



# POINT GREY

## FlyCapture 2.1

### API Programming Reference

Revised October 18, 2010

#### **Point Grey Research Inc.**

12051 Riverside Way • Richmond, BC • Canada • V6W 1K7 • T (604) 242-9937 •  
[www.ptgrey.com](http://www.ptgrey.com)

## **Software Warranty**

Point Grey Research warrants to the Original Purchaser, for a period of one (1) year from date of purchase that:

1. The diskette on which the Software is furnished and the accompanying documentation are not defective;
2. The Software is properly recorded upon the diskettes enclosed;
3. The documentation is substantially complete and contains all the information Point Grey Research deems necessary to use the Software;
4. The Software functions substantially as described in the documentation.

Point Grey Research, Inc.'s entire liability and the Original Purchaser's exclusive remedy shall be the replacement of any diskette or documentation not meeting these warranties. On such an occasion, a copy of the paid receipt accompanied with the faulty diskette or documentation must be returned to Point Grey Research, Inc. or an authorized dealer.

Point Grey Research, Inc. expressly disclaims and excludes all other warranties, express, implied and statutory, including, but without limitation, warranty of merchantability and fitness for a particular application or purpose. In no event shall Point Grey Research, Inc. be liable to the Original Purchaser or any third party for direct, indirect, incidental, consequential, special or accidental damages, including without limitation damages for business interruption, loss of profits, revenue, data or bodily injury or death.

## Software License Agreement

The FlyCapture<sup>®</sup> Software Development Kit (the "Software") is owned and copyrighted by Point Grey Research, Inc. All rights are reserved. The Original Purchaser is granted a license to use the Software subject to the following restrictions and limitations.

1. The license is to the Original Purchaser only, and is nontransferable unless you have received written permission of Point Grey Research, Inc.
2. The Original Purchaser may use the Software only with Point Grey Research, Inc. cameras owned by the Original Purchaser, including but not limited to, Firefly<sup>®</sup>, Firefly<sup>®</sup>2, Firefly<sup>®</sup> MV, Flea<sup>®</sup>, Scorpion<sup>™</sup>, Dragonfly<sup>®</sup>, Dragonfly<sup>®</sup>2, Dragonfly Express<sup>™</sup>, Grasshopper<sup>™</sup> or Chameleon<sup>™</sup> Camera Modules.
3. The Original Purchaser may make back-up copies of the Software for his or her own use only, subject to the use limitations of this license.
4. Subject to s.5 below, the Original Purchaser may not engage in, nor permit third parties to engage in, any of the following:
  - A. Providing or disclosing the Software to third parties.
  - B. Making alterations or copies of any kind of the Software (except as specifically permitted in s.3 above).
  - C. Attempting to un-assemble, de-compile or reverse engineer the Software in any way.
  - D. Granting sublicenses, leases or other rights in the Software to others.
5. Original Purchasers who are Original Equipment Manufacturers may make Derivative Products with the Software. Derivative Products are new software products developed, in whole or in part, using the Software and other Point Grey Research, Inc. products. Point Grey Research, Inc. hereby grants a license to Original Equipment Manufacturers to incorporate and distribute the libraries found in the Software with the Derivative Products. The components of any Derivative Product that contain the Software libraries may only be used with Point Grey Research, Inc. products, or images derived from such products.
  - 5.1 By the distribution of the Software libraries with Derivative Products, Original Purchasers agree to:
    - A. not permit further redistribution of the Software libraries by end-user customers;
    - B. include a valid copyright notice on any Derivative Product; and
    - C. indemnify, hold harmless, and defend Point Grey Research, Inc. from and against any claims or lawsuits, including attorney's fees, that arise or result from the use or distribution of any Derivative Product.

Point Grey Research, Inc. reserves the right to terminate this license if there are any violations of its terms or if there is a default committed by the Original Purchaser. Upon termination, for any reason, all copies of the Software must be immediately returned to Point Grey Research, Inc. and the Original Purchaser shall be liable to Point Grey Research, Inc. for any and all damages suffered as a result of the violation or default.



# Contents

<b>1</b>	<b>Module Index</b>	<b>1</b>
1.1	Modules . . . . .	1
<b>2</b>	<b>Namespace Index</b>	<b>3</b>
2.1	Namespace List . . . . .	3
<b>3</b>	<b>Class Index</b>	<b>5</b>
3.1	Class Hierarchy . . . . .	5
<b>4</b>	<b>Class Index</b>	<b>7</b>
4.1	Class List . . . . .	7
<b>5</b>	<b>File Index</b>	<b>9</b>
5.1	File List . . . . .	9
<b>6</b>	<b>Module Documentation</b>	<b>11</b>
6.1	Global constants . . . . .	11
6.1.1	Variable Documentation . . . . .	11
6.1.1.1	sk_maxNumPorts . . . . .	11
6.1.1.2	sk_maxStringLength . . . . .	11
6.2	Enumerations . . . . .	12
6.2.1	Enumeration Type Documentation . . . . .	18
6.2.1.1	BandwidthAllocation . . . . .	18
6.2.1.2	BayerTileFormat . . . . .	19
6.2.1.3	BusCallbackType . . . . .	19
6.2.1.4	BusSpeed . . . . .	19
6.2.1.5	ColorProcessingAlgorithm . . . . .	20
6.2.1.6	DriverType . . . . .	20
6.2.1.7	ErrorType . . . . .	21
6.2.1.8	FrameRate . . . . .	22

6.2.1.9	GrabMode	23
6.2.1.10	GrabTimeout	23
6.2.1.11	ImageFileFormat	23
6.2.1.12	InterfaceType	24
6.2.1.13	Mode	24
6.2.1.14	PixelFormat	25
6.2.1.15	PropertyType	26
6.2.1.16	VideoMode	26
6.3	GigE specific enumerations	28
6.3.1	Detailed Description	28
6.3.2	Enumeration Type Documentation	28
6.3.2.1	GigEPropertyType	28
6.4	Structures	29
6.4.1	Typedef Documentation	31
6.4.1.1	TriggerDelay	31
6.4.1.2	TriggerDelayInfo	31
6.5	GigE specific structures	32
6.5.1	Detailed Description	32
6.6	IIDC specific structures	33
6.6.1	Detailed Description	33
6.7	Image saving structures.	34
6.7.1	Detailed Description	34
<b>7</b>	<b>Namespace Documentation</b>	<b>35</b>
7.1	FlyCapture2 Namespace Reference	35
7.1.1	Typedef Documentation	46
7.1.1.1	AsyncCommandCallback	46
7.1.1.2	BusEventCallback	46
7.1.1.3	CallbackHandle	46
7.1.1.4	ImageEventCallback	46
7.1.2	Enumeration Type Documentation	46
7.1.2.1	ByteOrder	46
7.1.2.2	OSType	47
<b>8</b>	<b>Class Documentation</b>	<b>49</b>
8.1	AVIOption Struct Reference	49
8.1.1	Detailed Description	49

8.1.2	Constructor & Destructor Documentation	49
8.1.2.1	AVIOption	49
8.1.3	Member Data Documentation	49
8.1.3.1	frameRate	49
8.1.3.2	reserved	50
8.2	AVIRecorder Class Reference	51
8.2.1	Detailed Description	51
8.2.2	Constructor & Destructor Documentation	51
8.2.2.1	AVIRecorder	51
8.2.2.2	~AVIRecorder	51
8.2.3	Member Function Documentation	51
8.2.3.1	AVIAppend	51
8.2.3.2	AVIClose	52
8.2.3.3	AVIOpen	52
8.3	BusManager Class Reference	53
8.3.1	Detailed Description	54
8.3.2	Constructor & Destructor Documentation	54
8.3.2.1	BusManager	54
8.3.2.2	~BusManager	54
8.3.3	Member Function Documentation	54
8.3.3.1	DiscoverGigECameras	54
8.3.3.2	FireBusReset	55
8.3.3.3	ForceIPAddressToCamera	55
8.3.3.4	GetCameraFromIndex	56
8.3.3.5	GetCameraFromIPAddress	56
8.3.3.6	GetCameraFromSerialNumber	56
8.3.3.7	GetCameraSerialNumberFromIndex	57
8.3.3.8	GetDeviceFromIndex	57
8.3.3.9	GetInterfaceTypeFromGuid	57
8.3.3.10	GetNumOfCameras	58
8.3.3.11	GetNumOfDevices	58
8.3.3.12	GetTopology	58
8.3.3.13	ReadPhyRegister	58
8.3.3.14	RegisterCallback	59
8.3.3.15	RescanBus	59
8.3.3.16	UnregisterCallback	59

8.3.3.17	WritePhyRegister	59
8.4	Camera Class Reference	61
8.4.1	Detailed Description	65
8.4.2	Constructor & Destructor Documentation	65
8.4.2.1	Camera	65
8.4.2.2	~Camera	65
8.4.3	Member Function Documentation	66
8.4.3.1	Connect	66
8.4.3.2	Disconnect	66
8.4.3.3	EnableLUT	66
8.4.3.4	FireSoftwareTrigger	66
8.4.3.5	GetActiveLUTBank	67
8.4.3.6	GetCameraInfo	67
8.4.3.7	GetConfiguration	67
8.4.3.8	GetEmbeddedImageInfo	67
8.4.3.9	GetFormat7Configuration	68
8.4.3.10	GetFormat7Info	68
8.4.3.11	GetGPIOPinDirection	69
8.4.3.12	GetLUTBankInfo	69
8.4.3.13	GetLUTChannel	69
8.4.3.14	GetLUTInfo	70
8.4.3.15	GetMemoryChannel	70
8.4.3.16	GetMemoryChannelInfo	71
8.4.3.17	GetProperty	71
8.4.3.18	GetPropertyInfo	71
8.4.3.19	GetRegisterString	72
8.4.3.20	GetStrobe	72
8.4.3.21	GetStrobeInfo	72
8.4.3.22	GetTriggerDelay	73
8.4.3.23	GetTriggerDelayInfo	73
8.4.3.24	GetTriggerMode	74
8.4.3.25	GetTriggerModeInfo	74
8.4.3.26	GetVideoModeAndFrameRate	74
8.4.3.27	GetVideoModeAndFrameRateInfo	75
8.4.3.28	IsConnected	75
8.4.3.29	ReadRegister	75



8.4.3.30	ReadRegisterBlock	76
8.4.3.31	RestoreFromMemoryChannel	76
8.4.3.32	RetrieveBuffer	77
8.4.3.33	SaveToMemoryChannel	77
8.4.3.34	SetActiveLUTBank	77
8.4.3.35	SetCallback	78
8.4.3.36	SetConfiguration	78
8.4.3.37	SetEmbeddedImageInfo	78
8.4.3.38	SetFormat7Configuration	79
8.4.3.39	SetFormat7Configuration	79
8.4.3.40	SetGPIOPinDirection	79
8.4.3.41	SetLUTChannel	80
8.4.3.42	SetProperty	80
8.4.3.43	SetStrobe	81
8.4.3.44	SetTriggerDelay	81
8.4.3.45	SetTriggerMode	82
8.4.3.46	SetUserBuffers	82
8.4.3.47	SetVideoModeAndFrameRate	82
8.4.3.48	StartCapture	83
8.4.3.49	StartSyncCapture	83
8.4.3.50	StopCapture	83
8.4.3.51	ValidateFormat7Settings	84
8.4.3.52	WaitForBufferEvent	84
8.4.3.53	WriteRegister	84
8.4.3.54	WriteRegisterBlock	85
8.5	CameraBase Class Reference	86
8.5.1	Detailed Description	90
8.5.2	Constructor & Destructor Documentation	90
8.5.2.1	CameraBase	90
8.5.2.2	~CameraBase	90
8.5.3	Member Function Documentation	90
8.5.3.1	Connect	90
8.5.3.2	Disconnect	91
8.5.3.3	EnableLUT	91
8.5.3.4	FireSoftwareTrigger	91
8.5.3.5	GetActiveLUTBank	92

8.5.3.6	GetCameraInfo	92
8.5.3.7	GetConfiguration	92
8.5.3.8	GetEmbeddedImageInfo	92
8.5.3.9	GetGPIOPinDirection	93
8.5.3.10	GetLUTBankInfo	93
8.5.3.11	GetLUTChannel	94
8.5.3.12	GetLUTInfo	94
8.5.3.13	GetMemoryChannel	94
8.5.3.14	GetMemoryChannelInfo	95
8.5.3.15	GetProperty	95
8.5.3.16	GetPropertyInfo	96
8.5.3.17	GetRegisterString	96
8.5.3.18	GetStrobe	96
8.5.3.19	GetStrobeInfo	97
8.5.3.20	GetTriggerDelay	97
8.5.3.21	GetTriggerDelayInfo	97
8.5.3.22	GetTriggerMode	98
8.5.3.23	GetTriggerModeInfo	98
8.5.3.24	IsConnected	98
8.5.3.25	ReadRegister	99
8.5.3.26	ReadRegisterBlock	99
8.5.3.27	RestoreFromMemoryChannel	100
8.5.3.28	RetrieveBuffer	100
8.5.3.29	SaveToMemoryChannel	100
8.5.3.30	SetActiveLUTBank	101
8.5.3.31	SetCallback	101
8.5.3.32	SetConfiguration	101
8.5.3.33	SetEmbeddedImageInfo	102
8.5.3.34	SetGPIOPinDirection	102
8.5.3.35	SetLUTChannel	102
8.5.3.36	SetProperty	103
8.5.3.37	SetStrobe	103
8.5.3.38	SetTriggerDelay	104
8.5.3.39	SetTriggerMode	104
8.5.3.40	SetUserBuffers	105
8.5.3.41	StartCapture	105

8.5.3.42	StartSyncCapture	106
8.5.3.43	StopCapture	106
8.5.3.44	WaitForBufferEvent	106
8.5.3.45	WriteRegister	107
8.5.3.46	WriteRegisterBlock	107
8.5.4	Member Data Documentation	108
8.5.4.1	m_pCameraData	108
8.6	CameraControlDlg Class Reference	109
8.6.1	Detailed Description	109
8.6.2	Constructor & Destructor Documentation	109
8.6.2.1	CameraControlDlg	109
8.6.2.2	~CameraControlDlg	109
8.6.3	Member Function Documentation	110
8.6.3.1	Connect	110
8.6.3.2	Disconnect	110
8.6.3.3	Hide	110
8.6.3.4	IsVisible	110
8.6.3.5	Show	110
8.7	CameraInfo Struct Reference	111
8.7.1	Detailed Description	113
8.7.2	Constructor & Destructor Documentation	113
8.7.2.1	CameraInfo	113
8.7.3	Member Data Documentation	113
8.7.3.1	bayerTileFormat	113
8.7.3.2	configROM	113
8.7.3.3	defaultGateway	113
8.7.3.4	driverName	113
8.7.3.5	driverType	113
8.7.3.6	firmwareBuildTime	113
8.7.3.7	firmwareVersion	113
8.7.3.8	gigEMajorVersion	113
8.7.3.9	gigEMinorVersion	113
8.7.3.10	iidcVer	114
8.7.3.11	interfaceType	114
8.7.3.12	ipAddress	114
8.7.3.13	isColorCamera	114

8.7.3.14	macAddress	114
8.7.3.15	maximumBusSpeed	114
8.7.3.16	modelName	114
8.7.3.17	reserved	114
8.7.3.18	sensorInfo	114
8.7.3.19	sensorResolution	114
8.7.3.20	serialNumber	114
8.7.3.21	subnetMask	115
8.7.3.22	userDefinedName	115
8.7.3.23	vendorName	115
8.7.3.24	xmlURL1	115
8.7.3.25	xmlURL2	115
8.8	CameraSelectionDlg Class Reference	116
8.8.1	Detailed Description	116
8.8.2	Constructor & Destructor Documentation	116
8.8.2.1	CameraSelectionDlg	116
8.8.2.2	~CameraSelectionDlg	116
8.8.3	Member Function Documentation	116
8.8.3.1	ShowModal	116
8.9	CameraStats Struct Reference	117
8.9.1	Detailed Description	118
8.9.2	Constructor & Destructor Documentation	118
8.9.2.1	CameraStats	118
8.9.3	Member Data Documentation	118
8.9.3.1	cameraCurrents	118
8.9.3.2	cameraPowerUp	118
8.9.3.3	cameraVoltages	118
8.9.3.4	imageCorrupt	118
8.9.3.5	imageDriverDropped	118
8.9.3.6	imageDropped	118
8.9.3.7	imageXmitFailed	118
8.9.3.8	numCurrents	118
8.9.3.9	numVoltages	118
8.9.3.10	portErrors	118
8.9.3.11	regReadFailed	118
8.9.3.12	regWriteFailed	118

8.9.3.13	reserved	118
8.9.3.14	temperature	119
8.9.3.15	timeSinceBusReset	119
8.9.3.16	timeSinceInitialization	119
8.9.3.17	timeStamp	119
8.10	ConfigROM Struct Reference	120
8.10.1	Detailed Description	121
8.10.2	Constructor & Destructor Documentation	121
8.10.2.1	ConfigROM	121
8.10.3	Member Data Documentation	121
8.10.3.1	chipIdHi	121
8.10.3.2	chipIdLo	121
8.10.3.3	nodeVendorId	121
8.10.3.4	pszKeyword	121
8.10.3.5	reserved	121
8.10.3.6	unitSpecId	121
8.10.3.7	unitSubSWVer	121
8.10.3.8	unitSWVer	121
8.10.3.9	vendorUniqueInfo_0	121
8.10.3.10	vendorUniqueInfo_1	122
8.10.3.11	vendorUniqueInfo_2	122
8.10.3.12	vendorUniqueInfo_3	122
8.11	DCAMFormats Struct Reference	123
8.11.1	Detailed Description	123
8.11.2	Member Data Documentation	123
8.11.2.1	numFormats	123
8.11.2.2	videoModes	123
8.12	EmbeddedImageInfo Struct Reference	124
8.12.1	Detailed Description	124
8.12.2	Member Data Documentation	125
8.12.2.1	brightness	125
8.12.2.2	exposure	125
8.12.2.3	frameCounter	125
8.12.2.4	gain	125
8.12.2.5	GPIOPinState	125
8.12.2.6	ROIPosition	125

8.12.2.7	shutter	125
8.12.2.8	strobePattern	125
8.12.2.9	timestamp	125
8.12.2.10	whiteBalance	125
8.13	EmbeddedImageInfoProperty Struct Reference	126
8.13.1	Detailed Description	126
8.13.2	Constructor & Destructor Documentation	126
8.13.2.1	EmbeddedImageInfoProperty	126
8.13.3	Member Data Documentation	126
8.13.3.1	available	126
8.13.3.2	onOff	126
8.14	Error Class Reference	127
8.14.1	Detailed Description	128
8.14.2	Constructor & Destructor Documentation	128
8.14.2.1	Error	128
8.14.2.2	Error	128
8.14.2.3	~Error	128
8.14.3	Member Function Documentation	128
8.14.3.1	CollectSupportInformation	128
8.14.3.2	GetBuildDate	128
8.14.3.3	GetCause	129
8.14.3.4	GetDescription	129
8.14.3.5	GetFilename	129
8.14.3.6	GetLine	129
8.14.3.7	GetType	129
8.14.3.8	operator!=	129
8.14.3.9	operator!=	129
8.14.3.10	operator=	130
8.14.3.11	operator==	130
8.14.3.12	operator==	130
8.14.3.13	PrintErrorTrace	130
8.14.4	Friends And Related Function Documentation	130
8.14.4.1	InternalError	130
8.15	FC2Config Struct Reference	131
8.15.1	Detailed Description	131
8.15.2	Constructor & Destructor Documentation	132

8.15.2.1	FC2Config	132
8.15.3	Member Data Documentation	132
8.15.3.1	asyncBusSpeed	132
8.15.3.2	bandwidthAllocation	132
8.15.3.3	grabMode	132
8.15.3.4	grabTimeout	132
8.15.3.5	isochBusSpeed	132
8.15.3.6	numBuffers	132
8.15.3.7	numImageNotifications	132
8.15.3.8	reserved	133
8.16	FC2Version Struct Reference	134
8.16.1	Detailed Description	134
8.16.2	Member Data Documentation	134
8.16.2.1	build	134
8.16.2.2	major	134
8.16.2.3	minor	134
8.16.2.4	type	134
8.17	Format7ImageSettings Struct Reference	135
8.17.1	Detailed Description	135
8.17.2	Constructor & Destructor Documentation	135
8.17.2.1	Format7ImageSettings	135
8.17.3	Member Data Documentation	135
8.17.3.1	height	135
8.17.3.2	mode	136
8.17.3.3	offsetX	136
8.17.3.4	offsetY	136
8.17.3.5	pixelFormat	136
8.17.3.6	reserved	136
8.17.3.7	width	136
8.18	Format7Info Struct Reference	137
8.18.1	Detailed Description	138
8.18.2	Constructor & Destructor Documentation	138
8.18.2.1	Format7Info	138
8.18.3	Member Data Documentation	138
8.18.3.1	imageHStepSize	138
8.18.3.2	imageVStepSize	138

8.18.3.3	maxHeight	138
8.18.3.4	maxPacketSize	138
8.18.3.5	maxWidth	138
8.18.3.6	minPacketSize	138
8.18.3.7	mode	138
8.18.3.8	offsetHStepSize	138
8.18.3.9	offsetVStepSize	138
8.18.3.10	packetSize	139
8.18.3.11	percentage	139
8.18.3.12	pixelFormatBitField	139
8.18.3.13	reserved	139
8.19	Format7PacketInfo Struct Reference	140
8.19.1	Detailed Description	140
8.19.2	Constructor & Destructor Documentation	140
8.19.2.1	Format7PacketInfo	140
8.19.3	Member Data Documentation	140
8.19.3.1	maxBytesPerPacket	140
8.19.3.2	recommendedBytesPerPacket	140
8.19.3.3	reserved	140
8.19.3.4	unitBytesPerPacket	141
8.20	GigECamera Class Reference	142
8.20.1	Detailed Description	147
8.20.2	Constructor & Destructor Documentation	147
8.20.2.1	GigECamera	147
8.20.2.2	~GigECamera	147
8.20.3	Member Function Documentation	147
8.20.3.1	Connect	147
8.20.3.2	Disconnect	148
8.20.3.3	DiscoverGigEPacketSize	148
8.20.3.4	EnableLUT	148
8.20.3.5	FireSoftwareTrigger	149
8.20.3.6	GetActiveLUTBank	149
8.20.3.7	GetCameraInfo	149
8.20.3.8	GetConfiguration	149
8.20.3.9	GetEmbeddedImageInfo	150
8.20.3.10	GetGigEImageBinningSettings	150



8.20.3.11 GetGigEImageSettings . . . . .	150
8.20.3.12 GetGigEImageSettingsInfo . . . . .	151
8.20.3.13 GetGigEImagingMode . . . . .	151
8.20.3.14 GetGigEProperty . . . . .	151
8.20.3.15 GetGigEStreamChannelInfo . . . . .	151
8.20.3.16 GetGPIOPinDirection . . . . .	152
8.20.3.17 GetLUTBankInfo . . . . .	152
8.20.3.18 GetLUTChannel . . . . .	152
8.20.3.19 GetLUTInfo . . . . .	153
8.20.3.20 GetMemoryChannel . . . . .	153
8.20.3.21 GetMemoryChannelInfo . . . . .	154
8.20.3.22 GetNumStreamChannels . . . . .	154
8.20.3.23 GetProperty . . . . .	154
8.20.3.24 GetPropertyInfo . . . . .	155
8.20.3.25 GetRegisterString . . . . .	155
8.20.3.26 GetStrobe . . . . .	155
8.20.3.27 GetStrobeInfo . . . . .	156
8.20.3.28 GetTriggerDelay . . . . .	156
8.20.3.29 GetTriggerDelayInfo . . . . .	156
8.20.3.30 GetTriggerMode . . . . .	157
8.20.3.31 GetTriggerModeInfo . . . . .	157
8.20.3.32 IsConnected . . . . .	157
8.20.3.33 QueryGigEImagingMode . . . . .	158
8.20.3.34 ReadGVCPMemory . . . . .	158
8.20.3.35 ReadGVCPRegister . . . . .	158
8.20.3.36 ReadGVCPRegisterBlock . . . . .	159
8.20.3.37 ReadRegister . . . . .	159
8.20.3.38 ReadRegisterBlock . . . . .	159
8.20.3.39 RestoreFromMemoryChannel . . . . .	160
8.20.3.40 RetrieveBuffer . . . . .	160
8.20.3.41 SaveToMemoryChannel . . . . .	160
8.20.3.42 SetActiveLUTBank . . . . .	161
8.20.3.43 SetCallback . . . . .	161
8.20.3.44 SetConfiguration . . . . .	161
8.20.3.45 SetEmbeddedImageInfo . . . . .	162
8.20.3.46 SetGigEImageBinningSettings . . . . .	162

8.20.3.47 SetGigEImageSettings . . . . .	162
8.20.3.48 SetGigEImagingMode . . . . .	162
8.20.3.49 SetGigEProperty . . . . .	163
8.20.3.50 SetGigEStreamChannelInfo . . . . .	163
8.20.3.51 SetGPIOPinDirection . . . . .	163
8.20.3.52 SetLUTChannel . . . . .	164
8.20.3.53 SetProperty . . . . .	164
8.20.3.54 SetStrobe . . . . .	165
8.20.3.55 SetTriggerDelay . . . . .	165
8.20.3.56 SetTriggerMode . . . . .	165
8.20.3.57 SetUserBuffers . . . . .	166
8.20.3.58 StartCapture . . . . .	166
8.20.3.59 StartSyncCapture . . . . .	167
8.20.3.60 StopCapture . . . . .	167
8.20.3.61 WaitForBufferEvent . . . . .	167
8.20.3.62 WriteGVCPMemory . . . . .	168
8.20.3.63 WriteGVCPRegister . . . . .	168
8.20.3.64 WriteGVCPRegisterBlock . . . . .	168
8.20.3.65 WriteRegister . . . . .	168
8.20.3.66 WriteRegisterBlock . . . . .	169
8.21 GigEConfig Struct Reference . . . . .	170
8.21.1 Detailed Description . . . . .	170
8.21.2 Constructor & Destructor Documentation . . . . .	171
8.21.2.1 GigEConfig . . . . .	171
8.21.3 Member Data Documentation . . . . .	171
8.21.3.1 channels . . . . .	171
8.21.3.2 numChannels . . . . .	171
8.22 GigEImageSettings Struct Reference . . . . .	172
8.22.1 Detailed Description . . . . .	172
8.22.2 Constructor & Destructor Documentation . . . . .	172
8.22.2.1 GigEImageSettings . . . . .	172
8.22.3 Member Data Documentation . . . . .	172
8.22.3.1 height . . . . .	172
8.22.3.2 offsetX . . . . .	172
8.22.3.3 offsetY . . . . .	173
8.22.3.4 pixelFormat . . . . .	173

8.22.3.5	reserved	173
8.22.3.6	width	173
8.23	GigEImageSettingsInfo Struct Reference	174
8.23.1	Detailed Description	174
8.23.2	Constructor & Destructor Documentation	174
8.23.2.1	GigEImageSettingsInfo	174
8.23.3	Member Data Documentation	174
8.23.3.1	imageHStepSize	174
8.23.3.2	imageVStepSize	175
8.23.3.3	maxHeight	175
8.23.3.4	maxWidth	175
8.23.3.5	offsetHStepSize	175
8.23.3.6	offsetVStepSize	175
8.23.3.7	pixelFormatBitField	175
8.23.3.8	reserved	175
8.24	GigEProperty Struct Reference	176
8.24.1	Detailed Description	176
8.24.2	Member Data Documentation	176
8.24.2.1	isReadable	176
8.24.2.2	isWritable	176
8.24.2.3	max	176
8.24.2.4	min	176
8.24.2.5	propType	177
8.24.2.6	value	177
8.25	GigEStreamChannel Struct Reference	178
8.25.1	Detailed Description	178
8.25.2	Constructor & Destructor Documentation	179
8.25.2.1	GigEStreamChannel	179
8.25.3	Member Data Documentation	179
8.25.3.1	destinationIpAddress	179
8.25.3.2	doNotFragment	179
8.25.3.3	hostPort	179
8.25.3.4	interPacketDelay	179
8.25.3.5	networkInterfaceIndex	179
8.25.3.6	packetSize	179
8.25.3.7	sourcePort	179

8.26	HostAdapterStats Struct Reference	180
8.26.1	Detailed Description	180
8.26.2	Constructor & Destructor Documentation	181
8.26.2.1	HostAdapterStats	181
8.26.3	Member Data Documentation	181
8.26.3.1	busErrors	181
8.26.3.2	busResets	181
8.26.3.3	cycleTime	181
8.26.3.4	deviceArrivals	181
8.26.3.5	deviceRemovals	181
8.26.3.6	fifoOverflows	181
8.26.3.7	gapCount	181
8.26.3.8	numPorts	181
8.26.3.9	portErrors	181
8.26.3.10	vendor	181
8.27	Image Class Reference	182
8.27.1	Detailed Description	184
8.27.2	Constructor & Destructor Documentation	185
8.27.2.1	Image	185
8.27.2.2	Image	185
8.27.2.3	Image	185
8.27.2.4	Image	185
8.27.2.5	Image	186
8.27.2.6	~Image	186
8.27.3	Member Function Documentation	186
8.27.3.1	CalculateStatistics	186
8.27.3.2	Convert	186
8.27.3.3	Convert	186
8.27.3.4	DeepCopy	187
8.27.3.5	DetermineBitsPerPixel	187
8.27.3.6	GetBayerTileFormat	187
8.27.3.7	GetBitsPerPixel	187
8.27.3.8	GetColorProcessing	188
8.27.3.9	GetCols	188
8.27.3.10	GetData	188
8.27.3.11	GetData	188

8.27.3.12	GetDataSize	188
8.27.3.13	GetDefaultColorProcessing	188
8.27.3.14	GetDefaultOutputFormat	189
8.27.3.15	GetDimensions	189
8.27.3.16	GetMetadata	189
8.27.3.17	GetPixelFormat	189
8.27.3.18	GetRows	189
8.27.3.19	GetStride	190
8.27.3.20	GetTimeStamp	190
8.27.3.21	operator()	190
8.27.3.22	operator=	190
8.27.3.23	operator[]	190
8.27.3.24	ReleaseBuffer	191
8.27.3.25	Save	191
8.27.3.26	Save	191
8.27.3.27	Save	191
8.27.3.28	Save	192
8.27.3.29	Save	192
8.27.3.30	Save	192
8.27.3.31	Save	192
8.27.3.32	SetColorProcessing	193
8.27.3.33	SetData	193
8.27.3.34	SetDefaultColorProcessing	193
8.27.3.35	SetDefaultOutputFormat	194
8.27.3.36	SetDimensions	194
8.27.4	Friends And Related Function Documentation	194
8.27.4.1	Iso	194
8.28	ImageMetadata Struct Reference	195
8.28.1	Detailed Description	195
8.28.2	Constructor & Destructor Documentation	196
8.28.2.1	ImageMetadata	196
8.28.3	Member Data Documentation	196
8.28.3.1	embeddedBrightness	196
8.28.3.2	embeddedExposure	196
8.28.3.3	embeddedFrameCounter	196
8.28.3.4	embeddedGain	196

8.28.3.5	<a href="#">embeddedGPIOPinState</a>	196
8.28.3.6	<a href="#">embeddedROIPosition</a>	196
8.28.3.7	<a href="#">embeddedShutter</a>	196
8.28.3.8	<a href="#">embeddedStrobePattern</a>	196
8.28.3.9	<a href="#">embeddedTimeStamp</a>	196
8.28.3.10	<a href="#">embeddedWhiteBalance</a>	196
8.28.3.11	<a href="#">reserved</a>	197
8.29	<a href="#">ImageStatistics Class Reference</a>	198
8.29.1	<a href="#">Detailed Description</a>	199
8.29.2	<a href="#">Member Enumeration Documentation</a>	199
8.29.2.1	<a href="#">StatisticsChannel</a>	199
8.29.3	<a href="#">Constructor &amp; Destructor Documentation</a>	200
8.29.3.1	<a href="#">ImageStatistics</a>	200
8.29.3.2	<a href="#">~ImageStatistics</a>	200
8.29.3.3	<a href="#">ImageStatistics</a>	200
8.29.4	<a href="#">Member Function Documentation</a>	200
8.29.4.1	<a href="#">DisableAll</a>	200
8.29.4.2	<a href="#">EnableAll</a>	200
8.29.4.3	<a href="#">EnableGreyOnly</a>	200
8.29.4.4	<a href="#">EnableHSLOnly</a>	200
8.29.4.5	<a href="#">EnableRGBOnly</a>	200
8.29.4.6	<a href="#">GetChannelStatus</a>	201
8.29.4.7	<a href="#">GetHistogram</a>	201
8.29.4.8	<a href="#">GetMean</a>	201
8.29.4.9	<a href="#">GetNumPixelValues</a>	201
8.29.4.10	<a href="#">GetPixelValueRange</a>	202
8.29.4.11	<a href="#">GetRange</a>	202
8.29.4.12	<a href="#">GetStatistics</a>	202
8.29.4.13	<a href="#">operator=</a>	203
8.29.4.14	<a href="#">SetChannelStatus</a>	203
8.29.5	<a href="#">Friends And Related Function Documentation</a>	203
8.29.5.1	<a href="#">ImageStatsCalculator</a>	203
8.30	<a href="#">IPAddress Struct Reference</a>	204
8.30.1	<a href="#">Detailed Description</a>	204
8.30.2	<a href="#">Constructor &amp; Destructor Documentation</a>	204
8.30.2.1	<a href="#">IPAddress</a>	204

8.30.2.2	IPAddress	204
8.30.3	Member Function Documentation	204
8.30.3.1	operator!=	204
8.30.3.2	operator==	204
8.30.4	Member Data Documentation	204
8.30.4.1	octets	204
8.31	JPEGOption Struct Reference	205
8.31.1	Detailed Description	205
8.31.2	Constructor & Destructor Documentation	205
8.31.2.1	JPEGOption	205
8.31.3	Member Data Documentation	205
8.31.3.1	progressive	205
8.31.3.2	quality	205
8.31.3.3	reserved	205
8.32	JPG2Option Struct Reference	206
8.32.1	Detailed Description	206
8.32.2	Constructor & Destructor Documentation	206
8.32.2.1	JPG2Option	206
8.32.3	Member Data Documentation	206
8.32.3.1	quality	206
8.32.3.2	reserved	206
8.33	LUTData Struct Reference	207
8.33.1	Detailed Description	207
8.33.2	Constructor & Destructor Documentation	207
8.33.2.1	LUTData	207
8.33.3	Member Data Documentation	207
8.33.3.1	enabled	207
8.33.3.2	inputBitDepth	207
8.33.3.3	numBanks	208
8.33.3.4	numChannels	208
8.33.3.5	numEntries	208
8.33.3.6	outputBitDepth	208
8.33.3.7	reserved	208
8.33.3.8	supported	208
8.34	MACAddress Struct Reference	209
8.34.1	Detailed Description	209

8.34.2	Constructor & Destructor Documentation	209
8.34.2.1	MACAddress	209
8.34.2.2	MACAddress	209
8.34.3	Member Function Documentation	209
8.34.3.1	operator!=	209
8.34.3.2	operator==	209
8.34.4	Member Data Documentation	209
8.34.4.1	octets	209
8.35	PGMOption Struct Reference	210
8.35.1	Detailed Description	210
8.35.2	Constructor & Destructor Documentation	210
8.35.2.1	PGMOption	210
8.35.3	Member Data Documentation	210
8.35.3.1	binaryFile	210
8.35.3.2	reserved	210
8.36	PGRGuid Class Reference	211
8.36.1	Detailed Description	211
8.36.2	Constructor & Destructor Documentation	211
8.36.2.1	PGRGuid	211
8.36.3	Member Function Documentation	211
8.36.3.1	operator!=	211
8.36.3.2	operator==	211
8.36.4	Member Data Documentation	211
8.36.4.1	value	211
8.37	PNGOption Struct Reference	212
8.37.1	Detailed Description	212
8.37.2	Constructor & Destructor Documentation	212
8.37.2.1	PNGOption	212
8.37.3	Member Data Documentation	212
8.37.3.1	compressionLevel	212
8.37.3.2	interlaced	212
8.37.3.3	reserved	212
8.38	PPMOption Struct Reference	213
8.38.1	Detailed Description	213
8.38.2	Constructor & Destructor Documentation	213
8.38.2.1	PPMOption	213



8.38.3	Member Data Documentation	213
8.38.3.1	binaryFile	213
8.38.3.2	reserved	213
8.39	Property Struct Reference	214
8.39.1	Detailed Description	214
8.39.2	Constructor & Destructor Documentation	215
8.39.2.1	Property	215
8.39.2.2	Property	215
8.39.3	Member Data Documentation	215
8.39.3.1	absControl	215
8.39.3.2	absValue	215
8.39.3.3	autoManualMode	215
8.39.3.4	onePush	215
8.39.3.5	onOff	215
8.39.3.6	present	215
8.39.3.7	reserved	215
8.39.3.8	type	215
8.39.3.9	valueA	215
8.39.3.10	valueB	215
8.40	PropertyInfo Struct Reference	216
8.40.1	Detailed Description	217
8.40.2	Constructor & Destructor Documentation	217
8.40.2.1	PropertyInfo	217
8.40.2.2	PropertyInfo	217
8.40.3	Member Data Documentation	217
8.40.3.1	absMax	217
8.40.3.2	absMin	217
8.40.3.3	absValSupported	217
8.40.3.4	autoSupported	217
8.40.3.5	manualSupported	217
8.40.3.6	max	217
8.40.3.7	min	217
8.40.3.8	onePushSupported	217
8.40.3.9	onOffSupported	217
8.40.3.10	present	218
8.40.3.11	pUnitAbbr	218

8.40.3.12	pUnits	218
8.40.3.13	readOutSupported	218
8.40.3.14	reserved	218
8.40.3.15	type	218
8.41	StrobeControl Struct Reference	219
8.41.1	Detailed Description	219
8.41.2	Constructor & Destructor Documentation	219
8.41.2.1	StrobeControl	219
8.41.3	Member Data Documentation	219
8.41.3.1	delay	219
8.41.3.2	duration	219
8.41.3.3	onOff	219
8.41.3.4	polarity	220
8.41.3.5	reserved	220
8.41.3.6	source	220
8.42	StrobeInfo Struct Reference	221
8.42.1	Detailed Description	221
8.42.2	Constructor & Destructor Documentation	221
8.42.2.1	StrobeInfo	221
8.42.3	Member Data Documentation	221
8.42.3.1	maxValue	221
8.42.3.2	minValue	221
8.42.3.3	onOffSupported	222
8.42.3.4	polaritySupported	222
8.42.3.5	present	222
8.42.3.6	readOutSupported	222
8.42.3.7	reserved	222
8.42.3.8	source	222
8.43	SystemInfo Struct Reference	223
8.43.1	Detailed Description	223
8.43.2	Member Data Documentation	223
8.43.2.1	byteOrder	223
8.43.2.2	cpuDescription	224
8.43.2.3	driverList	224
8.43.2.4	gpuDescription	224
8.43.2.5	libraryList	224

8.43.2.6	numCpuCores	224
8.43.2.7	osDescription	224
8.43.2.8	osType	224
8.43.2.9	reserved	224
8.43.2.10	screenHeight	224
8.43.2.11	screenWidth	224
8.43.2.12	sysMemSize	224
8.44	TIFFOption Struct Reference	225
8.44.1	Detailed Description	225
8.44.2	Member Enumeration Documentation	225
8.44.2.1	CompressionMethod	225
8.44.3	Constructor & Destructor Documentation	226
8.44.3.1	TIFFOption	226
8.44.4	Member Data Documentation	226
8.44.4.1	compression	226
8.44.4.2	reserved	226
8.45	TimeStamp Struct Reference	227
8.45.1	Detailed Description	227
8.45.2	Constructor & Destructor Documentation	227
8.45.2.1	TimeStamp	227
8.45.3	Member Data Documentation	227
8.45.3.1	cycleCount	227
8.45.3.2	cycleOffset	227
8.45.3.3	cycleSeconds	227
8.45.3.4	microSeconds	227
8.45.3.5	reserved	228
8.45.3.6	seconds	228
8.46	TopologyNode Class Reference	229
8.46.1	Detailed Description	230
8.46.2	Member Enumeration Documentation	230
8.46.2.1	NodeType	230
8.46.2.2	PortType	230
8.46.3	Constructor & Destructor Documentation	230
8.46.3.1	TopologyNode	230
8.46.3.2	TopologyNode	230
8.46.3.3	~TopologyNode	230

8.46.3.4	TopologyNode	231
8.46.4	Member Function Documentation	231
8.46.4.1	AddChild	231
8.46.4.2	AddPort	231
8.46.4.3	AssignGuidToNode	231
8.46.4.4	AssignGuidToNode	231
8.46.4.5	GetChild	231
8.46.4.6	GetDeviceId	231
8.46.4.7	GetGuid	232
8.46.4.8	GetInterfaceType	232
8.46.4.9	GetNodeType	232
8.46.4.10	GetNumChildren	232
8.46.4.11	GetNumPorts	232
8.46.4.12	GetPortType	232
8.46.4.13	operator=	232
8.47	TriggerMode Struct Reference	233
8.47.1	Detailed Description	233
8.47.2	Constructor & Destructor Documentation	233
8.47.2.1	TriggerMode	233
8.47.3	Member Data Documentation	233
8.47.3.1	mode	233
8.47.3.2	onOff	233
8.47.3.3	parameter	233
8.47.3.4	polarity	233
8.47.3.5	reserved	233
8.47.3.6	source	234
8.48	TriggerModeInfo Struct Reference	235
8.48.1	Detailed Description	235
8.48.2	Constructor & Destructor Documentation	235
8.48.2.1	TriggerModeInfo	235
8.48.3	Member Data Documentation	235
8.48.3.1	modeMask	235
8.48.3.2	onOffSupported	235
8.48.3.3	polaritySupported	235
8.48.3.4	present	236
8.48.3.5	readOutSupported	236

8.48.3.6	reserved	236
8.48.3.7	softwareTriggerSupported	236
8.48.3.8	sourceMask	236
8.48.3.9	valueReadable	236
8.49	Utilities Class Reference	237
8.49.1	Detailed Description	237
8.49.2	Member Function Documentation	237
8.49.2.1	GetLibraryVersion	237
8.49.2.2	GetSystemInfo	237
8.49.2.3	LaunchBrowser	237
8.49.2.4	LaunchCommand	238
8.49.2.5	LaunchCommandAsync	238
8.49.2.6	LaunchHelp	238
8.50	VideoModes Struct Reference	239
8.50.1	Detailed Description	239
8.50.2	Member Data Documentation	239
8.50.2.1	frameRate	239
8.50.2.2	height	239
8.50.2.3	videoMode	239
8.50.2.4	width	239
<b>9</b>	<b>File Documentation</b>	<b>241</b>
9.1	AVIRecorder.h File Reference	241
9.2	BusManager.h File Reference	242
9.3	Camera.h File Reference	243
9.4	CameraBase.h File Reference	244
9.5	Error.h File Reference	245
9.6	FlyCapture2.h File Reference	246
9.7	FlyCapture2Defs.h File Reference	247
9.7.1	Define Documentation	254
9.7.1.1	FULL_32BIT_VALUE	254
9.7.1.2	NULL	254
9.8	FlyCapture2GUI.h File Reference	255
9.9	FlyCapture2Platform.h File Reference	256
9.9.1	Define Documentation	256
9.9.1.1	FLYCAPTURE2_API	256
9.10	GigECamera.h File Reference	257

9.11 Image.h File Reference . . . . .	258
9.12 ImageStatistics.h File Reference . . . . .	259
9.13 PGRDirectShow.h File Reference . . . . .	260
9.13.1 Function Documentation . . . . .	265
9.13.1.1 GetAbsBrightness . . . . .	265
9.13.1.2 GetAbsBrightnessRange . . . . .	265
9.13.1.3 GetAbsExposure . . . . .	265
9.13.1.4 GetAbsExposureRange . . . . .	265
9.13.1.5 GetAbsGain . . . . .	265
9.13.1.6 GetAbsGainRange . . . . .	265
9.13.1.7 GetAbsGamma . . . . .	265
9.13.1.8 GetAbsGammaRange . . . . .	265
9.13.1.9 GetAbsHue . . . . .	265
9.13.1.10 GetAbsHueRange . . . . .	265
9.13.1.11 GetAbsPan . . . . .	265
9.13.1.12 GetAbsPanRange . . . . .	265
9.13.1.13 GetAbsSaturation . . . . .	265
9.13.1.14 GetAbsSaturationRange . . . . .	265
9.13.1.15 GetAbsSharpness . . . . .	265
9.13.1.16 GetAbsSharpnessRange . . . . .	265
9.13.1.17 GetAbsShutter . . . . .	265
9.13.1.18 GetAbsShutterRange . . . . .	265
9.13.1.19 GetAbsTilt . . . . .	265
9.13.1.20 GetAbsTiltRange . . . . .	265
9.13.1.21 GetAbsWhiteBalance . . . . .	265
9.13.1.22 GetAbsWhiteBalanceRange . . . . .	265
9.13.1.23 GetBrightness . . . . .	265
9.13.1.24 GetBrightnessRange . . . . .	265
9.13.1.25 GetCustomImage . . . . .	265
9.13.1.26 GetCustomImageMode . . . . .	266
9.13.1.27 GetExposure . . . . .	266
9.13.1.28 GetExposureRange . . . . .	266
9.13.1.29 GetGain . . . . .	266
9.13.1.30 GetGainRange . . . . .	266
9.13.1.31 GetGamma . . . . .	266
9.13.1.32 GetGammaRange . . . . .	266

9.13.1.33 GetHue . . . . .	266
9.13.1.34 GetHueRange . . . . .	266
9.13.1.35 GetImageFormat . . . . .	266
9.13.1.36 GetOutputVerticalFlip . . . . .	266
9.13.1.37 GetPan . . . . .	267
9.13.1.38 GetPanRange . . . . .	267
9.13.1.39 GetSaturation . . . . .	267
9.13.1.40 GetSaturationRange . . . . .	267
9.13.1.41 GetSharpness . . . . .	267
9.13.1.42 GetSharpnessRange . . . . .	267
9.13.1.43 GetShutter . . . . .	267
9.13.1.44 GetShutterRange . . . . .	267
9.13.1.45 GetTilt . . . . .	267
9.13.1.46 GetTiltRange . . . . .	267
9.13.1.47 GetWhiteBalance . . . . .	267
9.13.1.48 GetWhiteBalanceRange . . . . .	267
9.13.1.49 QueryCustomImage . . . . .	267
9.13.1.50 ReadRegister . . . . .	267
9.13.1.51 SetAbsBrightness . . . . .	268
9.13.1.52 SetAbsExposure . . . . .	268
9.13.1.53 SetAbsGain . . . . .	268
9.13.1.54 SetAbsGamma . . . . .	268
9.13.1.55 SetAbsHue . . . . .	268
9.13.1.56 SetAbsPan . . . . .	268
9.13.1.57 SetAbsSaturation . . . . .	268
9.13.1.58 SetAbsSharpness . . . . .	268
9.13.1.59 SetAbsShutter . . . . .	268
9.13.1.60 SetAbsTilt . . . . .	268
9.13.1.61 SetAbsWhiteBalance . . . . .	268
9.13.1.62 SetBrightness . . . . .	268
9.13.1.63 SetCustomImage . . . . .	268
9.13.1.64 SetCustomImageMode . . . . .	269
9.13.1.65 SetExposure . . . . .	269
9.13.1.66 SetGain . . . . .	269
9.13.1.67 SetGamma . . . . .	269
9.13.1.68 SetHue . . . . .	269

---

9.13.1.69 SetImageFormat . . . . .	269
9.13.1.70 SetOutputVerticalFlip . . . . .	269
9.13.1.71 SetPan . . . . .	270
9.13.1.72 SetSaturation . . . . .	270
9.13.1.73 SetSharpness . . . . .	270
9.13.1.74 SetShutter . . . . .	270
9.13.1.75 SetTilt . . . . .	270
9.13.1.76 SetWhiteBalance . . . . .	270
9.13.1.77 WriteRegister . . . . .	270
9.13.2 Variable Documentation . . . . .	270
9.13.2.1 IID_IFlyCaptureProperties . . . . .	270
9.14 TopologyNode.h File Reference . . . . .	271
9.15 Utilities.h File Reference . . . . .	272



# Chapter 1

## Module Index

### 1.1 Modules

Here is a list of all modules:

Global constants . . . . .	11
Enumerations . . . . .	12
GigE specific enumerations . . . . .	28
Structures . . . . .	29
GigE specific structures . . . . .	32
IIDC specific structures . . . . .	33
Image saving structures. . . . .	34



# Chapter 2

## Namespace Index

### 2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">FlyCapture2</a>	.....	35
-----------------------------	-------	----



# Chapter 3

## Class Index

### 3.1 Class Hierarchy

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

AVIOption . . . . .	49
AVIRecorder . . . . .	51
BusManager . . . . .	53
CameraBase . . . . .	86
Camera . . . . .	61
GigECamera . . . . .	142
CameraControlDlg . . . . .	109
CameraInfo . . . . .	111
CameraSelectionDlg . . . . .	116
CameraStats . . . . .	117
ConfigROM . . . . .	120
DCAMFormats . . . . .	123
EmbeddedImageInfo . . . . .	124
EmbeddedImageInfoProperty . . . . .	126
Error . . . . .	127
FC2Config . . . . .	131
FC2Version . . . . .	134
Format7ImageSettings . . . . .	135
Format7Info . . . . .	137
Format7PacketInfo . . . . .	140
GigEConfig . . . . .	170
GigEImageSettings . . . . .	172
GigEImageSettingsInfo . . . . .	174
GigEProperty . . . . .	176
GigEStreamChannel . . . . .	178
HostAdapterStats . . . . .	180
Image . . . . .	182
ImageMetadata . . . . .	195
ImageStatistics . . . . .	198
IPAddress . . . . .	204
JPEGOption . . . . .	205
JPG2Option . . . . .	206
LUTData . . . . .	207

MACAddress . . . . .	209
PGMOption . . . . .	210
PGRGuid . . . . .	211
PNGOption . . . . .	212
PPMOption . . . . .	213
Property . . . . .	214
PropertyInfo . . . . .	216
StrobeControl . . . . .	219
StrobeInfo . . . . .	221
SystemInfo . . . . .	223
TIFFOption . . . . .	225
TimeStamp . . . . .	227
TopologyNode . . . . .	229
TriggerMode . . . . .	233
TriggerModeInfo . . . . .	235
Utilities . . . . .	237
VideoModes . . . . .	239

# Chapter 4

## Class Index

### 4.1 Class List

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

<a href="#">AVIOption</a> (Options for saving AVI files ) . . . . .	49
<a href="#">AVIRecorder</a> (Functionality for the user to record images to an AVI file ) . . . . .	51
<a href="#">BusManager</a> (Functionality for the user to get an <a href="#">PGRGuid</a> for a desired camera or device easily ) . . . . .	53
<a href="#">Camera</a> (The <a href="#">Camera</a> object represents a physical camera that uses the IIDC register set ) . . . . .	61
<a href="#">CameraBase</a> (Abstract base class that defines a general interface to a camera ) . . . . .	86
<a href="#">CameraControlDlg</a> (The <a href="#">CameraControlDlg</a> object represents a GTKmm dialog that provides a graphical interface to a specified camera ) . . . . .	109
<a href="#">CameraInfo</a> ( <a href="#">Camera</a> information ) . . . . .	111
<a href="#">CameraSelectionDlg</a> (The <a href="#">CameraSelectionDlg</a> object represents a GTKmm dialog that provides a graphical interface that lists the number of cameras available to the library ) . . . . .	116
<a href="#">CameraStats</a> ( <a href="#">Camera</a> diagnostic information ) . . . . .	117
<a href="#">ConfigROM</a> ( <a href="#">Camera</a> configuration ROM ) . . . . .	120
<a href="#">DCAMFormats</a> (DCAM formats supported by the camera as returned by <code>GetAvailableFormats()</code> ) . . . . .	123
<a href="#">EmbeddedImageInfo</a> (Properties of the possible embedded image information ) . . . . .	124
<a href="#">EmbeddedImageInfoProperty</a> (Properties of a single embedded image info property ) . . . . .	126
<a href="#">Error</a> (The <a href="#">Error</a> object represents an error that is returned from the library ) . . . . .	127
<a href="#">FC2Config</a> (Configuration for a camera ) . . . . .	131
<a href="#">FC2Version</a> (The current version of the library ) . . . . .	134
<a href="#">Format7ImageSettings</a> (Format 7 image settings ) . . . . .	135
<a href="#">Format7Info</a> (Format 7 information for a single mode ) . . . . .	137
<a href="#">Format7PacketInfo</a> (Format 7 packet information ) . . . . .	140
<a href="#">GigECamera</a> (The <a href="#">GigECamera</a> object represents a physical Gigabit Ethernet camera ) . . . . .	142
<a href="#">GigEConfig</a> (Configuration for a GigE camera ) . . . . .	170
<a href="#">GigEImageSettings</a> ( <a href="#">Image</a> settings for a GigE camera ) . . . . .	172
<a href="#">GigEImageSettingsInfo</a> (Format 7 information for a single mode ) . . . . .	174
<a href="#">GigEProperty</a> (A GigE property ) . . . . .	176
<a href="#">GigEStreamChannel</a> (Information about a single GigE stream channel ) . . . . .	178
<a href="#">HostAdapterStats</a> (Information about the host adapter's statistics ) . . . . .	180
<a href="#">Image</a> (Used to retrieve images from a camera, convert between multiple pixel formats and save images to disk ) . . . . .	182
<a href="#">ImageMetadata</a> (Metadata related to an image ) . . . . .	195
<a href="#">ImageStatistics</a> (The <a href="#">ImageStatistics</a> object represents image statistics for an image ) . . . . .	198
<a href="#">IPAddress</a> (IPv4 address ) . . . . .	204

<a href="#">JPEGOption</a> (Options for saving JPEG image ) . . . . .	205
<a href="#">JPG2Option</a> (Options for saving JPEG2000 image ) . . . . .	206
<a href="#">LUTData</a> (Information about the camera's look up table ) . . . . .	207
<a href="#">MACAddress</a> (MAC address ) . . . . .	209
<a href="#">PGMOption</a> (Options for saving PGM images ) . . . . .	210
<a href="#">PGRGuid</a> (A GUID to the camera ) . . . . .	211
<a href="#">PNGOption</a> (Options for saving PNG images ) . . . . .	212
<a href="#">PPMOption</a> (Options for saving PPM images ) . . . . .	213
<a href="#">Property</a> (A specific camera property ) . . . . .	214
<a href="#">PropertyInfo</a> (Information about a specific camera property ) . . . . .	216
<a href="#">StrobeControl</a> (A camera strobe ) . . . . .	219
<a href="#">StrobeInfo</a> (A camera strobe property ) . . . . .	221
<a href="#">SystemInfo</a> (Description of the system ) . . . . .	223
<a href="#">TIFFOption</a> (Options for saving TIFF images ) . . . . .	225
<a href="#">TimeStamp</a> (Timestamp information ) . . . . .	227
<a href="#">TopologyNode</a> (Topology information that can be used to generate a tree structure of all cameras and devices connected to a computer ) . . . . .	229
<a href="#">TriggerMode</a> (A camera trigger ) . . . . .	233
<a href="#">TriggerModeInfo</a> (Information about a camera trigger property ) . . . . .	235
<a href="#">Utilities</a> (The Utility class is generally used to query for general system information such as operating system, available memory etc ) . . . . .	237
<a href="#">VideoModes</a> (Information about a single DCAM format ) . . . . .	239



# Chapter 5

## File Index

### 5.1 File List

Here is a list of all files with brief descriptions:

<a href="#">AVIRecorder.h</a>	241
<a href="#">BusManager.h</a>	242
<a href="#">Camera.h</a>	243
<a href="#">CameraBase.h</a>	244
<a href="#">Error.h</a>	245
<a href="#">FlyCapture2.h</a>	246
<a href="#">FlyCapture2Defs.h</a>	247
<a href="#">FlyCapture2GUI.h</a>	255
<a href="#">FlyCapture2Platform.h</a>	256
<a href="#">GigECamera.h</a>	257
<a href="#">Image.h</a>	258
<a href="#">ImageStatistics.h</a>	259
<a href="#">PGRDirectShow.h</a>	260
<a href="#">TopologyNode.h</a>	271
<a href="#">Utilities.h</a>	272



# Chapter 6

## Module Documentation

### 6.1 Global constants

#### Variables

- static const unsigned int `sk_maxStringLength` = 512

*The maximum length that is allocated for a string.*

- static const unsigned int `sk_maxNumPorts` = 32

*The maximum number of ports one device can have.*

#### 6.1.1 Variable Documentation

**6.1.1.1** `const unsigned int sk_maxNumPorts = 32` `[static]`

The maximum number of ports one device can have.

**6.1.1.2** `const unsigned int sk_maxStringLength = 512` `[static]`

The maximum length that is allocated for a string.

## 6.2 Enumerations

### Enumerations

- enum `ErrorType` {  
    `PGRERROR_UNDEFINED` = -1,  
    `PGRERROR_OK`,  
    `PGRERROR_FAILED`,  
    `PGRERROR_NOT_IMPLEMENTED`,  
    `PGRERROR_FAILED_BUS_MASTER_CONNECTION`,  
    `PGRERROR_NOT_CONNECTED`,  
    `PGRERROR_INIT_FAILED`,  
    `PGRERROR_NOT_INTIALIZED`,  
    `PGRERROR_INVALID_PARAMETER`,  
    `PGRERROR_INVALID_SETTINGS`,  
    `PGRERROR_INVALID_BUS_MANAGER`,  
    `PGRERROR_MEMORY_ALLOCATION_FAILED`,  
    `PGRERROR_LOW_LEVEL_FAILURE`,  
    `PGRERROR_NOT_FOUND`,  
    `PGRERROR_FAILED_GUID`,  
    `PGRERROR_INVALID_PACKET_SIZE`,  
    `PGRERROR_INVALID_MODE`,  
    `PGRERROR_NOT_IN_FORMAT7`,  
    `PGRERROR_NOT_SUPPORTED`,  
    `PGRERROR_TIMEOUT`,  
    `PGRERROR_BUS_MASTER_FAILED`,  
    `PGRERROR_INVALID_GENERATION`,  
    `PGRERROR_LUT_FAILED`,  
    `PGRERROR_IIDC_FAILED`,  
    `PGRERROR_STROBE_FAILED`,  
    `PGRERROR_TRIGGER_FAILED`,  
    `PGRERROR_PROPERTY_FAILED`,  
    `PGRERROR_PROPERTY_NOT_PRESENT`,  
    `PGRERROR_REGISTER_FAILED`,  
    `PGRERROR_READ_REGISTER_FAILED`,  
    `PGRERROR_WRITE_REGISTER_FAILED`,  
    `PGRERROR_ISOCH_FAILED`,  
    `PGRERROR_ISOCH_ALREADY_STARTED`,  
    `PGRERROR_ISOCH_NOT_STARTED`,  
    `PGRERROR_ISOCH_START_FAILED`,  
    `PGRERROR_ISOCH_RETRIEVE_BUFFER_FAILED`,

```
PGRERROR_ISOCH_STOP_FAILED,  
PGRERROR_ISOCH_SYNC_FAILED,  
PGRERROR_ISOCH_BANDWIDTH_EXCEEDED,  
PGRERROR_IMAGE_CONVERSION_FAILED,  
PGRERROR_IMAGE_LIBRARY_FAILURE,  
PGRERROR_BUFFER_TOO_SMALL,  
PGRERROR_IMAGE_CONSISTENCY_ERROR,  
PGRERROR_FORCE_32BITS = FULL_32BIT_VALUE }
```

*The error types returned by functions.*

- enum `BusCallbackType` {  
    `BUS_RESET`,  
    `ARRIVAL`,  
    `REMOVAL`,  
    `CALLBACK_TYPE_FORCE_32BITS` = `FULL_32BIT_VALUE` }

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

- enum `GrabMode` {  
    `DROP_FRAMES`,  
    `BUFFER_FRAMES`,  
    `UNSPECIFIED_GRAB_MODE`,  
    `GRAB_MODE_FORCE_32BITS` = `FULL_32BIT_VALUE` }

*The grab strategy employed during image transfer.*

- enum `GrabTimeout` {  
    `TIMEOUT_NONE` = 0,  
    `TIMEOUT_INFINITE` = -1,  
    `TIMEOUT_UNSPECIFIED` = -2,  
    `GRAB_TIMEOUT_FORCE_32BITS` = `FULL_32BIT_VALUE` }

*Timeout options for grabbing images.*

- enum `BandwidthAllocation` {  
    `BANDWIDTH_ALLOCATION_OFF` = 0,  
    `BANDWIDTH_ALLOCATION_ON` = 1,  
    `BANDWIDTH_ALLOCATION_UNSUPPORTED` = 2,  
    `BANDWIDTH_ALLOCATION_UNSPECIFIED` = 3,  
    `BANDWIDTH_ALLOCATION_FORCE_32BITS` = `FULL_32BIT_VALUE` }

*Bandwidth allocation options for 1394 devices.*

- enum `InterfaceType` {  
    `INTERFACE_IEEE1394`,  
    `INTERFACE_USB2`,  
    `INTERFACE_GIGE`,  
    `INTERFACE_UNKNOWN`,  
    `INTERFACE_TYPE_FORCE_32BITS` = `FULL_32BIT_VALUE` }

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

- enum `PropertyType` {  
    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,  
    PROPERTY\_TYPE\_FORCE\_32BITS = FULL\_32BIT\_VALUE }

*Camera properties.*

- enum `FrameRate` {  
    FRAMERATE\_1\_875,  
    FRAMERATE\_3\_75,  
    FRAMERATE\_7\_5,  
    FRAMERATE\_15,  
    FRAMERATE\_30,  
    FRAMERATE\_60,  
    FRAMERATE\_120,  
    FRAMERATE\_240,  
    FRAMERATE\_FORMAT7,  
    NUM\_FRAMERATES,  
    FRAMERATE\_FORCE\_32BITS = FULL\_32BIT\_VALUE }

*Frame rates in frames per second.*

- enum VideoMode {  
    VIDEOMODE\_160x120YUV444,  
    VIDEOMODE\_320x240YUV422,  
    VIDEOMODE\_640x480YUV411,  
    VIDEOMODE\_640x480YUV422,  
    VIDEOMODE\_640x480RGB,  
    VIDEOMODE\_640x480Y8,  
    VIDEOMODE\_640x480Y16,  
    VIDEOMODE\_800x600YUV422,  
    VIDEOMODE\_800x600RGB,  
    VIDEOMODE\_800x600Y8,  
    VIDEOMODE\_800x600Y16,  
    VIDEOMODE\_1024x768YUV422,  
    VIDEOMODE\_1024x768RGB,  
    VIDEOMODE\_1024x768Y8,  
    VIDEOMODE\_1024x768Y16,  
    VIDEOMODE\_1280x960YUV422,  
    VIDEOMODE\_1280x960RGB,  
    VIDEOMODE\_1280x960Y8,  
    VIDEOMODE\_1280x960Y16,  
    VIDEOMODE\_1600x1200YUV422,  
    VIDEOMODE\_1600x1200RGB,  
    VIDEOMODE\_1600x1200Y8,  
    VIDEOMODE\_1600x1200Y16,  
    VIDEOMODE\_FORMAT7,  
    NUM\_VIDEOMODES,  
    VIDEOMODE\_FORCE\_32BITS = FULL\_32BIT\_VALUE }

*DCAM video modes.*

- enum Mode {  
    MODE\_0 = 0,  
    MODE\_1,  
    MODE\_2,  
    MODE\_3,  
    MODE\_4,  
    MODE\_5,  
    MODE\_6,  
    MODE\_7,  
    MODE\_8,  
    MODE\_9,  
    MODE\_10,

```

MODE_11,
MODE_12,
MODE_13,
MODE_14,
MODE_15,
MODE_16,
MODE_17,
MODE_18,
MODE_19,
MODE_20,
MODE_21,
MODE_22,
MODE_23,
MODE_24,
MODE_25,
MODE_26,
MODE_27,
MODE_28,
MODE_29,
MODE_30,
MODE_31,
NUM_MODES,
MODE_FORCE_32BITS = FULL_32BIT_VALUE }

```

*Camera modes for DCAM formats as well as Format7.*

- `enum PixelFormat {`

```

PIXEL_FORMAT_MONO8 = 0x80000000,
PIXEL_FORMAT_411YUV8 = 0x40000000,
PIXEL_FORMAT_422YUV8 = 0x20000000,
PIXEL_FORMAT_444YUV8 = 0x10000000,
PIXEL_FORMAT_RGB8 = 0x08000000,
PIXEL_FORMAT_MONO16 = 0x04000000,
PIXEL_FORMAT_RGB16 = 0x02000000,
PIXEL_FORMAT_S_MONO16 = 0x01000000,
PIXEL_FORMAT_S_RGB16 = 0x00800000,
PIXEL_FORMAT_RAW8 = 0x00400000,
PIXEL_FORMAT_RAW16 = 0x00200000,
PIXEL_FORMAT_MONO12 = 0x00100000,
PIXEL_FORMAT_RAW12 = 0x00080000,
PIXEL_FORMAT_BGR = 0x80000008,
PIXEL_FORMAT_BGRU = 0x40000008,

```



```
PIXEL_FORMAT_RGB = PIXEL_FORMAT_RGB8,  
PIXEL_FORMAT_RGBA = 0x40000002,  
NUM_PIXEL_FORMATS = 15,  
UNSPECIFIED_PIXEL_FORMAT = 0 }
```

*Pixel formats available for Format7 modes.*

- enum `BusSpeed` {  
    BUSSPEED\_S100,  
    BUSSPEED\_S200,  
    BUSSPEED\_S400,  
    BUSSPEED\_S480,  
    BUSSPEED\_S800,  
    BUSSPEED\_S1600,  
    BUSSPEED\_S3200,  
    BUSSPEED\_S5000,  
    BUSSPEED\_10BASE\_T,  
    BUSSPEED\_100BASE\_T,  
    BUSSPEED\_1000BASE\_T,  
    BUSSPEED\_10000BASE\_T,  
    BUSSPEED\_S\_FASTEST,  
    BUSSPEED\_ANY,  
    BUSSPEED\_SPEED\_UNKNOWN = -1,  
    BUSSPEED\_FORCE\_32BITS = FULL\_32BIT\_VALUE }

*Bus speeds.*

- enum `DriverType` {  
    DRIVER\_1394\_CAM,  
    DRIVER\_1394\_PRO,  
    DRIVER\_1394\_JUJU,  
    DRIVER\_1394\_VIDEO1394,  
    DRIVER\_1394\_RAW1394,  
    DRIVER\_USB\_NONE,  
    DRIVER\_USB\_CAM,  
    DRIVER\_USB3\_PRO,  
    DRIVER\_GIGE\_NONE,  
    DRIVER\_GIGE\_FILTER,  
    DRIVER\_GIGE\_PRO,  
    DRIVER\_UNKNOWN = -1,  
    DRIVER\_FORCE\_32BITS = FULL\_32BIT\_VALUE }

*Types of low level drivers that flycapture uses.*

- enum `ColorProcessingAlgorithm` {  
`DEFAULT`,  
`NO_COLOR_PROCESSING`,  
`NEAREST_NEIGHBOR`,  
`EDGE_SENSING`,  
`HQ_LINEAR`,  
`RIGOROUS`,  
`IPP`,  
`COLOR_PROCESSING_ALGORITHM_FORCE_32BITS` = `FULL_32BIT_VALUE` }

*Color processing algorithms.*

- enum `BayerTileFormat` {  
`NONE`,  
`RGGB`,  
`GRBG`,  
`GBRG`,  
`BGGR`,  
`BT_FORCE_32BITS` = `FULL_32BIT_VALUE` }

*Bayer tile formats.*

- enum `ImageFileFormat` {  
`FROM_FILE_EXT` = -1,  
`PGM`,  
`PPM`,  
`BMP`,  
`JPEG`,  
`JPEG2000`,  
`TIFF`,  
`PNG`,  
`RAW`,  
`IMAGE_FILE_FORMAT_FORCE_32BITS` = `FULL_32BIT_VALUE` }

*File formats to be used for saving images to disk.*

## 6.2.1 Enumeration Type Documentation

### 6.2.1.1 enum `BandwidthAllocation`

Bandwidth allocation options for 1394 devices.

#### Enumerator:

**`BANDWIDTH_ALLOCATION_OFF`** Do not allocate bandwidth.

**`BANDWIDTH_ALLOCATION_ON`** Allocate bandwidth.

This is the default setting.

***BANDWIDTH\_ALLOCATION\_UNSUPPORTED*** Bandwidth allocation is not supported by either the camera or operating system.

***BANDWIDTH\_ALLOCATION\_UNSPECIFIED*** Not specified.  
This leaves the current setting unchanged.

***BANDWIDTH\_ALLOCATION\_FORCE\_32BITS***

#### 6.2.1.2 enum BayerTileFormat

Bayer tile formats.

Enumerator:

***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.

***BT\_FORCE\_32BITS***

#### 6.2.1.3 enum BusCallbackType

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

Enumerator:

***BUS\_RESET*** Register for all bus events.

***ARRIVAL*** Register for arrivals only.

***REMOVAL*** Register for removals only.

***CALLBACK\_TYPE\_FORCE\_32BITS***

#### 6.2.1.4 enum BusSpeed

Bus speeds.

Enumerator:

***BUSSPEED\_S100*** 100Mbits/sec.

***BUSSPEED\_S200*** 200Mbits/sec.

***BUSSPEED\_S400*** 400Mbits/sec.

***BUSSPEED\_S480*** 480Mbits/sec.

Only for USB2 cameras.

***BUSSPEED\_S800*** 800Mbits/sec.

***BUSSPEED\_S1600*** 1600Mbits/sec.

***BUSSPEED\_S3200*** 3200Mbits/sec.

***BUSSPEED\_S5000*** 5000Mbps/sec.

Only for USB3 cameras.

***BUSSPEED\_10BASE\_T*** 10Base-T.

Only for GigE Vision cameras.

***BUSSPEED\_100BASE\_T*** 100Base-T.

Only for GigE Vision cameras.

***BUSSPEED\_1000BASE\_T*** 1000Base-T (Gigabit Ethernet).

Only for GigE Vision cameras.

***BUSSPEED\_10000BASE\_T*** 10000Base-T.

Only for GigE Vision cameras.

***BUSSPEED\_S\_FASTEST*** The fastest speed available.

***BUSSPEED\_ANY*** Any speed that is available.

***BUSSPEED\_SPEED\_UNKNOWN*** Unknown bus speed.

***BUSSPEED\_FORCE\_32BITS***

#### 6.2.1.5 enum ColorProcessingAlgorithm

Color processing algorithms.

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

**Enumerator:**

***DEFAULT*** Default method.

***NO\_COLOR\_PROCESSING*** No color processing.

***NEAREST\_NEIGHBOR*** Fastest but lowest quality.

Equivalent to FLYCAPTURE\_NEAREST\_NEIGHBOR\_FAST in FlyCapture.

***EDGE\_SENSING*** Weights surrounding pixels based on localized edge orientation.

***HQ\_LINEAR*** Similar quality to rigorous but much faster.

***RIGOROUS*** Slowest but produces the best results.

***IPP*** Multithreaded with similar results to edge sensing.

***COLOR\_PROCESSING\_ALGORITHM\_FORCE\_32BITS***

#### 6.2.1.6 enum DriverType

Types of low level drivers that flycapture uses.

**Enumerator:**

***DRIVER\_1394\_CAM*** PGRCam.sys.

***DRIVER\_1394\_PRO*** PGR1394.sys.

***DRIVER\_1394\_JUJU*** firewire\_core.

***DRIVER\_1394\_VIDEO1394*** video1394.

***DRIVER\_1394\_RAW1394*** raw1394.

***DRIVER\_USB\_NONE*** No usb driver used just BSD stack.  
 (Linux only)  
***DRIVER\_USB\_CAM*** PGRUsbCam.sys.  
***DRIVER\_USB3\_PRO*** PGRXHCl.sys.  
***DRIVER\_GIGE\_NONE*** no gige drivers used,MS/BSD stack.  
***DRIVER\_GIGE\_FILTER*** PGRGigE.sys.  
***DRIVER\_GIGE\_PRO*** PGRGigEPro.sys.  
***DRIVER\_UNKNOWN*** Unknown driver type.  
***DRIVER\_FORCE\_32BITS***

### 6.2.1.7 enum ErrorType

The error types returned by functions.

**Enumerator:**

***PGRERROR\_UNDEFINED*** Undefined.  
***PGRERROR\_OK*** Function returned with no errors.  
***PGRERROR\_FAILED*** General failure.  
***PGRERROR\_NOT\_IMPLEMENTED*** Function has not been implemented.  
***PGRERROR\_FAILED\_BUS\_MASTER\_CONNECTION*** Could not connect to Bus Master.  
***PGRERROR\_NOT\_CONNECTED*** [Camera](#) has not been connected.  
***PGRERROR\_INIT\_FAILED*** Initialization failed.  
***PGRERROR\_NOT\_INTIALIZED*** [Camera](#) has not been initialized.  
***PGRERROR\_INVALID\_PARAMETER*** Invalid parameter passed to function.  
***PGRERROR\_INVALID\_SETTINGS*** Setting set to camera is invalid.  
***PGRERROR\_INVALID\_BUS\_MANAGER*** Invalid Bus Manager object.  
***PGRERROR\_MEMORY\_ALLOCATION\_FAILED*** Could not allocate memory.  
***PGRERROR\_LOW\_LEVEL\_FAILURE*** Low level error.  
***PGRERROR\_NOT\_FOUND*** Device not found.  
***PGRERROR\_FAILED\_GUID*** GUID failure.  
***PGRERROR\_INVALID\_PACKET\_SIZE*** Packet size set to camera is invalid.  
***PGRERROR\_INVALID\_MODE*** Invalid mode has been passed to function.  
***PGRERROR\_NOT\_IN\_FORMAT7*** [Error](#) due to not being in Format7.  
***PGRERROR\_NOT\_SUPPORTED*** This feature is unsupported.  
***PGRERROR\_TIMEOUT*** Timeout error.  
***PGRERROR\_BUS\_MASTER\_FAILED*** Bus Master Failure.  
***PGRERROR\_INVALID\_GENERATION*** Generation Count Mismatch.  
***PGRERROR\_LUT\_FAILED*** Look Up Table failure.  
***PGRERROR\_IIDC\_FAILED*** IIDC failure.  
***PGRERROR\_STROBE\_FAILED*** Strobe failure.  
***PGRERROR\_TRIGGER\_FAILED*** Trigger failure.

***PGRERROR\_PROPERTY\_FAILED*** [Property](#) failure.

***PGRERROR\_PROPERTY\_NOT\_PRESENT*** [Property](#) is not present.

***PGRERROR\_REGISTER\_FAILED*** Register access failed.

***PGRERROR\_READ\_REGISTER\_FAILED*** Register read failed.

***PGRERROR\_WRITE\_REGISTER\_FAILED*** Register write failed.

***PGRERROR\_ISOCH\_FAILED*** Isochronous failure.

***PGRERROR\_ISOCH\_ALREADY\_STARTED*** Isochronous transfer has already been started.

***PGRERROR\_ISOCH\_NOT\_STARTED*** Isochronous transfer has not been started.

***PGRERROR\_ISOCH\_START\_FAILED*** Isochronous start failed.

***PGRERROR\_ISOCH\_RETRIEVE\_BUFFER\_FAILED*** Isochronous retrieve buffer failed.

***PGRERROR\_ISOCH\_STOP\_FAILED*** Isochronous stop failed.

***PGRERROR\_ISOCH\_SYNC\_FAILED*** Isochronous image synchronization failed.

***PGRERROR\_ISOCH\_BANDWIDTH\_EXCEEDED*** Isochronous bandwidth exceeded.

***PGRERROR\_IMAGE\_CONVERSION\_FAILED*** [Image](#) conversion failed.

***PGRERROR\_IMAGE\_LIBRARY\_FAILURE*** [Image](#) library failure.

***PGRERROR\_BUFFER\_TOO\_SMALL*** Buffer is too small.

***PGRERROR\_IMAGE\_CONSISTENCY\_ERROR*** There is an image consistency error.

***PGRERROR\_FORCE\_32BITS***

#### 6.2.1.8 enum FrameRate

Frame rates in frames per second.

Enumerator:

***FRAMERATE\_1\_875*** 1.875 fps.

***FRAMERATE\_3\_75*** 3.75 fps.

***FRAMERATE\_7\_5*** 7.5 fps.

***FRAMERATE\_15*** 15 fps.

***FRAMERATE\_30*** 30 fps.

***FRAMERATE\_60*** 60 fps.

***FRAMERATE\_120*** 120 fps.

***FRAMERATE\_240*** 240 fps.

***FRAMERATE\_FORMAT7*** Custom frame rate for Format7 functionality.

***NUM\_FRAMERATES*** Number of possible camera frame rates.

***FRAMERATE\_FORCE\_32BITS***

### 6.2.1.9 enum GrabMode

The grab strategy employed during image transfer.

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

**Enumerator:**

***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.

***GRAB\_MODE\_FORCE\_32BITS***

### 6.2.1.10 enum GrabTimeout

Timeout options for grabbing images.

**Enumerator:**

***TIMEOUT\_NONE*** Non-blocking wait.

***TIMEOUT\_INFINITE*** Wait indefinitely.

***TIMEOUT\_UNSPECIFIED*** Unspecified timeout setting.

***GRAB\_TIMEOUT\_FORCE\_32BITS***

### 6.2.1.11 enum ImageFileFormat

File formats to be used for saving images to disk.

**Enumerator:**

***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.

*IMAGE\_FILE\_FORMAT\_FORCE\_32BITS*

#### 6.2.1.12 enum InterfaceType

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

**Enumerator:**

*INTERFACE\_IEEE1394* IEEE-1394 (Includes 1394a and 1394b).

*INTERFACE\_USB2* USB 2.0.

*INTERFACE\_GIGE* GigE.

*INTERFACE\_UNKNOWN* Unknown interface.

*INTERFACE\_TYPE\_FORCE\_32BITS*

#### 6.2.1.13 enum Mode

[Camera](#) modes for DCAM formats as well as Format7.

**Enumerator:**

*MODE\_0*

*MODE\_1*

*MODE\_2*

*MODE\_3*

*MODE\_4*

*MODE\_5*

*MODE\_6*

*MODE\_7*

*MODE\_8*

*MODE\_9*

*MODE\_10*

*MODE\_11*

*MODE\_12*

*MODE\_13*

*MODE\_14*

*MODE\_15*

*MODE\_16*

*MODE\_17*

*MODE\_18*

*MODE\_19*

*MODE\_20*



**MODE\_21**

**MODE\_22**

**MODE\_23**

**MODE\_24**

**MODE\_25**

**MODE\_26**

**MODE\_27**

**MODE\_28**

**MODE\_29**

**MODE\_30**

**MODE\_31**

**NUM\_MODES** Number of modes.

**MODE\_FORCE\_32BITS**

#### 6.2.1.14 enum PixelFormat

Pixel formats available for Format7 modes.

**Enumerator:**

**PIXEL\_FORMAT\_MONO8** 8 bits of mono information.

**PIXEL\_FORMAT\_411YUV8** YUV 4:1:1.

**PIXEL\_FORMAT\_422YUV8** YUV 4:2:2.

**PIXEL\_FORMAT\_444YUV8** YUV 4:4:4.

**PIXEL\_FORMAT\_RGB8** R = G = B = 8 bits.

**PIXEL\_FORMAT\_MONO16** 16 bits of mono information.

**PIXEL\_FORMAT\_RGB16** R = G = B = 16 bits.

**PIXEL\_FORMAT\_S\_MONO16** 16 bits of signed mono information.

**PIXEL\_FORMAT\_S\_RGB16** R = G = B = 16 bits signed.

**PIXEL\_FORMAT\_RAW8** 8 bit raw data output of sensor.

**PIXEL\_FORMAT\_RAW16** 16 bit raw data output of sensor.

**PIXEL\_FORMAT\_MONO12** 12 bits of mono information.

**PIXEL\_FORMAT\_RAW12** 12 bit raw data output of sensor.

**PIXEL\_FORMAT\_BGR** 24 bit BGR.

**PIXEL\_FORMAT\_BGRU** 32 bit BGRU.

**PIXEL\_FORMAT\_RGB** 24 bit RGB.

**PIXEL\_FORMAT\_RGBU** 32 bit RGBU.

**NUM\_PIXEL\_FORMATS** Number of pixel formats.

**UNSPECIFIED\_PIXEL\_FORMAT** Unspecified pixel format.

### 6.2.1.15 enum PropertyType

Camera properties.

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

Enumerator:

**BRIGHTNESS** Brightness.  
**AUTO\_EXPOSURE** Auto exposure.  
**SHARPNESS** Sharpness.  
**WHITE\_BALANCE** White balance.  
**HUE** Hue.  
**SATURATION** Saturation.  
**GAMMA** Gamma.  
**IRIS** Iris.  
**FOCUS** Focus.  
**ZOOM** Zoom.  
**PAN** Pan.  
**TILT** Tilt.  
**SHUTTER** Shutter.  
**GAIN** Gain.  
**TRIGGER\_MODE** Trigger mode.  
**TRIGGER\_DELAY** Trigger delay.  
**FRAME\_RATE** Frame rate.  
**TEMPERATURE** Temperature.  
**UNSPECIFIED\_PROPERTY\_TYPE** Unspecified property type.  
**PROPERTY\_TYPE\_FORCE\_32BITS**

### 6.2.1.16 enum VideoMode

DCAM video modes.

Enumerator:

**VIDEOMODE\_160x120YUV444** 160x120 YUV444.  
**VIDEOMODE\_320x240YUV422** 320x240 YUV422.  
**VIDEOMODE\_640x480YUV411** 640x480 YUV411.  
**VIDEOMODE\_640x480YUV422** 640x480 YUV422.  
**VIDEOMODE\_640x480RGB** 640x480 24-bit RGB.  
**VIDEOMODE\_640x480Y8** 640x480 8-bit.  
**VIDEOMODE\_640x480Y16** 640x480 16-bit.  
**VIDEOMODE\_800x600YUV422** 800x600 YUV422.  
**VIDEOMODE\_800x600RGB** 800x600 RGB.  
**VIDEOMODE\_800x600Y8** 800x600 8-bit.

***VIDEOMODE\_800x600Y16*** 800x600 16-bit.  
***VIDEOMODE\_1024x768YUV422*** 1024x768 YUV422.  
***VIDEOMODE\_1024x768RGB*** 1024x768 RGB.  
***VIDEOMODE\_1024x768Y8*** 1024x768 8-bit.  
***VIDEOMODE\_1024x768Y16*** 1024x768 16-bit.  
***VIDEOMODE\_1280x960YUV422*** 1280x960 YUV422.  
***VIDEOMODE\_1280x960RGB*** 1280x960 RGB.  
***VIDEOMODE\_1280x960Y8*** 1280x960 8-bit.  
***VIDEOMODE\_1280x960Y16*** 1280x960 16-bit.  
***VIDEOMODE\_1600x1200YUV422*** 1600x1200 YUV422.  
***VIDEOMODE\_1600x1200RGB*** 1600x1200 RGB.  
***VIDEOMODE\_1600x1200Y8*** 1600x1200 8-bit.  
***VIDEOMODE\_1600x1200Y16*** 1600x1200 16-bit.  
***VIDEOMODE\_FORMAT7*** Custom video mode for Format7 functionality.  
***NUM\_VIDEOMODES*** Number of possible video modes.  
***VIDEOMODE\_FORCE\_32BITS***

## 6.3 GigE specific enumerations

These enumerations are specific to GigE camera operation only.

### Enumerations

- `enum GigEPropertyType {`  
    `HEARTBEAT,`  
    `HEARTBEAT_TIMEOUT,`  
    `PACKET_SIZE,`  
    `PACKET_DELAY }`

*Possible properties that can be queried from the camera.*

### 6.3.1 Detailed Description

These enumerations are specific to GigE camera operation only.

### 6.3.2 Enumeration Type Documentation

#### 6.3.2.1 `enum GigEPropertyType`

Possible properties that can be queried from the camera.

#### Enumerator:

***HEARTBEAT***

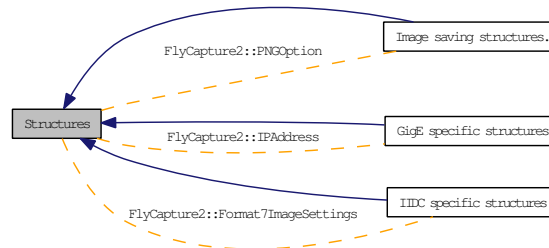
***HEARTBEAT\_TIMEOUT***

***PACKET\_SIZE***

***PACKET\_DELAY***

## 6.4 Structures

Collaboration diagram for Structures:



### Classes

- struct [FC2Version](#)  
*The current version of the library.*
- class [PGRGuid](#)  
*A GUID to the camera.*
- struct [IPAddress](#)  
*IPv4 address.*
- struct [Format7ImageSettings](#)  
*Format 7 image settings.*
- struct [FC2Config](#)  
*Configuration for a camera.*
- struct [PropertyInfo](#)  
*Information about a specific camera property.*
- struct [Property](#)  
*A specific camera property.*
- struct [TriggerModeInfo](#)  
*Information about a camera trigger property.*
- struct [TriggerMode](#)  
*A camera trigger.*
- struct [StrobeInfo](#)  
*A camera strobe property.*
- struct [StrobeControl](#)  
*A camera strobe.*
- struct [TimeStamp](#)

*Timestamp information.*

- struct [ConfigROM](#)  
*Camera configuration ROM.*
- struct [CameraInfo](#)  
*Camera information.*
- struct [EmbeddedImageInfoProperty](#)  
*Properties of a single embedded image info property.*
- struct [EmbeddedImageInfo](#)  
*Properties of the possible embedded image information.*
- struct [ImageMetadata](#)  
*Metadata related to an image.*
- struct [LUTData](#)  
*Information about the camera's look up table.*
- struct [HostAdapterStats](#)  
*Information about the host adapter's statistics.*
- struct [CameraStats](#)  
*Camera diagnostic information.*
- struct [PNGOption](#)  
*Options for saving PNG images.*

## Modules

- [GigE specific structures](#)  
*These structures are specific to GigE camera operation only.*
- [IIDC specific structures](#)  
*These structures are specific to IIDC camera operation only.*
- [Image saving structures.](#)  
*These structures define various parameters used for saving images.*

## Typedefs

- typedef PropertyInfo [TriggerDelayInfo](#)  
*The TriggerDelayInfo structure is identical to [PropertyInfo](#).*
- typedef Property [TriggerDelay](#)  
*The TriggerDelay structure is identical to [Property](#).*

## 6.4.1 Typedef Documentation

### 6.4.1.1 typedef Property TriggerDelay

The TriggerDelay structure is identical to [Property](#).

### 6.4.1.2 typedef PropertyInfo TriggerDelayInfo

The TriggerDelayInfo structure is identical to [PropertyInfo](#).

## 6.5 GigE specific structures

These structures are specific to GigE camera operation only.

Collaboration diagram for GigE specific structures:



### Classes

- struct [IPAddress](#)  
*IPv4 address.*
- struct [MACAddress](#)  
*MAC address.*
- struct [GigEProperty](#)  
*A GigE property.*
- struct [GigEStreamChannel](#)  
*Information about a single GigE stream channel.*
- struct [GigEConfig](#)  
*Configuration for a GigE camera.*
- struct [GigEImageSettingsInfo](#)  
*Format 7 information for a single mode.*
- struct [GigEImageSettings](#)  
*Image settings for a GigE camera.*

### 6.5.1 Detailed Description

These structures are specific to GigE camera operation only.



## 6.6 IIDC specific structures

These structures are specific to IIDC camera operation only.

Collaboration diagram for IIDC specific structures:



### Classes

- struct [Format7ImageSettings](#)  
*Format 7 image settings.*
- struct [Format7Info](#)  
*Format 7 information for a single mode.*
- struct [Format7PacketInfo](#)  
*Format 7 packet information.*

### 6.6.1 Detailed Description

These structures are specific to IIDC camera operation only.

## 6.7 Image saving structures.

These structures define various parameters used for saving images.

Collaboration diagram for Image saving structures.:



### Classes

- struct [PNGOption](#)  
*Options for saving PNG images.*
- struct [PPMOption](#)  
*Options for saving PPM images.*
- struct [PGMOption](#)  
*Options for saving PGM images.*
- struct [TIFFOption](#)  
*Options for saving TIFF images.*
- struct [JPEGOption](#)  
*Options for saving JPEG image.*
- struct [JPG2Option](#)  
*Options for saving JPEG2000 image.*
- struct [AVIOption](#)  
*Options for saving AVI files.*

### 6.7.1 Detailed Description

These structures define various parameters used for saving images.

# Chapter 7

## Namespace Documentation

### 7.1 FlyCapture2 Namespace Reference

#### Classes

- class [AVIRecorder](#)

*The [AVIRecorder](#) class provides the functionality for the user to record images to an AVI file.*

- class [BusManager](#)

*The [BusManager](#) class provides the functionality for the user to get an [PGRGuid](#) for a desired camera or device easily.*

- class [Camera](#)

*The [Camera](#) object represents a physical camera that uses the IIDC register set.*

- class [CameraBase](#)

*The [CameraBase](#) class is an abstract base class that defines a general interface to a camera.*

- class [Error](#)

*The [Error](#) object represents an error that is returned from the library.*

- struct [FC2Version](#)

*The current version of the library.*

- class [PGRGuid](#)

*A GUID to the camera.*

- struct [IPAddress](#)

*IPv4 address.*

- struct [MACAddress](#)

*MAC address.*

- struct [GigEProperty](#)

*A GigE property.*

- struct [GigEStreamChannel](#)  
*Information about a single GigE stream channel.*
- struct [GigEConfig](#)  
*Configuration for a GigE camera.*
- struct [GigEImageSettingsInfo](#)  
*Format 7 information for a single mode.*
- struct [GigEImageSettings](#)  
*Image settings for a GigE camera.*
- struct [Format7ImageSettings](#)  
*Format 7 image settings.*
- struct [Format7Info](#)  
*Format 7 information for a single mode.*
- struct [Format7PacketInfo](#)  
*Format 7 packet information.*
- struct [FC2Config](#)  
*Configuration for a camera.*
- struct [PropertyInfo](#)  
*Information about a specific camera property.*
- struct [Property](#)  
*A specific camera property.*
- struct [TriggerModeInfo](#)  
*Information about a camera trigger property.*
- struct [TriggerMode](#)  
*A camera trigger.*
- struct [StrobeInfo](#)  
*A camera strobe property.*
- struct [StrobeControl](#)  
*A camera strobe.*
- struct [TimeStamp](#)  
*Timestamp information.*
- struct [ConfigROM](#)  
*Camera configuration ROM.*
- struct [CameraInfo](#)

*Camera information.*

- struct [EmbeddedImageInfoProperty](#)  
*Properties of a single embedded image info property.*
- struct [EmbeddedImageInfo](#)  
*Properties of the possible embedded image information.*
- struct [ImageMetadata](#)  
*Metadata related to an image.*
- struct [LUTData](#)  
*Information about the camera's look up table.*
- struct [HostAdapterStats](#)  
*Information about the host adapter's statistics.*
- struct [CameraStats](#)  
*Camera diagnostic information.*
- struct [PNGOption](#)  
*Options for saving PNG images.*
- struct [PPMOption](#)  
*Options for saving PPM images.*
- struct [PGMOption](#)  
*Options for saving PGM images.*
- struct [TIFFOption](#)  
*Options for saving TIFF images.*
- struct [JPEGOption](#)  
*Options for saving JPEG image.*
- struct [JPG2Option](#)  
*Options for saving JPEG2000 image.*
- struct [AVIOption](#)  
*Options for saving AVI files.*
- class [CameraControlDlg](#)  
*The [CameraControlDlg](#) object represents a GTKmm dialog that provides a graphical interface to a specified camera.*
- class [CameraSelectionDlg](#)  
*The [CameraSelectionDlg](#) object represents a GTKmm dialog that provides a graphical interface that lists the number of cameras available to the library.*
- class [GigECamera](#)

The [GigECamera](#) object represents a physical Gigabit Ethernet camera.

- class [Image](#)

The [Image](#) class is used to retrieve images from a camera, convert between multiple pixel formats and save images to disk.

- class [ImageStatistics](#)

The [ImageStatistics](#) object represents image statistics for an image.

- class [TopologyNode](#)

The [TopologyNode](#) class contains topology information that can be used to generate a tree structure of all cameras and devices connected to a computer.

- struct [SystemInfo](#)

Description of the system.

- class [Utilities](#)

The [Utility](#) class is generally used to query for general system information such as operating system, available memory etc.

## Typedefs

- typedef void(\* [BusEventCallback](#) )(void \*pParameter, unsigned int serialNumber)

[Bus](#) event callback function prototype.

- typedef void \* [CallbackHandle](#)

Handle that is returned when registering a callback.

- typedef void(\* [ImageEventCallback](#) )(class [Image](#) \*pImage, const void \*pCallbackData)

[Image](#) event callback function prototype.

- typedef [PropertyInfo](#) [TriggerDelayInfo](#)

The [TriggerDelayInfo](#) structure is identical to [PropertyInfo](#).

- typedef [Property](#) [TriggerDelay](#)

The [TriggerDelay](#) structure is identical to [Property](#).

- typedef void(\* [AsyncCommandCallback](#) )(class [Error](#) retError, void \*pUserData)

[Async](#) command callback function prototype.

## Enumerations

- enum [ErrorType](#) {  
[PGRERROR\\_UNDEFINED](#) = -1,  
[PGRERROR\\_OK](#),  
[PGRERROR\\_FAILED](#),  
[PGRERROR\\_NOT\\_IMPLEMENTED](#),

PGRERROR\_FAILED\_BUS\_MASTER\_CONNECTION,  
PGRERROR\_NOT\_CONNECTED,  
PGRERROR\_INIT\_FAILED,  
PGRERROR\_NOT\_INITIALIZED,  
PGRERROR\_INVALID\_PARAMETER,  
PGRERROR\_INVALID\_SETTINGS,  
PGRERROR\_INVALID\_BUS\_MANAGER,  
PGRERROR\_MEMORY\_ALLOCATION\_FAILED,  
PGRERROR\_LOW\_LEVEL\_FAILURE,  
PGRERROR\_NOT\_FOUND,  
PGRERROR\_FAILED\_GUID,  
PGRERROR\_INVALID\_PACKET\_SIZE,  
PGRERROR\_INVALID\_MODE,  
PGRERROR\_NOT\_IN\_FORMAT7,  
PGRERROR\_NOT\_SUPPORTED,  
PGRERROR\_TIMEOUT,  
PGRERROR\_BUS\_MASTER\_FAILED,  
PGRERROR\_INVALID\_GENERATION,  
PGRERROR\_LUT\_FAILED,  
PGRERROR\_IIDC\_FAILED,  
PGRERROR\_STROBE\_FAILED,  
PGRERROR\_TRIGGER\_FAILED,  
PGRERROR\_PROPERTY\_FAILED,  
PGRERROR\_PROPERTY\_NOT\_PRESENT,  
PGRERROR\_REGISTER\_FAILED,  
PGRERROR\_READ\_REGISTER\_FAILED,  
PGRERROR\_WRITE\_REGISTER\_FAILED,  
PGRERROR\_ISOCH\_FAILED,  
PGRERROR\_ISOCH\_ALREADY\_STARTED,  
PGRERROR\_ISOCH\_NOT\_STARTED,  
PGRERROR\_ISOCH\_START\_FAILED,  
PGRERROR\_ISOCH\_RETRIEVE\_BUFFER\_FAILED,  
PGRERROR\_ISOCH\_STOP\_FAILED,  
PGRERROR\_ISOCH\_SYNC\_FAILED,  
PGRERROR\_ISOCH\_BANDWIDTH\_EXCEEDED,  
PGRERROR\_IMAGE\_CONVERSION\_FAILED,  
PGRERROR\_IMAGE\_LIBRARY\_FAILURE,  
PGRERROR\_BUFFER\_TOO\_SMALL,  
PGRERROR\_IMAGE\_CONSISTENCY\_ERROR,  
PGRERROR\_FORCE\_32BITS = FULL\_32BIT\_VALUE }

*The error types returned by functions.*

- enum `BusCallbackType` {  
    `BUS_RESET`,  
    `ARRIVAL`,  
    `REMOVAL`,  
    `CALLBACK_TYPE_FORCE_32BITS` = `FULL_32BIT_VALUE` }  
    *The type of bus callback to register a callback function for.*
- enum `GrabMode` {  
    `DROP_FRAMES`,  
    `BUFFER_FRAMES`,  
    `UNSPECIFIED_GRAB_MODE`,  
    `GRAB_MODE_FORCE_32BITS` = `FULL_32BIT_VALUE` }  
    *The grab strategy employed during image transfer.*
- enum `GrabTimeout` {  
    `TIMEOUT_NONE` = 0,  
    `TIMEOUT_INFINITE` = -1,  
    `TIMEOUT_UNSPECIFIED` = -2,  
    `GRAB_TIMEOUT_FORCE_32BITS` = `FULL_32BIT_VALUE` }  
    *Timeout options for grabbing images.*
- enum `BandwidthAllocation` {  
    `BANDWIDTH_ALLOCATION_OFF` = 0,  
    `BANDWIDTH_ALLOCATION_ON` = 1,  
    `BANDWIDTH_ALLOCATION_UNSUPPORTED` = 2,  
    `BANDWIDTH_ALLOCATION_UNSPECIFIED` = 3,  
    `BANDWIDTH_ALLOCATION_FORCE_32BITS` = `FULL_32BIT_VALUE` }  
    *Bandwidth allocation options for 1394 devices.*
- enum `InterfaceType` {  
    `INTERFACE_IEEE1394`,  
    `INTERFACE_USB2`,  
    `INTERFACE_GIGE`,  
    `INTERFACE_UNKNOWN`,  
    `INTERFACE_TYPE_FORCE_32BITS` = `FULL_32BIT_VALUE` }  
    *Interfaces that a camera may use to communicate with a host.*
- enum `PropertyType` {  
    `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,  
PROPERTY\_TYPE\_FORCE\_32BITS = FULL\_32BIT\_VALUE }

*Camera properties.*

- enum FrameRate {  
FRAMERATE\_1\_875,  
FRAMERATE\_3\_75,  
FRAMERATE\_7\_5,  
FRAMERATE\_15,  
FRAMERATE\_30,  
FRAMERATE\_60,  
FRAMERATE\_120,  
FRAMERATE\_240,  
FRAMERATE\_FORMAT7,  
NUM\_FRAMERATES,  
FRAMERATE\_FORCE\_32BITS = FULL\_32BIT\_VALUE }

*Frame rates in frames per second.*

- enum VideoMode {  
VIDEOMODE\_160x120YUV444,  
VIDEOMODE\_320x240YUV422,  
VIDEOMODE\_640x480YUV411,  
VIDEOMODE\_640x480YUV422,  
VIDEOMODE\_640x480RGB,  
VIDEOMODE\_640x480Y8,  
VIDEOMODE\_640x480Y16,  
VIDEOMODE\_800x600YUV422,  
VIDEOMODE\_800x600RGB,  
VIDEOMODE\_800x600Y8,

```

VIDEOMODE_800x600Y16,
VIDEOMODE_1024x768YUV422,
VIDEOMODE_1024x768RGB,
VIDEOMODE_1024x768Y8,
VIDEOMODE_1024x768Y16,
VIDEOMODE_1280x960YUV422,
VIDEOMODE_1280x960RGB,
VIDEOMODE_1280x960Y8,
VIDEOMODE_1280x960Y16,
VIDEOMODE_1600x1200YUV422,
VIDEOMODE_1600x1200RGB,
VIDEOMODE_1600x1200Y8,
VIDEOMODE_1600x1200Y16,
VIDEOMODE_FORMAT7,
NUM_VIDEOMODES,
VIDEOMODE_FORCE_32BITS = FULL_32BIT_VALUE }

```

*DCAM video modes.*

- enum Mode {
  - MODE\_0 = 0,
  - MODE\_1,
  - MODE\_2,
  - MODE\_3,
  - MODE\_4,
  - MODE\_5,
  - MODE\_6,
  - MODE\_7,
  - MODE\_8,
  - MODE\_9,
  - MODE\_10,
  - MODE\_11,
  - MODE\_12,
  - MODE\_13,
  - MODE\_14,
  - MODE\_15,
  - MODE\_16,
  - MODE\_17,
  - MODE\_18,
  - MODE\_19,
  - MODE\_20,
  - MODE\_21,

```
MODE_22,  
MODE_23,  
MODE_24,  
MODE_25,  
MODE_26,  
MODE_27,  
MODE_28,  
MODE_29,  
MODE_30,  
MODE_31,  
NUM_MODES,  
MODE_FORCE_32BITS = FULL_32BIT_VALUE }
```

*Camera modes for DCAM formats as well as Format7.*

- enum PixelFormat {  
PIXEL\_FORMAT\_MONO8 = 0x80000000,  
PIXEL\_FORMAT\_411YUV8 = 0x40000000,  
PIXEL\_FORMAT\_422YUV8 = 0x20000000,  
PIXEL\_FORMAT\_444YUV8 = 0x10000000,  
PIXEL\_FORMAT\_RGB8 = 0x08000000,  
PIXEL\_FORMAT\_MONO16 = 0x04000000,  
PIXEL\_FORMAT\_RGB16 = 0x02000000,  
PIXEL\_FORMAT\_S\_MONO16 = 0x01000000,  
PIXEL\_FORMAT\_S\_RGB16 = 0x00800000,  
PIXEL\_FORMAT\_RAW8 = 0x00400000,  
PIXEL\_FORMAT\_RAW16 = 0x00200000,  
PIXEL\_FORMAT\_MONO12 = 0x00100000,  
PIXEL\_FORMAT\_RAW12 = 0x00080000,  
PIXEL\_FORMAT\_BGR = 0x80000008,  
PIXEL\_FORMAT\_BGRU = 0x40000008,  
PIXEL\_FORMAT\_RGB = PIXEL\_FORMAT\_RGB8,  
PIXEL\_FORMAT\_RGBU = 0x40000002,  
NUM\_PIXEL\_FORMATS = 15,  
UNSPECIFIED\_PIXEL\_FORMAT = 0 }

*Pixel formats available for Format7 modes.*

- enum BusSpeed {  
BUSSPEED\_S100,  
BUSSPEED\_S200,  
BUSSPEED\_S400,  
BUSSPEED\_S480,  
BUSSPEED\_S800,

```

BUSSPEED_S1600,
BUSSPEED_S3200,
BUSSPEED_S5000,
BUSSPEED_10BASE_T,
BUSSPEED_100BASE_T,
BUSSPEED_1000BASE_T,
BUSSPEED_10000BASE_T,
BUSSPEED_S_FASTEST,
BUSSPEED_ANY,
BUSSPEED_SPEED_UNKNOWN = -1,
BUSSPEED_FORCE_32BITS = FULL_32BIT_VALUE }

```

*Bus speeds.*

- enum `DriverType` {

```

DRIVER_1394_CAM,
DRIVER_1394_PRO,
DRIVER_1394_JUJU,
DRIVER_1394_VIDEO1394,
DRIVER_1394_RAW1394,
DRIVER_USB_NONE,
DRIVER_USB_CAM,
DRIVER_USB3_PRO,
DRIVER_GIGE_NONE,
DRIVER_GIGE_FILTER,
DRIVER_GIGE_PRO,
DRIVER_UNKNOWN = -1,
DRIVER_FORCE_32BITS = FULL_32BIT_VALUE }

```

*Types of low level drivers that flycapture uses.*

- enum `ColorProcessingAlgorithm` {

```

DEFAULT,
NO_COLOR_PROCESSING,
NEAREST_NEIGHBOR,
EDGE_SENSING,
HQ_LINEAR,
RIGOROUS,
IPP,
COLOR_PROCESSING_ALGORITHM_FORCE_32BITS = FULL_32BIT_VALUE }

```

*Color processing algorithms.*

- enum `BayerTileFormat` {  
    `NONE`,  
    `RGBB`,  
    `GRBG`,  
    `GBRG`,  
    `BGGR`,  
    `BT_FORCE_32BITS` = `FULL_32BIT_VALUE` }  
    *Bayer tile formats.*
- enum `ImageFileFormat` {  
    `FROM_FILE_EXT` = -1,  
    `PGM`,  
    `PPM`,  
    `BMP`,  
    `JPEG`,  
    `JPEG2000`,  
    `TIFF`,  
    `PNG`,  
    `RAW`,  
    `IMAGE_FILE_FORMAT_FORCE_32BITS` = `FULL_32BIT_VALUE` }  
    *File formats to be used for saving images to disk.*
- enum `GigEPropertyType` {  
    `HEARTBEAT`,  
    `HEARTBEAT_TIMEOUT`,  
    `PACKET_SIZE`,  
    `PACKET_DELAY` }  
    *Possible properties that can be queried from the camera.*
- enum `OSType` {  
    `WINDOWS_X86`,  
    `WINDOWS_X64`,  
    `LINUX_X86`,  
    `LINUX_X64`,  
    `MAC`,  
    `UNKNOWN_OS`,  
    `OSTYPE_FORCE_32BITS` = `FULL_32BIT_VALUE` }  
    *Possible operating systems.*
- enum `ByteOrder` {  
    `BYTE_ORDER_LITTLE_ENDIAN`,  
    `BYTE_ORDER_BIG_ENDIAN`,  
    `BYTE_ORDER_FORCE_32BITS` = `FULL_32BIT_VALUE` }  
    *Possible byte orders.*

## Variables

- static const unsigned int `sk_maxStringLength` = 512  
*The maximum length that is allocated for a string.*
- static const unsigned int `sk_maxNumPorts` = 32  
*The maximum number of ports one device can have.*

## 7.1.1 Typedef Documentation

### 7.1.1.1 typedef void(\* AsyncCommandCallback)(class Error retError, void \*pUserData)

Async command callback function prototype.

Defines the syntax of the async command function that is passed into LaunchCommandAsync().

### 7.1.1.2 typedef void(\* BusEventCallback)(void \*pParameter, unsigned int serialNumber)

Bus event callback function prototype.

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

### 7.1.1.3 typedef void\* CallbackHandle

Handle that is returned when registering a callback.

It is required when unregistering the callback.

### 7.1.1.4 typedef void(\* ImageEventCallback)(class Image \*pImage, const void \*pCallbackData)

[Image](#) event callback function prototype.

Defines the syntax of the image callback function that is passed into StartCapture(). It is possible for this function to be called simultaneously. Therefore, users must make sure that code in the callback is thread safe.

## 7.1.2 Enumeration Type Documentation

### 7.1.2.1 enum ByteOrder

Possible byte orders.

Enumerator:

*BYTE\_ORDER\_LITTLE\_ENDIAN*  
*BYTE\_ORDER\_BIG\_ENDIAN*  
*BYTE\_ORDER\_FORCE\_32BITS*

### 7.1.2.2 enum OSType

Possible operating systems.

**Enumerator:**

***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.

***OSTYPE\_FORCE\_32BITS***





# Chapter 8

## Class Documentation

### 8.1 AVIOption Struct Reference

Options for saving AVI files.

#### Public Member Functions

- [AVIOption\(\)](#)

#### Public Attributes

- float [frameRate](#)  
*Frame rate of the stream.*
- unsigned int [reserved](#) [256]  
*Reserved for future use.*

#### 8.1.1 Detailed Description

Options for saving AVI files.

#### 8.1.2 Constructor & Destructor Documentation

##### 8.1.2.1 [AVIOption\(\)](#) [inline]

#### 8.1.3 Member Data Documentation

##### 8.1.3.1 float [frameRate](#)

Frame rate of the stream.

### 8.1.3.2 unsigned int reserved[256]

Reserved for future use.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.2 AVIRecorder Class Reference

The [AVIRecorder](#) class provides the functionality for the user to record images to an AVI file.

### Public Member Functions

- [AVIRecorder](#) ()  
*Default constructor.*
- virtual [~AVIRecorder](#) ()  
*Default destructor.*
- virtual [Error AVIOpen](#) (const char \*pFileName, [AVIOption](#) \*pOption)  
*Open an AVI file in preparation for writing Images to disk.*
- virtual [Error AVIAppend](#) ([Image](#) \*pImage)  
*Append an image to the AVI file.*
- virtual [Error AVIClose](#) ()  
*Close the AVI file.*

### 8.2.1 Detailed Description

The [AVIRecorder](#) class provides the functionality for the user to record images to an AVI file.

### 8.2.2 Constructor & Destructor Documentation

#### 8.2.2.1 AVIRecorder ()

Default constructor.

#### 8.2.2.2 virtual ~AVIRecorder () [virtual]

Default destructor.

### 8.2.3 Member Function Documentation

#### 8.2.3.1 virtual Error AVIAppend (Image \*pImage) [virtual]

Append an image to the AVI file.

##### Parameters:

*pImage* The image to append.

##### Returns:

An [Error](#) indicating the success or failure of the function.

### 8.2.3.2 virtual Error AVIClose () [virtual]

Close the AVI file.

See also:

[AVIOpen\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

### 8.2.3.3 virtual Error AVIOpen (const char \**pFileName*, AVIOption \**pOption*) [virtual]

Open an AVI file in preparation for writing Images to disk.

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

Parameters:

*pFileName* The filename of the AVI file.

*pOption* Options to apply to the AVI file.

See also:

[AVIClose\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

The documentation for this class was generated from the following file:

- [AVIRecorder.h](#)

## 8.3 BusManager Class Reference

The [BusManager](#) class provides the functionality for the user to get an [PGRGuid](#) for a desired camera or device easily.

### Public Member Functions

- [BusManager](#) ()  
*Default constructor.*
- virtual [~BusManager](#) ()  
*Default destructor.*
- virtual [Error FireBusReset](#) ([PGRGuid](#) \*pGuid)  
*Fire a bus reset.*
- virtual [Error GetNumOfCameras](#) (unsigned int \*pNumCameras)  
*Gets the number of cameras attached to the PC.*
- virtual [Error GetCameraFromIPAddress](#) ([IPAddress](#) ipAddress, [PGRGuid](#) \*pGuid)  
*Gets the [PGRGuid](#) for a camera with the specified IPv4 address.*
- virtual [Error GetCameraFromIndex](#) (unsigned int index, [PGRGuid](#) \*pGuid)  
*Gets the [PGRGuid](#) for a camera on the PC.*
- virtual [Error GetCameraFromSerialNumber](#) (unsigned int serialNumber, [PGRGuid](#) \*pGuid)  
*Gets the [PGRGuid](#) for a camera on the PC.*
- virtual [Error GetCameraSerialNumberFromIndex](#) (unsigned int index, unsigned int \*pSerialNumber)  
*Gets the serial number of the camera with the specified index.*
- virtual [Error GetInterfaceTypeFromGuid](#) ([PGRGuid](#) \*pGuid, [InterfaceType](#) \*pInterfaceType)  
*Gets the interface type associated with a [PGRGuid](#).*
- virtual [Error GetNumOfDevices](#) (unsigned int \*pNumDevices)  
*Gets the number of devices.*
- virtual [Error GetDeviceFromIndex](#) (unsigned int index, [PGRGuid](#) \*pGuid)  
*Gets the [PGRGuid](#) for a device.*
- virtual [Error ReadPhyRegister](#) ([PGRGuid](#) guid, unsigned int page, unsigned int port, unsigned int address, unsigned int \*pValue)  
*Read a phy register on the specified device.*
- virtual [Error WritePhyRegister](#) ([PGRGuid](#) guid, unsigned int page, unsigned int port, unsigned int address, unsigned int value)  
*Write a phy register on the specified device.*
- virtual [Error GetTopology](#) ([TopologyNode](#) \*pNode)

*Gets the topology information for the PC.*

- virtual [Error RegisterCallback](#) ([BusEventCallback](#) busEventCallback, [BusCallbackType](#) callbackType, void \*pParameter, [CallbackHandle](#) \*pCallbackHandle)

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

- virtual [Error UnregisterCallback](#) ([CallbackHandle](#) callbackHandle)

*Unregister a callback function.*

- virtual [Error RescanBus](#) ()

*Force a rescan of the buses.*

## Static Public Member Functions

- static [Error ForceIPAddressToCamera](#) ([MACAddress](#) macAddress, [IPAddress](#) ipAddress, [IPAddress](#) subnetMask, [IPAddress](#) defaultGateway)

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

- static [Error DiscoverGigECameras](#) ([CameraInfo](#) \*gigECameras, unsigned int \*arraySize)

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

### 8.3.1 Detailed Description

The [BusManager](#) class provides the functionality for the user to get an [PGRGuid](#) for a desired camera or device easily.

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

### 8.3.2 Constructor & Destructor Documentation

#### 8.3.2.1 [BusManager](#) ()

Default constructor.

#### 8.3.2.2 [virtual ~BusManager](#) () [virtual]

Default destructor.

### 8.3.3 Member Function Documentation

#### 8.3.3.1 [static Error DiscoverGigECameras](#) ([CameraInfo](#) \* *gigECameras*, unsigned int \* *arraySize*) [static]

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

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

**Parameters:**

*gigECameras* Pointer to an array of [CameraInfo](#) structures.

*arraySize* Size of the array. Number of discovered cameras is returned in the same value.

**Returns:**

An [Error](#) indicating the success or failure of the function. If the error is PGRERROR\_BUFFER\_TOO\_SMALL then arraySize will contain the minimum size needed for gigECameras array.

**8.3.3.2 virtual Error FireBusReset (PGRGuid \*pGuid) [virtual]**

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:**

*pGuid* [PGRGuid](#) of the camera or the device to cause bus reset.

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.3.3.3 static Error ForceIPAddressToCamera (MACAddress macAddress, IPAddress ipAddress, IPAddress subnetMask, IPAddress defaultGateway) [static]**

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

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

**Parameters:**

*macAddress* MAC address of the camera.

*ipAddress* IP address to set on the camera.

*subnetMask* Subnet mask to set on the camera.

*defaultGateway* Default gateway to set on the camera.

**Returns:**

An [Error](#) indicating the success or failure of the function.

#### 8.3.3.4 **virtual Error GetCameraFromIndex (unsigned int *index*, PGRGuid \* *pGuid*)** [virtual]

Gets the [PGRGuid](#) for a camera on the PC.

It uniquely identifies the camera specified by the index and is used to identify the camera during a [Camera::Connect\(\)](#) call.

##### Parameters:

*index* Zero based index of camera.

*pGuid* Unique [PGRGuid](#) for the camera.

##### See also:

[GetCameraFromSerialNumber\(\)](#)

##### Returns:

An [Error](#) indicating the success or failure of the function.

#### 8.3.3.5 **virtual Error GetCameraFromIPAddress (IPAddress *ipAddress*, PGRGuid \* *pGuid*)** [virtual]

Gets the [PGRGuid](#) for a camera with the specified IPv4 address.

##### Parameters:

*ipAddress* IP address to get GUID for.

*pGuid* Unique [PGRGuid](#) for the camera.

##### Returns:

An [Error](#) indicating the success or failure of the function.

#### 8.3.3.6 **virtual Error GetCameraFromSerialNumber (unsigned int *serialNumber*, PGRGuid \* *pGuid*)** [virtual]

Gets the [PGRGuid](#) for a camera on the PC.

It uniquely identifies the camera specified by the serial number and is used to identify the camera during a [Camera::Connect\(\)](#) call.

##### Parameters:

*serialNumber* Serial number of camera.

*pGuid* Unique [PGRGuid](#) for the camera.

##### See also:

[GetCameraFromIndex\(\)](#)

##### Returns:

An [Error](#) indicating the success or failure of the function.



### 8.3.3.7 virtual Error GetCameraSerialNumberFromIndex (unsigned int *index*, unsigned int \* *pSerialNumber*) [virtual]

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

#### Parameters:

*index* Zero based index of desired camera.

*pSerialNumber* Serial number of camera.

#### Returns:

An [Error](#) indicating the success or failure of the function.

### 8.3.3.8 virtual Error GetDeviceFromIndex (unsigned int *index*, PGRGuid \* *pGuid*) [virtual]

Gets the [PGRGuid](#) for a device.

It uniquely identifies the device specified by the index.

#### Parameters:

*index* Zero based index of device.

*pGuid* Unique [PGRGuid](#) for the device.

#### See also:

[GetNumOfDevices\(\)](#)

#### Returns:

An [Error](#) indicating the success or failure of the function.

### 8.3.3.9 virtual Error GetInterfaceTypeFromGuid (PGRGuid \* *pGuid*, InterfaceType \* *pInterfaceType*) [virtual]

Gets the interface type associated with a [PGRGuid](#).

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

#### Parameters:

*pGuid* The [PGRGuid](#) to get the interface for.

*pInterfaceType* The interface type of the [PGRGuid](#).

#### Returns:

An [Error](#) indicating the success or failure of the function.

**8.3.3.10 virtual Error GetNumOfCameras (unsigned int \* *pNumCameras*) [virtual]**

Gets the number of cameras attached to the PC.

**Parameters:**

*pNumCameras* The number of cameras attached.

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.3.3.11 virtual Error GetNumOfDevices (unsigned int \* *pNumDevices*) [virtual]**

Gets the number of devices.

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

**Parameters:**

*pNumDevices* The number of devices found.

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.3.3.12 virtual Error GetTopology (TopologyNode \* *pNode*) [virtual]**

Gets the topology information for the PC.

**Parameters:**

*pNode* [TopologyNode](#) object that will contain the topology information.

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.3.3.13 virtual Error ReadPhyRegister (PGRGuid *guid*, unsigned int *page*, unsigned int *port*, unsigned int *address*, unsigned int \* *pValue*) [virtual]**

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* [PGRGuid](#) of the device to read from.

*page* Page to read from.

*port* Port to read from.

*address* Address to read from.

*pValue* Value read from the phy register.

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.3.3.14 virtual Error RegisterCallback (BusEventCallback *busEventCallback*, BusCallbackType *callbackType*, void \**pParameter*, CallbackHandle \**pCallbackHandle*)** [virtual]

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

**Parameters:**

*busEventCallback* Pointer to function that will receive the callback.

*callbackType* Type of callback to register for.

*pParameter* Callback parameter to be passed to callback.

*pCallbackHandle* Unique callback handle used for unregistering callback.

**See also:**

[UnregisterCallback\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.3.3.15 virtual Error RescanBus ()** [virtual]

Force a rescan of the buses.

This does not trigger a bus reset. However, any current connections to a [Camera](#) object will be invalidated.

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.3.3.16 virtual Error UnregisterCallback (CallbackHandle *callbackHandle*)** [virtual]

Unregister a callback function.

**Parameters:**

*callbackHandle* Unique callback handle.

**See also:**

[RegisterCallback\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.3.3.17 virtual Error WritePhyRegister (PGRGuid *guid*, unsigned int *page*, unsigned int *port*, unsigned int *address*, unsigned int *value*)** [virtual]

Write a phy register on the specified device.

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

**Parameters:**

*guid* [PGRGuid](#) of the device to write to.

*page* Page to write to.

*port* Port to write to.

*address* Address to write to.

*value* Value to write to phy register.

**Returns:**

An [Error](#) indicating the success or failure of the function.

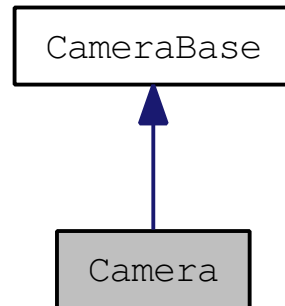
The documentation for this class was generated from the following file:

- [BusManager.h](#)

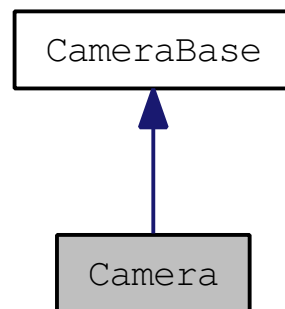
## 8.4 Camera Class Reference

The [Camera](#) object represents a physical camera that uses the IIDC register set.

Inheritance diagram for Camera:



Collaboration diagram for Camera:



### Public Member Functions

- [Camera](#) ()  
*Default constructor.*
- virtual [~Camera](#) ()  
*Default destructor.*
- virtual [Error Connect](#) ([PGRGuid](#) \*pGuid=NULL)  
*The following functions are inherited from [CameraBase](#).*
- virtual [Error Disconnect](#) ()  
*Disconnects the camera object from the camera.*
- virtual bool [IsConnected](#) ()  
*Checks if the camera object is currently connected to a physical camera.*
- virtual [Error SetCallback](#) ([ImageEventCallback](#) callbackFn, const void \*pCallbackData=NULL)

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

- virtual [Error StartCapture](#) ([ImageEventCallback](#) callbackFn=NULL, const void \*pCallbackData=NULL)

*Starts isochronous image capture.*

- virtual [Error RetrieveBuffer](#) ([Image](#) \*pImage)

*Retrieves the the next image object containing the next image.*

- virtual [Error StopCapture](#) ()

*Stops isochronous image transfer and cleans up all associated resources.*

- virtual [Error WaitForBufferEvent](#) ([Image](#) \*pImage, unsigned int eventNumber)

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

- virtual [Error SetUserBuffers](#) (unsigned char \*const pMemBuffers, int size, int numBuffers)

*Specify user allocated buffers to use as image data buffers.*

- virtual [Error GetConfiguration](#) ([FC2Config](#) \*pConfig)

*Get the configuration associated with the camera object.*

- virtual [Error SetConfiguration](#) (const [FC2Config](#) \*pConfig)

*Set the configuration associated with the camera object.*

- virtual [Error GetCameraInfo](#) ([CameraInfo](#) \*pCameraInfo)

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

- virtual [Error GetPropertyInfo](#) ([PropertyInfo](#) \*pPropInfo)

*Retrieves information about the specified camera property.*

- virtual [Error GetProperty](#) ([Property](#) \*pProp)

*Reads the settings for the specified property from the camera.*

- virtual [Error SetProperty](#) (const [Property](#) \*pProp, bool broadcast=false)

*Writes the settings for the specified property to the camera.*

- virtual [Error GetGPIOPinDirection](#) (unsigned int pin, unsigned int \*pDirection)

*Get the GPIO pin direction for the specified pin.*

- virtual [Error SetGPIOPinDirection](#) (unsigned int pin, unsigned int direction, bool broadcast=false)

*Set the GPIO pin direction for the specified pin.*

- virtual [Error GetTriggerModeInfo](#) ([TriggerModeInfo](#) \*pTriggerModeInfo)

*Retrieve trigger information from the camera.*

- virtual [Error GetTriggerMode](#) ([TriggerMode](#) \*pTriggerMode)

*Retrieve current trigger settings from the camera.*

- virtual [Error SetTriggerMode](#) (const [TriggerMode](#) \*pTriggerMode, bool broadcast=false)

*Set the specified trigger settings to the camera.*

- virtual [Error FireSoftwareTrigger](#) (bool broadcast=false)  
*Fire the software trigger according to the DCAM specifications.*
- virtual [Error GetTriggerDelayInfo](#) ([TriggerDelayInfo](#) \*pTriggerDelayInfo)  
*Retrieve trigger delay information from the camera.*
- virtual [Error GetTriggerDelay](#) ([TriggerDelay](#) \*pTriggerDelay)  
*Retrieve current trigger delay settings from the camera.*
- virtual [Error SetTriggerDelay](#) (const [TriggerDelay](#) \*pTriggerDelay, bool broadcast=false)  
*Set the specified trigger delay settings to the camera.*
- virtual [Error GetStrobeInfo](#) ([StrobeInfo](#) \*pStrobeInfo)  
*Retrieve strobe information from the camera.*
- virtual [Error GetStrobe](#) ([StrobeControl](#) \*pStrobeControl)  
*Retrieve current strobe settings from the camera.*
- virtual [Error SetStrobe](#) (const [StrobeControl](#) \*pStrobeControl, bool broadcast=false)  
*Set current strobe settings to the camera.*
- virtual [Error GetLUTInfo](#) ([LUTData](#) \*pData)  
*Query if LUT support is available on the camera.*
- virtual [Error GetLUTBankInfo](#) (unsigned int bank, bool \*pReadSupported, bool \*pWriteSupported)  
*Query the read/write status of a single LUT bank.*
- virtual [Error GetActiveLUTBank](#) (unsigned int \*pActiveBank)  
*Get the LUT bank that is currently being used.*
- virtual [Error SetActiveLUTBank](#) (unsigned int activeBank)  
*Set the LUT bank that will be used.*
- virtual [Error EnableLUT](#) (bool on)  
*Enable or disable LUT functionality on the camera.*
- virtual [Error GetLUTChannel](#) (unsigned int bank, unsigned int channel, unsigned int sizeEntries, unsigned int \*pEntries)  
*Get the LUT channel settings from the camera.*
- virtual [Error SetLUTChannel](#) (unsigned int bank, unsigned int channel, unsigned int sizeEntries, const unsigned int \*pEntries)  
*Set the LUT channel settings to the camera.*
- virtual [Error GetMemoryChannel](#) (unsigned int \*pCurrentChannel)  
*Retrieve the current memory channel from the camera.*
- virtual [Error SaveToMemoryChannel](#) (unsigned int channel)

*Save the current settings to the specified current memory channel.*

- virtual [Error RestoreFromMemoryChannel](#) (unsigned int channel)  
*Restore the specified current memory channel.*
- virtual [Error GetMemoryChannelInfo](#) (unsigned int \*pNumChannels)  
*Query the camera for memory channel support.*
- virtual [Error GetEmbeddedImageInfo](#) ([EmbeddedImageInfo](#) \*pInfo)  
*Get the current status of the embedded image information register, as well as the availability of each embedded property.*
- virtual [Error SetEmbeddedImageInfo](#) ([EmbeddedImageInfo](#) \*pInfo)  
*Sets the on/off values of the embedded image information structure to the camera.*
- virtual [Error WriteRegister](#) (unsigned int address, unsigned int value, bool broadcast=false)  
*Write to the specified register on the camera.*
- virtual [Error ReadRegister](#) (unsigned int address, unsigned int \*pValue)  
*Read the specified register from the camera.*
- virtual [Error WriteRegisterBlock](#) (unsigned short addressHigh, unsigned int addressLow, const unsigned int \*pBuffer, unsigned int length)  
*Write to the specified register block on the camera.*
- virtual [Error ReadRegisterBlock](#) (unsigned short addressHigh, unsigned int addressLow, unsigned int \*pBuffer, unsigned int length)  
*Read from the specified register block on the camera.*

## Static Public Member Functions

- static [Error StartSyncCapture](#) (unsigned int numCameras, const [Camera](#) \*\*ppCameras, const [ImageEventCallback](#) \*pCallbackFns=NULL, const void \*\*pCallbackDataArray=NULL)
- static const char \* [GetRegisterString](#) (unsigned int registerVal)  
*Returns a text representation of the register value.*

## DCAM Formats

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

- virtual [Error GetVideoModeAndFrameRateInfo](#) ([VideoMode](#) videoMode, [FrameRate](#) frameRate, bool \*pSupported)  
*Query the camera to determine if the specified video mode and frame rate is supported.*
- virtual [Error GetVideoModeAndFrameRate](#) ([VideoMode](#) \*pVideoMode, [FrameRate](#) \*pFrameRate)  
*Get the current video mode and frame rate from the camera.*



- virtual [Error SetVideoModeAndFrameRate](#) ([VideoMode](#) videoMode, [FrameRate](#) frameRate)

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

## Format7

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

- virtual [Error GetFormat7Info](#) ([Format7Info](#) \*pInfo, bool \*pSupported)

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

- virtual [Error ValidateFormat7Settings](#) (const [Format7ImageSettings](#) \*pImageSettings, bool \*pSettingsAreValid, [Format7PacketInfo](#) \*pPacketInfo)

*Validates [Format7ImageSettings](#) structure and returns valid packet size information if the image settings are valid.*

- virtual [Error GetFormat7Configuration](#) ([Format7ImageSettings](#) \*pImageSettings, unsigned int \*pPacketSize, float \*pPercentage)

*Get the current Format7 configuration from the camera.*

- virtual [Error SetFormat7Configuration](#) (const [Format7ImageSettings](#) \*pImageSettings, unsigned int packetSize)

*Set the current Format7 configuration to the camera.*

- virtual [Error SetFormat7Configuration](#) (const [Format7ImageSettings](#) \*pImageSettings, float percentSpeed)

*Set the current Format7 configuration to the camera.*

### 8.4.1 Detailed Description

The [Camera](#) object represents a physical camera that uses the IIDC register set.

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

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

### 8.4.2 Constructor & Destructor Documentation

#### 8.4.2.1 Camera ()

Default constructor.

#### 8.4.2.2 virtual ~Camera () [virtual]

Default destructor.

### 8.4.3 Member Function Documentation

#### 8.4.3.1 virtual Error Connect (PGRGuid \* *pGuid* = NULL) [virtual]

The following functions are inherited from [CameraBase](#).

See [CameraBase.h](#) for further information.

Implements [CameraBase](#).

#### 8.4.3.2 virtual Error Disconnect () [virtual]

Disconnects the camera object from the camera.

This allows another physical camera to be connected to the camera object.

See also:

[Connect\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

#### 8.4.3.3 virtual Error EnableLUT (bool *on*) [virtual]

Enable or disable LUT functionality on the camera.

**Parameters:**

*on* Whether to enable or disable LUT.

See also:

[GetLUTInfo\(\)](#)

[GetLUTChannel\(\)](#)

[SetLUTChannel\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

#### 8.4.3.4 virtual Error FireSoftwareTrigger (bool *broadcast* = false) [virtual]

Fire the software trigger according to the DCAM specifications.

**Parameters:**

*broadcast* Whether the action should be broadcast.

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.4.3.5 virtual Error GetActiveLUTBank (unsigned int \* *pActiveBank*) [virtual]**

Get the LUT bank that is currently being used.

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

**Parameters:**

*pActiveBank* The currently active bank.

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.4.3.6 virtual Error GetCameraInfo (CameraInfo \* *pCameraInfo*) [virtual]**

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

**Parameters:**

*pCameraInfo* Pointer to the camera information structure to be filled.

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.4.3.7 virtual Error GetConfiguration (FC2Config \* *pConfig*) [virtual]**

Get the configuration associated with the camera object.

**Parameters:**

*pConfig* Pointer to the configuration structure to be filled.

**See also:**

[SetConfiguration\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.4.3.8 virtual Error GetEmbeddedImageInfo (EmbeddedImageInfo \* *pInfo*) [virtual]**

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

**Parameters:**

*pInfo* Structure to be filled.

See also:

[SetEmbeddedImageInfo\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

#### 8.4.3.9 virtual Error GetFormat7Configuration (Format7ImageSettings \* *pImageSettings*, unsigned int \* *pPacketSize*, float \* *pPercentage*) [virtual]

Get the current Format7 configuration from the camera.

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

Parameters:

*pImageSettings* Current image settings.

*pPacketSize* Current packet size.

*pPercentage* Current packet size as a percentage.

See also:

[GetFormat7Info\(\)](#)

[ValidateFormat7Settings\(\)](#)

[SetFormat7Configuration\(\)](#)

[GetVideoModeAndFrameRate\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

#### 8.4.3.10 virtual Error GetFormat7Info (Format7Info \* *pInfo*, bool \* *pSupported*) [virtual]

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

The mode must be specified in the [Format7Info](#) structure in order for the function to succeed.

Parameters:

*pInfo* Structure to be filled with the capabilities of the specified mode and the current state in the specified mode.

*pSupported* Whether the specified mode is supported.

See also:

[ValidateFormat7Settings\(\)](#)

[GetFormat7Configuration\(\)](#)

[SetFormat7Configuration\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

#### 8.4.3.11 virtual Error GetGPIOPinDirection (unsigned int *pin*, unsigned int \* *pDirection*) [virtual]

Get the GPIO pin direction for the specified pin.

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

##### Parameters:

*pin* Pin to get the direction for.

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

##### See also:

[SetGPIOPinDirection\(\)](#)

##### Returns:

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

#### 8.4.3.12 virtual Error GetLUTBankInfo (unsigned int *bank*, bool \* *pReadSupported*, bool \* *pWriteSupported*) [virtual]

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

##### Parameters:

*bank* The bank to query.

*pReadSupported* Whether reading from the bank is supported.

*pWriteSupported* Whether writing to the bank is supported.

##### Returns:

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

#### 8.4.3.13 virtual Error GetLUTChannel (unsigned int *bank*, unsigned int *channel*, unsigned int *sizeEntries*, unsigned int \* *pEntries*) [virtual]

Get the LUT channel settings from the camera.

##### Parameters:

*bank* Bank to retrieve.

*channel* Channel to retrieve.

*sizeEntries* Number of entries in LUT table to read.

*pEntries* Array to store LUT entries.

**See also:**

[GetLUTInfo\(\)](#)  
[EnableLUT\(\)](#)  
[SetLUTChannel\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.4.3.14 virtual Error GetLUTInfo (LUTData \* *pData*) [virtual]**

Query if LUT support is available on the camera.

**Parameters:**

*pData* The LUT structure to be filled.

**See also:**

[EnableLUT\(\)](#)  
[GetLUTChannel\(\)](#)  
[SetLUTChannel\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.4.3.15 virtual Error GetMemoryChannel (unsigned int \* *pCurrentChannel*) [virtual]**

Retrieve the current memory channel from the camera.

**Parameters:**

*pCurrentChannel* Current memory channel.

**See also:**

[SaveToMemoryChannel\(\)](#)  
[RestoreFromMemoryChannel\(\)](#)  
[GetMemoryChannelInfo\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.4.3.16 virtual Error GetMemoryChannelInfo (unsigned int \* *pNumChannels*)** [virtual]

Query the camera for memory channel support.

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

**Parameters:**

*pNumChannels* Number of memory channels supported.

**See also:**

[GetMemoryChannel\(\)](#)  
[SaveToMemoryChannel\(\)](#)  
[RestoreFromMemoryChannel\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.4.3.17 virtual Error GetProperty (Property \* *pProp*)** [virtual]

Reads the settings for the specified property from the camera.

The property type must be specified in the [Property](#) structure passed into the function in order for the function to succeed. If auto is on, the integer and abs values returned may not be consistent with each other.

**Parameters:**

*pProp* Pointer to the [Property](#) structure to be filled.

**See also:**

[GetPropertyInfo\(\)](#)  
[SetProperty\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.4.3.18 virtual Error GetPropertyInfo (PropertyInfo \* *pPropInfo*)** [virtual]

Retrieves information about the specified camera property.

The property type must be specified in the [PropertyInfo](#) structure passed into the function in order for the function to succeed.

**Parameters:**

*pPropInfo* Pointer to the [PropertyInfo](#) structure to be filled.

See also:

[GetProperty\(\)](#)  
[SetProperty\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

#### 8.4.3.19 static const char\* GetRegisterString (unsigned int *registerVal*) [static]

Returns a text representation of the register value.

Parameters:

*registerVal* The register value to query.

Returns:

The text representation of the register.

Reimplemented from [CameraBase](#).

#### 8.4.3.20 virtual Error GetStrobe (StrobeControl \* *pStrobeControl*) [virtual]

Retrieve current strobe settings from the camera.

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

Parameters:

*pStrobeControl* Structure to receive strobe settings.

See also:

[GetStrobeInfo\(\)](#)  
[SetStrobe\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

#### 8.4.3.21 virtual Error GetStrobeInfo (StrobeInfo \* *pStrobeInfo*) [virtual]

Retrieve strobe information from the camera.

Parameters:

*pStrobeInfo* Structure to receive strobe information.



**See also:**

[GetStrobe\(\)](#)  
[SetStrobe\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.4.3.22 virtual Error GetTriggerDelay (TriggerDelay \* *pTriggerDelay*)** [virtual]

Retrieve current trigger delay settings from the camera.

**Parameters:**

*pTriggerDelay* Structure to receive trigger delay settings.

**See also:**

[GetTriggerModeInfo\(\)](#)  
[GetTriggerMode\(\)](#)  
[SetTriggerMode\(\)](#)  
[GetTriggerDelayInfo\(\)](#)  
[SetTriggerDelay\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.4.3.23 virtual Error GetTriggerDelayInfo (TriggerDelayInfo \* *pTriggerDelayInfo*)** [virtual]

Retrieve trigger delay information from the camera.

**Parameters:**

*pTriggerDelayInfo* Structure to receive trigger delay information.

**See also:**

[GetTriggerModeInfo\(\)](#)  
[GetTriggerMode\(\)](#)  
[SetTriggerMode\(\)](#)  
[GetTriggerDelay\(\)](#)  
[SetTriggerDelay\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

#### 8.4.3.24 virtual Error GetTriggerMode (TriggerMode \* *pTriggerMode*) [virtual]

Retrieve current trigger settings from the camera.

##### Parameters:

*pTriggerMode* Structure to receive trigger mode settings.

##### See also:

[GetTriggerModeInfo\(\)](#)  
[SetTriggerMode\(\)](#)  
[GetTriggerDelayInfo\(\)](#)  
[GetTriggerDelay\(\)](#)  
[SetTriggerDelay\(\)](#)

##### Returns:

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

#### 8.4.3.25 virtual Error GetTriggerModeInfo (TriggerModeInfo \* *pTriggerModeInfo*) [virtual]

Retrieve trigger information from the camera.

##### Parameters:

*pTriggerModeInfo* Structure to receive trigger information.

##### See also:

[GetTriggerMode\(\)](#)  
[SetTriggerMode\(\)](#)  
[GetTriggerDelayInfo\(\)](#)  
[GetTriggerDelay\(\)](#)  
[SetTriggerDelay\(\)](#)

##### Returns:

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

#### 8.4.3.26 virtual Error GetVideoModeAndFrameRate (VideoMode \* *pVideoMode*, FrameRate \* *pFrameRate*) [virtual]

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

If the camera is in Format7, the video mode will be VIDEOMODE\_FORMAT7 and the frame rate will be FRAMERATE\_FORMAT7.

##### Parameters:

*pVideoMode* Current video mode.

*pFrameRate* Current frame rate.

See also:

[GetVideoModeAndFrameRateInfo\(\)](#)  
[SetVideoModeAndFrameRate\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

#### 8.4.3.27 virtual Error GetVideoModeAndFrameRateInfo (VideoMode *videoMode*, FrameRate *frameRate*, bool \**pSupported*) [virtual]

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

Parameters:

*videoMode* Video mode to check.  
*frameRate* Frame rate to check.  
*pSupported* Whether the video mode and frame rate is supported.

See also:

[GetVideoModeAndFrameRate\(\)](#)  
[SetVideoModeAndFrameRate\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

#### 8.4.3.28 virtual bool IsConnected () [virtual]

Checks if the camera object is currently connected to a physical camera.

See also:

[Connect\(\)](#)  
[Disconnect\(\)](#)

Returns:

Whether the camera object is connected to a physical camera.

Implements [CameraBase](#).

#### 8.4.3.29 virtual Error ReadRegister (unsigned int *address*, unsigned int \**pValue*) [virtual]

Read the specified register from the camera.

Parameters:

*address* DCAM address to be read from.

*pValue* The value that is read.

See also:

[WriteRegister\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

#### 8.4.3.30 virtual Error ReadRegisterBlock (unsigned short *addressHigh*, unsigned int *addressLow*, unsigned int \* *pBuffer*, unsigned int *length*) [virtual]

Read from the specified register block on the camera.

Parameters:

*addressHigh* Top 16 bits of the 48 bit absolute address to read from.

*addressLow* Bottom 32 bits of the 48 bits absolute address to read from.

*pBuffer* Array to store read data.

*length* Size of array, in quadlets.

See also:

[WriteRegisterBlock\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

#### 8.4.3.31 virtual Error RestoreFromMemoryChannel (unsigned int *channel*) [virtual]

Restore the specified current memory channel.

Parameters:

*channel* Memory channel to restore from.

See also:

[GetMemoryChannel\(\)](#)

[SaveToMemoryChannel\(\)](#)

[GetMemoryChannelInfo\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.4.3.32 virtual Error RetrieveBuffer (Image \* *pImage*)** [virtual]

Retrieves the the next image object containing the next image.

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

**Parameters:**

*pImage* Pointer to [Image](#) object to store image data.

**See also:**

[StartCapture\(\)](#)  
[StopCapture\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.4.3.33 virtual Error SaveToMemoryChannel (unsigned int *channel*)** [virtual]

Save the current settings to the specfied current memory channel.

**Parameters:**

*channel* Memory channel to save to.

**See also:**

[GetMemoryChannel\(\)](#)  
[RestoreFromMemoryChannel\(\)](#)  
[GetMemoryChannelInfo\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.4.3.34 virtual Error SetActiveLUTBank (unsigned int *activeBank*)** [virtual]

Set the LUT bank that will be used.

**Parameters:**

*activeBank* The bank to be set as active.

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

#### 8.4.3.35 **virtual Error SetCallback (ImageEventCallback *callbackFn*, const void \* *pCallbackData* = NULL) [virtual]**

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

To clear the current stored callback data, pass in NULL for both arguments.

##### Parameters:

*callbackFn* A function to be called when a new image is received.

*pCallbackData* A pointer to data that can be passed to the callback function.

##### See also:

[StartCapture\(\)](#)

##### Returns:

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

#### 8.4.3.36 **virtual Error SetConfiguration (const FC2Config \* *pConfig*) [virtual]**

Set the configuration associated with the camera object.

##### Parameters:

*pConfig* Pointer to the configuration structure to be used.

##### See also:

[GetConfiguration\(\)](#)

##### Returns:

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

#### 8.4.3.37 **virtual Error SetEmbeddedImageInfo (EmbeddedImageInfo \* *pInfo*) [virtual]**

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

##### Parameters:

*pInfo* Structure to be used.

##### See also:

[GetEmbeddedImageInfo\(\)](#)

##### Returns:

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

#### 8.4.3.38 virtual Error SetFormat7Configuration (const Format7ImageSettings \* *pImageSettings*, float *percentSpeed*) [virtual]

Set the current Format7 configuration to the camera.

##### Parameters:

*pImageSettings* [Image](#) settings to be written to the camera.

*percentSpeed* Percentage of packet size to be written to the camera.

##### See also:

[GetFormat7Info\(\)](#)

[ValidateFormat7Settings\(\)](#)

[GetFormat7Configuration\(\)](#)

##### Returns:

An [Error](#) indicating the success or failure of the function.

#### 8.4.3.39 virtual Error SetFormat7Configuration (const Format7ImageSettings \* *pImageSettings*, unsigned int *packetSize*) [virtual]

Set the current Format7 configuration to the camera.

##### Parameters:

*pImageSettings* [Image](#) settings to be written to the camera.

*packetSize* Packet size to be written to the camera.

##### See also:

[GetFormat7Info\(\)](#)

[ValidateFormat7Settings\(\)](#)

[GetFormat7Configuration\(\)](#)

##### Returns:

An [Error](#) indicating the success or failure of the function.

#### 8.4.3.40 virtual Error SetGPIOPinDirection (unsigned int *pin*, unsigned int *direction*, bool *broadcast* = false) [virtual]

Set the GPIO pin direction for the specified pin.

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

##### Parameters:

*pin* Pin to get the direction for.

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

*broadcast* Whether the action should be broadcast.

See also:

[GetGPIOPinDirection\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

#### 8.4.3.41 virtual Error SetLUTChannel (unsigned int *bank*, unsigned int *channel*, unsigned int *sizeEntries*, const unsigned int \**pEntries*) [virtual]

Set the LUT channel settings to the camera.

Parameters:

*bank* Bank to set.

*channel* Channel to set.

*sizeEntries* Number of entries in LUT table to write.

*pEntries* Array containing LUT entries to write.

See also:

[GetLUTInfo\(\)](#)

[EnableLUT\(\)](#)

[GetLUTChannel\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

#### 8.4.3.42 virtual Error SetProperty (const Property \**pProp*, bool *broadcast* = false) [virtual]

Writes the settings for the specified property to the camera.

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

Parameters:

*pProp* Pointer to the [Property](#) structure to be used.

*broadcast* Whether the action should be broadcast.

See also:

[GetPropertyInfo\(\)](#)

[GetProperty\(\)](#)



**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.4.3.43 virtual Error SetStrobe (const StrobeControl \* *pStrobeControl*, bool *broadcast* = false)**  
[virtual]

Set current strobe settings to the camera.

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

**Parameters:**

*pStrobeControl* Structure providing strobe settings.

*broadcast* Whether the action should be broadcast.

**See also:**

[GetStrobeInfo\(\)](#)

[GetStrobe\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.4.3.44 virtual Error SetTriggerDelay (const TriggerDelay \* *pTriggerDelay*, bool *broadcast* = false)**  
[virtual]

Set the specified trigger delay settings to the camera.

**Parameters:**

*pTriggerDelay* Structure providing trigger delay settings.

*broadcast* Whether the action should be broadcast.

**See also:**

[GetTriggerModeInfo\(\)](#)

[GetTriggerMode\(\)](#)

[SetTriggerMode\(\)](#)

[GetTriggerDelayInfo\(\)](#)

[GetTriggerDelay\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

#### 8.4.3.45 **virtual Error SetTriggerMode (const TriggerMode \* *pTriggerMode*, bool *broadcast* = false) [virtual]**

Set the specified trigger settings to the camera.

##### Parameters:

*pTriggerMode* Structure providing trigger mode settings.  
*broadcast* Whether the action should be broadcast.

##### See also:

[GetTriggerModeInfo\(\)](#)  
[GetTriggerMode\(\)](#)  
[GetTriggerDelayInfo\(\)](#)  
[GetTriggerDelay\(\)](#)  
[SetTriggerDelay\(\)](#)

##### Returns:

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

#### 8.4.3.46 **virtual Error SetUserBuffers (unsigned char \*const *pMemBuffers*, int *size*, int *numBuffers*) [virtual]**

Specify user allocated buffers to use as image data buffers.

To prevent image tearing, the size of each buffer should be equal to ((unsigned int)(bufferSize + packetSize - 1)/packetSize) \* packetSize. The total size should be (size \* numBuffers) or larger.

##### Parameters:

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

##### See also:

[StartCapture\(\)](#)  
[RetrieveBuffer\(\)](#)  
[StopCapture\(\)](#)

##### Returns:

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

#### 8.4.3.47 **virtual Error SetVideoModeAndFrameRate (VideoMode *videoMode*, FrameRate *frameRate*) [virtual]**

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

It is not possible to set the camera to VIDEOMODE\_FORMAT7 or FRAMERATE\_FORMAT7. Use the Format7 functions to set the camera into Format7.

**Parameters:**

*videoMode* Video mode to set to camera.

*frameRate* Frame rate to set to camera.

**See also:**

[GetVideoModeAndFrameRateInfo\(\)](#)

[GetVideoModeAndFrameRate\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.4.3.48** `virtual Error StartCapture (ImageEventCallback callbackFn = NULL, const void * pCallbackData = NULL) [virtual]`

Starts isochronous image capture.

It will use either the current video mode or the most recently set video mode of the camera. The optional callback function parameter is called on completion of image transfer. Alternatively, the callback parameter can be set to NULL and [RetrieveBuffer\(\)](#) can be called as a blocking call to get the image data.

**Parameters:**

*callbackFn* A function to be called when a new image is received.

*pCallbackData* A pointer to data that can be passed to the callback function.

**See also:**

[RetrieveBuffer\(\)](#)

[StartSyncCapture\(\)](#)

[StopCapture\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.4.3.49** `static Error StartSyncCapture (unsigned int numCameras, const Camera ** ppCameras, const ImageEventCallback * pCallbackFns = NULL, const void ** pCallbackDataArray = NULL) [static]`

**8.4.3.50** `virtual Error StopCapture () [virtual]`

Stops isochronous image transfer and cleans up all associated resources.

**See also:**

[StartCapture\(\)](#)

[RetrieveBuffer\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

#### 8.4.3.51 virtual Error ValidateFormat7Settings (const Format7ImageSettings \* *pImageSettings*, bool \* *pSettingsAreValid*, Format7PacketInfo \* *pPacketInfo*) [virtual]

Validates [Format7ImageSettings](#) structure and returns valid packet size information if the image settings are valid.

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

##### Parameters:

*pImageSettings* Structure containing the image settings.

*pSettingsAreValid* Whether the settings are valid.

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

##### See also:

[GetFormat7Info\(\)](#)

[GetFormat7Configuration\(\)](#)

[SetFormat7Configuration\(\)](#)

##### Returns:

An [Error](#) indicating the success or failure of the function.

#### 8.4.3.52 virtual Error WaitForBufferEvent (Image \* *pImage*, unsigned int *eventNumber*) [virtual]

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

##### Parameters:

*pImage* Pointer to [Image](#) object to store image data.

*eventNumber* The event number to wait for.

##### See also:

[StartCapture\(\)](#)

[RetrieveBuffer\(\)](#)

[StopCapture\(\)](#)

##### Returns:

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

#### 8.4.3.53 virtual Error WriteRegister (unsigned int *address*, unsigned int *value*, bool *broadcast* = false) [virtual]

Write to the specified register on the camera.

##### Parameters:

*address* DCAM address to be written to.

*value* The value to be written.

*broadcast* Whether the action should be broadcast.

See also:

[ReadRegister\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.4.3.54 virtual Error WriteRegisterBlock (unsigned short *addressHigh*, unsigned int *addressLow*, const unsigned int \* *pBuffer*, unsigned int *length*)** [virtual]

Write to the specified register block on the camera.

Parameters:

*addressHigh* Top 16 bits of the 48 bit absolute address to write to.

*addressLow* Bottom 32 bits of the 48 bits absolute address to write to.

*pBuffer* Array containing data to be written.

*length* Size of array, in quadlets.

See also:

[ReadRegisterBlock\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

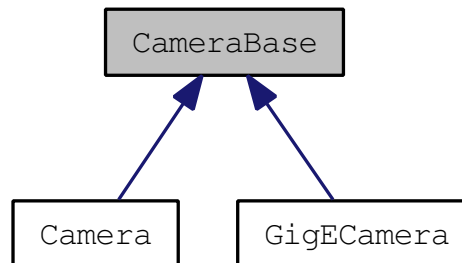
The documentation for this class was generated from the following file:

- [Camera.h](#)

## 8.5 CameraBase Class Reference

The [CameraBase](#) class is an abstract base class that defines a general interface to a camera.

Inheritance diagram for CameraBase:



### Public Member Functions

- [CameraBase](#) ()  
*Default constructor.*
- virtual [~CameraBase](#) ()  
*Default destructor.*

### Protected Attributes

- CameraData \* [m\\_pCameraData](#)

### Connection and Image Retrieval

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

- virtual [Error Connect](#) ([PGRGuid](#) \*pGuid=NULL)=0  
*Connects the camera object to the camera specified by the GUID.*
- virtual [Error Disconnect](#) ()=0  
*Disconnects the camera object from the camera.*
- virtual bool [IsConnected](#) ()=0  
*Checks if the camera object is currently connected to a physical camera.*
- virtual [Error SetCallback](#) ([ImageEventCallback](#) callbackFn, const void \*pCallbackData=NULL)=0  
*Sets the callback data to be used on completion of image transfer.*
- virtual [Error StartCapture](#) ([ImageEventCallback](#) callbackFn=NULL, const void \*pCallbackData=NULL)=0  
*Starts isochronous image capture.*

- virtual [Error RetrieveBuffer](#) ([Image](#) \*pImage)=0  
*Retrieves the the next image object containing the next image.*
- virtual [Error StopCapture](#) ()=0  
*Stops isochronous image transfer and cleans up all associated resources.*
- virtual [Error WaitForBufferEvent](#) ([Image](#) \*pImage, unsigned int eventNumber)=0  
*Retrieves the next image event containing the next part of the image.*
- virtual [Error SetUserBuffers](#) (unsigned char \*const pMemBuffers, int size, int numBuffers)=0  
*Specify user allocated buffers to use as image data buffers.*
- virtual [Error GetConfiguration](#) ([FC2Config](#) \*pConfig)=0  
*Get the configuration associated with the camera object.*
- virtual [Error SetConfiguration](#) (const [FC2Config](#) \*pConfig)=0  
*Set the configuration associated with the camera object.*
- static [Error StartSyncCapture](#) (unsigned int numCameras, const [CameraBase](#) \*\*ppCameras, const [ImageEventCallback](#) \*pCallbackFns=NULL, const void \*\*pCallbackDataArray=NULL)  
*Starts isochronous image capture on multiple cameras.*

## Information and Properties

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

- virtual [Error GetCameraInfo](#) ([CameraInfo](#) \*pCameraInfo)=0  
*Retrieves information from the camera such as serial number, model name and other camera information.*
- virtual [Error GetPropertyInfo](#) ([PropertyInfo](#) \*pPropInfo)=0  
*Retrieves information about the specified camera property.*
- virtual [Error GetProperty](#) ([Property](#) \*pProp)=0  
*Reads the settings for the specified property from the camera.*
- virtual [Error SetProperty](#) (const [Property](#) \*pProp, bool broadcast=false)=0  
*Writes the settings for the specified property to the camera.*

## General Purpose Input / Output

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

- virtual [Error GetGPIOPinDirection](#) (unsigned int pin, unsigned int \*pDirection)=0  
*Get the GPIO pin direction for the specified pin.*
- virtual [Error SetGPIOPinDirection](#) (unsigned int pin, unsigned int direction, bool broadcast=false)=0  
*Set the GPIO pin direction for the specified pin.*

## Trigger

These functions deal with trigger control on the camera.

- virtual [Error GetTriggerModeInfo](#) ([TriggerModeInfo](#) \*pTriggerModeInfo)=0  
*Retrieve trigger information from the camera.*
- virtual [Error GetTriggerMode](#) ([TriggerMode](#) \*pTriggerMode)=0  
*Retrieve current trigger settings from the camera.*
- virtual [Error SetTriggerMode](#) (const [TriggerMode](#) \*pTriggerMode, bool broadcast=false)=0  
*Set the specified trigger settings to the camera.*
- virtual [Error FireSoftwareTrigger](#) (bool broadcast=false)=0  
*Fire the software trigger according to the DCAM specifications.*
- virtual [Error GetTriggerDelayInfo](#) ([TriggerDelayInfo](#) \*pTriggerDelayInfo)=0  
*Retrieve trigger delay information from the camera.*
- virtual [Error GetTriggerDelay](#) ([TriggerDelay](#) \*pTriggerDelay)=0  
*Retrieve current trigger delay settings from the camera.*
- virtual [Error SetTriggerDelay](#) (const [TriggerDelay](#) \*pTriggerDelay, bool broadcast=false)=0  
*Set the specified trigger delay settings to the camera.*

## Strobe

These functions deal with strobe control on the camera.

- virtual [Error GetStrobeInfo](#) ([StrobeInfo](#) \*pStrobeInfo)=0  
*Retrieve strobe information from the camera.*
- virtual [Error GetStrobe](#) ([StrobeControl](#) \*pStrobeControl)=0  
*Retrieve current strobe settings from the camera.*
- virtual [Error SetStrobe](#) (const [StrobeControl](#) \*pStrobeControl, bool broadcast=false)=0  
*Set current strobe settings to the camera.*

## Look Up Table

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

- virtual [Error GetLUTInfo](#) ([LUTData](#) \*pData)=0  
*Query if LUT support is available on the camera.*
- virtual [Error GetLUTBankInfo](#) (unsigned int bank, bool \*pReadSupported, bool \*pWriteSupported)=0



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

- virtual [Error GetActiveLUTBank](#) (unsigned int \*pActiveBank)=0  
*Get the LUT bank that is currently being used.*
- virtual [Error SetActiveLUTBank](#) (unsigned int activeBank)=0  
*Set the LUT bank that will be used.*
- virtual [Error EnableLUT](#) (bool on)=0  
*Enable or disable LUT functionality on the camera.*
- virtual [Error GetLUTChannel](#) (unsigned int bank, unsigned int channel, unsigned int sizeEntries, unsigned int \*pEntries)=0  
*Get the LUT channel settings from the camera.*
- virtual [Error SetLUTChannel](#) (unsigned int bank, unsigned int channel, unsigned int sizeEntries, const unsigned int \*pEntries)=0  
*Set the LUT channel settings to the camera.*

## Memory Channels

These functions deal with memory channel control on the camera.

- virtual [Error GetMemoryChannel](#) (unsigned int \*pCurrentChannel)=0  
*Retrieve the current memory channel from the camera.*
- virtual [Error SaveToMemoryChannel](#) (unsigned int channel)=0  
*Save the current settings to the specified current memory channel.*
- virtual [Error RestoreFromMemoryChannel](#) (unsigned int channel)=0  
*Restore the specified current memory channel.*
- virtual [Error GetMemoryChannelInfo](#) (unsigned int \*pNumChannels)=0  
*Query the camera for memory channel support.*

## Embedded Image Information

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

- virtual [Error GetEmbeddedImageInfo](#) (EmbeddedImageInfo \*pInfo)=0  
*Get the current status of the embedded image information register, as well as the availability of each embedded property.*
- virtual [Error SetEmbeddedImageInfo](#) (EmbeddedImageInfo \*pInfo)=0  
*Sets the on/off values of the embedded image information structure to the camera.*

## Register Operation

These functions deal with register operation on the camera.

- virtual [Error WriteRegister](#) (unsigned int address, unsigned int value, bool broadcast=false)=0  
*Write to the specified register on the camera.*
- virtual [Error ReadRegister](#) (unsigned int address, unsigned int \*pValue)=0  
*Read the specified register from the camera.*
- virtual [Error WriteRegisterBlock](#) (unsigned short addressHigh, unsigned int addressLow, const unsigned int \*pBuffer, unsigned int length)=0  
*Write to the specified register block on the camera.*
- virtual [Error ReadRegisterBlock](#) (unsigned short addressHigh, unsigned int addressLow, unsigned int \*pBuffer, unsigned int length)=0  
*Read from the specified register block on the camera.*
- static const char \* [GetRegisterString](#) (unsigned int registerVal)  
*Returns a text representation of the register value.*

### 8.5.1 Detailed Description

The [CameraBase](#) class is an abstract base class that defines a general interface to a camera.

### 8.5.2 Constructor & Destructor Documentation

#### 8.5.2.1 [CameraBase \(\)](#) [inline]

Default constructor.

#### 8.5.2.2 [virtual ~CameraBase \(\)](#) [inline, virtual]

Default destructor.

### 8.5.3 Member Function Documentation

#### 8.5.3.1 [virtual Error Connect \(PGRGuid \\*pGuid = NULL\)](#) [pure virtual]

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

If the guid is omitted or set to NULL, the connection will be made to the first camera detected on the PC (i.e. index = 0).

#### Parameters:

*pGuid* The unique identifier for a specific camera on the PC.

See also:

[BusManager::GetCameraFromIndex\(\)](#)  
[BusManager::GetCameraFromSerialNumber\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

#### 8.5.3.2 virtual Error Disconnect () [pure virtual]

Disconnects the camera object from the camera.

This allows another physical camera to be connected to the camera object.

See also:

[Connect\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

#### 8.5.3.3 virtual Error EnableLUT (bool on) [pure virtual]

Enable or disable LUT functionality on the camera.

Parameters:

*on* Whether to enable or disable LUT.

See also:

[GetLUTInfo\(\)](#)  
[GetLUTChannel\(\)](#)  
[SetLUTChannel\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

#### 8.5.3.4 virtual Error FireSoftwareTrigger (bool broadcast = false) [pure virtual]

Fire the software trigger according to the DCAM specifications.

Parameters:

*broadcast* Whether the action should be broadcast.

Returns:

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

**8.5.3.5 virtual Error GetActiveLUTBank (unsigned int \* *pActiveBank*)** [pure virtual]

Get the LUT bank that is currently being used.

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

**Parameters:**

*pActiveBank* The currently active bank.

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

**8.5.3.6 virtual Error GetCameraInfo (CameraInfo \* *pCameraInfo*)** [pure virtual]

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

**Parameters:**

*pCameraInfo* Pointer to the camera information structure to be filled.

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

**8.5.3.7 virtual Error GetConfiguration (FC2Config \* *pConfig*)** [pure virtual]

Get the configuration associated with the camera object.

**Parameters:**

*pConfig* Pointer to the configuration structure to be filled.

**See also:**

[SetConfiguration\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

**8.5.3.8 virtual Error GetEmbeddedImageInfo (EmbeddedImageInfo \* *pInfo*)** [pure virtual]

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

**Parameters:**

*pInfo* Structure to be filled.

**See also:**

[SetEmbeddedImageInfo\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

### 8.5.3.9 virtual Error GetGPIOPinDirection (unsigned int *pin*, unsigned int \* *pDirection*) [pure virtual]

Get the GPIO pin direction for the specified pin.

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

**Parameters:**

*pin* Pin to get the direction for.

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

**See also:**

[SetGPIOPinDirection\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

### 8.5.3.10 virtual Error GetLUTBankInfo (unsigned int *bank*, bool \* *pReadSupported*, bool \* *pWriteSupported*) [pure virtual]

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

**Parameters:**

*bank* The bank to query.

*pReadSupported* Whether reading from the bank is supported.

*pWriteSupported* Whether writing to the bank is supported.

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

### 8.5.3.11 virtual Error GetLUTChannel (unsigned int *bank*, unsigned int *channel*, unsigned int *sizeEntries*, unsigned int \**pEntries*) [pure virtual]

Get the LUT channel settings from the camera.

#### Parameters:

*bank* Bank to retrieve.

*channel* Channel to retrieve.

*sizeEntries* Number of entries in LUT table to read.

*pEntries* Array to store LUT entries.

#### See also:

[GetLUTInfo\(\)](#)

[EnableLUT\(\)](#)

[SetLUTChannel\(\)](#)

#### Returns:

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

### 8.5.3.12 virtual Error GetLUTInfo (LUTData \**pData*) [pure virtual]

Query if LUT support is available on the camera.

#### Parameters:

*pData* The LUT structure to be filled.

#### See also:

[EnableLUT\(\)](#)

[GetLUTChannel\(\)](#)

[SetLUTChannel\(\)](#)

#### Returns:

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

### 8.5.3.13 virtual Error GetMemoryChannel (unsigned int \**pCurrentChannel*) [pure virtual]

Retrieve the current memory channel from the camera.

#### Parameters:

*pCurrentChannel* Current memory channel.

**See also:**

[SaveToMemoryChannel\(\)](#)  
[RestoreFromMemoryChannel\(\)](#)  
[GetMemoryChannelInfo\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

**8.5.3.14 virtual Error GetMemoryChannelInfo (unsigned int \* *pNumChannels*) [pure virtual]**

Query the camera for memory channel support.

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

**Parameters:**

*pNumChannels* Number of memory channels supported.

**See also:**

[GetMemoryChannel\(\)](#)  
[SaveToMemoryChannel\(\)](#)  
[RestoreFromMemoryChannel\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

**8.5.3.15 virtual Error GetProperty (Property \* *pProp*) [pure virtual]**

Reads the settings for the specified property from the camera.

The property type must be specified in the [Property](#) structure passed into the function in order for the function to succeed. If auto is on, the integer and abs values returned may not be consistent with each other.

**Parameters:**

*pProp* Pointer to the [Property](#) structure to be filled.

**See also:**

[GetPropertyInfo\(\)](#)  
[SetProperty\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

**8.5.3.16 virtual Error GetPropertyInfo (PropertyInfo \* *pPropInfo*)** [pure virtual]

Retrieves information about the specified camera property.

The property type must be specified in the [PropertyInfo](#) structure passed into the function in order for the function to succeed.

**Parameters:**

*pPropInfo* Pointer to the [PropertyInfo](#) structure to be filled.

**See also:**

[GetProperty\(\)](#)  
[SetProperty\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

**8.5.3.17 static const char\* GetRegisterString (unsigned int *registerVal*)** [static]

Returns a text representation of the register value.

**Parameters:**

*registerVal* The register value to query.

**Returns:**

The text representation of the register.

Reimplemented in [Camera](#), and [GigECamera](#).

**8.5.3.18 virtual Error GetStrobe (StrobeControl \* *pStrobeControl*)** [pure virtual]

Retrieve current strobe settings from the camera.

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

**Parameters:**

*pStrobeControl* Structure to receive strobe settings.

**See also:**

[GetStrobeInfo\(\)](#)  
[SetStrobe\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).



**8.5.3.19 virtual Error GetStrobeInfo (StrobeInfo \* *pStrobeInfo*)** [pure virtual]

Retrieve strobe information from the camera.

**Parameters:**

*pStrobeInfo* Structure to receive strobe information.

**See also:**

[GetStrobe\(\)](#)  
[SetStrobe\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

**8.5.3.20 virtual Error GetTriggerDelay (TriggerDelay \* *pTriggerDelay*)** [pure virtual]

Retrieve current trigger delay settings from the camera.

**Parameters:**

*pTriggerDelay* Structure to receive trigger delay settings.

**See also:**

[GetTriggerModeInfo\(\)](#)  
[GetTriggerMode\(\)](#)  
[SetTriggerMode\(\)](#)  
[GetTriggerDelayInfo\(\)](#)  
[SetTriggerDelay\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

**8.5.3.21 virtual Error GetTriggerDelayInfo (TriggerDelayInfo \* *pTriggerDelayInfo*)** [pure virtual]

Retrieve trigger delay information from the camera.

**Parameters:**

*pTriggerDelayInfo* Structure to receive trigger delay information.

**See also:**

[GetTriggerModeInfo\(\)](#)  
[GetTriggerMode\(\)](#)  
[SetTriggerMode\(\)](#)  
[GetTriggerDelay\(\)](#)  
[SetTriggerDelay\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

**8.5.3.22 virtual Error GetTriggerMode (TriggerMode \* *pTriggerMode*)** [pure virtual]

Retrieve current trigger settings from the camera.

**Parameters:**

*pTriggerMode* Structure to receive trigger mode settings.

**See also:**

[GetTriggerModeInfo\(\)](#)  
[SetTriggerMode\(\)](#)  
[GetTriggerDelayInfo\(\)](#)  
[GetTriggerDelay\(\)](#)  
[SetTriggerDelay\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

**8.5.3.23 virtual Error GetTriggerModeInfo (TriggerModeInfo \* *pTriggerModeInfo*)** [pure virtual]

Retrieve trigger information from the camera.

**Parameters:**

*pTriggerModeInfo* Structure to receive trigger information.

**See also:**

[GetTriggerMode\(\)](#)  
[SetTriggerMode\(\)](#)  
[GetTriggerDelayInfo\(\)](#)  
[GetTriggerDelay\(\)](#)  
[SetTriggerDelay\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

**8.5.3.24 virtual bool IsConnected ()** [pure virtual]

Checks if the camera object is currently connected to a physical camera.

See also:

[Connect\(\)](#)  
[Disconnect\(\)](#)

Returns:

Whether the camera object is connected to a physical camera.

Implemented in [Camera](#), and [GigECamera](#).

**8.5.3.25 virtual Error ReadRegister (unsigned int *address*, unsigned int \* *pValue*)** [pure virtual]

Read the specified register from the camera.

Parameters:

*address* DCAM address to be read from.  
*pValue* The value that is read.

See also:

[WriteRegister\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

**8.5.3.26 virtual Error ReadRegisterBlock (unsigned short *addressHigh*, unsigned int *addressLow*, unsigned int \* *pBuffer*, unsigned int *length*)** [pure virtual]

Read from the specified register block on the camera.

Parameters:

*addressHigh* Top 16 bits of the 48 bit absolute address to read from.  
*addressLow* Bottom 32 bits of the 48 bits absolute address to read from.  
*pBuffer* Array to store read data.  
*length* Size of array, in quadlets.

See also:

[WriteRegisterBlock\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

**8.5.3.27 virtual Error RestoreFromMemoryChannel (unsigned int *channel*)** [pure virtual]

Restore the specified current memory channel.

**Parameters:**

*channel* Memory channel to restore from.

**See also:**

[GetMemoryChannel\(\)](#)  
[SaveToMemoryChannel\(\)](#)  
[GetMemoryChannelInfo\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

**8.5.3.28 virtual Error RetrieveBuffer (Image \**pImage*)** [pure virtual]

Retrieves the the next image object containing the next image.

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

**Parameters:**

*pImage* Pointer to [Image](#) object to store image data.

**See also:**

[StartCapture\(\)](#)  
[StopCapture\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

**8.5.3.29 virtual Error SaveToMemoryChannel (unsigned int *channel*)** [pure virtual]

Save the current settings to the specified current memory channel.

**Parameters:**

*channel* Memory channel to save to.

**See also:**

[GetMemoryChannel\(\)](#)  
[RestoreFromMemoryChannel\(\)](#)  
[GetMemoryChannelInfo\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

**8.5.3.30 virtual Error SetActiveLUTBank (unsigned int *activeBank*) [pure virtual]**

Set the LUT bank that will be used.

**Parameters:**

*activeBank* The bank to be set as active.

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

**8.5.3.31 virtual Error SetCallback (ImageEventCallback *callbackFn*, const void \* *pCallbackData* = NULL) [pure virtual]**

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

To clear the current stored callback data, pass in NULL for both arguments.

**Parameters:**

*callbackFn* A function to be called when a new image is received.

*pCallbackData* A pointer to data that can be passed to the callback function.

**See also:**

[StartCapture\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

**8.5.3.32 virtual Error SetConfiguration (const FC2Config \* *pConfig*) [pure virtual]**

Set the configuration associated with the camera object.

**Parameters:**

*pConfig* Pointer to the configuration structure to be used.

**See also:**

[GetConfiguration\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

### 8.5.3.33 virtual Error SetEmbeddedImageInfo (EmbeddedImageInfo \* *pInfo*) [pure virtual]

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

#### Parameters:

*pInfo* Structure to be used.

#### See also:

[GetEmbeddedImageInfo\(\)](#)

#### Returns:

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

### 8.5.3.34 virtual Error SetGPIOPinDirection (unsigned int *pin*, unsigned int *direction*, bool *broadcast* = false) [pure virtual]

Set the GPIO pin direction for the specified pin.

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

#### Parameters:

*pin* Pin to get the direction for.

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

*broadcast* Whether the action should be broadcast.

#### See also:

[GetGPIOPinDirection\(\)](#)

#### Returns:

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

### 8.5.3.35 virtual Error SetLUTChannel (unsigned int *bank*, unsigned int *channel*, unsigned int *sizeEntries*, const unsigned int \* *pEntries*) [pure virtual]

Set the LUT channel settings to the camera.

#### Parameters:

*bank* Bank to set.

*channel* Channel to set.

*sizeEntries* Number of entries in LUT table to write.

*pEntries* Array containing LUT entries to write.

See also:

[GetLUTInfo\(\)](#)  
[EnableLUT\(\)](#)  
[GetLUTChannel\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

#### 8.5.3.36 virtual Error SetProperty (const Property \*pProp, bool broadcast = false) [pure virtual]

Writes the settings for the specified property to the camera.

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

Parameters:

*pProp* Pointer to the [Property](#) structure to be used.  
*broadcast* Whether the action should be broadcast.

See also:

[GetPropertyInfo\(\)](#)  
[GetProperty\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

#### 8.5.3.37 virtual Error SetStrobe (const StrobeControl \*pStrobeControl, bool broadcast = false) [pure virtual]

Set current strobe settings to the camera.

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

Parameters:

*pStrobeControl* Structure providing strobe settings.  
*broadcast* Whether the action should be broadcast.

See also:

[GetStrobeInfo\(\)](#)  
[GetStrobe\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

**8.5.3.38** `virtual Error SetTriggerDelay (const TriggerDelay * pTriggerDelay, bool broadcast = false) [pure virtual]`

Set the specified trigger delay settings to the camera.

**Parameters:**

*pTriggerDelay* Structure providing trigger delay settings.

*broadcast* Whether the action should be broadcast.

**See also:**

[GetTriggerModeInfo\(\)](#)  
[GetTriggerMode\(\)](#)  
[SetTriggerMode\(\)](#)  
[GetTriggerDelayInfo\(\)](#)  
[GetTriggerDelay\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

**8.5.3.39** `virtual Error SetTriggerMode (const TriggerMode * pTriggerMode, bool broadcast = false) [pure virtual]`

Set the specified trigger settings to the camera.

**Parameters:**

*pTriggerMode* Structure providing trigger mode settings.

*broadcast* Whether the action should be broadcast.

**See also:**

[GetTriggerModeInfo\(\)](#)  
[GetTriggerMode\(\)](#)  
[GetTriggerDelayInfo\(\)](#)  
[GetTriggerDelay\(\)](#)  
[SetTriggerDelay\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).



#### 8.5.3.40 virtual Error SetUserBuffers (unsigned char \*const *pMemBuffers*, int *size*, int *numBuffers*) [pure virtual]

Specify user allocated buffers to use as image data buffers.

To prevent image tearing, the size of each buffer should be equal to ((unsigned int)(bufferSize + packetSize - 1)/packetSize) \* packetSize. The total size should be (size \* numBuffers) or larger.

##### Parameters:

*pMemBuffers* Pointer to memory buffers to be written to.

*size* The size of each buffer (in bytes).

*numBuffers* Number of buffers in the array.

##### See also:

[StartCapture\(\)](#)  
[RetrieveBuffer\(\)](#)  
[StopCapture\(\)](#)

##### Returns:

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

#### 8.5.3.41 virtual Error StartCapture (ImageEventCallback *callbackFn* = NULL, const void \* *pCallbackData* = NULL) [pure virtual]

Starts isochronous image capture.

It will use either the current video mode or the most recently set video mode of the camera. The optional callback function parameter is called on completion of image transfer. Alternatively, the callback parameter can be set to NULL and [RetrieveBuffer\(\)](#) can be called as a blocking call to get the image data.

##### Parameters:

*callbackFn* A function to be called when a new image is received.

*pCallbackData* A pointer to data that can be passed to the callback function.

##### See also:

[RetrieveBuffer\(\)](#)  
[StartSyncCapture\(\)](#)  
[StopCapture\(\)](#)

##### Returns:

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

**8.5.3.42 static Error StartSyncCapture (unsigned int *numCameras*, const CameraBase \*\*  
*ppCameras*, const ImageEventCallback \* *pCallbackFns* = NULL, const void \*\*  
*pCallbackdataArray* = NULL) [static]**

Starts isochronous image capture on multiple cameras.

On each frame, the time stamps across the cameras are aligned which means the frames are synchronized. Note that the cameras must be synchronized by external means in order for this function to work. This means that the cameras should either be on the same bus, hardware synchronized (e.g. through triggering) or Multisync is running.

**Parameters:**

*numCameras* Number of [Camera](#) objects in the *ppCameras* array.

*ppCameras* Array of pointers to [Camera](#) objects containing the cameras to be started and synchronized.

*pCallbackFns* Array of callback functions for each camera.

*pCallbackdataArray* Array of callback data pointers.

**See also:**

[RetrieveBuffer\(\)](#)  
[StartCapture\(\)](#)  
[StopCapture\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.5.3.43 virtual Error StopCapture () [pure virtual]**

Stops isochronous image transfer and cleans up all associated resources.

**See also:**

[StartCapture\(\)](#)  
[RetrieveBuffer\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

**8.5.3.44 virtual Error WaitForBufferEvent (Image \* *pImage*, unsigned int *eventNumber*) [pure virtual]**

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

**Parameters:**

*pImage* Pointer to [Image](#) object to store image data.

*eventNumber* The event number to wait for.

See also:

[StartCapture\(\)](#)  
[RetrieveBuffer\(\)](#)  
[StopCapture\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

**8.5.3.45** `virtual Error WriteRegister (unsigned int address, unsigned int value, bool broadcast = false) [pure virtual]`

Write to the specified register on the camera.

Parameters:

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

See also:

[ReadRegister\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

**8.5.3.46** `virtual Error WriteRegisterBlock (unsigned short addressHigh, unsigned int addressLow, const unsigned int * pBuffer, unsigned int length) [pure virtual]`

Write to the specified register block on the camera.

Parameters:

*addressHigh* Top 16 bits of the 48 bit absolute address to write to.  
*addressLow* Bottom 32 bits of the 48 bits absolute address to write to.  
*pBuffer* Array containing data to be written.  
*length* Size of array, in quadlets.

See also:

[ReadRegisterBlock\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

Implemented in [Camera](#), and [GigECamera](#).

## 8.5.4 Member Data Documentation

### 8.5.4.1 `CameraData* m_pCameraData` [protected]

The documentation for this class was generated from the following file:

- [CameraBase.h](#)

## 8.6 CameraControlDlg Class Reference

The [CameraControlDlg](#) object represents a GTKmm dialog that provides a graphical interface to a specified camera.

### Public Member Functions

- [CameraControlDlg](#) ()  
*Default constructor.*
- [~CameraControlDlg](#) ()  
*Default destructor.*
- void [Connect](#) ([CameraBase](#) \*pCamera)  
*Connect dialog to a camera.*
- void [Disconnect](#) ()  
*Disconnect a connected camera from the dialog.*
- void [Show](#) ()  
*Show the dialog.*
- void [Hide](#) ()  
*Hide the dialog.*
- bool [IsVisible](#) ()  
*Get the visibility of the dialog.*

### 8.6.1 Detailed Description

The [CameraControlDlg](#) object represents a GTKmm dialog that provides a graphical interface to a specified camera.

### 8.6.2 Constructor & Destructor Documentation

#### 8.6.2.1 CameraControlDlg ()

Default constructor.

#### 8.6.2.2 ~CameraControlDlg ()

Default destructor.

## 8.6.3 Member Function Documentation

### 8.6.3.1 void Connect (CameraBase \* *pCamera*)

Connect dialog to a camera.

**Parameters:**

*pCamera* [Camera](#) object to connect the dialog to.

### 8.6.3.2 void Disconnect ()

Disconnect a connected camera from the dialog.

### 8.6.3.3 void Hide ()

Hide the dialog.

### 8.6.3.4 bool IsVisible ()

Get the visibility of the dialog.

**Returns:**

Whether the dialog is visible.

### 8.6.3.5 void Show ()

Show the dialog.

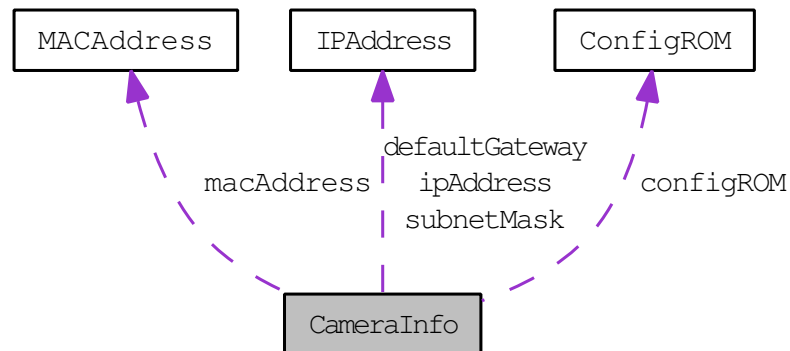
The documentation for this class was generated from the following file:

- [FlyCapture2GUI.h](#)

## 8.7 CameraInfo Struct Reference

Camera information.

Collaboration diagram for CameraInfo:



### Public Member Functions

- [CameraInfo](#) ()

### Public Attributes

- unsigned int [serialNumber](#)  
*Device serial number.*
- [InterfaceType](#) [interfaceType](#)  
*Interface type.*
- [DriverType](#) [driverType](#)  
*Driver type.*
- bool [isColorCamera](#)  
*Flag indicating if this is a color camera.*
- char [modelName](#) [[sk\\_maxStringLength](#)]  
*Device model name.*
- char [vendorName](#) [[sk\\_maxStringLength](#)]  
*Device vendor name.*
- char [sensorInfo](#) [[sk\\_maxStringLength](#)]  
*String detailing the sensor information.*
- char [sensorResolution](#) [[sk\\_maxStringLength](#)]  
*String providing the sensor resolution.*
- char [driverName](#) [[sk\\_maxStringLength](#)]

*Driver name of driver being used.*

- char [firmwareVersion](#) [sk\_maxStringLength]  
*Firmware version of camera.*
- char [firmwareBuildTime](#) [sk\_maxStringLength]  
*Firmware build time.*
- [BusSpeed](#) [maximumBusSpeed](#)  
*Maximum bus speed.*
- [BayerTileFormat](#) [bayerTileFormat](#)  
*Bayer tile format.*
- unsigned int [reserved](#) [16]  
*Reserved for future use.*

#### IIDC specific information

- unsigned int [iideVer](#)  
*DCAM version.*
- [ConfigROM](#) [configROM](#)  
*Configuration ROM data.*

#### GigE specific information

- unsigned int [gigEMajorVersion](#)  
*GigE Vision version.*
- unsigned int [gigEMinorVersion](#)  
*GigE Vision minor version.*
- char [userDefinedName](#) [sk\_maxStringLength]  
*User defined name.*
- char [xmlURL1](#) [sk\_maxStringLength]  
*XML URL 1.*
- char [xmlURL2](#) [sk\_maxStringLength]  
*XML URL 2.*
- [MACAddress](#) [macAddress](#)  
*MAC address.*
- [IPAddress](#) [ipAddress](#)  
*IP address.*
- [IPAddress](#) [subnetMask](#)  
*Subnet mask.*
- [IPAddress](#) [defaultGateway](#)  
*Default gateway.*



## 8.7.1 Detailed Description

Camera information.

## 8.7.2 Constructor & Destructor Documentation

### 8.7.2.1 CameraInfo () [inline]

## 8.7.3 Member Data Documentation

### 8.7.3.1 BayerTileFormat bayerTileFormat

Bayer tile format.

### 8.7.3.2 ConfigROM configROM

Configuration ROM data.

### 8.7.3.3 IPAddress defaultGateway

Default gateway.

### 8.7.3.4 char driverName[sk\_maxStringLength]

Driver name of driver being used.

### 8.7.3.5 DriverType driverType

Driver type.

### 8.7.3.6 char firmwareBuildTime[sk\_maxStringLength]

Firmware build time.

### 8.7.3.7 char firmwareVersion[sk\_maxStringLength]

Firmware version of camera.

### 8.7.3.8 unsigned int gigEMajorVersion

GigE Vision version.

### 8.7.3.9 unsigned int gigEMinorVersion

GigE Vision minor version.

**8.7.3.10 unsigned int iideVer**

DCAM version.

**8.7.3.11 InterfaceType interfaceType**

Interface type.

**8.7.3.12 IPAddress ipAddress**

IP address.

**8.7.3.13 bool isColorCamera**

Flag indicating if this is a color camera.

**8.7.3.14 MACAddress macAddress**

MAC address.

**8.7.3.15 BusSpeed maximumBusSpeed**

Maximum bus speed.

**8.7.3.16 char modelName[sk\_maxStringLength]**

Device model name.

**8.7.3.17 unsigned int reserved[16]**

Reserved for future use.

**8.7.3.18 char sensorInfo[sk\_maxStringLength]**

String detailing the sensor information.

**8.7.3.19 char sensorResolution[sk\_maxStringLength]**

String providing the sensor resolution.

**8.7.3.20 unsigned int serialNumber**

Device serial number.

**8.7.3.21 IPAddress subnetMask**

Subnet mask.

**8.7.3.22 char userDefinedName[sk\_maxStringLength]**

User defined name.

**8.7.3.23 char vendorName[sk\_maxStringLength]**

Device vendor name.

**8.7.3.24 char xmlURL1[sk\_maxStringLength]**

XML URL 1.

**8.7.3.25 char xmlURL2[sk\_maxStringLength]**

XML URL 2.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.8 CameraSelectionDlg Class Reference

The [CameraSelectionDlg](#) object represents a GTKmm dialog that provides a graphical interface that lists the number of cameras available to the library.

### Public Member Functions

- [CameraSelectionDlg](#) ()  
*Default constructor.*
- [~CameraSelectionDlg](#) ()  
*Default destructor.*
- void [ShowModal](#) (bool \*pOk, [PGRGuid](#) \*pGuid, unsigned int \*pSize)  
*Show the [CameraSelectionDlg](#).*

### 8.8.1 Detailed Description

The [CameraSelectionDlg](#) object represents a GTKmm dialog that provides a graphical interface that lists the number of cameras available to the library.

### 8.8.2 Constructor & Destructor Documentation

#### 8.8.2.1 [CameraSelectionDlg](#) ()

Default constructor.

#### 8.8.2.2 [~CameraSelectionDlg](#) ()

Default destructor.

### 8.8.3 Member Function Documentation

#### 8.8.3.1 void [ShowModal](#) (bool \*pOk, [PGRGuid](#) \*pGuid, unsigned int \*pSize)

Show the [CameraSelectionDlg](#).

#### Parameters:

- pOk* Whether Ok (true) or Cancel (false) was clicked.
- pGuid* Array of [PGRGuid](#)s containing the selected cameras.
- pSize* Size of [PGRGuid](#) array.

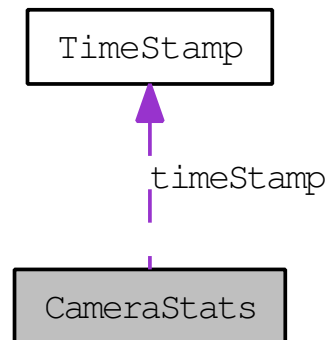
The documentation for this class was generated from the following file:

- [FlyCapture2GUI.h](#)

## 8.9 CameraStats Struct Reference

[Camera](#) diagnostic information.

Collaboration diagram for CameraStats:



### Public Member Functions

- [CameraStats](#) ()

### Public Attributes

- unsigned int [imageDropped](#)
- unsigned int [imageCorrupt](#)
- unsigned int [imageXmitFailed](#)
- unsigned int [imageDriverDropped](#)
- unsigned int [regReadFailed](#)
- unsigned int [regWriteFailed](#)
- unsigned int [portErrors](#)
- bool [cameraPowerUp](#)
- float [cameraVoltages](#) [8]
- unsigned int [numVoltages](#)

*The number of voltage registers available.*

- float [cameraCurrents](#) [8]
- unsigned int [numCurrents](#)

*The number of current registers available.*

- unsigned int [temperature](#)
- unsigned int [timeSinceInitialization](#)
- unsigned int [timeSinceBusReset](#)
- [TimeStamp](#) [timeStamp](#)
- unsigned int [reserved](#) [16]

*Reserved for future use.*

## 8.9.1 Detailed Description

[Camera](#) diagnostic information.

## 8.9.2 Constructor & Destructor Documentation

### 8.9.2.1 CameraStats () [inline]

## 8.9.3 Member Data Documentation

### 8.9.3.1 float cameraCurrents[8]

### 8.9.3.2 bool cameraPowerUp

### 8.9.3.3 float cameraVoltages[8]

### 8.9.3.4 unsigned int imageCorrupt

### 8.9.3.5 unsigned int imageDriverDropped

### 8.9.3.6 unsigned int imageDropped

### 8.9.3.7 unsigned int imageXmitFailed

### 8.9.3.8 unsigned int numCurrents

The number of current registers available.

0: the values in cameraCurrents[] are invalid.

### 8.9.3.9 unsigned int numVoltages

The number of voltage registers available.

0: the values in cameraVoltages[] are invalid.

### 8.9.3.10 unsigned int portErrors

### 8.9.3.11 unsigned int regReadFailed

### 8.9.3.12 unsigned int regWriteFailed

### 8.9.3.13 unsigned int reserved[16]

Reserved for future use.

**8.9.3.14**   unsigned int temperature

**8.9.3.15**   unsigned int timeSinceBusReset

**8.9.3.16**   unsigned int timeSinceInitialization

**8.9.3.17**   TimeStamp timeStamp

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.10 ConfigROM Struct Reference

Camera configuration ROM.

### Public Member Functions

- [ConfigROM](#) ()

### Public Attributes

- unsigned int [nodeVendorId](#)  
*Vendor ID of a node.*
- unsigned int [chipIdHi](#)  
*Chip ID (high part).*
- unsigned int [chipIdLo](#)  
*Chip ID (low part).*
- unsigned int [unitSpecId](#)  
*Unit Spec ID, usually 0xa02d.*
- unsigned int [unitSWVer](#)  
*Unit software version.*
- unsigned int [unitSubSWVer](#)  
*Unit sub software version.*
- unsigned int [vendorUniqueInfo\\_0](#)  
*Vendor unique info 0.*
- unsigned int [vendorUniqueInfo\\_1](#)  
*Vendor unique info 1.*
- unsigned int [vendorUniqueInfo\\_2](#)  
*Vendor unique info 2.*
- unsigned int [vendorUniqueInfo\\_3](#)  
*Vendor unique info 3.*
- char [pszKeyword](#) [[sk\\_maxStringLength](#)]  
*Keyword.*
- unsigned int [reserved](#) [16]  
*Reserved for future use.*



### 8.10.1 Detailed Description

Camera configuration ROM.

### 8.10.2 Constructor & Destructor Documentation

#### 8.10.2.1 ConfigROM () [inline]

### 8.10.3 Member Data Documentation

#### 8.10.3.1 unsigned int chipIdHi

Chip ID (high part).

#### 8.10.3.2 unsigned int chipIdLo

Chip ID (low part).

#### 8.10.3.3 unsigned int nodeVendorId

Vendor ID of a node.

#### 8.10.3.4 char pszKeyword[sk\_maxStringLength]

Keyword.

#### 8.10.3.5 unsigned int reserved[16]

Reserved for future use.

#### 8.10.3.6 unsigned int unitSpecId

Unit Spec ID, usually 0xa02d.

#### 8.10.3.7 unsigned int unitSubSWVer

Unit sub software version.

#### 8.10.3.8 unsigned int unitSWVer

Unit software version.

#### 8.10.3.9 unsigned int vendorUniqueInfo\_0

Vendor unique info 0.

**8.10.3.10 unsigned int vendorUniqueInfo\_1**

Vendor unique info 1.

**8.10.3.11 unsigned int vendorUniqueInfo\_2**

Vendor unique info 2.

**8.10.3.12 unsigned int vendorUniqueInfo\_3**

Vendor unique info 3.

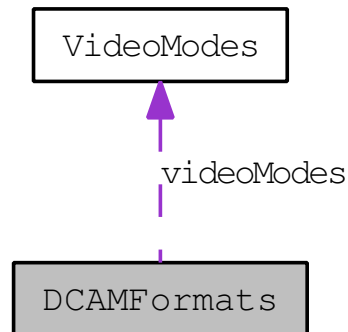
The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.11 DCAMFormats Struct Reference

DCAM formats supported by the camera as returned by `GetAvailableFormats()`.

Collaboration diagram for DCAMFormats:



### Public Attributes

- [VideoModes videoModes](#) [FlyCapture2::NUM\_VIDEOMODES]
- unsigned int [numFormats](#)

### 8.11.1 Detailed Description

DCAM formats supported by the camera as returned by `GetAvailableFormats()`.

### 8.11.2 Member Data Documentation

#### 8.11.2.1 unsigned int numFormats

#### 8.11.2.2 VideoModes videoModes[FlyCapture2::NUM\_VIDEOMODES]

The documentation for this struct was generated from the following file:

- [PGRDirectShow.h](#)

## 8.12 EmbeddedImageInfo Struct Reference

Properties of the possible embedded image information.

Collaboration diagram for EmbeddedImageInfo:



### Public Attributes

- [EmbeddedImageInfoProperty timestamp](#)
- [EmbeddedImageInfoProperty gain](#)
- [EmbeddedImageInfoProperty shutter](#)
- [EmbeddedImageInfoProperty brightness](#)
- [EmbeddedImageInfoProperty exposure](#)
- [EmbeddedImageInfoProperty whiteBalance](#)
- [EmbeddedImageInfoProperty frameCounter](#)
- [EmbeddedImageInfoProperty strobePattern](#)
- [EmbeddedImageInfoProperty GPIOPinState](#)
- [EmbeddedImageInfoProperty ROIPosition](#)

### 8.12.1 Detailed Description

Properties of the possible embedded image information.

## 8.12.2 Member Data Documentation

8.12.2.1 EmbeddedImageInfoProperty brightness

8.12.2.2 EmbeddedImageInfoProperty exposure

8.12.2.3 EmbeddedImageInfoProperty frameCounter

8.12.2.4 EmbeddedImageInfoProperty gain

8.12.2.5 EmbeddedImageInfoProperty GPIOPinState

8.12.2.6 EmbeddedImageInfoProperty ROIPosition

8.12.2.7 EmbeddedImageInfoProperty shutter

8.12.2.8 EmbeddedImageInfoProperty strobePattern

8.12.2.9 EmbeddedImageInfoProperty timestamp

8.12.2.10 EmbeddedImageInfoProperty whiteBalance

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.13 EmbeddedImageInfoProperty Struct Reference

Properties of a single embedded image info property.

### Public Member Functions

- [EmbeddedImageInfoProperty \(\)](#)

### Public Attributes

- bool [available](#)  
*Whether this property is available.*
- bool [onOff](#)  
*Whether this property is on or off.*

### 8.13.1 Detailed Description

Properties of a single embedded image info property.

### 8.13.2 Constructor & Destructor Documentation

#### 8.13.2.1 EmbeddedImageInfoProperty () `[inline]`

### 8.13.3 Member Data Documentation

#### 8.13.3.1 bool available

Whether this property is available.

#### 8.13.3.2 bool onOff

Whether this property is on or off.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.14 Error Class Reference

The [Error](#) object represents an error that is returned from the library.

### Public Member Functions

- [Error](#) ()  
*Default constructor.*
- [Error](#) (const [Error](#) &error)  
*Copy constructor.*
- virtual [~Error](#) ()  
*Default destructor.*
- virtual [Error](#) & [operator=](#) (const [Error](#) &error)  
*Assignment operator.*
- virtual bool [operator==](#) (const [Error](#) &error)  
*Equality operator.*
- virtual bool [operator==](#) (const [ErrorType](#) &errorType)  
*Equality operator.*
- virtual bool [operator!=](#) (const [Error](#) &error)  
*Inequality operator.*
- virtual bool [operator!=](#) (const [ErrorType](#) &errorType)  
*Inequality operator.*
- virtual [ErrorType](#) [GetType](#) () const  
*Retrieve the ErrorType of the error.*
- virtual const char \* [GetDescription](#) () const  
*Retrieve the top level description of the error that occurred.*
- virtual unsigned int [GetLine](#) () const  
*Retrieve the line number where the error originated.*
- virtual const char \* [GetFilename](#) () const  
*Retrieve the source filename where the error originated.*
- virtual [Error](#) [GetCause](#) () const  
*Get the error which caused this error.*
- virtual const char \* [GetBuildDate](#) () const  
*Retrieve the build date of the file where the error originated.*
- virtual const char \* [CollectSupportInformation](#) () const

*Retrieve the support information.*

- virtual void [PrintErrorTrace](#) () const  
*Print a formatted log trace to stderr.*

## Friends

- class [InternalError](#)

### 8.14.1 Detailed Description

The [Error](#) object represents an error that is returned from the library.

Overloaded operators allow comparisons against other [Error](#) objects or the `ErrorType` enumeration.

### 8.14.2 Constructor & Destructor Documentation

#### 8.14.2.1 `Error ()`

Default constructor.

#### 8.14.2.2 `Error (const Error & error)`

Copy constructor.

#### 8.14.2.3 `virtual ~Error ()` [virtual]

Default destructor.

### 8.14.3 Member Function Documentation

#### 8.14.3.1 `virtual const char* CollectSupportInformation () const` [virtual]

Retrieve the support information.

It is not implemented in this release.

##### Returns:

A string containing support information.

#### 8.14.3.2 `virtual const char* GetBuildDate () const` [virtual]

Retrieve the build date of the file where the error originated.

##### Returns:

A string with the build date and time.



**8.14.3.3 virtual Error GetCause () const** [virtual]

Get the error which caused this error.

**Returns:**

An error object representing the cause of this error.

**8.14.3.4 virtual const char\* GetDescription () const** [virtual]

Retrieve the top level description of the error that occurred.

**Returns:**

A string with the error description.

**8.14.3.5 virtual const char\* GetFilename () const** [virtual]

Retrieve the source filename where the error originated.

**Returns:**

A string with the file name.

**8.14.3.6 virtual unsigned int GetLine () const** [virtual]

Retrieve the line number where the error originated.

**Returns:**

The line number.

**8.14.3.7 virtual ErrorType GetType () const** [virtual]

Retrieve the ErrorType of the error.

**Returns:**

The ErrorType of the error.

**8.14.3.8 virtual bool operator!= (const ErrorType & *errorType*)** [virtual]

Inequality operator.

This overloaded operator compares the ErrorType of the [Error](#) against the specified ErrorType.

**8.14.3.9 virtual bool operator!= (const Error & *error*)** [virtual]

Inequality operator.

**8.14.3.10 virtual Error& operator= (const Error & *error*) [virtual]**

Assignment operator.

**8.14.3.11 virtual bool operator== (const ErrorType & *errorType*) [virtual]**

Equality operator.

This overloaded operator compares the ErrorType of the [Error](#) against the specified ErrorType.

**8.14.3.12 virtual bool operator== (const Error & *error*) [virtual]**

Equality operator.

**8.14.3.13 virtual void PrintErrorTrace () const [virtual]**

Print a formatted log trace to stderr.

**8.14.4 Friends And Related Function Documentation****8.14.4.1 friend class InternalError [friend]**

The documentation for this class was generated from the following file:

- [Error.h](#)

## 8.15 FC2Config Struct Reference

Configuration for a camera.

### Public Member Functions

- [FC2Config](#) ()

### Public Attributes

- unsigned int [numBuffers](#)  
*Number of buffers used by the [FlyCapture2](#) library to grab images.*
- unsigned int [numImageNotifications](#)  
*Number of notifications per image.*
- int [grabTimeout](#)  
*Time in milliseconds that [RetrieveBuffer\(\)](#) and [WaitForBufferEvent\(\)](#) will wait for an image before timing out and returning.*
- [GrabMode](#) [grabMode](#)  
*Grab mode for the camera.*
- [BusSpeed](#) [isochBusSpeed](#)  
*Isochronous bus speed.*
- [BusSpeed](#) [asyncBusSpeed](#)  
*Asynchronous bus speed.*
- [BandwidthAllocation](#) [bandwidthAllocation](#)  
*Bandwidth allocation flag that tells the camera the bandwidth allocation strategy to employ.*
- unsigned int [reserved](#) [16]  
*Reserved for future use.*

### 8.15.1 Detailed Description

Configuration for a camera.

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

## 8.15.2 Constructor & Destructor Documentation

### 8.15.2.1 FC2Config () [inline]

## 8.15.3 Member Data Documentation

### 8.15.3.1 BusSpeed asyncBusSpeed

Asynchronous bus speed.

### 8.15.3.2 BandwidthAllocation bandwidthAllocation

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

### 8.15.3.3 GrabMode grabMode

Grab mode for the camera.

The default is DROP\_FRAMES.

### 8.15.3.4 int grabTimeout

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

### 8.15.3.5 BusSpeed isochBusSpeed

Isochronous bus speed.

### 8.15.3.6 unsigned int numBuffers

Number of buffers used by the [FlyCapture2](#) library to grab images.

### 8.15.3.7 unsigned int numImageNotifications

Number of notifications per image.

The default number of notifications is 1.

There are 4 general scenarios:

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

#### 8.15.3.8 unsigned int reserved[16]

Reserved for future use.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.16 FC2Version Struct Reference

The current version of the library.

### Public Attributes

- unsigned int [major](#)  
*Major version number.*
- unsigned int [minor](#)  
*Minor version number.*
- unsigned int [type](#)  
*Type version number.*
- unsigned int [build](#)  
*Build version number.*

### 8.16.1 Detailed Description

The current version of the library.

### 8.16.2 Member Data Documentation

#### 8.16.2.1 unsigned int build

Build version number.

#### 8.16.2.2 unsigned int major

Major version number.

#### 8.16.2.3 unsigned int minor

Minor version number.

#### 8.16.2.4 unsigned int type

Type version number.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.17 Format7ImageSettings Struct Reference

Format 7 image settings.

### Public Member Functions

- [Format7ImageSettings \(\)](#)

### Public Attributes

- [Mode mode](#)  
*Format 7 mode.*
- unsigned int [offsetX](#)  
*Horizontal image offset.*
- unsigned int [offsetY](#)  
*Vertical image offset.*
- unsigned int [width](#)  
*Width of image.*
- unsigned int [height](#)  
*Height of image.*
- [PixelFormat pixelFormat](#)  
*Pixel format of image.*
- unsigned int [reserved](#) [8]  
*Reserved for future use.*

### 8.17.1 Detailed Description

Format 7 image settings.

### 8.17.2 Constructor & Destructor Documentation

#### 8.17.2.1 [Format7ImageSettings \(\)](#) `[inline]`

### 8.17.3 Member Data Documentation

#### 8.17.3.1 unsigned int height

Height of image.

**8.17.3.2 Mode mode**

Format 7 mode.

**8.17.3.3 unsigned int offsetX**

Horizontal image offset.

**8.17.3.4 unsigned int offsetY**

Vertical image offset.

**8.17.3.5 PixelFormat pixelFormat**

Pixel format of image.

**8.17.3.6 unsigned int reserved[8]**

Reserved for future use.

**8.17.3.7 unsigned int width**

Width of image.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)



## 8.18 Format7Info Struct Reference

Format 7 information for a single mode.

### Public Member Functions

- [Format7Info \(\)](#)

### Public Attributes

- [Mode mode](#)  
*Format 7 mode.*
- unsigned int [maxWidth](#)  
*Maximum image width.*
- unsigned int [maxHeight](#)  
*Maximum image height.*
- unsigned int [offsetHStepSize](#)  
*Horizontal step size for the offset.*
- unsigned int [offsetVStepSize](#)  
*Vertical step size for the offset.*
- unsigned int [imageHStepSize](#)  
*Horizontal step size for the image.*
- unsigned int [imageVStepSize](#)  
*Vertical step size for the image.*
- unsigned int [pixelFormatBitField](#)  
*Supported pixel formats in a bit field.*
- unsigned int [packetSize](#)  
*Current packet size in bytes.*
- unsigned int [minPacketSize](#)  
*Minimum packet size in bytes for current mode.*
- unsigned int [maxPacketSize](#)  
*Maximum packet size in bytes for current mode.*
- float [percentage](#)  
*Current packet size as a percentage of maximum packet size.*
- unsigned int [reserved](#) [16]  
*Reserved for future use.*

### 8.18.1 Detailed Description

Format 7 information for a single mode.

### 8.18.2 Constructor & Destructor Documentation

#### 8.18.2.1 `Format7Info ()` `[inline]`

### 8.18.3 Member Data Documentation

#### 8.18.3.1 `unsigned int imageHStepSize`

Horizontal step size for the image.

#### 8.18.3.2 `unsigned int imageVStepSize`

Vertical step size for the image.

#### 8.18.3.3 `unsigned int maxHeight`

Maximum image height.

#### 8.18.3.4 `unsigned int maxPacketSize`

Maximum packet size in bytes for current mode.

#### 8.18.3.5 `unsigned int maxWidth`

Maximum image width.

#### 8.18.3.6 `unsigned int minPacketSize`

Minimum packet size in bytes for current mode.

#### 8.18.3.7 `Mode mode`

Format 7 mode.

#### 8.18.3.8 `unsigned int offsetHStepSize`

Horizontal step size for the offset.

#### 8.18.3.9 `unsigned int offsetVStepSize`

Vertical step size for the offset.

**8.18.3.10 unsigned int packetSize**

Current packet size in bytes.

**8.18.3.11 float percentage**

Current packet size as a percentage of maximum packet size.

**8.18.3.12 unsigned int pixelFormatBitField**

Supported pixel formats in a bit field.

**8.18.3.13 unsigned int reserved[16]**

Reserved for future use.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.19 Format7PacketInfo Struct Reference

Format 7 packet information.

### Public Member Functions

- [Format7PacketInfo \(\)](#)

### Public Attributes

- unsigned int [recommendedBytesPerPacket](#)  
*Recommended bytes per packet.*
- unsigned int [maxBytesPerPacket](#)  
*Maximum bytes per packet.*
- unsigned int [unitBytesPerPacket](#)  
*Minimum bytes per packet.*
- unsigned int [reserved](#) [8]  
*Reserved for future use.*

### 8.19.1 Detailed Description

Format 7 packet information.

### 8.19.2 Constructor & Destructor Documentation

#### 8.19.2.1 [Format7PacketInfo \(\)](#) `[inline]`

### 8.19.3 Member Data Documentation

#### 8.19.3.1 unsigned int [maxBytesPerPacket](#)

Maximum bytes per packet.

#### 8.19.3.2 unsigned int [recommendedBytesPerPacket](#)

Recommended bytes per packet.

#### 8.19.3.3 unsigned int [reserved](#)[8]

Reserved for future use.

#### 8.19.3.4 unsigned int unitBytesPerPacket

Minimum bytes per packet.

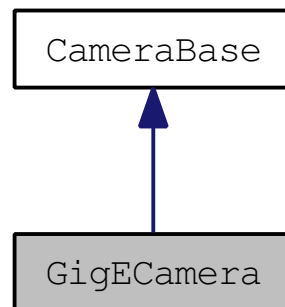
The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

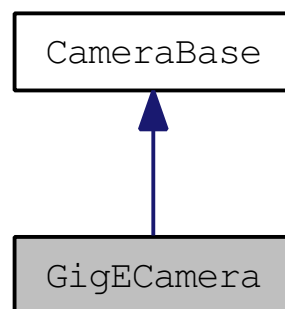
## 8.20 GigECamera Class Reference

The [GigECamera](#) object represents a physical Gigabit Ethernet camera.

Inheritance diagram for GigECamera:



Collaboration diagram for GigECamera:



### Public Member Functions

- [GigECamera](#) ()  
*Default constructor.*
- virtual [~GigECamera](#) ()  
*Default destructor.*
- virtual [Error Connect](#) ([PGRGuid](#) \*pGuid=NULL)  
*The following functions are inherited from [CameraBase](#).*
- virtual [Error Disconnect](#) ()  
*Disconnects the camera object from the camera.*
- virtual bool [IsConnected](#) ()  
*Checks if the camera object is currently connected to a physical camera.*
- virtual [Error SetCallback](#) ([ImageEventCallback](#) callbackFn, const void \*pCallbackData=NULL)

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

- virtual [Error StartCapture](#) ([ImageEventCallback](#) callbackFn=NULL, const void \*pCallbackData=NULL)

*Starts isochronous image capture.*

- virtual [Error RetrieveBuffer](#) ([Image](#) \*pImage)

*Retrieves the the next image object containing the next image.*

- virtual [Error StopCapture](#) ()

*Stops isochronous image transfer and cleans up all associated resources.*

- virtual [Error WaitForBufferEvent](#) ([Image](#) \*pImage, unsigned int eventNumber)

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

- virtual [Error SetUserBuffers](#) (unsigned char \*const pMemBuffers, int size, int numBuffers)

*Specify user allocated buffers to use as image data buffers.*

- virtual [Error GetConfiguration](#) ([FC2Config](#) \*pConfig)

*Get the configuration associated with the camera object.*

- virtual [Error SetConfiguration](#) (const [FC2Config](#) \*pConfig)

*Set the configuration associated with the camera object.*

- virtual [Error GetCameraInfo](#) ([CameraInfo](#) \*pCameraInfo)

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

- virtual [Error GetPropertyInfo](#) ([PropertyInfo](#) \*pPropInfo)

*Retrieves information about the specified camera property.*

- virtual [Error GetProperty](#) ([Property](#) \*pProp)

*Reads the settings for the specified property from the camera.*

- virtual [Error SetProperty](#) (const [Property](#) \*pProp, bool broadcast=false)

*Writes the settings for the specified property to the camera.*

- virtual [Error GetGPIOPinDirection](#) (unsigned int pin, unsigned int \*pDirection)

*Get the GPIO pin direction for the specified pin.*

- virtual [Error SetGPIOPinDirection](#) (unsigned int pin, unsigned int direction, bool broadcast=false)

*Set the GPIO pin direction for the specified pin.*

- virtual [Error GetTriggerModeInfo](#) ([TriggerModeInfo](#) \*pTriggerModeInfo)

*Retrieve trigger information from the camera.*

- virtual [Error GetTriggerMode](#) ([TriggerMode](#) \*pTriggerMode)

*Retrieve current trigger settings from the camera.*

- virtual [Error SetTriggerMode](#) (const [TriggerMode](#) \*pTriggerMode, bool broadcast=false)

*Set the specified trigger settings to the camera.*

- virtual [Error FireSoftwareTrigger](#) (bool broadcast=false)  
*Fire the software trigger according to the DCAM specifications.*
- virtual [Error GetTriggerDelayInfo](#) ([TriggerDelayInfo](#) \*pTriggerDelayInfo)  
*Retrieve trigger delay information from the camera.*
- virtual [Error GetTriggerDelay](#) ([TriggerDelay](#) \*pTriggerDelay)  
*Retrieve current trigger delay settings from the camera.*
- virtual [Error SetTriggerDelay](#) (const [TriggerDelay](#) \*pTriggerDelay, bool broadcast=false)  
*Set the specified trigger delay settings to the camera.*
- virtual [Error GetStrobeInfo](#) ([StrobeInfo](#) \*pStrobeInfo)  
*Retrieve strobe information from the camera.*
- virtual [Error GetStrobe](#) ([StrobeControl](#) \*pStrobeControl)  
*Retrieve current strobe settings from the camera.*
- virtual [Error SetStrobe](#) (const [StrobeControl](#) \*pStrobeControl, bool broadcast=false)  
*Set current strobe settings to the camera.*
- virtual [Error GetLUTInfo](#) ([LUTData](#) \*pData)  
*Query if LUT support is available on the camera.*
- virtual [Error GetLUTBankInfo](#) (unsigned int bank, bool \*pReadSupported, bool \*pWriteSupported)  
*Query the read/write status of a single LUT bank.*
- virtual [Error GetActiveLUTBank](#) (unsigned int \*pActiveBank)  
*Get the LUT bank that is currently being used.*
- virtual [Error SetActiveLUTBank](#) (unsigned int activeBank)  
*Set the LUT bank that will be used.*
- virtual [Error EnableLUT](#) (bool on)  
*Enable or disable LUT functionality on the camera.*
- virtual [Error GetLUTChannel](#) (unsigned int bank, unsigned int channel, unsigned int sizeEntries, unsigned int \*pEntries)  
*Get the LUT channel settings from the camera.*
- virtual [Error SetLUTChannel](#) (unsigned int bank, unsigned int channel, unsigned int sizeEntries, const unsigned int \*pEntries)  
*Set the LUT channel settings to the camera.*
- virtual [Error GetMemoryChannel](#) (unsigned int \*pCurrentChannel)  
*Retrieve the current memory channel from the camera.*
- virtual [Error SaveToMemoryChannel](#) (unsigned int channel)



*Save the current settings to the specified current memory channel.*

- virtual [Error RestoreFromMemoryChannel](#) (unsigned int channel)  
*Restore the specified current memory channel.*
- virtual [Error GetMemoryChannelInfo](#) (unsigned int \*pNumChannels)  
*Query the camera for memory channel support.*
- virtual [Error GetEmbeddedImageInfo](#) ([EmbeddedImageInfo](#) \*pInfo)  
*Get the current status of the embedded image information register, as well as the availability of each embedded property.*
- virtual [Error SetEmbeddedImageInfo](#) ([EmbeddedImageInfo](#) \*pInfo)  
*Sets the on/off values of the embedded image information structure to the camera.*
- virtual [Error WriteRegister](#) (unsigned int address, unsigned int value, bool broadcast=false)  
*Write to the specified register on the camera.*
- virtual [Error ReadRegister](#) (unsigned int address, unsigned int \*pValue)  
*Read the specified register from the camera.*
- virtual [Error WriteRegisterBlock](#) (unsigned short addressHigh, unsigned int addressLow, const unsigned int \*pBuffer, unsigned int length)  
*Write to the specified register block on the camera.*
- virtual [Error ReadRegisterBlock](#) (unsigned short addressHigh, unsigned int addressLow, unsigned int \*pBuffer, unsigned int length)  
*Read from the specified register block on the camera.*

## Static Public Member Functions

- static [Error StartSyncCapture](#) (unsigned int numCameras, const [GigECamera](#) \*\*ppCameras, const [ImageEventCallback](#) \*pCallbackFns=NULL, const void \*\*pCallbackdataArray=NULL)
- static const char \* [GetRegisterString](#) (unsigned int registerVal)  
*Returns a text representation of the register value.*

## GVCP Register Operation

These functions deal with GVCP register operation on the camera.

- virtual [Error WriteGVCPRegister](#) (unsigned int address, unsigned int value, bool broadcast=false)  
*Write a GVCP register.*
- virtual [Error ReadGVCPRegister](#) (unsigned int address, unsigned int \*pValue)  
*Read a GVCP register.*
- virtual [Error WriteGVCPRegisterBlock](#) (unsigned int address, const unsigned int \*pBuffer, unsigned int length)

*Write a GVCP register block.*

- virtual [Error ReadGVCPRegisterBlock](#) (unsigned int address, unsigned int \*pBuffer, unsigned int length)

*Read a GVCP register block.*

- virtual [Error WriteGVCPMemory](#) (unsigned int address, const unsigned char \*pBuffer, unsigned int length)

*Write a GVCP Memory block.*

- virtual [Error ReadGVCPMemory](#) (unsigned int address, unsigned char \*pBuffer, unsigned int length)

*Read a GVCP memory block.*

## GigE property manipulation

These functions deal with GigE properties.

- virtual [Error GetGigEProperty](#) (GigEProperty \*pGigEProp)

*Get the specified GigEProperty.*

- virtual [Error SetGigEProperty](#) (const GigEProperty \*pGigEProp)

*Set the specified GigEProperty.*

- virtual [Error DiscoverGigEPacketSize](#) (unsigned int \*packetSize)

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

## GigE image settings

These functions deal with GigE image setting.

- virtual [Error QueryGigEImagingMode](#) (Mode mode, bool \*isSupported)

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

- virtual [Error GetGigEImagingMode](#) (Mode \*mode)

*Get the current imaging mode on the camera.*

- virtual [Error SetGigEImagingMode](#) (Mode mode)

*Set the current imaging mode to the camera.*

- virtual [Error GetGigEImageSettingsInfo](#) (GigEImageSettingsInfo \*pInfo)

*Get information about the image settings possible on the camera.*

- virtual [Error GetGigEImageSettings](#) (GigEImageSettings \*pImageSettings)

*Get the current image settings on the camera.*

- virtual [Error SetGigEImageSettings](#) (const GigEImageSettings \*pImageSettings)

*Set the image settings specified to the camera.*

## GigE image binning settings

These functions deal with GigE image binning settings.

- virtual [Error](#) [GetGigEImageBinningSettings](#) (unsigned int \*horzBinningValue, unsigned int \*vertBinningValue)  
*Get the current binning settings on the camera.*
- virtual [Error](#) [SetGigEImageBinningSettings](#) (unsigned int horzBinningValue, unsigned int vertBinningValue)  
*Set the specified binning values to the camera.*

## GigE image stream configuration

These functions deal with GigE image stream configuration.

- virtual [Error](#) [GetNumStreamChannels](#) (unsigned int \*numChannels)  
*Get the number of stream channels present on the camera.*
- virtual [Error](#) [GetGigEStreamChannelInfo](#) (unsigned int channel, [GigEStreamChannel](#) \*pChannel)  
*Get the stream channel information for the specified channel.*
- virtual [Error](#) [SetGigEStreamChannelInfo](#) (unsigned int channel, [GigEStreamChannel](#) \*pChannel)  
*Set the stream channel information for the specified channel.*

### 8.20.1 Detailed Description

The [GigECamera](#) object represents a physical Gigabit Ethernet camera.

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

Please see [Camera.h](#) for basic functions that this class inherits from.

### 8.20.2 Constructor & Destructor Documentation

#### 8.20.2.1 GigECamera ()

Default constructor.

#### 8.20.2.2 virtual ~GigECamera () [virtual]

Default destructor.

### 8.20.3 Member Function Documentation

#### 8.20.3.1 virtual Error Connect (PGRGuid \*pGuid = NULL) [virtual]

The following functions are inherited from [CameraBase](#).

See [CameraBase.h](#) for further information.

Implements [CameraBase](#).

### 8.20.3.2 virtual Error Disconnect () [virtual]

Disconnects the camera object from the camera.

This allows another physical camera to be connected to the camera object.

**See also:**

[Connect\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

### 8.20.3.3 virtual Error DiscoverGigEPacketSize (unsigned int \* *packetSize*) [virtual]

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

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

**Parameters:**

*packetSize* The maximum packet size supported by the link.

**Returns:**

An [Error](#) indicating the success or failure of the function.

### 8.20.3.4 virtual Error EnableLUT (bool *on*) [virtual]

Enable or disable LUT functionality on the camera.

**Parameters:**

*on* Whether to enable or disable LUT.

**See also:**

[GetLUTInfo\(\)](#)  
[GetLUTChannel\(\)](#)  
[SetLUTChannel\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.5 virtual Error FireSoftwareTrigger (bool *broadcast* = false) [virtual]**

Fire the software trigger according to the DCAM specifications.

**Parameters:**

*broadcast* Whether the action should be broadcast.

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.6 virtual Error GetActiveLUTBank (unsigned int \**pActiveBank*) [virtual]**

Get the LUT bank that is currently being used.

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

**Parameters:**

*pActiveBank* The currently active bank.

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.7 virtual Error GetCameraInfo (CameraInfo \**pCameraInfo*) [virtual]**

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

**Parameters:**

*pCameraInfo* Pointer to the camera information structure to be filled.

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.8 virtual Error GetConfiguration (FC2Config \**pConfig*) [virtual]**

Get the configuration associated with the camera object.

**Parameters:**

*pConfig* Pointer to the configuration structure to be filled.

**See also:**

[SetConfiguration\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.9 virtual Error GetEmbeddedImageInfo (EmbeddedImageInfo \* *pInfo*) [virtual]**

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

**Parameters:**

*pInfo* Structure to be filled.

**See also:**

[SetEmbeddedImageInfo\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.10 virtual Error GetGigEImageBinningSettings (unsigned int \* *horzBinnningValue*, unsigned int \* *vertBinnningValue*) [virtual]**

Get the current binning settings on the camera.

**Parameters:**

*horzBinnningValue* Current horizontal binning value.

*vertBinnningValue* Current vertical binning value.

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.20.3.11 virtual Error GetGigEImageSettings (GigEImageSettings \* *pImageSettings*) [virtual]**

Get the current image settings on the camera.

**Parameters:**

*pImageSettings* Current image settings on camera.

**Returns:**

An [Error](#) indicating the success or failure of the function.

### 8.20.3.12 virtual Error GetGigEImageSettingsInfo (GigEImageSettingsInfo \* *pInfo*) [virtual]

Get information about the image settings possible on the camera.

#### Parameters:

*pInfo* Image settings information.

#### Returns:

An [Error](#) indicating the success or failure of the function.

### 8.20.3.13 virtual Error GetGigEImagingMode (Mode \* *mode*) [virtual]

Get the current imaging mode on the camera.

#### Parameters:

*mode* Current imaging mode on the camera.

#### Returns:

An [Error](#) indicating the success or failure of the function.

### 8.20.3.14 virtual Error GetGigEProperty (GigEProperty \* *pGigEProp*) [virtual]

Get the specified [GigEProperty](#).

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

#### Parameters:

*pGigEProp* The GigE property to get.

#### Returns:

An [Error](#) indicating the success or failure of the function.

### 8.20.3.15 virtual Error GetGigEStreamChannelInfo (unsigned int *channel*, GigEStreamChannel \* *pChannel*) [virtual]

Get the stream channel information for the specified channel.

#### Parameters:

*channel* Channel number to use.

*pChannel* Stream channel information for the specified channel.

#### Returns:

An [Error](#) indicating the success or failure of the function.

### 8.20.3.16 virtual Error GetGPIOPinDirection (unsigned int *pin*, unsigned int \* *pDirection*) [virtual]

Get the GPIO pin direction for the specified pin.

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

#### Parameters:

*pin* Pin to get the direction for.

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

#### See also:

[SetGPIOPinDirection\(\)](#)

#### Returns:

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

### 8.20.3.17 virtual Error GetLUTBankInfo (unsigned int *bank*, bool \* *pReadSupported*, bool \* *pWriteSupported*) [virtual]

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

#### Parameters:

*bank* The bank to query.

*pReadSupported* Whether reading from the bank is supported.

*pWriteSupported* Whether writing to the bank is supported.

#### Returns:

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

### 8.20.3.18 virtual Error GetLUTChannel (unsigned int *bank*, unsigned int *channel*, unsigned int *sizeEntries*, unsigned int \* *pEntries*) [virtual]

Get the LUT channel settings from the camera.

#### Parameters:

*bank* Bank to retrieve.

*channel* Channel to retrieve.

*sizeEntries* Number of entries in LUT table to read.

*pEntries* Array to store LUT entries.



**See also:**

[GetLUTInfo\(\)](#)  
[EnableLUT\(\)](#)  
[SetLUTChannel\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.19 virtual Error GetLUTInfo (LUTData \* *pData*)** [virtual]

Query if LUT support is available on the camera.

**Parameters:**

*pData* The LUT structure to be filled.

**See also:**

[EnableLUT\(\)](#)  
[GetLUTChannel\(\)](#)  
[SetLUTChannel\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.20 virtual Error GetMemoryChannel (unsigned int \* *pCurrentChannel*)** [virtual]

Retrieve the current memory channel from the camera.

**Parameters:**

*pCurrentChannel* Current memory channel.

**See also:**

[SaveToMemoryChannel\(\)](#)  
[RestoreFromMemoryChannel\(\)](#)  
[GetMemoryChannelInfo\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.21 virtual Error GetMemoryChannelInfo (unsigned int \* *pNumChannels*)** [virtual]

Query the camera for memory channel support.

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

**Parameters:**

*pNumChannels* Number of memory channels supported.

**See also:**

[GetMemoryChannel\(\)](#)  
[SaveToMemoryChannel\(\)](#)  
[RestoreFromMemoryChannel\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.22 virtual Error GetNumStreamChannels (unsigned int \* *numChannels*)** [virtual]

Get the number of stream channels present on the camera.

**Parameters:**

*numChannels* Number of stream channels present.

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.20.3.23 virtual Error GetProperty (Property \* *pProp*)** [virtual]

Reads the settings for the specified property from the camera.

The property type must be specified in the [Property](#) structure passed into the function in order for the function to succeed. If auto is on, the integer and abs values returned may not be consistent with each other.

**Parameters:**

*pProp* Pointer to the [Property](#) structure to be filled.

**See also:**

[GetPropertyInfo\(\)](#)  
[SetProperty\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.24 virtual Error GetPropertyInfo (PropertyInfo \* *pPropInfo*)** [virtual]

Retrieves information about the specified camera property.

The property type must be specified in the [PropertyInfo](#) structure passed into the function in order for the function to succeed.

**Parameters:**

*pPropInfo* Pointer to the [PropertyInfo](#) structure to be filled.

**See also:**

[GetProperty\(\)](#)  
[SetProperty\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.25 static const char\* GetRegisterString (unsigned int *registerVal*)** [static]

Returns a text representation of the register value.

**Parameters:**

*registerVal* The register value to query.

**Returns:**

The text representation of the register.

Reimplemented from [CameraBase](#).

**8.20.3.26 virtual Error GetStrobe (StrobeControl \* *pStrobeControl*)** [virtual]

Retrieve current strobe settings from the camera.

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

**Parameters:**

*pStrobeControl* Structure to receive strobe settings.

**See also:**

[GetStrobeInfo\(\)](#)  
[SetStrobe\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.27 virtual Error GetStrobeInfo (StrobeInfo \* *pStrobeInfo*)** [virtual]

Retrieve strobe information from the camera.

**Parameters:**

*pStrobeInfo* Structure to receive strobe information.

**See also:**

[GetStrobe\(\)](#)  
[SetStrobe\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.28 virtual Error GetTriggerDelay (TriggerDelay \* *pTriggerDelay*)** [virtual]

Retrieve current trigger delay settings from the camera.

**Parameters:**

*pTriggerDelay* Structure to receive trigger delay settings.

**See also:**

[GetTriggerModeInfo\(\)](#)  
[GetTriggerMode\(\)](#)  
[SetTriggerMode\(\)](#)  
[GetTriggerDelayInfo\(\)](#)  
[SetTriggerDelay\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.29 virtual Error GetTriggerDelayInfo (TriggerDelayInfo \* *pTriggerDelayInfo*)**  
[virtual]

Retrieve trigger delay information from the camera.

**Parameters:**

*pTriggerDelayInfo* Structure to receive trigger delay information.

**See also:**

[GetTriggerModeInfo\(\)](#)  
[GetTriggerMode\(\)](#)  
[SetTriggerMode\(\)](#)  
[GetTriggerDelay\(\)](#)  
[SetTriggerDelay\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.30 virtual Error GetTriggerMode (TriggerMode \* *pTriggerMode*) [virtual]**

Retrieve current trigger settings from the camera.

**Parameters:**

*pTriggerMode* Structure to receive trigger mode settings.

**See also:**

[GetTriggerModeInfo\(\)](#)  
[SetTriggerMode\(\)](#)  
[GetTriggerDelayInfo\(\)](#)  
[GetTriggerDelay\(\)](#)  
[SetTriggerDelay\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.31 virtual Error GetTriggerModeInfo (TriggerModeInfo \* *pTriggerModeInfo*) [virtual]**

Retrieve trigger information from the camera.

**Parameters:**

*pTriggerModeInfo* Structure to receive trigger information.

**See also:**

[GetTriggerMode\(\)](#)  
[SetTriggerMode\(\)](#)  
[GetTriggerDelayInfo\(\)](#)  
[GetTriggerDelay\(\)](#)  
[SetTriggerDelay\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.32 virtual bool IsConnected () [virtual]**

Checks if the camera object is currently connected to a physical camera.

See also:

[Connect\(\)](#)  
[Disconnect\(\)](#)

Returns:

Whether the camera object is connected to a physical camera.

Implements [CameraBase](#).

#### 8.20.3.33 virtual Error QueryGigEImagingMode (Mode *mode*, bool \* *isSupported*) [virtual]

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

Parameters:

*mode* The mode to check.

*isSupported* Whether the mode is supported.

Returns:

An [Error](#) indicating the success or failure of the function.

#### 8.20.3.34 virtual Error ReadGVCPMemory (unsigned int *address*, unsigned char \* *pBuffer*, unsigned int *length*) [virtual]

Read a GVCP memory block.

Parameters:

*address* GVCP address to be read from.

*pBuffer* Array for data to be read into.

*length* Size of array, in quadlets.

Returns:

An [Error](#) indicating the success or failure of the function.

#### 8.20.3.35 virtual Error ReadGVCPRegister (unsigned int *address*, unsigned int \* *pValue*) [virtual]

Read a GVCP register.

Parameters:

*address* GVCP address to be read from.

*pValue* The value that is read.

Returns:

An [Error](#) indicating the success or failure of the function.

**8.20.3.36 virtual Error ReadGVCPRegisterBlock (unsigned int *address*, unsigned int \* *pBuffer*, unsigned int *length*)** [virtual]

Read a GVCP register block.

**Parameters:**

*address* GVCP address to be read from.  
*pBuffer* Array for data to be read into.  
*length* Size of array, in quadlets.

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.20.3.37 virtual Error ReadRegister (unsigned int *address*, unsigned int \* *pValue*)** [virtual]

Read the specified register from the camera.

**Parameters:**

*address* DCAM address to be read from.  
*pValue* The value that is read.

**See also:**

[WriteRegister\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.38 virtual Error ReadRegisterBlock (unsigned short *addressHigh*, unsigned int *addressLow*, unsigned int \* *pBuffer*, unsigned int *length*)** [virtual]

Read from the specified register block on the camera.

**Parameters:**

*addressHigh* Top 16 bits of the 48 bit absolute address to read from.  
*addressLow* Bottom 32 bits of the 48 bits absolute address to read from.  
*pBuffer* Array to store read data.  
*length* Size of array, in quadlets.

**See also:**

[WriteRegisterBlock\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.39 virtual Error RestoreFromMemoryChannel (unsigned int *channel*)** [virtual]

Restore the specified current memory channel.

**Parameters:**

*channel* Memory channel to restore from.

**See also:**

[GetMemoryChannel\(\)](#)  
[SaveToMemoryChannel\(\)](#)  
[GetMemoryChannelInfo\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.40 virtual Error RetrieveBuffer (Image \**pImage*)** [virtual]

Retrieves the the next image object containing the next image.

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

**Parameters:**

*pImage* Pointer to [Image](#) object to store image data.

**See also:**

[StartCapture\(\)](#)  
[StopCapture\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.41 virtual Error SaveToMemoryChannel (unsigned int *channel*)** [virtual]

Save the current settings to the specified current memory channel.

**Parameters:**

*channel* Memory channel to save to.

**See also:**

[GetMemoryChannel\(\)](#)  
[RestoreFromMemoryChannel\(\)](#)  
[GetMemoryChannelInfo\(\)](#)



**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.42 virtual Error SetActiveLUTBank (unsigned int *activeBank*) [virtual]**

Set the LUT bank that will be used.

**Parameters:**

*activeBank* The bank to be set as active.

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.43 virtual Error SetCallback (ImageEventCallback *callbackFn*, const void \**pCallbackData* = NULL) [virtual]**

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

To clear the current stored callback data, pass in NULL for both arguments.

**Parameters:**

*callbackFn* A function to be called when a new image is received.

*pCallbackData* A pointer to data that can be passed to the callback function.

**See also:**

[StartCapture\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.44 virtual Error SetConfiguration (const FC2Config \**pConfig*) [virtual]**

Set the configuration associated with the camera object.

**Parameters:**

*pConfig* Pointer to the configuration structure to be used.

**See also:**

[GetConfiguration\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.45 virtual Error SetEmbeddedImageInfo (EmbeddedImageInfo \* *pInfo*)** [virtual]

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

**Parameters:**

*pInfo* Structure to be used.

**See also:**

[GetEmbeddedImageInfo\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.46 virtual Error SetGigEImageBinningSettings (unsigned int *horzBinningValue*, unsigned int *vertBinningValue*)** [virtual]

Set the specified binning values to the camera.

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

**Parameters:**

*horzBinningValue* Horizontal binning value.

*vertBinningValue* Vertical binning value.

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.20.3.47 virtual Error SetGigEImageSettings (const GigEImageSettings \* *pImageSettings*)** [virtual]

Set the image settings specified to the camera.

**Parameters:**

*pImageSettings* [Image](#) settings to set to camera.

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.20.3.48 virtual Error SetGigEImagingMode (Mode *mode*)** [virtual]

Set the current imaging mode to the camera.

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

**Parameters:**

*mode* Imaging mode to set to the camera.

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.20.3.49 virtual Error SetGigEProperty (const GigEProperty \* *pGigEProp*) [virtual]**

Set the specified [GigEProperty](#).

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

**Parameters:**

*pGigEProp* The GigE property to set.

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.20.3.50 virtual Error SetGigEStreamChannelInfo (unsigned int *channel*, GigEStreamChannel \* *pChannel*) [virtual]**

Set the stream channel information for the specified channel.

**Parameters:**

*channel* Channel number to use.

*pChannel* Stream channel information to use for the specified channel.

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.20.3.51 virtual Error SetGPIOPinDirection (unsigned int *pin*, unsigned int *direction*, bool *broadcast* = false) [virtual]**

Set the GPIO pin direction for the specified pin.

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

**Parameters:**

*pin* Pin to get the direction for.

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

*broadcast* Whether the action should be broadcast.

**See also:**

[GetGPIOPinDirection\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.52 virtual Error SetLUTChannel (unsigned int *bank*, unsigned int *channel*, unsigned int *sizeEntries*, const unsigned int \**pEntries*) [virtual]**

Set the LUT channel settings to the camera.

**Parameters:**

*bank* Bank to set.

*channel* Channel to set.

*sizeEntries* Number of entries in LUT table to write.

*pEntries* Array containing LUT entries to write.

**See also:**

[GetLUTInfo\(\)](#)

[EnableLUT\(\)](#)

[GetLUTChannel\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.53 virtual Error SetProperty (const Property \**pProp*, bool *broadcast* = false) [virtual]**

Writes the settings for the specified property to the camera.

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

**Parameters:**

*pProp* Pointer to the [Property](#) structure to be used.

*broadcast* Whether the action should be broadcast.

**See also:**

[GetPropertyInfo\(\)](#)

[GetProperty\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.54 virtual Error SetStrobe (const StrobeControl \* *pStrobeControl*, bool *broadcast* = false)**  
[virtual]

Set current strobe settings to the camera.

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

**Parameters:**

*pStrobeControl* Structure providing strobe settings.

*broadcast* Whether the action should be broadcast.

**See also:**

[GetStrobeInfo\(\)](#)

[GetStrobe\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.55 virtual Error SetTriggerDelay (const TriggerDelay \* *pTriggerDelay*, bool *broadcast* = false)**  
[virtual]

Set the specified trigger delay settings to the camera.

**Parameters:**

*pTriggerDelay* Structure providing trigger delay settings.

*broadcast* Whether the action should be broadcast.

**See also:**

[GetTriggerModeInfo\(\)](#)

[GetTriggerMode\(\)](#)

[SetTriggerMode\(\)](#)

[GetTriggerDelayInfo\(\)](#)

[GetTriggerDelay\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.56 virtual Error SetTriggerMode (const TriggerMode \* *pTriggerMode*, bool *broadcast* = false)**  
[virtual]

Set the specified trigger settings to the camera.

**Parameters:**

*pTriggerMode* Structure providing trigger mode settings.

*broadcast* Whether the action should be broadcast.

See also:

[GetTriggerModeInfo\(\)](#)  
[GetTriggerMode\(\)](#)  
[GetTriggerDelayInfo\(\)](#)  
[GetTriggerDelay\(\)](#)  
[SetTriggerDelay\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

### 8.20.3.57 virtual Error SetUserBuffers (unsigned char \*const *pMemBuffers*, int *size*, int *numBuffers*) [virtual]

Specify user allocated buffers to use as image data buffers.

To prevent image tearing, the size of each buffer should be equal to ((unsigned int)(bufferSize + packetSize - 1)/packetSize) \* packetSize. The total size should be (size \* numBuffers) or larger.

Parameters:

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

See also:

[StartCapture\(\)](#)  
[RetrieveBuffer\(\)](#)  
[StopCapture\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

### 8.20.3.58 virtual Error StartCapture (ImageEventCallback *callbackFn* = NULL, const void \* *pCallbackData* = NULL) [virtual]

Starts isochronous image capture.

It will use either the current video mode or the most recently set video mode of the camera. The optional callback function parameter is called on completion of image transfer. Alternatively, the callback parameter can be set to NULL and [RetrieveBuffer\(\)](#) can be called as a blocking call to get the image data.

Parameters:

*callbackFn* A function to be called when a new image is received.

*pCallbackData* A pointer to data that can be passed to the callback function.

See also:

[RetrieveBuffer\(\)](#)  
[StartSyncCapture\(\)](#)  
[StopCapture\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.59** static [Error](#) StartSyncCapture (unsigned int *numCameras*, const [GigECamera](#) \*\*  
*ppCameras*, const [ImageEventCallback](#) \* *pCallbackFns* = NULL, const void \*\*  
*pCallbackDataArray* = NULL) [static]

**8.20.3.60** virtual [Error](#) StopCapture () [virtual]

Stops isochronous image transfer and cleans up all associated resources.

See also:

[StartCapture\(\)](#)  
[RetrieveBuffer\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.61** virtual [Error](#) WaitForBufferEvent ([Image](#) \* *pImage*, unsigned int *eventNumber*)  
[virtual]

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

Parameters:

*pImage* Pointer to [Image](#) object to store image data.  
*eventNumber* The event number to wait for.

See also:

[StartCapture\(\)](#)  
[RetrieveBuffer\(\)](#)  
[StopCapture\(\)](#)

Returns:

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.62 virtual Error WriteGVCPMemory (unsigned int *address*, const unsigned char \**pBuffer*, unsigned int *length*)** [virtual]

Write a GVCP Memory block.

**Parameters:**

*address* GVCP address to be write to.

*pBuffer* Array containing data to be written in increments.

*length* Size of array, in quadlets.

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.20.3.63 virtual Error WriteGVCPRegister (unsigned int *address*, unsigned int *value*, bool *broadcast* = false)** [virtual]

Write a GVCP register.

**Parameters:**

*address* GVCP address to be written to.

*value* The value to be written.

*broadcast* Whether the action should be broadcast.

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.20.3.64 virtual Error WriteGVCPRegisterBlock (unsigned int *address*, const unsigned int \**pBuffer*, unsigned int *length*)** [virtual]

Write a GVCP register block.

**Parameters:**

*address* GVCP address to be write to.

*pBuffer* Array containing data to be written.

*length* Size of array, in quadlets.

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.20.3.65 virtual Error WriteRegister (unsigned int *address*, unsigned int *value*, bool *broadcast* = false)** [virtual]

Write to the specified register on the camera.



**Parameters:**

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

**See also:**

[ReadRegister\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

**8.20.3.66 virtual Error WriteRegisterBlock (unsigned short *addressHigh*, unsigned int *addressLow*, const unsigned int \* *pBuffer*, unsigned int *length*)** [virtual]

Write to the specified register block on the camera.

**Parameters:**

*addressHigh* Top 16 bits of the 48 bit absolute address to write to.  
*addressLow* Bottom 32 bits of the 48 bits absolute address to write to.  
*pBuffer* Array containing data to be written.  
*length* Size of array, in quadlets.

**See also:**

[ReadRegisterBlock\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

Implements [CameraBase](#).

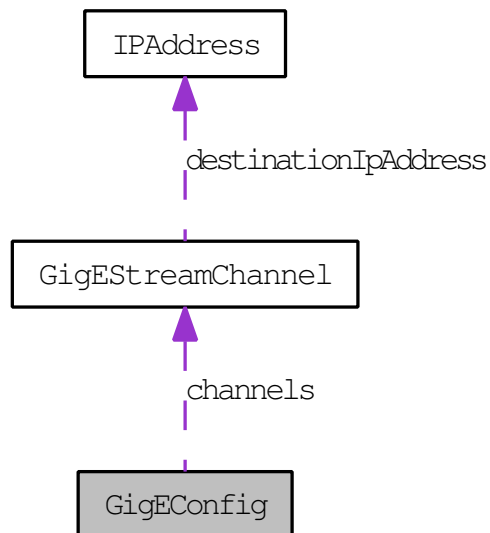
The documentation for this class was generated from the following file:

- [GigECamera.h](#)

## 8.21 GigEConfig Struct Reference

Configuration for a GigE camera.

Collaboration diagram for GigEConfig:



### Public Member Functions

- [GigEConfig](#) ()

### Public Attributes

- unsigned int [numChannels](#)  
*Number of stream channels.*
- [GigEStreamChannel](#) `channels` [512]  
*Array of stream channel data.*

#### 8.21.1 Detailed Description

Configuration for a GigE camera.

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

## 8.21.2 Constructor & Destructor Documentation

### 8.21.2.1 GigEConfig () `[inline]`

## 8.21.3 Member Data Documentation

### 8.21.3.1 GigEStreamChannel channels[512]

Array of stream channel data.

### 8.21.3.2 unsigned int numChannels

Number of stream channels.

Read only.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.22 GigEImageSettings Struct Reference

[Image](#) settings for a GigE camera.

### Public Member Functions

- [GigEImageSettings](#) ()

### Public Attributes

- unsigned int [offsetX](#)  
*Horizontal image offset.*
- unsigned int [offsetY](#)  
*Vertical image offset.*
- unsigned int [width](#)  
*Width of image.*
- unsigned int [height](#)  
*Height of image.*
- [PixelFormat](#) [pixelFormat](#)  
*Pixel format of image.*
- unsigned int [reserved](#) [8]  
*Reserved for future use.*

### 8.22.1 Detailed Description

[Image](#) settings for a GigE camera.

### 8.22.2 Constructor & Destructor Documentation

#### 8.22.2.1 [GigEImageSettings](#) () `[inline]`

### 8.22.3 Member Data Documentation

#### 8.22.3.1 unsigned int height

Height of image.

#### 8.22.3.2 unsigned int offsetX

Horizontal image offset.

**8.22.3.3 unsigned int offsetY**

Vertical image offset.

**8.22.3.4 PixelFormat pixelFormat**

Pixel format of image.

**8.22.3.5 unsigned int reserved[8]**

Reserved for future use.

**8.22.3.6 unsigned int width**

Width of image.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.23 GigEImageSettingsInfo Struct Reference

Format 7 information for a single mode.

### Public Member Functions

- [GigEImageSettingsInfo](#) ()

### Public Attributes

- unsigned int [maxWidth](#)  
*Maximum image width.*
- unsigned int [maxHeight](#)  
*Maximum image height.*
- unsigned int [offsetHStepSize](#)  
*Horizontal step size for the offset.*
- unsigned int [offsetVStepSize](#)  
*Vertical step size for the offset.*
- unsigned int [imageHStepSize](#)  
*Horizontal step size for the image.*
- unsigned int [imageVStepSize](#)  
*Vertical step size for the image.*
- unsigned int [pixelFormatBitField](#)  
*Supported pixel formats in a bit field.*
- unsigned int [reserved](#) [16]  
*Reserved for future use.*

### 8.23.1 Detailed Description

Format 7 information for a single mode.

### 8.23.2 Constructor & Destructor Documentation

#### 8.23.2.1 [GigEImageSettingsInfo](#) () [inline]

### 8.23.3 Member Data Documentation

#### 8.23.3.1 unsigned int [imageHStepSize](#)

Horizontal step size for the image.

**8.23.3.2 unsigned int imageVStepSize**

Vertical step size for the image.

**8.23.3.3 unsigned int maxHeight**

Maximum image height.

**8.23.3.4 unsigned int maxWidth**

Maximum image width.

**8.23.3.5 unsigned int offsetHStepSize**

Horizontal step size for the offset.

**8.23.3.6 unsigned int offsetVStepSize**

Vertical step size for the offset.

**8.23.3.7 unsigned int pixelFormatBitField**

Supported pixel formats in a bit field.

**8.23.3.8 unsigned int reserved[16]**

Reserved for future use.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.24 GigEProperty Struct Reference

A GigE property.

### Public Attributes

- [GigEPropertyType propType](#)  
*The type of property.*
- [bool isReadable](#)  
*Whether the property is readable.*
- [bool isWritable](#)  
*Whether the property is writable.*
- [unsigned int min](#)  
*Minimum value.*
- [unsigned int max](#)  
*Maximum value.*
- [unsigned int value](#)  
*Current value.*

### 8.24.1 Detailed Description

A GigE property.

### 8.24.2 Member Data Documentation

#### 8.24.2.1 bool isReadable

Whether the property is readable.

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

#### 8.24.2.2 bool isWritable

Whether the property is writable.

#### 8.24.2.3 unsigned int max

Maximum value.

#### 8.24.2.4 unsigned int min

Minimum value.



**8.24.2.5 GigEPropertyType propType**

The type of property.

**8.24.2.6 unsigned int value**

Current value.

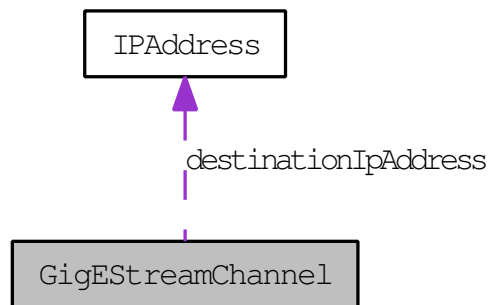
The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.25 GigEStreamChannel Struct Reference

Information about a single GigE stream channel.

Collaboration diagram for GigEStreamChannel:



### Public Member Functions

- [GigEStreamChannel](#) ()

### Public Attributes

- unsigned int [networkInterfaceIndex](#)  
*Network interface index used (or to use).*
- unsigned int [hostPost](#)  
*Host port on the PC where the camera will send the data stream.*
- bool [doNotFragment](#)  
*Disable IP fragmentation of packets.*
- unsigned int [packetSize](#)  
*Packet size, in bytes.*
- unsigned int [interPacketDelay](#)  
*Inter packet delay, in timestamp counter units.*
- [IPAddress](#) [destinationIpAddress](#)  
*Destination IP address.*
- unsigned int [sourcePort](#)  
*Source UDP port of the stream channel.*

### 8.25.1 Detailed Description

Information about a single GigE stream channel.

## 8.25.2 Constructor & Destructor Documentation

### 8.25.2.1 GigEStreamChannel () [inline]

## 8.25.3 Member Data Documentation

### 8.25.3.1 IPAddress destinationIpAddress

Destination IP address.

It can be a multicast or unicast address.

### 8.25.3.2 bool doNotFragment

Disable IP fragmentation of packets.

### 8.25.3.3 unsigned int hostPort

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

### 8.25.3.4 unsigned int interPacketDelay

Inter packet delay, in timestamp counter units.

### 8.25.3.5 unsigned int networkInterfaceIndex

Network interface index used (or to use).

### 8.25.3.6 unsigned int packetSize

Packet size, in bytes.

### 8.25.3.7 unsigned int sourcePort

Source UDP port of the stream channel.

Read only.

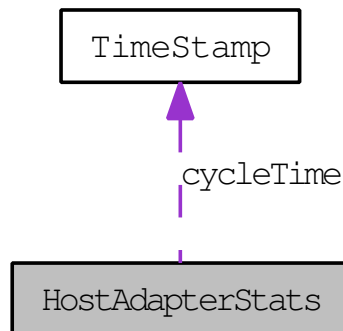
The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.26 HostAdapterStats Struct Reference

Information about the host adapter's statistics.

Collaboration diagram for HostAdapterStats:



### Public Member Functions

- [HostAdapterStats \(\)](#)

### Public Attributes

- char [vendor](#) [[sk\\_maxStringLength](#)]
- unsigned int [numPorts](#)
- unsigned int [portErrors](#) [[sk\\_maxNumPorts](#)]
- unsigned int [fifoOverflows](#)
- unsigned int [busResets](#)
- unsigned int [deviceArrivals](#)
- unsigned int [deviceRemovals](#)
- unsigned int [busErrors](#)
- unsigned int [gapCount](#)
- [TimeStamp](#) [cycleTime](#)

### 8.26.1 Detailed Description

Information about the host adapter's statistics.

## 8.26.2 Constructor & Destructor Documentation

8.26.2.1 `HostAdapterStats()` `[inline]`

## 8.26.3 Member Data Documentation

8.26.3.1 `unsigned int busErrors`

8.26.3.2 `unsigned int busResets`

8.26.3.3 `TimeStamp cycleTime`

8.26.3.4 `unsigned int deviceArrivals`

8.26.3.5 `unsigned int deviceRemovals`

8.26.3.6 `unsigned int fifoOverflows`

8.26.3.7 `unsigned int gapCount`

8.26.3.8 `unsigned int numPorts`

8.26.3.9 `unsigned int portErrors[sk_maxNumPorts]`

8.26.3.10 `char vendor[sk_maxStringLength]`

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.27 Image Class Reference

The [Image](#) class is used to retrieve images from a camera, convert between multiple pixel formats and save images to disk.

### Public Member Functions

- [Image](#) ()  
*Default constructor.*
- [Image](#) (unsigned int rows, unsigned int cols, unsigned int stride, unsigned char \*pData, unsigned int dataSize, [PixelFormat](#) format, [BayerTileFormat](#) bayerFormat=NONE)  
*Construct an [Image](#) object with the specified arguments.*
- [Image](#) (unsigned char \*pData, unsigned int dataSize)  
*Construct an [Image](#) object with the specified arguments.*
- [Image](#) (unsigned int rows, unsigned int cols, [PixelFormat](#) format, [BayerTileFormat](#) bayerFormat=NONE)  
*Construct an [Image](#) object with the specified arguments.*
- [Image](#) (const [Image](#) &image)  
*Copy constructor.*
- virtual [~Image](#) ()  
*Default destructor.*
- virtual [Image](#) & [operator=](#) (const [Image](#) &image)  
*Assignment operator.*
- virtual unsigned char \* [operator\[ \]](#) (unsigned int index)  
*Indexing operator.*
- virtual unsigned char \* [operator\(\)](#) (unsigned int row, unsigned int col)  
*Indexing operator.*
- virtual [Error DeepCopy](#) (const [Image](#) \*pImage)  
*Perform a deep copy of the [Image](#).*
- virtual [Error SetDimensions](#) (unsigned int rows, unsigned int cols, unsigned int stride, [PixelFormat](#) pixelFormat, [BayerTileFormat](#) bayerFormat)  
*Sets the dimensions of the image object.*
- virtual [Error SetData](#) (const unsigned char \*pData, unsigned int dataSize)  
*Set the data of the [Image](#) object.*
- virtual [PixelFormat GetPixelFormat](#) () const  
*Get the current pixel format.*
- virtual [ColorProcessingAlgorithm GetColorProcessing](#) () const

*Get the current color processing algorithm.*

- virtual [Error](#) [SetColorProcessing](#) ([ColorProcessingAlgorithm](#) colorProc)  
*Set the color processing algorithm.*
- virtual unsigned int [GetCols](#) () const  
*Get the number of columns in the image.*
- virtual unsigned int [GetRows](#) () const  
*Get the number of rows in the image.*
- virtual unsigned int [GetStride](#) () const  
*Get the stride in the image.*
- virtual unsigned int [GetBitsPerPixel](#) () const  
*Get the bits per pixel of the image.*
- virtual [BayerTileFormat](#) [GetBayerTileFormat](#) () const  
*Get the Bayer tile format of the image.*
- virtual unsigned int [GetDataSize](#) () const  
*Get the size of the buffer associated with the image, in bytes.*
- virtual void [GetDimensions](#) (unsigned int \*pRows, unsigned int \*pCols=NULL, unsigned int \*pStride=NULL, [PixelFormat](#) \*pPixelFormat=NULL, [BayerTileFormat](#) \*pBayerFormat=NULL) const  
*Get the image dimensions associated with the image.*
- virtual unsigned char \* [GetData](#) ()  
*Get a pointer to the data associated with the image.*
- virtual unsigned char \*const [GetData](#) () const
- virtual [ImageMetadata](#) [GetMetadata](#) () const  
*Get the metadata associated with the image.*
- virtual [Error](#) [CalculateStatistics](#) ([ImageStatistics](#) \*pStatistics)  
*Calculate statistics associated with the image.*
- virtual [TimeStamp](#) [GetTimeStamp](#) () const  
*Get the timestamp data associated with the image.*
- virtual [Error](#) [Save](#) (const char \*pFilename, [ImageFileFormat](#) format=FROM\_FILE\_EXT)  
*Save the image to the specified file name with the file format specified.*
- virtual [Error](#) [Save](#) (const char \*pFilename, [PNGOption](#) \*pOption)  
*Save the image to the specified file name with the options specified.*
- virtual [Error](#) [Save](#) (const char \*pFilename, [PPMOption](#) \*pOption)  
*Save the image to the specified file name with the options specified.*

- virtual [Error Save](#) (const char \*pFilename, [PGMOption](#) \*pOption)  
*Save the image to the specified file name with the options specified.*
- virtual [Error Save](#) (const char \*pFilename, [TIFFOption](#) \*pOption)  
*Save the image to the specified file name with the options specified.*
- virtual [Error Save](#) (const char \*pFilename, [JPEGOption](#) \*pOption)  
*Save the image to the specified file name with the options specified.*
- virtual [Error Save](#) (const char \*pFilename, [JPG2Option](#) \*pOption)  
*Save the image to the specified file name with the options specified.*
- virtual [Error Convert](#) ([PixelFormat](#) format, [Image](#) \*pDestImage) const  
*Converts the current image buffer to the specified output format and stores the result in the specified image.*
- virtual [Error Convert](#) ([Image](#) \*pDestImage) const  
*Converts the current image buffer to the specified output format and stores the result in the specified image.*
- virtual [Error ReleaseBuffer](#) ()  
*Release the buffer associated with the [Image](#).*

## Static Public Member Functions

- static [Error SetDefaultColorProcessing](#) ([ColorProcessingAlgorithm](#) defaultMethod)  
*Set the default color processing algorithm.*
- static [ColorProcessingAlgorithm GetDefaultColorProcessing](#) ()  
*Get the default color processing algorithm.*
- static [Error SetDefaultOutputFormat](#) ([PixelFormat](#) format)  
*Set the default output pixel format.*
- static [PixelFormat GetDefaultOutputFormat](#) ()  
*Get the default output pixel format.*
- static unsigned int [DetermineBitsPerPixel](#) ([PixelFormat](#) format)  
*Calculate the bits per pixel for the specified pixel format.*

## Friends

- class [Iso](#)

### 8.27.1 Detailed Description

The [Image](#) class is used to retrieve images from a camera, convert between multiple pixel formats and save images to disk.

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



## 8.27.2 Constructor & Destructor Documentation

### 8.27.2.1 Image ()

Default constructor.

### 8.27.2.2 Image (unsigned int rows, unsigned int cols, unsigned int stride, unsigned char \* pData, unsigned int dataSize, PixelFormat format, BayerTileFormat bayerFormat = NONE)

Construct an [Image](#) object with the specified arguments.

Ownership of the image buffer is not transferred to the [Image](#) object. It is the user's responsibility to delete the buffer when it is no longer in use.

#### Parameters:

- rows* Rows in the image.
- cols* Columns in the image.
- stride* Stride of the image buffer.
- pData* Pointer to the image buffer.
- dataSize* Size of the image buffer.
- format* Pixel format.
- bayerFormat* Format of the Bayer tiled raw image.

### 8.27.2.3 Image (unsigned char \* pData, unsigned int dataSize)

Construct an [Image](#) object with the specified arguments.

Ownership of the image buffer is not transferred to the [Image](#) object. It is the user's responsibility to delete the buffer when it is no longer in use.

#### Parameters:

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

### 8.27.2.4 Image (unsigned int rows, unsigned int cols, PixelFormat format, BayerTileFormat bayerFormat = NONE)

Construct an [Image](#) object with the specified arguments.

#### Parameters:

- rows* Rows in the image.
- cols* Columns in the image.
- format* Pixel format.
- bayerFormat* Format of the Bayer tiled raw image.

### 8.27.2.5 Image (const Image & *image*)

Copy constructor.

Both images will point to the same image buffer internally.

### 8.27.2.6 virtual ~Image () [virtual]

Default destructor.

The internal image buffer will be released if there are no other [Image](#) objects holding a reference to it. This will also allow the buffer to be requested internally.

## 8.27.3 Member Function Documentation

### 8.27.3.1 virtual Error CalculateStatistics (ImageStatistics \* *pStatistics*) [virtual]

Calculate statistics associated with the image.

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

#### Parameters:

*pStatistics* The [ImageStatistics](#) object to hold the statistics.

#### Returns:

An [Error](#) indicating the success or failure of the function.

### 8.27.3.2 virtual Error Convert (Image \* *pDestImage*) const [virtual]

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

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

#### Parameters:

*pDestImage* Destination image.

#### Returns:

An [Error](#) indicating the success or failure of the function.

### 8.27.3.3 virtual Error Convert (PixelFormat *format*, Image \* *pDestImage*) const [virtual]

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

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

#### Parameters:

*format* Output format of the converted image.

*pDestImage* Destination image.

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.27.3.4 virtual Error DeepCopy (const Image \* pImage) [virtual]**

Perform a deep copy of the [Image](#).

After this operation, the image contents and member variables will be the same. The Images will not share a buffer. The Image's current buffer will not be released.

**Parameters:**

*pImage* The [Image](#) to copy the data from.

**Returns:**

An [Error](#) indicating the success or failure of the function.

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

Calculate the bits per pixel for the specified pixel format.

**Parameters:**

*format* The pixel format.

**Returns:**

The bits per pixel.

**8.27.3.6 virtual BayerTileFormat GetBayerTileFormat () const [virtual]**

Get the Bayer tile format of the image.

**Returns:**

The Bayer tile format.

**8.27.3.7 virtual unsigned int GetBitsPerPixel () const [virtual]**

Get the bits per pixel of the image.

**Returns:**

The bits per pixel.

**8.27.3.8 virtual ColorProcessingAlgorithm GetColorProcessing () const** [virtual]

Get the current color processing algorithm.

See also:

[SetColorProcessing\(\)](#)

Returns:

The current color processing algorithm.

**8.27.3.9 virtual unsigned int GetCols () const** [virtual]

Get the number of columns in the image.

Returns:

The number of columns.

**8.27.3.10 virtual unsigned char\* const GetData () const** [virtual]**8.27.3.11 virtual unsigned char\* GetData ()** [virtual]

Get a pointer to the data associated with the image.

This function is considered unsafe. The pointer returned could be invalidated if the buffer is resized or released. The pointer may also be invalidated if the [Image](#) object is passed to [Camera::RetrieveBuffer\(\)](#). It is recommended that a [Image::DeepCopy\(\)](#) be performed if a separate copy of the [Image](#) data is required for further processing.

Returns:

A pointer to the image data.

**8.27.3.12 virtual unsigned int GetDataSize () const** [virtual]

Get the size of the buffer associated with the image, in bytes.

Returns:

The size of the buffer, in bytes.

**8.27.3.13 static ColorProcessingAlgorithm GetDefaultColorProcessing ()** [static]

Get the default color processing algorithm.

See also:

[SetDefaultColorProcessing\(\)](#)

Returns:

The default color processing algorithm.

**8.27.3.14 static PixelFormat GetDefaultOutputFormat () [static]**

Get the default output pixel format.

See also:

[SetDefaultOutputFormat\(\)](#)

**Returns:**

The default pixel format.

**8.27.3.15 virtual void GetDimensions (unsigned int \* *pRows*, unsigned int \* *pCols* = NULL, unsigned int \* *pStride* = NULL, PixelFormat \* *pPixelFormat* = NULL, BayerTileFormat \* *pBayerFormat* = NULL) const [virtual]**

Get the image dimensions associated with the image.

**Parameters:**

*pRows* Number of rows.

*pCols* Number of columns.

*pStride* The stride.

*pPixelFormat* Pixel format.

*pBayerFormat* Bayer tile format.

**8.27.3.16 virtual ImageMetadata GetMetadata () const [virtual]**

Get the metadata associated with the image.

This includes embedded image information.

**Returns:**

Metadata associated with the image.

**8.27.3.17 virtual PixelFormat GetPixelFormat () const [virtual]**

Get the current pixel format.

**Returns:**

The current pixel format.

**8.27.3.18 virtual unsigned int GetRows () const [virtual]**

Get the number of rows in the image.

**Returns:**

The number of rows.

**8.27.3.19 virtual unsigned int GetStride () const** [virtual]

Get the stride in the image.

**Returns:**

The stride (The number of bytes between rows of the image).

**8.27.3.20 virtual TimeStamp GetTimeStamp () const** [virtual]

Get the timestamp data associated with the image.

**Returns:**

Timestamp data associated with the image.

**8.27.3.21 virtual unsigned char\* operator() (unsigned int *row*, unsigned int *col*)** [virtual]

Indexing operator.

**Parameters:**

*row* The row of the pixel to return.

*col* The column of the pixel to return.

**Returns:**

The address of the specified byte from the image data.

**8.27.3.22 virtual Image& operator= (const Image & *image*)** [virtual]

Assignment operator.

Both images will point to the same image buffer internally. If the [Image](#) already has a buffer attached to it, it will be released.

**Parameters:**

*image* The image to copy from.

**8.27.3.23 virtual unsigned char\* operator[] (unsigned int *index*)** [virtual]

Indexing operator.

**Parameters:**

*index* The index of the byte to return.

**Returns:**

The address of the specified byte from the image data.

**8.27.3.24 virtual Error ReleaseBuffer ()** [virtual]

Release the buffer associated with the [Image](#).

If no buffer is associated, the function does nothing.

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.27.3.25 virtual Error Save (const char \* *pFilename*, JPG2Option \* *pOption*)** [virtual]

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

**Parameters:**

*pFilename* Filename to save image with.

*pOption* Options to use while saving image.

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.27.3.26 virtual Error Save (const char \* *pFilename*, JPEGOption \* *pOption*)** [virtual]

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

**Parameters:**

*pFilename* Filename to save image with.

*pOption* Options to use while saving image.

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.27.3.27 virtual Error Save (const char \* *pFilename*, TIFFOption \* *pOption*)** [virtual]

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

**Parameters:**

*pFilename* Filename to save image with.

*pOption* Options to use while saving image.

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.27.3.28 virtual Error Save (const char \* *pFilename*, PGMOption \* *pOption*)** [virtual]

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

**Parameters:**

*pFilename* Filename to save image with.

*pOption* Options to use while saving image.

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.27.3.29 virtual Error Save (const char \* *pFilename*, PPMOption \* *pOption*)** [virtual]

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

**Parameters:**

*pFilename* Filename to save image with.

*pOption* Options to use while saving image.

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.27.3.30 virtual Error Save (const char \* *pFilename*, PNGOption \* *pOption*)** [virtual]

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

**Parameters:**

*pFilename* Filename to save image with.

*pOption* Options to use while saving image.

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.27.3.31 virtual Error Save (const char \* *pFilename*, ImageFileFormat *format* = FROM\_FILE\_EXT)** [virtual]

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

**Parameters:**

*pFilename* Filename to save image with.

*format* File format to save in.

**Returns:**

An [Error](#) indicating the success or failure of the function.



**8.27.3.32 virtual Error SetColorProcessing (ColorProcessingAlgorithm *colorProc*)** [virtual]

Set the color processing algorithm.

This should be set on the input [Image](#) object.

**Parameters:**

*colorProc* The color processing algorithm to use.

**See also:**

[GetColorProcessing\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.27.3.33 virtual Error SetData (const unsigned char \* *pData*, unsigned int *dataSize*)** [virtual]

Set the data of the [Image](#) object.

Ownership of the image buffer is not transferred to the [Image](#) object. It is the user's responsibility to delete the buffer when it is no longer in use.

**Parameters:**

*pData* Pointer to the image buffer.

*dataSize* Size of the image buffer.

**8.27.3.34 static Error SetDefaultColorProcessing (ColorProcessingAlgorithm *defaultMethod*)** [static]

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* The color processing algorithm to set.

**See also:**

[GetDefaultColorProcessing\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.27.3.35 static Error SetDefaultOutputFormat (PixelFormat *format*)** [static]

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* The output pixel format to set.

**See also:**

[GetDefaultOutputFormat\(\)](#)

**Returns:**

The default color processing algorithm.

**8.27.3.36 virtual Error SetDimensions (unsigned int *rows*, unsigned int *cols*, unsigned int *stride*, PixelFormat *pixelFormat*, BayerTileFormat *bayerFormat*)** [virtual]

Sets the dimensions of the image object.

**Parameters:**

*rows* Number of rows to set.

*cols* Number of cols to set.

*stride* Stride to set.

*pixelFormat* Pixel format to set.

*bayerFormat* Bayer tile format to set.

**See also:**

[GetDimensions\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.27.4 Friends And Related Function Documentation****8.27.4.1 friend class Iso** [friend]

The documentation for this class was generated from the following file:

- [Image.h](#)

## 8.28 ImageMetadata Struct Reference

Metadata related to an image.

### Public Member Functions

- [ImageMetadata](#) ()

### Public Attributes

- unsigned int [embeddedTimeStamp](#)  
*Embedded timestamp.*
- unsigned int [embeddedGain](#)  
*Embedded gain.*
- unsigned int [embeddedShutter](#)  
*Embedded shutter.*
- unsigned int [embeddedBrightness](#)  
*Embedded brightness.*
- unsigned int [embeddedExposure](#)  
*Embedded exposure.*
- unsigned int [embeddedWhiteBalance](#)  
*Embedded white balance.*
- unsigned int [embeddedFrameCounter](#)  
*Embedded frame counter.*
- unsigned int [embeddedStrobePattern](#)  
*Embedded strobe pattern.*
- unsigned int [embeddedGPIOPinState](#)  
*Embedded GPIO pin state.*
- unsigned int [embeddedROIPosition](#)  
*Embedded ROI position.*
- unsigned int [reserved](#) [31]  
*Reserved for future use.*

### 8.28.1 Detailed Description

Metadata related to an image.

## 8.28.2 Constructor & Destructor Documentation

### 8.28.2.1 ImageMetadata () [inline]

## 8.28.3 Member Data Documentation

### 8.28.3.1 unsigned int embeddedBrightness

Embedded brightness.

### 8.28.3.2 unsigned int embeddedExposure

Embedded exposure.

### 8.28.3.3 unsigned int embeddedFrameCounter

Embedded frame counter.

### 8.28.3.4 unsigned int embeddedGain

Embedded gain.

### 8.28.3.5 unsigned int embeddedGPIOPinState

Embedded GPIO pin state.

### 8.28.3.6 unsigned int embeddedROIPosition

Embedded ROI position.

### 8.28.3.7 unsigned int embeddedShutter

Embedded shutter.

### 8.28.3.8 unsigned int embeddedStrobePattern

Embedded strobe pattern.

### 8.28.3.9 unsigned int embeddedTimeStamp

Embedded timestamp.

### 8.28.3.10 unsigned int embeddedWhiteBalance

Embedded white balance.

**8.28.3.11 unsigned int reserved[31]**

Reserved for future use.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.29 ImageStatistics Class Reference

The [ImageStatistics](#) object represents image statistics for an image.

### Public Types

- enum [StatisticsChannel](#) {  
    [GREY](#),  
    [RED](#),  
    [GREEN](#),  
    [BLUE](#),  
    [HUE](#),  
    [SATURATION](#),  
    [LIGHTNESS](#),  
    [NUM\\_STATISTICS\\_CHANNELS](#) }  
    *Channels that allow statistics to be calculated.*

### Public Member Functions

- [ImageStatistics](#) ()  
    *Default constructor.*
- virtual [~ImageStatistics](#) ()  
    *Default destructor.*
- [ImageStatistics](#) (const [ImageStatistics](#) &other)  
    *Copy constructor.*
- [ImageStatistics](#) & operator= (const [ImageStatistics](#) &other)  
    *Assignment operator.*
- [Error EnableAll](#) ()  
    *Enable all channels.*
- [Error DisableAll](#) ()  
    *Disable all channels.*
- [Error EnableGreyOnly](#) ()  
    *Enable only the grey channel.*
- [Error EnableRGBOnly](#) ()  
    *Enable only the RGB channels.*
- [Error EnableHSLOnly](#) ()  
    *Enable only the HSL channels.*
- [Error GetChannelStatus](#) ([StatisticsChannel](#) channel, bool \*pEnabled) const  
    *Get the status of a statistics channel.*

- [Error SetChannelStatus](#) ([StatisticsChannel](#) channel, bool enabled)  
*Set the status of a statistics channel.*
- [Error GetRange](#) ([StatisticsChannel](#) channel, unsigned int \*pMin, unsigned int \*pMax) const  
*Get the range of a statistics channel.*
- [Error GetPixelValueRange](#) ([StatisticsChannel](#) channel, unsigned int \*pPixelValueMin, unsigned int \*pPixelValueMax) const  
*Get the range of a statistics channel.*
- [Error GetNumPixelValues](#) ([StatisticsChannel](#) channel, unsigned int \*pNumPixelValues) const  
*Get the number of unique pixel values in the image.*
- [Error GetMean](#) ([StatisticsChannel](#) channel, float \*pPixelValueMean) const  
*Get the mean of the image.*
- [Error GetHistogram](#) ([StatisticsChannel](#) channel, int \*\*ppHistogram) const  
*Get the histogram for the image.*
- [Error GetStatistics](#) ([StatisticsChannel](#) channel, unsigned int \*pRangeMin=NULL, unsigned int \*pRangeMax=NULL, unsigned int \*pPixelValueMin=NULL, unsigned int \*pPixelValueMax=NULL, unsigned int \*pNumPixelValues=NULL, float \*pPixelValueMean=NULL, int \*\*ppHistogram=NULL) const  
*Get all statistics for the image.*

## Friends

- class [ImageStatsCalculator](#)

### 8.29.1 Detailed Description

The [ImageStatistics](#) object represents image statistics for an image.

### 8.29.2 Member Enumeration Documentation

#### 8.29.2.1 enum StatisticsChannel

Channels that allow statistics to be calculated.

**Enumerator:**

*GREY*  
*RED*  
*GREEN*  
*BLUE*  
*HUE*  
*SATURATION*  
*LIGHTNESS*  
*NUM\_STATISTICS\_CHANNELS*

### 8.29.3 Constructor & Destructor Documentation

#### 8.29.3.1 ImageStatistics ()

Default constructor.

#### 8.29.3.2 virtual ~ImageStatistics () [virtual]

Default destructor.

#### 8.29.3.3 ImageStatistics (const ImageStatistics & *other*)

Copy constructor.

### 8.29.4 Member Function Documentation

#### 8.29.4.1 Error DisableAll ()

Disable all channels.

**Returns:**

An [Error](#) indicating the success or failure of the function.

#### 8.29.4.2 Error EnableAll ()

Enable all channels.

**Returns:**

An [Error](#) indicating the success or failure of the function.

#### 8.29.4.3 Error EnableGreyOnly ()

Enable only the grey channel.

**Returns:**

An [Error](#) indicating the success or failure of the function.

#### 8.29.4.4 Error EnableHSLOnly ()

Enable only the HSL channels.

**Returns:**

An [Error](#) indicating the success or failure of the function.

#### 8.29.4.5 Error EnableRGBOnly ()

Enable only the RGB channels.

**Returns:**

An [Error](#) indicating the success or failure of the function.



**8.29.4.6 Error GetChannelStatus (StatisticsChannel *channel*, bool \* *pEnabled*) const**

Get the status of a statistics channel.

**Parameters:**

*channel* The statistics channel.  
*pEnabled* Whether the channel is enabled.

**See also:**

[SetChannelStatus\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.29.4.7 Error GetHistogram (StatisticsChannel *channel*, int \*\* *ppHistogram*) const**

Get the histogram for the image.

**Parameters:**

*channel* The statistics channel.  
*ppHistogram* Pointer to an array containing the histogram.

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.29.4.8 Error GetMean (StatisticsChannel *channel*, float \* *pPixelValueMean*) const**

Get the mean of the image.

**Parameters:**

*channel* The statistics channel.  
*pPixelValueMean* The mean of the image.

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.29.4.9 Error GetNumPixelValues (StatisticsChannel *channel*, unsigned int \* *pNumPixelValues*) const**

Get the number of unique pixel values in the image.

**Parameters:**

*channel* The statistics channel.  
*pNumPixelValues* The number of unique pixel values.

**Returns:**

An [Error](#) indicating the success or failure of the function.

#### 8.29.4.10 Error GetPixelValueRange (StatisticsChannel *channel*, unsigned int \* *pPixelValueMin*, unsigned int \* *pPixelValueMax*) const

Get the range of a statistics channel.

The values returned are the maximum values recorded for all pixels in the image.

##### Parameters:

*channel* The statistics channel.

*pPixelValueMin* The minimum pixel value.

*pPixelValueMax* The maximum pixel value.

##### Returns:

An [Error](#) indicating the success or failure of the function.

#### 8.29.4.11 Error GetRange (StatisticsChannel *channel*, unsigned int \* *pMin*, unsigned int \* *pMax*) const

Get the range of a statistics channel.

The values returned are the maximum possible values for any given pixel in the image. This is generally 0-255 for 8 bit images, and 0-65535 for 16 bit images.

##### Parameters:

*channel* The statistics channel.

*pMin* The minimum possible value.

*pMax* The maximum possible value.

##### Returns:

An [Error](#) indicating the success or failure of the function.

#### 8.29.4.12 Error GetStatistics (StatisticsChannel *channel*, unsigned int \* *pRangeMin* = NULL, unsigned int \* *pRangeMax* = NULL, unsigned int \* *pPixelValueMin* = NULL, unsigned int \* *pPixelValueMax* = NULL, unsigned int \* *pNumPixelValues* = NULL, float \* *pPixelValueMean* = NULL, int \*\* *ppHistogram* = NULL) const

Get all statistics for the image.

##### Parameters:

*channel* The statistics channel.

*pRangeMin* The minimum possible value.

*pRangeMax* The maximum possible value.

*pPixelValueMin* The minimum pixel value.

*pPixelValueMax* The maximum pixel value.

*pNumPixelValues* The number of unique pixel values.

*pPixelValueMean* The mean of the image.

*ppHistogram* Pointer to an array containing the histogram.

##### Returns:

An [Error](#) indicating the success or failure of the function.

#### 8.29.4.13 ImageStatistics& operator= (const ImageStatistics & *other*)

Assignment operator.

**Parameters:**

*other* The [ImageStatistics](#) object to copy from.

#### 8.29.4.14 Error SetChannelStatus (StatisticsChannel *channel*, bool *enabled*)

Set the status of a statistics channel.

**Parameters:**

*channel* The statistics channel.

*enabled* Whether the channel should be enabled.

**See also:**

[GetChannelStatus\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

### 8.29.5 Friends And Related Function Documentation

#### 8.29.5.1 friend class ImageStatsCalculator [friend]

The documentation for this class was generated from the following file:

- [ImageStatistics.h](#)

## 8.30 IPAddress Struct Reference

IPv4 address.

### Public Member Functions

- [IPAddress](#) ()
- [IPAddress](#) (unsigned int ipAddressVal)
- bool [operator==](#) (const [IPAddress](#) &address)  
*Equality operator.*
- bool [operator!=](#) (const [IPAddress](#) &address)  
*Inequality operator.*

### Public Attributes

- unsigned char [octets](#) [4]

#### 8.30.1 Detailed Description

IPv4 address.

#### 8.30.2 Constructor & Destructor Documentation

**8.30.2.1** [IPAddress](#) () [inline]

**8.30.2.2** [IPAddress](#) (unsigned int *ipAddressVal*) [inline]

#### 8.30.3 Member Function Documentation

**8.30.3.1** bool [operator!=](#) (const [IPAddress](#) & *address*) [inline]

Inequality operator.

**8.30.3.2** bool [operator==](#) (const [IPAddress](#) & *address*) [inline]

Equality operator.

#### 8.30.4 Member Data Documentation

**8.30.4.1** unsigned char [octets](#)[4]

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.31 JPEGOption Struct Reference

Options for saving JPEG image.

### Public Member Functions

- [JPEGOption \(\)](#)

### Public Attributes

- bool [progressive](#)  
*Whether to save as a progressive JPEG file.*
- unsigned int [quality](#)  
*JPEG image quality in range (0-100).*
- unsigned int [reserved](#) [16]  
*Reserved for future use.*

### 8.31.1 Detailed Description

Options for saving JPEG image.

### 8.31.2 Constructor & Destructor Documentation

#### 8.31.2.1 JPEGOption () [inline]

### 8.31.3 Member Data Documentation

#### 8.31.3.1 bool progressive

Whether to save as a progressive JPEG file.

#### 8.31.3.2 unsigned int quality

JPEG image quality in range (0-100).

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

#### 8.31.3.3 unsigned int reserved[16]

Reserved for future use.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.32 JPG2Option Struct Reference

Options for saving JPEG2000 image.

### Public Member Functions

- [JPG2Option \(\)](#)

### Public Attributes

- unsigned int [quality](#)  
*JPEG saving quality in range (1-512).*
- unsigned int [reserved](#) [16]  
*Reserved for future use.*

### 8.32.1 Detailed Description

Options for saving JPEG2000 image.

### 8.32.2 Constructor & Destructor Documentation

#### 8.32.2.1 [JPG2Option \(\)](#) `[inline]`

### 8.32.3 Member Data Documentation

#### 8.32.3.1 unsigned int [quality](#)

JPEG saving quality in range (1-512).

#### 8.32.3.2 unsigned int [reserved](#)[16]

Reserved for future use.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.33 LUTData Struct Reference

Information about the camera's look up table.

### Public Member Functions

- [LUTData \(\)](#)

### Public Attributes

- bool [supported](#)  
*Flag indicating if LUT is supported.*
- bool [enabled](#)  
*Flag indicating if LUT is enabled.*
- unsigned int [numBanks](#)  
*The number of LUT banks available (Always 1 for PGR LUT).*
- unsigned int [numChannels](#)  
*The number of LUT channels per bank available.*
- unsigned int [inputBitDepth](#)  
*The input bit depth of the LUT.*
- unsigned int [outputBitDepth](#)  
*The output bit depth of the LUT.*
- unsigned int [numEntries](#)  
*The number of entries in the LUT.*
- unsigned int [reserved](#) [8]  
*Reserved for future use.*

### 8.33.1 Detailed Description

Information about the camera's look up table.

### 8.33.2 Constructor & Destructor Documentation

#### 8.33.2.1 [LUTData \(\)](#) [inline]

### 8.33.3 Member Data Documentation

#### 8.33.3.1 bool [enabled](#)

Flag indicating if LUT is enabled.

#### 8.33.3.2 unsigned int [inputBitDepth](#)

The input bit depth of the LUT.

**8.33.3.3 unsigned int numBanks**

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

**8.33.3.4 unsigned int numChannels**

The number of LUT channels per bank available.

**8.33.3.5 unsigned int numEntries**

The number of entries in the LUT.

**8.33.3.6 unsigned int outputBitDepth**

The output bit depth of the LUT.

**8.33.3.7 unsigned int reserved[8]**

Reserved for future use.

**8.33.3.8 bool supported**

Flag indicating if LUT is supported.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)



## 8.34 MACAddress Struct Reference

MAC address.

### Public Member Functions

- [MACAddress](#) ()
- [MACAddress](#) (unsigned int macAddressValHigh, unsigned int macAddressValLow)
- bool [operator==](#) (const [MACAddress](#) &address)  
*Equality operator.*
- bool [operator!=](#) (const [MACAddress](#) &address)  
*Inequality operator.*

### Public Attributes

- unsigned char [octets](#) [6]

#### 8.34.1 Detailed Description

MAC address.

#### 8.34.2 Constructor & Destructor Documentation

**8.34.2.1** [MACAddress](#) () `[inline]`

**8.34.2.2** [MACAddress](#) (unsigned int *macAddressValHigh*, unsigned int *macAddressValLow*)  
`[inline]`

#### 8.34.3 Member Function Documentation

**8.34.3.1** bool [operator!=](#) (const [MACAddress](#) & *address*) `[inline]`

Inequality operator.

**8.34.3.2** bool [operator==](#) (const [MACAddress](#) & *address*) `[inline]`

Equality operator.

#### 8.34.4 Member Data Documentation

**8.34.4.1** unsigned char [octets](#)[6]

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.35 PGMOption Struct Reference

Options for saving PGM images.

### Public Member Functions

- [PGMOption \(\)](#)

### Public Attributes

- bool [binaryFile](#)  
*Whether to save the PPM as a binary file.*
- unsigned int [reserved](#) [16]  
*Reserved for future use.*

### 8.35.1 Detailed Description

Options for saving PGM images.

### 8.35.2 Constructor & Destructor Documentation

#### 8.35.2.1 PGMOption () [inline]

### 8.35.3 Member Data Documentation

#### 8.35.3.1 bool binaryFile

Whether to save the PPM as a binary file.

#### 8.35.3.2 unsigned int reserved[16]

Reserved for future use.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.36 PGRGuid Class Reference

A GUID to the camera.

### Public Member Functions

- [PGRGuid \(\)](#)  
*Constructor.*
- bool [operator==](#) (const [PGRGuid](#) &guid)  
*Equality operator.*
- bool [operator!=](#) (const [PGRGuid](#) &guid)  
*Inequality operator.*

### Public Attributes

- unsigned int [value](#) [4]

#### 8.36.1 Detailed Description

A GUID to the camera.

It is used to uniquely identify a camera.

#### 8.36.2 Constructor & Destructor Documentation

##### 8.36.2.1 [PGRGuid \(\)](#) `[inline]`

Constructor.

#### 8.36.3 Member Function Documentation

##### 8.36.3.1 [bool operator!= \(const PGRGuid & guid\)](#) `[inline]`

Inequality operator.

##### 8.36.3.2 [bool operator== \(const PGRGuid & guid\)](#) `[inline]`

Equality operator.

#### 8.36.4 Member Data Documentation

##### 8.36.4.1 [unsigned int value\[4\]](#)

The documentation for this class was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.37 PNGOption Struct Reference

Options for saving PNG images.

### Public Member Functions

- [PNGOption \(\)](#)

### Public Attributes

- bool [interlaced](#)  
*Whether to save the PNG as interlaced.*
- unsigned int [compressionLevel](#)  
*Compression level (0-9).*
- unsigned int [reserved](#) [16]  
*Reserved for future use.*

### 8.37.1 Detailed Description

Options for saving PNG images.

### 8.37.2 Constructor & Destructor Documentation

#### 8.37.2.1 PNGOption () [inline]

### 8.37.3 Member Data Documentation

#### 8.37.3.1 unsigned int compressionLevel

Compression level (0-9).

0 is no compression, 9 is best compression.

#### 8.37.3.2 bool interlaced

Whether to save the PNG as interlaced.

#### 8.37.3.3 unsigned int reserved[16]

Reserved for future use.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.38 PPMOption Struct Reference

Options for saving PPM images.

### Public Member Functions

- [PPMOption \(\)](#)

### Public Attributes

- bool [binaryFile](#)  
*Whether to save the PPM as a binary file.*
- unsigned int [reserved](#) [16]  
*Reserved for future use.*

### 8.38.1 Detailed Description

Options for saving PPM images.

### 8.38.2 Constructor & Destructor Documentation

#### 8.38.2.1 PPMOption () [inline]

### 8.38.3 Member Data Documentation

#### 8.38.3.1 bool binaryFile

Whether to save the PPM as a binary file.

#### 8.38.3.2 unsigned int reserved[16]

Reserved for future use.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.39 Property Struct Reference

A specific camera property.

### Public Member Functions

- [Property](#) ()
- [Property](#) ([PropertyType](#) propType)

### Public Attributes

- [PropertyType](#) type  
*Property info type.*
- bool [present](#)  
*Flag indicating if the property is present.*
- bool [absControl](#)  
*Flag controlling absolute mode.*
- bool [onePush](#)  
*Flag controlling one push.*
- bool [onOff](#)  
*Flag controlling on/off.*
- bool [autoManualMode](#)  
*Flag controlling auto.*
- unsigned int [valueA](#)  
*Value A (integer).*
- unsigned int [valueB](#)  
*Value B (integer).*
- float [absValue](#)  
*Floating point value.*
- unsigned int [reserved](#) [8]  
*Reserved for future use.*

### 8.39.1 Detailed Description

A specific camera property.

## 8.39.2 Constructor & Destructor Documentation

### 8.39.2.1 Property () [inline]

### 8.39.2.2 Property (PropertyType *propType*) [inline]

## 8.39.3 Member Data Documentation

### 8.39.3.1 bool absControl

Flag controlling absolute mode.

### 8.39.3.2 float absValue

Floating point value.

### 8.39.3.3 bool autoManualMode

Flag controlling auto.

### 8.39.3.4 bool onePush

Flag controlling one push.

### 8.39.3.5 bool onOff

Flag controlling on/off.

### 8.39.3.6 bool present

Flag indicating if the property is present.

### 8.39.3.7 unsigned int reserved[8]

Reserved for future use.

### 8.39.3.8 PropertyType type

[Property](#) info type.

### 8.39.3.9 unsigned int valueA

Value A (integer).

### 8.39.3.10 unsigned int valueB

Value B (integer).

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

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.40 PropertyInfo Struct Reference

Information about a specific camera property.

### Public Member Functions

- [PropertyInfo](#) ()
- [PropertyInfo](#) ([PropertyType](#) propType)

### Public Attributes

- [PropertyType](#) type  
*Property info type.*
- bool [present](#)  
*Flag indicating if the property is present.*
- bool [autoSupported](#)  
*Flag indicating if auto is supported.*
- bool [manualSupported](#)  
*Flag indicating if manual is supported.*
- bool [onOffSupported](#)  
*Flag indicating if on/off is supported.*
- bool [onePushSupported](#)  
*Flag indicating if one push is supported.*
- bool [absValSupported](#)  
*Flag indicating if absolute mode is supported.*
- bool [readOutSupported](#)  
*Flag indicating if property value can be read out.*
- unsigned int [min](#)  
*Minimum value (as an integer).*
- unsigned int [max](#)  
*Maximum value (as an integer).*
- float [absMin](#)  
*Minimum value (as a floating point value).*
- float [absMax](#)  
*Maximum value (as a floating point value).*
- char [pUnits](#) [[sk\\_maxStringLength](#)]  
*Textual description of units.*
- char [pUnitAbbr](#) [[sk\\_maxStringLength](#)]  
*Abbreviated textual description of units.*
- unsigned int [reserved](#) [8]  
*Reserved for future use.*



### 8.40.1 Detailed Description

Information about a specific camera property.

This structure is also used as the TriggerDelayInfo structure.

### 8.40.2 Constructor & Destructor Documentation

**8.40.2.1** `PropertyInfo ()` [inline]

**8.40.2.2** `PropertyInfo (PropertyType propType)` [inline]

### 8.40.3 Member Data Documentation

**8.40.3.1** `float absMax`

Maximum value (as a floating point value).

**8.40.3.2** `float absMin`

Minimum value (as a floating point value).

**8.40.3.3** `bool absValSupported`

Flag indicating if absolute mode is supported.

**8.40.3.4** `bool autoSupported`

Flag indicating if auto is supported.

**8.40.3.5** `bool manualSupported`

Flag indicating if manual is supported.

**8.40.3.6** `unsigned int max`

Maximum value (as an integer).

**8.40.3.7** `unsigned int min`

Minimum value (as an integer).

**8.40.3.8** `bool onePushSupported`

Flag indicating if one push is supported.

**8.40.3.9** `bool onOffSupported`

Flag indicating if on/off is supported.

**8.40.3.10 bool present**

Flag indicating if the property is present.

**8.40.3.11 char pUnitAbbr[sk\_maxStringLength]**

Abbreviated textual description of units.

**8.40.3.12 char pUnits[sk\_maxStringLength]**

Textual description of units.

**8.40.3.13 bool readOutSupported**

Flag indicating if property value can be read out.

**8.40.3.14 unsigned int reserved[8]**

Reserved for future use.

**8.40.3.15 PropertyType type**

[Property](#) info type.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.41 StrobeControl Struct Reference

A camera strobe.

### Public Member Functions

- [StrobeControl \(\)](#)

### Public Attributes

- unsigned int [source](#)  
*Source value.*
- bool [onOff](#)  
*Flag controlling on/off.*
- unsigned int [polarity](#)  
*Signal polarity.*
- float [delay](#)  
*Signal delay (in ms).*
- float [duration](#)  
*Signal duration (in ms).*
- unsigned int [reserved](#) [8]  
*Reserved for future use.*

### 8.41.1 Detailed Description

A camera strobe.

### 8.41.2 Constructor & Destructor Documentation

#### 8.41.2.1 [StrobeControl \(\)](#) [inline]

### 8.41.3 Member Data Documentation

#### 8.41.3.1 float [delay](#)

Signal delay (in ms).

#### 8.41.3.2 float [duration](#)

Signal duration (in ms).

#### 8.41.3.3 bool [onOff](#)

Flag controlling on/off.

**8.41.3.4 unsigned int polarity**

Signal polarity.

**8.41.3.5 unsigned int reserved[8]**

Reserved for future use.

**8.41.3.6 unsigned int source**

Source value.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.42 StrobeInfo Struct Reference

A camera strobe property.

### Public Member Functions

- [StrobeInfo](#) ()

### Public Attributes

- unsigned int [source](#)  
*Source value.*
- bool [present](#)  
*Presence of strobe.*
- bool [readOutSupported](#)  
*Flag indicating if strobe value can be read out.*
- bool [onOffSupported](#)  
*Flag indicating if on/off is supported.*
- bool [polaritySupported](#)  
*Flag indicating if polarity is supported.*
- float [minValue](#)  
*Minimum value.*
- float [maxValue](#)  
*Maximum value.*
- unsigned int [reserved](#) [8]  
*Reserved for future use.*

### 8.42.1 Detailed Description

A camera strobe property.

### 8.42.2 Constructor & Destructor Documentation

#### 8.42.2.1 [StrobeInfo](#) () [inline]

### 8.42.3 Member Data Documentation

#### 8.42.3.1 float [maxValue](#)

Maximum value.

#### 8.42.3.2 float [minValue](#)

Minimum value.

**8.42.3.3 bool onOffSupported**

Flag indicating if on/off is supported.

**8.42.3.4 bool polaritySupported**

Flag indicating if polarity is supported.

**8.42.3.5 bool present**

Presence of strobe.

**8.42.3.6 bool readOutSupported**

Flag indicating if strobe value can be read out.

**8.42.3.7 unsigned int reserved[8]**

Reserved for future use.

**8.42.3.8 unsigned int source**

Source value.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.43 SystemInfo Struct Reference

Description of the system.

### Public Attributes

- [OSType](#) `osType`  
*Operating system type as described by OSType.*
- `char` [osDescription](#) [`sk_maxStringLength`]  
*Detailed description of the operating system.*
- [ByteOrder](#) `byteOrder`  
*Byte order of the system.*
- `size_t` [sysMemSize](#)  
*Amount of memory available on the system.*
- `char` [cpuDescription](#) [`sk_maxStringLength`]  
*Detailed description of the CPU.*
- `size_t` [numCpuCores](#)  
*Number of cores on all CPUs on the system.*
- `char` [driverList](#) [`sk_maxStringLength`]  
*List of drivers used.*
- `char` [libraryList](#) [`sk_maxStringLength`]  
*List of libraries used.*
- `char` [gpuDescription](#) [`sk_maxStringLength`]  
*Detailed description of the GPU.*
- `size_t` [screenWidth](#)  
*Screen resolution width in pixels.*
- `size_t` [screenHeight](#)  
*Screen resolution height in pixels.*
- `unsigned int` [reserved](#) [16]  
*Reserved for future use.*

### 8.43.1 Detailed Description

Description of the system.

### 8.43.2 Member Data Documentation

#### 8.43.2.1 ByteOrder `byteOrder`

Byte order of the system.

**8.43.2.2 char cpuDescription[sk\_maxStringLength]**

Detailed description of the CPU.

**8.43.2.3 char driverList[sk\_maxStringLength]**

List of drivers used.

**8.43.2.4 char gpuDescription[sk\_maxStringLength]**

Detailed description of the GPU.

**8.43.2.5 char libraryList[sk\_maxStringLength]**

List of libraries used.

**8.43.2.6 size\_t numCpuCores**

Number of cores on all CPUs on the system.

**8.43.2.7 char osDescription[sk\_maxStringLength]**

Detailed description of the operating system.

**8.43.2.8 OSType osType**

Operating system type as described by OSType.

**8.43.2.9 unsigned int reserved[16]**

Reserved for future use.

**8.43.2.10 size\_t screenHeight**

Screen resolution height in pixels.

**8.43.2.11 size\_t screenWidth**

Screen resolution width in pixels.

**8.43.2.12 size\_t sysMemSize**

Amount of memory available on the system.

The documentation for this struct was generated from the following file:

- [Utilities.h](#)



## 8.44 TIFFOption Struct Reference

Options for saving TIFF images.

### Public Types

```
– enum CompressionMethod {
    NONE = 1,
    PACKBITS,
    DEFLATE,
    ADOBE_DEFLATE,
    CCITTFAX3,
    CCITTFAX4,
    LZW,
    JPEG }
```

### Public Member Functions

```
* TIFFOption ()
```

### Public Attributes

```
* CompressionMethod compression
    Compression method to use for encoding TIFF images.

* unsigned int reserved [16]
    Reserved for future use.
```

### 8.44.1 Detailed Description

Options for saving TIFF images.

### 8.44.2 Member Enumeration Documentation

#### 8.44.2.1 enum CompressionMethod

Enumerator:

**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.

### 8.44.3 Constructor & Destructor Documentation

#### 8.44.3.1 `TIFFOption()` `[inline]`

### 8.44.4 Member Data Documentation

#### 8.44.4.1 `CompressionMethod` `compression`

Compression method to use for encoding TIFF images.

#### 8.44.4.2 `unsigned int reserved[16]`

Reserved for future use.

The documentation for this struct was generated from the following file:

- \* [FlyCapture2Defs.h](#)

## 8.45 TimeStamp Struct Reference

Timestamp information.

### Public Member Functions

- \* [TimeStamp](#) ()

### Public Attributes

- \* long long [seconds](#)  
*Seconds.*
- \* unsigned int [microSeconds](#)  
*Microseconds.*
- \* unsigned int [cycleSeconds](#)  
*1394 cycle time seconds.*
- \* unsigned int [cycleCount](#)  
*1394 cycle time count.*
- \* unsigned int [cycleOffset](#)  
*1394 cycle time offset.*
- \* unsigned int [reserved](#) [8]  
*Reserved for future use.*

### 8.45.1 Detailed Description

Timestamp information.

### 8.45.2 Constructor & Destructor Documentation

#### 8.45.2.1 [TimeStamp](#) () [inline]

### 8.45.3 Member Data Documentation

#### 8.45.3.1 unsigned int [cycleCount](#)

1394 cycle time count.

#### 8.45.3.2 unsigned int [cycleOffset](#)

1394 cycle time offset.

#### 8.45.3.3 unsigned int [cycleSeconds](#)

1394 cycle time seconds.

#### 8.45.3.4 unsigned int [microSeconds](#)

Microseconds.

#### 8.45.3.5 unsigned int reserved[8]

Reserved for future use.

#### 8.45.3.6 long long seconds

Seconds.

The documentation for this struct was generated from the following file:

- \* [FlyCapture2Defs.h](#)

## 8.46 TopologyNode Class Reference

The [TopologyNode](#) class contains topology information that can be used to generate a tree structure of all cameras and devices connected to a computer.

### Public Types

```
* enum PortType {
    NOT\_CONNECTED = 1,
    CONNECTED\_TO\_PARENT,
    CONNECTED\_TO\_CHILD }
    Possible states of a port on a node.
```

```
* enum NodeType {
    COMPUTER,
    BUS,
    CAMERA,
    NODE }
    Type of node.
```

### Public Member Functions

- [TopologyNode](#) ()  
*Default constructor.*
- [TopologyNode](#) ([PGRGuid](#) guid, int deviceId, [NodeType](#) nodeType, [InterfaceType](#) interfaceType)  
*Constructor.*
- virtual [~TopologyNode](#) ()  
*Default destructor.*
- [TopologyNode](#) (const [TopologyNode](#) &other)  
*Copy constructor.*
- virtual [TopologyNode](#) & operator= (const [TopologyNode](#) &other)  
*Assignment operator.*
- virtual [PGRGuid](#) [GetGuid](#) ()  
*Get the [PGRGuid](#) associated with the node.*
- virtual int [GetDeviceId](#) ()  
*Get the device ID associated with the node.*
- virtual [NodeType](#) [GetNodeType](#) ()  
*Get the node type associated with the node.*
- virtual [InterfaceType](#) [GetInterfaceType](#) ()  
*Get the interface type associated with the node.*
- virtual unsigned int [GetNumChildren](#) ()  
*Get the number of child nodes.*
- virtual [TopologyNode](#) [GetChild](#) (unsigned int position)  
*Get child node located at the specified position.*
- virtual void [AddChild](#) ([TopologyNode](#) childNode)  
*Add the specified [TopologyNode](#) as a child of the node.*

- virtual unsigned int [GetNumPorts](#) ()  
*Get the number of ports.*
- virtual [PortType](#) [GetPortType](#) (unsigned int position)  
*Get type of port located at the specified position.*
- virtual void [AddPort](#) ([PortType](#) childPort)  
*Add the specified PortType as a port of the node.*
- virtual bool [AssignGuidToNode](#) ([PGRGuid](#) guid, int deviceId)  
*Assign a [PGRGuid](#) and device ID to the node.*
- virtual bool [AssignGuidToNode](#) ([PGRGuid](#) guid, int deviceId, [NodeType](#) nodeType)  
*Assign a [PGRGuid](#), device ID and nodeType to the node.*

### 8.46.1 Detailed Description

The [TopologyNode](#) class contains topology information that can be used to generate a tree structure of all cameras and devices connected to a computer.

### 8.46.2 Member Enumeration Documentation

#### 8.46.2.1 enum NodeType

Type of node.

**Enumerator:**

***COMPUTER***  
***BUS***  
***CAMERA***  
***NODE***

#### 8.46.2.2 enum PortType

Possible states of a port on a node.

**Enumerator:**

***NOT\_CONNECTED***  
***CONNECTED\_TO\_PARENT***  
***CONNECTED\_TO\_CHILD***

### 8.46.3 Constructor & Destructor Documentation

#### 8.46.3.1 TopologyNode ()

Default constructor.

#### 8.46.3.2 TopologyNode ([PGRGuid](#) guid, int deviceId, [NodeType](#) nodeType, [InterfaceType](#) interfaceType)

Constructor.

**Parameters:**

***guid*** The [PGRGuid](#) of the node (if applicable).  
***deviceId*** Device ID of the node.  
***nodeType*** Type of the node.  
***interfaceType*** Interface type of the node.

#### 8.46.3.3 virtual ~TopologyNode () [virtual]

Default destructor.

### 8.46.3.4 TopologyNode (const TopologyNode & other)

Copy constructor.

## 8.46.4 Member Function Documentation

### 8.46.4.1 virtual void AddChild (TopologyNode *childNode*) [virtual]

Add the specified [TopologyNode](#) as a child of the node.

**Parameters:**

*childNode* The [TopologyNode](#) to add.

### 8.46.4.2 virtual void AddPort (PortType *childPort*) [virtual]

Add the specified PortType as a port of the node.

**Parameters:**

*childPort* The port to add.

### 8.46.4.3 virtual bool AssignGuidToNode (PGRGuid *guid*, int *deviceId*, NodeType *nodeType*) [virtual]

Assign a [PGRGuid](#), device ID and nodeType to the node.

**Parameters:**

*guid* [PGRGuid](#) to be assigned.  
*deviceId* Device ID to be assigned.  
*nodeType* NodeType to be assigned

**Returns:**

Whether the data was successfully set to the node.

### 8.46.4.4 virtual bool AssignGuidToNode (PGRGuid *guid*, int *deviceId*) [virtual]

Assign a [PGRGuid](#) and device ID to the node.

**Parameters:**

*guid* [PGRGuid](#) to be assigned.  
*deviceId* Device ID to be assigned.

**Returns:**

Whether the data was successfully set to the node.

### 8.46.4.5 virtual TopologyNode GetChild (unsigned int *position*) [virtual]

Get child node located at the specified position.

**Parameters:**

*position* Position of the node.

**Returns:**

[TopologyNode](#) at the specified position.

### 8.46.4.6 virtual int GetDeviceId () [virtual]

Get the device ID associated with the node.

**Returns:**

Device ID of the node.

**8.46.4.7 virtual PGRGuid GetGuid () [virtual]**

Get the [PGRGuid](#) associated with the node.

**Returns:**

[PGRGuid](#) of the node.

**8.46.4.8 virtual InterfaceType GetInterfaceType () [virtual]**

Get the interface type associated with the node.

**Returns:**

Interface type of the node.

**8.46.4.9 virtual NodeType GetNodeType () [virtual]**

Get the node type associated with the node.

**Returns:**

Node type of the node.

**8.46.4.10 virtual unsigned int GetNumChildren () [virtual]**

Get the number of child nodes.

**Returns:**

Number of child nodes.

**8.46.4.11 virtual unsigned int GetNumPorts () [virtual]**

Get the number of ports.

**Returns:**

Number of ports.

**8.46.4.12 virtual PortType GetPortType (unsigned int *position*) [virtual]**

Get type of port located at the specified position.

**Parameters:**

*position* Position of the port.

**Returns:**

PortType at the specified position.

**8.46.4.13 virtual TopologyNode& operator= (const TopologyNode & *other*)**

[virtual]

Assignment operator.

**Parameters:**

*other* The [TopologyNode](#) to copy from.

The documentation for this class was generated from the following file:

- [TopologyNode.h](#)



## 8.47 TriggerMode Struct Reference

A camera trigger.

### Public Member Functions

- [TriggerMode](#) ()

### Public Attributes

- bool [onOff](#)  
*Flag controlling on/off.*
- unsigned int [polarity](#)  
*Polarity value.*
- unsigned int [source](#)  
*Source value.*
- unsigned int [mode](#)  
*Mode value.*
- unsigned int [parameter](#)  
*Parameter value.*
- unsigned int [reserved](#) [8]  
*Reserved for future use.*

### 8.47.1 Detailed Description

A camera trigger.

### 8.47.2 Constructor & Destructor Documentation

#### 8.47.2.1 [TriggerMode](#) () [inline]

### 8.47.3 Member Data Documentation

#### 8.47.3.1 unsigned int mode

Mode value.

#### 8.47.3.2 bool onOff

Flag controlling on/off.

#### 8.47.3.3 unsigned int parameter

Parameter value.

#### 8.47.3.4 unsigned int polarity

Polarity value.

#### 8.47.3.5 unsigned int reserved[8]

Reserved for future use.

#### 8.47.3.6 unsigned int source

Source value.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.48 TriggerModeInfo Struct Reference

Information about a camera trigger property.

### Public Member Functions

- [TriggerModeInfo \(\)](#)

### Public Attributes

- bool [present](#)  
*Presence of trigger mode.*
- bool [readOutSupported](#)  
*Flag indicating if trigger value can be read out.*
- bool [onOffSupported](#)  
*Flag indicating if on/off is supported.*
- bool [polaritySupported](#)  
*Flag indicating if polarity is supported.*
- bool [valueReadable](#)  
*Flag indicating if the value is readable.*
- unsigned int [sourceMask](#)  
*Source mask.*
- bool [softwareTriggerSupported](#)  
*Flag indicating if software trigger is supported.*
- unsigned int [modeMask](#)  
*Mode mask.*
- unsigned int [reserved](#) [8]  
*Reserved for future use.*

### 8.48.1 Detailed Description

Information about a camera trigger property.

### 8.48.2 Constructor & Destructor Documentation

#### 8.48.2.1 [TriggerModeInfo \(\)](#) [inline]

### 8.48.3 Member Data Documentation

#### 8.48.3.1 unsigned int [modeMask](#)

Mode mask.

#### 8.48.3.2 bool [onOffSupported](#)

Flag indicating if on/off is supported.

#### 8.48.3.3 bool [polaritySupported](#)

Flag indicating if polarity is supported.

**8.48.3.4 bool present**

Presence of trigger mode.

**8.48.3.5 bool readOutSupported**

Flag indicating if trigger value can be read out.

**8.48.3.6 unsigned int reserved[8]**

Reserved for future use.

**8.48.3.7 bool softwareTriggerSupported**

Flag indicating if software trigger is supported.

**8.48.3.8 unsigned int sourceMask**

Source mask.

**8.48.3.9 bool valueReadable**

Flag indicating if the value is readable.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 8.49 Utilities Class Reference

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

### Static Public Member Functions

- static [Error GetSystemInfo](#) ([SystemInfo](#) \*pSystemInfo)  
*Get system information.*
- static [Error GetLibraryVersion](#) ([FC2Version](#) \*pVersion)  
*Get library version.*
- static [Error LaunchBrowser](#) (const char \*pAddress)  
*Launch a URL in the system default browser.*
- static [Error LaunchHelp](#) (const char \*pFileName)  
*Open a CHM file in the system default CHM viewer.*
- static [Error LaunchCommand](#) (const char \*pCommand)  
*Execute a command in the terminal.*
- static [Error LaunchCommandAsync](#) (const char \*pCommand, [AsyncCommandCallback](#) pCallback, void \*pUserData)  
*Execute a command in the terminal.*

### 8.49.1 Detailed Description

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

It can also be used to launch browsers, CHM viewers or terminal commands.

### 8.49.2 Member Function Documentation

#### 8.49.2.1 static Error GetLibraryVersion ([FC2Version](#) \*pVersion) [static]

Get library version.

**Parameters:**

**pVersion** Structure to receive the library version.

**Returns:**

An [Error](#) indicating the success or failure of the function.

#### 8.49.2.2 static Error GetSystemInfo ([SystemInfo](#) \*pSystemInfo) [static]

Get system information.

**Parameters:**

**pSystemInfo** Structure to receive system information.

**Returns:**

An [Error](#) indicating the success or failure of the function.

#### 8.49.2.3 static Error LaunchBrowser (const char \*pAddress) [static]

Launch a URL in the system default browser.

**Parameters:**

*pAddress* URL to open in browser.

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.49.2.4 static Error LaunchCommand (const char \* *pCommand*) [static]**

Execute a command in the terminal.

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

**Parameters:**

*pCommand* Command to execute.

**See also:**

[LaunchCommandAsync\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.49.2.5 static Error LaunchCommandAsync (const char \* *pCommand*, AsyncCommandCallback *pCallback*, void \* *pUserData*) [static]**

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:**

*pCommand* Command to execute.

*pCallback* Callback to fire when command is complete.

*pUserData* Data pointer to pass to callback.

**See also:**

[LaunchCommand\(\)](#)

**Returns:**

An [Error](#) indicating the success or failure of the function.

**8.49.2.6 static Error LaunchHelp (const char \* *pFileName*) [static]**

Open a CHM file in the system default CHM viewer.

**Parameters:**

*pFileName* Filename of CHM file to open.

**Returns:**

An [Error](#) indicating the success or failure of the function.

The documentation for this class was generated from the following file:

· [Utilities.h](#)

## 8.50 VideoModes Struct Reference

Information about a single DCAM format.

### Public Attributes

- [FlyCapture2::VideoMode](#) videoMode
- [FlyCapture2::FrameRate](#) frameRate
- unsigned int width
- unsigned int height

### 8.50.1 Detailed Description

Information about a single DCAM format.

### 8.50.2 Member Data Documentation

**8.50.2.1** [FlyCapture2::FrameRate](#) frameRate

**8.50.2.2** unsigned int height

**8.50.2.3** [FlyCapture2::VideoMode](#) videoMode

**8.50.2.4** unsigned int width

The documentation for this struct was generated from the following file:

- [PGRDirectShow.h](#)



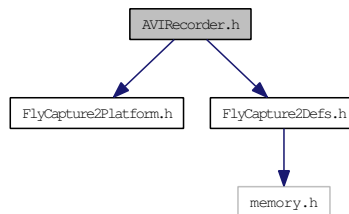


## Chapter 9

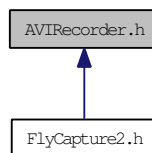
# File Documentation

### 9.1 AVIRecorder.h File Reference

Include dependency graph for AVIRecorder.h:



This graph shows which files directly or indirectly include this file:



#### Classes

- class [AVIRecorder](#)

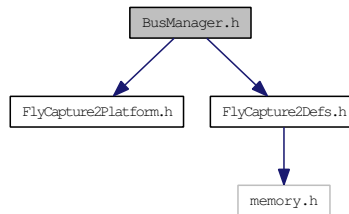
*The [AVIRecorder](#) class provides the functionality for the user to record images to an AVI file.*

#### Namespaces

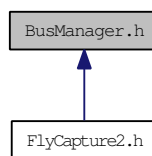
- namespace [FlyCapture2](#)

## 9.2 BusManager.h File Reference

Include dependency graph for BusManager.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class [BusManager](#)  
*The [BusManager](#) class provides the functionality for the user to get an [PGRGuid](#) for a desired camera or device easily.*

### Namespaces

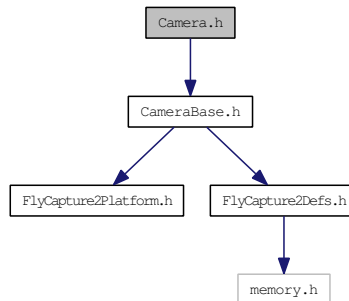
- namespace [FlyCapture2](#)

### Typedefs

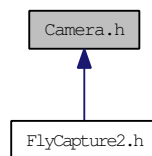
- typedef void(\* [BusEventCallback](#))(void \*pParameter, unsigned int serialNumber)  
*Bus event callback function prototype.*
- typedef void \* [CallbackHandle](#)  
*Handle that is returned when registering a callback.*

## 9.3 Camera.h File Reference

Include dependency graph for Camera.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class [Camera](#)

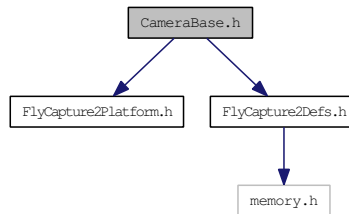
*The [Camera](#) object represents a physical camera that uses the IIDC register set.*

### Namespaces

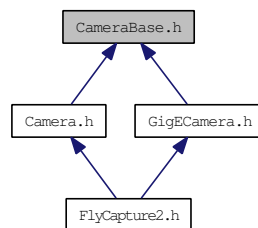
- namespace [FlyCapture2](#)

## 9.4 CameraBase.h File Reference

Include dependency graph for CameraBase.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class [CameraBase](#)  
The [CameraBase](#) class is an abstract base class that defines a general interface to a camera.

### Namespaces

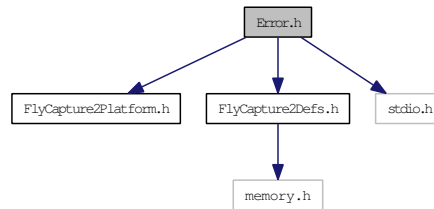
- namespace [FlyCapture2](#)

### Typedefs

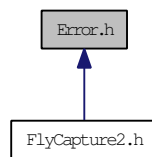
- typedef void(\* [ImageEventCallback](#) )(class Image \*pImage, const void \*pCallbackData)  
[Image](#) event callback function prototype.

## 9.5 Error.h File Reference

Include dependency graph for Error.h:



This graph shows which files directly or indirectly include this file:



### Classes

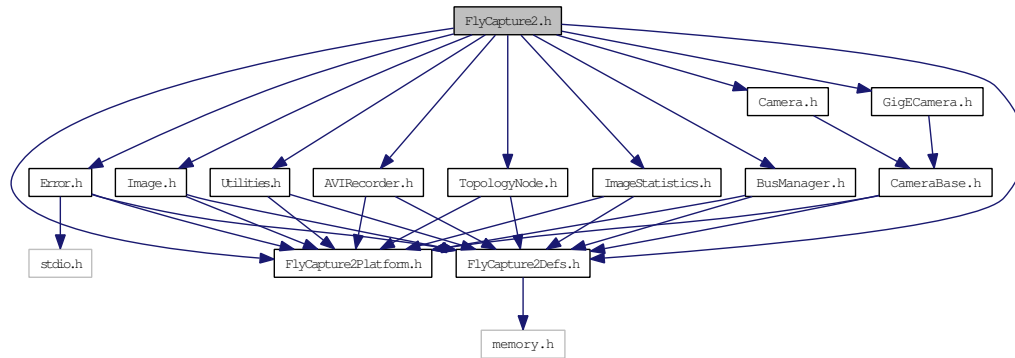
- class [Error](#)  
*The [Error](#) object represents an error that is returned from the library.*

### Namespaces

- namespace [FlyCapture2](#)

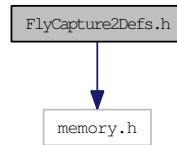
## 9.6 FlyCapture2.h File Reference

Include dependency graph for FlyCapture2.h:

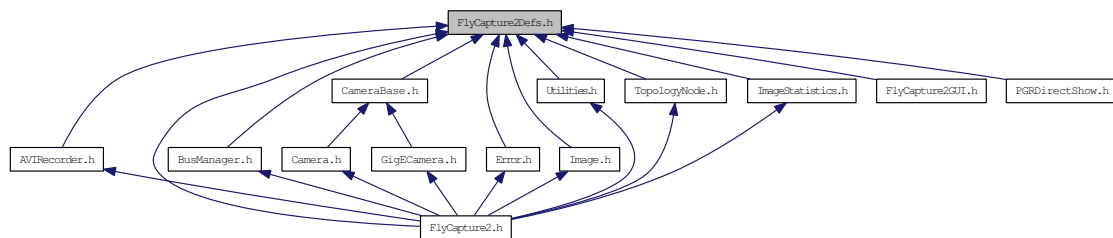


## 9.7 FlyCapture2Defs.h File Reference

Include dependency graph for FlyCapture2Defs.h:



This graph shows which files directly or indirectly include this file:



### Classes

- struct [FC2Version](#)  
*The current version of the library.*
- class [PGRGuid](#)  
*A GUID to the camera.*
- struct [IPAddress](#)  
*IPv4 address.*
- struct [MACAddress](#)  
*MAC address.*
- struct [GigEProperty](#)  
*A GigE property.*
- struct [GigEStreamChannel](#)  
*Information about a single GigE stream channel.*
- struct [GigEConfig](#)  
*Configuration for a GigE camera.*
- struct [GigEImageSettingsInfo](#)  
*Format 7 information for a single mode.*
- struct [GigEImageSettings](#)  
*Image settings for a GigE camera.*
- struct [Format7ImageSettings](#)  
*Format 7 image settings.*
- struct [Format7Info](#)  
*Format 7 information for a single mode.*
- struct [Format7PacketInfo](#)  
*Format 7 packet information.*

- struct [FC2Config](#)  
*Configuration for a camera.*
- struct [PropertyInfo](#)  
*Information about a specific camera property.*
- struct [Property](#)  
*A specific camera property.*
- struct [TriggerModeInfo](#)  
*Information about a camera trigger property.*
- struct [TriggerMode](#)  
*A camera trigger.*
- struct [StrobeInfo](#)  
*A camera strobe property.*
- struct [StrobeControl](#)  
*A camera strobe.*
- struct [TimeStamp](#)  
*Timestamp information.*
- struct [ConfigROM](#)  
*Camera configuration ROM.*
- struct [CameraInfo](#)  
*Camera information.*
- struct [EmbeddedImageInfoProperty](#)  
*Properties of a single embedded image info property.*
- struct [EmbeddedImageInfo](#)  
*Properties of the possible embedded image information.*
- struct [ImageMetadata](#)  
*Metadata related to an image.*
- struct [LUTData](#)  
*Information about the camera's look up table.*
- struct [HostAdapterStats](#)  
*Information about the host adapter's statistics.*
- struct [CameraStats](#)  
*Camera diagnostic information.*
- struct [PNGOption](#)  
*Options for saving PNG images.*
- struct [PPMOption](#)  
*Options for saving PPM images.*
- struct [PGMOption](#)  
*Options for saving PGM images.*
- struct [TIFFOption](#)  
*Options for saving TIFF images.*
- struct [JPEGOption](#)  
*Options for saving JPEG image.*



- struct [JPG2Option](#)  
*Options for saving JPEG2000 image.*
- struct [AVIOption](#)  
*Options for saving AVI files.*

## Namespaces

- namespace [FlyCapture2](#)

## Defines

- #define [NULL](#) 0
- #define [FULL\\_32BIT\\_VALUE](#) 0x7FFFFFFF

## Typedefs

- typedef PropertyInfo [TriggerDelayInfo](#)  
*The TriggerDelayInfo structure is identical to [PropertyInfo](#).*
- typedef Property [TriggerDelay](#)  
*The TriggerDelay structure is identical to [Property](#).*

## Enumerations

- enum [ErrorType](#) {  
    [PGRERROR\\_UNDEFINED](#) = -1,  
    [PGRERROR\\_OK](#),  
    [PGRERROR\\_FAILED](#),  
    [PGRERROR\\_NOT\\_IMPLEMENTED](#),  
    [PGRERROR\\_FAILED\\_BUS\\_MASTER\\_CONNECTION](#),  
    [PGRERROR\\_NOT\\_CONNECTED](#),  
    [PGRERROR\\_INIT\\_FAILED](#),  
    [PGRERROR\\_NOT\\_INITIALIZED](#),  
    [PGRERROR\\_INVALID\\_PARAMETER](#),  
    [PGRERROR\\_INVALID\\_SETTINGS](#),  
    [PGRERROR\\_INVALID\\_BUS\\_MANAGER](#),  
    [PGRERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#),  
    [PGRERROR\\_LOW\\_LEVEL\\_FAILURE](#),  
    [PGRERROR\\_NOT\\_FOUND](#),  
    [PGRERROR\\_FAILED\\_GUID](#),  
    [PGRERROR\\_INVALID\\_PACKET\\_SIZE](#),  
    [PGRERROR\\_INVALID\\_MODE](#),  
    [PGRERROR\\_NOT\\_IN\\_FORMAT7](#),  
    [PGRERROR\\_NOT\\_SUPPORTED](#),  
    [PGRERROR\\_TIMEOUT](#),  
    [PGRERROR\\_BUS\\_MASTER\\_FAILED](#),  
    [PGRERROR\\_INVALID\\_GENERATION](#),  
    [PGRERROR\\_LUT\\_FAILED](#),  
    [PGRERROR\\_IIDC\\_FAILED](#),  
    [PGRERROR\\_STROBE\\_FAILED](#),  
    [PGRERROR\\_TRIGGER\\_FAILED](#),  
    [PGRERROR\\_PROPERTY\\_FAILED](#),  
}

```

PGRERROR_PROPERTY_NOT_PRESENT,
PGRERROR_REGISTER_FAILED,
PGRERROR_READ_REGISTER_FAILED,
PGRERROR_WRITE_REGISTER_FAILED,
PGRERROR_ISOCH_FAILED,
PGRERROR_ISOCH_ALREADY_STARTED,
PGRERROR_ISOCH_NOT_STARTED,
PGRERROR_ISOCH_START_FAILED,
PGRERROR_ISOCH_RETRIEVE_BUFFER_FAILED,
PGRERROR_ISOCH_STOP_FAILED,
PGRERROR_ISOCH_SYNC_FAILED,
PGRERROR_ISOCH_BANDWIDTH_EXCEEDED,
PGRERROR_IMAGE_CONVERSION_FAILED,
PGRERROR_IMAGE_LIBRARY_FAILURE,
PGRERROR_BUFFER_TOO_SMALL,
PGRERROR_IMAGE_CONSISTENCY_ERROR,
PGRERROR_FORCE_32BITS = FULL_32BIT_VALUE }

```

*The error types returned by functions.*

```

· enum BusCallbackType {
  BUS_RESET,
  ARRIVAL,
  REMOVAL,
  CALLBACK_TYPE_FORCE_32BITS = FULL_32BIT_VALUE }

```

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

```

· enum GrabMode {
  DROP_FRAMES,
  BUFFER_FRAMES,
  UNSPECIFIED_GRAB_MODE,
  GRAB_MODE_FORCE_32BITS = FULL_32BIT_VALUE }

```

*The grab strategy employed during image transfer.*

```

· enum GrabTimeout {
  TIMEOUT_NONE = 0,
  TIMEOUT_INFINITE = -1,
  TIMEOUT_UNSPECIFIED = -2,
  GRAB_TIMEOUT_FORCE_32BITS = FULL_32BIT_VALUE }

```

*Timeout options for grabbing images.*

```

· enum BandwidthAllocation {
  BANDWIDTH_ALLOCATION_OFF = 0,
  BANDWIDTH_ALLOCATION_ON = 1,
  BANDWIDTH_ALLOCATION_UNSUPPORTED = 2,
  BANDWIDTH_ALLOCATION_UNSPECIFIED = 3,
  BANDWIDTH_ALLOCATION_FORCE_32BITS = FULL_32BIT_VALUE }

```

*Bandwidth allocation options for 1394 devices.*

```

· enum InterfaceType {
  INTERFACE_IEEE1394,
  INTERFACE_USB2,
  INTERFACE_GIGE,
  INTERFACE_UNKNOWN,
  INTERFACE_TYPE_FORCE_32BITS = FULL_32BIT_VALUE }

```

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

- enum PropertyType {  
    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,  
    PROPERTY\_TYPE\_FORCE\_32BITS = FULL\_32BIT\_VALUE }  
    *Camera properties.*

- enum FrameRate {  
    FRAMERATE\_1\_875,  
    FRAMERATE\_3\_75,  
    FRAMERATE\_7\_5,  
    FRAMERATE\_15,  
    FRAMERATE\_30,  
    FRAMERATE\_60,  
    FRAMERATE\_120,  
    FRAMERATE\_240,  
    FRAMERATE\_FORMAT7,  
    NUM\_FRAMERATES,  
    FRAMERATE\_FORCE\_32BITS = FULL\_32BIT\_VALUE }  
    *Frame rates in frames per second.*

- enum VideoMode {  
    VIDEOMODE\_160x120YUV444,  
    VIDEOMODE\_320x240YUV422,  
    VIDEOMODE\_640x480YUV411,  
    VIDEOMODE\_640x480YUV422,  
    VIDEOMODE\_640x480RGB,  
    VIDEOMODE\_640x480Y8,  
    VIDEOMODE\_640x480Y16,  
    VIDEOMODE\_800x600YUV422,  
    VIDEOMODE\_800x600RGB,  
    VIDEOMODE\_800x600Y8,  
    VIDEOMODE\_800x600Y16,  
    VIDEOMODE\_1024x768YUV422,  
    VIDEOMODE\_1024x768RGB,  
    VIDEOMODE\_1024x768Y8,  
    VIDEOMODE\_1024x768Y16,  
    VIDEOMODE\_1280x960YUV422,

```

VIDEOMODE_1280x960RGB,
VIDEOMODE_1280x960Y8,
VIDEOMODE_1280x960Y16,
VIDEOMODE_1600x1200YUV422,
VIDEOMODE_1600x1200RGB,
VIDEOMODE_1600x1200Y8,
VIDEOMODE_1600x1200Y16,
VIDEOMODE_FORMAT7,
NUM_VIDEOMODES,
VIDEOMODE_FORCE_32BITS = FULL_32BIT_VALUE }

```

*DCAM video modes.*

```

· enum Mode {
  MODE_0 = 0,
  MODE_1,
  MODE_2,
  MODE_3,
  MODE_4,
  MODE_5,
  MODE_6,
  MODE_7,
  MODE_8,
  MODE_9,
  MODE_10,
  MODE_11,
  MODE_12,
  MODE_13,
  MODE_14,
  MODE_15,
  MODE_16,
  MODE_17,
  MODE_18,
  MODE_19,
  MODE_20,
  MODE_21,
  MODE_22,
  MODE_23,
  MODE_24,
  MODE_25,
  MODE_26,
  MODE_27,
  MODE_28,
  MODE_29,
  MODE_30,
  MODE_31,
  NUM_MODES,
  MODE_FORCE_32BITS = FULL_32BIT_VALUE }

```

*Camera modes for DCAM formats as well as Format7.*

```

· enum PixelFormat {
  PIXEL_FORMAT_MONO8 = 0x80000000,
  PIXEL_FORMAT_411YUV8 = 0x40000000,
  PIXEL_FORMAT_422YUV8 = 0x20000000,
  PIXEL_FORMAT_444YUV8 = 0x10000000,

```

```

PIXEL_FORMAT_RGB8 = 0x08000000,
PIXEL_FORMAT_MONO16 = 0x04000000,
PIXEL_FORMAT_RGB16 = 0x02000000,
PIXEL_FORMAT_S_MONO16 = 0x01000000,
PIXEL_FORMAT_S_RGB16 = 0x00800000,
PIXEL_FORMAT_RAW8 = 0x00400000,
PIXEL_FORMAT_RAW16 = 0x00200000,
PIXEL_FORMAT_MONO12 = 0x00100000,
PIXEL_FORMAT_RAW12 = 0x00080000,
PIXEL_FORMAT_BGR = 0x80000008,
PIXEL_FORMAT_BGRU = 0x40000008,
PIXEL_FORMAT_RGB = PIXEL_FORMAT_RGB8,
PIXEL_FORMAT_RGBU = 0x40000002,
NUM_PIXEL_FORMATS = 15,
UNSPECIFIED_PIXEL_FORMAT = 0 }

```

*Pixel formats available for Format7 modes.*

```

· enum BusSpeed {
    BUSSPEED_S100,
    BUSSPEED_S200,
    BUSSPEED_S400,
    BUSSPEED_S480,
    BUSSPEED_S800,
    BUSSPEED_S1600,
    BUSSPEED_S3200,
    BUSSPEED_S5000,
    BUSSPEED_10BASE_T,
    BUSSPEED_100BASE_T,
    BUSSPEED_1000BASE_T,
    BUSSPEED_10000BASE_T,
    BUSSPEED_S_FASTEST,
    BUSSPEED_ANY,
    BUSSPEED_SPEED_UNKNOWN = -1,
    BUSSPEED_FORCE_32BITS = FULL_32BIT_VALUE }

```

*Bus speeds.*

```

· enum DriverType {
    DRIVER_1394_CAM,
    DRIVER_1394_PRO,
    DRIVER_1394_JUJU,
    DRIVER_1394_VIDEO1394,
    DRIVER_1394_RAW1394,
    DRIVER_USB_NONE,
    DRIVER_USB_CAM,
    DRIVER_USB3_PRO,
    DRIVER_GIGE_NONE,
    DRIVER_GIGE_FILTER,
    DRIVER_GIGE_PRO,
    DRIVER_UNKNOWN = -1,
    DRIVER_FORCE_32BITS = FULL_32BIT_VALUE }

```

*Types of low level drivers that flycapture uses.*

```

· enum ColorProcessingAlgorithm {
    DEFAULT,
    NO_COLOR_PROCESSING,

```

```

NEAREST_NEIGHBOR,
EDGE_SENSING,
HQ_LINEAR,
RIGOROUS,
IPP,
COLOR_PROCESSING_ALGORITHM_FORCE_32BITS = FULL_32BIT_VALUE
}

```

*Color processing algorithms.*

```

· enum BayerTileFormat {
    NONE,
    RGGB,
    GRBG,
    GBRG,
    BGGR,
    BT_FORCE_32BITS = FULL_32BIT_VALUE }

```

*Bayer tile formats.*

```

· enum ImageFileFormat {
    FROM_FILE_EXT = -1,
    PGM,
    PPM,
    BMP,
    JPEG,
    JPEG2000,
    TIFF,
    PNG,
    RAW,
    IMAGE_FILE_FORMAT_FORCE_32BITS = FULL_32BIT_VALUE }

```

*File formats to be used for saving images to disk.*

```

· enum GigEPropertyType {
    HEARTBEAT,
    HEARTBEAT_TIMEOUT,
    PACKET_SIZE,
    PACKET_DELAY }

```

*Possible properties that can be queried from the camera.*

## Variables

- static const unsigned int `sk_maxStringLength` = 512  
*The maximum length that is allocated for a string.*
- static const unsigned int `sk_maxNumPorts` = 32  
*The maximum number of ports one device can have.*

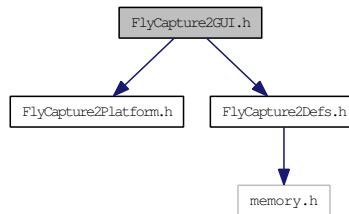
### 9.7.1 Define Documentation

9.7.1.1 `#define FULL_32BIT_VALUE 0x7FFFFFFF`

9.7.1.2 `#define NULL 0`

## 9.8 FlyCapture2GUI.h File Reference

Include dependency graph for FlyCapture2GUI.h:



### Classes

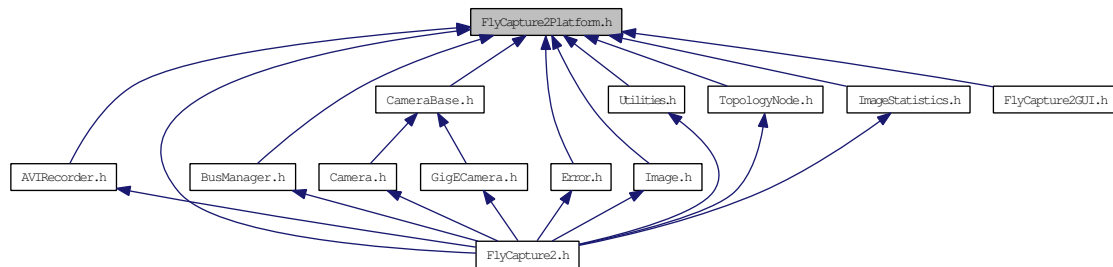
- class [CameraControlDlg](#)  
*The [CameraControlDlg](#) object represents a GTKmm dialog that provides a graphical interface to a specified camera.*
- class [CameraSelectionDlg](#)  
*The [CameraSelectionDlg](#) object represents a GTKmm dialog that provides a graphical interface that lists the number of cameras available to the library.*

### Namespaces

- namespace [FlyCapture2](#)

## 9.9 FlyCapture2Platform.h File Reference

This graph shows which files directly or indirectly include this file:



### Defines

- `#define FLYCAPTURE2_API`

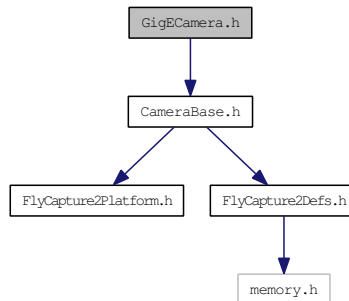
#### 9.9.1 Define Documentation

##### 9.9.1.1 `#define FLYCAPTURE2_API`

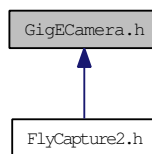


## 9.10 GigECamera.h File Reference

Include dependency graph for GigECamera.h:



This graph shows which files directly or indirectly include this file:



### Classes

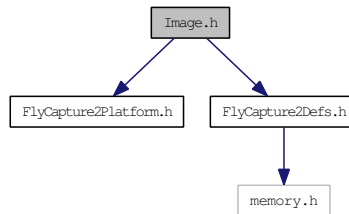
- class [GigECamera](#)  
*The [GigECamera](#) object represents a physical Gigabit Ethernet camera.*

### Namespaces

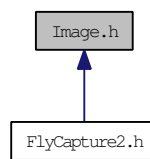
- namespace [FlyCapture2](#)

## 9.11 Image.h File Reference

Include dependency graph for Image.h:



This graph shows which files directly or indirectly include this file:



### Classes

- class [Image](#)

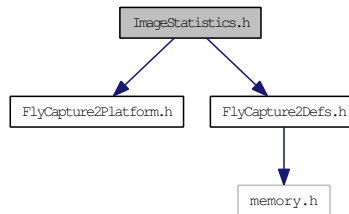
*The [Image](#) class is used to retrieve images from a camera, convert between multiple pixel formats and save images to disk.*

### Namespaces

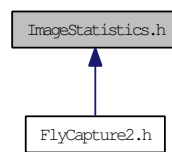
- namespace [FlyCapture2](#)

## 9.12 ImageStatistics.h File Reference

Include dependency graph for ImageStatistics.h:



This graph shows which files directly or indirectly include this file:



### Classes

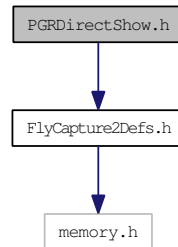
- class [ImageStatistics](#)  
*The [ImageStatistics](#) object represents image statistics for an image.*

### Namespaces

- namespace [FlyCapture2](#)

## 9.13 PGRDirectShow.h File Reference

Include dependency graph for PGRDirectShow.h:



### Classes

- struct [VideoModes](#)  
*Information about a single DCAM format.*
- struct [DCAMFormats](#)  
*DCAM formats supported by the camera as returned by `GetAvailableFormats()`.*

### Functions

- STDMETHOD() [GetImageFormat](#) (unsigned long \*pulFormat)=0  
*Returns the index of the format currently set.*
- STDMETHOD() [SetImageFormat](#) (unsigned long ulFormat)=0  
*Sets the index of the format to be used.*

#### Register functions

*These functions deal with register manipulation on the camera.*

- STDMETHOD() [ReadRegister](#) (unsigned int offset, unsigned int \*pValue)=0  
*Read the specified register from the camera.*
- STDMETHOD() [WriteRegister](#) (unsigned int offset, unsigned int value, bool broadcast=false)=0  
*Write to the specified register on the camera.*

#### Brightness functions

*These functions deal with brightness control on the camera.*

- STDMETHOD() [GetBrightness](#) (long \*pBrightness, bool \*pIsAutoEnabled=NULL)=0
- STDMETHOD() [SetBrightness](#) (long brightness, bool isAutoEnabled=false)=0
- STDMETHOD() [GetBrightnessRange](#) (long \*pMin, long \*pMax, bool \*pIsAutoSupported=NULL)=0
- STDMETHOD() [GetAbsBrightness](#) (float \*pBrightness, bool \*pIsAutoEnabled=NULL)=0
- STDMETHOD() [SetAbsBrightness](#) (float brightness, bool isAutoEnabled=false)=0
- STDMETHOD() [GetAbsBrightnessRange](#) (float \*pMin, float \*pMax, bool \*pIsAutoSupported=NULL, const char \*\*pUnits=NULL)=0

#### Exposure functions

*These functions deal with exposure control on the camera.*

- STDMETHOD() [GetExposure](#) (long \*pExposure, bool \*pIsAutoEnabled=NULL)=0
- STDMETHOD() [SetExposure](#) (long exposure, bool isAutoEnabled=false)=0

- STDMETHOD() [GetExposureRange](#) (long \*pMin, long \*pMax, bool \*pIsAutoSupported=NULL)=0
- STDMETHOD() [GetAbsExposure](#) (float \*pExposure, bool \*pIsAutoEnabled=NULL)=0
- STDMETHOD() [SetAbsExposure](#) (float exposure, bool isAutoEnabled=false)=0
- STDMETHOD() [GetAbsExposureRange](#) (float \*pMin, float \*pMax, bool \*pIsAutoSupported=NULL, const char \*\*pUnits=NULL)=0

#### Shutter functions

*These functions deal with shutter control on the camera.*

- STDMETHOD() [GetShutter](#) (long \*pShutter, bool \*pIsAutoEnabled=NULL)=0
- STDMETHOD() [SetShutter](#) (long shutter, bool isAutoEnabled=false)=0
- STDMETHOD() [GetShutterRange](#) (long \*pMin, long \*pMax, bool \*pIsAutoSupported=NULL)=0
- STDMETHOD() [GetAbsShutter](#) (float \*pShutter, bool \*pIsAutoEnabled=NULL)=0
- STDMETHOD() [SetAbsShutter](#) (float shutter, bool isAutoEnabled=false)=0
- STDMETHOD() [GetAbsShutterRange](#) (float \*pMin, float \*pMax, bool \*pIsAutoSupported=NULL, const char \*\*pUnits=NULL)=0

#### Sharpness functions

*These functions deal with sharpness control on the camera.*

- STDMETHOD() [GetSharpness](#) (long \*pSharpness, bool \*pIsAutoEnabled=NULL)=0
- STDMETHOD() [SetSharpness](#) (long sharpness, bool isAutoEnabled=false)=0
- STDMETHOD() [GetSharpnessRange](#) (long \*pMin, long \*pMax, bool \*pIsAutoSupported=NULL)=0
- STDMETHOD() [GetAbsSharpness](#) (float \*pSharpness, bool \*pIsAutoEnabled=NULL)=0
- STDMETHOD() [SetAbsSharpness](#) (float lSharpness, bool isAutoEnabled=false)=0
- STDMETHOD() [GetAbsSharpnessRange](#) (float \*pMin, float \*pMax, bool \*pIsAutoSupported=NULL, const char \*\*pUnits=NULL)=0

#### Gain functions

*These functions deal with gain control on the camera.*

- STDMETHOD() [GetGain](#) (long \*pGain, bool \*pIsAutoEnabled=NULL)=0
- STDMETHOD() [SetGain](#) (long gain, bool isAutoEnabled=false)=0
- STDMETHOD() [GetGainRange](#) (long \*pMin, long \*pMax, bool \*pIsAutoSupported=NULL)=0
- STDMETHOD() [GetAbsGain](#) (float \*pGain, bool \*pIsAutoEnabled=NULL)=0
- STDMETHOD() [SetAbsGain](#) (float lGain, bool isAutoEnabled=false)=0
- STDMETHOD() [GetAbsGainRange](#) (float \*pMin, float \*pMax, bool \*pIsAutoSupported=NULL, const char \*\*pUnits=NULL)=0

#### Hue functions

*These functions deal with hue control on the camera.*

- STDMETHOD() [GetHue](#) (long \*pHue, bool \*pIsAutoEnabled=NULL)=0
- STDMETHOD() [SetHue](#) (long hue, bool isAutoEnabled=false)=0
- STDMETHOD() [GetHueRange](#) (long \*pMin, long \*pMax, bool \*pIsAutoSupported=NULL)=0
- STDMETHOD() [GetAbsHue](#) (float \*pHue, bool \*pIsAutoEnabled=NULL)=0
- STDMETHOD() [SetAbsHue](#) (float hue, bool isAutoEnabled=false)=0
- STDMETHOD() [GetAbsHueRange](#) (float \*pMin, float \*pMax, bool \*pIsAutoSupported=NULL, const char \*\*pUnits=NULL)=0

#### Saturation functions

*These functions deal with saturation control on the camera.*

- STDMETHOD() [GetSaturation](#) (long \*pSaturation, bool \*pIsAutoEnabled=NULL)=0
- STDMETHOD() [SetSaturation](#) (long saturation, bool isAutoEnabled=false)=0
- STDMETHOD() [GetSaturationRange](#) (long \*pMin, long \*pMax, bool \*pIsAutoSupported=NULL)=0
- STDMETHOD() [GetAbsSaturation](#) (float \*pSaturation, bool \*pIsAutoEnabled=NULL)=0
- STDMETHOD() [SetAbsSaturation](#) (float saturation, bool isAutoEnabled=false)=0
- STDMETHOD() [GetAbsSaturationRange](#) (float \*pMin, float \*pMax, bool \*pIsAutoSupported=NULL, const char \*\*pUnits=NULL)=0

**Gamma functions**

*These functions deal with gamma control on the camera.*

- STDMETHOD() [GetGamma](#) (long \*pGamma, bool \*pIsAutoEnabled=NULL)=0
- STDMETHOD() [SetGamma](#) (long gamma, bool isAutoEnabled=false)=0
- STDMETHOD() [GetGammaRange](#) (long \*pMin, long \*pMax, bool \*pIsAutoSupported=NULL)=0
- STDMETHOD() [GetAbsGamma](#) (float \*pGamma, bool \*pIsAutoEnabled=NULL)=0
- STDMETHOD() [SetAbsGamma](#) (float gamma, bool isAutoEnabled=false)=0
- STDMETHOD() [GetAbsGammaRange](#) (float \*pMin, float \*pMax, bool \*pIsAutoSupported=NULL, const char \*\*pUnits=NULL)=0

**Pan functions**

*These functions deal with pan control on the camera.*

- STDMETHOD() [GetPan](#) (long \*pPan, bool \*pIsAutoEnabled=NULL)=0
- STDMETHOD() [SetPan](#) (long pan, bool isAutoEnabled=false)=0
- STDMETHOD() [GetPanRange](#) (long \*pMin, long \*pMax, bool \*pIsAutoSupported=NULL)=0
- STDMETHOD() [GetAbsPan](#) (float \*pPan, bool \*pIsAutoEnabled=NULL)=0
- STDMETHOD() [SetAbsPan](#) (float pan, bool isAutoEnabled=false)=0
- STDMETHOD() [GetAbsPanRange](#) (float \*pMin, float \*pMax, bool \*pIsAutoSupported=NULL, const char \*\*pUnits=NULL)=0

**Tilt functions**

*These functions deal with tilt control on the camera.*

- STDMETHOD() [GetTilt](#) (long \*pTilt, bool \*pIsAutoEnabled=NULL)=0
- STDMETHOD() [SetTilt](#) (long tilt, bool isAutoEnabled=false)=0
- STDMETHOD() [GetTiltRange](#) (long \*pMin, long \*pMax, bool \*pIsAutoSupported=NULL)=0
- STDMETHOD() [GetAbsTilt](#) (float \*pTilt, bool \*pIsAutoEnabled=NULL)=0
- STDMETHOD() [SetAbsTilt](#) (float tilt, bool isAutoEnabled=false)=0
- STDMETHOD() [GetAbsTiltRange](#) (float \*pMin, float \*pMax, bool \*pIsAutoSupported=NULL, const char \*\*pUnits=NULL)=0

**White balance functions**

*These functions deal with white balance control on the camera.*

- STDMETHOD() [GetWhiteBalance](#) (long \*pIWhiteBalance, bool \*pIsAutoEnabled=NULL)=0
- STDMETHOD() [SetWhiteBalance](#) (long IWhiteBalance, bool isAutoEnabled=false)=0
- STDMETHOD() [GetWhiteBalanceRange](#) (long \*pMin, long \*pMax, bool \*pIsAutoSupported=NULL)=0
- STDMETHOD() [GetAbsWhiteBalance](#) (float \*pIWhiteBalance, bool \*pIsAutoEnabled=NULL)=0
- STDMETHOD() [SetAbsWhiteBalance](#) (float IWhiteBalance, bool isAutoEnabled=false)=0
- STDMETHOD() [GetAbsWhiteBalanceRange](#) (float \*pMin, float \*pMax, bool \*pIsAutoSupported=NULL, const char \*\*pUnits=NULL)=0

**Custom Image / Format7 functions**

*These functions deal with custom image and Format7 image control.*

- STDMETHOD() [GetOutputVerticalFlip](#) (bool \*pIsVerticallyFlipped)=0  
*Get whether the image is vertically flipped.*
- STDMETHOD() [SetOutputVerticalFlip](#) (bool enableVerticalFlip)=0  
*Set the image to be vertically flipped (or not).*
- STDMETHOD() [GetCustomImageMode](#) (bool \*pIsInCustomImageMode)=0  
*Get whether the camera is in custom image / Format7 mode.*
- STDMETHOD() [SetCustomImageMode](#) (bool enableCustomImageMode)=0  
*Set the camera into custom image / Format7 mode (or not).*
- STDMETHOD() [QueryCustomImage](#) (unsigned int mode, bool \*pAvailable,

unsigned int \*pMaxWidth, unsigned int \*pMaxHeight, unsigned int \*pPixelFormatsBitMask)=0

*Get information for a particular custom image / Format7 mode.*

· **STDMETHOD()** [SetCustomImage](#) (unsigned int mode, unsigned int left, unsigned int top, unsigned int width, unsigned int height, unsigned int pixelFormat)=0

*Set custom image / Format7 image settings to camera.*

· **STDMETHOD()** [GetCustomImage](#) (unsigned int \*pMode, unsigned int \*pLeft, unsigned int \*pTop, unsigned int \*pWidth, unsigned int \*pHeight, unsigned int \*pPixelFormat)=0

*Get custom image / Format7 settings from camera.*

## Variables

· **const IID** [IID\\_IFlyCaptureProperties](#) = {0x2bd99656, 0x1552, 0x4d98, {0xb6,0x48,0xd,0xd0,0x19,0x6d,0x16,0x49}}

*This is the Interface that allows Users to get and set Properties on the camera.*





### 9.13.1 Function Documentation

**9.13.1.1** **STDMETHOD()** GetAbsBrightness (float \* *pBrightness*, bool \* *pIsAutoEnabled* = NULL) [pure virtual]

**9.13.1.2** **STDMETHOD()** GetAbsBrightnessRange (float \* *pMin*, float \* *pMax*, bool \* *pIsAutoSupported* = NULL, const char \*\* *pUnits* = NULL) [pure virtual]

**9.13.1.3** **STDMETHOD()** GetAbsExposure (float \* *pExposure*, bool \* *pIsAutoEnabled* = NULL) [pure virtual]

**9.13.1.4** **STDMETHOD()** GetAbsExposureRange (float \* *pMin*, float \* *pMax*, bool \* *pIsAutoSupported* = NULL, const char \*\* *pUnits* = NULL) [pure virtual]

**9.13.1.5** **STDMETHOD()** GetAbsGain (float \* *gain*, bool \* *pIsAutoEnabled* = NULL) [pure virtual]

**9.13.1.6** **STDMETHOD()** GetAbsGainRange (float \* *pMin*, float \* *pMax*, bool \* *pIsAutoSupported* = NULL, const char \*\* *pUnits* = NULL) [pure virtual]

**9.13.1.7** **STDMETHOD()** GetAbsGamma (float \* *pGamma*, bool \* *pIsAutoEnabled* = NULL) [pure virtual]

**9.13.1.8** **STDMETHOD()** GetAbsGammaRange (float \* *pMin*, float \* *pMax*, bool \* *pIsAutoSupported* = NULL, const char \*\* *pUnits* = NULL) [pure virtual]

**9.13.1.9** **STDMETHOD()** GetAbsHue (float \* *pHue*, bool \* *pIsAutoEnabled* = NULL) [pure virtual]

**9.13.1.10** **STDMETHOD()** GetAbsHueRange (float \* *pMin*, float \* *pMax*, bool \* *pIsAutoSupported* = NULL, const char \*\* *pUnits* = NULL) [pure virtual]

**9.13.1.11** **STDMETHOD()** GetAbsPan (float \* *pPan*, bool \* *pIsAutoEnabled* = NULL) [pure virtual]

**9.13.1.12** **STDMETHOD()** GetAbsPanRange (float \* *pMin*, float \* *pMax*, bool \* *pIsAutoSupported* = NULL, const char \*\* *pUnits* = NULL) [pure virtual]

**9.13.1.13** **STDMETHOD()** GetAbsSaturation (float \* *pSaturation*, bool \* *pIsAutoEnabled* = NULL) [pure virtual]

**9.13.1.14** **STDMETHOD()** GetAbsSaturationRange (float \* *pMin*, float \* *pMax*, bool \* *pIsAutoSupported* = NULL, const char \*\* *pUnits* = NULL) [pure virtual]

**9.13.1.15** **STDMETHOD()** GetAbsSharpness (float \* *pSharpness*, bool \* *pIsAutoEnabled* = NULL) [pure virtual]

**9.13.1.16** **STDMETHOD()** GetAbsSharpnessRange (float \* *pMin*, float \* *pMax*, bool \* *pIsAutoSupported* = NULL, const char \*\* *pUnits* = NULL) [pure virtual]

**9.13.1.17** **STDMETHOD()** GetAbsShutter (float \* *pShutter*, bool \* *pIsAutoEnabled* = NULL) [pure virtual]

**9.13.1.18** **STDMETHOD()** GetAbsShutterRange (float \* *pMin*, float \* *pMax*, bool \* *pIsAutoSupported* = NULL, const char \*\* *pUnits* = NULL) [pure virtual]

**9.13.1.19** **STDMETHOD()** GetAbsTilt (float \* *pTilt*, bool \* *pIsAutoEnabled* = NULL) [pure virtual]

**9.13.1.20** **STDMETHOD()** GetAbsTiltRange (float \* *pMin*, float \* *pMax*, bool \* *pIsAutoSupported* = NULL, const char \*\* *pUnits* = NULL) [pure virtual]

**9.13.1.21** **STDMETHOD()** GetAbsWhiteBalance (float \* *pWhiteBalance*, bool \* *pIsAutoEnabled* = NULL) [pure virtual]

**9.13.1.22** **STDMETHOD()** GetAbsWhiteBalanceRange (float \* *pMin*, float \* *pMax*, bool \* *pIsAutoSupported* = NULL, const char \*\* *pUnits* = NULL) [pure virtual]

**Parameters:**

*pMode* Custom image / Format7 settings on camera.  
*pLeft* Left offset.  
*pTop* Top offset.  
*pWidth* Image width.  
*pHeight* Image height.  
*pPixelFormat* Pixel format.

**Returns:**

An HRESULT error code indicating the success or failure of the function.

**9.13.1.26 STDMETHODCALLTYPE GetCustomImageMode (bool \*  
*pIsInCustomImageMode*) [pure virtual]**

Get whether the camera is in custom image / Format7 mode.

**Parameters:**

*pIsInCustomImageMode* Whether the camera is in custom image / Format7 mode.

**Returns:**

An HRESULT error code indicating the success or failure of the function.

**9.13.1.27 STDMETHODCALLTYPE GetExposure (long \**pExposure*, bool \**pIsAutoEnabled*  
 = NULL) [pure virtual]**
**9.13.1.28 STDMETHODCALLTYPE GetExposureRange (long \**pMin*, long \**pMax*, bool \*  
*pIsAutoSupported* = NULL) [pure virtual]**
**9.13.1.29 STDMETHODCALLTYPE GetGain (long \**pGain*, bool \**pIsAutoEnabled* = NULL)  
 [pure virtual]**
**9.13.1.30 STDMETHODCALLTYPE GetGainRange (long \**pMin*, long \**pMax*, bool \*  
*pIsAutoSupported* = NULL) [pure virtual]**
**9.13.1.31 STDMETHODCALLTYPE GetGamma (long \**pGamma*, bool \**pIsAutoEnabled* =  
 NULL) [pure virtual]**
**9.13.1.32 STDMETHODCALLTYPE GetGammaRange (long \**pMin*, long \**pMax*, bool \*  
*pIsAutoSupported* = NULL) [pure virtual]**
**9.13.1.33 STDMETHODCALLTYPE GetHue (long \**pHue*, bool \**pIsAutoEnabled* = NULL)  
 [pure virtual]**
**9.13.1.34 STDMETHODCALLTYPE GetHueRange (long \**pMin*, long \**pMax*, bool \*  
*pIsAutoSupported* = NULL) [pure virtual]**
**9.13.1.35 STDMETHODCALLTYPE GetImageFormat (unsigned long \**pulFormat*) [pure  
 virtual]**

Returns the index of the format currently set.

The index refers to the index of the videoModes array in the [DCAMFormats](#) structure returned by a call to GetAvailableFormats().

**Parameters:**

*pulFormat* Index of currently set format.

**Returns:**

An HRESULT indicating the success or failure of the function.

**9.13.1.36 STDMETHODCALLTYPE GetOutputVerticalFlip (bool \**pIsVerticallyFlipped*)  
 [pure virtual]**

Get whether the image is vertically flipped.

**Parameters:**

*pIsVerticallyFlipped* Whether the image is flipped.

**Returns:**

An HRESULT error code indicating the success or failure of the function.

**9.13.1.37** **STDMETHOD()** **GetPan** (long \* *pPan*, bool \* *pIsAutoEnabled* = NULL)  
[pure virtual]

**9.13.1.38** **STDMETHOD()** **GetPanRange** (long \* *pMin*, long \* *pMax*, bool \* *pIsAutoSupported* = NULL) [pure virtual]

**9.13.1.39** **STDMETHOD()** **GetSaturation** (long \* *pSaturation*, bool \* *pIsAutoEnabled* = NULL) [pure virtual]

**9.13.1.40** **STDMETHOD()** **GetSaturationRange** (long \* *pMin*, long \* *pMax*, bool \* *pIsAutoSupported* = NULL) [pure virtual]

**9.13.1.41** **STDMETHOD()** **GetSharpness** (long \* *pSharpness*, bool \* *pIsAutoEnabled* = NULL) [pure virtual]

**9.13.1.42** **STDMETHOD()** **GetSharpnessRange** (long \* *pMin*, long \* *pMax*, bool \* *pIsAutoSupported* = NULL) [pure virtual]

**9.13.1.43** **STDMETHOD()** **GetShutter** (long \* *pShutter*, bool \* *pIsAutoEnabled* = NULL) [pure virtual]

**9.13.1.44** **STDMETHOD()** **GetShutterRange** (long \* *pMin*, long \* *pMax*, bool \* *pIsAutoSupported* = NULL) [pure virtual]

**9.13.1.45** **STDMETHOD()** **GetTilt** (long \* *pTilt*, bool \* *pIsAutoEnabled* = NULL)  
[pure virtual]

**9.13.1.46** **STDMETHOD()** **GetTiltRange** (long \* *pMin*, long \* *pMax*, bool \* *pIsAutoSupported* = NULL) [pure virtual]

**9.13.1.47** **STDMETHOD()** **GetWhiteBalance** (long \* *pWhiteBalance*, bool \* *pIsAutoEnabled* = NULL) [pure virtual]

**9.13.1.48** **STDMETHOD()** **GetWhiteBalanceRange** (long \* *pMin*, long \* *pMax*, bool \* *pIsAutoSupported* = NULL) [pure virtual]

**9.13.1.49** **STDMETHOD()** **QueryCustomImage** (unsigned int *mode*, bool \* *pAvailable*, unsigned int \* *pMaxWidth*, unsigned int \* *pMaxHeight*, unsigned int \* *pPixelFormatBitMask*) [pure virtual]

Get information for a particular custom image / Format7 mode.

**Parameters:**

*mode* Mode to query.

*pAvailable* Whether mode is supported.

*pMaxWidth* Maximum width.

*pMaxHeight* Maximum height.

*pPixelFormatBitMask* Pixel format bit mask.

**Returns:**

An HRESULT error code indicating the success or failure of the function.

**9.13.1.50** **STDMETHOD()** **ReadRegister** (unsigned int *offset*, unsigned int \* *pValue*) [pure virtual]

Read the specified register from the camera.

**Parameters:**

*offset* DCAM address to be read from.  
*pValue* The value that is read.

**Returns:**

An HRESULT error code indicating the success or failure of the function.

**9.13.1.51** **STDMETHOD()** **SetAbsBrightness** (float *brightness*, bool *isAutoEnabled* = false) [pure virtual]

**9.13.1.52** **STDMETHOD()** **SetAbsExposure** (float *exposure*, bool *isAutoEnabled* = false) [pure virtual]

**9.13.1.53** **STDMETHOD()** **SetAbsGain** (float *lGain*, bool *isAutoEnabled* = false) [pure virtual]

**9.13.1.54** **STDMETHOD()** **SetAbsGamma** (float *gamma*, bool *isAutoEnabled* = false) [pure virtual]

**9.13.1.55** **STDMETHOD()** **SetAbsHue** (float *hue*, bool *isAutoEnabled* = false) [pure virtual]

**9.13.1.56** **STDMETHOD()** **SetAbsPan** (float *pan*, bool *isAutoEnabled* = false) [pure virtual]

**9.13.1.57** **STDMETHOD()** **SetAbsSaturation** (float *saturation*, bool *isAutoEnabled* = false) [pure virtual]

**9.13.1.58** **STDMETHOD()** **SetAbsSharpness** (float *lSharpness*, bool *isAutoEnabled* = false) [pure virtual]

**9.13.1.59** **STDMETHOD()** **SetAbsShutter** (float *shutter*, bool *isAutoEnabled* = false) [pure virtual]

**9.13.1.60** **STDMETHOD()** **SetAbsTilt** (float *tilt*, bool *isAutoEnabled* = false) [pure virtual]

**9.13.1.61** **STDMETHOD()** **SetAbsWhiteBalance** (float *lWhiteBalance*, bool *isAutoEnabled* = false) [pure virtual]

**9.13.1.62** **STDMETHOD()** **SetBrightness** (long *brightness*, bool *isAutoEnabled* = false) [pure virtual]

**9.13.1.63** **STDMETHOD()** **SetCustomImage** (unsigned int *mode*, unsigned int *left*, unsigned int *top*, unsigned int *width*, unsigned int *height*, unsigned int *pixelFormat*) [pure virtual]

Set custom image / Format7 image settings to camera.

**Parameters:**

*mode* Custom image / Format7 settings to use.  
*left* Left offset.  
*top* Top offset.  
*width* Image width.  
*height* Image height.  
*pixelFormat* Pixel format index to use.

**Returns:**

An HRESULT error code indicating the success or failure of the function.

**9.13.1.64 STDMETHOD() SetCustomImageMode (bool *enableCustomImageMode*)**  
[pure virtual]

Set the camera into custom image / Format7 mode (or not).

**Parameters:**

*enableCustomImageMode* Whether to set the camera into custom image / Format7 mode.

**Returns:**

An HRESULT error code indicating the success or failure of the function.

**9.13.1.65 STDMETHOD() SetExposure (long *exposure*, bool *isAutoEnabled* = false)**  
[pure virtual]

**9.13.1.66 STDMETHOD() SetGain (long *gain*, bool *isAutoEnabled* = false)**  
[pure virtual]

**9.13.1.67 STDMETHOD() SetGamma (long *gamma*, bool *isAutoEnabled* = false)**  
[pure virtual]

**9.13.1.68 STDMETHOD() SetHue (long *hue*, bool *isAutoEnabled* = false)**  
[pure virtual]

**9.13.1.69 STDMETHOD() SetImageFormat (unsigned long *ulFormat*)** [pure virtual]

Sets the index of the format to be used.

The index refers to the index of the videoModes array in the [DCAMFormats](#) structure returned by a call to GetAvailableFormats().

**Parameters:**

*ulFormat* index of the format to be set.

**Returns:**

An HRESULT error code indicating the success or failure of the function.

**9.13.1.70 STDMETHOD() SetOutputVerticalFlip (bool *enableVerticalFlip*)**  
[pure virtual]

Set the image to be vertically flipped (or not).

**Parameters:**

*enableVerticalFlip* Whether the image should be vertically flipped.

**Returns:**

An HRESULT error code indicating the success or failure of the function.

**9.13.1.71** **STDMETHOD()** **SetPan** (long *pan*, bool *isAutoEnabled* = false)  
[pure virtual]

**9.13.1.72** **STDMETHOD()** **SetSaturation** (long *saturation*, bool *isAutoEnabled* = false) [pure virtual]

**9.13.1.73** **STDMETHOD()** **SetSharpness** (long *sharpness*, bool *isAutoEnabled* = false) [pure virtual]

**9.13.1.74** **STDMETHOD()** **SetShutter** (long *shutter*, bool *isAutoEnabled* = false)  
[pure virtual]

**9.13.1.75** **STDMETHOD()** **SetTilt** (long *tilt*, bool *isAutoEnabled* = false) [pure virtual]

**9.13.1.76** **STDMETHOD()** **SetWhiteBalance** (long *lWhiteBalance*, bool *isAutoEnabled* = false) [pure virtual]

**9.13.1.77** **STDMETHOD()** **WriteRegister** (unsigned int *offset*, unsigned int *value*, bool *broadcast* = false) [pure virtual]

Write to the specified register on the camera.

**Parameters:**

*offset* DCAM address to be written to.

*value* The value to be written.

*broadcast* Whether the action should be broadcast.

**Returns:**

An HRESULT error code indicating the success or failure of the function.

## 9.13.2 Variable Documentation

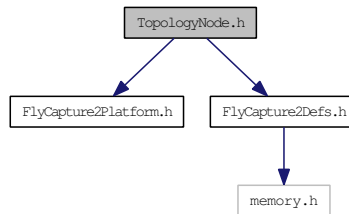
**9.13.2.1** **const IID IID\_IFlyCaptureProperties** = {0x2bd99656, 0x1552, 0x4d98, {0xb6, 0x48, 0xd, 0xd0, 0x19, 0x6d, 0x16, 0x49}}

This is the Interface that allows Users to get and set Properties on the camera.

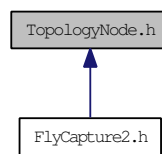
You can query the PGRDirectShowSource Capture Filter for IID\_IFlyCaptureProperties and it will return a pointer to the IFlyCaptureProperties interface. The GUID is equivalent to {2BD99656-1552-4d98-B648-0DD0196D1649}.

## 9.14 TopologyNode.h File Reference

Include dependency graph for TopologyNode.h:



This graph shows which files directly or indirectly include this file:



### Classes

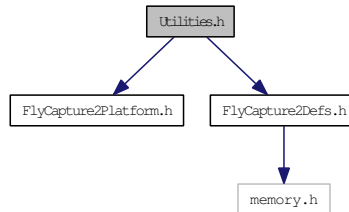
- class [TopologyNode](#)  
*The [TopologyNode](#) class contains topology information that can be used to generate a tree structure of all cameras and devices connected to a computer.*

### Namespaces

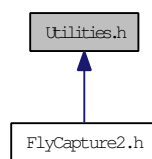
- namespace [FlyCapture2](#)

## 9.15 Utilities.h File Reference

Include dependency graph for Utilities.h:



This graph shows which files directly or indirectly include this file:



### Classes

- struct [SystemInfo](#)  
*Description of the system.*
- class [Utilities](#)  
*The Utility class is generally used to query for general system information such as operating system, available memory etc.*

### Namespaces

- namespace [FlyCapture2](#)

### Typedefs

- typedef void(\* [AsyncCommandCallback](#) )(class Error retError, void \*pUserData)  
*Async command callback function prototype.*

### Enumerations

- enum [OSType](#) {  
[WINDOWS\\_X86](#),  
[WINDOWS\\_X64](#),  
[LINUX\\_X86](#),  
[LINUX\\_X64](#),  
[MAC](#),  
[UNKNOWN\\_OS](#),  
[OSTYPE\\_FORCE\\_32BITS](#) = [FULL\\_32BIT\\_VALUE](#) }  
*Possible operating systems.*
- enum [ByteOrder](#) {  
[BYTE\\_ORDER\\_LITTLE\\_ENDIAN](#),  
[BYTE\\_ORDER\\_BIG\\_ENDIAN](#),



---

```
BYTE_ORDER_FORCE_32BITS = FULL_32BIT_VALUE }
```

*Possible byte orders.*

# Index

- ~AVIRecorder
  - FlyCapture2::AVIRecorder, [51](#)
- ~BusManager
  - FlyCapture2::BusManager, [54](#)
- ~Camera
  - FlyCapture2::Camera, [65](#)
- ~CameraBase
  - FlyCapture2::CameraBase, [90](#)
- ~CameraControlDlg
  - FlyCapture2::CameraControlDlg, [109](#)
- ~CameraSelectionDlg
  - FlyCapture2::CameraSelectionDlg, [116](#)
- ~Error
  - FlyCapture2::Error, [128](#)
- ~GigECamera
  - FlyCapture2::GigECamera, [147](#)
- ~Image
  - FlyCapture2::Image, [186](#)
- ~ImageStatistics
  - FlyCapture2::ImageStatistics, [200](#)
- ~TopologyNode
  - FlyCapture2::TopologyNode, [230](#)
- absControl
  - FlyCapture2::Property, [215](#)
- absMax
  - FlyCapture2::PropertyInfo, [217](#)
- absMin
  - FlyCapture2::PropertyInfo, [217](#)
- absValSupported
  - FlyCapture2::PropertyInfo, [217](#)
- absValue
  - FlyCapture2::Property, [215](#)
- AddChild
  - FlyCapture2::TopologyNode, [231](#)
- AddPort
  - FlyCapture2::TopologyNode, [231](#)
- ADOBE\_DEFLATE
  - FlyCapture2::TIFFOption, [225](#)
- ARRIVAL
  - Enumerations, [19](#)
- AssignGuidToNode
  - FlyCapture2::TopologyNode, [231](#)
- asyncBusSpeed
  - FlyCapture2::FC2Config, [132](#)
- AsyncCommandCallback
  - FlyCapture2, [46](#)
- AUTO\_EXPOSURE
  - Enumerations, [26](#)
- autoManualMode
  - FlyCapture2::Property, [215](#)
- autoSupported
  - FlyCapture2::PropertyInfo, [217](#)
- available
  - FlyCapture2::EmbeddedImageInfoProperty, [126](#)
- AVIAppend
  - FlyCapture2::AVIRecorder, [51](#)
- AVIClose
  - FlyCapture2::AVIRecorder, [51](#)
- AVIOpen
  - FlyCapture2::AVIRecorder, [52](#)
- AVIOption
  - FlyCapture2::AVIOption, [49](#)
- AVIRecorder
  - FlyCapture2::AVIRecorder, [51](#)
- AVIRecorder.h, [241](#)
- BANDWIDTH\_ALLOCATION\_FORCE\_32BITS
  - Enumerations, [19](#)
- BANDWIDTH\_ALLOCATION\_OFF
  - Enumerations, [18](#)
- BANDWIDTH\_ALLOCATION\_ON
  - Enumerations, [18](#)
- BANDWIDTH\_ALLOCATION\_UNSPECIFIED
  - Enumerations, [19](#)
- BANDWIDTH\_ALLOCATION\_UNSUPPORTED
  - Enumerations, [18](#)
- BandwidthAllocation
  - Enumerations, [18](#)
- bandwidthAllocation
  - FlyCapture2::FC2Config, [132](#)
- BayerTileFormat
  - Enumerations, [19](#)
- bayerTileFormat
  - FlyCapture2::CameraInfo, [113](#)
- BGGR
  - Enumerations, [19](#)
- binaryFile
  - FlyCapture2::PGMOption, [210](#)
  - FlyCapture2::PPMOption, [213](#)

- BLUE
  - FlyCapture2::ImageStatistics, [199](#)
- BMP
  - Enumerations, [23](#)
- BRIGHTNESS
  - Enumerations, [26](#)
- brightness
  - FlyCapture2::EmbeddedImageInfo, [125](#)
- BT\_FORCE\_32BITS
  - Enumerations, [19](#)
- BUFFER\_FRAMES
  - Enumerations, [23](#)
- build
  - FlyCapture2::FC2Version, [134](#)
- BUS
  - FlyCapture2::TopologyNode, [230](#)
- BUS\_RESET
  - Enumerations, [19](#)
- BusCallbackType
  - Enumerations, [19](#)
- busErrors
  - FlyCapture2::HostAdapterStats, [181](#)
- BusEventCallback
  - FlyCapture2, [46](#)
- BusManager
  - FlyCapture2::BusManager, [54](#)
- BusManager.h, [242](#)
- busResets
  - FlyCapture2::HostAdapterStats, [181](#)
- BusSpeed
  - Enumerations, [19](#)
- BUSSPEED\_10000BASE\_T
  - Enumerations, [20](#)
- BUSSPEED\_1000BASE\_T
  - Enumerations, [20](#)
- BUSSPEED\_100BASE\_T
  - Enumerations, [20](#)
- BUSSPEED\_10BASE\_T
  - Enumerations, [20](#)
- BUSSPEED\_ANY
  - Enumerations, [20](#)
- BUSSPEED\_FORCE\_32BITS
  - Enumerations, [20](#)
- BUSSPEED\_S100
  - Enumerations, [19](#)
- BUSSPEED\_S1600
  - Enumerations, [19](#)
- BUSSPEED\_S200
  - Enumerations, [19](#)
- BUSSPEED\_S3200
  - Enumerations, [19](#)
- BUSSPEED\_S400
  - Enumerations, [19](#)
- BUSSPEED\_S480
  - Enumerations, [19](#)
- BUSSPEED\_S5000
  - Enumerations, [19](#)
- BUSSPEED\_S800
  - Enumerations, [19](#)
- BUSSPEED\_S\_FASTEST
  - Enumerations, [20](#)
- BUSSPEED\_SPEED\_UNKNOWN
  - Enumerations, [20](#)
- BYTE\_ORDER\_BIG\_ENDIAN
  - FlyCapture2, [46](#)
- BYTE\_ORDER\_FORCE\_32BITS
  - FlyCapture2, [46](#)
- BYTE\_ORDER\_LITTLE\_ENDIAN
  - FlyCapture2, [46](#)
- ByteOrder
  - FlyCapture2, [46](#)
- byteOrder
  - FlyCapture2::SystemInfo, [223](#)
- CalculateStatistics
  - FlyCapture2::Image, [186](#)
- CALLBACK\_TYPE\_FORCE\_32BITS
  - Enumerations, [19](#)
- CallbackHandle
  - FlyCapture2, [46](#)
- CAMERA
  - FlyCapture2::TopologyNode, [230](#)
- Camera
  - FlyCapture2::Camera, [65](#)
- Camera.h, [243](#)
- CameraBase
  - FlyCapture2::CameraBase, [90](#)
- CameraBase.h, [244](#)
- CameraControlDlg
  - FlyCapture2::CameraControlDlg, [109](#)
- cameraCurrents
  - FlyCapture2::CameraStats, [118](#)
- CameraInfo
  - FlyCapture2::CameraInfo, [113](#)
- cameraPowerUp
  - FlyCapture2::CameraStats, [118](#)
- CameraSelectionDlg
  - FlyCapture2::CameraSelectionDlg, [116](#)
- CameraStats
  - FlyCapture2::CameraStats, [118](#)
- cameraVoltages
  - FlyCapture2::CameraStats, [118](#)
- CCITTFAX3
  - FlyCapture2::TIFFOption, [225](#)
- CCITTFAX4
  - FlyCapture2::TIFFOption, [225](#)
- channels
  - FlyCapture2::GigEConfig, [171](#)

- chipIdHi
  - FlyCapture2::ConfigROM, 121
- chipIdLo
  - FlyCapture2::ConfigROM, 121
- CollectSupportInformation
  - FlyCapture2::Error, 128
- COLOR\_PROCESSING\_ALGORITHM\_FORCE\_32BITSFlyCapture2::Image, 187
  - Enumerations, 20
- ColorProcessingAlgorithm
  - Enumerations, 20
- compression
  - FlyCapture2::TIFFOption, 226
- compressionLevel
  - FlyCapture2::PNGOption, 212
- CompressionMethod
  - FlyCapture2::TIFFOption, 225
- COMPUTER
  - FlyCapture2::TopologyNode, 230
- ConfigROM
  - FlyCapture2::ConfigROM, 121
- configROM
  - FlyCapture2::CameraInfo, 113
- Connect
  - FlyCapture2::Camera, 66
  - FlyCapture2::CameraBase, 90
  - FlyCapture2::CameraControlDlg, 110
  - FlyCapture2::GigECamera, 147
- CONNECTED\_TO\_CHILD
  - FlyCapture2::TopologyNode, 230
- CONNECTED\_TO\_PARENT
  - FlyCapture2::TopologyNode, 230
- Convert
  - FlyCapture2::Image, 186
- cpuDescription
  - FlyCapture2::SystemInfo, 223
- cycleCount
  - FlyCapture2::TimeStamp, 227
- cycleOffset
  - FlyCapture2::TimeStamp, 227
- cycleSeconds
  - FlyCapture2::TimeStamp, 227
- cycleTime
  - FlyCapture2::HostAdapterStats, 181
- DCAMFormats, 123
  - numFormats, 123
  - videoModes, 123
- DeepCopy
  - FlyCapture2::Image, 187
- DEFAULT
  - Enumerations, 20
- defaultGateway
  - FlyCapture2::CameraInfo, 113
- DEFLATE
  - FlyCapture2::TIFFOption, 225
- delay
  - FlyCapture2::StrobeControl, 219
- destinationIpAddress
  - FlyCapture2::GigEStreamChannel, 179
- DetermineBitsPerPixel
  - FlyCapture2::Image, 187
- deviceArrivals
  - FlyCapture2::HostAdapterStats, 181
- deviceRemovals
  - FlyCapture2::HostAdapterStats, 181
- DisableAll
  - FlyCapture2::ImageStatistics, 200
- Disconnect
  - FlyCapture2::Camera, 66
  - FlyCapture2::CameraBase, 91
  - FlyCapture2::CameraControlDlg, 110
  - FlyCapture2::GigECamera, 148
- DiscoverGigECameras
  - FlyCapture2::BusManager, 54
- DiscoverGigEPacketSize
  - FlyCapture2::GigECamera, 148
- doNotFragment
  - FlyCapture2::GigEStreamChannel, 179
- DRIVER\_1394\_CAM
  - Enumerations, 20
- DRIVER\_1394\_JUJU
  - Enumerations, 20
- DRIVER\_1394\_PRO
  - Enumerations, 20
- DRIVER\_1394\_RAW1394
  - Enumerations, 20
- DRIVER\_1394\_VIDEO1394
  - Enumerations, 20
- DRIVER\_FORCE\_32BITS
  - Enumerations, 21
- DRIVER\_GIGE\_FILTER
  - Enumerations, 21
- DRIVER\_GIGE\_NONE
  - Enumerations, 21
- DRIVER\_GIGE\_PRO
  - Enumerations, 21
- DRIVER\_UNKNOWN
  - Enumerations, 21
- DRIVER\_USB3\_PRO
  - Enumerations, 21
- DRIVER\_USB\_CAM
  - Enumerations, 21
- DRIVER\_USB\_NONE
  - Enumerations, 20
- driverList
  - FlyCapture2::SystemInfo, 224
- driverName
  - FlyCapture2::CameraInfo, 113

- DriverType
  - Enumerations, [20](#)
- driverType
  - FlyCapture2::CameraInfo, [113](#)
- DROP\_FRAMES
  - Enumerations, [23](#)
- duration
  - FlyCapture2::StrobeControl, [219](#)
- EDGE\_SENSING
  - Enumerations, [20](#)
- embeddedBrightness
  - FlyCapture2::ImageMetadata, [196](#)
- embeddedExposure
  - FlyCapture2::ImageMetadata, [196](#)
- embeddedFrameCounter
  - FlyCapture2::ImageMetadata, [196](#)
- embeddedGain
  - FlyCapture2::ImageMetadata, [196](#)
- embeddedGPIOPinState
  - FlyCapture2::ImageMetadata, [196](#)
- EmbeddedImageInfoProperty
  - FlyCapture2::EmbeddedImageInfoProperty, [126](#)
- embeddedROIPosition
  - FlyCapture2::ImageMetadata, [196](#)
- embeddedShutter
  - FlyCapture2::ImageMetadata, [196](#)
- embeddedStrobePattern
  - FlyCapture2::ImageMetadata, [196](#)
- embeddedTimeStamp
  - FlyCapture2::ImageMetadata, [196](#)
- embeddedWhiteBalance
  - FlyCapture2::ImageMetadata, [196](#)
- EnableAll
  - FlyCapture2::ImageStatistics, [200](#)
- enabled
  - FlyCapture2::LUTData, [207](#)
- EnableGreyOnly
  - FlyCapture2::ImageStatistics, [200](#)
- EnableHSLOnly
  - FlyCapture2::ImageStatistics, [200](#)
- EnableLUT
  - FlyCapture2::Camera, [66](#)
  - FlyCapture2::CameraBase, [91](#)
  - FlyCapture2::GigECamera, [148](#)
- EnableRGBOnly
  - FlyCapture2::ImageStatistics, [200](#)
- Enumerations, [12](#)
  - ARRIVAL, [19](#)
  - AUTO\_EXPOSURE, [26](#)
  - BANDWIDTH\_ALLOCATION\_FORCE\_32BITS, [19](#)
  - BANDWIDTH\_ALLOCATION\_OFF, [18](#)
  - BANDWIDTH\_ALLOCATION\_ON, [18](#)
  - BANDWIDTH\_ALLOCATION\_UNSPECIFIED, [19](#)
  - BANDWIDTH\_ALLOCATION\_UNSUPPORTED, [18](#)
  - BandwidthAllocation, [18](#)
  - BayerTileFormat, [19](#)
  - BGGR, [19](#)
  - BMP, [23](#)
  - BRIGHTNESS, [26](#)
  - BT\_FORCE\_32BITS, [19](#)
  - BUFFER\_FRAMES, [23](#)
  - BUS\_RESET, [19](#)
  - BusCallbackType, [19](#)
  - BusSpeed, [19](#)
  - BUSSPEED\_10000BASE\_T, [20](#)
  - BUSSPEED\_1000BASE\_T, [20](#)
  - BUSSPEED\_100BASE\_T, [20](#)
  - BUSSPEED\_10BASE\_T, [20](#)
  - BUSSPEED\_ANY, [20](#)
  - BUSSPEED\_FORCE\_32BITS, [20](#)
  - BUSSPEED\_S100, [19](#)
  - BUSSPEED\_S1600, [19](#)
  - BUSSPEED\_S200, [19](#)
  - BUSSPEED\_S3200, [19](#)
  - BUSSPEED\_S400, [19](#)
  - BUSSPEED\_S480, [19](#)
  - BUSSPEED\_S5000, [19](#)
  - BUSSPEED\_S800, [19](#)
  - BUSSPEED\_S\_FASTEST, [20](#)
  - BUSSPEED\_SPEED\_UNKNOWN, [20](#)
  - CALLBACK\_TYPE\_FORCE\_32BITS, [19](#)
  - COLOR\_PROCESSING\_ALGORITHM\_FORCE, [20](#)
  - ColorProcessingAlgorithm, [20](#)
  - DEFAULT, [20](#)
  - DRIVER\_1394\_CAM, [20](#)
  - DRIVER\_1394\_JUJU, [20](#)
  - DRIVER\_1394\_PRO, [20](#)
  - DRIVER\_1394\_RAW1394, [20](#)
  - DRIVER\_1394\_VIDEO1394, [20](#)
  - DRIVER\_FORCE\_32BITS, [21](#)
  - DRIVER\_GIGE\_FILTER, [21](#)
  - DRIVER\_GIGE\_NONE, [21](#)
  - DRIVER\_GIGE\_PRO, [21](#)
  - DRIVER\_UNKNOWN, [21](#)
  - DRIVER\_USB3\_PRO, [21](#)
  - DRIVER\_USB\_CAM, [21](#)
  - DRIVER\_USB\_NONE, [20](#)
  - DriverType, [20](#)
  - DROP\_FRAMES, [23](#)
  - EDGE\_SENSING, [20](#)
  - ErrorType, [21](#)
  - FOCUS, [26](#)
  - FRAME\_RATE, [26](#)
  - FrameRate, [22](#)
  - FRAMERATE\_120, [22](#)
  - FRAMERATE\_15, [22](#)
  - FRAMERATE\_1\_875, [22](#)

FRAMERATE\_240, 22  
 FRAMERATE\_30, 22  
 FRAMERATE\_3\_75, 22  
 FRAMERATE\_60, 22  
 FRAMERATE\_7\_5, 22  
 FRAMERATE\_FORCE\_32BITS, 22  
 FRAMERATE\_FORMAT7, 22  
 FROM\_FILE\_EXT, 23  
 GAIN, 26  
 GAMMA, 26  
 GBRG, 19  
 GRAB\_MODE\_FORCE\_32BITS, 23  
 GRAB\_TIMEOUT\_FORCE\_32BITS, 23  
 GrabMode, 22  
 GrabTimeout, 23  
 GRBG, 19  
 HQ\_LINEAR, 20  
 HUE, 26  
 IMAGE\_FILE\_FORMAT\_FORCE\_32BITS, 24  
 ImageFileFormat, 23  
 INTERFACE\_GIGE, 24  
 INTERFACE\_IEEE1394, 24  
 INTERFACE\_TYPE\_FORCE\_32BITS, 24  
 INTERFACE\_UNKNOWN, 24  
 INTERFACE\_USB2, 24  
 InterfaceType, 24  
 IPP, 20  
 IRIS, 26  
 JPEG, 23  
 JPEG2000, 23  
 Mode, 24  
 MODE\_0, 24  
 MODE\_1, 24  
 MODE\_10, 24  
 MODE\_11, 24  
 MODE\_12, 24  
 MODE\_13, 24  
 MODE\_14, 24  
 MODE\_15, 24  
 MODE\_16, 24  
 MODE\_17, 24  
 MODE\_18, 24  
 MODE\_19, 24  
 MODE\_2, 24  
 MODE\_20, 24  
 MODE\_21, 24  
 MODE\_22, 25  
 MODE\_23, 25  
 MODE\_24, 25  
 MODE\_25, 25  
 MODE\_26, 25  
 MODE\_27, 25  
 MODE\_28, 25  
 MODE\_29, 25  
 MODE\_3, 24  
 MODE\_30, 25  
 MODE\_31, 25  
 MODE\_4, 24  
 MODE\_5, 24  
 MODE\_6, 24  
 MODE\_7, 24  
 MODE\_8, 24  
 MODE\_9, 24  
 MODE\_FORCE\_32BITS, 25  
 NEAREST\_NEIGHBOR, 20  
 NO\_COLOR\_PROCESSING, 20  
 NONE, 19  
 NUM\_FRAMERATES, 22  
 NUM\_MODES, 25  
 NUM\_PIXEL\_FORMATS, 25  
 NUM\_VIDEOMODES, 27  
 PAN, 26  
 PGM, 23  
 PGRERROR\_BUFFER\_TOO\_SMALL, 22  
 PGRERROR\_BUS\_MASTER\_FAILED, 21  
 PGRERROR\_FAILED, 21  
 PGRERROR\_FAILED\_BUS\_MASTER\_CONNECTION, 21  
 PGRERROR\_FAILED\_GUID, 21  
 PGRERROR\_FORCE\_32BITS, 22  
 PGRERROR\_IIDC\_FAILED, 21  
 PGRERROR\_IMAGE\_CONSISTENCY\_ERROR, 21  
 PGRERROR\_IMAGE\_CONVERSION\_FAILED, 21  
 PGRERROR\_IMAGE\_LIBRARY\_FAILURE, 22  
 PGRERROR\_INIT\_FAILED, 21  
 PGRERROR\_INVALID\_BUS\_MANAGER, 21  
 PGRERROR\_INVALID\_GENERATION, 21  
 PGRERROR\_INVALID\_MODE, 21  
 PGRERROR\_INVALID\_PACKET\_SIZE, 21  
 PGRERROR\_INVALID\_PARAMETER, 21  
 PGRERROR\_INVALID\_SETTINGS, 21  
 PGRERROR\_ISOCH\_ALREADY\_STARTED, 22  
 PGRERROR\_ISOCH\_BANDWIDTH\_EXCEEDED, 22  
 PGRERROR\_ISOCH\_FAILED, 22  
 PGRERROR\_ISOCH\_NOT\_STARTED, 22  
 PGRERROR\_ISOCH\_RETRIEVE\_BUFFER\_FAILED, 22  
 PGRERROR\_ISOCH\_START\_FAILED, 22  
 PGRERROR\_ISOCH\_STOP\_FAILED, 22  
 PGRERROR\_ISOCH\_SYNC\_FAILED, 22  
 PGRERROR\_LOW\_LEVEL\_FAILURE, 21  
 PGRERROR\_LUT\_FAILED, 21  
 PGRERROR\_MEMORY\_ALLOCATION\_FAILED, 21  
 PGRERROR\_NOT\_CONNECTED, 21  
 PGRERROR\_NOT\_FOUND, 21  
 PGRERROR\_NOT\_IMPLEMENTED, 21  
 PGRERROR\_NOT\_IN\_FORMAT7, 21  
 PGRERROR\_NOT\_INITIALIZED, 21  
 PGRERROR\_NOT\_SUPPORTED, 21  
 PGRERROR\_OK, 21

- PGRERROR\_PROPERTY\_FAILED, [21](#)
- PGRERROR\_PROPERTY\_NOT\_PRESENT, [22](#)
- PGRERROR\_READ\_REGISTER\_FAILED, [22](#)
- PGRERROR\_REGISTER\_FAILED, [22](#)
- PGRERROR\_STROBE\_FAILED, [21](#)
- PGRERROR\_TIMEOUT, [21](#)
- PGRERROR\_TRIGGER\_FAILED, [21](#)
- PGRERROR\_UNDEFINED, [21](#)
- PGRERROR\_WRITE\_REGISTER\_FAILED, [22](#)
- PIXEL\_FORMAT\_411YUV8, [25](#)
- PIXEL\_FORMAT\_422YUV8, [25](#)
- PIXEL\_FORMAT\_444YUV8, [25](#)
- PIXEL\_FORMAT\_BGR, [25](#)
- PIXEL\_FORMAT\_BGRU, [25](#)
- PIXEL\_FORMAT\_MONO12, [25](#)
- PIXEL\_FORMAT\_MONO16, [25](#)
- PIXEL\_FORMAT\_MONO8, [25](#)
- PIXEL\_FORMAT\_RAW12, [25](#)
- PIXEL\_FORMAT\_RAW16, [25](#)
- PIXEL\_FORMAT\_RAW8, [25](#)
- PIXEL\_FORMAT\_RGB, [25](#)
- PIXEL\_FORMAT\_RGB16, [25](#)
- PIXEL\_FORMAT\_RGB8, [25](#)
- PIXEL\_FORMAT\_RGBU, [25](#)
- PIXEL\_FORMAT\_S\_MONO16, [25](#)
- PIXEL\_FORMAT\_S\_RGB16, [25](#)
- PixelFormat, [25](#)
- PNG, [23](#)
- PPM, [23](#)
- PROPERTY\_TYPE\_FORCE\_32BITS, [26](#)
- PropertyType, [25](#)
- RAW, [24](#)
- REMOVAL, [19](#)
- RGGB, [19](#)
- RIGOROUS, [20](#)
- SATURATION, [26](#)
- SHARPNESS, [26](#)
- SHUTTER, [26](#)
- TEMPERATURE, [26](#)
- TIFF, [23](#)
- TILT, [26](#)
- TIMEOUT\_INFINITE, [23](#)
- TIMEOUT\_NONE, [23](#)
- TIMEOUT\_UNSPECIFIED, [23](#)
- TRIGGER\_DELAY, [26](#)
- TRIGGER\_MODE, [26](#)
- UNSPECIFIED\_GRAB\_MODE, [23](#)
- UNSPECIFIED\_PIXEL\_FORMAT, [25](#)
- UNSPECIFIED\_PROPERTY\_TYPE, [26](#)
- VideoMode, [26](#)
- VIDEOMODE\_1024x768RGB, [27](#)
- VIDEOMODE\_1024x768Y16, [27](#)
- VIDEOMODE\_1024x768Y8, [27](#)
- VIDEOMODE\_1024x768YUV422, [27](#)
- VIDEOMODE\_1280x960RGB, [27](#)
- VIDEOMODE\_1280x960Y16, [27](#)
- VIDEOMODE\_1280x960Y8, [27](#)
- VIDEOMODE\_1280x960YUV422, [27](#)
- VIDEOMODE\_1600x1200RGB, [27](#)
- VIDEOMODE\_1600x1200Y16, [27](#)
- VIDEOMODE\_1600x1200Y8, [27](#)
- VIDEOMODE\_1600x1200YUV422, [27](#)
- VIDEOMODE\_160x120YUV444, [26](#)
- VIDEOMODE\_320x240YUV422, [26](#)
- VIDEOMODE\_640x480RGB, [26](#)
- VIDEOMODE\_640x480Y16, [26](#)
- VIDEOMODE\_640x480Y8, [26](#)
- VIDEOMODE\_640x480YUV411, [26](#)
- VIDEOMODE\_640x480YUV422, [26](#)
- VIDEOMODE\_800x600RGB, [26](#)
- VIDEOMODE\_800x600Y16, [26](#)
- VIDEOMODE\_800x600Y8, [26](#)
- VIDEOMODE\_800x600YUV422, [26](#)
- VIDEOMODE\_FORCE\_32BITS, [27](#)
- VIDEOMODE\_FORMAT7, [27](#)
- WHITE\_BALANCE, [26](#)
- ZOOM, [26](#)
- Error
  - FlyCapture2::Error, [128](#)
- Error.h, [245](#)
- ErrorType
  - Enumerations, [21](#)
- exposure
  - FlyCapture2::EmbeddedImageInfo, [125](#)
- FC2Config
  - FlyCapture2::FC2Config, [132](#)
- fifoOverflows
  - FlyCapture2::HostAdapterStats, [181](#)
- FireBusReset
  - FlyCapture2::BusManager, [55](#)
- FireSoftwareTrigger
  - FlyCapture2::Camera, [66](#)
  - FlyCapture2::CameraBase, [91](#)
  - FlyCapture2::GigECamera, [148](#)
- firmwareBuildTime
  - FlyCapture2::CameraInfo, [113](#)
- firmwareVersion
  - FlyCapture2::CameraInfo, [113](#)
- FlyCapture2, [35](#)
  - AsyncCommandCallback, [46](#)
  - BusEventCallback, [46](#)
  - BYTE\_ORDER\_BIG\_ENDIAN, [46](#)
  - BYTE\_ORDER\_FORCE\_32BITS, [46](#)
  - BYTE\_ORDER\_LITTLE\_ENDIAN, [46](#)
  - ByteOrder, [46](#)
  - CallbackHandle, [46](#)
  - ImageEventCallback, [46](#)



- LINUX\_X64, 47
- LINUX\_X86, 47
- MAC, 47
- OSType, 46
- OSTYPE\_FORCE\_32BITS, 47
- UNKNOWN\_OS, 47
- WINDOWS\_X64, 47
- WINDOWS\_X86, 47
- FlyCapture2.h, 246
- FlyCapture2::AVIOption, 49
  - AVIOption, 49
  - frameRate, 49
  - reserved, 49
- FlyCapture2::AVIRecorder, 51
  - ~AVIRecorder, 51
  - AVIAppend, 51
  - AVIClose, 51
  - AVIOpen, 52
  - AVIRecorder, 51
- FlyCapture2::BusManager, 53
  - ~BusManager, 54
  - BusManager, 54
  - DiscoverGigECameras, 54
  - FireBusReset, 55
  - ForceIPAddressToCamera, 55
  - GetCameraFromIndex, 55
  - GetCameraFromIPAddress, 56
  - GetCameraFromSerialNumber, 56
  - GetCameraSerialNumberFromIndex, 56
  - GetDeviceFromIndex, 57
  - GetInterfaceTypeFromGuid, 57
  - GetNumOfCameras, 57
  - GetNumOfDevices, 58
  - GetTopology, 58
  - ReadPhyRegister, 58
  - RegisterCallback, 58
  - RescanBus, 59
  - UnregisterCallback, 59
  - WritePhyRegister, 59
- FlyCapture2::Camera, 61
  - ~Camera, 65
  - Camera, 65
  - Connect, 66
  - Disconnect, 66
  - EnableLUT, 66
  - FireSoftwareTrigger, 66
  - GetActiveLUTBank, 66
  - GetCameraInfo, 67
  - GetConfiguration, 67
  - GetEmbeddedImageInfo, 67
  - GetFormat7Configuration, 68
  - GetFormat7Info, 68
  - GetGPIOPinDirection, 68
  - GetLUTBankInfo, 69
  - GetLUTChannel, 69
  - GetLUTInfo, 70
  - GetMemoryChannel, 70
  - GetMemoryChannelInfo, 70
  - GetProperty, 71
  - GetPropertyInfo, 71
  - GetRegisterString, 72
  - GetStrobe, 72
  - GetStrobeInfo, 72
  - GetTriggerDelay, 73
  - GetTriggerDelayInfo, 73
  - GetTriggerMode, 73
  - GetTriggerModeInfo, 74
  - GetVideoModeAndFrameRate, 74
  - GetVideoModeAndFrameRateInfo, 75
  - IsConnected, 75
  - ReadRegister, 75
  - ReadRegisterBlock, 76
  - RestoreFromMemoryChannel, 76
  - RetrieveBuffer, 76
  - SaveToMemoryChannel, 77
  - SetActiveLUTBank, 77
  - SetCallback, 77
  - SetConfiguration, 78
  - SetEmbeddedImageInfo, 78
  - SetFormat7Configuration, 78, 79
  - SetGPIOPinDirection, 79
  - SetLUTChannel, 80
  - SetProperty, 80
  - SetStrobe, 81
  - SetTriggerDelay, 81
  - SetTriggerMode, 81
  - SetUserBuffers, 82
  - SetVideoModeAndFrameRate, 82
  - StartCapture, 83
  - StartSyncCapture, 83
  - StopCapture, 83
  - ValidateFormat7Settings, 83
  - WaitForBufferEvent, 84
  - WriteRegister, 84
  - WriteRegisterBlock, 85
- FlyCapture2::CameraBase, 86
  - ~CameraBase, 90
  - CameraBase, 90
  - Connect, 90
  - Disconnect, 91
  - EnableLUT, 91
  - FireSoftwareTrigger, 91
  - GetActiveLUTBank, 91
  - GetCameraInfo, 92
  - GetConfiguration, 92
  - GetEmbeddedImageInfo, 92
  - GetGPIOPinDirection, 93
  - GetLUTBankInfo, 93



- GetLUTChannel, 93
- GetLUTInfo, 94
- GetMemoryChannel, 94
- GetMemoryChannelInfo, 95
- GetProperty, 95
- GetPropertyInfo, 95
- GetRegisterString, 96
- GetStrobe, 96
- GetStrobeInfo, 96
- GetTriggerDelay, 97
- GetTriggerDelayInfo, 97
- GetTriggerMode, 98
- GetTriggerModeInfo, 98
- IsConnected, 98
- m\_pCameraData, 108
- ReadRegister, 99
- ReadRegisterBlock, 99
- RestoreFromMemoryChannel, 99
- RetrieveBuffer, 100
- SaveToMemoryChannel, 100
- SetActiveLUTBank, 101
- SetCallback, 101
- SetConfiguration, 101
- SetEmbeddedImageInfo, 101
- SetGPIOPinDirection, 102
- SetLUTChannel, 102
- SetProperty, 103
- SetStrobe, 103
- SetTriggerDelay, 104
- SetTriggerMode, 104
- SetUserBuffers, 104
- StartCapture, 105
- StartSyncCapture, 105
- StopCapture, 106
- WaitForBufferEvent, 106
- WriteRegister, 107
- WriteRegisterBlock, 107
- FlyCapture2::CameraControlDlg, 109
  - ~CameraControlDlg, 109
  - CameraControlDlg, 109
  - Connect, 110
  - Disconnect, 110
  - Hide, 110
  - IsVisible, 110
  - Show, 110
- FlyCapture2::CameraInfo, 111
  - bayerTileFormat, 113
  - CameraInfo, 113
  - configROM, 113
  - defaultGateway, 113
  - driverName, 113
  - driverType, 113
  - firmwareBuildTime, 113
  - firmwareVersion, 113
  - gigEMajorVersion, 113
  - gigEMinorVersion, 113
  - iideVer, 113
  - interfaceType, 114
  - ipAddress, 114
  - isColorCamera, 114
  - macAddress, 114
  - maximumBusSpeed, 114
  - modelName, 114
  - reserved, 114
  - sensorInfo, 114
  - sensorResolution, 114
  - serialNumber, 114
  - subnetMask, 114
  - userDefinedName, 115
  - vendorName, 115
  - xmlURL1, 115
  - xmlURL2, 115
- FlyCapture2::CameraSelectionDlg, 116
  - ~CameraSelectionDlg, 116
  - CameraSelectionDlg, 116
  - ShowModal, 116
- FlyCapture2::CameraStats, 117
  - cameraCurrents, 118
  - cameraPowerUp, 118
  - CameraStats, 118
  - cameraVoltages, 118
  - imageCorrupt, 118
  - imageDriverDropped, 118
  - imageDropped, 118
  - imageXmitFailed, 118
  - numCurrents, 118
  - numVoltages, 118
  - portErrors, 118
  - regReadFailed, 118
  - regWriteFailed, 118
  - reserved, 118
  - temperature, 118
  - timeSinceBusReset, 119
  - timeSinceInitialization, 119
  - timeStamp, 119
- FlyCapture2::ConfigROM, 120
  - chipIdHi, 121
  - chipIdLo, 121
  - ConfigROM, 121
  - nodeVendorId, 121
  - pszKeyword, 121
  - reserved, 121
  - unitSpecId, 121
  - unitSubSWVer, 121
  - unitSWVer, 121
  - vendorUniqueInfo\_0, 121
  - vendorUniqueInfo\_1, 121
  - vendorUniqueInfo\_2, 122

- vendorUniqueInfo\_3, 122
- FlyCapture2::EmbeddedImageInfo, 124
  - brightness, 125
  - exposure, 125
  - frameCounter, 125
  - gain, 125
  - GPIOPinState, 125
  - ROIPosition, 125
  - shutter, 125
  - strobePattern, 125
  - timestamp, 125
  - whiteBalance, 125
- FlyCapture2::EmbeddedImageInfoProperty, 126
  - available, 126
  - EmbeddedImageInfoProperty, 126
  - onOff, 126
- FlyCapture2::Error, 127
  - ~Error, 128
  - CollectSupportInformation, 128
  - Error, 128
  - GetBuildDate, 128
  - GetCause, 128
  - GetDescription, 129
  - GetFilename, 129
  - GetLine, 129
  - GetType, 129
  - InternalError, 130
  - operator=, 129
  - operator==, 130
  - PrintErrorTrace, 130
- FlyCapture2::FC2Config, 131
  - asyncBusSpeed, 132
  - bandwidthAllocation, 132
  - FC2Config, 132
  - grabMode, 132
  - grabTimeout, 132
  - isochBusSpeed, 132
  - numBuffers, 132
  - numImageNotifications, 132
  - reserved, 132
- FlyCapture2::FC2Version, 134
  - build, 134
  - major, 134
  - minor, 134
  - type, 134
- FlyCapture2::Format7ImageSettings, 135
  - Format7ImageSettings, 135
  - height, 135
  - mode, 135
  - offsetX, 136
  - offsetY, 136
  - pixelFormat, 136
  - reserved, 136
  - width, 136
- FlyCapture2::Format7Info, 137
  - Format7Info, 138
  - imageHStepSize, 138
  - imageVStepSize, 138
  - maxHeight, 138
  - maxPacketSize, 138
  - maxWidth, 138
  - minPacketSize, 138
  - mode, 138
  - offsetHStepSize, 138
  - offsetVStepSize, 138
  - packetSize, 138
  - percentage, 139
  - pixelFormatBitField, 139
  - reserved, 139
- FlyCapture2::Format7PacketInfo, 140
  - Format7PacketInfo, 140
  - maxBytesPerPacket, 140
  - recommendedBytesPerPacket, 140
  - reserved, 140
  - unitBytesPerPacket, 140
- FlyCapture2::GigECamera, 142
  - ~GigECamera, 147
  - Connect, 147
  - Disconnect, 148
  - DiscoverGigEPacketSize, 148
  - EnableLUT, 148
  - FireSoftwareTrigger, 148
  - GetActiveLUTBank, 149
  - GetCameraInfo, 149
  - GetConfiguration, 149
  - GetEmbeddedImageInfo, 150
  - GetGigEImageBinningSettings, 150
  - GetGigEImageSettings, 150
  - GetGigEImageSettingsInfo, 150
  - GetGigEImagingMode, 151
  - GetGigEProperty, 151
  - GetGigEStreamChannelInfo, 151
  - GetGPIOPinDirection, 151
  - GetLUTBankInfo, 152
  - GetLUTChannel, 152
  - GetLUTInfo, 153
  - GetMemoryChannel, 153
  - GetMemoryChannelInfo, 153
  - GetNumStreamChannels, 154
  - GetProperty, 154
  - GetPropertyInfo, 154
  - GetRegisterString, 155
  - GetStrobe, 155
  - GetStrobeInfo, 155
  - GetTriggerDelay, 156
  - GetTriggerDelayInfo, 156
  - GetTriggerMode, 157
  - GetTriggerModeInfo, 157

- GigECamera, 147
- IsConnected, 157
- QueryGigEImagingMode, 158
- ReadGVCPMemory, 158
- ReadGVCPRegister, 158
- ReadGVCPRegisterBlock, 158
- ReadRegister, 159
- ReadRegisterBlock, 159
- RestoreFromMemoryChannel, 159
- RetrieveBuffer, 160
- SaveToMemoryChannel, 160
- SetActiveLUTBank, 161
- SetCallback, 161
- SetConfiguration, 161
- SetEmbeddedImageInfo, 161
- SetGigEImageBinningSettings, 162
- SetGigEImageSettings, 162
- SetGigEImagingMode, 162
- SetGigEProperty, 163
- SetGigEStreamChannelInfo, 163
- SetGPIOPinDirection, 163
- SetLUTChannel, 164
- SetProperty, 164
- SetStrobe, 164
- SetTriggerDelay, 165
- SetTriggerMode, 165
- SetUserBuffers, 166
- StartCapture, 166
- StartSyncCapture, 167
- StopCapture, 167
- WaitForBufferEvent, 167
- WriteGVCPMemory, 167
- WriteGVCPRegister, 168
- WriteGVCPRegisterBlock, 168
- WriteRegister, 168
- WriteRegisterBlock, 169
- FlyCapture2::GigEConfig, 170
  - channels, 171
  - GigEConfig, 171
  - numChannels, 171
- FlyCapture2::GigEImageSettings, 172
  - GigEImageSettings, 172
  - height, 172
  - offsetX, 172
  - offsetY, 172
  - pixelFormat, 173
  - reserved, 173
  - width, 173
- FlyCapture2::GigEImageSettingsInfo, 174
  - GigEImageSettingsInfo, 174
  - imageHStepSize, 174
  - imageVStepSize, 174
  - maxHeight, 175
  - maxWidth, 175
  - offsetHStepSize, 175
  - offsetVStepSize, 175
  - pixelFormatBitField, 175
  - reserved, 175
- FlyCapture2::GigEProperty, 176
  - isReadable, 176
  - isWritable, 176
  - max, 176
  - min, 176
  - propType, 176
  - value, 177
- FlyCapture2::GigEStreamChannel, 178
  - destinationIpAddress, 179
  - doNotFragment, 179
  - GigEStreamChannel, 179
  - hostPort, 179
  - interPacketDelay, 179
  - networkInterfaceIndex, 179
  - packetSize, 179
  - sourcePort, 179
- FlyCapture2::HostAdapterStats, 180
  - busErrors, 181
  - busResets, 181
  - cycleTime, 181
  - deviceArrivals, 181
  - deviceRemovals, 181
  - fifoOverflows, 181
  - gapCount, 181
  - HostAdapterStats, 181
  - numPorts, 181
  - portErrors, 181
  - vendor, 181
- FlyCapture2::Image, 182
  - ~Image, 186
  - CalculateStatistics, 186
  - Convert, 186
  - DeepCopy, 187
  - DetermineBitsPerPixel, 187
  - GetBayerTileFormat, 187
  - GetBitsPerPixel, 187
  - GetColorProcessing, 187
  - GetCols, 188
  - GetData, 188
  - GetDataSize, 188
  - GetDefaultColorProcessing, 188
  - GetDefaultOutputFormat, 188
  - GetDimensions, 189
  - GetMetadata, 189
  - GetPixelFormat, 189
  - GetRows, 189
  - GetStride, 189
  - GetTimeStamp, 190
  - Image, 185
  - Iso, 194

- operator(), 190
- operator=, 190
- ReleaseBuffer, 190
- Save, 191, 192
- SetColorProcessing, 192
- SetData, 193
- SetDefaultColorProcessing, 193
- SetDefaultOutputFormat, 193
- SetDimensions, 194
- FlyCapture2::ImageMetadata, 195
  - embeddedBrightness, 196
  - embeddedExposure, 196
  - embeddedFrameCounter, 196
  - embeddedGain, 196
  - embeddedGPIOPinState, 196
  - embeddedROIPosition, 196
  - embeddedShutter, 196
  - embeddedStrobePattern, 196
  - embeddedTimeStamp, 196
  - embeddedWhiteBalance, 196
  - ImageMetadata, 196
  - reserved, 196
- FlyCapture2::ImageStatistics, 198
  - ~ImageStatistics, 200
  - BLUE, 199
  - DisableAll, 200
  - EnableAll, 200
  - EnableGreyOnly, 200
  - EnableHSLOnly, 200
  - EnableRGBOnly, 200
  - GetChannelStatus, 200
  - GetHistogram, 201
  - GetMean, 201
  - GetNumPixelValues, 201
  - GetPixelValueRange, 201
  - GetRange, 202
  - GetStatistics, 202
  - GREEN, 199
  - GREY, 199
  - HUE, 199
  - ImageStatistics, 200
  - ImageStatsCalculator, 203
  - LIGHTNESS, 199
  - NUM\_STATISTICS\_CHANNELS, 199
  - operator=, 202
  - RED, 199
  - SATURATION, 199
  - SetChannelStatus, 203
  - StatisticsChannel, 199
- FlyCapture2::IPAddress, 204
  - IPAddress, 204
  - octets, 204
  - operator==, 204
- FlyCapture2::JPEGOption, 205
  - JPEGOption, 205
  - progressive, 205
  - quality, 205
  - reserved, 205
- FlyCapture2::JPG2Option, 206
  - JPG2Option, 206
  - quality, 206
  - reserved, 206
- FlyCapture2::LUTData, 207
  - enabled, 207
  - inputBitDepth, 207
  - LUTData, 207
  - numBanks, 207
  - numChannels, 208
  - numEntries, 208
  - outputBitDepth, 208
  - reserved, 208
  - supported, 208
- FlyCapture2::MACAddress, 209
  - MACAddress, 209
  - octets, 209
  - operator==, 209
- FlyCapture2::PGMOption, 210
  - binaryFile, 210
  - PGMOption, 210
  - reserved, 210
- FlyCapture2::PGRGuid, 211
  - operator==, 211
  - PGRGuid, 211
  - value, 211
- FlyCapture2::PNGOption, 212
  - compressionLevel, 212
  - interlaced, 212
  - PNGOption, 212
  - reserved, 212
- FlyCapture2::PPMOption, 213
  - binaryFile, 213
  - PPMOption, 213
  - reserved, 213
- FlyCapture2::Property, 214
  - absControl, 215
  - absValue, 215
  - autoManualMode, 215
  - onePush, 215
  - onOff, 215
  - present, 215
  - Property, 215
  - reserved, 215
  - type, 215
  - valueA, 215
  - valueB, 215
- FlyCapture2::PropertyInfo, 216
  - absMax, 217
  - absMin, 217

- absValSupported, 217
- autoSupported, 217
- manualSupported, 217
- max, 217
- min, 217
- onePushSupported, 217
- onOffSupported, 217
- present, 217
- PropertyInfo, 217
- pUnitAbbr, 218
- pUnits, 218
- readOutSupported, 218
- reserved, 218
- type, 218
- FlyCapture2::StrobeControl, 219
  - delay, 219
  - duration, 219
  - onOff, 219
  - polarity, 219
  - reserved, 220
  - source, 220
  - StrobeControl, 219
- FlyCapture2::StrobeInfo, 221
  - maxValue, 221
  - minValue, 221
  - onOffSupported, 221
  - polaritySupported, 222
  - present, 222
  - readOutSupported, 222
  - reserved, 222
  - source, 222
  - StrobeInfo, 221
- FlyCapture2::SystemInfo, 223
  - byteOrder, 223
  - cpuDescription, 223
  - driverList, 224
  - gpuDescription, 224
  - libraryList, 224
  - numCpuCores, 224
  - osDescription, 224
  - osType, 224
  - reserved, 224
  - screenHeight, 224
  - screenWidth, 224
  - sysMemSize, 224
- FlyCapture2::TIFFOption, 225
  - ADOBE\_DEFLATE, 225
  - CCITTFAX3, 225
  - CCITTFAX4, 225
  - compression, 226
  - CompressionMethod, 225
  - DEFLATE, 225
  - JPEG, 225
  - LZW, 225
  - NONE, 225
  - PACKBITS, 225
  - reserved, 226
  - TIFFOption, 226
- FlyCapture2::TimeStamp, 227
  - cycleCount, 227
  - cycleOffset, 227
  - cycleSeconds, 227
  - microSeconds, 227
  - reserved, 227
  - seconds, 228
  - TimeStamp, 227
- FlyCapture2::TopologyNode, 229
  - ~TopologyNode, 230
  - AddChild, 231
  - AddPort, 231
  - AssignGuidToNode, 231
  - BUS, 230
  - CAMERA, 230
  - COMPUTER, 230
  - CONNECTED\_TO\_CHILD, 230
  - CONNECTED\_TO\_PARENT, 230
  - GetChild, 231
  - GetDeviceId, 231
  - GetGuid, 231
  - GetInterfaceType, 232
  - GetNodeType, 232
  - GetNumChildren, 232
  - GetNumPorts, 232
  - GetPortType, 232
  - NODE, 230
  - NodeType, 230
  - NOT\_CONNECTED, 230
  - operator=, 232
  - PortType, 230
  - TopologyNode, 230
- FlyCapture2::TriggerMode, 233
  - mode, 233
  - onOff, 233
  - parameter, 233
  - polarity, 233
  - reserved, 233
  - source, 233
  - TriggerMode, 233
- FlyCapture2::TriggerModeInfo, 235
  - modeMask, 235
  - onOffSupported, 235
  - polaritySupported, 235
  - present, 235
  - readOutSupported, 236
  - reserved, 236
  - softwareTriggerSupported, 236
  - sourceMask, 236
  - TriggerModeInfo, 235

- valueReadable, [236](#)
- FlyCapture2::Utilities, [237](#)
  - GetLibraryVersion, [237](#)
  - GetSystemInfo, [237](#)
  - LaunchBrowser, [237](#)
  - LaunchCommand, [238](#)
  - LaunchCommandAsync, [238](#)
  - LaunchHelp, [238](#)
- FLYCAPTURE2\_API
  - FlyCapture2Platform.h, [256](#)
- FlyCapture2Defs.h, [247](#)
  - FULL\_32BIT\_VALUE, [254](#)
  - NULL, [254](#)
- FlyCapture2GUI.h, [255](#)
- FlyCapture2Platform.h, [256](#)
  - FLYCAPTURE2\_API, [256](#)
- FOCUS
  - Enumerations, [26](#)
- ForceIPAddressToCamera
  - FlyCapture2::BusManager, [55](#)
- Format7ImageSettings
  - FlyCapture2::Format7ImageSettings, [135](#)
- Format7Info
  - FlyCapture2::Format7Info, [138](#)
- Format7PacketInfo
  - FlyCapture2::Format7PacketInfo, [140](#)
- FRAME\_RATE
  - Enumerations, [26](#)
- frameCounter
  - FlyCapture2::EmbeddedImageInfo, [125](#)
- FrameRate
  - Enumerations, [22](#)
- frameRate
  - FlyCapture2::AVIOption, [49](#)
  - VideoModes, [239](#)
- FRAMERATE\_120
  - Enumerations, [22](#)
- FRAMERATE\_15
  - Enumerations, [22](#)
- FRAMERATE\_1\_875
  - Enumerations, [22](#)
- FRAMERATE\_240
  - Enumerations, [22](#)
- FRAMERATE\_30
  - Enumerations, [22](#)
- FRAMERATE\_3\_75
  - Enumerations, [22](#)
- FRAMERATE\_60
  - Enumerations, [22](#)
- FRAMERATE\_7\_5
  - Enumerations, [22](#)
- FRAMERATE\_FORCE\_32BITS
  - Enumerations, [22](#)
- FRAMERATE\_FORMAT7
  - Enumerations, [22](#)
- FROM\_FILE\_EXT
  - Enumerations, [23](#)
- FULL\_32BIT\_VALUE
  - FlyCapture2Defs.h, [254](#)
- GAIN
  - Enumerations, [26](#)
- gain
  - FlyCapture2::EmbeddedImageInfo, [125](#)
- GAMMA
  - Enumerations, [26](#)
- gapCount
  - FlyCapture2::HostAdapterStats, [181](#)
- GBRG
  - Enumerations, [19](#)
- GetAbsBrightness
  - PGRDirectShow.h, [265](#)
- GetAbsBrightnessRange
  - PGRDirectShow.h, [265](#)
- GetAbsExposure
  - PGRDirectShow.h, [265](#)
- GetAbsExposureRange
  - PGRDirectShow.h, [265](#)
- GetAbsGain
  - PGRDirectShow.h, [265](#)
- GetAbsGainRange
  - PGRDirectShow.h, [265](#)
- GetAbsGamma
  - PGRDirectShow.h, [265](#)
- GetAbsGammaRange
  - PGRDirectShow.h, [265](#)
- GetAbsHue
  - PGRDirectShow.h, [265](#)
- GetAbsHueRange
  - PGRDirectShow.h, [265](#)
- GetAbsPan
  - PGRDirectShow.h, [265](#)
- GetAbsPanRange
  - PGRDirectShow.h, [265](#)
- GetAbsSaturation
  - PGRDirectShow.h, [265](#)
- GetAbsSaturationRange
  - PGRDirectShow.h, [265](#)
- GetAbsSharpness
  - PGRDirectShow.h, [265](#)
- GetAbsSharpnessRange
  - PGRDirectShow.h, [265](#)
- GetAbsShutter
  - PGRDirectShow.h, [265](#)
- GetAbsShutterRange
  - PGRDirectShow.h, [265](#)
- GetAbsTilt
  - PGRDirectShow.h, [265](#)

- GetAbsTiltRange
  - PGRDirectShow.h, [265](#)
- GetAbsWhiteBalance
  - PGRDirectShow.h, [265](#)
- GetAbsWhiteBalanceRange
  - PGRDirectShow.h, [265](#)
- GetActiveLUTBank
  - FlyCapture2::Camera, [66](#)
  - FlyCapture2::CameraBase, [91](#)
  - FlyCapture2::GigECamera, [149](#)
- GetBayerTileFormat
  - FlyCapture2::Image, [187](#)
- GetBitsPerPixel
  - FlyCapture2::Image, [187](#)
- GetBrightness
  - PGRDirectShow.h, [265](#)
- GetBrightnessRange
  - PGRDirectShow.h, [265](#)
- GetBuildDate
  - FlyCapture2::Error, [128](#)
- GetCameraFromIndex
  - FlyCapture2::BusManager, [55](#)
- GetCameraFromIPAddress
  - FlyCapture2::BusManager, [56](#)
- GetCameraFromSerialNumber
  - FlyCapture2::BusManager, [56](#)
- GetCameraInfo
  - FlyCapture2::Camera, [67](#)
  - FlyCapture2::CameraBase, [92](#)
  - FlyCapture2::GigECamera, [149](#)
- GetCameraSerialNumberFromIndex
  - FlyCapture2::BusManager, [56](#)
- GetCause
  - FlyCapture2::Error, [128](#)
- GetChannelStatus
  - FlyCapture2::ImageStatistics, [200](#)
- GetChild
  - FlyCapture2::TopologyNode, [231](#)
- GetColorProcessing
  - FlyCapture2::Image, [187](#)
- GetCols
  - FlyCapture2::Image, [188](#)
- GetConfiguration
  - FlyCapture2::Camera, [67](#)
  - FlyCapture2::CameraBase, [92](#)
  - FlyCapture2::GigECamera, [149](#)
- GetCustomImage
  - PGRDirectShow.h, [265](#)
- GetCustomImageMode
  - PGRDirectShow.h, [266](#)
- GetData
  - FlyCapture2::Image, [188](#)
- GetDataSize
  - FlyCapture2::Image, [188](#)
- GetDefaultColorProcessing
  - FlyCapture2::Image, [188](#)
- GetDefaultOutputFormat
  - FlyCapture2::Image, [188](#)
- GetDescription
  - FlyCapture2::Error, [129](#)
- GetDeviceFromIndex
  - FlyCapture2::BusManager, [57](#)
- GetDeviceId
  - FlyCapture2::TopologyNode, [231](#)
- GetDimensions
  - FlyCapture2::Image, [189](#)
- GetEmbeddedImageInfo
  - FlyCapture2::Camera, [67](#)
  - FlyCapture2::CameraBase, [92](#)
  - FlyCapture2::GigECamera, [150](#)
- GetExposure
  - PGRDirectShow.h, [266](#)
- GetExposureRange
  - PGRDirectShow.h, [266](#)
- GetFilename
  - FlyCapture2::Error, [129](#)
- GetFormat7Configuration
  - FlyCapture2::Camera, [68](#)
- GetFormat7Info
  - FlyCapture2::Camera, [68](#)
- GetGain
  - PGRDirectShow.h, [266](#)
- GetGainRange
  - PGRDirectShow.h, [266](#)
- GetGamma
  - PGRDirectShow.h, [266](#)
- GetGammaRange
  - PGRDirectShow.h, [266](#)
- GetGigEImageBinningSettings
  - FlyCapture2::GigECamera, [150](#)
- GetGigEImageSettings
  - FlyCapture2::GigECamera, [150](#)
- GetGigEImageSettingsInfo
  - FlyCapture2::GigECamera, [150](#)
- GetGigEImagingMode
  - FlyCapture2::GigECamera, [151](#)
- GetGigEProperty
  - FlyCapture2::GigECamera, [151](#)
- GetGigEStreamChannelInfo
  - FlyCapture2::GigECamera, [151](#)
- GetGPIOPinDirection
  - FlyCapture2::Camera, [68](#)
  - FlyCapture2::CameraBase, [93](#)
  - FlyCapture2::GigECamera, [151](#)
- GetGuid
  - FlyCapture2::TopologyNode, [231](#)
- GetHistogram
  - FlyCapture2::ImageStatistics, [201](#)



- GetHue
  - PGRDirectShow.h, 266
- GetHueRange
  - PGRDirectShow.h, 266
- GetImageFormat
  - PGRDirectShow.h, 266
- GetInterfaceType
  - FlyCapture2::TopologyNode, 232
- GetInterfaceTypeFromGuid
  - FlyCapture2::BusManager, 57
- GetLibraryVersion
  - FlyCapture2::Utilities, 237
- GetLine
  - FlyCapture2::Error, 129
- GetLUTBankInfo
  - FlyCapture2::Camera, 69
  - FlyCapture2::CameraBase, 93
  - FlyCapture2::GigECamera, 152
- GetLUTChannel
  - FlyCapture2::Camera, 69
  - FlyCapture2::CameraBase, 93
  - FlyCapture2::GigECamera, 152
- GetLUTInfo
  - FlyCapture2::Camera, 70
  - FlyCapture2::CameraBase, 94
  - FlyCapture2::GigECamera, 153
- GetMean
  - FlyCapture2::ImageStatistics, 201
- GetMemoryChannel
  - FlyCapture2::Camera, 70
  - FlyCapture2::CameraBase, 94
  - FlyCapture2::GigECamera, 153
- GetMemoryChannelInfo
  - FlyCapture2::Camera, 70
  - FlyCapture2::CameraBase, 95
  - FlyCapture2::GigECamera, 153
- GetMetadata
  - FlyCapture2::Image, 189
- GetNodeType
  - FlyCapture2::TopologyNode, 232
- GetNumChildren
  - FlyCapture2::TopologyNode, 232
- GetNumOfCameras
  - FlyCapture2::BusManager, 57
- GetNumOfDevices
  - FlyCapture2::BusManager, 58
- GetNumPixelValues
  - FlyCapture2::ImageStatistics, 201
- GetNumPorts
  - FlyCapture2::TopologyNode, 232
- GetNumStreamChannels
  - FlyCapture2::GigECamera, 154
- GetOutputVerticalFlip
  - PGRDirectShow.h, 266
- GetPan
  - PGRDirectShow.h, 267
- GetPanRange
  - PGRDirectShow.h, 267
- GetPixelFormat
  - FlyCapture2::Image, 189
- GetPixelValueRange
  - FlyCapture2::ImageStatistics, 201
- GetPortType
  - FlyCapture2::TopologyNode, 232
- GetProperty
  - FlyCapture2::Camera, 71
  - FlyCapture2::CameraBase, 95
  - FlyCapture2::GigECamera, 154
- GetPropertyInfo
  - FlyCapture2::Camera, 71
  - FlyCapture2::CameraBase, 95
  - FlyCapture2::GigECamera, 154
- GetRange
  - FlyCapture2::ImageStatistics, 202
- GetRegisterString
  - FlyCapture2::Camera, 72
  - FlyCapture2::CameraBase, 96
  - FlyCapture2::GigECamera, 155
- GetRows
  - FlyCapture2::Image, 189
- GetSaturation
  - PGRDirectShow.h, 267
- GetSaturationRange
  - PGRDirectShow.h, 267
- GetSharpness
  - PGRDirectShow.h, 267
- GetSharpnessRange
  - PGRDirectShow.h, 267
- GetShutter
  - PGRDirectShow.h, 267
- GetShutterRange
  - PGRDirectShow.h, 267
- GetStatistics
  - FlyCapture2::ImageStatistics, 202
- GetStride
  - FlyCapture2::Image, 189
- GetStrobe
  - FlyCapture2::Camera, 72
  - FlyCapture2::CameraBase, 96
  - FlyCapture2::GigECamera, 155
- GetStrobeInfo
  - FlyCapture2::Camera, 72
  - FlyCapture2::CameraBase, 96
  - FlyCapture2::GigECamera, 155
- GetSystemInfo
  - FlyCapture2::Utilities, 237
- GetTilt
  - PGRDirectShow.h, 267



- GetTiltRange
  - PGRDirectShow.h, 267
- GetTimeStamp
  - FlyCapture2::Image, 190
- GetTopology
  - FlyCapture2::BusManager, 58
- GetTriggerDelay
  - FlyCapture2::Camera, 73
  - FlyCapture2::CameraBase, 97
  - FlyCapture2::GigECamera, 156
- GetTriggerDelayInfo
  - FlyCapture2::Camera, 73
  - FlyCapture2::CameraBase, 97
  - FlyCapture2::GigECamera, 156
- GetTriggerMode
  - FlyCapture2::Camera, 73
  - FlyCapture2::CameraBase, 98
  - FlyCapture2::GigECamera, 157
- GetTriggerModeInfo
  - FlyCapture2::Camera, 74
  - FlyCapture2::CameraBase, 98
  - FlyCapture2::GigECamera, 157
- GetType
  - FlyCapture2::Error, 129
- GetVideoModeAndFrameRate
  - FlyCapture2::Camera, 74
- GetVideoModeAndFrameRateInfo
  - FlyCapture2::Camera, 75
- GetWhiteBalance
  - PGRDirectShow.h, 267
- GetWhiteBalanceRange
  - PGRDirectShow.h, 267
- GigE specific enumerations, 28
- GigE specific structures, 32
- GigECamera
  - FlyCapture2::GigECamera, 147
- GigECamera.h, 257
- GigEConfig
  - FlyCapture2::GigEConfig, 171
- GigEEnums
  - GigEPropertyType, 28
  - HEARTBEAT, 28
  - HEARTBEAT\_TIMEOUT, 28
  - PACKET\_DELAY, 28
  - PACKET\_SIZE, 28
- GigEImageSettings
  - FlyCapture2::GigEImageSettings, 172
- GigEImageSettingsInfo
  - FlyCapture2::GigEImageSettingsInfo, 174
- gigEMajorVersion
  - FlyCapture2::CameraInfo, 113
- gigEMinorVersion
  - FlyCapture2::CameraInfo, 113
- GigEPropertyType
  - GigEEnums, 28
- GigEStreamChannel
  - FlyCapture2::GigEStreamChannel, 179
- Global constants, 11
- GlobalConstants
  - sk\_maxNumPorts, 11
  - sk\_maxStringLength, 11
- GPIOPinState
  - FlyCapture2::EmbeddedImageInfo, 125
- gpuDescription
  - FlyCapture2::SystemInfo, 224
- GRAB\_MODE\_FORCE\_32BITS
  - Enumerations, 23
- GRAB\_TIMEOUT\_FORCE\_32BITS
  - Enumerations, 23
- GrabMode
  - Enumerations, 22
- grabMode
  - FlyCapture2::FC2Config, 132
- GrabTimeout
  - Enumerations, 23
- grabTimeout
  - FlyCapture2::FC2Config, 132
- GRBG
  - Enumerations, 19
- GREEN
  - FlyCapture2::ImageStatistics, 199
- GREY
  - FlyCapture2::ImageStatistics, 199
- HEARTBEAT
  - GigEEnums, 28
- HEARTBEAT\_TIMEOUT
  - GigEEnums, 28
- height
  - FlyCapture2::Format7ImageSettings, 135
  - FlyCapture2::GigEImageSettings, 172
  - VideoModes, 239
- Hide
  - FlyCapture2::CameraControlDlg, 110
- HostAdapterStats
  - FlyCapture2::HostAdapterStats, 181
- hostPost
  - FlyCapture2::GigEStreamChannel, 179
- HQ\_LINEAR
  - Enumerations, 20
- HUE
  - Enumerations, 26
  - FlyCapture2::ImageStatistics, 199
- IID\_IFlyCaptureProperties
  - PGRDirectShow.h, 270
- IIDC specific structures, 33
- iidcVer

- FlyCapture2::CameraInfo, 113
- Image
  - FlyCapture2::Image, 185
- Image saving structures., 34
- Image.h, 258
- IMAGE\_FILE\_FORMAT\_FORCE\_32BITS
  - Enumerations, 24
- imageCorrupt
  - FlyCapture2::CameraStats, 118
- imageDriverDropped
  - FlyCapture2::CameraStats, 118
- imageDropped
  - FlyCapture2::CameraStats, 118
- ImageEventCallback
  - FlyCapture2, 46
- ImageFileFormat
  - Enumerations, 23
- imageHStepSize
  - FlyCapture2::Format7Info, 138
  - FlyCapture2::GigEImageSettingsInfo, 174
- ImageMetadata
  - FlyCapture2::ImageMetadata, 196
- ImageStatistics
  - FlyCapture2::ImageStatistics, 200
- ImageStatistics.h, 259
- ImageStatsCalculator
  - FlyCapture2::ImageStatistics, 203
- imageVStepSize
  - FlyCapture2::Format7Info, 138
  - FlyCapture2::GigEImageSettingsInfo, 174
- imageXmitFailed
  - FlyCapture2::CameraStats, 118
- inputBitDepth
  - FlyCapture2::LUTData, 207
- INTERFACE\_GIGE
  - Enumerations, 24
- INTERFACE\_IEEE1394
  - Enumerations, 24
- INTERFACE\_TYPE\_FORCE\_32BITS
  - Enumerations, 24
- INTERFACE\_UNKNOWN
  - Enumerations, 24
- INTERFACE\_USB2
  - Enumerations, 24
- InterfaceType
  - Enumerations, 24
- interfaceType
  - FlyCapture2::CameraInfo, 114
- interlaced
  - FlyCapture2::PNGOption, 212
- InternalError
  - FlyCapture2::Error, 130
- interPacketDelay
  - FlyCapture2::GigEStreamChannel, 179
- IPAddress
  - FlyCapture2::IPAddress, 204
- ipAddress
  - FlyCapture2::CameraInfo, 114
- IPP
  - Enumerations, 20
- IRIS
  - Enumerations, 26
- isColorCamera
  - FlyCapture2::CameraInfo, 114
- IsConnected
  - FlyCapture2::Camera, 75
  - FlyCapture2::CameraBase, 98
  - FlyCapture2::GigECamera, 157
- Iso
  - FlyCapture2::Image, 194
- isochBusSpeed
  - FlyCapture2::FC2Config, 132
- isReadable
  - FlyCapture2::GigEProperty, 176
- IsVisible
  - FlyCapture2::CameraControlDlg, 110
- isWritable
  - FlyCapture2::GigEProperty, 176
- JPEG
  - Enumerations, 23
  - FlyCapture2::TIFFOption, 225
- JPEG2000
  - Enumerations, 23
- JPEGOption
  - FlyCapture2::JPEGOption, 205
- JPG2Option
  - FlyCapture2::JPG2Option, 206
- LaunchBrowser
  - FlyCapture2::Utilities, 237
- LaunchCommand
  - FlyCapture2::Utilities, 238
- LaunchCommandAsync
  - FlyCapture2::Utilities, 238
- LaunchHelp
  - FlyCapture2::Utilities, 238
- libraryList
  - FlyCapture2::SystemInfo, 224
- LIGHTNESS
  - FlyCapture2::ImageStatistics, 199
- LINUX\_X64
  - FlyCapture2, 47
- LINUX\_X86
  - FlyCapture2, 47
- LUTData
  - FlyCapture2::LUTData, 207
- LZW

- FlyCapture2::TIFFOption, 225
- m\_pCameraData
  - FlyCapture2::CameraBase, 108
- MAC
  - FlyCapture2, 47
- MACAddress
  - FlyCapture2::MACAddress, 209
- macAddress
  - FlyCapture2::CameraInfo, 114
- major
  - FlyCapture2::FC2Version, 134
- manualSupported
  - FlyCapture2::PropertyInfo, 217
- max
  - FlyCapture2::GigEProperty, 176
  - FlyCapture2::PropertyInfo, 217
- maxBytesPerPacket
  - FlyCapture2::Format7PacketInfo, 140
- maxHeight
  - FlyCapture2::Format7Info, 138
  - FlyCapture2::GigEImageSettingsInfo, 175
- maximumBusSpeed
  - FlyCapture2::CameraInfo, 114
- maxPacketSize
  - FlyCapture2::Format7Info, 138
- maxValue
  - FlyCapture2::StrobeInfo, 221
- maxWidth
  - FlyCapture2::Format7Info, 138
  - FlyCapture2::GigEImageSettingsInfo, 175
- microSeconds
  - FlyCapture2::TimeStamp, 227
- min
  - FlyCapture2::GigEProperty, 176
  - FlyCapture2::PropertyInfo, 217
- minor
  - FlyCapture2::FC2Version, 134
- minPacketSize
  - FlyCapture2::Format7Info, 138
- minValue
  - FlyCapture2::StrobeInfo, 221
- Mode
  - Enumerations, 24
- mode
  - FlyCapture2::Format7ImageSettings, 135
  - FlyCapture2::Format7Info, 138
  - FlyCapture2::TriggerMode, 233
- MODE\_0
  - Enumerations, 24
- MODE\_1
  - Enumerations, 24
- MODE\_10
  - Enumerations, 24
- MODE\_11
  - Enumerations, 24
- MODE\_12
  - Enumerations, 24
- MODE\_13
  - Enumerations, 24
- MODE\_14
  - Enumerations, 24
- MODE\_15
  - Enumerations, 24
- MODE\_16
  - Enumerations, 24
- MODE\_17
  - Enumerations, 24
- MODE\_18
  - Enumerations, 24
- MODE\_19
  - Enumerations, 24
- MODE\_2
  - Enumerations, 24
- MODE\_20
  - Enumerations, 24
- MODE\_21
  - Enumerations, 24
- MODE\_22
  - Enumerations, 25
- MODE\_23
  - Enumerations, 25
- MODE\_24
  - Enumerations, 25
- MODE\_25
  - Enumerations, 25
- MODE\_26
  - Enumerations, 25
- MODE\_27
  - Enumerations, 25
- MODE\_28
  - Enumerations, 25
- MODE\_29
  - Enumerations, 25
- MODE\_3
  - Enumerations, 24
- MODE\_30
  - Enumerations, 25
- MODE\_31
  - Enumerations, 25
- MODE\_4
  - Enumerations, 24
- MODE\_5
  - Enumerations, 24
- MODE\_6
  - Enumerations, 24
- MODE\_7
  - Enumerations, 24

- MODE\_8
  - Enumerations, [24](#)
- MODE\_9
  - Enumerations, [24](#)
- MODE\_FORCE\_32BITS
  - Enumerations, [25](#)
- modelName
  - FlyCapture2::CameraInfo, [114](#)
- modeMask
  - FlyCapture2::TriggerModeInfo, [235](#)
- NEAREST\_NEIGHBOR
  - Enumerations, [20](#)
- networkInterfaceIndex
  - FlyCapture2::GigEStreamChannel, [179](#)
- NO\_COLOR\_PROCESSING
  - Enumerations, [20](#)
- NODE
  - FlyCapture2::TopologyNode, [230](#)
- NodeType
  - FlyCapture2::TopologyNode, [230](#)
- nodeVendorId
  - FlyCapture2::ConfigROM, [121](#)
- NONE
  - Enumerations, [19](#)
  - FlyCapture2::TIFFOption, [225](#)
- NOT\_CONNECTED
  - FlyCapture2::TopologyNode, [230](#)
- NULL
  - FlyCapture2Defs.h, [254](#)
- NUM\_FRAMERATES
  - Enumerations, [22](#)
- NUM\_MODES
  - Enumerations, [25](#)
- NUM\_PIXEL\_FORMATS
  - Enumerations, [25](#)
- NUM\_STATISTICS\_CHANNELS
  - FlyCapture2::ImageStatistics, [199](#)
- NUM\_VIDEOMODES
  - Enumerations, [27](#)
- numBanks
  - FlyCapture2::LUTData, [207](#)
- numBuffers
  - FlyCapture2::FC2Config, [132](#)
- numChannels
  - FlyCapture2::GigEConfig, [171](#)
  - FlyCapture2::LUTData, [208](#)
- numCpuCores
  - FlyCapture2::SystemInfo, [224](#)
- numCurrents
  - FlyCapture2::CameraStats, [118](#)
- numEntries
  - FlyCapture2::LUTData, [208](#)
- numFormats
  - DCAMFormats, [123](#)
- numImageNotifications
  - FlyCapture2::FC2Config, [132](#)
- numPorts
  - FlyCapture2::HostAdapterStats, [181](#)
- numVoltages
  - FlyCapture2::CameraStats, [118](#)
- octets
  - FlyCapture2::IPAddress, [204](#)
  - FlyCapture2::MACAddress, [209](#)
- offsetHStepSize
  - FlyCapture2::Format7Info, [138](#)
  - FlyCapture2::GigEImageSettingsInfo, [175](#)
- offsetVStepSize
  - FlyCapture2::Format7Info, [138](#)
  - FlyCapture2::GigEImageSettingsInfo, [175](#)
- offsetX
  - FlyCapture2::Format7ImageSettings, [136](#)
  - FlyCapture2::GigEImageSettings, [172](#)
- offsetY
  - FlyCapture2::Format7ImageSettings, [136](#)
  - FlyCapture2::GigEImageSettings, [172](#)
- onePush
  - FlyCapture2::Property, [215](#)
- onePushSupported
  - FlyCapture2::PropertyInfo, [217](#)
- onOff
  - FlyCapture2::EmbeddedImageInfoProperty, [126](#)
  - FlyCapture2::Property, [215](#)
  - FlyCapture2::StrobeControl, [219](#)
  - FlyCapture2::TriggerMode, [233](#)
- onOffSupported
  - FlyCapture2::PropertyInfo, [217](#)
  - FlyCapture2::StrobeInfo, [221](#)
  - FlyCapture2::TriggerModeInfo, [235](#)
- operator()
  - FlyCapture2::Image, [190](#)
- operator=
  - FlyCapture2::Error, [129](#)
  - FlyCapture2::Image, [190](#)
  - FlyCapture2::ImageStatistics, [202](#)
  - FlyCapture2::TopologyNode, [232](#)
- operator==
  - FlyCapture2::Error, [130](#)
  - FlyCapture2::IPAddress, [204](#)
  - FlyCapture2::MACAddress, [209](#)
  - FlyCapture2::PGRGuid, [211](#)
- osDescription
  - FlyCapture2::SystemInfo, [224](#)
- OSType
  - FlyCapture2, [46](#)
- osType
  - FlyCapture2::SystemInfo, [224](#)

- OSTYPE\_FORCE\_32BITS
  - FlyCapture2, [47](#)
- outputBitDepth
  - FlyCapture2::LUTData, [208](#)
- PACKBITS
  - FlyCapture2::TIFFOption, [225](#)
- PACKET\_DELAY
  - GigEEnums, [28](#)
- PACKET\_SIZE
  - GigEEnums, [28](#)
- packetSize
  - FlyCapture2::Format7Info, [138](#)
  - FlyCapture2::GigEStreamChannel, [179](#)
- PAN
  - Enumerations, [26](#)
- parameter
  - FlyCapture2::TriggerMode, [233](#)
- percentage
  - FlyCapture2::Format7Info, [139](#)
- PGM
  - Enumerations, [23](#)
- PGMOption
  - FlyCapture2::PGMOption, [210](#)
- PGRDirectShow.h, [260](#)
  - GetAbsBrightness, [265](#)
  - GetAbsBrightnessRange, [265](#)
  - GetAbsExposure, [265](#)
  - GetAbsExposureRange, [265](#)
  - GetAbsGain, [265](#)
  - GetAbsGainRange, [265](#)
  - GetAbsGamma, [265](#)
  - GetAbsGammaRange, [265](#)
  - GetAbsHue, [265](#)
  - GetAbsHueRange, [265](#)
  - GetAbsPan, [265](#)
  - GetAbsPanRange, [265](#)
  - GetAbsSaturation, [265](#)
  - GetAbsSaturationRange, [265](#)
  - GetAbsSharpness, [265](#)
  - GetAbsSharpnessRange, [265](#)
  - GetAbsShutter, [265](#)
  - GetAbsShutterRange, [265](#)
  - GetAbsTilt, [265](#)
  - GetAbsTiltRange, [265](#)
  - GetAbsWhiteBalance, [265](#)
  - GetAbsWhiteBalanceRange, [265](#)
  - GetBrightness, [265](#)
  - GetBrightnessRange, [265](#)
  - GetCustomImage, [265](#)
  - GetCustomImageMode, [266](#)
  - GetExposure, [266](#)
  - GetExposureRange, [266](#)
  - GetGain, [266](#)
  - GetGainRange, [266](#)
  - GetGamma, [266](#)
  - GetGammaRange, [266](#)
  - GetHue, [266](#)
  - GetHueRange, [266](#)
  - GetImageFormat, [266](#)
  - GetOutputVerticalFlip, [266](#)
  - GetPan, [267](#)
  - GetPanRange, [267](#)
  - GetSaturation, [267](#)
  - GetSaturationRange, [267](#)
  - GetSharpness, [267](#)
  - GetSharpnessRange, [267](#)
  - GetShutter, [267](#)
  - GetShutterRange, [267](#)
  - GetTilt, [267](#)
  - GetTiltRange, [267](#)
  - GetWhiteBalance, [267](#)
  - GetWhiteBalanceRange, [267](#)
  - IID\_IFlyCaptureProperties, [270](#)
  - QueryCustomImage, [267](#)
  - ReadRegister, [267](#)
  - SetAbsBrightness, [268](#)
  - SetAbsExposure, [268](#)
  - SetAbsGain, [268](#)
  - SetAbsGamma, [268](#)
  - SetAbsHue, [268](#)
  - SetAbsPan, [268](#)
  - SetAbsSaturation, [268](#)
  - SetAbsSharpness, [268](#)
  - SetAbsShutter, [268](#)
  - SetAbsTilt, [268](#)
  - SetAbsWhiteBalance, [268](#)
  - SetBrightness, [268](#)
  - SetCustomImage, [268](#)
  - SetCustomImageMode, [268](#)
  - SetExposure, [269](#)
  - SetGain, [269](#)
  - SetGamma, [269](#)
  - SetHue, [269](#)
  - SetImageFormat, [269](#)
  - SetOutputVerticalFlip, [269](#)
  - SetPan, [269](#)
  - SetSaturation, [270](#)
  - SetSharpness, [270](#)
  - SetShutter, [270](#)
  - SetTilt, [270](#)
  - SetWhiteBalance, [270](#)
  - WriteRegister, [270](#)
- PGRERROR\_BUFFER\_TOO\_SMALL
  - Enumerations, [22](#)
- PGRERROR\_BUS\_MASTER\_FAILED
  - Enumerations, [21](#)
- PGRERROR\_FAILED

- Enumerations, [21](#)
- PGRERROR\_FAILED\_BUS\_MASTER\_CONNECTION
  - Enumerations, [21](#)
- PGRERROR\_FAILED\_GUID
  - Enumerations, [21](#)
- PGRERROR\_FORCE\_32BITS
  - Enumerations, [22](#)
- PGRERROR\_IIDC\_FAILED
  - Enumerations, [21](#)
- PGRERROR\_IMAGE\_CONSISTENCY\_ERROR
  - Enumerations, [22](#)
- PGRERROR\_IMAGE\_CONVERSION\_FAILED
  - Enumerations, [22](#)
- PGRERROR\_IMAGE\_LIBRARY\_FAILURE
  - Enumerations, [22](#)
- PGRERROR\_INIT\_FAILED
  - Enumerations, [21](#)
- PGRERROR\_INVALID\_BUS\_MANAGER
  - Enumerations, [21](#)
- PGRERROR\_INVALID\_GENERATION
  - Enumerations, [21](#)
- PGRERROR\_INVALID\_MODE
  - Enumerations, [21](#)
- PGRERROR\_INVALID\_PACKET\_SIZE
  - Enumerations, [21](#)
- PGRERROR\_INVALID\_PARAMETER
  - Enumerations, [21](#)
- PGRERROR\_INVALID\_SETTINGS
  - Enumerations, [21](#)
- PGRERROR\_ISOCH\_ALREADY\_STARTED
  - Enumerations, [22](#)
- PGRERROR\_ISOCH\_BANDWIDTH\_EXCEEDED
  - Enumerations, [22](#)
- PGRERROR\_ISOCH\_FAILED
  - Enumerations, [22](#)
- PGRERROR\_ISOCH\_NOT\_STARTED
  - Enumerations, [22](#)
- PGRERROR\_ISOCH\_RETRIEVE\_BUFFER\_FAILED
  - Enumerations, [22](#)
- PGRERROR\_ISOCH\_START\_FAILED
  - Enumerations, [22](#)
- PGRERROR\_ISOCH\_STOP\_FAILED
  - Enumerations, [22](#)
- PGRERROR\_ISOCH\_SYNC\_FAILED
  - Enumerations, [22](#)
- PGRERROR\_LOW\_LEVEL\_FAILURE
  - Enumerations, [21](#)
- PGRERROR\_LUT\_FAILED
  - Enumerations, [21](#)
- PGRERROR\_MEMORY\_ALLOCATION\_FAILED
  - Enumerations, [21](#)
- PGRERROR\_NOT\_CONNECTED
  - Enumerations, [21](#)
- PGRERROR\_NOT\_FOUND
  - Enumerations, [21](#)
- PGRERROR\_NOT\_IMPLEMENTED
  - Enumerations, [21](#)
- PGRERROR\_NOT\_IN\_FORMAT7
  - Enumerations, [21](#)
- PGRERROR\_NOT\_INITIALIZED
  - Enumerations, [21](#)
- PGRERROR\_NOT\_SUPPORTED
  - Enumerations, [21](#)
- PGRERROR\_OK
  - Enumerations, [21](#)
- PGRERROR\_PROPERTY\_FAILED
  - Enumerations, [21](#)
- PGRERROR\_PROPERTY\_NOT\_PRESENT
  - Enumerations, [22](#)
- PGRERROR\_READ\_REGISTER\_FAILED
  - Enumerations, [22](#)
- PGRERROR\_REGISTER\_FAILED
  - Enumerations, [22](#)
- PGRERROR\_STROBE\_FAILED
  - Enumerations, [21](#)
- PGRERROR\_TIMEOUT
  - Enumerations, [21](#)
- PGRERROR\_TRIGGER\_FAILED
  - Enumerations, [21](#)
- PGRERROR\_UNDEFINED
  - Enumerations, [21](#)
- PGRERROR\_WRITE\_REGISTER\_FAILED
  - Enumerations, [22](#)
- PGRGuid
  - FlyCapture2::PGRGuid, [211](#)
- PIXEL\_FORMAT\_411YUV8
  - Enumerations, [25](#)
- PIXEL\_FORMAT\_422YUV8
  - Enumerations, [25](#)
- PIXEL\_FORMAT\_444YUV8
  - Enumerations, [25](#)
- PIXEL\_FORMAT\_BGR
  - Enumerations, [25](#)
- PIXEL\_FORMAT\_BGRU
  - Enumerations, [25](#)
- PIXEL\_FORMAT\_MONO12
  - Enumerations, [25](#)
- PIXEL\_FORMAT\_MONO16
  - Enumerations, [25](#)
- PIXEL\_FORMAT\_MONO8
  - Enumerations, [25](#)
- PIXEL\_FORMAT\_RAW12
  - Enumerations, [25](#)
- PIXEL\_FORMAT\_RAW16
  - Enumerations, [25](#)
- PIXEL\_FORMAT\_RAW8
  - Enumerations, [25](#)
- PIXEL\_FORMAT\_RGB
  - Enumerations, [25](#)



- Enumerations, [25](#)
- PIXEL\_FORMAT\_RGB16
  - Enumerations, [25](#)
- PIXEL\_FORMAT\_RGB8
  - Enumerations, [25](#)
- PIXEL\_FORMAT\_RGBA
  - Enumerations, [25](#)
- PIXEL\_FORMAT\_S\_MONO16
  - Enumerations, [25](#)
- PIXEL\_FORMAT\_S\_RGB16
  - Enumerations, [25](#)
- PixelFormat
  - Enumerations, [25](#)
- pixelFormat
  - FlyCapture2::Format7ImageSettings, [136](#)
  - FlyCapture2::GigEImageSettings, [173](#)
- pixelFormatBitField
  - FlyCapture2::Format7Info, [139](#)
  - FlyCapture2::GigEImageSettingsInfo, [175](#)
- PNG
  - Enumerations, [23](#)
- PNGOption
  - FlyCapture2::PNGOption, [212](#)
- polarity
  - FlyCapture2::StrobeControl, [219](#)
  - FlyCapture2::TriggerMode, [233](#)
- polaritySupported
  - FlyCapture2::StrobeInfo, [222](#)
  - FlyCapture2::TriggerModeInfo, [235](#)
- portErrors
  - FlyCapture2::CameraStats, [118](#)
  - FlyCapture2::HostAdapterStats, [181](#)
- PortType
  - FlyCapture2::TopologyNode, [230](#)
- PPM
  - Enumerations, [23](#)
- PPMOption
  - FlyCapture2::PPMOption, [213](#)
- present
  - FlyCapture2::Property, [215](#)
  - FlyCapture2::PropertyInfo, [217](#)
  - FlyCapture2::StrobeInfo, [222](#)
  - FlyCapture2::TriggerModeInfo, [235](#)
- PrintErrorTrace
  - FlyCapture2::Error, [130](#)
- progressive
  - FlyCapture2::JPEGOption, [205](#)
- Property
  - FlyCapture2::Property, [215](#)
- PROPERTY\_TYPE\_FORCE\_32BITS
  - Enumerations, [26](#)
- PropertyInfo
  - FlyCapture2::PropertyInfo, [217](#)
- PropertyType
  - Enumerations, [25](#)
- propType
  - FlyCapture2::GigEProperty, [176](#)
- pszKeyword
  - FlyCapture2::ConfigROM, [121](#)
- pUnitAbbr
  - FlyCapture2::PropertyInfo, [218](#)
- pUnits
  - FlyCapture2::PropertyInfo, [218](#)
- quality
  - FlyCapture2::JPEGOption, [205](#)
  - FlyCapture2::JPG2Option, [206](#)
- QueryCustomImage
  - PGRDirectShow.h, [267](#)
- QueryGigEImagingMode
  - FlyCapture2::GigECamera, [158](#)
- RAW
  - Enumerations, [24](#)
- ReadGVCPMemory
  - FlyCapture2::GigECamera, [158](#)
- ReadGVCPRegister
  - FlyCapture2::GigECamera, [158](#)
- ReadGVCPRegisterBlock
  - FlyCapture2::GigECamera, [158](#)
- readOutSupported
  - FlyCapture2::PropertyInfo, [218](#)
  - FlyCapture2::StrobeInfo, [222](#)
  - FlyCapture2::TriggerModeInfo, [236](#)
- ReadPhyRegister
  - FlyCapture2::BusManager, [58](#)
- ReadRegister
  - FlyCapture2::Camera, [75](#)
  - FlyCapture2::CameraBase, [99](#)
  - FlyCapture2::GigECamera, [159](#)
  - PGRDirectShow.h, [267](#)
- ReadRegisterBlock
  - FlyCapture2::Camera, [76](#)
  - FlyCapture2::CameraBase, [99](#)
  - FlyCapture2::GigECamera, [159](#)
- recommendedBytesPerPacket
  - FlyCapture2::Format7PacketInfo, [140](#)
- RED
  - FlyCapture2::ImageStatistics, [199](#)
- RegisterCallback
  - FlyCapture2::BusManager, [58](#)
- regReadFailed
  - FlyCapture2::CameraStats, [118](#)
- regWriteFailed
  - FlyCapture2::CameraStats, [118](#)
- ReleaseBuffer
  - FlyCapture2::Image, [190](#)
- REMOVAL

- Enumerations, [19](#)
- RescanBus
  - FlyCapture2::BusManager, [59](#)
- reserved
  - FlyCapture2::AVIOption, [49](#)
  - FlyCapture2::CameraInfo, [114](#)
  - FlyCapture2::CameraStats, [118](#)
  - FlyCapture2::ConfigROM, [121](#)
  - FlyCapture2::FC2Config, [132](#)
  - FlyCapture2::Format7ImageSettings, [136](#)
  - FlyCapture2::Format7Info, [139](#)
  - FlyCapture2::Format7PacketInfo, [140](#)
  - FlyCapture2::GigEImageSettings, [173](#)
  - FlyCapture2::GigEImageSettingsInfo, [175](#)
  - FlyCapture2::ImageMetadata, [196](#)
  - FlyCapture2::JPEGOption, [205](#)
  - FlyCapture2::JPG2Option, [206](#)
  - FlyCapture2::LUTData, [208](#)
  - FlyCapture2::PGMOption, [210](#)
  - FlyCapture2::PNGOption, [212](#)
  - FlyCapture2::PPMOption, [213](#)
  - FlyCapture2::Property, [215](#)
  - FlyCapture2::PropertyInfo, [218](#)
  - FlyCapture2::StrobeControl, [220](#)
  - FlyCapture2::StrobeInfo, [222](#)
  - FlyCapture2::SystemInfo, [224](#)
  - FlyCapture2::TIFFOption, [226](#)
  - FlyCapture2::TimeStamp, [227](#)
  - FlyCapture2::TriggerMode, [233](#)
  - FlyCapture2::TriggerModeInfo, [236](#)
- RestoreFromMemoryChannel
  - FlyCapture2::Camera, [76](#)
  - FlyCapture2::CameraBase, [99](#)
  - FlyCapture2::GigECamera, [159](#)
- RetrieveBuffer
  - FlyCapture2::Camera, [76](#)
  - FlyCapture2::CameraBase, [100](#)
  - FlyCapture2::GigECamera, [160](#)
- RGBA
  - Enumerations, [19](#)
- RIGOROUS
  - Enumerations, [20](#)
- ROIPosition
  - FlyCapture2::EmbeddedImageInfo, [125](#)
- SATURATION
  - Enumerations, [26](#)
  - FlyCapture2::ImageStatistics, [199](#)
- Save
  - FlyCapture2::Image, [191](#), [192](#)
- SaveToMemoryChannel
  - FlyCapture2::Camera, [77](#)
  - FlyCapture2::CameraBase, [100](#)
  - FlyCapture2::GigECamera, [160](#)
- screenHeight
  - FlyCapture2::SystemInfo, [224](#)
- screenWidth
  - FlyCapture2::SystemInfo, [224](#)
- seconds
  - FlyCapture2::TimeStamp, [228](#)
- sensorInfo
  - FlyCapture2::CameraInfo, [114](#)
- sensorResolution
  - FlyCapture2::CameraInfo, [114](#)
- serialNumber
  - FlyCapture2::CameraInfo, [114](#)
- SetAbsBrightness
  - PGRDirectShow.h, [268](#)
- SetAbsExposure
  - PGRDirectShow.h, [268](#)
- SetAbsGain
  - PGRDirectShow.h, [268](#)
- SetAbsGamma
  - PGRDirectShow.h, [268](#)
- SetAbsHue
  - PGRDirectShow.h, [268](#)
- SetAbsPan
  - PGRDirectShow.h, [268](#)
- SetAbsSaturation
  - PGRDirectShow.h, [268](#)
- SetAbsSharpness
  - PGRDirectShow.h, [268](#)
- SetAbsShutter
  - PGRDirectShow.h, [268](#)
- SetAbsTilt
  - PGRDirectShow.h, [268](#)
- SetAbsWhiteBalance
  - PGRDirectShow.h, [268](#)
- SetActiveLUTBank
  - FlyCapture2::Camera, [77](#)
  - FlyCapture2::CameraBase, [101](#)
  - FlyCapture2::GigECamera, [161](#)
- SetBrightness
  - PGRDirectShow.h, [268](#)
- SetCallback
  - FlyCapture2::Camera, [77](#)
  - FlyCapture2::CameraBase, [101](#)
  - FlyCapture2::GigECamera, [161](#)
- SetChannelStatus
  - FlyCapture2::ImageStatistics, [203](#)
- SetColorProcessing
  - FlyCapture2::Image, [192](#)
- SetConfiguration
  - FlyCapture2::Camera, [78](#)
  - FlyCapture2::CameraBase, [101](#)
  - FlyCapture2::GigECamera, [161](#)
- SetCustomImage
  - PGRDirectShow.h, [268](#)



- SetCustomImageMode
  - PGRDirectShow.h, 268
- SetData
  - FlyCapture2::Image, 193
- SetDefaultColorProcessing
  - FlyCapture2::Image, 193
- SetDefaultOutputFormat
  - FlyCapture2::Image, 193
- SetDimensions
  - FlyCapture2::Image, 194
- SetEmbeddedImageInfo
  - FlyCapture2::Camera, 78
  - FlyCapture2::CameraBase, 101
  - FlyCapture2::GigECamera, 161
- SetExposure
  - PGRDirectShow.h, 269
- SetFormat7Configuration
  - FlyCapture2::Camera, 78, 79
- SetGain
  - PGRDirectShow.h, 269
- SetGamma
  - PGRDirectShow.h, 269
- SetGigEImageBinningSettings
  - FlyCapture2::GigECamera, 162
- SetGigEImageSettings
  - FlyCapture2::GigECamera, 162
- SetGigEImagingMode
  - FlyCapture2::GigECamera, 162
- SetGigEProperty
  - FlyCapture2::GigECamera, 163
- SetGigEStreamChannelInfo
  - FlyCapture2::GigECamera, 163
- SetGPIOPinDirection
  - FlyCapture2::Camera, 79
  - FlyCapture2::CameraBase, 102
  - FlyCapture2::GigECamera, 163
- SetHue
  - PGRDirectShow.h, 269
- SetImageFormat
  - PGRDirectShow.h, 269
- SetLUTChannel
  - FlyCapture2::Camera, 80
  - FlyCapture2::CameraBase, 102
  - FlyCapture2::GigECamera, 164
- SetOutputVerticalFlip
  - PGRDirectShow.h, 269
- SetPan
  - PGRDirectShow.h, 269
- SetProperty
  - FlyCapture2::Camera, 80
  - FlyCapture2::CameraBase, 103
  - FlyCapture2::GigECamera, 164
- SetSaturation
  - PGRDirectShow.h, 270
- SetSharpness
  - PGRDirectShow.h, 270
- SetShutter
  - PGRDirectShow.h, 270
- SetStrobe
  - FlyCapture2::Camera, 81
  - FlyCapture2::CameraBase, 103
  - FlyCapture2::GigECamera, 164
- SetTilt
  - PGRDirectShow.h, 270
- SetTriggerDelay
  - FlyCapture2::Camera, 81
  - FlyCapture2::CameraBase, 104
  - FlyCapture2::GigECamera, 165
- SetTriggerMode
  - FlyCapture2::Camera, 81
  - FlyCapture2::CameraBase, 104
  - FlyCapture2::GigECamera, 165
- SetUserBuffers
  - FlyCapture2::Camera, 82
  - FlyCapture2::CameraBase, 104
  - FlyCapture2::GigECamera, 166
- SetVideoModeAndFrameRate
  - FlyCapture2::Camera, 82
- SetWhiteBalance
  - PGRDirectShow.h, 270
- SHARPNESSEnumerations, 26
- Show
  - FlyCapture2::CameraControlDlg, 110
- ShowModal
  - FlyCapture2::CameraSelectionDlg, 116
- SHUTTEREnumerations, 26
- shutter
  - FlyCapture2::EmbeddedImageInfo, 125
- sk\_maxNumPorts
  - GlobalConstants, 11
- sk\_maxStringLength
  - GlobalConstants, 11
- softwareTriggerSupported
  - FlyCapture2::TriggerModeInfo, 236
- source
  - FlyCapture2::StrobeControl, 220
  - FlyCapture2::StrobeInfo, 222
  - FlyCapture2::TriggerMode, 233
- sourceMask
  - FlyCapture2::TriggerModeInfo, 236
- sourcePort
  - FlyCapture2::GigEStreamChannel, 179
- StartCapture
  - FlyCapture2::Camera, 83
  - FlyCapture2::CameraBase, 105
  - FlyCapture2::GigECamera, 166

- StartSyncCapture
  - FlyCapture2::Camera, 83
  - FlyCapture2::CameraBase, 105
  - FlyCapture2::GigECamera, 167
- StatisticsChannel
  - FlyCapture2::ImageStatistics, 199
- StopCapture
  - FlyCapture2::Camera, 83
  - FlyCapture2::CameraBase, 106
  - FlyCapture2::GigECamera, 167
- StrobeControl
  - FlyCapture2::StrobeControl, 219
- StrobeInfo
  - FlyCapture2::StrobeInfo, 221
- strobePattern
  - FlyCapture2::EmbeddedImageInfo, 125
- Structures, 29
  - TriggerDelay, 31
  - TriggerDelayInfo, 31
- subnetMask
  - FlyCapture2::CameraInfo, 114
- supported
  - FlyCapture2::LUTData, 208
- sysMemSize
  - FlyCapture2::SystemInfo, 224
- TEMPERATURE
  - Enumerations, 26
- temperature
  - FlyCapture2::CameraStats, 118
- TIFF
  - Enumerations, 23
- TIFFOption
  - FlyCapture2::TIFFOption, 226
- TILT
  - Enumerations, 26
- TIMEOUT\_INFINITE
  - Enumerations, 23
- TIMEOUT\_NONE
  - Enumerations, 23
- TIMEOUT\_UNSPECIFIED
  - Enumerations, 23
- timeSinceBusReset
  - FlyCapture2::CameraStats, 119
- timeSinceInitialization
  - FlyCapture2::CameraStats, 119
- TimeStamp
  - FlyCapture2::TimeStamp, 227
- timeStamp
  - FlyCapture2::CameraStats, 119
- timestamp
  - FlyCapture2::EmbeddedImageInfo, 125
- TopologyNode
  - FlyCapture2::TopologyNode, 230
- TopologyNode.h, 271
- TRIGGER\_DELAY
  - Enumerations, 26
- TRIGGER\_MODE
  - Enumerations, 26
- TriggerDelay
  - Structures, 31
- TriggerDelayInfo
  - Structures, 31
- TriggerMode
  - FlyCapture2::TriggerMode, 233
- TriggerModeInfo
  - FlyCapture2::TriggerModeInfo, 235
- type
  - FlyCapture2::FC2Version, 134
  - FlyCapture2::Property, 215
  - FlyCapture2::PropertyInfo, 218
- unitBytesPerPacket
  - FlyCapture2::Format7PacketInfo, 140
- unitSpecId
  - FlyCapture2::ConfigROM, 121
- unitSubSWVer
  - FlyCapture2::ConfigROM, 121
- unitSWVer
  - FlyCapture2::ConfigROM, 121
- UNKNOWN\_OS
  - FlyCapture2, 47
- UnregisterCallback
  - FlyCapture2::BusManager, 59
- UNSPECIFIED\_GRAB\_MODE
  - Enumerations, 23
- UNSPECIFIED\_PIXEL\_FORMAT
  - Enumerations, 25
- UNSPECIFIED\_PROPERTY\_TYPE
  - Enumerations, 26
- userDefinedName
  - FlyCapture2::CameraInfo, 115
- Utilities.h, 272
- ValidateFormat7Settings
  - FlyCapture2::Camera, 83
- value
  - FlyCapture2::GigEProperty, 177
  - FlyCapture2::PGRGuid, 211
- valueA
  - FlyCapture2::Property, 215
- valueB
  - FlyCapture2::Property, 215
- valueReadable
  - FlyCapture2::TriggerModeInfo, 236
- vendor
  - FlyCapture2::HostAdapterStats, 181
- vendorName

- FlyCapture2::CameraInfo, 115
- vendorUniqueInfo\_0
  - FlyCapture2::ConfigROM, 121
- vendorUniqueInfo\_1
  - FlyCapture2::ConfigROM, 121
- vendorUniqueInfo\_2
  - FlyCapture2::ConfigROM, 122
- vendorUniqueInfo\_3
  - FlyCapture2::ConfigROM, 122
- VideoMode
  - Enumerations, 26
- videoMode
  - VideoModes, 239
- VIDEOMODE\_1024x768RGB
  - Enumerations, 27
- VIDEOMODE\_1024x768Y16
  - Enumerations, 27
- VIDEOMODE\_1024x768Y8
  - Enumerations, 27
- VIDEOMODE\_1024x768YUV422
  - Enumerations, 27
- VIDEOMODE\_1280x960RGB
  - Enumerations, 27
- VIDEOMODE\_1280x960Y16
  - Enumerations, 27
- VIDEOMODE\_1280x960Y8
  - Enumerations, 27
- VIDEOMODE\_1280x960YUV422
  - Enumerations, 27
- VIDEOMODE\_1600x1200RGB
  - Enumerations, 27
- VIDEOMODE\_1600x1200Y16
  - Enumerations, 27
- VIDEOMODE\_1600x1200Y8
  - Enumerations, 27
- VIDEOMODE\_1600x1200YUV422
  - Enumerations, 27
- VIDEOMODE\_160x120YUV444
  - Enumerations, 26
- VIDEOMODE\_320x240YUV422
  - Enumerations, 26
- VIDEOMODE\_640x480RGB
  - Enumerations, 26
- VIDEOMODE\_640x480Y16
  - Enumerations, 26
- VIDEOMODE\_640x480Y8
  - Enumerations, 26
- VIDEOMODE\_640x480YUV411
  - Enumerations, 26
- VIDEOMODE\_640x480YUV422
  - Enumerations, 26
- VIDEOMODE\_800x600RGB
  - Enumerations, 26
- VIDEOMODE\_800x600Y16
  - Enumerations, 26
- VIDEOMODE\_800x600Y8
  - Enumerations, 26
- VIDEOMODE\_800x600YUV422
  - Enumerations, 26
- VIDEOMODE\_FORCE\_32BITS
  - Enumerations, 27
- VIDEOMODE\_FORMAT7
  - Enumerations, 27
- VideoModes, 239
  - frameRate, 239
  - height, 239
  - videoMode, 239
  - width, 239
- videoModes
  - DCAMFormats, 123
- WaitForBufferEvent
  - FlyCapture2::Camera, 84
  - FlyCapture2::CameraBase, 106
  - FlyCapture2::GigECamera, 167
- WHITE\_BALANCE
  - Enumerations, 26
- whiteBalance
  - FlyCapture2::EmbeddedImageInfo, 125
- width
  - FlyCapture2::Format7ImageSettings, 136
  - FlyCapture2::GigEImageSettings, 173
  - VideoModes, 239
- WINDOWS\_X64
  - FlyCapture2, 47
- WINDOWS\_X86
  - FlyCapture2, 47
- WriteGVCPMemory
  - FlyCapture2::GigECamera, 167
- WriteGVCPRegister
  - FlyCapture2::GigECamera, 168
- WriteGVCPRegisterBlock
  - FlyCapture2::GigECamera, 168
- WritePhyRegister
  - FlyCapture2::BusManager, 59
- WriteRegister
  - FlyCapture2::Camera, 84
  - FlyCapture2::CameraBase, 107
  - FlyCapture2::GigECamera, 168
  - PGRDirectShow.h, 270
- WriteRegisterBlock
  - FlyCapture2::Camera, 85
  - FlyCapture2::CameraBase, 107
  - FlyCapture2::GigECamera, 169
- xmlURL1
  - FlyCapture2::CameraInfo, 115
- xmlURL2

FlyCapture2::CameraInfo, [115](#)

ZOOM

Enumerations, [26](#)