to read from.
- **address** (*int*) – The address to read from.

Returns The value of the phy register.

Return type *value* (*int*)

registerCallback (*callbackType*, *function*, **arguments*) → None

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

Parameters

- **callbackType** (*int*) – The type of callback to register an event to. Use PyCapture2.BUS_CALLBACK_TYPE to set correctly.
- **function** (*function*) – The function to call when the specified event occurs.
- **arguments** – Any additional arguments will be passed to the function when it is called.

rescanBus () → None

Force a rescan of the buses.

This does not trigger a bus reset. However, any current connections to a Camera object will be invalidated.

unregisterCallback (*callbackType*) → None

Unregister a callback function.

Parameters **callbackType** (*int*) – The type of callback event to deregister. Use PyCapture2.BUS_CALLBACK_TYPE to set correctly.

writePhyRegister (*guid*, *page*, *port*, *address*, *value*) → None

Write to a phy register on the specified device.

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

Parameters

- **guid** (*int*, *int*, *int*, *int*) – The guid device to write to.
- **page** (*int*) – The page to write to.
- **port** (*int*) – The port to write to.
- **address** (*int*) – The address to write to.
- **value** (*int*) – The value to write.

class PyCapture2.COLOR_PROCESSING

Bases: object

Color processing algorithms.

Please refer to our knowledge base at article at <http://www.ptgrey.com/KB/10141> for complete details for each algorithm.

DEFAULT

Default method.

NO_COLOR_PROCESSING

No color processing.

NEAREST_NEIGHBOR_FAST

Fastest but lowest quality. Equivalent to FLYCAPTURE_NEAREST_NEIGHBOR_FAST in FlyCapture.

EDGE_SENSING

Weights surrounding pixels based on localized edge orientation.

HQ_LINEAR

Well-balanced speed and quality.

RIGOROUS

Slowest but produces good results.

IPP

Multithreaded with similar results to edge sensing.

DIRECTIONAL

Best quality but much faster than rigorous

DEFAULT = 0

DIRECTIONAL = 7

EDGE_SENSING = 3

HQ_LINEAR = 4

IPP = 6

NEAREST_NEIGHBOR_FAST = 2

NO_COLOR_PROCESSING = 1

RIGOROUS = 5

class `PyCapture2.CallbackData`

Data returned from an event callback.

This must be the first argument for any event callback functions.

eventName

str – The event name used to register the event.

eventID

long – The device register that ‘eventName’ maps to.

eventTimestamp

long – Time as reported by the camera at which the exposure operation completed. Please note this is NOT a `PyCapture2.TimeStamp` object!

class `PyCapture2.Camera`

Bases: `PyCapture2.BaseCamera`

A camera object from FlyCapture2. This is specifically for USB cameras.

getFormat7Configuration () → imageSettings, packetSize, percentage

Get the current Format7 configuration from the camera.

This call will raise an `Fc2error` if the camera is not already in Format7.

Returns

tuple containing: imageSettings (`PyCapture2.Format7ImageSettings`): Current image settings. packetSize (`int`): Current packet size. percentage (`float`): current packet size as a percentage of maximum packet size.

Return type (tuple)

getFormat7Info (*mode*) → format7Info, supported

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

Parameters *mode* (*int*) – Format7 mode to query. Use PyCapture2.MODE to set correctly.

Returns

tuple containing: format7Info (PyCapture2.Format7Info): A Format7Info object with capabilities and the current state of the specified mode. supported (bool): Whether the specified mode is supported.

Return type (tuple)

getVideoModeAndFrameRate () → videoMode, frameRate

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

If the camera is in Format7, the video mode will be VIDEO_MODE.FORMAT7 and the framerate will be FRAME_RATE.FORMAT7.

Returns

tuple containing: videoMode (int): Current video mode of the camera. Use VIDEO_MODE to check its value. frameRate (int): Current frame rate of the camera. Use FRAME_RATE to check its value.

Return type (tuple)

getVideoModeAndFrameRateInfo (*videoMode*, *frameRate*) → supported

Query the camera to determine if the specified video mode and framerate are supported.

Parameters

- **videoMode** (*int*) – Video mode to check. Use PyCapture2.VIDEO_MODE to set correctly.
- **frameRate** (*int*) – Frame rate to check. Use PyCapture2.FRAME_RATE to set correctly.

Returns Whether or not the specified video mode and frame rate are supported.

Return type *supported* (bool)

setFormat7Configuration (*percentSpeed*, *imageSettings* = None, ***kwargs*) → None

Set the given Format7 configuration to the camera.

There are two ways to call this function: The first is with a Format7ImageSettings object that has all the values to be updated.

The second is by specifying each property to update the arguments. The arguments that can be specified are:

mode (int)
offsetX (int)
offsetY (int)
width (int)
height (int)
pixelFormat (int)

Information about these properties can be found in the Format7ImageSettings documentation.

Ex. camera.setFormat7Settings(50.0, pixelFormat = PyCapture2.PIXEL_FORMAT.MONO8)
Will set the current mode to 8bit greyscale image capture, with 50% of max packet size.

Other Arguments: percentSpeed (float): packet size, in percentage of max packet size.

imageSettings (PyCapture2.Format7ImageSettings): The image settings object to get all properties from. kwargs ({ ... }): The second method of calling: specify the properties to change as keywords.

setFormat7ConfigurationPacket ()

camera.setFormat7Configuration(packetSize, imageSettings = None, ****kwargs**) -> None

Set the given Format7 configuration to the camera.

There are two ways to call this function: The first is with a Format7ImageSettings object that has all the values to be updated.

The second is by specifying each property to update the arguments. The arguments that can be specified are:

mode (int)
offsetX (int)
offsetY (int)
width (int)
height (int)
pixelFormat (int)

Information about these properties can be found in the Format7ImageSettings documentation.

Ex. camera.setFormat7Settings(50.0, pixelFormat = PyCapture2.PIXEL_FORMAT.MONO8)

Will set the current mode to 8bit greyscale image capture, with 50% of max packet size.

Other Arguments: packetSize (int): packet size, in bytes.

imageSettings (PyCapture2.Format7ImageSettings): The image settings object to get all properties from. kwargs ({ ... }): The second method of calling: specify the properties to change as keywords.

setVideoModeAndFrameRate (videoMode, frameRate) → None

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

It is not possible to set the camera to VIDEO_MODE.FORMAT7 or FRAME_RATE.FORMAT7. Use the Format7 functions to use Format7 functionality.

Parameters

- **videoMode** (*int*) – Video mode to set. Use PyCapture2.VIDEO_MODE to set correctly.
- **frameRate** (*int*) – Frame rate to set. Use PyCapture2.FRAME_RATE to set correctly.

validateFormat7Settings (imageSettings = None, ****kwargs**) → packetInfo, isValid

Check whether given Format7ImageSettings properties are valid, and return packet information and whether the settings are valid.

There are two ways to call this function: The first is with a Format7ImageSettings object that has all the values to be updated.

The second is by specifying each property to update the arguments. The arguments that can be specified are:

mode (int)
offsetX (int)
offsetY (int)
width (int)
height (int)
pixelFormat (int)

Information about these properties can be found in the `Format7ImageSettings` documentation.

Ex. `camera.validateFormat7Settings(pixelFormat = PyCapture2.PIXEL_FORMAT.MONO8)`
Will check if the current mode supports 8bit greyscale image capture.

Parameters

- **imageSettings** (`PyCapture2.Format7ImageSettings`) – The property object to get all properties from.
- **kwargs** (`{...}`) – The second method of calling: specify the properties to change as keywords.

Returns

tuple containing: `packetInfo` (`PyCapture2.Format7PacketInfo`): Packet size information used to determine a valid packet size. `isValid` (`bool`): Whether the settings are valid.

Return type (`tuple`)

class `PyCapture2.CameraInfo`

camera information class.

serialNumber

int – The serial number of the device.

interfaceType

int – The device's interface type. Use `PyCapture2.INTERFACE_TYPE` to set correctly.

driverType

int – The device's driver type. Use `PyCapture2.DRIVER_TYPE` to set correctly.

isColorCamera

bool – Indicate whether the Camera can capture color.

modelName

str – The name of the device's model name.

vendorName

str – The name of the device's vendor.

sensorInfo

str – Details about the sensor.

sensorResolution

str – String providing the sensor resolution.

driverName

str – Name of the driver being used.

firmwareVersion

str – The firmware version of the camera.

firmwareBuildTime

str – The time the firmware was built.

maximumBusSpeed

int – Maximum bus speed. Use `PyCapture2.BUS_SPEED` to set correctly.

bayerTileFormat

int – The bayer tile format. Use `PyCapture2.BAYER_FORMAT` to set correctly.

pcieBusSpeed

int – The bus number. Use PyCapture2.PCIE_BUS_SPEED to set correctly. Set to 0 for GigE and USB cameras.

nodeNumber

int – ieee1394 Node number. Set to 0 for GigE and USB cameras.

busNumber

int – PCIe Bus Speed. Use PyCapture2.PCIE_BUS_SPEED to set correctly. Set to “UNKNOWN” for unsupported drivers.

class PyCapture2.CameraStats

Diagnostic information about the camera.

imageDropped

int – Number of images dropped.

imageCorrupt

int – Number of corrupted images.

imageXmitFailed

int – Number of times XMIT failed.

imageDriverDropped

int – Number of times the driver dropped an image.

regReadFailed

int – Number of times a register read failed.

regWriteFailed

int – Number of times a register write failed.

portErrors

int – Number of port errors.

cameraPowerUp

bool – Whether the camera is currently on.

cameraVoltages

[float, float, float, ...] – The values of the camera’s voltage registers. Maximum length is 8.

cameraCurrents

[float, float, float, ...] – The values of the camera’s current registers. Maximum length is 8.

temperature

int – The temperature of the camera.

timeSinceInitialization

int – The time since the camera was initialized.

timeSinceBusReset

int – The time since the last bus reset event.

timeStamp

PyCapture2.TimeStamp – The timestamp of when this data was gathered.

numResendPacketsRequested

int – The number of packets requested to be resent.

numResendPacketsReceived

int – The number of resent packets recieved.

class PyCapture2.Config

Structure containing the configuration for a camera.

numBuffers

int – The number of buffers used by the FlyCapture2 library to grab images.

numImageNotifications

int – The number of notifications per image.

minNumImageNotifications

int – The minimum number of notifications needed for the current image settings on the camera.

grabTimeout

int – Time (in milliseconds) that camera.retrieveBuffer() and camera.waitForBufferEvent() will wait for an image before timing out and returning.

grabMode

int – Grab mode for the camera. Use PyCapture2.GRAB_MODE to set correctly.

highPerformanceRetrieveBuffer

bool – This attribute enables retrieveBuffer to run in high performance mode.

isochBusSpeed

int – Isynchronous bus speed. Use PyCapture2.BUS_SPEED to set correctly.

asyncBusSpeed

int – Asynchronous bus speed. Use PyCapture2.BUS_SPEED to set correctly.

bandwidthAllocation

Bandwidth allocation flag that tells the camera the bandwidth allocation strategy to employ. Use PyCapture2.BANDWIDTH_ALLOCATION to set correctly.

registerTimeoutRetries

int – The number of retries to perform when a register read/write timeout is received by the library.

registerTimeout

int – Register read/write timeout value (in microseconds).

class PyCapture2.ConfigROM

camera configuration ROM class.

nodeVendorId

int – Vendor ID of a node.

chipIdHi

int – Chip ID (high part).

chipIdLo

int – Chip ID (low part).

unitSpecID

int – Unit Specification ID, usually 0xa02d

unitSWVer

int – Unit software version.

unitSubSWVer

int – Unit sub software version.

vendorUniqueInfo_0

int – first vendor unique info.

vendorUniqueInfo_1

int – second vendor unique info.

vendorUniqueInfo_2

int – third vendor unique info.

vendorUniqueInfo_3
int – last vendor unique info.

pszKeyword
str – keyword.

class PyCapture2.**DRIVER_TYPE**

Bases: object

Types of low level drivers that FlyCapture uses.

CAM_1394
PGRCam.sys.

PRO_1394
PGR1394.sys.

JUJU_1394
firewire_core.

VIDEO1394
video1394.

RAW1394
raw1394.

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

USB_CAM
PGRUsbCam.sys.

USB3_PRO
PGRXHCl.sys.

GIGE_NONE
no GigE drivers used, MS/BSD stack.

GIGE_FILTER
PGRGigE.sys.

GIGE_PRO
PGRGigEPro.sys.

GIGE_LWF
PgrLwf.sys.

UNKNOWN
Unknown driver type.

CAM_1394 = 0

GIGE_FILTER = 9

GIGE_LWF = 11

GIGE_NONE = 8

GIGE_PRO = 10

JUJU_1394 = 2

PRO_1394 = 1

RAW1394 = 4

UNKNOWN = -1
USB3_PRO = 7
USB_CAM = 6
USB_NONE = 5
VIDEO1394 = 3

class `PyCapture2.EmbeddedImageInfo`

Structure containing the current status and availability (via the “available” attribute) of image properties.

timestamp

bool – Whether timestamping is currently active.

gain

bool – Whether gain is currently active.

shutter

bool – Whether the shutter is currently set.

brightness

bool – Whether the brightness is currently available for adjustment.

exposure

bool – Whether the exposure is currently available for adjustment.

whiteBalance

bool – Whether the white balance is currently available for adjustment.

frameCounter

bool – Whether the camera’s frame counter is active.

strobePattern

bool – Whether the strobe is currently available for adjustment.

GPIOPinState

bool – Whether the GPIOPinState is settable and gettable.

ROIPosition

bool – Whether the region of interest is available for adjustment.

available

PyCapture2.availableImageInfo – Whether these properties are available for modification. See *AvailableImageInfo* for reference.

class `PyCapture2.FRAME_RATE`

Bases: `object`

Frame rates in frames per second.

FR_1_875

1.875 fps.

FR_3_75

3.75 fps.

FR_7_5

7.5 fps.

FR_15

15 fps.

FR_30
30 fps.

FR_60
60 fps.

FR_120
120 fps.

FR_240
240 fps.

FORMAT7
Custom frame rate for Format7 functionality.

FORMAT7 = 8

FR_120 = 6

FR_15 = 3

FR_1_875 = 0

FR_240 = 7

FR_30 = 4

FR_3_75 = 1

FR_60 = 5

FR_7_5 = 2

exception `PyCapture2.Fc2error`
Bases: `exceptions.Exception`

An error raised by the PyCapture2 library.

class `PyCapture2.Format7ImageSettings`
Format 7 image settings.

Used to get/set a mode's format 7 settings.

mode
int – Format 7 mode to get/set. Use `PyCapture2.MODE` to set correctly.

offsetX
int – Horizontal image offset.

offsetY
int – Vertical image offset.

width
int – Width of image.

height
int – Height of image.

PixelFormat
int – Pixel format of image. Use `PIXEL_FORMAT` to set correctly.

class `PyCapture2.Format7Info`
Format 7 information for a single mode.

Used to get a mode's format 7 capabilities and settings.

mode
int – Format 7 mode to query. Use `PyCapture2.MODE` to set correctly.

maxWidth
int – The maximum image width.

maxHeight
int – The maximum image height.

offsetHStepSize
int – Horizontal step size for the offset.

offsetVStepSize
int – Vertical stem size for the offset.

imageHStepSize
int – Horizontal step size for the image.

imageVStepSize
int – Vertical step size for the image.

pixelFormatBitField
int – Supported pixel formats in a bit field.

vendorPixelFormatBitField
int – Vendor-unique pixel formats in a bit field.

packetSize
int – Current packet size (in bytes).

minPacketSize
int – Minimum packet size (in bytes) for the current mode.

maxPacketSize
int – Maximum packet size (in bytes) for the current mode.

percentage
float – Current packet size as a percentage of maximum.

class `PyCapture2.Format7PacketInfo`

Format 7 packet information.

Used to determine the possible number of bytes per packet, as well as the recommended.

recommendedBytesPerPacket
int – Recommended bytes per packet.

maxBytesPerPacket
int – Maximum bytes per packet.

minBytesPerPacket
int – Minimum bytes per packet.

class `PyCapture2.GIGE_PROPERTY_TYPE`

Bases: `object`

Possible properties that can be queried from the camera.

GIGE_HEARTBEAT

GIGE_HEARTBEAT_TIMEOUT

GIGE_PACKET_SIZE

GIGE_PACKET_DELAY

GIGE_HEARTBEAT = 0

GIGE_HEARTBEAT_TIMEOUT = 1

GIGE_PACKET_DELAY = 3

GIGE_PACKET_SIZE = 2

class `PyCapture2.GRAB_MODE`

Bases: `object`

The grab strategy employed during image transfer.

This type controls how images that stream off the camera accumulate in a user buffer for handling.

DROP_FRAMES

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.

BUFFER_FRAMES

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_GRAB_MODE `Unspecified grab mode.`

BUFFER_FRAMES = 1

DROP_FRAMES = 0

UNSPECIFIED_GRAB_MODE = 2

class `PyCapture2.GigECamera`

Bases: `PyCapture2.BaseCamera`

A camera object from FlyCapture2. This is specifically for GigE and ethernet cameras.

discoverGigEPacketSize() → `packetSize`

Return the largest packet size possible for the link between the PC and the camera.

Returns The maximum packet size supported by the link.

Return type `packetSize` (int)

getGigEConfig() → `gigEConfig`

Return the camera's current GigE configuration.

Returns `GigEConfig` object containing configuration information.

Return type `gigEConfig` (`PyCapture2.GigEConfig`)

getGigEImageBinningSettings() → (`horzValue`, `vertValue`)

Return the current binning settings on the camera.

Returns

tuple containing: `horzValue` (int): The current horizontal binning value. `vertValue` (int): The current vertical binning value.

Return type (tuple)

getGigEImageSettings () → *gigEImageSettings*

Return the current image settings on the camera.

Returns The current image settings on the camera.

Return type *gigEImageSettings* (*GigEImageSettings*)

getGigEImageSettingsInfo () → *gigEImageSettingsInfo*

Get information about the camera's possible image settings.

Returns *GigEImageSettingsInfo* object holding image settings information.

Return type *GigEImageSettings* (*PyCapture2.GigEImageSettingsInfo*)

getGigEImagingMode () → *mode*

Return the current imaging mode on the camera.

Returns Current imaging mode on the camera. Use *PyCapture2.MODE* to read correctly.

Return type *mode* (int)

getGigEProperty (*propertyType*) → *gigEProperty*

Return information on the specified GigE property.

Parameters **propertyType** (*int*) – The type of property to retrieve information about. Use *PyCapture2.PROPERTY_TYPE* to set correctly.

Returns A *GigEProperty* object containing information about the specified property.

Return type *gigEProperty* (*PyCapture2.GigEProperty*)

getGigEStreamChannelInfo (*channel*) → *gigEStreamChannel*

Return information about the specified stream channel.

Parameters **channel** (*int*) – The channel to retrieve information about.

Returns *GigEStreamChannel* object containing information about the channel.

Return type *gigEStreamChannel* (*PyCapture2.GigEStreamChannel*)

getNumStreamChannels () → *numChannels*

Return the number of stream channels present on the camera.

Returns the number of channels present.

Return type *numChannels* (int)

queryGigEImagingMode (*mode*) → *isSupported*

Return whether the given imaging mode is supported by the camera.

Parameters **mode** (*int*) – The mode to check. Use *PyCapture2.MODE* to set correctly.

Returns Whether the given mode is supported.

Return type *isSupported* (bool)

readGVCPMemory (*address*, *length*) → *values*

Read a GVCP memory block on the camera.

Parameters

- **address** (*int*) – The GVCP address to read from.
- **length** (*int*) – The size of the memory to read.

Returns The list of integers containing read data.

Return type *values* ([int, int, ..])

readGVCPRegister (*address*) → value

Read a GVCP register's value.

Parameters **address** (*int*) – The address of the register to read.

Returns The value of the register.

Return type *value* (*int*)

readGVCPRegisterBlock (*address*, *length*) → values

Read a block of GVCP registers.

Parameters

- **address** (*int*) – The address to read from.
- **length** (*int*) – The number of registers to read.

Returns The list of values read from the registers.

Return type values ([int, int, ..])

setGigEConfig (*gigEConfig* = None, ***kwargs*) → None

Set the camera's GigE configuration.

There are two ways to call this function: The first is with a GigEConfig object that has all the values to be updated.

The second is by specifying each property to update in the arguments. The arguments that can be specified are:

enablePacketResend
registerTimeoutRetries
registerTimeout

Information about these properties can be found in the GigEConfig documentation.

Parameters

- **gigEConfig** (`PyCapture2.GigEConfig`) – The configuration object to get all properties from.
- **kwargs** (*{...}*) – The second method of calling: specify the properties to change as keywords.

setGigEImageBinningSettings (*horzValue*, *vertValue*) → None

Set the binning settings on the camera.

Parameters

- **horzValue** (*int*) – Horizontal binning value to set.
- **vertValue** (*int*) – Vertical binning value to set.

setGigEImageSettings (*gigEImageSettings* = None, ***kwargs*) → None

Set the specified image settings to the camera.

There are two ways to call this function: The first is with a GigEImageSettings object that has all the values to be updated.

The second is by specifying each property to update in the arguments. The arguments that can be specified are:

offsetX (*int*)
offsetY (*int*)

width (int)
height (int)
pixelFormat (int)

Information about these properties can be found in the `GigEImageSettings` documentation.

Parameters

- **imageSettings** (`PyCapture2.GigEImageSettings`) – The image settings object to get all properties from.
- **kwargs** (`{...}`) – The second method of calling: specify the properties to change as keywords.

setGigEImagingMode (*mode*) → None

Set the camera's imaging mode to the specified mode.

Parameters *mode* (*int*) – Mode to set the camera to. Use `PyCapture2.MODE` to set correctly.

setGigEProperty (*gigEProperty = None, **kwargs*) → None

There are two ways to call this function: The first is with a `GigEProperty` object that has all the values to be updated.

The second is by specifying each property to update in the arguments. The arguments that can be specified are:

propType (int)
isReadable (bool)
isWritable (bool)
min (int)
max (int)
value (int)

propType MUST be specified for this method of calling `setGigEProperty` to work correctly.

Information about these properties can be found in the `GigEProperty` documentation.

Parameters

- **gigEProperty** (`PyCapture2.GigEProperty`) – The property object to get all properties from.
- **kwargs** (`{...}`) – The second method of calling: specify the properties to change as keywords.

writeGVCPMemory (*address, values*) → None

Write a GVCP memory block on the camera.

Parameters

- **address** (*int*) – The GVCP address to write to.
- **values** (*[int, int, ...]*) – The list of integer values to write to the memory.

writeGVCPRegister (*address, value*) → None

Write to a GVCP (GigE Vision Control Protocol) register.

Parameters

- **address** (*int*) – GVCP address to be written to.

- **value** (*int*) – The value to be written.

writeGVCPRegisterBlock (*address*, *values*) → None

Write to a block of GVCP registers.

Parameters

- **address** (*int*) – The address to write to
- **values** (*[int, int, ...]*) – The list of integer values to write to register.

class PyCapture2.**GigEConfig**

Configuration for a GigE camera.

These options should generally be set before starting isochronous transfer.

enablePacketResend

bool – Turn on/off packet resend functionality.

registerTimeoutRetries

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

registerTimeout

int – Register read/write timeout value (in microseconds).

class PyCapture2.**GigEImageSettings**

Image settings for a GigE camera.

Used to get/set a GigE Camera's settings.

offsetX

int – Horizontal image offset.

offsetY

int – Vertical image offset.

width

int – Width of image.

height

int – Height of image.

pixelFormat

int – Pixel format of image. Use PyCapture2.PIXEL_FORMAT to set correctly.

class PyCapture2.**GigEImageSettingsInfo**

Format 7 information for a single mode.

Used to get a mode's format 7 capabilities and settings.

maxWidth

int – Maximum image width.

maxHeight

int – Maximum image height.

offsetHStepSize

int – Horizontal step size for the offset.

offsetVStepSize

int – Vertical step size for the offset.

imageHStepSize

int – Horizontal step size for the image.

imageVStepSize

int – Vertical step size for the image.

pixelFormatBitField

int – Supported pixel formats in a bit field.

vendorPixelFormatBitField

int – Vendor unique pixel formats in a bit field.

class PyCapture2.GigEProperty

A property specific to GigE cameras.

Used to get/set GigE properties.

propType

int – The type of property to get/set. Use GIGE_PROPERTY_TYPE to set correctly.

isReadable

bool – Whether the property is readable.

isWritable

bool – Whether the property is writable.

min

int – Minimum value of the property.

max

int – Maximum value of the property.

value

int – Current value of the property.

class PyCapture2.GigEStreamChannel

Information about a single GigE stream channel.

Used to get/set stream channel info.

networkInterfaceIndex

int – Network interface index used/to use.

hostPort

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

doNotFragment

bool – Disable IP fragmentation of packets.

packetSize

int – Size of a packet (in bytes).

interPacketDelay

int – Inter-Packet delay, in timestamp counter units.

destinationIpAddress

int, int, int, int – IP address of packet destination.

sourcePort

int – Source UDP port of the stream channel.

class PyCapture2.IMAGE_FILE_FORMAT

Bases: object

File formats to be used for saving images to disk.

FROM_FILE_EXT

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.

BMP = 2

FROM_FILE_EXT = -1

JPEG = 3

JPEG2000 = 4

PGM = 0

PNG = 6

PPM = 1

RAW = 7

TIFF = 5

class PyCapture2.**INTERFACE_TYPE**

Bases: object

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

IEEE1394

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

USB_2

USB 2.0.

USB_3

USB 3.0.

GIGE

GigE.

UNKNOWN

Unknown interface.

GIGE = 3

IEEE1394 = 0

UNKNOWN = 4

USB_2 = 1

USB_3 = 2

class `PyCapture2.Image`

Bases: `object`

An image from the FlyCapture 2 library.

convert (*format*) → `image`

Return an image with the same data in a different format.

Parameters **format** (*int*) – Format to convert the image to. Use `PyCapture2.PIXEL_FORMAT` to set correctly.

Returns An image object in the specified format.

Return type `image` (`PyCapture2.Image`)

getBayerTileFormat () → `bayerFormat`

Return the bayer tile format of the image.

Returns Bayer tile format of the image. Use `BAYER_FORMAT` to read correctly.

Return type `bayerFormat` (`int`)

getCols () → `numCols`

Return the number of cols in the image.

Returns The number of cols in the image.

Return type `numCols` (`int`)

getData () → `imageData`

Return a list of the raw image data. This is a very large list of integers.

Returns The raw image data.

Return type `imageData` (`[int, int, ..]`)

getDataSize () → `dataSize`

Return the maximum length of data possible.

Returns The maximum possible length of image data.

Return type `dataSize` (`int`)

getPixelFormat () → `format`

Return the pixel format of the image.

Returns Pixel format of the image. Use `PyCapture2.PIXEL_FORMAT` to read correctly.

Return type `format` (`int`)

getRecievedDataSize () → `dataSize`

Return the actual length of the image data.

Returns The length of the image data.

Return type `dataSize` (`int`)

getRows () → `numRows`

Return the number of rows in the image.

Returns The number of rows in the image.

Return type `numRows` (`int`)

getStride () → stride

Return the stride of the image.

Returns The stride of the image.

Return type stride (int)

getTimeStamp () → timeStamp

Return timestamp from image capture.

Returns TimeStamp object containing image capture time.

Return type *timeStamp* (*PyCapture2.TimeStamp*)

save (fileName, format, option = None) → None

Save an image as fileName with type format. option specifies additional options for saving.

Parameters

- **fileName** (*str*) – The name to save the file as.
- **format** (*int*) – The format to save the file as. Use PyCapture2.IMAGE_FILE_FORMAT to set correctly.
- **option** – Additional parameters for saving the image. The type must match the file format! option can be one of these types:

PNGOption PPMOption PGMOption TIFFOption JPEGOption JPG2Option BMPOption

class PyCapture2.**ImageStatistics**

Bases: object

calculateStatistics (image) → None

Calculate image statistics from the given image.

Channels to be used must be enabled before this function call.

Parameters **image** (*PyCapture2.Image*) – The image to calculate statistics from.

disableAllChannels () → None

Disable all channels for image statistics analysis.

enableAllChannels () → None

Enable all channels for image statistics analysis.

enableGreyChannel () → None

Enable only the grey channel for image statistics analysis.

enableHSLChannel () → None

Enable only the hue, saturation, and lightness channels for image statistics analysis.

enableRGBChannel () → None

Enable only the red, green, and blue channels for image statistics analysis.

getChannelStatus (channel) → enabled

Return the status for the given channel.

Parameters **channel** (*int*) – The channel to retrieve status for. Use PyCapture2.STATISTICS_CHANNEL to set correctly.

Returns Whether the channel is currently enabled.

Return type *enabled* (bool)

getHistogram (*channel*) → histogram

Return a histogram of the values in the image.

Parameters **channel** (*int*) – The channel to create the histogram from. Use PyCapture2.STATISTICS_CHANNEL to set correctly.

Returns A list containing the histogram values. Its length is equal to the number of possible values of a pixel in the image.

Return type histogram ([int, int, ..])

getMean (*channel*) → mean

Return the mean value of a channel of the image.

Parameters **channel** (*int*) – The channel to determine the mean from. Use PyCapture2.STATISTICS_CHANNEL to set correctly.

Returns The mean value of the image.

Return type mean (float)

getNumPixelValues (*channel*) → numValues

Return the number of unique pixel values in the image.

Parameters **channel** (*int*) – The channel to determine the number of values from. Use PyCapture2.STATISTICS_CHANNEL to set correctly.

Returns The number of unique pixel values.

Return type numValues (int)

getPixelValueRange (*channel*) -> (*min*, *max*)

Get the actual range of the pixel values in a statistics channel.

Parameters **channel** (*int*) – The channel to determine the range of. Use PyCapture2.STATISTICS_CHANNEL to set correctly.

Returns

tuple containing: min (int): The minimum pixel value in the image. max (int): The maximum pixel value in the image.

Return type (tuple)

getRange (*channel*) -> (*min*, *max*)

Return the maximum and minimum possible values for any given pixel in the image.

This is generally 0-255 for 8bit images, and 0-65535 for 16bit images.

Parameters **channel** (*int*) – The channel to determine the range of. Use PyCapture2.STATISTICS_CHANNEL to set correctly.

Returns

tuple containing: min (int): The minimum value a pixel can have. max (int): The maximum value a pixel can have.

Return type (tuple)

getStatistics (*channel*) → (rangeMin, rangeMax), (pixValMin, pixValMax), numPixVals, pixValMean, histogram

Return all statistics for the given channel.

Parameters **channel** (*int*) – The channel to create the statistics from. Use PyCapture2.STATISTICS_CHANNEL to set correctly.

Returns

tuple containing: rangeMin (int): The minimum value a pixel can have. rangeMax (int): The maximum value a pixel can have. pixValMin (int): The minimum pixel value in the image. pixValMax (int): The maximum pixel value in the image. numPixVals (int): The number of unique pixel values. pixValMean (float): The mean value of the image. histogram ([int, int, ...]): A list containing the histogram values. Its length is equal to the number of possible values of a pixel in the image.

Return type (tuple)

setChannelStatus (*channel, enabled*) → None
Set the status of a single statistics channel.

Parameters

- **channel** (*int*) – The channel whose status to set. Use PyCapture2.STATISTICS_CHANNEL to set correctly.
- **enabled** (*bool*) – Whether the channel will be enabled or disabled.

class PyCapture2.**JPEGOption**
Structure containing options for saving JPEG images.

progressive
bool – Whether to save as a progressive JPEG file.

quality
int – JPEG image quality in range (0-100)

class PyCapture2.**JPG2Option**
Structure containing options for saving JPEG2000 images.

quality
int – JPEG saving quality in range (1-512)

class PyCapture2.**LUTData**
Information about the camera's lookup table.

Used to get/set lookup table settings.

supported
bool – Flag indicating if LUT is supported.

enabled
bool – Flag indicating if LUT is enabled.

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

numChannels
int – The number of channels per bank available.

inputBitDepth
int – The input bit depth of the LUT.

outputBitDepth
int – The output bit depth of the LUT.

numEntries
int – The number of entries in the LUT.

class PyCapture2.**MODE**
Bases: object
Camera modes for DCAM formats as well as Format7.

FC2_MODE_0
FC2_MODE_1
FC2_MODE_2
FC2_MODE_3
FC2_MODE_4
FC2_MODE_5
FC2_MODE_6
FC2_MODE_7
FC2_MODE_8
FC2_MODE_9
FC2_MODE_10
FC2_MODE_11
FC2_MODE_12
FC2_MODE_13
FC2_MODE_14
FC2_MODE_15
FC2_MODE_16
FC2_MODE_17
FC2_MODE_18
FC2_MODE_19
FC2_MODE_20
FC2_MODE_21
FC2_MODE_22
FC2_MODE_23
FC2_MODE_24
FC2_MODE_25
FC2_MODE_26
FC2_MODE_27
FC2_MODE_28
FC2_MODE_29
FC2_MODE_30
FC2_MODE_31
FC2_NUM_MODES
 Number of modes.
MODE_0 = 0
MODE_1 = 1

MODE_10 = 10
MODE_11 = 11
MODE_12 = 12
MODE_13 = 13
MODE_14 = 14
MODE_15 = 15
MODE_16 = 16
MODE_17 = 17
MODE_18 = 18
MODE_19 = 19
MODE_2 = 2
MODE_20 = 20
MODE_21 = 21
MODE_22 = 22
MODE_23 = 23
MODE_24 = 24
MODE_25 = 25
MODE_26 = 26
MODE_27 = 27
MODE_28 = 28
MODE_29 = 29
MODE_3 = 3
MODE_30 = 30
MODE_31 = 31
MODE_4 = 4
MODE_5 = 5
MODE_6 = 6
MODE_7 = 7
MODE_8 = 8
MODE_9 = 9
NUM_MODES = 32

class PyCapture2.**NODE_TYPE**
Bases: object
Type of node.
NODE_COMPUTER
The node is a computer.

NODE_BUS

The node is a bus.

NODE_CAMERA

The node is a camera.

NODE_NODE

Unknown node type.

NODE_BUS = 1

NODE_CAMERA = 2

NODE_COMPUTER = 0

NODE_NODE = 3

class PyCapture2.OS_TYPE

Bases: object

Possible operating systems.

WINDOWS_X86

All Windows 32-bit variants.

WINDOWS_X64

All Windows 64-bit variants.

LINUX_X86

All Linux 32-bit variants.

LINUX_X64

All Linux 32-bit variants.

MAC

Mac OSX.

UNKNOWN_OS

Unknown operating system

LINUX_X64 = 3

LINUX_X86 = 2

MAC = 4

UNKNOWN_OS = 5

WINDOWS_X64 = 1

WINDOWS_X86 = 0

class PyCapture2.PCIE_BUS_SPEED

Bases: object

Speed of PCIE busses.

FC2_PCIE_BUSSPEED_2_5

2.5 Gb/s

FC2_PCIE_BUSSPEED_5_0

5.0 Gb/s

FC2_PCIE_BUSSPEED_UNKNOWN

Speed is unknown

PCIE_2_5 = 0

PCIE_5_0 = 1

UNKNOWN = -1

class PyCapture2.**PGMOption**

Structure containing options for saving PGM images.

binaryFile

bool – Whether to save the PGM as a binary file.

class PyCapture2.**PIXEL_FORMAT**

Bases: *object*

Pixel formats available for Format7 modes.

MONO8

8 bits of mono information.

YUV8_411

YUV 4:1:1.

YUV8_422

YUV 4:2:2.

444YUV8_444

YUV 4:4:4.

RGB8

R = G = B = 8 bits.

MONO16

16 bits of mono information.

RGB16

R = G = B = 16 bits.

S_MONO16

16 bits of signed mono information.

S_RGB16

R = G = B = 16 bits signed.

RAW8

8 bit raw data output of sensor.

RAW16

16 bit raw data output of sensor.

MONO12

12 bits of mono information.

RAW12

12 bit raw data output of sensor.

BGR

24 bit BGR.

BGRU

32 bit BGRU.

RGB

24 bit RGB.

RGBU

32 bit RGBU.

BGR16

R = G = B = 16 bits.

BGRU16

64 bit BGRU.

YUV8_JPEG_422

JPEG compressed stream.

NUM_PIXEL_FORMATS

Number of pixel formats.

UNSPECIFIED_PIXEL_FORMAT

Unspecified pixel format.

BGR = -2147483640

BGR16 = 33554433

BGRU = 1073741832

BGRU16 = 33554434

MONO12 = 1048576

MONO16 = 67108864

MONO8 = -2147483648

NUM_PIXEL_FORMATS = 20

RAW12 = 524288

RAW16 = 2097152

RAW8 = 4194304

RGB = 134217728

RGB16 = 33554432

RGB8 = 134217728

RGBU = 1073741826

S_MONO16 = 16777216

S_RGB16 = 8388608

UNSPECIFIED_PIXEL_FORMAT = 0

YUV8_411 = 1073741824

YUV8_422 = 536870912

YUV8_444 = 268435456

YUV8_JPEG_422 = 1073741825

class `PyCapture2.PNGOption`

Structure containing options for saving PNG images.

interlaced

bool – Whether to save the PNG as interlaced.

compressionLevel

int – Level of compression for the image, on the range (0-9)

class PyCapture2.**PORT_TYPE**

Bases: object

Possible states of a port on a node.

PORT_NOT_CONNECTED

PORT_CONNECTED_TO_PARENT

PORT_CONNECTED_TO_CHILD

PORT_CONNECTED_TO_CHILD = 3

PORT_CONNECTED_TO_PARENT = 2

PORT_NOT_CONNECTED = 1

class PyCapture2.**PPMOption**

Structure containing options for saving PPM images.

binaryFile

bool – Whether to save the PPM as a binary file.

class PyCapture2.**PROPERTY_TYPE**

Bases: object

Camera properties.

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

BRIGHTNESS

AUTO_EXPOSURE

SHARPNESS

WHITE_BALANCE

HUE

SATURATION

GAMMA

IRIS

FOCUS

ZOOM

PAN

TILT

SHUTTER

GAIN

TRIGGER_MODE

TRIGGER_DELAY

FRAME_RATE

TEMPERATURE

UNSPECIFIED_PROPERTY_TYPE

AUTO_EXPOSURE = 1

BRIGHTNESS = 0
FOCUS = 8
FRAME_RATE = 16
GAIN = 13
GAMMA = 6
HUE = 4
IRIS = 7
PAN = 10
SATURATION = 5
SHARPNESS = 2
SHUTTER = 12
TEMPERATURE = 17
TILT = 11
TRIGGER_DELAY = 15
TRIGGER_MODE = 14
UNSPECIFIED_PROPERTY_TYPE = 18
WHITE_BALANCE = 3
ZOOM = 9

class `PyCapture2.Property`

Data from a specific camera property.

Used to get and set property information.

type

PyCapture2.PROPERTY_TYPE – The type of property that is described with the following values.

present

bool – The flag controlling if the property is present on the camera.

absControl

bool – The flag controlling absolute mode (real world units) or non-absolute mode (camera internal units)

onePush

bool – The flag controlling one push.

onOff

bool – Flag controlling activation of property.

autoManualMode

bool – Flag controlling auto/manual.

ValueA

int – Value A

ValueB

int – Value B

absValue

float – Floating point value.

class PyCapture2.**PropertyInfo**

Information about a specific camera property.

type

int – The type of property that is described with the following values. Use PyCapture2.PROPERTY_TYPE to set correctly.

present

bool – The flag indicating if the property is present.

autoSupported

bool – The flag indicating if auto is supported.

manualSupported

bool – The flag indicating if manual is supported.

onOffSupported

bool – The flag indicating if activation is supported.

onePushSupported

bool – The flag indicating if one push is supported.

absValSupported

bool – The flag indicating if absolute mode is supported.

readOutSupported

bool – The flag indicating if a property value can be read out.

min

int – Minimum value.

max

int – Maximum value.

absMin

float – Minimum value (as a float).

absMax

float – Maximum value (as a float).

units

str – Textual description of units.

unitAbbr

str – Abbreviated textual description of units.

class PyCapture2.**STATISTICS_CHANNEL**

Bases: object

Channels that allow statistics to be calculated.

GREY

RED

GREEN

BLUE

HUE

SATURATION

LIGHTNESS

BLUE = 3

GREEN = 2

GREY = 0

HUE = 4

LIGHTNESS = 6

RED = 1

SATURATION = 5

class `PyCapture2.StrobeControl`

Data about the camera strobe.

Used to get/set strobe settings.

source

int – Source value.

onOff

bool – Flag controlling state of strobe.

polarity

int – Signal polarity.

delay

float – Signal delay (in ms)

duration

float – Signal duration (in ms)

class `PyCapture2.StrobeInfo`

Information about the camera's strobe settings and capabilities.

source

int – The source value.

present

bool – Whether strobe is present.

readOutSupported

bool – Flag indicating if the strobe value can be read.

onOffSupported

bool – Flag indicating if activating/deactivating strobe is supported.

polaritySupported

bool – Flag indicating if strobe polarity is supported.

minValue

float – Minimum value.

maxValue

float – Maximum value.

class `PyCapture2.SystemInfo`

Description of the system connected to the camera.

osType

int – Operating system type. Use `OS_TYPE` to get/set correctly.

osDescription

str – Detailed description of the operating system.

byteOrder

int – Byte order of the system. Use `BYTE_ORDER` to set correctly.

sysMemSize

int – Amount of memory available on the system.

cpuDescription

str – Detailed description of the CPU.

numCpuCores

int – Number of cores on all CPUs on the system.

driverList

str – List of drivers used.

libraryList

str – List of libraries used.

gpuDescription

str – Detailed description of the GPU.

screenWidth

int – Screen resolution width, in pixels.

screenHeight

int – Screen resolution height, in pixels.

class PyCapture2.TIFFOption

Structure containing options for saving TIFF images.

compression

int – Compression method to use for encoding. Use `TIFF_COMPRESSION` to set correctly.

class PyCapture2.TIFF_COMPRESSION

Bases: `object`

TIFF compression method.

NONE

Save without any compression.

PACKBITS

Save using PACKBITS compression.

DEFLATE

Save using DEFLATE compression (ZLIB compression).

ADOBE_DEFLATE

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.

ADOBE_DEFLATE = 4

CCITTFAX3 = 5

CCITTFAX4 = 6

DEFLATE = 3

JPEG = 8

NONE = 1

PACKBITS = 2

class `PyCapture2.Timestamp`

Information detailing the time an image was taken.

seconds

long – Seconds.

microSeconds

int – Microseconds.

cycleSeconds

int – 1394 cycle time, in seconds.

cycleCount

int – 1394 cycle time count.

cycleOffset

int – 1394 cycle time offset.

class `PyCapture2.TopologyNode`

Bases: `object`

addChild (*child*) → `None`

Add the specified topologyNode as a child of the node.

Parameters **child** (`PyCapture2.topologyNode`) – The child node to add.

addPortType (*type*) → `None`

Add the specified port type as a port of the node.

Parameters **type** (*int*) – The type of port to add. Use `PyCapture2.PORT_TYPE` to read correctly.

assignGuid (*guid*, *deviceId*, *nodeType* = `None`) → `None`

Assign a guid, device ID, and optionally a node type to the node.

Parameters

- **guid** (*int*, *int*, *int*, *int*) – The GUID to be assigned.
- **deviceId** (*int*) – The device ID to be assigned.
- **nodeType** (*int*) – The optional node type to be assigned. Use `PyCapture2.NODE_TYPE` to set correctly.

getChild (*index*) → `childNode`

Get child node located at the specified position.

Parameters **index** (*int*) – The position of the node.

Returns The topologyNode of the child.

Return type `childNode` (`PyCapture2.TopologyNode`)

getDeviceID () → deviceID

Get the device ID associated with the node.

Returns The ID of the object represented by the node.

Return type deviceID (int)

getGuid () → guid

Get the guid associated with the node.

Returns The guid of the object represented by the node.

Return type guid (int, int, int, int)

getInterfaceType () → interfaceType

Get the interface type associated with the node.

Returns The interface type of the node. Use PyCapture2.INTERFACE_TYPE to read correctly.

Return type *interfaceType* (int)

getNodeType () → nodeType

Get the type of the node.

Returns The type of the node. Use PyCapture2.NODE_TYPE to read correctly.

Return type nodeType (int)

getNumChildren () → children

Get the number of children associated with the node.

Returns The number of child nodes.

Return type children (int)

getNumPorts () → numPorts

Get the number of ports of the node.

Returns The number of ports.

Return type numPorts (int)

getPortType (*position*) → portType

Get the type of port located at the specified position.

Parameters **position** (*int*) – The position of the port.

Returns The type of port. Use PyCapture2.PORT_TYPE to read correctly.

Return type portType (int)

class PyCapture2.**TriggerMode**

Properties of a camera trigger.

Used with Camera.setTriggerMode().

onOff

bool – The flag controlling activation of the trigger.

polarity

int – The polarity value.

source

int – The source value.

mode

int – The mode value.

parameter*int* – The parameter value.**class** PyCapture2.VIDEO_MODE

Bases: object

DCAM video modes.

VM_160x120YUV444

160x120 YUV444.

VM_320x240YUV422

320x240 YUV422.

VM_640x480YUV411

640x480 YUV411.

VM_640x480YUV422

640x480 YUV422.

VM_640x480RGB

640x480 24-bit RGB.

VM_640x480Y8

640x480 8-bit.

VM_640x480Y16

640x480 16-bit

VM_800x600YUV422

800x600 YUV422.

VM_800x600RGB

800x600 RGB.

VM_800x600Y8

800x600 8-bit.

VM_800x600Y16

800x600 16-bit.

VM_1024x768YUV422

1024x768 YUV422.

VM_1024x768RGB

1024x768 RGB.

VM_1024x768Y8

1024x768 8-bit.

VM_1024x768Y16

1024x768 16-bit.

VM_1280x960YUV422

1280x960 YUV422.

VM_1280x960RGB

1280x960 RGB.

VM_1280x960Y8

1280x960 8-bit.

VM_1280x960Y16

1280x960 16-bit.

VM_1600x1200YUV422
1600x1200 YUV422.

VM_1600x1200RGB
1600x1200 RGB.

VM_VM_1600x1200Y8
1600x1200 8-bit.

VM_1600x1200Y16
1600x1200 16-bit.

FORMAT7
Custom video mode for Format7 functionality.

NUM_VIDEOMODES
Number of possible video modes

FORMAT7 = 23

NUM_VIDEOMODES = 24

VM_1024x768RGB = 12

VM_1024x768Y16 = 14

VM_1024x768Y8 = 13

VM_1024x768YUV422 = 11

VM_1280x960RGB = 16

VM_1280x960Y16 = 18

VM_1280x960Y8 = 17

VM_1280x960YUV422 = 15

VM_1600x1200RGB = 20

VM_1600x1200Y16 = 22

VM_1600x1200Y8 = 21

VM_1600x1200YUV422 = 19

VM_160x120YUV444 = 0

VM_320x240YUV422 = 1

VM_640x480RGB = 4

VM_640x480Y16 = 6

VM_640x480Y8 = 5

VM_640x480YUV411 = 2

VM_640x480YUV422 = 3

VM_800x600RGB = 8

VM_800x600Y16 = 10

VM_800x600Y8 = 9

VM_800x600YUV422 = 7

`PyCapture2.checkDriver(guid)` → None

Check for driver compatibility for the given camera guid.

Parameters `guid(int, int, int, int)` – The guid of the device to check.

`PyCapture2.determineBitsPerPixel(format)` → bitsPerPixel

Calculate the bits per pixel for the specified pixel format.

Parameters `format(int)` – The pixel format. Use `PyCapture2.PIXEL_FORMAT` to set correctly.

Returns The bits per pixel.

Return type bitsPerPixel (int)

`PyCapture2.getDefaultColorProcessing()` → defaultMethod

Get the default color processing algorithm.

Returns The default method. Use `PyCapture2.COLOR_PROCESSING` to read correctly.

Return type defaultMethod (int)

`PyCapture2.getDefaultOutputFormat()` → format

Get the default output format.

Returns The default pixel format. Use `PyCapture2.PIXEL_FORMAT` to read correctly.

Return type format (int)

`PyCapture2.getDriverDeviceName(guid)` → name

Get the driver's name for a device.

Parameters `guid(int, int, int, int)` – The guid of the device to check.

Returns The name of the device.

Return type name (str)

`PyCapture2.getLibraryVersion()` → version

Get the FlyCapture 2 library version.

Returns The library version. It's format is (major, minor, type, build)

Return type version (int, int, int, int)

`PyCapture2.getRegisterString(registerValue)` → registerString

Return a text representation of the register value.

Parameters `registerValue(int)` – The register value to query.

Returns The textual representation of the register value.

Return type registerString (str)

`PyCapture2.getSystemInfo()` → systemInfo

Get system information.

Returns A SystemInfo object containing the system information.

Return type systemInfo (*PyCapture2.SystemInfo*)

`PyCapture2.launchBrowser(url)` → None

Launch a URL in the system's default browser.

Parameters `url(str)` – The URL to open.

`PyCapture2.launchCommand(command)` → None

Execute a command in the terminal.

This is a blocking call that will return when the command completes.

Parameters **command** (*str*) – The command to execute.

`PyCapture2.launchCommandAsync (command, function, *arguments) → None`

Execute a command in the terminal.

This is a non-blocking call that will return immediately. The return value of the command can be retrieved in the callback.

Parameters

- **command** (*str*) – The command to execute.
- **function** (*function*) – The function to call when the command is complete.
- **arguments** – Any additional arguments will be passed to the callback function when it is called.

`PyCapture2.launchHelp (fileName) → None`

Open a CHM file in the system default CHM viewer.

Parameters **fileName** (*str*) – Filename of the CHM file to open.

`PyCapture2.setDefaultColorProcessing (defaultMethod) → None`

Set the default color processing algorithm.

This method will be used for any image with the DEFAULT algorithm set. The method used is determined at the time of the Convert() call, therefore the most recent execution of this function will take precedence. The default setting is shared within the current process.

Parameters **defaultMethod** (*int*) – The method to set as default. Use `PyCapture2.COLOR_PROCESSING` to set correctly.

`PyCapture2.setDefaultOutputFormat (format) → None`

Set the default output pixel format.

This format will be used for any call to Convert() that does not specify an output format. The format used will be determined at the time of the Convert() call, therefore the most recent execution of this function will take precedence. The default is shared within the current process.

Parameters **format** (*int*) – The pixel format to set as default. Use `PyCapture2.PIXEL_FORMAT` to set correctly.

`PyCapture2.startSyncCapture (cameras) → None`

Start synchronized isochronous image capture on multiple cameras.

Parameters **cameras** (*[PyCapture2.Camera, PyCapture2.Camera, ...]*) – A list of cameras to start isochronous capture.

PYTHON MODULE INDEX

p

PyCapture2, [67](#)

Symbols

444YUV8_444 (PIXEL_FORMAT attribute), 6
 444YUV8_444 (PyCapture2.PIXEL_FORMAT attribute), 107

A

absControl (Property attribute), 25
 absControl (PyCapture2.Property attribute), 110
 absMax (PropertyInfo attribute), 24
 absMax (PyCapture2.PropertyInfo attribute), 111
 absMin (PropertyInfo attribute), 24
 absMin (PyCapture2.PropertyInfo attribute), 111
 absValSupported (PropertyInfo attribute), 24
 absValSupported (PyCapture2.PropertyInfo attribute), 111
 absValue (Property attribute), 25
 absValue (PyCapture2.Property attribute), 110
 addChild() (PyCapture2.TopologyNode method), 61, 114
 addPortType() (PyCapture2.TopologyNode method), 61, 114
 ADOBE_DEFLATE (PyCapture2.TIFF_COMPRESSION attribute), 113
 ADOBE_DEFLATE (TIFF_COMPRESSION attribute), 15
 ANY (BUS_SPEED attribute), 10
 ANY (PyCapture2.BUS_SPEED attribute), 70
 append() (PyCapture2.AVIRecorder method), 55, 68
 ARRIVAL (BUS_CALLBACK_TYPE attribute), 17
 ARRIVAL (PyCapture2.BUS_CALLBACK_TYPE attribute), 69
 assignGuid() (PyCapture2.TopologyNode method), 61, 114
 asyncBusSpeed (Config attribute), 23
 asyncBusSpeed (PyCapture2.Config attribute), 88
 AUTO_EXPOSURE (PROPERTY_TYPE attribute), 10
 AUTO_EXPOSURE (PyCapture2.PROPERTY_TYPE attribute), 109
 autoManualMode (Property attribute), 25
 autoManualMode (PyCapture2.Property attribute), 110
 autoSupported (PropertyInfo attribute), 24
 autoSupported (PyCapture2.PropertyInfo attribute), 111

available (EmbeddedImageInfo attribute), 23
 available (PyCapture2.EmbeddedImageInfo attribute), 90
 AvailableImageInfo (class in PyCapture2), 21, 68
 AVIOpen() (PyCapture2.AVIRecorder method), 55, 67
 AVIRecorder (class in PyCapture2), 55, 67

B

BANDWIDTH_ALLOCATION (class in PyCapture2), 8, 68
 bandwidthAllocation (Config attribute), 23
 bandwidthAllocation (PyCapture2.Config attribute), 88
 BASE_T_10 (BUS_SPEED attribute), 9
 BASE_T_10 (PyCapture2.BUS_SPEED attribute), 70
 BASE_T_100 (BUS_SPEED attribute), 9
 BASE_T_100 (PyCapture2.BUS_SPEED attribute), 70
 BASE_T_1000 (BUS_SPEED attribute), 9
 BASE_T_1000 (PyCapture2.BUS_SPEED attribute), 70
 BASE_T_10000 (BUS_SPEED attribute), 10
 BASE_T_10000 (PyCapture2.BUS_SPEED attribute), 70
 BaseCamera (class in PyCapture2), 39, 71
 BAYER_FORMAT (class in PyCapture2), 14, 69
 bayerTileFormat (CameraInfo attribute), 21
 bayerTileFormat (PyCapture2.CameraInfo attribute), 86
 BGGR (PyCapture2.BAYER_FORMAT attribute), 69
 BGR (PIXEL_FORMAT attribute), 7
 BGR (PyCapture2.PIXEL_FORMAT attribute), 107, 108
 BGR16 (PIXEL_FORMAT attribute), 7
 BGR16 (PyCapture2.PIXEL_FORMAT attribute), 107, 108
 BGRU (PIXEL_FORMAT attribute), 7
 BGRU (PyCapture2.PIXEL_FORMAT attribute), 107, 108
 BGRU16 (PIXEL_FORMAT attribute), 7
 BGRU16 (PyCapture2.PIXEL_FORMAT attribute), 108
 BIG_ENDIAN (BYTE_ORDER attribute), 16
 BIG_ENDIAN (PyCapture2.BYTE_ORDER attribute), 71
 binaryFile (PGMOption attribute), 33
 binaryFile (PPMOption attribute), 32
 binaryFile (PyCapture2.PGMOption attribute), 107
 binaryFile (PyCapture2.PPMOption attribute), 109

- BLUE (PyCapture2.STATISTICS_CHANNEL attribute), 111
 - BLUE (STATISTICS_CHANNEL attribute), 15
 - BMP (IMAGE_FILE_FORMAT attribute), 6
 - BMP (PyCapture2.IMAGE_FILE_FORMAT attribute), 99
 - BMPOption (class in PyCapture2), 33, 69
 - brightness (AvailableImageInfo attribute), 22
 - brightness (EmbeddedImageInfo attribute), 22
 - BRIGHTNESS (PROPERTY_TYPE attribute), 10
 - brightness (PyCapture2.AvailableImageInfo attribute), 68
 - brightness (PyCapture2.EmbeddedImageInfo attribute), 90
 - BRIGHTNESS (PyCapture2.PROPERTY_TYPE attribute), 109
 - BUFFER_FRAMES (GRAB_MODE attribute), 9
 - BUFFER_FRAMES (PyCapture2.GRAB_MODE attribute), 93
 - BUS_CALLBACK_TYPE (class in PyCapture2), 17, 69
 - BUS_RESET (BUS_CALLBACK_TYPE attribute), 17
 - BUS_RESET (PyCapture2.BUS_CALLBACK_TYPE attribute), 69
 - BUS_SPEED (class in PyCapture2), 9, 69
 - BusManager (class in PyCapture2), 35, 79
 - busNumber (CameraInfo attribute), 21
 - busNumber (PyCapture2.CameraInfo attribute), 87
 - BYTE_ORDER (class in PyCapture2), 16, 71
 - byteOrder (PyCapture2.SystemInfo attribute), 112
 - byteOrder (SystemInfo attribute), 31
- ## C
- calculateStatistics() (PyCapture2.ImageStatistics method), 57, 101
 - CallbackData (class in PyCapture2), 32, 83
 - CAM_1394 (DRIVER_TYPE attribute), 8
 - CAM_1394 (PyCapture2.DRIVER_TYPE attribute), 89
 - Camera (class in PyCapture2), 47, 83
 - cameraCurrents (CameraStats attribute), 27
 - cameraCurrents (PyCapture2.CameraStats attribute), 87
 - CameraInfo (class in PyCapture2), 21, 86
 - cameraPowerUp (CameraStats attribute), 27
 - cameraPowerUp (PyCapture2.CameraStats attribute), 87
 - CameraStats (class in PyCapture2), 27, 87
 - cameraVoltages (CameraStats attribute), 27
 - cameraVoltages (PyCapture2.CameraStats attribute), 87
 - CCITTFAX3 (PyCapture2.TIFF_COMPRESSION attribute), 113, 114
 - CCITTFAX3 (TIFF_COMPRESSION attribute), 15
 - CCITTFAX4 (PyCapture2.TIFF_COMPRESSION attribute), 113, 114
 - CCITTFAX4 (TIFF_COMPRESSION attribute), 15
 - checkDriver() (in module PyCapture2), 65, 117
 - chipIdHi (ConfigROM attribute), 20
 - chipIdHi (PyCapture2.ConfigROM attribute), 88
 - chipIdLo (ConfigROM attribute), 20
 - chipIdLo (PyCapture2.ConfigROM attribute), 88
 - close() (PyCapture2.AVIRecorder method), 55, 68
 - COLOR_PROCESSING (class in PyCapture2), 14, 82
 - compression (PyCapture2.TIFFOption attribute), 113
 - compression (TIFFOption attribute), 33
 - compressionLevel (PNGOption attribute), 32
 - compressionLevel (PyCapture2.PNGOption attribute), 108
 - Config (class in PyCapture2), 23, 87
 - ConfigROM (class in PyCapture2), 20, 88
 - connect() (PyCapture2.BaseCamera method), 39, 71
 - convert() (PyCapture2.Image method), 100
 - cpuDescription (PyCapture2.SystemInfo attribute), 113
 - cpuDescription (SystemInfo attribute), 31
 - cycleCount (PyCapture2.TimeStamp attribute), 114
 - cycleCount (TimeStamp attribute), 27
 - cycleOffset (PyCapture2.TimeStamp attribute), 114
 - cycleOffset (TimeStamp attribute), 27
 - cycleSeconds (PyCapture2.TimeStamp attribute), 114
 - cycleSeconds (TimeStamp attribute), 27
- ## D
- DEFAULT (COLOR_PROCESSING attribute), 14
 - DEFAULT (PyCapture2.COLOR_PROCESSING attribute), 82, 83
 - DEFLATE (PyCapture2.TIFF_COMPRESSION attribute), 113, 114
 - DEFLATE (TIFF_COMPRESSION attribute), 15
 - delay (PyCapture2.StrobeControl attribute), 112
 - delay (StrobeControl attribute), 26
 - deregisterAllEvents() (PyCapture2.BaseCamera method), 39, 71
 - deregisterEvent() (PyCapture2.BaseCamera method), 39, 71
 - destinationIpAddress (GigEStreamChannel attribute), 31
 - destinationIpAddress (PyCapture2.GigEStreamChannel attribute), 98
 - determineBitsPerPixel() (in module PyCapture2), 65, 118
 - DIRECTIONAL (COLOR_PROCESSING attribute), 15
 - DIRECTIONAL (PyCapture2.COLOR_PROCESSING attribute), 83
 - disableAllChannels() (PyCapture2.ImageStatistics method), 57, 101
 - disconnect() (PyCapture2.BaseCamera method), 39, 71
 - discoverGigECameras() (PyCapture2.BusManager method), 35, 79
 - discoverGigEPacketSize() (PyCapture2.GigECamera method), 50, 93
 - doNotFragment (GigEStreamChannel attribute), 31
 - doNotFragment (PyCapture2.GigEStreamChannel attribute), 98
 - DRIVER_TYPE (class in PyCapture2), 8, 89
 - driverList (PyCapture2.SystemInfo attribute), 113

driverList (SystemInfo attribute), 32
driverName (CameraInfo attribute), 21
driverName (PyCapture2.CameraInfo attribute), 86
driverType (CameraInfo attribute), 21
driverType (PyCapture2.CameraInfo attribute), 86
DROP_FRAMES (GRAB_MODE attribute), 9
DROP_FRAMES (PyCapture2.GRAB_MODE attribute), 93
duration (PyCapture2.StrobeControl attribute), 112
duration (StrobeControl attribute), 26

E

EDGE_SENSING (COLOR_PROCESSING attribute), 15
EDGE_SENSING (PyCapture2.COLOR_PROCESSING attribute), 82, 83
EmbeddedImageInfo (class in PyCapture2), 22, 90
enableAllChannels() (PyCapture2.ImageStatistics method), 57, 101
enabled (LUTData attribute), 26
enabled (PyCapture2.LUTData attribute), 103
enableGreyChannel() (PyCapture2.ImageStatistics method), 57, 101
enableHSLChannel() (PyCapture2.ImageStatistics method), 57, 101
enableLUT() (PyCapture2.BaseCamera method), 39, 71
enablePacketResend (GigEConfig attribute), 31
enablePacketResend (PyCapture2.GigEConfig attribute), 97
enableRGBChannel() (PyCapture2.ImageStatistics method), 57, 101
eventID (CallbackData attribute), 32
eventID (PyCapture2.CallbackData attribute), 83
eventName (CallbackData attribute), 32
eventName (PyCapture2.CallbackData attribute), 83
eventTimestamp (CallbackData attribute), 32
eventTimestamp (PyCapture2.CallbackData attribute), 83
exposure (AvailableImageInfo attribute), 22
exposure (EmbeddedImageInfo attribute), 22
exposure (PyCapture2.AvailableImageInfo attribute), 68
exposure (PyCapture2.EmbeddedImageInfo attribute), 90

F

FC2_MODE_0 (MODE attribute), 12
FC2_MODE_0 (PyCapture2.MODE attribute), 104
FC2_MODE_1 (MODE attribute), 12
FC2_MODE_1 (PyCapture2.MODE attribute), 104
FC2_MODE_10 (MODE attribute), 13
FC2_MODE_10 (PyCapture2.MODE attribute), 104
FC2_MODE_11 (MODE attribute), 13
FC2_MODE_11 (PyCapture2.MODE attribute), 104
FC2_MODE_12 (MODE attribute), 13
FC2_MODE_12 (PyCapture2.MODE attribute), 104
FC2_MODE_13 (MODE attribute), 13

FC2_MODE_13 (PyCapture2.MODE attribute), 104
FC2_MODE_14 (MODE attribute), 13
FC2_MODE_14 (PyCapture2.MODE attribute), 104
FC2_MODE_15 (MODE attribute), 13
FC2_MODE_15 (PyCapture2.MODE attribute), 104
FC2_MODE_16 (MODE attribute), 13
FC2_MODE_16 (PyCapture2.MODE attribute), 104
FC2_MODE_17 (MODE attribute), 13
FC2_MODE_17 (PyCapture2.MODE attribute), 104
FC2_MODE_18 (MODE attribute), 13
FC2_MODE_18 (PyCapture2.MODE attribute), 104
FC2_MODE_19 (MODE attribute), 13
FC2_MODE_19 (PyCapture2.MODE attribute), 104
FC2_MODE_2 (MODE attribute), 12
FC2_MODE_2 (PyCapture2.MODE attribute), 104
FC2_MODE_20 (MODE attribute), 13
FC2_MODE_20 (PyCapture2.MODE attribute), 104
FC2_MODE_21 (MODE attribute), 13
FC2_MODE_21 (PyCapture2.MODE attribute), 104
FC2_MODE_22 (MODE attribute), 13
FC2_MODE_22 (PyCapture2.MODE attribute), 104
FC2_MODE_23 (MODE attribute), 13
FC2_MODE_23 (PyCapture2.MODE attribute), 104
FC2_MODE_24 (MODE attribute), 13
FC2_MODE_24 (PyCapture2.MODE attribute), 104
FC2_MODE_25 (MODE attribute), 13
FC2_MODE_25 (PyCapture2.MODE attribute), 104
FC2_MODE_26 (MODE attribute), 13
FC2_MODE_26 (PyCapture2.MODE attribute), 104
FC2_MODE_27 (MODE attribute), 13
FC2_MODE_27 (PyCapture2.MODE attribute), 104
FC2_MODE_28 (MODE attribute), 13
FC2_MODE_28 (PyCapture2.MODE attribute), 104
FC2_MODE_29 (MODE attribute), 13
FC2_MODE_29 (PyCapture2.MODE attribute), 104
FC2_MODE_3 (MODE attribute), 13
FC2_MODE_3 (PyCapture2.MODE attribute), 104
FC2_MODE_30 (MODE attribute), 13
FC2_MODE_30 (PyCapture2.MODE attribute), 104
FC2_MODE_31 (MODE attribute), 13
FC2_MODE_31 (PyCapture2.MODE attribute), 104
FC2_MODE_4 (MODE attribute), 13
FC2_MODE_4 (PyCapture2.MODE attribute), 104
FC2_MODE_5 (MODE attribute), 13
FC2_MODE_5 (PyCapture2.MODE attribute), 104
FC2_MODE_6 (MODE attribute), 13
FC2_MODE_6 (PyCapture2.MODE attribute), 104
FC2_MODE_7 (MODE attribute), 13
FC2_MODE_7 (PyCapture2.MODE attribute), 104
FC2_MODE_8 (MODE attribute), 13
FC2_MODE_8 (PyCapture2.MODE attribute), 104
FC2_MODE_9 (MODE attribute), 13
FC2_MODE_9 (PyCapture2.MODE attribute), 104
FC2_NUM_MODES (MODE attribute), 13

- FC2_NUM_MODES (PyCapture2.MODE attribute), 104
- FC2_PCIE_BUSSPEED_2_5 (PCIE_BUS_SPEED attribute), 14
- FC2_PCIE_BUSSPEED_2_5 (PyCapture2.PCIE_BUS_SPEED attribute), 106
- FC2_PCIE_BUSSPEED_5_0 (PCIE_BUS_SPEED attribute), 14
- FC2_PCIE_BUSSPEED_5_0 (PyCapture2.PCIE_BUS_SPEED attribute), 106
- FC2_PCIE_BUSSPEED_UNKNOWN (PCIE_BUS_SPEED attribute), 14
- FC2_PCIE_BUSSPEED_UNKNOWN (PyCapture2.PCIE_BUS_SPEED attribute), 106
- Fc2error, 91
- Fc2error (class in PyCapture2), 20
- fireBusReset() (PyCapture2.BusManager method), 35, 79
- fireSoftwareTrigger() (PyCapture2.BaseCamera method), 39, 71
- firmwareBuildTime (CameraInfo attribute), 21
- firmwareBuildTime (PyCapture2.CameraInfo attribute), 86
- firmwareVersion (CameraInfo attribute), 21
- firmwareVersion (PyCapture2.CameraInfo attribute), 86
- FOCUS (PROPERTY_TYPE attribute), 10
- FOCUS (PyCapture2.PROPERTY_TYPE attribute), 109, 110
- forceAllIPAddressesAutomatically() (PyCapture2.BusManager method), 35, 80
- forceIPAddressToCamera() (PyCapture2.BusManager method), 35, 80
- FORMAT7 (FRAMERATE attribute), 12
- FORMAT7 (PyCapture2.FRAMERATE attribute), 91
- FORMAT7 (PyCapture2.VIDEO_MODE attribute), 117
- FORMAT7 (VIDEO_MODE attribute), 12
- Format7ImageSettings (class in PyCapture2), 28, 91
- Format7Info (class in PyCapture2), 28, 91
- Format7PacketInfo (class in PyCapture2), 29, 92
- FR_120 (FRAMERATE attribute), 12
- FR_120 (PyCapture2.FRAMERATE attribute), 91
- FR_15 (FRAMERATE attribute), 12
- FR_15 (PyCapture2.FRAMERATE attribute), 90, 91
- FR_1_875 (FRAMERATE attribute), 12
- FR_1_875 (PyCapture2.FRAMERATE attribute), 90, 91
- FR_240 (FRAMERATE attribute), 12
- FR_240 (PyCapture2.FRAMERATE attribute), 91
- FR_30 (FRAMERATE attribute), 12
- FR_30 (PyCapture2.FRAMERATE attribute), 90, 91
- FR_3_75 (FRAMERATE attribute), 12
- FR_3_75 (PyCapture2.FRAMERATE attribute), 90, 91
- FR_60 (FRAMERATE attribute), 12
- FR_60 (PyCapture2.FRAMERATE attribute), 91
- FR_7_5 (FRAMERATE attribute), 12
- FR_7_5 (PyCapture2.FRAMERATE attribute), 90, 91
- FRAME_RATE (PROPERTY_TYPE attribute), 10
- FRAME_RATE (PyCapture2.PROPERTY_TYPE attribute), 109, 110
- frameCounter (AvailableImageInfo attribute), 22
- frameCounter (EmbeddedImageInfo attribute), 22
- frameCounter (PyCapture2.AvailableImageInfo attribute), 68
- frameCounter (PyCapture2.EmbeddedImageInfo attribute), 90
- FRAMERATE (class in PyCapture2), 12, 90
- FROM_FILE_EXT (IMAGE_FILE_FORMAT attribute), 6
- FROM_FILE_EXT (PyCapture2.IMAGE_FILE_FORMAT attribute), 98, 99
- ## G
- gain (AvailableImageInfo attribute), 22
- gain (EmbeddedImageInfo attribute), 22
- GAIN (PROPERTY_TYPE attribute), 10
- gain (PyCapture2.AvailableImageInfo attribute), 68
- gain (PyCapture2.EmbeddedImageInfo attribute), 90
- GAIN (PyCapture2.PROPERTY_TYPE attribute), 109, 110
- GAMMA (PROPERTY_TYPE attribute), 10
- GAMMA (PyCapture2.PROPERTY_TYPE attribute), 109, 110
- GBRG (BAYER_FORMAT attribute), 14
- GBRG (PyCapture2.BAYER_FORMAT attribute), 69
- getActiveLUTBank() (PyCapture2.BaseCamera method), 39, 71
- getBayerTileFormat() (PyCapture2.Image method), 100
- getCameraFromIndex() (PyCapture2.BusManager method), 36, 80
- getCameraFromIPAddress() (PyCapture2.BusManager method), 35, 80
- getCameraFromSerialNumber() (PyCapture2.BusManager method), 36, 80
- getCameraInfo() (PyCapture2.BaseCamera method), 40, 71
- getCameraSerialNumberFromIndex() (PyCapture2.BusManager method), 36, 80
- getChannelStatus() (PyCapture2.ImageStatistics method), 57, 101
- getChild() (PyCapture2.TopologyNode method), 61, 114
- getCols() (PyCapture2.Image method), 100
- getConfiguration() (PyCapture2.BaseCamera method), 40, 71
- getCycleTime() (PyCapture2.BaseCamera method), 40, 72
- getData() (PyCapture2.Image method), 100
- getDataSize() (PyCapture2.Image method), 100
- getDefaultColorProcessing() (in module PyCapture2), 64, 118

`getDefaultOutputFormat()` (in module `PyCapture2`), 64, 118

`getDeviceFromIndex()` (`PyCapture2.BusManager` method), 36, 80

`getDeviceID()` (`PyCapture2.TopologyNode` method), 61, 114

`getDriverDeviceName()` (in module `PyCapture2`), 65, 118

`getEmbeddedImageInfo()` (`PyCapture2.BaseCamera` method), 40, 72

`getFormat7Configuration()` (`PyCapture2.Camera` method), 47, 83

`getFormat7Info()` (`PyCapture2.Camera` method), 48, 83

`getGigEConfig()` (`PyCapture2.GigECamera` method), 50, 93

`getGigEImageBinningSettings()` (`PyCapture2.GigECamera` method), 50, 93

`getGigEImageSettings()` (`PyCapture2.GigECamera` method), 50, 93

`getGigEImageSettingsInfo()` (`PyCapture2.GigECamera` method), 51, 94

`getGigEImagingMode()` (`PyCapture2.GigECamera` method), 51, 94

`getGigEProperty()` (`PyCapture2.GigECamera` method), 51, 94

`getGigEStreamChannelInfo()` (`PyCapture2.GigECamera` method), 51, 94

`getGPIOPinDirection()` (`PyCapture2.BaseCamera` method), 40, 72

`getGuid()` (`PyCapture2.TopologyNode` method), 61, 115

`getHistogram()` (`PyCapture2.ImageStatistics` method), 57, 101

`getInterfaceType()` (`PyCapture2.TopologyNode` method), 61, 115

`getInterfaceTypeFromGuid()` (`PyCapture2.BusManager` method), 36, 81

`getLibraryVersion()` (in module `PyCapture2`), 63, 118

`getLUTBankInfo()` (`PyCapture2.BaseCamera` method), 40, 72

`getLUTChannel()` (`PyCapture2.BaseCamera` method), 40, 72

`getLUTInfo()` (`PyCapture2.BaseCamera` method), 41, 72

`getMean()` (`PyCapture2.ImageStatistics` method), 57, 102

`getMemoryChannel()` (`PyCapture2.BaseCamera` method), 41, 72

`getMemoryChannelInfo()` (`PyCapture2.BaseCamera` method), 41, 73

`getNodeType()` (`PyCapture2.TopologyNode` method), 62, 115

`getNumChildren()` (`PyCapture2.TopologyNode` method), 62, 115

`getNumOfCameras()` (`PyCapture2.BusManager` method), 36, 81

`getNumOfDevices()` (`PyCapture2.BusManager` method), 36, 81

`getNumPixelValues()` (`PyCapture2.ImageStatistics` method), 58, 102

`getNumPorts()` (`PyCapture2.TopologyNode` method), 62, 115

`getNumStreamChannels()` (`PyCapture2.GigECamera` method), 51, 94

`getPixelFormat()` (`PyCapture2.Image` method), 100

`getPixelValueRange()` (`PyCapture2.ImageStatistics` method), 58, 102

`getPortType()` (`PyCapture2.TopologyNode` method), 62, 115

`getProperty()` (`PyCapture2.BaseCamera` method), 41, 73

`getPropertyInfo()` (`PyCapture2.BaseCamera` method), 41, 73

`getRange()` (`PyCapture2.ImageStatistics` method), 58, 102

`getRecievedDataSize()` (`PyCapture2.Image` method), 100

`getRegisterString()` (in module `PyCapture2`), 63, 118

`getRows()` (`PyCapture2.Image` method), 100

`getStatistics()` (`PyCapture2.ImageStatistics` method), 58, 102

`getStats()` (`PyCapture2.BaseCamera` method), 41, 73

`getStride()` (`PyCapture2.Image` method), 100

`getStrobe()` (`PyCapture2.BaseCamera` method), 41, 73

`getStrobeInfo()` (`PyCapture2.BaseCamera` method), 41, 73

`getSystemInfo()` (in module `PyCapture2`), 65, 118

`getTimeStamp()` (`PyCapture2.Image` method), 101

`getTopology()` (`PyCapture2.BusManager` method), 36, 81

`getTriggerDelay()` (`PyCapture2.BaseCamera` method), 42, 73

`getTriggerDelayInfo()` (`PyCapture2.BaseCamera` method), 42, 73

`getTriggerMode()` (`PyCapture2.BaseCamera` method), 42, 74

`getUsbLinkInfo()` (`PyCapture2.BusManager` method), 37, 81

`getUsbPortStatus()` (`PyCapture2.BusManager` method), 37, 81

`getVideoModeAndFrameRate()` (`PyCapture2.Camera` method), 48, 84

`getVideoModeAndFrameRateInfo()` (`PyCapture2.Camera` method), 48, 84

`GIGE` (`INTERFACE_TYPE` attribute), 7

`GIGE` (`PyCapture2.INTERFACE_TYPE` attribute), 99

`GIGE_FILTER` (`DRIVER_TYPE` attribute), 8

`GIGE_FILTER` (`PyCapture2.DRIVER_TYPE` attribute), 89

`GIGE_HEARTBEAT` (`GIGE_PROPERTY_TYPE` attribute), 13

`GIGE_HEARTBEAT` (`PyCapture2.GIGE_PROPERTY_TYPE` attribute), 92

`GIGE_HEARTBEAT_TIMEOUT` (`GIGE_PROPERTY_TYPE` attribute), 14

- GIGE_HEARTBEAT_TIMEOUT (PyCapture2.GIGE_PROPERTY_TYPE attribute), 92, 93
- GIGE_LWF (DRIVER_TYPE attribute), 8
- GIGE_LWF (PyCapture2.DRIVER_TYPE attribute), 89
- GIGE_NONE (DRIVER_TYPE attribute), 8
- GIGE_NONE (PyCapture2.DRIVER_TYPE attribute), 89
- GIGE_PACKET_DELAY (GIGE_PROPERTY_TYPE attribute), 14
- GIGE_PACKET_DELAY (PyCapture2.GIGE_PROPERTY_TYPE attribute), 92, 93
- GIGE_PACKET_SIZE (GIGE_PROPERTY_TYPE attribute), 14
- GIGE_PACKET_SIZE (PyCapture2.GIGE_PROPERTY_TYPE attribute), 92, 93
- GIGE_PRO (DRIVER_TYPE attribute), 8
- GIGE_PRO (PyCapture2.DRIVER_TYPE attribute), 89
- GIGE_PROPERTY_TYPE (class in PyCapture2), 13, 92
- GigECamera (class in PyCapture2), 50, 93
- GigEConfig (class in PyCapture2), 31, 97
- GigEImageSettings (class in PyCapture2), 30, 97
- GigEImageSettingsInfo (class in PyCapture2), 30, 97
- GigEProperty (class in PyCapture2), 29, 98
- GigEStreamChannel (class in PyCapture2), 30, 98
- GPIOPinState (AvailableImageInfo attribute), 22
- GPIOPinState (EmbeddedImageInfo attribute), 22
- GPIOPinState (PyCapture2.AvailableImageInfo attribute), 68
- GPIOPinState (PyCapture2.EmbeddedImageInfo attribute), 90
- gpuDescription (PyCapture2.SystemInfo attribute), 113
- gpuDescription (SystemInfo attribute), 32
- GRAB_MODE (class in PyCapture2), 9, 93
- grabMode (Config attribute), 23
- grabMode (PyCapture2.Config attribute), 88
- grabTimeout (Config attribute), 23
- grabTimeout (PyCapture2.Config attribute), 88
- GRBG (BAYER_FORMAT attribute), 14
- GRBG (PyCapture2.BAYER_FORMAT attribute), 69
- GREEN (PyCapture2.STATISTICS_CHANNEL attribute), 111
- GREEN (STATISTICS_CHANNEL attribute), 15
- GREY (PyCapture2.STATISTICS_CHANNEL attribute), 111, 112
- GREY (STATISTICS_CHANNEL attribute), 15
- ## H
- H264Open() (PyCapture2.AVIRecorder method), 55, 67
- height (Format7ImageSettings attribute), 29
- height (GigEImageSettings attribute), 30
- height (PyCapture2.Format7ImageSettings attribute), 91
- height (PyCapture2.GigEImageSettings attribute), 97
- highPerformanceRetrieveBuffer (Config attribute), 23
- highPerformanceRetrieveBuffer (PyCapture2.Config attribute), 88
- hostPort (GigEStreamChannel attribute), 31
- hostPort (PyCapture2.GigEStreamChannel attribute), 98
- HQ_LINEAR (COLOR_PROCESSING attribute), 15
- HQ_LINEAR (PyCapture2.COLOR_PROCESSING attribute), 83
- HUE (PROPERTY_TYPE attribute), 10
- HUE (PyCapture2.PROPERTY_TYPE attribute), 109, 110
- HUE (PyCapture2.STATISTICS_CHANNEL attribute), 111, 112
- HUE (STATISTICS_CHANNEL attribute), 15
- ## I
- IEEE1394 (INTERFACE_TYPE attribute), 7
- IEEE1394 (PyCapture2.INTERFACE_TYPE attribute), 99
- Image (class in PyCapture2), 100
- IMAGE_FILE_FORMAT (class in PyCapture2), 6, 98
- imageCorrupt (CameraStats attribute), 27
- imageCorrupt (PyCapture2.CameraStats attribute), 87
- imageDriverDropped (CameraStats attribute), 27
- imageDriverDropped (PyCapture2.CameraStats attribute), 87
- imageDropped (CameraStats attribute), 27
- imageDropped (PyCapture2.CameraStats attribute), 87
- imageHStepSize (Format7Info attribute), 28
- imageHStepSize (GigEImageSettingsInfo attribute), 30
- imageHStepSize (PyCapture2.Format7Info attribute), 92
- imageHStepSize (PyCapture2.GigEImageSettingsInfo attribute), 97
- ImageStatistics (class in PyCapture2), 57, 101
- imageVStepSize (Format7Info attribute), 28
- imageVStepSize (GigEImageSettingsInfo attribute), 30
- imageVStepSize (PyCapture2.Format7Info attribute), 92
- imageVStepSize (PyCapture2.GigEImageSettingsInfo attribute), 97
- imageXmitFailed (CameraStats attribute), 27
- imageXmitFailed (PyCapture2.CameraStats attribute), 87
- indexedColor_8bit (BMPOption attribute), 33
- indexedColor_8bit (PyCapture2.BMPOption attribute), 69
- inputBitDepth (LUTData attribute), 26
- inputBitDepth (PyCapture2.LUTData attribute), 103
- INTERFACE_TYPE (class in PyCapture2), 7, 99
- interfaceType (CameraInfo attribute), 21
- interfaceType (PyCapture2.CameraInfo attribute), 86
- interlaced (PNGOption attribute), 32
- interlaced (PyCapture2.PNGOption attribute), 108
- interPacketDelay (GigEStreamChannel attribute), 31

interPacketDelay (PyCapture2.GigEStreamChannel attribute), 98
IPP (COLOR_PROCESSING attribute), 15
IPP (PyCapture2.COLOR_PROCESSING attribute), 83
IRIS (PROPERTY_TYPE attribute), 10
IRIS (PyCapture2.PROPERTY_TYPE attribute), 109, 110
isCameraControllable() (PyCapture2.BusManager method), 37, 81
isColorCamera (CameraInfo attribute), 21
isColorCamera (PyCapture2.CameraInfo attribute), 86
isConnected (PyCapture2.BaseCamera attribute), 42, 74
isochBusSpeed (Config attribute), 23
isochBusSpeed (PyCapture2.Config attribute), 88
isReadable (GigEProperty attribute), 29
isReadable (PyCapture2.GigEProperty attribute), 98
isWritable (GigEProperty attribute), 29
isWritable (PyCapture2.GigEProperty attribute), 98

J

JPEG (IMAGE_FILE_FORMAT attribute), 6
JPEG (PyCapture2.IMAGE_FILE_FORMAT attribute), 99
JPEG (PyCapture2.TIFF_COMPRESSION attribute), 113, 114
JPEG (TIFF_COMPRESSION attribute), 16
JPEG2000 (IMAGE_FILE_FORMAT attribute), 6
JPEG2000 (PyCapture2.IMAGE_FILE_FORMAT attribute), 99
JPEGOOption (class in PyCapture2), 33, 103
JPG2Option (class in PyCapture2), 33, 103
JUJU_1394 (DRIVER_TYPE attribute), 8
JUJU_1394 (PyCapture2.DRIVER_TYPE attribute), 89

L

launchBrowser() (in module PyCapture2), 65, 118
launchCommand() (in module PyCapture2), 66, 118
launchCommandAsync() (in module PyCapture2), 66, 119
launchHelp() (in module PyCapture2), 66, 119
libraryList (PyCapture2.SystemInfo attribute), 113
libraryList (SystemInfo attribute), 32
LIGHTNESS (PyCapture2.STATISTICS_CHANNEL attribute), 111, 112
LIGHTNESS (STATISTICS_CHANNEL attribute), 15
LINUX_X64 (OS_TYPE attribute), 16
LINUX_X64 (PyCapture2.OS_TYPE attribute), 106
LINUX_X86 (OS_TYPE attribute), 16
LINUX_X86 (PyCapture2.OS_TYPE attribute), 106
LITTLE_ENDIAN (BYTE_ORDER attribute), 16
LITTLE_ENDIAN (PyCapture2.BYTE_ORDER attribute), 71
LUTData (class in PyCapture2), 26, 103

LZW (PyCapture2.TIFF_COMPRESSION attribute), 113
LZW (TIFF_COMPRESSION attribute), 16

M

MAC (OS_TYPE attribute), 16
MAC (PyCapture2.OS_TYPE attribute), 106
manualSupported (PropertyInfo attribute), 24
manualSupported (PyCapture2.PropertyInfo attribute), 111
max (GigEProperty attribute), 29
max (PropertyInfo attribute), 24
max (PyCapture2.GigEProperty attribute), 98
max (PyCapture2.PropertyInfo attribute), 111
maxBytesPerPacket (Format7PacketInfo attribute), 29
maxBytesPerPacket (PyCapture2.Format7PacketInfo attribute), 92
maxHeight (Format7Info attribute), 28
maxHeight (GigEImageSettingsInfo attribute), 30
maxHeight (PyCapture2.Format7Info attribute), 92
maxHeight (PyCapture2.GigEImageSettingsInfo attribute), 97
maximumBusSpeed (CameraInfo attribute), 21
maximumBusSpeed (PyCapture2.CameraInfo attribute), 86
maxPacketSize (Format7Info attribute), 28
maxPacketSize (PyCapture2.Format7Info attribute), 92
maxValue (PyCapture2.StrobeInfo attribute), 112
maxValue (StrobeInfo attribute), 25
maxWidth (Format7Info attribute), 28
maxWidth (GigEImageSettingsInfo attribute), 30
maxWidth (PyCapture2.Format7Info attribute), 92
maxWidth (PyCapture2.GigEImageSettingsInfo attribute), 97
microSeconds (PyCapture2.TimeStamp attribute), 114
microSeconds (TimeStamp attribute), 26
min (GigEProperty attribute), 29
min (PropertyInfo attribute), 24
min (PyCapture2.GigEProperty attribute), 98
min (PyCapture2.PropertyInfo attribute), 111
minBytesPerPacket (Format7PacketInfo attribute), 29
minBytesPerPacket (PyCapture2.Format7PacketInfo attribute), 92
minNumImageNotifications (Config attribute), 23
minNumImageNotifications (PyCapture2.Config attribute), 88
minPacketSize (Format7Info attribute), 28
minPacketSize (PyCapture2.Format7Info attribute), 92
minValue (PyCapture2.StrobeInfo attribute), 112
minValue (StrobeInfo attribute), 25
MJPGOpen() (PyCapture2.AVIRecorder method), 55, 67
MODE (class in PyCapture2), 12, 103
mode (Format7ImageSettings attribute), 28
mode (Format7Info attribute), 28

- mode (PyCapture2.Format7ImageSettings attribute), 91
- mode (PyCapture2.Format7Info attribute), 91
- mode (PyCapture2.TriggerMode attribute), 115
- mode (TriggerMode attribute), 24
- MODE_0 (PyCapture2.MODE attribute), 104
- MODE_1 (PyCapture2.MODE attribute), 104
- MODE_10 (PyCapture2.MODE attribute), 104
- MODE_11 (PyCapture2.MODE attribute), 105
- MODE_12 (PyCapture2.MODE attribute), 105
- MODE_13 (PyCapture2.MODE attribute), 105
- MODE_14 (PyCapture2.MODE attribute), 105
- MODE_15 (PyCapture2.MODE attribute), 105
- MODE_16 (PyCapture2.MODE attribute), 105
- MODE_17 (PyCapture2.MODE attribute), 105
- MODE_18 (PyCapture2.MODE attribute), 105
- MODE_19 (PyCapture2.MODE attribute), 105
- MODE_2 (PyCapture2.MODE attribute), 105
- MODE_20 (PyCapture2.MODE attribute), 105
- MODE_21 (PyCapture2.MODE attribute), 105
- MODE_22 (PyCapture2.MODE attribute), 105
- MODE_23 (PyCapture2.MODE attribute), 105
- MODE_24 (PyCapture2.MODE attribute), 105
- MODE_25 (PyCapture2.MODE attribute), 105
- MODE_26 (PyCapture2.MODE attribute), 105
- MODE_27 (PyCapture2.MODE attribute), 105
- MODE_28 (PyCapture2.MODE attribute), 105
- MODE_29 (PyCapture2.MODE attribute), 105
- MODE_3 (PyCapture2.MODE attribute), 105
- MODE_30 (PyCapture2.MODE attribute), 105
- MODE_31 (PyCapture2.MODE attribute), 105
- MODE_4 (PyCapture2.MODE attribute), 105
- MODE_5 (PyCapture2.MODE attribute), 105
- MODE_6 (PyCapture2.MODE attribute), 105
- MODE_7 (PyCapture2.MODE attribute), 105
- MODE_8 (PyCapture2.MODE attribute), 105
- MODE_9 (PyCapture2.MODE attribute), 105
- modelName (CameraInfo attribute), 21
- modelName (PyCapture2.CameraInfo attribute), 86
- MONO12 (PIXEL_FORMAT attribute), 7
- MONO12 (PyCapture2.PIXEL_FORMAT attribute), 107, 108
- MONO16 (PIXEL_FORMAT attribute), 6
- MONO16 (PyCapture2.PIXEL_FORMAT attribute), 107, 108
- MONO8 (PIXEL_FORMAT attribute), 6
- MONO8 (PyCapture2.PIXEL_FORMAT attribute), 107, 108
- N**
- NEAREST_NEIGHBOR_FAST (COLOR_PROCESSING attribute), 14
- NEAREST_NEIGHBOR_FAST (PyCapture2.COLOR_PROCESSING attribute), 82, 83
- networkInterfaceIndex (GigEStreamChannel attribute), 31
- networkInterfaceIndex (PyCapture2.GigEStreamChannel attribute), 98
- NO_COLOR_PROCESSING (COLOR_PROCESSING attribute), 14
- NO_COLOR_PROCESSING (PyCapture2.COLOR_PROCESSING attribute), 82, 83
- NODE_BUS (NODE_TYPE attribute), 16
- NODE_BUS (PyCapture2.NODE_TYPE attribute), 105, 106
- NODE_CAMERA (NODE_TYPE attribute), 16
- NODE_CAMERA (PyCapture2.NODE_TYPE attribute), 106
- NODE_COMPUTER (NODE_TYPE attribute), 16
- NODE_COMPUTER (PyCapture2.NODE_TYPE attribute), 105, 106
- NODE_NODE (NODE_TYPE attribute), 17
- NODE_NODE (PyCapture2.NODE_TYPE attribute), 106
- NODE_TYPE (class in PyCapture2), 16, 105
- nodeNumber (CameraInfo attribute), 21
- nodeNumber (PyCapture2.CameraInfo attribute), 87
- nodeVendorId (ConfigROM attribute), 20
- nodeVendorId (PyCapture2.ConfigROM attribute), 88
- NONE (BAYER_FORMAT attribute), 14
- NONE (PyCapture2.BAYER_FORMAT attribute), 69
- NONE (PyCapture2.TIFF_COMPRESSION attribute), 113, 114
- NONE (TIFF_COMPRESSION attribute), 15
- NUM_MODES (PyCapture2.MODE attribute), 105
- NUM_PIXEL_FORMATS (PIXEL_FORMAT attribute), 7
- NUM_PIXEL_FORMATS (PyCapture2.PIXEL_FORMAT attribute), 108
- NUM_VIDEOMODES (PyCapture2.VIDEO_MODE attribute), 117
- NUM_VIDEOMODES (VIDEO_MODE attribute), 12
- numBanks (LUTData attribute), 26
- numBanks (PyCapture2.LUTData attribute), 103
- numBuffers (Config attribute), 23
- numBuffers (PyCapture2.Config attribute), 87
- numChannels (LUTData attribute), 26
- numChannels (PyCapture2.LUTData attribute), 103
- numCpuCores (PyCapture2.SystemInfo attribute), 113
- numCpuCores (SystemInfo attribute), 31
- numEntries (LUTData attribute), 26
- numEntries (PyCapture2.LUTData attribute), 103
- numImageNotifications (Config attribute), 23
- numImageNotifications (PyCapture2.Config attribute), 88
- numResendPacketsReceived (CameraStats attribute), 27
- numResendPacketsReceived (PyCapture2.CameraStats attribute), 87

numResendPacketsRequested (CameraStats attribute), 27
numResendPacketsRequested (PyCapture2.CameraStats attribute), 87

O

OFF (BANDWIDTH_ALLOCATION attribute), 8
OFF (PyCapture2.BANDWIDTH_ALLOCATION attribute), 68
offsetHStepSize (Format7Info attribute), 28
offsetHStepSize (GigEImageSettingsInfo attribute), 30
offsetHStepSize (PyCapture2.Format7Info attribute), 92
offsetHStepSize (PyCapture2.GigEImageSettingsInfo attribute), 97
offsetVStepSize (Format7Info attribute), 28
offsetVStepSize (GigEImageSettingsInfo attribute), 30
offsetVStepSize (PyCapture2.Format7Info attribute), 92
offsetVStepSize (PyCapture2.GigEImageSettingsInfo attribute), 97
offsetX (Format7ImageSettings attribute), 28
offsetX (GigEImageSettings attribute), 30
offsetX (PyCapture2.Format7ImageSettings attribute), 91
offsetX (PyCapture2.GigEImageSettings attribute), 97
offsetY (Format7ImageSettings attribute), 29
offsetY (GigEImageSettings attribute), 30
offsetY (PyCapture2.Format7ImageSettings attribute), 91
offsetY (PyCapture2.GigEImageSettings attribute), 97
ON (BANDWIDTH_ALLOCATION attribute), 8
ON (PyCapture2.BANDWIDTH_ALLOCATION attribute), 68
onePush (Property attribute), 25
onePush (PyCapture2.Property attribute), 110
onePushSupported (PropertyInfo attribute), 24
onePushSupported (PyCapture2.PropertyInfo attribute), 111
onOff (Property attribute), 25
onOff (PyCapture2.Property attribute), 110
onOff (PyCapture2.StrobeControl attribute), 112
onOff (PyCapture2.TriggerMode attribute), 115
onOff (StrobeControl attribute), 26
onOff (TriggerMode attribute), 23
onOffSupported (PropertyInfo attribute), 24
onOffSupported (PyCapture2.PropertyInfo attribute), 111
onOffSupported (PyCapture2.StrobeInfo attribute), 112
onOffSupported (StrobeInfo attribute), 25
OS_TYPE (class in PyCapture2), 16, 106
osDescription (PyCapture2.SystemInfo attribute), 112
osDescription (SystemInfo attribute), 31
osType (PyCapture2.SystemInfo attribute), 112
osType (SystemInfo attribute), 31
outputBitDepth (LUTData attribute), 26
outputBitDepth (PyCapture2.LUTData attribute), 103

P

PACKBITS (PyCapture2.TIFF_COMPRESSION at-

tribute), 113, 114
PACKBITS (TIFF_COMPRESSION attribute), 15
packetSize (Format7Info attribute), 28
packetSize (GigEStreamChannel attribute), 31
packetSize (PyCapture2.Format7Info attribute), 92
packetSize (PyCapture2.GigEStreamChannel attribute), 98
PAN (PROPERTY_TYPE attribute), 10
PAN (PyCapture2.PROPERTY_TYPE attribute), 109, 110
parameter (PyCapture2.TriggerMode attribute), 115
parameter (TriggerMode attribute), 24
PCIE_2_5 (PyCapture2.PCIE_BUS_SPEED attribute), 106
PCIE_5_0 (PyCapture2.PCIE_BUS_SPEED attribute), 106
PCIE_BUS_SPEED (class in PyCapture2), 14, 106
pcieBusSpeed (CameraInfo attribute), 21
pcieBusSpeed (PyCapture2.CameraInfo attribute), 86
percentage (Format7Info attribute), 28
percentage (PyCapture2.Format7Info attribute), 92
PGM (IMAGE_FILE_FORMAT attribute), 6
PGM (PyCapture2.IMAGE_FILE_FORMAT attribute), 98, 99
PGMOption (class in PyCapture2), 33, 107
PIXEL_FORMAT (class in PyCapture2), 6, 107
PixelFormat (Format7ImageSettings attribute), 29
pixelFormat (GigEImageSettings attribute), 30
PixelFormat (PyCapture2.Format7ImageSettings attribute), 91
pixelFormat (PyCapture2.GigEImageSettings attribute), 97
pixelFormatBitField (Format7Info attribute), 28
pixelFormatBitField (GigEImageSettingsInfo attribute), 30
pixelFormatBitField (PyCapture2.Format7Info attribute), 92
pixelFormatBitField (PyCapture2.GigEImageSettingsInfo attribute), 98
PNG (IMAGE_FILE_FORMAT attribute), 6
PNG (PyCapture2.IMAGE_FILE_FORMAT attribute), 99
PNGOption (class in PyCapture2), 32, 108
polarity (PyCapture2.StrobeControl attribute), 112
polarity (PyCapture2.TriggerMode attribute), 115
polarity (StrobeControl attribute), 26
polarity (TriggerMode attribute), 23
polaritySupported (PyCapture2.StrobeInfo attribute), 112
polaritySupported (StrobeInfo attribute), 25
PORT_CONNECTED_TO_CHILD (PORT_TYPE attribute), 17
PORT_CONNECTED_TO_CHILD (PyCapture2.PORT_TYPE attribute), 109

- PORT_CONNECTED_TO_PARENT (PORT_TYPE attribute), 17
 PORT_CONNECTED_TO_PARENT (PyCapture2.PORT_TYPE attribute), 109
 PORT_NOT_CONNECTED (PORT_TYPE attribute), 17
 PORT_NOT_CONNECTED (PyCapture2.PORT_TYPE attribute), 109
 PORT_TYPE (class in PyCapture2), 17, 108
 portErrors (CameraStats attribute), 27
 portErrors (PyCapture2.CameraStats attribute), 87
 PPM (IMAGE_FILE_FORMAT attribute), 6
 PPM (PyCapture2.IMAGE_FILE_FORMAT attribute), 99
 PPMOption (class in PyCapture2), 32, 109
 present (Property attribute), 25
 present (PropertyInfo attribute), 24
 present (PyCapture2.Property attribute), 110
 present (PyCapture2.PropertyInfo attribute), 111
 present (PyCapture2.StrobeInfo attribute), 112
 present (StrobeInfo attribute), 25
 PRO_1394 (DRIVER_TYPE attribute), 8
 PRO_1394 (PyCapture2.DRIVER_TYPE attribute), 89
 progressive (JPEGOOption attribute), 33
 progressive (PyCapture2.JPEGOOption attribute), 103
 Property (class in PyCapture2), 25, 110
 PROPERTY_TYPE (class in PyCapture2), 10, 109
 PropertyInfo (class in PyCapture2), 24, 110
 propType (GigEProperty attribute), 29
 propType (PyCapture2.GigEProperty attribute), 98
 pszKeyword (ConfigROM attribute), 20
 pszKeyword (PyCapture2.ConfigROM attribute), 89
 PyCapture2 (module), 1, 67
- ## Q
- quality (JPEGOOption attribute), 33
 quality (JPG2Option attribute), 33
 quality (PyCapture2.JPEGOOption attribute), 103
 quality (PyCapture2.JPG2Option attribute), 103
 queryGigEImagingMode() (PyCapture2.GigECamera method), 51, 94
- ## R
- RAW (IMAGE_FILE_FORMAT attribute), 6
 RAW (PyCapture2.IMAGE_FILE_FORMAT attribute), 99
 RAW12 (PIXEL_FORMAT attribute), 7
 RAW12 (PyCapture2.PIXEL_FORMAT attribute), 107, 108
 RAW1394 (DRIVER_TYPE attribute), 8
 RAW1394 (PyCapture2.DRIVER_TYPE attribute), 89
 RAW16 (PIXEL_FORMAT attribute), 7
 RAW16 (PyCapture2.PIXEL_FORMAT attribute), 107, 108
 RAW8 (PIXEL_FORMAT attribute), 7
 RAW8 (PyCapture2.PIXEL_FORMAT attribute), 107, 108
 readGVCPMemory() (PyCapture2.GigECamera method), 51, 94
 readGVCPRegister() (PyCapture2.GigECamera method), 51, 94
 readGVCPRegisterBlock() (PyCapture2.GigECamera method), 52, 95
 readOutSupported (PropertyInfo attribute), 24
 readOutSupported (PyCapture2.PropertyInfo attribute), 111
 readOutSupported (PyCapture2.StrobeInfo attribute), 112
 readOutSupported (StrobeInfo attribute), 25
 readPhyRegister() (PyCapture2.BusManager method), 37, 81
 readRegister() (PyCapture2.BaseCamera method), 42, 74
 readRegisterBlock() (PyCapture2.BaseCamera method), 42, 74
 recommendedBytesPerPacket (Format7PacketInfo attribute), 29
 recommendedBytesPerPacket (PyCapture2.Format7PacketInfo attribute), 92
 RED (PyCapture2.STATISTICS_CHANNEL attribute), 111, 112
 RED (STATISTICS_CHANNEL attribute), 15
 registerAllEvents() (PyCapture2.BaseCamera method), 42, 74
 registerCallback() (PyCapture2.BusManager method), 37, 82
 registerEvent() (PyCapture2.BaseCamera method), 42, 74
 registerTimeout (Config attribute), 23
 registerTimeout (GigEConfig attribute), 31
 registerTimeout (PyCapture2.Config attribute), 88
 registerTimeout (PyCapture2.GigEConfig attribute), 97
 registerTimeoutRetries (Config attribute), 23
 registerTimeoutRetries (GigEConfig attribute), 31
 registerTimeoutRetries (PyCapture2.Config attribute), 88
 registerTimeoutRetries (PyCapture2.GigEConfig attribute), 97
 regReadFailed (CameraStats attribute), 27
 regReadFailed (PyCapture2.CameraStats attribute), 87
 regWriteFailed (CameraStats attribute), 27
 regWriteFailed (PyCapture2.CameraStats attribute), 87
 REMOVAL (BUS_CALLBACK_TYPE attribute), 17
 REMOVAL (PyCapture2.BUS_CALLBACK_TYPE attribute), 69
 rescanBus() (PyCapture2.BusManager method), 37, 82
 restoreFromMemoryChannel() (PyCapture2.BaseCamera method), 43, 74
 retrieveBuffer() (PyCapture2.BaseCamera method), 43, 74
 RGB (PIXEL_FORMAT attribute), 7
 RGB (PyCapture2.PIXEL_FORMAT attribute), 107, 108
 RGB16 (PIXEL_FORMAT attribute), 6

- RGB16 (PyCapture2.PIXEL_FORMAT attribute), 107, 108
- RGB8 (PIXEL_FORMAT attribute), 6
- RGB8 (PyCapture2.PIXEL_FORMAT attribute), 107, 108
- RGBU (PIXEL_FORMAT attribute), 7
- RGBU (PyCapture2.PIXEL_FORMAT attribute), 107, 108
- RGBB (BAYER_FORMAT attribute), 14
- RGBB (PyCapture2.BAYER_FORMAT attribute), 69
- RIGOROUS (COLOR_PROCESSING attribute), 15
- RIGOROUS (PyCapture2.COLOR_PROCESSING attribute), 83
- ROIPosition (AvailableImageInfo attribute), 22
- ROIPosition (EmbeddedImageInfo attribute), 22
- ROIPosition (PyCapture2.AvailableImageInfo attribute), 68
- ROIPosition (PyCapture2.EmbeddedImageInfo attribute), 90
- ## S
- S100 (BUS_SPEED attribute), 9
- S100 (PyCapture2.BUS_SPEED attribute), 69, 70
- S1600 (BUS_SPEED attribute), 9
- S1600 (PyCapture2.BUS_SPEED attribute), 70
- S200 (BUS_SPEED attribute), 9
- S200 (PyCapture2.BUS_SPEED attribute), 69, 70
- S3200 (BUS_SPEED attribute), 9
- S3200 (PyCapture2.BUS_SPEED attribute), 70
- S400 (BUS_SPEED attribute), 9
- S400 (PyCapture2.BUS_SPEED attribute), 69, 70
- S480 (BUS_SPEED attribute), 9
- S480 (PyCapture2.BUS_SPEED attribute), 70
- S5000 (BUS_SPEED attribute), 9
- S5000 (PyCapture2.BUS_SPEED attribute), 70
- S800 (BUS_SPEED attribute), 9
- S800 (PyCapture2.BUS_SPEED attribute), 70
- S_FASTEST (BUS_SPEED attribute), 10
- S_FASTEST (PyCapture2.BUS_SPEED attribute), 70
- S_MONO16 (PIXEL_FORMAT attribute), 6
- S_MONO16 (PyCapture2.PIXEL_FORMAT attribute), 107, 108
- S_RGB16 (PIXEL_FORMAT attribute), 6
- S_RGB16 (PyCapture2.PIXEL_FORMAT attribute), 107, 108
- SATURATION (PROPERTY_TYPE attribute), 10
- SATURATION (PyCapture2.PROPERTY_TYPE attribute), 109, 110
- SATURATION (PyCapture2.STATISTICS_CHANNEL attribute), 111, 112
- SATURATION (STATISTICS_CHANNEL attribute), 15
- save() (PyCapture2.Image method), 101
- saveToMemoryChannel() (PyCapture2.BaseCamera method), 43, 75
- screenHeight (PyCapture2.SystemInfo attribute), 113
- screenHeight (SystemInfo attribute), 32
- screenWidth (PyCapture2.SystemInfo attribute), 113
- screenWidth (SystemInfo attribute), 32
- seconds (PyCapture2.TimeStamp attribute), 114
- seconds (TimeStamp attribute), 26
- sensorInfo (CameraInfo attribute), 21
- sensorInfo (PyCapture2.CameraInfo attribute), 86
- sensorResolution (CameraInfo attribute), 21
- sensorResolution (PyCapture2.CameraInfo attribute), 86
- serialNumber (CameraInfo attribute), 21
- serialNumber (PyCapture2.CameraInfo attribute), 86
- setActiveLUTBank() (PyCapture2.BaseCamera method), 43, 75
- setCallback() (PyCapture2.BaseCamera method), 43, 75
- setChannelStatus() (PyCapture2.ImageStatistics method), 58, 103
- setConfiguration() (PyCapture2.BaseCamera method), 43, 75
- setDefaultColorProcessing() (in module PyCapture2), 64, 119
- setDefaultOutputFormat() (in module PyCapture2), 64, 119
- setEmbeddedImageInfo() (PyCapture2.BaseCamera method), 44, 75
- setFormat7Configuration() (PyCapture2.Camera method), 48, 84
- setFormat7ConfigurationPacket() (PyCapture2.Camera method), 49, 85
- setGigEConfig() (PyCapture2.GigECamera method), 52, 95
- setGigEImageBinningSettings() (PyCapture2.GigECamera method), 52, 95
- setGigEImageSettings() (PyCapture2.GigECamera method), 52, 95
- setGigEImagingMode() (PyCapture2.GigECamera method), 53, 96
- setGigEProperty() (PyCapture2.GigECamera method), 53, 96
- setGPIOPinDirection() (PyCapture2.BaseCamera method), 44, 76
- setLUTChannel() (PyCapture2.BaseCamera method), 44, 76
- setProperty() (PyCapture2.BaseCamera method), 45, 76
- setStrobe() (PyCapture2.BaseCamera method), 45, 77
- setTriggerDelay() (PyCapture2.BaseCamera method), 46, 77
- setTriggerMode() (PyCapture2.BaseCamera method), 46, 78
- setVideoModeAndFrameRate() (PyCapture2.Camera method), 49, 85
- SHARPNESS (PROPERTY_TYPE attribute), 10
- SHARPNESS (PyCapture2.PROPERTY_TYPE attribute), 109, 110

shutter (AvailableImageInfo attribute), 22
 shutter (EmbeddedImageInfo attribute), 22
 SHUTTER (PROPERTY_TYPE attribute), 10
 shutter (PyCapture2.AvailableImageInfo attribute), 68
 shutter (PyCapture2.EmbeddedImageInfo attribute), 90
 SHUTTER (PyCapture2.PROPERTY_TYPE attribute), 109, 110
 source (PyCapture2.StrobeControl attribute), 112
 source (PyCapture2.StrobeInfo attribute), 112
 source (PyCapture2.TriggerMode attribute), 115
 source (StrobeControl attribute), 26
 source (StrobeInfo attribute), 25
 source (TriggerMode attribute), 24
 sourcePort (GigEStreamChannel attribute), 31
 sourcePort (PyCapture2.GigEStreamChannel attribute), 98
 SPEED_UNKNOWN (BUS_SPEED attribute), 10
 SPEED_UNKNOWN (PyCapture2.BUS_SPEED attribute), 70
 startCapture() (PyCapture2.BaseCamera method), 47, 78
 startSyncCapture() (in module PyCapture2), 63, 119
 STATISTICS_CHANNEL (class in PyCapture2), 15, 111
 stopCapture() (PyCapture2.BaseCamera method), 47, 79
 StrobeControl (class in PyCapture2), 26, 112
 StrobeInfo (class in PyCapture2), 25, 112
 strobePattern (AvailableImageInfo attribute), 22
 strobePattern (EmbeddedImageInfo attribute), 22
 strobePattern (PyCapture2.AvailableImageInfo attribute), 68
 strobePattern (PyCapture2.EmbeddedImageInfo attribute), 90
 supported (LUTData attribute), 26
 supported (PyCapture2.LUTData attribute), 103
 sysMemSize (PyCapture2.SystemInfo attribute), 113
 sysMemSize (SystemInfo attribute), 31
 SystemInfo (class in PyCapture2), 31, 112

T

temperature (CameraStats attribute), 27
 TEMPERATURE (PROPERTY_TYPE attribute), 10
 temperature (PyCapture2.CameraStats attribute), 87
 TEMPERATURE (PyCapture2.PROPERTY_TYPE attribute), 109, 110
 TIFF (IMAGE_FILE_FORMAT attribute), 6
 TIFF (PyCapture2.IMAGE_FILE_FORMAT attribute), 99
 TIFF_COMPRESSION (class in PyCapture2), 15, 113
 TIFFOption (class in PyCapture2), 33, 113
 TILT (PROPERTY_TYPE attribute), 10
 TILT (PyCapture2.PROPERTY_TYPE attribute), 109, 110
 timeSinceBusReset (CameraStats attribute), 27
 timeSinceBusReset (PyCapture2.CameraStats attribute), 87

timeSinceInitialization (CameraStats attribute), 27
 timeSinceInitialization (PyCapture2.CameraStats attribute), 87
 timestamp (AvailableImageInfo attribute), 21
 timeStamp (CameraStats attribute), 27
 TimeStamp (class in PyCapture2), 26, 114
 timestamp (EmbeddedImageInfo attribute), 22
 timestamp (PyCapture2.AvailableImageInfo attribute), 68
 timeStamp (PyCapture2.CameraStats attribute), 87
 timestamp (PyCapture2.EmbeddedImageInfo attribute), 90
 TopologyNode (class in PyCapture2), 61, 114
 TRIGGER_DELAY (PROPERTY_TYPE attribute), 10
 TRIGGER_DELAY (PyCapture2.PROPERTY_TYPE attribute), 109, 110
 TRIGGER_MODE (PROPERTY_TYPE attribute), 10
 TRIGGER_MODE (PyCapture2.PROPERTY_TYPE attribute), 109, 110
 TriggerMode (class in PyCapture2), 23, 115
 type (Property attribute), 25
 type (PropertyInfo attribute), 24
 type (PyCapture2.Property attribute), 110
 type (PyCapture2.PropertyInfo attribute), 111

U

unitAbbr (PropertyInfo attribute), 24
 unitAbbr (PyCapture2.PropertyInfo attribute), 111
 units (PropertyInfo attribute), 24
 units (PyCapture2.PropertyInfo attribute), 111
 unitSpecID (ConfigROM attribute), 20
 unitSpecID (PyCapture2.ConfigROM attribute), 88
 unitSubSWVer (ConfigROM attribute), 20
 unitSubSWVer (PyCapture2.ConfigROM attribute), 88
 unitSWVer (ConfigROM attribute), 20
 unitSWVer (PyCapture2.ConfigROM attribute), 88
 UNKNOWN (DRIVER_TYPE attribute), 8
 UNKNOWN (INTERFACE_TYPE attribute), 7
 UNKNOWN (PyCapture2.DRIVER_TYPE attribute), 89
 UNKNOWN (PyCapture2.INTERFACE_TYPE attribute), 99
 UNKNOWN (PyCapture2.PCIE_BUS_SPEED attribute), 107
 UNKNOWN_OS (OS_TYPE attribute), 16
 UNKNOWN_OS (PyCapture2.OS_TYPE attribute), 106
 unregisterCallback() (PyCapture2.BusManager method), 38, 82
 unsetCallback() (PyCapture2.BaseCamera method), 47, 79
 UNSPECIFIED (BANDWIDTH_ALLOCATION attribute), 8
 UNSPECIFIED (PyCapture2.BANDWIDTH_ALLOCATION attribute), 68

UNSPECIFIED_GRAB_MODE (PyCapture2.GRAB_MODE attribute), 93	vendorUniqueInfo_1 (PyCapture2.ConfigROM attribute), 88
UNSPECIFIED_PIXEL_FORMAT (PIXEL_FORMAT attribute), 7	vendorUniqueInfo_2 (ConfigROM attribute), 20
UNSPECIFIED_PIXEL_FORMAT (PyCapture2.PIXEL_FORMAT attribute), 108	vendorUniqueInfo_2 (PyCapture2.ConfigROM attribute), 88
UNSPECIFIED_PROPERTY_TYPE (PROPERTY_TYPE attribute), 10	vendorUniqueInfo_3 (ConfigROM attribute), 20
UNSPECIFIED_PROPERTY_TYPE (PyCapture2.PROPERTY_TYPE attribute), 109, 110	vendorUniqueInfo_3 (PyCapture2.ConfigROM attribute), 88
UNSUPPORTED (BANDWIDTH_ALLOCATION attribute), 8	VIDEO1394 (DRIVER_TYPE attribute), 8
UNSUPPORTED (PyCapture2.BANDWIDTH_ALLOCATION attribute), 68	VIDEO1394 (PyCapture2.DRIVER_TYPE attribute), 89, 90
USB3_PRO (DRIVER_TYPE attribute), 8	VIDEO_MODE (class in PyCapture2), 11, 116
USB3_PRO (PyCapture2.DRIVER_TYPE attribute), 89, 90	VM_1024x768RGB (PyCapture2.VIDEO_MODE attribute), 116, 117
USB_2 (INTERFACE_TYPE attribute), 7	VM_1024x768RGB (VIDEO_MODE attribute), 11
USB_2 (PyCapture2.INTERFACE_TYPE attribute), 99	VM_1024x768Y16 (PyCapture2.VIDEO_MODE attribute), 116, 117
USB_3 (INTERFACE_TYPE attribute), 7	VM_1024x768Y16 (VIDEO_MODE attribute), 11
USB_3 (PyCapture2.INTERFACE_TYPE attribute), 99, 100	VM_1024x768Y8 (PyCapture2.VIDEO_MODE attribute), 116, 117
USB_CAM (DRIVER_TYPE attribute), 8	VM_1024x768Y8 (VIDEO_MODE attribute), 11
USB_CAM (PyCapture2.DRIVER_TYPE attribute), 89, 90	VM_1024x768YUV422 (PyCapture2.VIDEO_MODE attribute), 116, 117
USB_NONE (DRIVER_TYPE attribute), 8	VM_1024x768YUV422 (VIDEO_MODE attribute), 11
USB_NONE (PyCapture2.DRIVER_TYPE attribute), 89, 90	VM_1280x960RGB (PyCapture2.VIDEO_MODE attribute), 116, 117
V	VM_1280x960RGB (VIDEO_MODE attribute), 11
validateFormat7Settings() (PyCapture2.Camera method), 49, 85	VM_1280x960Y16 (PyCapture2.VIDEO_MODE attribute), 116, 117
value (GigEProperty attribute), 29	VM_1280x960Y16 (VIDEO_MODE attribute), 11
value (PyCapture2.GigEProperty attribute), 98	VM_1280x960Y8 (PyCapture2.VIDEO_MODE attribute), 116, 117
ValueA (Property attribute), 25	VM_1280x960Y8 (VIDEO_MODE attribute), 11
ValueA (PyCapture2.Property attribute), 110	VM_1280x960YUV422 (PyCapture2.VIDEO_MODE attribute), 116, 117
ValueB (Property attribute), 25	VM_1280x960YUV422 (VIDEO_MODE attribute), 11
ValueB (PyCapture2.Property attribute), 110	VM_1600x1200RGB (PyCapture2.VIDEO_MODE attribute), 117
vendorName (CameraInfo attribute), 21	VM_1600x1200RGB (VIDEO_MODE attribute), 12
vendorName (PyCapture2.CameraInfo attribute), 86	VM_1600x1200Y16 (PyCapture2.VIDEO_MODE attribute), 117
vendorPixelFormatBitField (Format7Info attribute), 28	VM_1600x1200Y16 (VIDEO_MODE attribute), 12
vendorPixelFormatBitField (GigEImageSettingsInfo attribute), 30	VM_1600x1200Y8 (PyCapture2.VIDEO_MODE attribute), 117
vendorPixelFormatBitField (PyCapture2.Format7Info attribute), 92	VM_1600x1200YUV422 (PyCapture2.VIDEO_MODE attribute), 116, 117
vendorPixelFormatBitField (PyCapture2.GigEImageSettingsInfo attribute), 98	VM_1600x1200YUV422 (VIDEO_MODE attribute), 11
vendorUniqueInfo_0 (ConfigROM attribute), 20	VM_160x120YUV444 (PyCapture2.VIDEO_MODE attribute), 116, 117
vendorUniqueInfo_0 (PyCapture2.ConfigROM attribute), 88	VM_160x120YUV444 (VIDEO_MODE attribute), 11
vendorUniqueInfo_1 (ConfigROM attribute), 20	VM_320x240YUV422 (PyCapture2.VIDEO_MODE attribute), 116, 117
	VM_320x240YUV422 (VIDEO_MODE attribute), 11
	VM_640x480RGB (PyCapture2.VIDEO_MODE at-

tribute), 116, 117
 VM_640x480RGB (VIDEO_MODE attribute), 11
 VM_640x480Y16 (PyCapture2.VIDEO_MODE attribute), 116, 117
 VM_640x480Y16 (VIDEO_MODE attribute), 11
 VM_640x480Y8 (PyCapture2.VIDEO_MODE attribute), 116, 117
 VM_640x480Y8 (VIDEO_MODE attribute), 11
 VM_640x480YUV411 (PyCapture2.VIDEO_MODE attribute), 116, 117
 VM_640x480YUV411 (VIDEO_MODE attribute), 11
 VM_640x480YUV422 (PyCapture2.VIDEO_MODE attribute), 116, 117
 VM_640x480YUV422 (VIDEO_MODE attribute), 11
 VM_800x600RGB (PyCapture2.VIDEO_MODE attribute), 116, 117
 VM_800x600RGB (VIDEO_MODE attribute), 11
 VM_800x600Y16 (PyCapture2.VIDEO_MODE attribute), 116, 117
 VM_800x600Y16 (VIDEO_MODE attribute), 11
 VM_800x600Y8 (PyCapture2.VIDEO_MODE attribute), 116, 117
 VM_800x600Y8 (VIDEO_MODE attribute), 11
 VM_800x600YUV422 (PyCapture2.VIDEO_MODE attribute), 116, 117
 VM_800x600YUV422 (VIDEO_MODE attribute), 11
 VM_VM_1600x1200Y8 (PyCapture2.VIDEO_MODE attribute), 117
 VM_VM_1600x1200Y8 (VIDEO_MODE attribute), 12

W

waitForBufferEvent() (PyCapture2.BaseCamera method), 47, 79
 WHITE_BALANCE (PROPERTY_TYPE attribute), 10
 WHITE_BALANCE (PyCapture2.PROPERTY_TYPE attribute), 109, 110
 whiteBalance (AvailableImageInfo attribute), 22
 whiteBalance (EmbeddedImageInfo attribute), 22
 whiteBalance (PyCapture2.AvailableImageInfo attribute), 68
 whiteBalance (PyCapture2.EmbeddedImageInfo attribute), 90
 width (Format7ImageSettings attribute), 29
 width (GigEImageSettings attribute), 30
 width (PyCapture2.Format7ImageSettings attribute), 91
 width (PyCapture2.GigEImageSettings attribute), 97
 WINDOWS_X64 (OS_TYPE attribute), 16
 WINDOWS_X64 (PyCapture2.OS_TYPE attribute), 106
 WINDOWS_X86 (OS_TYPE attribute), 16
 WINDOWS_X86 (PyCapture2.OS_TYPE attribute), 106
 writeGVCPCMemory() (PyCapture2.GigECamera method), 53, 96
 writeGVCPCRegister() (PyCapture2.GigECamera method), 53, 96

writeGVCPCRegisterBlock() (PyCapture2.GigECamera method), 53, 97
 writePhyRegister() (PyCapture2.BusManager method), 38, 82
 writeRegister() (PyCapture2.BaseCamera method), 47, 79
 writeRegisterBlock() (PyCapture2.BaseCamera method), 47, 79

Y

YUV8_411 (PIXEL_FORMAT attribute), 6
 YUV8_411 (PyCapture2.PIXEL_FORMAT attribute), 107, 108
 YUV8_422 (PIXEL_FORMAT attribute), 6
 YUV8_422 (PyCapture2.PIXEL_FORMAT attribute), 107, 108
 YUV8_444 (PyCapture2.PIXEL_FORMAT attribute), 108
 YUV8_JPEG_422 (PIXEL_FORMAT attribute), 7
 YUV8_JPEG_422 (PyCapture2.PIXEL_FORMAT attribute), 108

Z

ZOOM (PROPERTY_TYPE attribute), 10
 ZOOM (PyCapture2.PROPERTY_TYPE attribute), 109, 110