



POINT GREY

FlyCapture 2.1

API Programming Reference

Revised September 16, 2010

Point Grey Research Inc.

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

Copyright © 2010 Point Grey Research Inc. All Rights Reserved.

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[®] 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[®], Firefly[®]2, Firefly[®] MV, Flea[®], Scorpion[™], Dragonfly[®], Dragonfly[®]2, Dragonfly Express[™], Grasshopper[™] or Chameleon[™] 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

1	Module Index	1
1.1	Modules	1
2	Namespace Index	3
2.1	Namespace List	3
3	Class Index	5
3.1	Class Hierarchy	5
4	Class Index	7
4.1	Class List	7
5	File Index	9
5.1	File List	9
6	Module Documentation	11
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	18
6.2.1.3	BusCallbackType	19
6.2.1.4	BusSpeed	19
6.2.1.5	ColorProcessingAlgorithm	19
6.2.1.6	ErrorType	20
6.2.1.7	FrameRate	21
6.2.1.8	GrabMode	22

6.2.1.9	GrabTimeout	22
6.2.1.10	ImageFileFormat	22
6.2.1.11	InterfaceType	23
6.2.1.12	Mode	23
6.2.1.13	PixelFormat	24
6.2.1.14	PropertyType	25
6.2.1.15	VideoMode	25
6.3	GigE specific enumerations	27
6.3.1	Detailed Description	27
6.3.2	Enumeration Type Documentation	27
6.3.2.1	GigEPropertyType	27
6.4	Structures	28
6.4.1	Typedef Documentation	30
6.4.1.1	TriggerDelay	30
6.4.1.2	TriggerDelayInfo	30
6.5	GigE specific structures	31
6.5.1	Detailed Description	31
6.6	IIDC specific structures	32
6.6.1	Detailed Description	32
6.7	Image saving structures.	33
6.7.1	Detailed Description	33
7	Namespace Documentation	35
7.1	FlyCapture2 Namespace Reference	35
7.1.1	Typedef Documentation	45
7.1.1.1	AsyncCommandCallback	45
7.1.1.2	BusEventCallback	45
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	46
8	Class Documentation	47
8.1	AVIOption Struct Reference	47
8.1.1	Detailed Description	47
8.1.2	Constructor & Destructor Documentation	47

8.1.2.1	AVIOption	47
8.1.3	Member Data Documentation	47
8.1.3.1	frameRate	47
8.1.3.2	reserved	48
8.2	AVIRecorder Class Reference	49
8.2.1	Detailed Description	49
8.2.2	Constructor & Destructor Documentation	49
8.2.2.1	AVIRecorder	49
8.2.2.2	~AVIRecorder	49
8.2.3	Member Function Documentation	49
8.2.3.1	AVIAppend	49
8.2.3.2	AVIClose	50
8.2.3.3	AVIOpen	50
8.3	BusManager Class Reference	51
8.3.1	Detailed Description	52
8.3.2	Constructor & Destructor Documentation	52
8.3.2.1	BusManager	52
8.3.2.2	~BusManager	52
8.3.3	Member Function Documentation	52
8.3.3.1	DiscoverGigECameras	52
8.3.3.2	FireBusReset	53
8.3.3.3	ForceIPAddressToCamera	53
8.3.3.4	GetCameraFromIndex	54
8.3.3.5	GetCameraFromIPAddress	54
8.3.3.6	GetCameraFromSerialNumber	54
8.3.3.7	GetCameraSerialNumberFromIndex	55
8.3.3.8	GetDeviceFromIndex	55
8.3.3.9	GetInterfaceTypeFromGuid	55
8.3.3.10	GetNumOfCameras	56
8.3.3.11	GetNumOfDevices	56
8.3.3.12	GetTopology	56
8.3.3.13	ReadPhyRegister	56
8.3.3.14	RegisterCallback	57
8.3.3.15	RescanBus	57
8.3.3.16	UnregisterCallback	57
8.3.3.17	WritePhyRegister	57

8.4	Camera Class Reference	59
8.4.1	Detailed Description	63
8.4.2	Constructor & Destructor Documentation	63
8.4.2.1	Camera	63
8.4.2.2	~Camera	63
8.4.3	Member Function Documentation	64
8.4.3.1	Connect	64
8.4.3.2	Disconnect	64
8.4.3.3	EnableLUT	64
8.4.3.4	FireSoftwareTrigger	64
8.4.3.5	GetActiveLUTBank	65
8.4.3.6	GetCameraInfo	65
8.4.3.7	GetConfiguration	65
8.4.3.8	GetEmbeddedImageInfo	65
8.4.3.9	GetFormat7Configuration	66
8.4.3.10	GetFormat7Info	66
8.4.3.11	GetGPIOPinDirection	67
8.4.3.12	GetLUTBankInfo	67
8.4.3.13	GetLUTChannel	67
8.4.3.14	GetLUTInfo	68
8.4.3.15	GetMemoryChannel	68
8.4.3.16	GetMemoryChannelInfo	69
8.4.3.17	GetProperty	69
8.4.3.18	GetPropertyInfo	69
8.4.3.19	GetRegisterString	70
8.4.3.20	GetStrobe	70
8.4.3.21	GetStrobeInfo	70
8.4.3.22	GetTriggerDelay	71
8.4.3.23	GetTriggerDelayInfo	71
8.4.3.24	GetTriggerMode	72
8.4.3.25	GetTriggerModeInfo	72
8.4.3.26	GetVideoModeAndFrameRate	72
8.4.3.27	GetVideoModeAndFrameRateInfo	73
8.4.3.28	IsConnected	73
8.4.3.29	ReadRegister	73
8.4.3.30	ReadRegisterBlock	74

8.4.3.31	RestoreFromMemoryChannel	74
8.4.3.32	RetrieveBuffer	75
8.4.3.33	SaveToMemoryChannel	75
8.4.3.34	SetActiveLUTBank	75
8.4.3.35	SetCallback	76
8.4.3.36	SetConfiguration	76
8.4.3.37	SetEmbeddedImageInfo	76
8.4.3.38	SetFormat7Configuration	77
8.4.3.39	SetFormat7Configuration	77
8.4.3.40	SetGPIOPinDirection	77
8.4.3.41	SetLUTChannel	78
8.4.3.42	SetProperty	78
8.4.3.43	SetStrobe	79
8.4.3.44	SetTriggerDelay	79
8.4.3.45	SetTriggerMode	80
8.4.3.46	SetUserBuffers	80
8.4.3.47	SetVideoModeAndFrameRate	80
8.4.3.48	StartCapture	81
8.4.3.49	StartSyncCapture	81
8.4.3.50	StopCapture	81
8.4.3.51	ValidateFormat7Settings	82
8.4.3.52	WaitForBufferEvent	82
8.4.3.53	WriteRegister	82
8.4.3.54	WriteRegisterBlock	83
8.5	CameraBase Class Reference	84
8.5.1	Detailed Description	88
8.5.2	Constructor & Destructor Documentation	88
8.5.2.1	CameraBase	88
8.5.2.2	~CameraBase	88
8.5.3	Member Function Documentation	88
8.5.3.1	Connect	88
8.5.3.2	Disconnect	89
8.5.3.3	EnableLUT	89
8.5.3.4	FireSoftwareTrigger	89
8.5.3.5	SetActiveLUTBank	90
8.5.3.6	GetCameraInfo	90

8.5.3.7	GetConfiguration	90
8.5.3.8	GetEmbeddedImageInfo	90
8.5.3.9	GetGPIOPinDirection	91
8.5.3.10	GetLUTBankInfo	91
8.5.3.11	GetLUTChannel	92
8.5.3.12	GetLUTInfo	92
8.5.3.13	GetMemoryChannel	92
8.5.3.14	GetMemoryChannelInfo	93
8.5.3.15	GetProperty	93
8.5.3.16	GetPropertyInfo	94
8.5.3.17	GetRegisterString	94
8.5.3.18	GetStrobe	94
8.5.3.19	GetStrobeInfo	95
8.5.3.20	GetTriggerDelay	95
8.5.3.21	GetTriggerDelayInfo	95
8.5.3.22	GetTriggerMode	96
8.5.3.23	GetTriggerModeInfo	96
8.5.3.24	IsConnected	96
8.5.3.25	ReadRegister	97
8.5.3.26	ReadRegisterBlock	97
8.5.3.27	RestoreFromMemoryChannel	98
8.5.3.28	RetrieveBuffer	98
8.5.3.29	SaveToMemoryChannel	98
8.5.3.30	SetActiveLUTBank	99
8.5.3.31	SetCallback	99
8.5.3.32	SetConfiguration	99
8.5.3.33	SetEmbeddedImageInfo	100
8.5.3.34	SetGPIOPinDirection	100
8.5.3.35	SetLUTChannel	100
8.5.3.36	SetProperty	101
8.5.3.37	SetStrobe	101
8.5.3.38	SetTriggerDelay	102
8.5.3.39	SetTriggerMode	102
8.5.3.40	SetUserBuffers	103
8.5.3.41	StartCapture	103
8.5.3.42	StartSyncCapture	104

8.5.3.43	StopCapture	104
8.5.3.44	WaitForBufferEvent	104
8.5.3.45	WriteRegister	105
8.5.3.46	WriteRegisterBlock	105
8.5.4	Member Data Documentation	106
8.5.4.1	m_pCameraData	106
8.6	CameraControlDlg Class Reference	107
8.6.1	Detailed Description	107
8.6.2	Constructor & Destructor Documentation	107
8.6.2.1	CameraControlDlg	107
8.6.2.2	~CameraControlDlg	107
8.6.3	Member Function Documentation	108
8.6.3.1	Connect	108
8.6.3.2	Disconnect	108
8.6.3.3	Hide	108
8.6.3.4	IsVisible	108
8.6.3.5	Show	108
8.7	CameraInfo Struct Reference	109
8.7.1	Detailed Description	111
8.7.2	Constructor & Destructor Documentation	111
8.7.2.1	CameraInfo	111
8.7.3	Member Data Documentation	111
8.7.3.1	bayerTileFormat	111
8.7.3.2	configROM	111
8.7.3.3	defaultGateway	111
8.7.3.4	driverName	111
8.7.3.5	firmwareBuildTime	111
8.7.3.6	firmwareVersion	111
8.7.3.7	gigEMajorVersion	111
8.7.3.8	gigEMinorVersion	111
8.7.3.9	iidcVer	111
8.7.3.10	interfaceType	112
8.7.3.11	ipAddress	112
8.7.3.12	isColorCamera	112
8.7.3.13	macAddress	112
8.7.3.14	maximumBusSpeed	112

8.7.3.15	modelName	112
8.7.3.16	reserved	112
8.7.3.17	sensorInfo	112
8.7.3.18	sensorResolution	112
8.7.3.19	serialNumber	112
8.7.3.20	subnetMask	112
8.7.3.21	userDefinedName	113
8.7.3.22	vendorName	113
8.7.3.23	xmlURL1	113
8.7.3.24	xmlURL2	113
8.8	CameraSelectionDlg Class Reference	114
8.8.1	Detailed Description	114
8.8.2	Constructor & Destructor Documentation	114
8.8.2.1	CameraSelectionDlg	114
8.8.2.2	~CameraSelectionDlg	114
8.8.3	Member Function Documentation	114
8.8.3.1	ShowModal	114
8.9	CameraStats Struct Reference	115
8.9.1	Detailed Description	116
8.9.2	Constructor & Destructor Documentation	116
8.9.2.1	CameraStats	116
8.9.3	Member Data Documentation	116
8.9.3.1	cameraCurrents	116
8.9.3.2	cameraPowerUp	116
8.9.3.3	cameraVoltages	116
8.9.3.4	imageCorrupt	116
8.9.3.5	imageDriverDropped	116
8.9.3.6	imageDropped	116
8.9.3.7	imageXmitFailed	116
8.9.3.8	numCurrents	116
8.9.3.9	numVoltages	116
8.9.3.10	portErrors	116
8.9.3.11	regReadFailed	116
8.9.3.12	regWriteFailed	116
8.9.3.13	reserved	116
8.9.3.14	temperature	117

8.9.3.15	timeSinceBusReset	117
8.9.3.16	timeSinceInitialization	117
8.9.3.17	timeStamp	117
8.10	ConfigROM Struct Reference	118
8.10.1	Detailed Description	119
8.10.2	Constructor & Destructor Documentation	119
8.10.2.1	ConfigROM	119
8.10.3	Member Data Documentation	119
8.10.3.1	chipIdHi	119
8.10.3.2	chipIdLo	119
8.10.3.3	nodeVendorId	119
8.10.3.4	pszKeyword	119
8.10.3.5	reserved	119
8.10.3.6	unitSpecId	119
8.10.3.7	unitSubSWVer	119
8.10.3.8	unitSWVer	119
8.10.3.9	vendorUniqueInfo_0	119
8.10.3.10	vendorUniqueInfo_1	120
8.10.3.11	vendorUniqueInfo_2	120
8.10.3.12	vendorUniqueInfo_3	120
8.11	DCAMFormats Struct Reference	121
8.11.1	Member Data Documentation	121
8.11.1.1	numFormats	121
8.11.1.2	videoModes	121
8.12	EmbeddedImageInfo Struct Reference	122
8.12.1	Detailed Description	122
8.12.2	Member Data Documentation	123
8.12.2.1	brightness	123
8.12.2.2	exposure	123
8.12.2.3	frameCounter	123
8.12.2.4	gain	123
8.12.2.5	GPIOPinState	123
8.12.2.6	ROIPosition	123
8.12.2.7	shutter	123
8.12.2.8	strobePattern	123
8.12.2.9	timestamp	123

8.12.2.10 whiteBalance	123
8.13 EmbeddedImageInfoProperty Struct Reference	124
8.13.1 Detailed Description	124
8.13.2 Constructor & Destructor Documentation	124
8.13.2.1 EmbeddedImageInfoProperty	124
8.13.3 Member Data Documentation	124
8.13.3.1 available	124
8.13.3.2 onOff	124
8.14 Error Class Reference	125
8.14.1 Detailed Description	126
8.14.2 Constructor & Destructor Documentation	126
8.14.2.1 Error	126
8.14.2.2 Error	126
8.14.2.3 ~Error	126
8.14.3 Member Function Documentation	126
8.14.3.1 CollectSupportInformation	126
8.14.3.2 GetBuildDate	126
8.14.3.3 GetCause	127
8.14.3.4 GetDescription	127
8.14.3.5 GetFilename	127
8.14.3.6 GetLine	127
8.14.3.7 GetType	127
8.14.3.8 operator!=	127
8.14.3.9 operator!=	127
8.14.3.10 operator=	128
8.14.3.11 operator==	128
8.14.3.12 operator==	128
8.14.3.13 PrintErrorTrace	128
8.14.4 Friends And Related Function Documentation	128
8.14.4.1 InternalError	128
8.15 FC2Config Struct Reference	129
8.15.1 Detailed Description	129
8.15.2 Constructor & Destructor Documentation	130
8.15.2.1 FC2Config	130
8.15.3 Member Data Documentation	130
8.15.3.1 asyncBusSpeed	130

8.15.3.2	bandwidthAllocation	130
8.15.3.3	grabMode	130
8.15.3.4	grabTimeout	130
8.15.3.5	isochBusSpeed	130
8.15.3.6	numBuffers	130
8.15.3.7	numImageNotifications	130
8.15.3.8	reserved	131
8.16	FC2Version Struct Reference	132
8.16.1	Detailed Description	132
8.16.2	Member Data Documentation	132
8.16.2.1	build	132
8.16.2.2	major	132
8.16.2.3	minor	132
8.16.2.4	type	132
8.17	Format7ImageSettings Struct Reference	133
8.17.1	Detailed Description	133
8.17.2	Constructor & Destructor Documentation	133
8.17.2.1	Format7ImageSettings	133
8.17.3	Member Data Documentation	133
8.17.3.1	height	133
8.17.3.2	mode	134
8.17.3.3	offsetX	134
8.17.3.4	offsetY	134
8.17.3.5	pixelFormat	134
8.17.3.6	reserved	134
8.17.3.7	width	134
8.18	Format7Info Struct Reference	135
8.18.1	Detailed Description	136
8.18.2	Constructor & Destructor Documentation	136
8.18.2.1	Format7Info	136
8.18.3	Member Data Documentation	136
8.18.3.1	imageHStepSize	136
8.18.3.2	imageVStepSize	136
8.18.3.3	maxHeight	136
8.18.3.4	maxPacketSize	136
8.18.3.5	maxWidth	136

8.18.3.6	minPacketSize	136
8.18.3.7	mode	136
8.18.3.8	offsetHStepSize	136
8.18.3.9	offsetVStepSize	136
8.18.3.10	packetSize	137
8.18.3.11	percentage	137
8.18.3.12	pixelFormatBitField	137
8.18.3.13	reserved	137
8.19	Format7PacketInfo Struct Reference	138
8.19.1	Detailed Description	138
8.19.2	Constructor & Destructor Documentation	138
8.19.2.1	Format7PacketInfo	138
8.19.3	Member Data Documentation	138
8.19.3.1	maxBytesPerPacket	138
8.19.3.2	recommendedBytesPerPacket	138
8.19.3.3	reserved	138
8.19.3.4	unitBytesPerPacket	139
8.20	GigECamera Class Reference	140
8.20.1	Detailed Description	145
8.20.2	Constructor & Destructor Documentation	145
8.20.2.1	GigECamera	145
8.20.2.2	~GigECamera	145
8.20.3	Member Function Documentation	145
8.20.3.1	Connect	145
8.20.3.2	Disconnect	146
8.20.3.3	DiscoverGigEPacketSize	146
8.20.3.4	EnableLUT	146
8.20.3.5	FireSoftwareTrigger	147
8.20.3.6	GetActiveLUTBank	147
8.20.3.7	GetCameraInfo	147
8.20.3.8	GetConfiguration	147
8.20.3.9	GetEmbeddedImageInfo	148
8.20.3.10	GetGigEImageBinningSettings	148
8.20.3.11	GetGigEImageSettings	148
8.20.3.12	GetGigEImageSettingsInfo	149
8.20.3.13	GetGigEImagingMode	149

8.20.3.14 GetGigEProperty	149
8.20.3.15 GetGigEStreamChannelInfo	149
8.20.3.16 GetGPIOPinDirection	150
8.20.3.17 GetLUTBankInfo	150
8.20.3.18 GetLUTChannel	150
8.20.3.19 GetLUTInfo	151
8.20.3.20 GetMemoryChannel	151
8.20.3.21 GetMemoryChannelInfo	152
8.20.3.22 GetNumStreamChannels	152
8.20.3.23 GetProperty	152
8.20.3.24 GetPropertyInfo	153
8.20.3.25 GetRegisterString	153
8.20.3.26 GetStrobe	153
8.20.3.27 GetStrobeInfo	154
8.20.3.28 GetTriggerDelay	154
8.20.3.29 GetTriggerDelayInfo	154
8.20.3.30 GetTriggerMode	155
8.20.3.31 GetTriggerModeInfo	155
8.20.3.32 IsConnected	155
8.20.3.33 QueryGigEImagingMode	156
8.20.3.34 ReadGVCPMemory	156
8.20.3.35 ReadGVCPRegister	156
8.20.3.36 ReadGVCPRegisterBlock	157
8.20.3.37 ReadRegister	157
8.20.3.38 ReadRegisterBlock	157
8.20.3.39 RestoreFromMemoryChannel	158
8.20.3.40 RetrieveBuffer	158
8.20.3.41 SaveToMemoryChannel	158
8.20.3.42 SetActiveLUTBank	159
8.20.3.43 SetCallback	159
8.20.3.44 SetConfiguration	159
8.20.3.45 SetEmbeddedImageInfo	160
8.20.3.46 SetGigEImageBinningSettings	160
8.20.3.47 SetGigEImageSettings	160
8.20.3.48 SetGigEImagingMode	160
8.20.3.49 SetGigEProperty	161

8.20.3.50 SetGigEStreamChannelInfo	161
8.20.3.51 SetGPIOPinDirection	161
8.20.3.52 SetLUTChannel	162
8.20.3.53 SetProperty	162
8.20.3.54 SetStrobe	163
8.20.3.55 SetTriggerDelay	163
8.20.3.56 SetTriggerMode	163
8.20.3.57 SetUserBuffers	164
8.20.3.58 StartCapture	164
8.20.3.59 StartSyncCapture	165
8.20.3.60 StopCapture	165
8.20.3.61 WaitForBufferEvent	165
8.20.3.62 WriteGVCPMemory	166
8.20.3.63 WriteGVCPRegister	166
8.20.3.64 WriteGVCPRegisterBlock	166
8.20.3.65 WriteRegister	166
8.20.3.66 WriteRegisterBlock	167
8.21 GigEConfig Struct Reference	168
8.21.1 Detailed Description	168
8.21.2 Constructor & Destructor Documentation	169
8.21.2.1 GigEConfig	169
8.21.3 Member Data Documentation	169
8.21.3.1 channels	169
8.21.3.2 numChannels	169
8.22 GigEImageSettings Struct Reference	170
8.22.1 Detailed Description	170
8.22.2 Constructor & Destructor Documentation	170
8.22.2.1 GigEImageSettings	170
8.22.3 Member Data Documentation	170
8.22.3.1 height	170
8.22.3.2 offsetX	170
8.22.3.3 offsetY	171
8.22.3.4 pixelFormat	171
8.22.3.5 reserved	171
8.22.3.6 width	171
8.23 GigEImageSettingsInfo Struct Reference	172

8.23.1 Detailed Description	172
8.23.2 Constructor & Destructor Documentation	172
8.23.2.1 GigEImageSettingsInfo	172
8.23.3 Member Data Documentation	172
8.23.3.1 imageHStepSize	172
8.23.3.2 imageVStepSize	173
8.23.3.3 maxHeight	173
8.23.3.4 maxWidth	173
8.23.3.5 offsetHStepSize	173
8.23.3.6 offsetVStepSize	173
8.23.3.7 pixelFormatBitField	173
8.23.3.8 reserved	173
8.24 GigEProperty Struct Reference	174
8.24.1 Detailed Description	174
8.24.2 Member Data Documentation	174
8.24.2.1 isReadable	174
8.24.2.2 isWritable	174
8.24.2.3 max	174
8.24.2.4 min	174
8.24.2.5 propType	175
8.24.2.6 value	175
8.25 GigEStreamChannel Struct Reference	176
8.25.1 Detailed Description	176
8.25.2 Constructor & Destructor Documentation	177
8.25.2.1 GigEStreamChannel	177
8.25.3 Member Data Documentation	177
8.25.3.1 destinationIpAddress	177
8.25.3.2 doNotFragment	177
8.25.3.3 hostPort	177
8.25.3.4 interPacketDelay	177
8.25.3.5 networkInterfaceIndex	177
8.25.3.6 packetSize	177
8.25.3.7 sourcePort	177
8.26 HostAdapterStats Struct Reference	178
8.26.1 Detailed Description	178
8.26.2 Constructor & Destructor Documentation	179

8.26.2.1	HostAdapterStats	179
8.26.3	Member Data Documentation	179
8.26.3.1	busErrors	179
8.26.3.2	busResets	179
8.26.3.3	cycleTime	179
8.26.3.4	deviceArrivals	179
8.26.3.5	deviceRemovals	179
8.26.3.6	fifoOverflows	179
8.26.3.7	gapCount	179
8.26.3.8	numPorts	179
8.26.3.9	portErrors	179
8.26.3.10	vendor	179
8.27	Image Class Reference	180
8.27.1	Detailed Description	182
8.27.2	Constructor & Destructor Documentation	183
8.27.2.1	Image	183
8.27.2.2	Image	183
8.27.2.3	Image	183
8.27.2.4	Image	183
8.27.2.5	Image	184
8.27.2.6	~Image	184
8.27.3	Member Function Documentation	184
8.27.3.1	CalculateStatistics	184
8.27.3.2	Convert	184
8.27.3.3	Convert	184
8.27.3.4	DeepCopy	185
8.27.3.5	DetermineBitsPerPixel	185
8.27.3.6	GetBayerTileFormat	185
8.27.3.7	GetBitsPerPixel	185
8.27.3.8	GetColorProcessing	186
8.27.3.9	GetCols	186
8.27.3.10	GetData	186
8.27.3.11	GetData	186
8.27.3.12	GetDataSize	186
8.27.3.13	GetDefaultColorProcessing	186
8.27.3.14	GetDefaultOutputFormat	187

8.27.3.15	GetDimensions	187
8.27.3.16	GetMetadata	187
8.27.3.17	GetPixelFormat	187
8.27.3.18	GetRows	187
8.27.3.19	GetStride	188
8.27.3.20	GetTimeStamp	188
8.27.3.21	operator()	188
8.27.3.22	operator=	188
8.27.3.23	operator[]	188
8.27.3.24	ReleaseBuffer	189
8.27.3.25	Save	189
8.27.3.26	Save	189
8.27.3.27	Save	189
8.27.3.28	Save	190
8.27.3.29	Save	190
8.27.3.30	Save	190
8.27.3.31	Save	190
8.27.3.32	SetColorProcessing	191
8.27.3.33	SetData	191
8.27.3.34	SetDefaultColorProcessing	191
8.27.3.35	SetDefaultOutputFormat	192
8.27.3.36	SetDimensions	192
8.27.4	Friends And Related Function Documentation	192
8.27.4.1	Iso	192
8.28	ImageMetadata Struct Reference	193
8.28.1	Detailed Description	193
8.28.2	Constructor & Destructor Documentation	194
8.28.2.1	ImageMetadata	194
8.28.3	Member Data Documentation	194
8.28.3.1	embeddedBrightness	194
8.28.3.2	embeddedExposure	194
8.28.3.3	embeddedFrameCounter	194
8.28.3.4	embeddedGain	194
8.28.3.5	embeddedGPIOPinState	194
8.28.3.6	embeddedROIPosition	194
8.28.3.7	embeddedShutter	194

8.28.3.8	embeddedStrobePattern	194
8.28.3.9	embeddedTimeStamp	194
8.28.3.10	embeddedWhiteBalance	194
8.28.3.11	reserved	195
8.29	ImageStatistics Class Reference	196
8.29.1	Detailed Description	197
8.29.2	Member Enumeration Documentation	197
8.29.2.1	StatisticsChannel	197
8.29.3	Constructor & Destructor Documentation	198
8.29.3.1	ImageStatistics	198
8.29.3.2	~ImageStatistics	198
8.29.3.3	ImageStatistics	198
8.29.4	Member Function Documentation	198
8.29.4.1	DisableAll	198
8.29.4.2	EnableAll	198
8.29.4.3	EnableGreyOnly	198
8.29.4.4	EnableHSLOnly	198
8.29.4.5	EnableRGBOnly	198
8.29.4.6	GetChannelStatus	199
8.29.4.7	GetHistogram	199
8.29.4.8	GetMean	199
8.29.4.9	GetNumPixelValues	199
8.29.4.10	GetPixelValueRange	200
8.29.4.11	GetRange	200
8.29.4.12	GetStatistics	200
8.29.4.13	operator=	201
8.29.4.14	SetChannelStatus	201
8.29.5	Friends And Related Function Documentation	201
8.29.5.1	ImageStatsCalculator	201
8.30	IPAddress Struct Reference	202
8.30.1	Detailed Description	202
8.30.2	Constructor & Destructor Documentation	202
8.30.2.1	IPAddress	202
8.30.2.2	IPAddress	202
8.30.3	Member Function Documentation	202
8.30.3.1	operator!=	202

8.30.3.2	operator==	202
8.30.4	Member Data Documentation	202
8.30.4.1	octets	202
8.31	JPEGOption Struct Reference	203
8.31.1	Detailed Description	203
8.31.2	Constructor & Destructor Documentation	203
8.31.2.1	JPEGOption	203
8.31.3	Member Data Documentation	203
8.31.3.1	progressive	203
8.31.3.2	quality	203
8.31.3.3	reserved	203
8.32	JPG2Option Struct Reference	204
8.32.1	Detailed Description	204
8.32.2	Constructor & Destructor Documentation	204
8.32.2.1	JPG2Option	204
8.32.3	Member Data Documentation	204
8.32.3.1	quality	204
8.32.3.2	reserved	204
8.33	LUTData Struct Reference	205
8.33.1	Detailed Description	205
8.33.2	Constructor & Destructor Documentation	205
8.33.2.1	LUTData	205
8.33.3	Member Data Documentation	205
8.33.3.1	enabled	205
8.33.3.2	inputBitDepth	205
8.33.3.3	numBanks	206
8.33.3.4	numChannels	206
8.33.3.5	numEntries	206
8.33.3.6	outputBitDepth	206
8.33.3.7	reserved	206
8.33.3.8	supported	206
8.34	MACAddress Struct Reference	207
8.34.1	Detailed Description	207
8.34.2	Constructor & Destructor Documentation	207
8.34.2.1	MACAddress	207
8.34.2.2	MACAddress	207

8.34.3	Member Function Documentation	207
8.34.3.1	operator!=	207
8.34.3.2	operator==	207
8.34.4	Member Data Documentation	207
8.34.4.1	octets	207
8.35	PGMOption Struct Reference	208
8.35.1	Detailed Description	208
8.35.2	Constructor & Destructor Documentation	208
8.35.2.1	PGMOption	208
8.35.3	Member Data Documentation	208
8.35.3.1	binaryFile	208
8.35.3.2	reserved	208
8.36	PGRGuid Class Reference	209
8.36.1	Detailed Description	209
8.36.2	Constructor & Destructor Documentation	209
8.36.2.1	PGRGuid	209
8.36.3	Member Function Documentation	209
8.36.3.1	operator!=	209
8.36.3.2	operator==	209
8.36.4	Member Data Documentation	209
8.36.4.1	value	209
8.37	PNGOption Struct Reference	210
8.37.1	Detailed Description	210
8.37.2	Constructor & Destructor Documentation	210
8.37.2.1	PNGOption	210
8.37.3	Member Data Documentation	210
8.37.3.1	compressionLevel	210
8.37.3.2	interlaced	210
8.37.3.3	reserved	210
8.38	PPMOption Struct Reference	211
8.38.1	Detailed Description	211
8.38.2	Constructor & Destructor Documentation	211
8.38.2.1	PPMOption	211
8.38.3	Member Data Documentation	211
8.38.3.1	binaryFile	211
8.38.3.2	reserved	211

8.39 Property Struct Reference	212
8.39.1 Detailed Description	212
8.39.2 Constructor & Destructor Documentation	213
8.39.2.1 Property	213
8.39.2.2 Property	213
8.39.3 Member Data Documentation	213
8.39.3.1 absControl	213
8.39.3.2 absValue	213
8.39.3.3 autoManualMode	213
8.39.3.4 onePush	213
8.39.3.5 onOff	213
8.39.3.6 present	213
8.39.3.7 reserved	213
8.39.3.8 type	213
8.39.3.9 valueA	213
8.39.3.10 valueB	213
8.40 PropertyInfo Struct Reference	214
8.40.1 Detailed Description	215
8.40.2 Constructor & Destructor Documentation	215
8.40.2.1 PropertyInfo	215
8.40.2.2 PropertyInfo	215
8.40.3 Member Data Documentation	215
8.40.3.1 absMax	215
8.40.3.2 absMin	215
8.40.3.3 absValSupported	215
8.40.3.4 autoSupported	215
8.40.3.5 manualSupported	215
8.40.3.6 max	215
8.40.3.7 min	215
8.40.3.8 onePushSupported	215
8.40.3.9 onOffSupported	215
8.40.3.10 present	216
8.40.3.11 pUnitAbbr	216
8.40.3.12 pUnits	216
8.40.3.13 readOutSupported	216
8.40.3.14 reserved	216

8.40.3.15 type	216
8.41 StrobeControl Struct Reference	217
8.41.1 Detailed Description	217
8.41.2 Constructor & Destructor Documentation	217
8.41.2.1 StrobeControl	217
8.41.3 Member Data Documentation	217
8.41.3.1 delay	217
8.41.3.2 duration	217
8.41.3.3 onOff	217
8.41.3.4 polarity	218
8.41.3.5 reserved	218
8.41.3.6 source	218
8.42 StrobeInfo Struct Reference	219
8.42.1 Detailed Description	219
8.42.2 Constructor & Destructor Documentation	219
8.42.2.1 StrobeInfo	219
8.42.3 Member Data Documentation	219
8.42.3.1 maxValu	219
8.42.3.2 minValu	219
8.42.3.3 onOffSupported	220
8.42.3.4 polaritySupported	220
8.42.3.5 present	220
8.42.3.6 readOutSupported	220
8.42.3.7 reserved	220
8.42.3.8 source	220
8.43 SystemInfo Struct Reference	221
8.43.1 Detailed Description	221
8.43.2 Member Data Documentation	221
8.43.2.1 byteOrder	221
8.43.2.2 cpuDescription	222
8.43.2.3 driverList	222
8.43.2.4 gpuDescription	222
8.43.2.5 libraryList	222
8.43.2.6 numCpuCores	222
8.43.2.7 osDescription	222
8.43.2.8 osType	222

8.43.2.9 reserved	222
8.43.2.10 screenHeight	222
8.43.2.11 screenWidth	222
8.43.2.12 sysMemSize	222
8.44 TIFFOption Struct Reference	223
8.44.1 Detailed Description	223
8.44.2 Member Enumeration Documentation	223
8.44.2.1 CompressionMethod	223
8.44.3 Constructor & Destructor Documentation	224
8.44.3.1 TIFFOption	224
8.44.4 Member Data Documentation	224
8.44.4.1 compression	224
8.44.4.2 reserved	224
8.45 TimeStamp Struct Reference	225
8.45.1 Detailed Description	225
8.45.2 Constructor & Destructor Documentation	225
8.45.2.1 TimeStamp	225
8.45.3 Member Data Documentation	225
8.45.3.1 cycleCount	225
8.45.3.2 cycleOffset	225
8.45.3.3 cycleSeconds	225
8.45.3.4 microSeconds	225
8.45.3.5 reserved	226
8.45.3.6 seconds	226
8.46 TopologyNode Class Reference	227
8.46.1 Detailed Description	228
8.46.2 Member Enumeration Documentation	228
8.46.2.1 NodeType	228
8.46.2.2 PortType	228
8.46.3 Constructor & Destructor Documentation	228
8.46.3.1 TopologyNode	228
8.46.3.2 TopologyNode	228
8.46.3.3 ~TopologyNode	228
8.46.3.4 TopologyNode	229
8.46.4 Member Function Documentation	229
8.46.4.1 AddChild	229

8.46.4.2	AddPort	229
8.46.4.3	AssignGuidToNode	229
8.46.4.4	AssignGuidToNode	229
8.46.4.5	GetChild	229
8.46.4.6	GetDeviceId	229
8.46.4.7	GetGuid	230
8.46.4.8	GetInterfaceType	230
8.46.4.9	GetNodeType	230
8.46.4.10	GetNumChildren	230
8.46.4.11	GetNumPorts	230
8.46.4.12	GetPortType	230
8.46.4.13	operator=	230
8.47	TriggerMode Struct Reference	231
8.47.1	Detailed Description	231
8.47.2	Constructor & Destructor Documentation	231
8.47.2.1	TriggerMode	231
8.47.3	Member Data Documentation	231
8.47.3.1	mode	231
8.47.3.2	onOff	231
8.47.3.3	parameter	231
8.47.3.4	polarity	231
8.47.3.5	reserved	231
8.47.3.6	source	232
8.48	TriggerModeInfo Struct Reference	233
8.48.1	Detailed Description	233
8.48.2	Constructor & Destructor Documentation	233
8.48.2.1	TriggerModeInfo	233
8.48.3	Member Data Documentation	233
8.48.3.1	modeMask	233
8.48.3.2	onOffSupported	233
8.48.3.3	polaritySupported	233
8.48.3.4	present	234
8.48.3.5	readOutSupported	234
8.48.3.6	reserved	234
8.48.3.7	softwareTriggerSupported	234
8.48.3.8	sourceMask	234

8.48.3.9	valueReadable	234
8.49	Utilities Class Reference	235
8.49.1	Detailed Description	235
8.49.2	Member Function Documentation	235
8.49.2.1	GetLibraryVersion	235
8.49.2.2	GetSystemInfo	235
8.49.2.3	LaunchBrowser	235
8.49.2.4	LaunchCommand	236
8.49.2.5	LaunchCommandAsync	236
8.49.2.6	LaunchHelp	236
8.50	VideoModes Struct Reference	237
8.50.1	Member Data Documentation	237
8.50.1.1	frameRate	237
8.50.1.2	height	237
8.50.1.3	videoMode	237
8.50.1.4	width	237
9	File Documentation	239
9.1	AVIRecorder.h File Reference	239
9.2	BusManager.h File Reference	240
9.3	Camera.h File Reference	241
9.4	CameraBase.h File Reference	242
9.5	Error.h File Reference	243
9.6	FlyCapture2.h File Reference	244
9.7	FlyCapture2Defs.h File Reference	245
9.7.1	Define Documentation	252
9.7.1.1	FULL_32BIT_VALUE	252
9.7.1.2	NULL	252
9.8	FlyCapture2GUI.h File Reference	253
9.9	FlyCapture2Platform.h File Reference	254
9.9.1	Define Documentation	254
9.9.1.1	FLYCAPTURE2_API	254
9.10	GigECamera.h File Reference	255
9.11	Image.h File Reference	256
9.12	ImageStatistics.h File Reference	257
9.13	PGRDirectShow.h File Reference	258
9.13.1	Function Documentation	262

9.13.1.1	GetAbsBrightness	262
9.13.1.2	GetAbsBrightnessRange	262
9.13.1.3	GetAbsExposure	262
9.13.1.4	GetAbsExposureRange	262
9.13.1.5	GetAbsGain	262
9.13.1.6	GetAbsGainRange	262
9.13.1.7	GetAbsGamma	262
9.13.1.8	GetAbsGammaRange	262
9.13.1.9	GetAbsHue	262
9.13.1.10	GetAbsHueRange	262
9.13.1.11	GetAbsPan	262
9.13.1.12	GetAbsPanRange	262
9.13.1.13	GetAbsSaturation	262
9.13.1.14	GetAbsSaturationRange	262
9.13.1.15	GetAbsSharpness	262
9.13.1.16	GetAbsSharpnessRange	262
9.13.1.17	GetAbsShutter	262
9.13.1.18	GetAbsShutterRange	262
9.13.1.19	GetAbsTilt	262
9.13.1.20	GetAbsTiltRange	262
9.13.1.21	GetAbsWhiteBalance	262
9.13.1.22	GetAbsWhiteBalanceRange	262
9.13.1.23	GetBrightness	262
9.13.1.24	GetBrightnessRange	262
9.13.1.25	GetCustomImage	262
9.13.1.26	GetCustomImageMode	262
9.13.1.27	GetExposure	262
9.13.1.28	GetExposureRange	262
9.13.1.29	GetGain	262
9.13.1.30	GetGainRange	262
9.13.1.31	GetGamma	262
9.13.1.32	GetGammaRange	262
9.13.1.33	GetHue	262
9.13.1.34	GetHueRange	262
9.13.1.35	GetImageFormat	262
9.13.1.36	GetOutputVerticalFlip	262

9.13.1.37 GetPan	262
9.13.1.38 GetPanRange	262
9.13.1.39 GetSaturation	262
9.13.1.40 GetSaturationRange	262
9.13.1.41 GetSharpness	262
9.13.1.42 GetSharpnessRange	262
9.13.1.43 GetShutter	262
9.13.1.44 GetShutterRange	262
9.13.1.45 GetTilt	262
9.13.1.46 GetTiltRange	262
9.13.1.47 GetWhiteBalance	262
9.13.1.48 GetWhiteBalanceRange	262
9.13.1.49 QueryCustomImage	262
9.13.1.50 ReadRegister	262
9.13.1.51 SetAbsBrightness	262
9.13.1.52 SetAbsExposure	262
9.13.1.53 SetAbsGain	262
9.13.1.54 SetAbsGamma	262
9.13.1.55 SetAbsHue	262
9.13.1.56 SetAbsPan	262
9.13.1.57 SetAbsSaturation	262
9.13.1.58 SetAbsSharpness	262
9.13.1.59 SetAbsShutter	262
9.13.1.60 SetAbsTilt	262
9.13.1.61 SetAbsWhiteBalance	262
9.13.1.62 SetBrightness	262
9.13.1.63 SetCustomImage	262
9.13.1.64 SetCustomImageMode	262
9.13.1.65 SetExposure	262
9.13.1.66 SetGain	262
9.13.1.67 SetGamma	262
9.13.1.68 SetHue	262
9.13.1.69 SetImageFormat	262
9.13.1.70 SetOutputVerticalFlip	262
9.13.1.71 SetPan	262
9.13.1.72 SetSaturation	262

9.13.1.73 SetSharpness	262
9.13.1.74 SetShutter	262
9.13.1.75 SetTilt	262
9.13.1.76 SetWhiteBalance	262
9.13.1.77 WriteRegister	262
9.13.2 Variable Documentation	262
9.13.2.1 IID_IFlyCaptureProperties	262
9.14 TopologyNode.h File Reference	263
9.15 Utilities.h File Reference	264

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Global constants	11
Enumerations	12
GigE specific enumerations	27
Structures	28
GigE specific structures	31
IIDC specific structures	32
Image saving structures.	33

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

FlyCapture2	35
-----------------------------	-------	----

Chapter 3

Class Index

3.1 Class Hierarchy

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

AVIOption	47
AVIRecorder	49
BusManager	51
CameraBase	84
Camera	59
GigECamera	140
CameraControlDlg	107
CameraInfo	109
CameraSelectionDlg	114
CameraStats	115
ConfigROM	118
DCAMFormats	121
EmbeddedImageInfo	122
EmbeddedImageInfoProperty	124
Error	125
FC2Config	129
FC2Version	132
Format7ImageSettings	133
Format7Info	135
Format7PacketInfo	138
GigEConfig	168
GigEImageSettings	170
GigEImageSettingsInfo	172
GigEProperty	174
GigEStreamChannel	176
HostAdapterStats	178
Image	180
ImageMetadata	193
ImageStatistics	196
IPAddress	202
JPEGOption	203
JPG2Option	204
LUTData	205

MACAddress	207
PGMOption	208
PGRGuid	209
PNGOption	210
PPMOption	211
Property	212
PropertyInfo	214
StrobeControl	217
StrobeInfo	219
SystemInfo	221
TIFFOption	223
TimeStamp	225
TopologyNode	227
TriggerMode	231
TriggerModeInfo	233
Utilities	235
VideoModes	237

Chapter 4

Class Index

4.1 Class List

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

AVIOption (Options for saving AVI files)	47
AVIRecorder (Functionality for the user to record images to an AVI file)	49
BusManager (Functionality for the user to get an PGRGuid for a desired camera or device easily)	51
Camera (The Camera object represents a physical camera that uses the IIDC register set)	59
CameraBase (Abstract base class that defines a general interface to a camera)	84
CameraControlDlg (The CameraControlDlg object represents a GTKmm dialog that provides a graphical interface to a specified camera)	107
CameraInfo (Camera information)	109
CameraSelectionDlg (The CameraSelectionDlg object represents a GTKmm dialog that provides a graphical interface that lists the number of cameras available to the library)	114
CameraStats (Camera diagnostic information)	115
ConfigROM (Camera configuration ROM)	118
DCAMFormats	121
EmbeddedImageInfo (Properties of the possible embedded image information)	122
EmbeddedImageInfoProperty (Properties of a single embedded image info property)	124
Error (The Error object represents an error that is returned from the library)	125
FC2Config (Configuration for a camera)	129
FC2Version (The current version of the library)	132
Format7ImageSettings (Format 7 image settings)	133
Format7Info (Format 7 information for a single mode)	135
Format7PacketInfo (Format 7 packet information)	138
GigECamera (The GigECamera object represents a physical Gigabit Ethernet camera)	140
GigEConfig (Configuration for a GigE camera)	168
GigEImageSettings (Image settings for a GigE camera)	170
GigEImageSettingsInfo (Format 7 information for a single mode)	172
GigEProperty (A GigE property)	174
GigEStreamChannel (Information about a single GigE stream channel)	176
HostAdapterStats (Information about the host adapter's statistics)	178
Image (Used to retrieve images from a camera, convert between multiple pixel formats and save images to disk)	180
ImageMetadata (Metadata related to an image)	193
ImageStatistics (The ImageStatistics object represents image statistics for an image)	196
IPAddress (IPv4 address)	202

JPEGOption (Options for saving JPEG image)	203
JPG2Option (Options for saving JPEG2000 image)	204
LUTData (Information about the camera's look up table)	205
MACAddress (MAC address)	207
PGMOption (Options for saving PGM images)	208
PGRGuid (A GUID to the camera)	209
PNGOption (Options for saving PNG images)	210
PPMOption (Options for saving PPM images)	211
Property (A specific camera property)	212
PropertyInfo (Information about a specific camera property)	214
StrobeControl (A camera strobe)	217
StrobeInfo (A camera strobe property)	219
SystemInfo (Description of the system)	221
TIFFOption (Options for saving TIFF images)	223
TimeStamp (Timestamp information)	225
TopologyNode (Topology information that can be used to generate a tree structure of all cameras and devices connected to a computer)	227
TriggerMode (A camera trigger)	231
TriggerModeInfo (Information about a camera trigger property)	233
Utilities (The Utility class is generally used to query for general system information such as operating system, available memory etc)	235
VideoModes	237

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

AVIRecorder.h	239
BusManager.h	240
Camera.h	241
CameraBase.h	242
Error.h	243
FlyCapture2.h	244
FlyCapture2Defs.h	245
FlyCapture2GUI.h	253
FlyCapture2Platform.h	254
GigECamera.h	255
Image.h	256
ImageStatistics.h	257
PGRDirectShow.h	258
TopologyNode.h	263
Utilities.h	264

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_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 `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 100Mbps/sec.

BUSSPEED_S200 200Mbps/sec.

BUSSPEED_S400 400Mbps/sec.

BUSSPEED_S480 480Mbps/sec.

Only for USB cameras.

BUSSPEED_S800 800Mbps/sec.

BUSSPEED_S1600 1600Mbps/sec.

BUSSPEED_S3200 3200Mbps/sec.

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 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_INITIALIZED [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

6.2.1.7 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

6.2.1.8 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

6.2.1.9 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

6.2.1.10 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

6.2.1.11 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

6.2.1.12 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*

6.2.1.13 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.

6.2.1.14 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

6.2.1.15 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

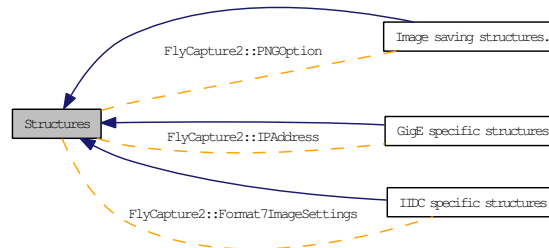
HEARTBEAT_TIMEOUT

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_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 `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.

- 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.

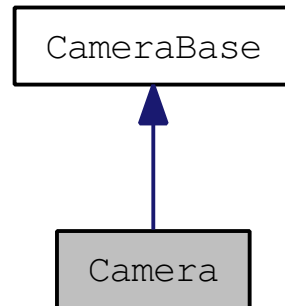
The documentation for this class was generated from the following file:

- [BusManager.h](#)

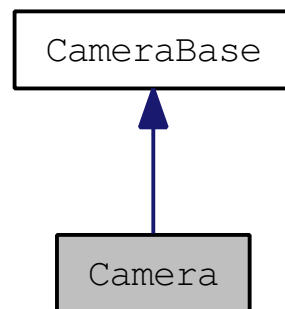
8.4 Camera Class Reference

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](#).

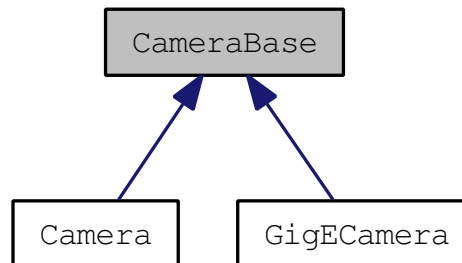
The documentation for this class was generated from the following file:

- [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.

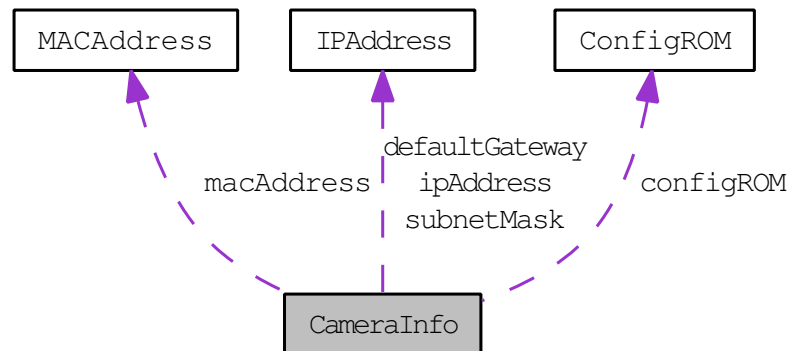
The documentation for this class was generated from the following file:

- [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.
- 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 char firmwareBuildTime[sk_maxStringLength]

Firmware build time.

8.7.3.6 char firmwareVersion[sk_maxStringLength]

Firmware version of camera.

8.7.3.7 unsigned int gigEMajorVersion

GigE Vision version.

8.7.3.8 unsigned int gigEMinorVersion

GigE Vision minor version.

8.7.3.9 unsigned int iidcVer

DCAM version.

8.7.3.10 InterfaceType interfaceType

Interface type.

8.7.3.11 IPAddress ipAddress

IP address.

8.7.3.12 bool isColorCamera

Flag indicating if this is a color camera.

8.7.3.13 MACAddress macAddress

MAC address.

8.7.3.14 BusSpeed maximumBusSpeed

Maximum bus speed.

8.7.3.15 char modelName[sk_maxStringLength]

Device model name.

8.7.3.16 unsigned int reserved[16]

Reserved for future use.

8.7.3.17 char sensorInfo[sk_maxStringLength]

String detailing the sensor information.

8.7.3.18 char sensorResolution[sk_maxStringLength]

String providing the sensor resolution.

8.7.3.19 unsigned int serialNumber

Device serial number.

8.7.3.20 IPAddress subnetMask

Subnet mask.

8.7.3.21 char userDefinedName[sk_maxStringLength]

User defined name.

8.7.3.22 char vendorName[sk_maxStringLength]

Device vendor name.

8.7.3.23 char xmlURL1[sk_maxStringLength]

XML URL 1.

8.7.3.24 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.

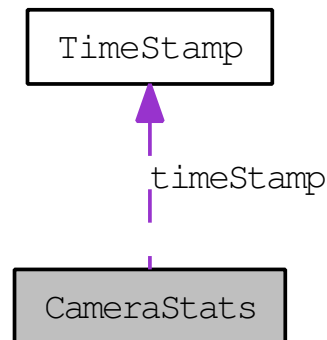
The documentation for this class was generated from the following file:

- [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

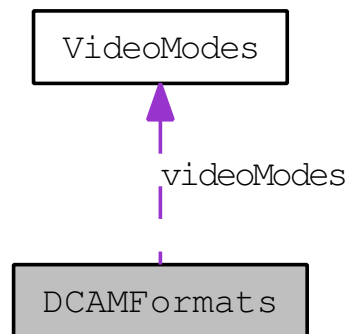
Vendor unique info 3.

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

- [FlyCapture2Defs.h](#)

8.11 DCAMFormats Struct Reference

Collaboration diagram for DCAMFormats:



Public Attributes

- [VideoModes videoModes](#) [FlyCapture2::NUM_VIDEOMODES]
- unsigned int [numFormats](#)

8.11.1 Member Data Documentation

8.11.1.1 unsigned int numFormats

8.11.1.2 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.

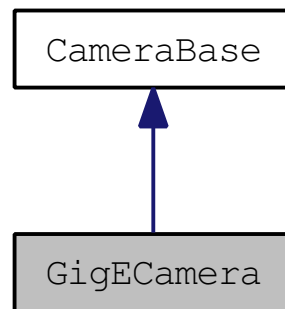
The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

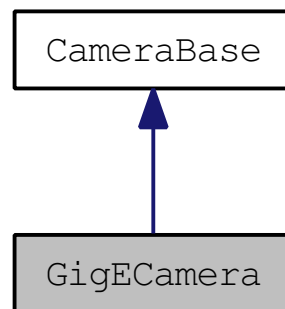
8.20 GigECamera Class Reference

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](#).

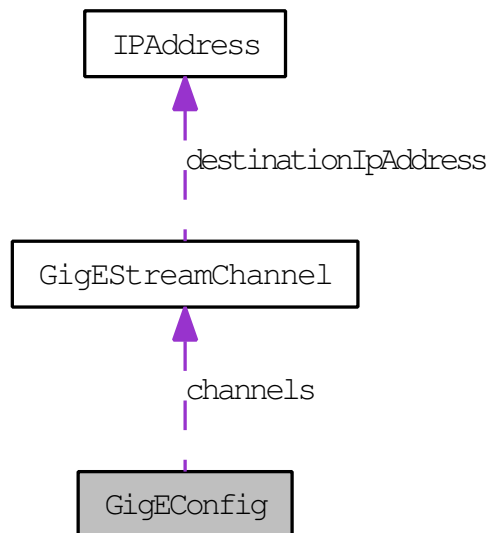
The documentation for this class was generated from the following file:

- [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.

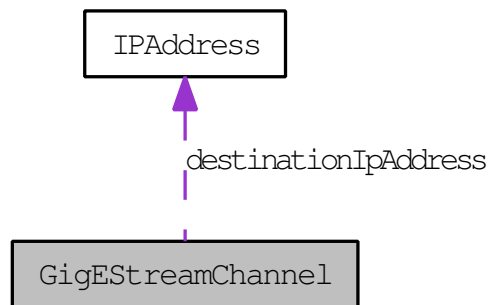
The documentation for this struct was generated from the following file:

- [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.

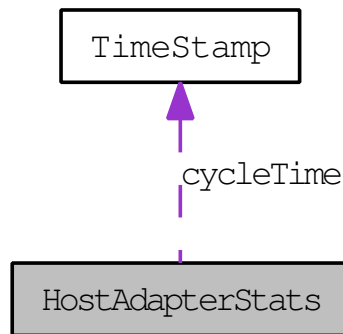
The documentation for this struct was generated from the following file:

- [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 red value. Use Value A for the blue 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

Public Attributes

- [FlyCapture2::VideoMode](#) videoMode
- [FlyCapture2::FrameRate](#) frameRate
- unsigned int width
- unsigned int height

8.50.1 Member Data Documentation

8.50.1.1 [FlyCapture2::FrameRate](#) frameRate

8.50.1.2 unsigned int height

8.50.1.3 [FlyCapture2::VideoMode](#) videoMode

8.50.1.4 unsigned int width

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

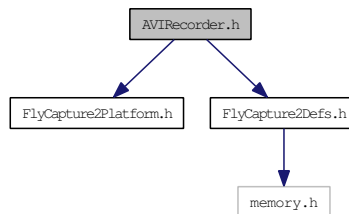
- [PGRDirectShow.h](#)

Chapter 9

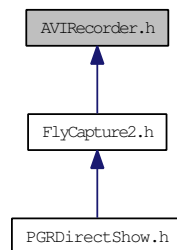
File Documentation

9.1 AVIRecorder.h File Reference

Include dependency graph for AVIRecorder.h:



This graph shows which files directly or indirectly include this file:



Classes

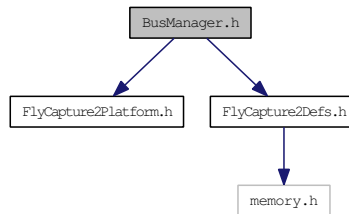
- class [AVIRecorder](#)
The [AVIRecorder](#) class provides the functionality for the user to record images to an AVI file.

Namespaces

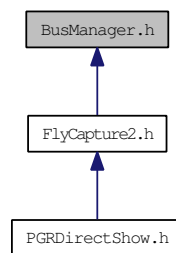
- namespace [FlyCapture2](#)

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

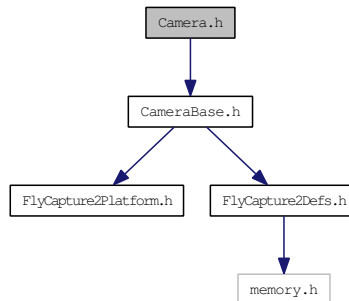
- namespace [FlyCapture2](#)

Typedefs

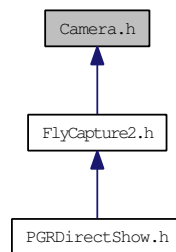
- typedef void(* [BusEventCallback](#))(void *pParameter, unsigned int serialNumber)
Bus event callback function prototype.
- typedef void * [CallbackHandle](#)
Handle that is returned when registering a callback.

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](#)

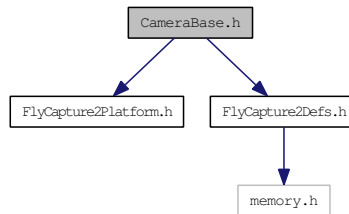
The [Camera](#) object represents a physical camera that uses the IIDC register set.

Namespaces

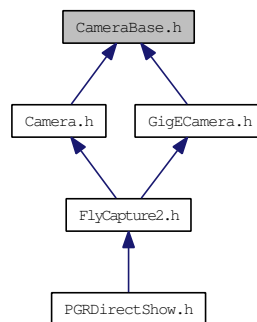
- namespace [FlyCapture2](#)

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

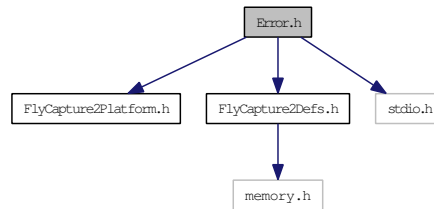
- namespace [FlyCapture2](#)

Typedefs

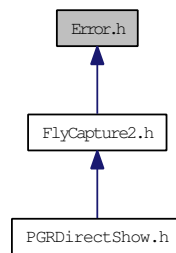
- typedef void(* [ImageEventCallback](#))(class Image *pImage, const void *pCallbackData)
[Image](#) event callback function prototype.

9.5 Error.h File Reference

Include dependency graph for Error.h:



This graph shows which files directly or indirectly include this file:



Classes

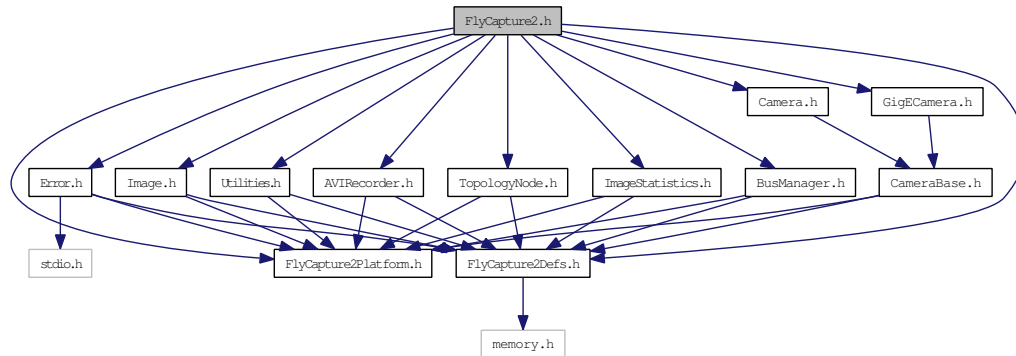
- class [Error](#)
The [Error](#) object represents an error that is returned from the library.

Namespaces

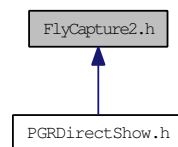
- namespace [FlyCapture2](#)

9.6 FlyCapture2.h File Reference

Include dependency graph for FlyCapture2.h:

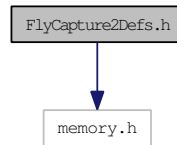


This graph shows which files directly or indirectly include this file:

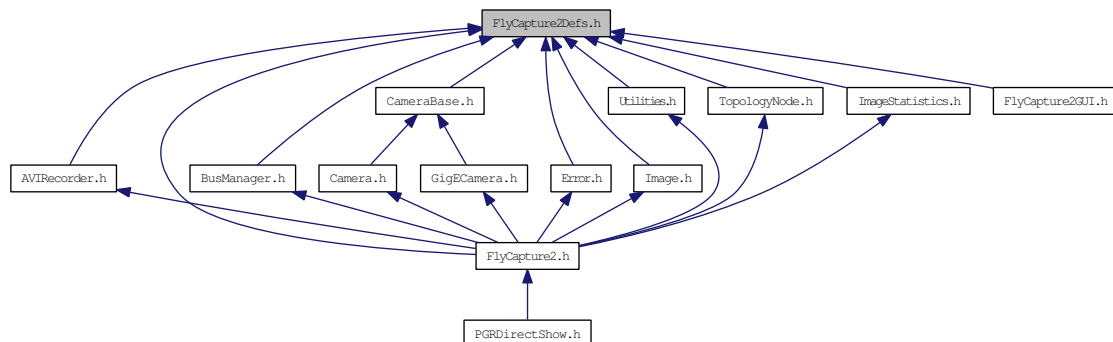


9.7 FlyCapture2Defs.h File Reference

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_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 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.

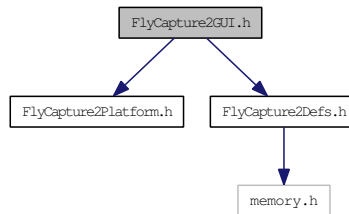
9.7.1 Define Documentation

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

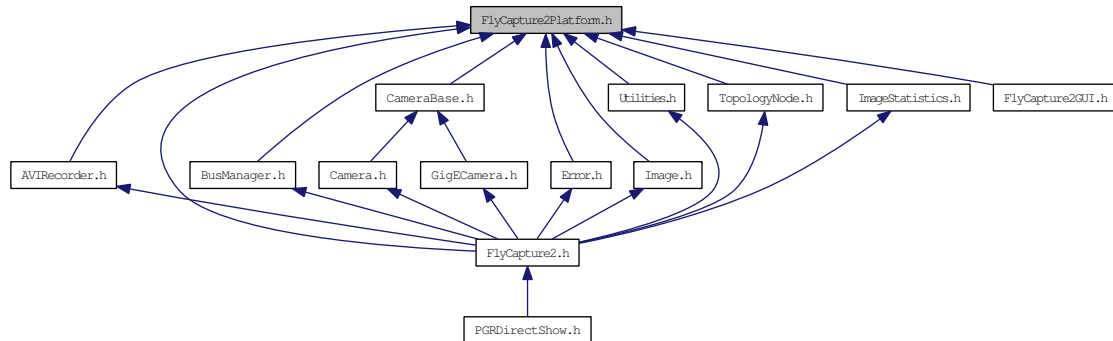
- class [CameraControlDlg](#)
The [CameraControlDlg](#) object represents a GTKmm dialog that provides a graphical interface to a specified camera.
- class [CameraSelectionDlg](#)
The [CameraSelectionDlg](#) object represents a GTKmm dialog that provides a graphical interface that lists the number of 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

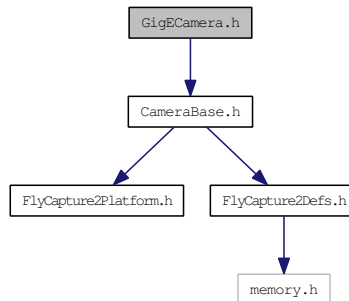
- `#define` [FLYCAPTURE2_API](#)

9.9.1 Define Documentation

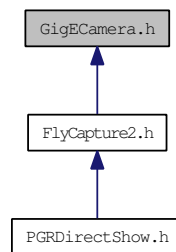
9.9.1.1 `#define` FLYCAPTURE2_API

9.10 GigECamera.h File Reference

Include dependency graph for GigECamera.h:



This graph shows which files directly or indirectly include this file:



Classes

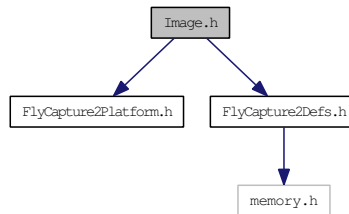
- class [GigECamera](#)
The [GigECamera](#) object represents a physical Gigabit Ethernet camera.

Namespaces

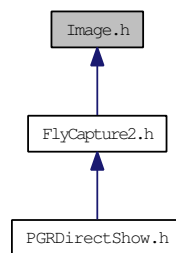
- namespace [FlyCapture2](#)

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](#)

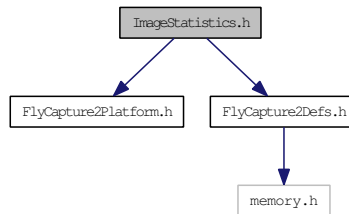
The [Image](#) class is used to retrieve images from a camera, convert between multiple pixel formats and save images to disk.

Namespaces

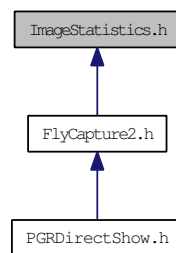
- namespace [FlyCapture2](#)

9.12 ImageStatistics.h File Reference

Include dependency graph for ImageStatistics.h:



This graph shows which files directly or indirectly include this file:



Classes

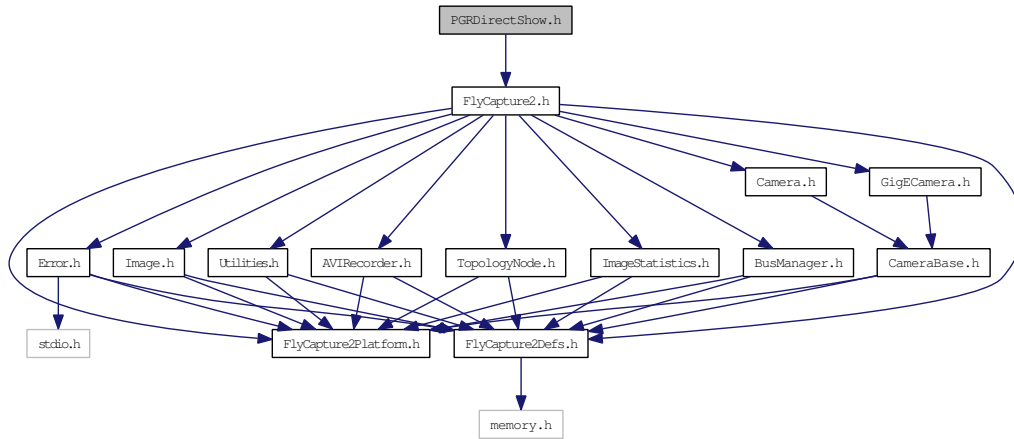
- class [ImageStatistics](#)
The [ImageStatistics](#) object represents image statistics for an image.

Namespaces

- namespace [FlyCapture2](#)

9.13 PGRDirectShow.h File Reference

Include dependency graph for PGRDirectShow.h:



Classes

- struct [VideoModes](#)
- struct [DCAMFormats](#)

Functions

- STDMETHOD() [GetImageFormat](#) (unsigned long *pulFormat)=0
- STDMETHOD() [SetImageFormat](#) (unsigned long ulFormat)=0
- STDMETHOD() [ReadRegister](#) (unsigned int offset, unsigned int *pValue)=0
- STDMETHOD() [WriteRegister](#) (unsigned int offset, unsigned int value, bool broadcast=false)=0
- STDMETHOD() [GetBrightness](#) (long *pBrightness, bool *pbAuto=NULL)=0
- STDMETHOD() [SetBrightness](#) (long lBrightness, bool bAuto=false)=0
- STDMETHOD() [GetBrightnessRange](#) (long *pMin, long *pMax, bool *pbAutoSupported=NULL)=0
- STDMETHOD() [GetAbsBrightness](#) (float *pBrightness, bool *pbAuto=NULL)=0
- STDMETHOD() [SetAbsBrightness](#) (float lBrightness, bool bAuto=false)=0
- STDMETHOD() [GetAbsBrightnessRange](#) (float *pMin, float *pMax, bool *pbAutoSupported=NULL, const char **pUnits=NULL)=0
- STDMETHOD() [GetExposure](#) (long *plExposure, bool *pbAuto=NULL)=0
- STDMETHOD() [SetExposure](#) (long lExposure, bool bAuto=false)=0
- STDMETHOD() [GetExposureRange](#) (long *pMin, long *pMax, bool *pbAutoSupported=NULL)=0
- STDMETHOD() [GetAbsExposure](#) (float *plExposure, bool *pbAuto=NULL)=0
- STDMETHOD() [SetAbsExposure](#) (float lExposure, bool bAuto=false)=0
- STDMETHOD() [GetAbsExposureRange](#) (float *pMin, float *pMax, bool *pbAutoSupported=NULL, const char **pUnits=NULL)=0
- STDMETHOD() [GetShutter](#) (long *plShutter, bool *pbAuto=NULL)=0
- STDMETHOD() [SetShutter](#) (long lShutter, bool bAuto=false)=0
- STDMETHOD() [GetShutterRange](#) (long *pMin, long *pMax, bool *pbAutoSupported=NULL)=0
- STDMETHOD() [GetAbsShutter](#) (float *plShutter, bool *pbAuto=NULL)=0
- STDMETHOD() [SetAbsShutter](#) (float lShutter, bool bAuto=false)=0
- STDMETHOD() [GetAbsShutterRange](#) (float *pMin, float *pMax, bool *pbAutoSupported=NULL, const char **pUnits=NULL)=0
- STDMETHOD() [GetSharpness](#) (long *plSharpness, bool *pbAuto=NULL)=0
- STDMETHOD() [SetSharpness](#) (long lSharpness, bool bAuto=false)=0
- STDMETHOD() [GetSharpnessRange](#) (long *pMin, long *pMax, bool *pbAutoSupported=NULL)=0
- STDMETHOD() [GetAbsSharpness](#) (float *plSharpness, bool *pbAuto=NULL)=0

```

· STDMETHODCALLTYPE SetAbsSharpness (float lSharpness, bool bAuto=false)=0
· STDMETHODCALLTYPE GetAbsSharpnessRange (float *pMin, float *pMax, bool
    *pbAutoSupported=NULL, const char **pUnits=NULL)=0
· STDMETHODCALLTYPE GetGain (long *pIGain, bool *pbAuto=NULL)=0
· STDMETHODCALLTYPE SetGain (long lGain, bool bAuto=false)=0
· STDMETHODCALLTYPE GetGainRange (long *pMin, long *pMax, bool
    *pbAutoSupported=NULL)=0
· STDMETHODCALLTYPE GetAbsGain (float *pIGain, bool *pbAuto=NULL)=0
· STDMETHODCALLTYPE SetAbsGain (float lGain, bool bAuto=false)=0
· STDMETHODCALLTYPE GetAbsGainRange (float *pMin, float *pMax, bool
    *pbAutoSupported=NULL, const char **pUnits=NULL)=0
· STDMETHODCALLTYPE GetHue (long *pIHue, bool *pbAuto=NULL)=0
· STDMETHODCALLTYPE SetHue (long lHue, bool bAuto=false)=0
· STDMETHODCALLTYPE GetHueRange (long *pMin, long *pMax, bool
    *pbAutoSupported=NULL)=0
· STDMETHODCALLTYPE GetAbsHue (float *pIHue, bool *pbAuto=NULL)=0
· STDMETHODCALLTYPE SetAbsHue (float lHue, bool bAuto=false)=0
· STDMETHODCALLTYPE GetAbsHueRange (float *pMin, float *pMax, bool
    *pbAutoSupported=NULL, const char **pUnits=NULL)=0
· STDMETHODCALLTYPE GetSaturation (long *pISaturation, bool *pbAuto=NULL)=0
· STDMETHODCALLTYPE SetSaturation (long lSaturation, bool bAuto=false)=0
· STDMETHODCALLTYPE GetSaturationRange (long *pMin, long *pMax, bool
    *pbAutoSupported=NULL)=0
· STDMETHODCALLTYPE GetAbsSaturation (float *pISaturation, bool *pbAuto=NULL)=0
· STDMETHODCALLTYPE SetAbsSaturation (float lSaturation, bool bAuto=false)=0
· STDMETHODCALLTYPE GetAbsSaturationRange (float *pMin, float *pMax, bool
    *pbAutoSupported=NULL, const char **pUnits=NULL)=0
· STDMETHODCALLTYPE GetGamma (long *pIGamma, bool *pbAuto=NULL)=0
· STDMETHODCALLTYPE SetGamma (long lGamma, bool bAuto=false)=0
· STDMETHODCALLTYPE GetGammaRange (long *pMin, long *pMax, bool
    *pbAutoSupported=NULL)=0
· STDMETHODCALLTYPE GetAbsGamma (float *pIGamma, bool *pbAuto=NULL)=0
· STDMETHODCALLTYPE SetAbsGamma (float lGamma, bool bAuto=false)=0
· STDMETHODCALLTYPE GetAbsGammaRange (float *pMin, float *pMax, bool
    *pbAutoSupported=NULL, const char **pUnits=NULL)=0
· STDMETHODCALLTYPE GetPan (long *pIPan, bool *pbAuto=NULL)=0
· STDMETHODCALLTYPE SetPan (long lPan, bool bAuto=false)=0
· STDMETHODCALLTYPE GetPanRange (long *pMin, long *pMax, bool
    *pbAutoSupported=NULL)=0
· STDMETHODCALLTYPE GetAbsPan (float *pIPan, bool *pbAuto=NULL)=0
· STDMETHODCALLTYPE SetAbsPan (float lPan, bool bAuto=false)=0
· STDMETHODCALLTYPE GetAbsPanRange (float *pMin, float *pMax, bool
    *pbAutoSupported=NULL, const char **pUnits=NULL)=0
· STDMETHODCALLTYPE GetTilt (long *pITilt, bool *pbAuto=NULL)=0
· STDMETHODCALLTYPE SetTilt (long lTilt, bool bAuto=false)=0
· STDMETHODCALLTYPE GetTiltRange (long *pMin, long *pMax, bool
    *pbAutoSupported=NULL)=0
· STDMETHODCALLTYPE GetAbsTilt (float *pITilt, bool *pbAuto=NULL)=0
· STDMETHODCALLTYPE SetAbsTilt (float lTilt, bool bAuto=false)=0
· STDMETHODCALLTYPE GetAbsTiltRange (float *pMin, float *pMax, bool
    *pbAutoSupported=NULL, const char **pUnits=NULL)=0
· STDMETHODCALLTYPE GetWhiteBalance (long *pIWhiteBalance, bool *pbAuto=NULL)=0
· STDMETHODCALLTYPE SetWhiteBalance (long lWhiteBalance, bool bAuto=false)=0
· STDMETHODCALLTYPE GetWhiteBalanceRange (long *pMin, long *pMax, bool
    *pbAutoSupported=NULL)=0
· STDMETHODCALLTYPE GetAbsWhiteBalance (float *pIWhiteBalance, bool
    *pbAuto=NULL)=0
· STDMETHODCALLTYPE SetAbsWhiteBalance (float lWhiteBalance, bool bAuto=false)=0
· STDMETHODCALLTYPE GetAbsWhiteBalanceRange (float *pMin, float *pMax, bool
    *pbAutoSupported=NULL, const char **pUnits=NULL)=0
· STDMETHODCALLTYPE GetOutputVerticalFlip (bool *pbFlag)=0
· STDMETHODCALLTYPE SetOutputVerticalFlip (bool bFlag)=0
· STDMETHODCALLTYPE GetCustomImageMode (bool *pbFlag)=0
· STDMETHODCALLTYPE SetCustomImageMode (bool bFlag)=0
· STDMETHODCALLTYPE QueryCustomImage (unsigned int uiMode, bool *pbAvailable,
    unsigned int *puiMaxWidth, unsigned int *puiMaxHeight, unsigned int

```

- *puiPixFormats)=0
- STDMETHOD() [SetCustomImage](#) (unsigned int uiMode, unsigned int uiLeft, unsigned int uiTop, unsigned int uiWidth, unsigned int uiHeight, unsigned int uiPixelFormat)=0
- STDMETHOD() [GetCustomImage](#) (unsigned int *uiMode, unsigned int *uiLeft, unsigned int *uiTop, unsigned int *uiWidth, unsigned int *uiHeight, unsigned int *uiPixelFormat)=0

Variables

- const IID [IID_IFlyCaptureProperties](#) = {0x2bd99656, 0x1552, 0x4d98, {0xb6,0x48,0xd,0xd0,0x19,0x6d,0x16,0x49}}

9.13.1 Function Documentation

9.13.1.1 **STDMETHOD()** GetAbsBrightness (float * *pBrightness*, bool * *pbAuto* = NULL) [pure virtual]

9.13.1.2 **STDMETHOD()** GetAbsBrightnessRange (float * *pMin*, float * *pMax*, bool * *pbAutoSupported* = NULL, const char ** *pUnits* = NULL) [pure virtual]

9.13.1.3 **STDMETHOD()** GetAbsExposure (float * *plExposure*, bool * *pbAuto* = NULL) [pure virtual]

9.13.1.4 **STDMETHOD()** GetAbsExposureRange (float * *pMin*, float * *pMax*, bool * *pbAutoSupported* = NULL, const char ** *pUnits* = NULL) [pure virtual]

9.13.1.5 **STDMETHOD()** GetAbsGain (float * *plGain*, bool * *pbAuto* = NULL) [pure virtual]

9.13.1.6 **STDMETHOD()** GetAbsGainRange (float * *pMin*, float * *pMax*, bool * *pbAutoSupported* = NULL, const char ** *pUnits* = NULL) [pure virtual]

9.13.1.7 **STDMETHOD()** GetAbsGamma (float * *plGamma*, bool * *pbAuto* = NULL) [pure virtual]

9.13.1.8 **STDMETHOD()** GetAbsGammaRange (float * *pMin*, float * *pMax*, bool * *pbAutoSupported* = NULL, const char ** *pUnits* = NULL) [pure virtual]

9.13.1.9 **STDMETHOD()** GetAbsHue (float * *plHue*, bool * *pbAuto* = NULL) [pure virtual]

9.13.1.10 **STDMETHOD()** GetAbsHueRange (float * *pMin*, float * *pMax*, bool * *pbAutoSupported* = NULL, const char ** *pUnits* = NULL) [pure virtual]

9.13.1.11 **STDMETHOD()** GetAbsPan (float * *plPan*, bool * *pbAuto* = NULL) [pure virtual]

9.13.1.12 **STDMETHOD()** GetAbsPanRange (float * *pMin*, float * *pMax*, bool * *pbAutoSupported* = NULL, const char ** *pUnits* = NULL) [pure virtual]

9.13.1.13 **STDMETHOD()** GetAbsSaturation (float * *plSaturation*, bool * *pbAuto* = NULL) [pure virtual]

9.13.1.14 **STDMETHOD()** GetAbsSaturationRange (float * *pMin*, float * *pMax*, bool * *pbAutoSupported* = NULL, const char ** *pUnits* = NULL) [pure virtual]

9.13.1.15 **STDMETHOD()** GetAbsSharpness (float * *plSharpness*, bool * *pbAuto* = NULL) [pure virtual]

9.13.1.16 **STDMETHOD()** GetAbsSharpnessRange (float * *pMin*, float * *pMax*, bool * *pbAutoSupported* = NULL, const char ** *pUnits* = NULL) [pure virtual]

9.13.1.17 **STDMETHOD()** GetAbsShutter (float * *plShutter*, bool * *pbAuto* = NULL) [pure virtual]

9.13.1.18 **STDMETHOD()** GetAbsShutterRange (float * *pMin*, float * *pMax*, bool * *pbAutoSupported* = NULL, const char ** *pUnits* = NULL) [pure virtual]

9.13.1.19 **STDMETHOD()** GetAbsTilt (float * *plTilt*, bool * *pbAuto* = NULL) [pure virtual]

9.13.1.20 **STDMETHOD()** GetAbsTiltRange (float * *pMin*, float * *pMax*, bool * *pbAutoSupported* = NULL, const char ** *pUnits* = NULL) [pure virtual]

9.13.1.21 **STDMETHOD()** GetAbsWhiteBalance (float * *plWhiteBalance*, bool * *pbAuto* = NULL) [pure virtual]

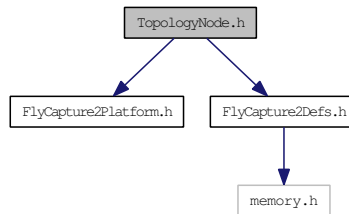
9.13.1.22 **STDMETHOD()** GetAbsWhiteBalanceRange (float * *pMin*, float * *pMax*, bool * *pbAutoSupported* = NULL, const char ** *pUnits* = NULL) [pure virtual]

9.13.1.23 **STDMETHOD()** GetBrightness (long * *pBrightness*, bool * *pbAuto* = NULL) [pure virtual]

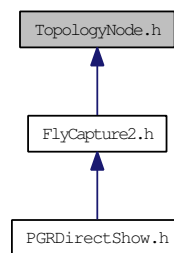
9.13.1.24 **STDMETHOD()** GetBrightnessRange (float * *pMin*, float * *pMax*, bool * *pbAutoSupported* = NULL, const char ** *pUnits* = NULL) [pure virtual]

9.14 TopologyNode.h File Reference

Include dependency graph for TopologyNode.h:



This graph shows which files directly or indirectly include this file:



Classes

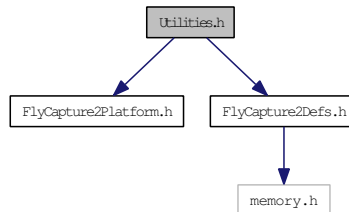
- class [TopologyNode](#)
The [TopologyNode](#) class contains topology information that can be used to generate a tree structure of all cameras and devices connected to a computer.

Namespaces

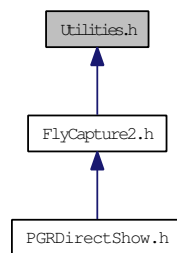
- namespace [FlyCapture2](#)

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, [49](#)
- ~BusManager
 - FlyCapture2::BusManager, [52](#)
- ~Camera
 - FlyCapture2::Camera, [63](#)
- ~CameraBase
 - FlyCapture2::CameraBase, [88](#)
- ~CameraControlDlg
 - FlyCapture2::CameraControlDlg, [107](#)
- ~CameraSelectionDlg
 - FlyCapture2::CameraSelectionDlg, [114](#)
- ~Error
 - FlyCapture2::Error, [126](#)
- ~GigECamera
 - FlyCapture2::GigECamera, [145](#)
- ~Image
 - FlyCapture2::Image, [184](#)
- ~ImageStatistics
 - FlyCapture2::ImageStatistics, [198](#)
- ~TopologyNode
 - FlyCapture2::TopologyNode, [228](#)
- absControl
 - FlyCapture2::Property, [213](#)
- absMax
 - FlyCapture2::PropertyInfo, [215](#)
- absMin
 - FlyCapture2::PropertyInfo, [215](#)
- absValSupported
 - FlyCapture2::PropertyInfo, [215](#)
- absValue
 - FlyCapture2::Property, [213](#)
- AddChild
 - FlyCapture2::TopologyNode, [229](#)
- AddPort
 - FlyCapture2::TopologyNode, [229](#)
- ADOBE_DEFLATE
 - FlyCapture2::TIFFOption, [223](#)
- ARRIVAL
 - Enumerations, [19](#)
- AssignGuidToNode
 - FlyCapture2::TopologyNode, [229](#)
- asyncBusSpeed
 - FlyCapture2::FC2Config, [130](#)
- AsyncCommandCallback
 - FlyCapture2, [45](#)
- AUTO_EXPOSURE
 - Enumerations, [25](#)
- autoManualMode
 - FlyCapture2::Property, [213](#)
- autoSupported
 - FlyCapture2::PropertyInfo, [215](#)
- available
 - FlyCapture2::EmbeddedImageInfoProperty, [124](#)
- AVIAppend
 - FlyCapture2::AVIRecorder, [49](#)
- AVIClose
 - FlyCapture2::AVIRecorder, [49](#)
- AVIOpen
 - FlyCapture2::AVIRecorder, [50](#)
- AVIOption
 - FlyCapture2::AVIOption, [47](#)
- AVIRecorder
 - FlyCapture2::AVIRecorder, [49](#)
- AVIRecorder.h, [239](#)
- BANDWIDTH_ALLOCATION_FORCE_32BITS
 - Enumerations, [18](#)
- BANDWIDTH_ALLOCATION_OFF
 - Enumerations, [18](#)
- BANDWIDTH_ALLOCATION_ON
 - Enumerations, [18](#)
- BANDWIDTH_ALLOCATION_UNSPECIFIED
 - Enumerations, [18](#)
- BANDWIDTH_ALLOCATION_UNSUPPORTED
 - Enumerations, [18](#)
- BandwidthAllocation
 - Enumerations, [18](#)
- bandwidthAllocation
 - FlyCapture2::FC2Config, [130](#)
- BayerTileFormat
 - Enumerations, [18](#)
- bayerTileFormat
 - FlyCapture2::CameraInfo, [111](#)
- BGGR
 - Enumerations, [18](#)
- binaryFile
 - FlyCapture2::PGMOption, [208](#)
 - FlyCapture2::PPMOption, [211](#)

- BLUE
 - FlyCapture2::ImageStatistics, [197](#)
- BMP
 - Enumerations, [22](#)
- BRIGHTNESS
 - Enumerations, [25](#)
- brightness
 - FlyCapture2::EmbeddedImageInfo, [123](#)
- BT_FORCE_32BITS
 - Enumerations, [18](#)
- BUFFER_FRAMES
 - Enumerations, [22](#)
- build
 - FlyCapture2::FC2Version, [132](#)
- BUS
 - FlyCapture2::TopologyNode, [228](#)
- BUS_RESET
 - Enumerations, [19](#)
- BusCallbackType
 - Enumerations, [18](#)
- busErrors
 - FlyCapture2::HostAdapterStats, [179](#)
- BusEventCallback
 - FlyCapture2, [45](#)
- BusManager
 - FlyCapture2::BusManager, [52](#)
- BusManager.h, [240](#)
- busResets
 - FlyCapture2::HostAdapterStats, [179](#)
- BusSpeed
 - Enumerations, [19](#)
- BUSSPEED_10000BASE_T
 - Enumerations, [19](#)
- BUSSPEED_1000BASE_T
 - Enumerations, [19](#)
- BUSSPEED_100BASE_T
 - Enumerations, [19](#)
- BUSSPEED_10BASE_T
 - Enumerations, [19](#)
- BUSSPEED_ANY
 - Enumerations, [19](#)
- BUSSPEED_FORCE_32BITS
 - Enumerations, [19](#)
- 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_S800
 - Enumerations, [19](#)
- BUSSPEED_S_FASTEST
 - Enumerations, [19](#)
- BUSSPEED_SPEED_UNKNOWN
 - Enumerations, [19](#)
- BYTE_ORDER_BIG_ENDIAN
 - FlyCapture2, [46](#)
- BYTE_ORDER_FORCE_32BITS
 - FlyCapture2, [46](#)
- BYTE_ORDER_LITTLE_ENDIAN
 - FlyCapture2, [46](#)
- ByteOrder
 - FlyCapture2, [46](#)
- byteOrder
 - FlyCapture2::SystemInfo, [221](#)
- CalculateStatistics
 - FlyCapture2::Image, [184](#)
- CALLBACK_TYPE_FORCE_32BITS
 - Enumerations, [19](#)
- CallbackHandle
 - FlyCapture2, [45](#)
- CAMERA
 - FlyCapture2::TopologyNode, [228](#)
- Camera
 - FlyCapture2::Camera, [63](#)
- Camera.h, [241](#)
- CameraBase
 - FlyCapture2::CameraBase, [88](#)
- CameraBase.h, [242](#)
- CameraControlDlg
 - FlyCapture2::CameraControlDlg, [107](#)
- cameraCurrents
 - FlyCapture2::CameraStats, [116](#)
- CameraInfo
 - FlyCapture2::CameraInfo, [111](#)
- cameraPowerUp
 - FlyCapture2::CameraStats, [116](#)
- CameraSelectionDlg
 - FlyCapture2::CameraSelectionDlg, [114](#)
- CameraStats
 - FlyCapture2::CameraStats, [116](#)
- cameraVoltages
 - FlyCapture2::CameraStats, [116](#)
- CCITTFAX3
 - FlyCapture2::TIFFOption, [223](#)
- CCITTFAX4
 - FlyCapture2::TIFFOption, [223](#)
- channels
 - FlyCapture2::GigEConfig, [169](#)
- chipIdHi
 - FlyCapture2::ConfigROM, [119](#)

- chipIdLo
 - FlyCapture2::ConfigROM, 119
- CollectSupportInformation
 - FlyCapture2::Error, 126
- COLOR_PROCESSING_ALGORITHM_FORCE_32BITSFlyCapture2::Image, 185
 - Enumerations, 20
- ColorProcessingAlgorithm
 - Enumerations, 19
- compression
 - FlyCapture2::TIFFOption, 224
- compressionLevel
 - FlyCapture2::PNGOption, 210
- CompressionMethod
 - FlyCapture2::TIFFOption, 223
- COMPUTER
 - FlyCapture2::TopologyNode, 228
- ConfigROM
 - FlyCapture2::ConfigROM, 119
- configROM
 - FlyCapture2::CameraInfo, 111
- Connect
 - FlyCapture2::Camera, 64
 - FlyCapture2::CameraBase, 88
 - FlyCapture2::CameraControlDlg, 108
 - FlyCapture2::GigECamera, 145
- CONNECTED_TO_CHILD
 - FlyCapture2::TopologyNode, 228
- CONNECTED_TO_PARENT
 - FlyCapture2::TopologyNode, 228
- Convert
 - FlyCapture2::Image, 184
- cpuDescription
 - FlyCapture2::SystemInfo, 221
- cycleCount
 - FlyCapture2::TimeStamp, 225
- cycleOffset
 - FlyCapture2::TimeStamp, 225
- cycleSeconds
 - FlyCapture2::TimeStamp, 225
- cycleTime
 - FlyCapture2::HostAdapterStats, 179
- DCAMFormats, 121
 - numFormats, 121
 - videoModes, 121
- DeepCopy
 - FlyCapture2::Image, 185
- DEFAULT
 - Enumerations, 20
- defaultGateway
 - FlyCapture2::CameraInfo, 111
- DEFLATE
 - FlyCapture2::TIFFOption, 223
- delay
 - FlyCapture2::StrobeControl, 217
- destinationIpAddress
 - FlyCapture2::GigEStreamChannel, 177
- DetermineBitsPerPixel
 - FlyCapture2::Image, 185
- deviceArrivals
 - FlyCapture2::HostAdapterStats, 179
- deviceRemovals
 - FlyCapture2::HostAdapterStats, 179
- DisableAll
 - FlyCapture2::ImageStatistics, 198
- Disconnect
 - FlyCapture2::Camera, 64
 - FlyCapture2::CameraBase, 89
 - FlyCapture2::CameraControlDlg, 108
 - FlyCapture2::GigECamera, 146
- DiscoverGigECameras
 - FlyCapture2::BusManager, 52
- DiscoverGigEPacketSize
 - FlyCapture2::GigECamera, 146
- doNotFragment
 - FlyCapture2::GigEStreamChannel, 177
- driverList
 - FlyCapture2::SystemInfo, 222
- driverName
 - FlyCapture2::CameraInfo, 111
- DROP_FRAMES
 - Enumerations, 22
- duration
 - FlyCapture2::StrobeControl, 217
- EDGE_SENSING
 - Enumerations, 20
- embeddedBrightness
 - FlyCapture2::ImageMetadata, 194
- embeddedExposure
 - FlyCapture2::ImageMetadata, 194
- embeddedFrameCounter
 - FlyCapture2::ImageMetadata, 194
- embeddedGain
 - FlyCapture2::ImageMetadata, 194
- embeddedGPIOPinState
 - FlyCapture2::ImageMetadata, 194
- EmbeddedImageInfoProperty
 - FlyCapture2::EmbeddedImageInfoProperty, 124
- embeddedROIPosition
 - FlyCapture2::ImageMetadata, 194
- embeddedShutter
 - FlyCapture2::ImageMetadata, 194
- embeddedStrobePattern
 - FlyCapture2::ImageMetadata, 194
- embeddedTimeStamp
 - FlyCapture2::ImageMetadata, 194
- embeddedWhiteBalance

- FlyCapture2::ImageMetadata, 194
- EnableAll
 - FlyCapture2::ImageStatistics, 198
- enabled
 - FlyCapture2::LUTData, 205
- EnableGreyOnly
 - FlyCapture2::ImageStatistics, 198
- EnableHSLOnly
 - FlyCapture2::ImageStatistics, 198
- EnableLUT
 - FlyCapture2::Camera, 64
 - FlyCapture2::CameraBase, 89
 - FlyCapture2::GigECamera, 146
- EnableRGBOnly
 - FlyCapture2::ImageStatistics, 198
- Enumerations, 12
 - ARRIVAL, 19
 - AUTO_EXPOSURE, 25
 - BANDWIDTH_ALLOCATION_FORCE_32BITS, 18
 - BANDWIDTH_ALLOCATION_OFF, 18
 - BANDWIDTH_ALLOCATION_ON, 18
 - BANDWIDTH_ALLOCATION_UNSPECIFIED, 18
 - BANDWIDTH_ALLOCATION_UNSUPPORTED, 18
 - BandwidthAllocation, 18
 - BayerTileFormat, 18
 - BGGR, 18
 - BMP, 22
 - BRIGHTNESS, 25
 - BT_FORCE_32BITS, 18
 - BUFFER_FRAMES, 22
 - BUS_RESET, 19
 - BusCallbackType, 18
 - BusSpeed, 19
 - BUSSPEED_1000BASE_T, 19
 - BUSSPEED_1000BASE_T, 19
 - BUSSPEED_100BASE_T, 19
 - BUSSPEED_10BASE_T, 19
 - BUSSPEED_ANY, 19
 - BUSSPEED_FORCE_32BITS, 19
 - BUSSPEED_S100, 19
 - BUSSPEED_S1600, 19
 - BUSSPEED_S200, 19
 - BUSSPEED_S3200, 19
 - BUSSPEED_S400, 19
 - BUSSPEED_S480, 19
 - BUSSPEED_S800, 19
 - BUSSPEED_S_FASTEST, 19
 - BUSSPEED_SPEED_UNKNOWN, 19
 - CALLBACK_TYPE_FORCE_32BITS, 19
 - COLOR_PROCESSING_ALGORITHM_FORCE_32BITS, 19
 - ColorProcessingAlgorithm, 19
 - DEFAULT, 20
 - DROP_FRAMES, 22
 - EDGE_SENSING, 20
 - ErrorType, 20
 - FOCUS, 25
 - FRAME_RATE, 25
 - FrameRate, 21
 - FRAMERATE_120, 21
 - FRAMERATE_15, 21
 - FRAMERATE_1_875, 21
 - FRAMERATE_240, 21
 - FRAMERATE_30, 21
 - FRAMERATE_3_75, 21
 - FRAMERATE_60, 21
 - FRAMERATE_7_5, 21
 - FRAMERATE_FORCE_32BITS, 21
 - FRAMERATE_FORMAT7, 21
 - FROM_FILE_EXT, 22
 - GAIN, 25
 - GAMMA, 25
 - GBRG, 18
 - GRAB_MODE_FORCE_32BITS, 22
 - GRAB_TIMEOUT_FORCE_32BITS, 22
 - GrabMode, 21
 - GrabTimeout, 22
 - GRBG, 18
 - HQ_LINEAR, 20
 - HUE, 25
 - IMAGE_FILE_FORMAT_FORCE_32BITS, 23
 - ImageFileFormat, 22
 - INTERFACE_GIGE, 23
 - INTERFACE_IEEE1394, 23
 - INTERFACE_TYPE_FORCE_32BITS, 23
 - INTERFACE_UNKNOWN, 23
 - INTERFACE_USB2, 23
 - InterfaceType, 23
 - IPP, 20
 - IRIS, 25
 - JPEG, 22
 - JPEG2000, 22
 - Mode, 23
 - MODE_0, 23
 - MODE_1, 23
 - MODE_10, 23
 - MODE_11, 23
 - MODE_12, 23
 - MODE_13, 23
 - MODE_14, 23
 - MODE_15, 23
 - MODE_16, 23
 - MODE_17, 23
 - MODE_18, 23
 - MODE_19, 23
 - MODE_2, 23
 - MODE_20, 23
 - MODE_21, 23
 - MODE_22, 24

- MODE_23, 24
- MODE_24, 24
- MODE_25, 24
- MODE_26, 24
- MODE_27, 24
- MODE_28, 24
- MODE_29, 24
- MODE_3, 23
- MODE_30, 24
- MODE_31, 24
- MODE_4, 23
- MODE_5, 23
- MODE_6, 23
- MODE_7, 23
- MODE_8, 23
- MODE_9, 23
- MODE_FORCE_32BITS, 24
- NEAREST_NEIGHBOR, 20
- NO_COLOR_PROCESSING, 20
- NONE, 18
- NUM_FRAMERATES, 21
- NUM_MODES, 24
- NUM_PIXEL_FORMATS, 24
- NUM_VIDEOMODES, 26
- PAN, 25
- PGM, 22
- PGRERROR_BUFFER_TOO_SMALL, 21
- PGRERROR_BUS_MASTER_FAILED, 20
- PGRERROR_FAILED, 20
- PGRERROR_FAILED_BUS_MASTER_CONNECTION, 20
- PGRERROR_FAILED_GUID, 20
- PGRERROR_FORCE_32BITS, 21
- PGRERROR_IIDC_FAILED, 20
- PGRERROR_IMAGE_CONSISTENCY_ERROR, 21
- PGRERROR_IMAGE_CONVERSION_FAILED, 21
- PGRERROR_IMAGE_LIBRARY_FAILURE, 21
- PGRERROR_INIT_FAILED, 20
- PGRERROR_INVALID_BUS_MANAGER, 20
- PGRERROR_INVALID_GENERATION, 20
- PGRERROR_INVALID_MODE, 20
- PGRERROR_INVALID_PACKET_SIZE, 20
- PGRERROR_INVALID_PARAMETER, 20
- PGRERROR_INVALID_SETTINGS, 20
- PGRERROR_ISOCH_ALREADY_STARTED, 21
- PGRERROR_ISOCH_BANDWIDTH_EXCEEDED, 21
- PGRERROR_ISOCH_FAILED, 21
- PGRERROR_ISOCH_NOT_STARTED, 21
- PGRERROR_ISOCH_RETRIEVE_BUFFER_FAILED, 21
- PGRERROR_ISOCH_START_FAILED, 21
- PGRERROR_ISOCH_STOP_FAILED, 21
- PGRERROR_ISOCH_SYNC_FAILED, 21
- PGRERROR_LOW_LEVEL_FAILURE, 20
- PGRERROR_LUT_FAILED, 20
- PGRERROR_MEMORY_ALLOCATION_FAILED, 20
- PGRERROR_NOT_CONNECTED, 20
- PGRERROR_NOT_FOUND, 20
- PGRERROR_NOT_IMPLEMENTED, 20
- PGRERROR_NOT_IN_FORMAT7, 20
- PGRERROR_NOT_INITIALIZED, 20
- PGRERROR_NOT_SUPPORTED, 20
- PGRERROR_OK, 20
- PGRERROR_PROPERTY_FAILED, 21
- PGRERROR_PROPERTY_NOT_PRESENT, 21
- PGRERROR_READ_REGISTER_FAILED, 21
- PGRERROR_REGISTER_FAILED, 21
- PGRERROR_STROBE_FAILED, 20
- PGRERROR_TIMEOUT, 20
- PGRERROR_TRIGGER_FAILED, 20
- PGRERROR_UNDEFINED, 20
- PGRERROR_WRITE_REGISTER_FAILED, 21
- PIXEL_FORMAT_411YUV8, 24
- PIXEL_FORMAT_422YUV8, 24
- PIXEL_FORMAT_444YUV8, 24
- PIXEL_FORMAT_BGR, 24
- PIXEL_FORMAT_BGRU, 24
- PIXEL_FORMAT_MONO12, 24
- PIXEL_FORMAT_MONO16, 24
- PIXEL_FORMAT_MONO8, 24
- PIXEL_FORMAT_RAW12, 24
- PIXEL_FORMAT_RAW16, 24
- PIXEL_FORMAT_RAW8, 24
- PIXEL_FORMAT_RGB, 24
- PIXEL_FORMAT_RGB16, 24
- PIXEL_FORMAT_RGB8, 24
- PIXEL_FORMAT_RGBU, 24
- PIXEL_FORMAT_S_MONO16, 24
- PIXEL_FORMAT_S_RGB16, 24
- PixelFormat, 24
- PNG, 22
- PPM, 22
- PROPERTY_TYPE_FORCE_32BITS, 25
- PropertyType, 24
- RAW, 23
- REMOVAL, 19
- RGGB, 18
- RIGOROUS, 20
- SATURATION, 25
- SHARPNESS, 25
- SHUTTER, 25
- TEMPERATURE, 25
- TIFF, 22
- TIFF, 25
- TIMEOUT_INFINITE, 22
- TIMEOUT_NONE, 22
- TIMEOUT_UNSPECIFIED, 22
- TRIGGER_DELAY, 25
- TRIGGER_MODE, 25
- UNSPECIFIED_GRAB_MODE, 22

- UNSPECIFIED_PIXEL_FORMAT, 24
- UNSPECIFIED_PROPERTY_TYPE, 25
- VideoMode, 25
- VIDEOMODE_1024x768RGB, 26
- VIDEOMODE_1024x768Y16, 26
- VIDEOMODE_1024x768Y8, 26
- VIDEOMODE_1024x768YUV422, 26
- VIDEOMODE_1280x960RGB, 26
- VIDEOMODE_1280x960Y16, 26
- VIDEOMODE_1280x960Y8, 26
- VIDEOMODE_1280x960YUV422, 26
- VIDEOMODE_1600x1200RGB, 26
- VIDEOMODE_1600x1200Y16, 26
- VIDEOMODE_1600x1200Y8, 26
- VIDEOMODE_1600x1200YUV422, 26
- VIDEOMODE_160x120YUV444, 25
- VIDEOMODE_320x240YUV422, 25
- VIDEOMODE_640x480RGB, 25
- VIDEOMODE_640x480Y16, 25
- VIDEOMODE_640x480Y8, 25
- VIDEOMODE_640x480YUV411, 25
- VIDEOMODE_640x480YUV422, 25
- VIDEOMODE_800x600RGB, 25
- VIDEOMODE_800x600Y16, 25
- VIDEOMODE_800x600Y8, 25
- VIDEOMODE_800x600YUV422, 25
- VIDEOMODE_FORCE_32BITS, 26
- VIDEOMODE_FORMAT7, 26
- WHITE_BALANCE, 25
- ZOOM, 25
- Error
 - FlyCapture2::Error, 126
- Error.h, 243
- ErrorType
 - Enumerations, 20
- exposure
 - FlyCapture2::EmbeddedImageInfo, 123
- FC2Config
 - FlyCapture2::FC2Config, 130
- fifoOverflows
 - FlyCapture2::HostAdapterStats, 179
- FireBusReset
 - FlyCapture2::BusManager, 53
- FireSoftwareTrigger
 - FlyCapture2::Camera, 64
 - FlyCapture2::CameraBase, 89
 - FlyCapture2::GigECamera, 146
- firmwareBuildTime
 - FlyCapture2::CameraInfo, 111
- firmwareVersion
 - FlyCapture2::CameraInfo, 111
- FlyCapture2, 35
 - AsyncCommandCallback, 45
 - BusEventCallback, 45
 - BYTE_ORDER_BIG_ENDIAN, 46
 - BYTE_ORDER_FORCE_32BITS, 46
 - BYTE_ORDER_LITTLE_ENDIAN, 46
 - ByteOrder, 46
 - CallbackHandle, 45
 - ImageEventCallback, 46
 - LINUX_X64, 46
 - LINUX_X86, 46
 - MAC, 46
 - OSType, 46
 - OSTYPE_FORCE_32BITS, 46
 - UNKNOWN_OS, 46
 - WINDOWS_X64, 46
 - WINDOWS_X86, 46
- FlyCapture2.h, 244
- FlyCapture2::AVIOption, 47
 - AVIOption, 47
 - frameRate, 47
 - reserved, 47
- FlyCapture2::AVIRecorder, 49
 - ~AVIRecorder, 49
 - AVIAppend, 49
 - AVIClose, 49
 - AVIOpen, 50
 - AVIRecorder, 49
- FlyCapture2::BusManager, 51
 - ~BusManager, 52
 - BusManager, 52
 - DiscoverGigECameras, 52
 - FireBusReset, 53
 - ForceIPAddressToCamera, 53
 - GetCameraFromIndex, 53
 - GetCameraFromIPAddress, 54
 - GetCameraFromSerialNumber, 54
 - GetCameraSerialNumberFromIndex, 54
 - GetDeviceFromIndex, 55
 - GetInterfaceTypeFromGuid, 55
 - GetNumOfCameras, 55
 - GetNumOfDevices, 56
 - GetTopology, 56
 - ReadPhyRegister, 56
 - RegisterCallback, 56
 - RescanBus, 57
 - UnregisterCallback, 57
 - WritePhyRegister, 57
- FlyCapture2::Camera, 59
 - ~Camera, 63
 - Camera, 63
 - Connect, 64
 - Disconnect, 64
 - EnableLUT, 64
 - FireSoftwareTrigger, 64
 - GetActiveLUTBank, 64

- GetCameraInfo, 65
- GetConfiguration, 65
- GetEmbeddedImageInfo, 65
- GetFormat7Configuration, 66
- GetFormat7Info, 66
- GetGPIOPinDirection, 66
- GetLUTBankInfo, 67
- GetLUTChannel, 67
- GetLUTInfo, 68
- GetMemoryChannel, 68
- GetMemoryChannelInfo, 68
- GetProperty, 69
- GetPropertyInfo, 69
- GetRegisterString, 70
- GetStrobe, 70
- GetStrobeInfo, 70
- GetTriggerDelay, 71
- GetTriggerDelayInfo, 71
- GetTriggerMode, 71
- GetTriggerModeInfo, 72
- GetVideoModeAndFrameRate, 72
- GetVideoModeAndFrameRateInfo, 73
- IsConnected, 73
- ReadRegister, 73
- ReadRegisterBlock, 74
- RestoreFromMemoryChannel, 74
- RetrieveBuffer, 74
- SaveToMemoryChannel, 75
- SetActiveLUTBank, 75
- SetCallback, 75
- SetConfiguration, 76
- SetEmbeddedImageInfo, 76
- SetFormat7Configuration, 76, 77
- SetGPIOPinDirection, 77
- SetLUTChannel, 78
- SetProperty, 78
- SetStrobe, 79
- SetTriggerDelay, 79
- SetTriggerMode, 79
- SetUserBuffers, 80
- SetVideoModeAndFrameRate, 80
- StartCapture, 81
- StartSyncCapture, 81
- StopCapture, 81
- ValidateFormat7Settings, 81
- WaitForBufferEvent, 82
- WriteRegister, 82
- WriteRegisterBlock, 83
- FlyCapture2::CameraBase, 84
 - ~CameraBase, 88
 - CameraBase, 88
 - Connect, 88
 - Disconnect, 89
 - EnableLUT, 89
 - FireSoftwareTrigger, 89
 - GetActiveLUTBank, 89
 - GetCameraInfo, 90
 - GetConfiguration, 90
 - GetEmbeddedImageInfo, 90
 - GetGPIOPinDirection, 91
 - GetLUTBankInfo, 91
 - GetLUTChannel, 91
 - GetLUTInfo, 92
 - GetMemoryChannel, 92
 - GetMemoryChannelInfo, 93
 - GetProperty, 93
 - GetPropertyInfo, 93
 - GetRegisterString, 94
 - GetStrobe, 94
 - GetStrobeInfo, 94
 - GetTriggerDelay, 95
 - GetTriggerDelayInfo, 95
 - GetTriggerMode, 96
 - GetTriggerModeInfo, 96
 - IsConnected, 96
 - m_pCameraData, 106
 - ReadRegister, 97
 - ReadRegisterBlock, 97
 - RestoreFromMemoryChannel, 97
 - RetrieveBuffer, 98
 - SaveToMemoryChannel, 98
 - SetActiveLUTBank, 99
 - SetCallback, 99
 - SetConfiguration, 99
 - SetEmbeddedImageInfo, 99
 - SetGPIOPinDirection, 100
 - SetLUTChannel, 100
 - SetProperty, 101
 - SetStrobe, 101
 - SetTriggerDelay, 102
 - SetTriggerMode, 102
 - SetUserBuffers, 102
 - StartCapture, 103
 - StartSyncCapture, 103
 - StopCapture, 104
 - WaitForBufferEvent, 104
 - WriteRegister, 105
 - WriteRegisterBlock, 105
- FlyCapture2::CameraControlDlg, 107
 - ~CameraControlDlg, 107
 - CameraControlDlg, 107
 - Connect, 108
 - Disconnect, 108
 - Hide, 108
 - IsVisible, 108
 - Show, 108
- FlyCapture2::CameraInfo, 109
 - bayerTileFormat, 111

- CameraInfo, 111
- configROM, 111
- defaultGateway, 111
- driverName, 111
- firmwareBuildTime, 111
- firmwareVersion, 111
- gigEMajorVersion, 111
- gigEMinorVersion, 111
- iidcVer, 111
- interfaceType, 111
- ipAddress, 112
- isColorCamera, 112
- macAddress, 112
- maximumBusSpeed, 112
- modelName, 112
- reserved, 112
- sensorInfo, 112
- sensorResolution, 112
- serialNumber, 112
- subnetMask, 112
- userDefinedName, 112
- vendorName, 113
- xmlURL1, 113
- xmlURL2, 113
- FlyCapture2::CameraSelectionDlg, 114
 - ~CameraSelectionDlg, 114
 - CameraSelectionDlg, 114
 - ShowModal, 114
- FlyCapture2::CameraStats, 115
 - cameraCurrents, 116
 - cameraPowerUp, 116
 - CameraStats, 116
 - cameraVoltages, 116
 - imageCorrupt, 116
 - imageDriverDropped, 116
 - imageDropped, 116
 - imageXmitFailed, 116
 - numCurrents, 116
 - numVoltages, 116
 - portErrors, 116
 - regReadFailed, 116
 - regWriteFailed, 116
 - reserved, 116
 - temperature, 116
 - timeSinceBusReset, 117
 - timeSinceInitialization, 117
 - timeStamp, 117
- FlyCapture2::ConfigROM, 118
 - chipIdHi, 119
 - chipIdLo, 119
 - ConfigROM, 119
 - nodeVendorId, 119
 - pszKeyword, 119
 - reserved, 119
 - unitSpecId, 119
 - unitSubSWVer, 119
 - unitSWVer, 119
 - vendorUniqueInfo_0, 119
 - vendorUniqueInfo_1, 119
 - vendorUniqueInfo_2, 120
 - vendorUniqueInfo_3, 120
- FlyCapture2::EmbeddedImageInfo, 122
 - brightness, 123
 - exposure, 123
 - frameCounter, 123
 - gain, 123
 - GPIOPinState, 123
 - ROIPosition, 123
 - shutter, 123
 - strobePattern, 123
 - timestamp, 123
 - whiteBalance, 123
- FlyCapture2::EmbeddedImageInfoProperty, 124
 - available, 124
 - EmbeddedImageInfoProperty, 124
 - onOff, 124
- FlyCapture2::Error, 125
 - ~Error, 126
 - CollectSupportInformation, 126
 - Error, 126
 - GetBuildDate, 126
 - GetCause, 126
 - GetDescription, 127
 - GetFilename, 127
 - GetLine, 127
 - GetType, 127
 - InternalError, 128
 - operator=, 127
 - operator==, 128
 - PrintErrorTrace, 128
- FlyCapture2::FC2Config, 129
 - asyncBusSpeed, 130
 - bandwidthAllocation, 130
 - FC2Config, 130
 - grabMode, 130
 - grabTimeout, 130
 - isochBusSpeed, 130
 - numBuffers, 130
 - numImageNotifications, 130
 - reserved, 130
- FlyCapture2::FC2Version, 132
 - build, 132
 - major, 132
 - minor, 132
 - type, 132
- FlyCapture2::Format7ImageSettings, 133
 - Format7ImageSettings, 133
 - height, 133

- mode, 133
- offsetX, 134
- offsetY, 134
- pixelFormat, 134
- reserved, 134
- width, 134
- FlyCapture2::Format7Info, 135
 - Format7Info, 136
 - imageHStepSize, 136
 - imageVStepSize, 136
 - maxHeight, 136
 - maxPacketSize, 136
 - maxWidth, 136
 - minPacketSize, 136
 - mode, 136
 - offsetHStepSize, 136
 - offsetVStepSize, 136
 - packetSize, 136
 - percentage, 137
 - pixelFormatBitField, 137
 - reserved, 137
- FlyCapture2::Format7PacketInfo, 138
 - Format7PacketInfo, 138
 - maxBytesPerPacket, 138
 - recommendedBytesPerPacket, 138
 - reserved, 138
 - unitBytesPerPacket, 138
- FlyCapture2::GigECamera, 140
 - ~GigECamera, 145
 - Connect, 145
 - Disconnect, 146
 - DiscoverGigEPacketSize, 146
 - EnableLUT, 146
 - FireSoftwareTrigger, 146
 - GetActiveLUTBank, 147
 - GetCameraInfo, 147
 - GetConfiguration, 147
 - GetEmbeddedImageInfo, 148
 - GetGigEImageBinningSettings, 148
 - GetGigEImageSettings, 148
 - GetGigEImageSettingsInfo, 148
 - GetGigEImagingMode, 149
 - GetGigEProperty, 149
 - GetGigEStreamChannelInfo, 149
 - GetGPIOPinDirection, 149
 - GetLUTBankInfo, 150
 - GetLUTChannel, 150
 - GetLUTInfo, 151
 - GetMemoryChannel, 151
 - GetMemoryChannelInfo, 151
 - GetNumStreamChannels, 152
 - GetProperty, 152
 - GetPropertyInfo, 152
 - GetRegisterString, 153
 - GetStrobe, 153
 - GetStrobeInfo, 153
 - GetTriggerDelay, 154
 - GetTriggerDelayInfo, 154
 - GetTriggerMode, 155
 - GetTriggerModeInfo, 155
 - GigECamera, 145
 - IsConnected, 155
 - QueryGigEImagingMode, 156
 - ReadGVCPMemory, 156
 - ReadGVCPRegister, 156
 - ReadGVCPRegisterBlock, 156
 - ReadRegister, 157
 - ReadRegisterBlock, 157
 - RestoreFromMemoryChannel, 157
 - RetrieveBuffer, 158
 - SaveToMemoryChannel, 158
 - SetActiveLUTBank, 159
 - SetCallback, 159
 - SetConfiguration, 159
 - SetEmbeddedImageInfo, 159
 - SetGigEImageBinningSettings, 160
 - SetGigEImageSettings, 160
 - SetGigEImagingMode, 160
 - SetGigEProperty, 161
 - SetGigEStreamChannelInfo, 161
 - SetGPIOPinDirection, 161
 - SetLUTChannel, 162
 - SetProperty, 162
 - SetStrobe, 162
 - SetTriggerDelay, 163
 - SetTriggerMode, 163
 - SetUserBuffers, 164
 - StartCapture, 164
 - StartSyncCapture, 165
 - StopCapture, 165
 - WaitForBufferEvent, 165
 - WriteGVCPMemory, 165
 - WriteGVCPRegister, 166
 - WriteGVCPRegisterBlock, 166
 - WriteRegister, 166
 - WriteRegisterBlock, 167
- FlyCapture2::GigEConfig, 168
 - channels, 169
 - GigEConfig, 169
 - numChannels, 169
- FlyCapture2::GigEImageSettings, 170
 - GigEImageSettings, 170
 - height, 170
 - offsetX, 170
 - offsetY, 170
 - pixelFormat, 171
 - reserved, 171
 - width, 171

- FlyCapture2::GigEImageSettingsInfo, 172
 - GigEImageSettingsInfo, 172
 - imageHStepSize, 172
 - imageVStepSize, 172
 - maxHeight, 173
 - maxWidth, 173
 - offsetHStepSize, 173
 - offsetVStepSize, 173
 - pixelFormatBitField, 173
 - reserved, 173
- FlyCapture2::GigEProperty, 174
 - isReadable, 174
 - isWritable, 174
 - max, 174
 - min, 174
 - propType, 174
 - value, 175
- FlyCapture2::GigEStreamChannel, 176
 - destinationIpAddress, 177
 - doNotFragment, 177
 - GigEStreamChannel, 177
 - hostPort, 177
 - interPacketDelay, 177
 - networkInterfaceIndex, 177
 - packetSize, 177
 - sourcePort, 177
- FlyCapture2::HostAdapterStats, 178
 - busErrors, 179
 - busResets, 179
 - cycleTime, 179
 - deviceArrivals, 179
 - deviceRemovals, 179
 - fifoOverflows, 179
 - gapCount, 179
 - HostAdapterStats, 179
 - numPorts, 179
 - portErrors, 179
 - vendor, 179
- FlyCapture2::Image, 180
 - ~Image, 184
 - CalculateStatistics, 184
 - Convert, 184
 - DeepCopy, 185
 - DetermineBitsPerPixel, 185
 - GetBayerTileFormat, 185
 - GetBitsPerPixel, 185
 - GetColorProcessing, 185
 - GetCols, 186
 - GetData, 186
 - GetDataSize, 186
 - GetDefaultColorProcessing, 186
 - GetDefaultOutputFormat, 186
 - GetDimensions, 187
 - GetMetadata, 187
 - GetPixelFormat, 187
 - GetRows, 187
 - GetStride, 187
 - GetTimeStamp, 188
 - Image, 183
 - Iso, 192
 - operator(), 188
 - operator=, 188
 - ReleaseBuffer, 188
 - Save, 189, 190
 - SetColorProcessing, 190
 - SetData, 191
 - SetDefaultColorProcessing, 191
 - SetDefaultOutputFormat, 191
 - SetDimensions, 192
- FlyCapture2::ImageMetadata, 193
 - embeddedBrightness, 194
 - embeddedExposure, 194
 - embeddedFrameCounter, 194
 - embeddedGain, 194
 - embeddedGPIOPinState, 194
 - embeddedROIPosition, 194
 - embeddedShutter, 194
 - embeddedStrobePattern, 194
 - embeddedTimeStamp, 194
 - embeddedWhiteBalance, 194
 - ImageMetadata, 194
 - reserved, 194
- FlyCapture2::ImageStatistics, 196
 - ~ImageStatistics, 198
 - BLUE, 197
 - DisableAll, 198
 - EnableAll, 198
 - EnableGreyOnly, 198
 - EnableHSLOnly, 198
 - EnableRGBOnly, 198
 - GetChannelStatus, 198
 - GetHistogram, 199
 - GetMean, 199
 - GetNumPixelValues, 199
 - GetPixelValueRange, 199
 - GetRange, 200
 - GetStatistics, 200
 - GREEN, 197
 - GREY, 197
 - HUE, 197
 - ImageStatistics, 198
 - ImageStatsCalculator, 201
 - LIGHTNESS, 197
 - NUM_STATISTICS_CHANNELS, 197
 - operator=, 200
 - RED, 197
 - SATURATION, 197
 - SetChannelStatus, 201

- StatisticsChannel, 197
- FlyCapture2::IPAddress, 202
 - IPAddress, 202
 - octets, 202
 - operator==, 202
- FlyCapture2::JPEGOption, 203
 - JPEGOption, 203
 - progressive, 203
 - quality, 203
 - reserved, 203
- FlyCapture2::JPG2Option, 204
 - JPG2Option, 204
 - quality, 204
 - reserved, 204
- FlyCapture2::LUTData, 205
 - enabled, 205
 - inputBitDepth, 205
 - LUTData, 205
 - numBanks, 205
 - numChannels, 206
 - numEntries, 206
 - outputBitDepth, 206
 - reserved, 206
 - supported, 206
- FlyCapture2::MACAddress, 207
 - MACAddress, 207
 - octets, 207
 - operator==, 207
- FlyCapture2::PGMOption, 208
 - binaryFile, 208
 - PGMOption, 208
 - reserved, 208
- FlyCapture2::PGRGuid, 209
 - operator==, 209
 - PGRGuid, 209
 - value, 209
- FlyCapture2::PNGOption, 210
 - compressionLevel, 210
 - interlaced, 210
 - PNGOption, 210
 - reserved, 210
- FlyCapture2::PPMOption, 211
 - binaryFile, 211
 - PPMOption, 211
 - reserved, 211
- FlyCapture2::Property, 212
 - absControl, 213
 - absValue, 213
 - autoManualMode, 213
 - onePush, 213
 - onOff, 213
 - present, 213
 - Property, 213
 - reserved, 213
 - type, 213
 - valueA, 213
 - valueB, 213
- FlyCapture2::PropertyInfo, 214
 - absMax, 215
 - absMin, 215
 - absValSupported, 215
 - autoSupported, 215
 - manualSupported, 215
 - max, 215
 - min, 215
 - onePushSupported, 215
 - onOffSupported, 215
 - present, 215
 - PropertyInfo, 215
 - pUnitAbbr, 216
 - pUnits, 216
 - readOutSupported, 216
 - reserved, 216
 - type, 216
- FlyCapture2::StrobeControl, 217
 - delay, 217
 - duration, 217
 - onOff, 217
 - polarity, 217
 - reserved, 218
 - source, 218
 - StrobeControl, 217
- FlyCapture2::StrobeInfo, 219
 - maxValue, 219
 - minValue, 219
 - onOffSupported, 219
 - polaritySupported, 220
 - present, 220
 - readOutSupported, 220
 - reserved, 220
 - source, 220
 - StrobeInfo, 219
- FlyCapture2::SystemInfo, 221
 - byteOrder, 221
 - cpuDescription, 221
 - driverList, 222
 - gpuDescription, 222
 - libraryList, 222
 - numCpuCores, 222
 - osDescription, 222
 - osType, 222
 - reserved, 222
 - screenHeight, 222
 - screenWidth, 222
 - sysMemSize, 222
- FlyCapture2::TIFFOption, 223
 - ADOBE_DEFLATE, 223
 - CCITTFAX3, 223

- CCITTFAX4, 223
- compression, 224
- CompressionMethod, 223
- DEFLATE, 223
- JPEG, 223
- LZW, 223
- NONE, 223
- PACKBITS, 223
- reserved, 224
- TIFFOption, 224
- FlyCapture2::TimeStamp, 225
 - cycleCount, 225
 - cycleOffset, 225
 - cycleSeconds, 225
 - microSeconds, 225
 - reserved, 225
 - seconds, 226
 - TimeStamp, 225
- FlyCapture2::TopologyNode, 227
 - ~TopologyNode, 228
 - AddChild, 229
 - AddPort, 229
 - AssignGuidToNode, 229
 - BUS, 228
 - CAMERA, 228
 - COMPUTER, 228
 - CONNECTED_TO_CHILD, 228
 - CONNECTED_TO_PARENT, 228
 - GetChild, 229
 - GetDeviceId, 229
 - GetGuid, 229
 - GetInterfaceType, 230
 - GetNodeType, 230
 - GetNumChildren, 230
 - GetNumPorts, 230
 - GetPortType, 230
 - NODE, 228
 - NodeType, 228
 - NOT_CONNECTED, 228
 - operator=, 230
 - PortType, 228
 - TopologyNode, 228
- FlyCapture2::TriggerMode, 231
 - mode, 231
 - onOff, 231
 - parameter, 231
 - polarity, 231
 - reserved, 231
 - source, 231
 - TriggerMode, 231
- FlyCapture2::TriggerModeInfo, 233
 - modeMask, 233
 - onOffSupported, 233
 - polaritySupported, 233
 - present, 233
 - readOutSupported, 234
 - reserved, 234
 - softwareTriggerSupported, 234
 - sourceMask, 234
 - TriggerModeInfo, 233
 - valueReadable, 234
- FlyCapture2::Utilities, 235
 - GetLibraryVersion, 235
 - GetSystemInfo, 235
 - LaunchBrowser, 235
 - LaunchCommand, 236
 - LaunchCommandAsync, 236
 - LaunchHelp, 236
- FLYCAPTURE2_API
 - FlyCapture2Platform.h, 254
- FlyCapture2Defs.h, 245
 - FULL_32BIT_VALUE, 252
 - NULL, 252
- FlyCapture2GUI.h, 253
- FlyCapture2Platform.h, 254
 - FLYCAPTURE2_API, 254
- FOCUS
 - Enumerations, 25
- ForceIPAddressToCamera
 - FlyCapture2::BusManager, 53
- Format7ImageSettings
 - FlyCapture2::Format7ImageSettings, 133
- Format7Info
 - FlyCapture2::Format7Info, 136
- Format7PacketInfo
 - FlyCapture2::Format7PacketInfo, 138
- FRAME_RATE
 - Enumerations, 25
- frameCounter
 - FlyCapture2::EmbeddedImageInfo, 123
- FrameRate
 - Enumerations, 21
- frameRate
 - FlyCapture2::AVIOption, 47
 - VideoModes, 237
- FRAMERATE_120
 - Enumerations, 21
- FRAMERATE_15
 - Enumerations, 21
- FRAMERATE_1_875
 - Enumerations, 21
- FRAMERATE_240
 - Enumerations, 21
- FRAMERATE_30
 - Enumerations, 21
- FRAMERATE_3_75
 - Enumerations, 21
- FRAMERATE_60

- Enumerations, [21](#)
- FRAMERATE_7_5
 - Enumerations, [21](#)
- FRAMERATE_FORCE_32BITS
 - Enumerations, [21](#)
- FRAMERATE_FORMAT7
 - Enumerations, [21](#)
- FROM_FILE_EXT
 - Enumerations, [22](#)
- FULL_32BIT_VALUE
 - FlyCapture2Defs.h, [252](#)
- GAIN
 - Enumerations, [25](#)
- gain
 - FlyCapture2::EmbeddedImageInfo, [123](#)
- GAMMA
 - Enumerations, [25](#)
- gapCount
 - FlyCapture2::HostAdapterStats, [179](#)
- GBRG
 - Enumerations, [18](#)
- GetAbsBrightness
 - PGRDirectShow.h, [262](#)
- GetAbsBrightnessRange
 - PGRDirectShow.h, [262](#)
- GetAbsExposure
 - PGRDirectShow.h, [262](#)
- GetAbsExposureRange
 - PGRDirectShow.h, [262](#)
- GetAbsGain
 - PGRDirectShow.h, [262](#)
- GetAbsGainRange
 - PGRDirectShow.h, [262](#)
- GetAbsGamma
 - PGRDirectShow.h, [262](#)
- GetAbsGammaRange
 - PGRDirectShow.h, [262](#)
- GetAbsHue
 - PGRDirectShow.h, [262](#)
- GetAbsHueRange
 - PGRDirectShow.h, [262](#)
- GetAbsPan
 - PGRDirectShow.h, [262](#)
- GetAbsPanRange
 - PGRDirectShow.h, [262](#)
- GetAbsSaturation
 - PGRDirectShow.h, [262](#)
- GetAbsSaturationRange
 - PGRDirectShow.h, [262](#)
- GetAbsSharpness
 - PGRDirectShow.h, [262](#)
- GetAbsSharpnessRange
 - PGRDirectShow.h, [262](#)
- GetAbsShutter
 - PGRDirectShow.h, [262](#)
- GetAbsShutterRange
 - PGRDirectShow.h, [262](#)
- GetAbsTilt
 - PGRDirectShow.h, [262](#)
- GetAbsTiltRange
 - PGRDirectShow.h, [262](#)
- GetAbsWhiteBalance
 - PGRDirectShow.h, [262](#)
- GetAbsWhiteBalanceRange
 - PGRDirectShow.h, [262](#)
- GetActiveLUTBank
 - FlyCapture2::Camera, [64](#)
 - FlyCapture2::CameraBase, [89](#)
 - FlyCapture2::GigECamera, [147](#)
- GetBayerTileFormat
 - FlyCapture2::Image, [185](#)
- GetBitsPerPixel
 - FlyCapture2::Image, [185](#)
- GetBrightness
 - PGRDirectShow.h, [262](#)
- GetBrightnessRange
 - PGRDirectShow.h, [262](#)
- GetBuildDate
 - FlyCapture2::Error, [126](#)
- GetCameraFromIndex
 - FlyCapture2::BusManager, [53](#)
- GetCameraFromIPAddress
 - FlyCapture2::BusManager, [54](#)
- GetCameraFromSerialNumber
 - FlyCapture2::BusManager, [54](#)
- GetCameraInfo
 - FlyCapture2::Camera, [65](#)
 - FlyCapture2::CameraBase, [90](#)
 - FlyCapture2::GigECamera, [147](#)
- GetCameraSerialNumberFromIndex
 - FlyCapture2::BusManager, [54](#)
- GetCause
 - FlyCapture2::Error, [126](#)
- GetChannelStatus
 - FlyCapture2::ImageStatistics, [198](#)
- GetChild
 - FlyCapture2::TopologyNode, [229](#)
- GetColorProcessing
 - FlyCapture2::Image, [185](#)
- GetCols
 - FlyCapture2::Image, [186](#)
- GetConfiguration
 - FlyCapture2::Camera, [65](#)
 - FlyCapture2::CameraBase, [90](#)
 - FlyCapture2::GigECamera, [147](#)
- GetCustomImage
 - PGRDirectShow.h, [262](#)

- GetCustomImageMode
 - PGRDirectShow.h, 262
- GetData
 - FlyCapture2::Image, 186
- GetDataSize
 - FlyCapture2::Image, 186
- GetDefaultColorProcessing
 - FlyCapture2::Image, 186
- GetDefaultOutputFormat
 - FlyCapture2::Image, 186
- GetDescription
 - FlyCapture2::Error, 127
- GetDeviceFromIndex
 - FlyCapture2::BusManager, 55
- GetDeviceId
 - FlyCapture2::TopologyNode, 229
- GetDimensions
 - FlyCapture2::Image, 187
- GetEmbeddedImageInfo
 - FlyCapture2::Camera, 65
 - FlyCapture2::CameraBase, 90
 - FlyCapture2::GigECamera, 148
- GetExposure
 - PGRDirectShow.h, 262
- GetExposureRange
 - PGRDirectShow.h, 262
- GetFilename
 - FlyCapture2::Error, 127
- GetFormat7Configuration
 - FlyCapture2::Camera, 66
- GetFormat7Info
 - FlyCapture2::Camera, 66
- GetGain
 - PGRDirectShow.h, 262
- GetGainRange
 - PGRDirectShow.h, 262
- GetGamma
 - PGRDirectShow.h, 262
- GetGammaRange
 - PGRDirectShow.h, 262
- GetGigEImageBinningSettings
 - FlyCapture2::GigECamera, 148
- GetGigEImageSettings
 - FlyCapture2::GigECamera, 148
- GetGigEImageSettingsInfo
 - FlyCapture2::GigECamera, 148
- GetGigEImagingMode
 - FlyCapture2::GigECamera, 149
- GetGigEProperty
 - FlyCapture2::GigECamera, 149
- GetGigEStreamChannelInfo
 - FlyCapture2::GigECamera, 149
- GetGPIOPinDirection
 - FlyCapture2::Camera, 66
- FlyCapture2::CameraBase, 91
- FlyCapture2::GigECamera, 149
- GetGuid
 - FlyCapture2::TopologyNode, 229
- GetHistogram
 - FlyCapture2::ImageStatistics, 199
- GetHue
 - PGRDirectShow.h, 262
- GetHueRange
 - PGRDirectShow.h, 262
- GetImageFormat
 - PGRDirectShow.h, 262
- GetInterfaceType
 - FlyCapture2::TopologyNode, 230
- GetInterfaceTypeFromGuid
 - FlyCapture2::BusManager, 55
- GetLibraryVersion
 - FlyCapture2::Utilities, 235
- GetLine
 - FlyCapture2::Error, 127
- GetLUTBankInfo
 - FlyCapture2::Camera, 67
 - FlyCapture2::CameraBase, 91
 - FlyCapture2::GigECamera, 150
- GetLUTChannel
 - FlyCapture2::Camera, 67
 - FlyCapture2::CameraBase, 91
 - FlyCapture2::GigECamera, 150
- GetLUTInfo
 - FlyCapture2::Camera, 68
 - FlyCapture2::CameraBase, 92
 - FlyCapture2::GigECamera, 151
- GetMean
 - FlyCapture2::ImageStatistics, 199
- GetMemoryChannel
 - FlyCapture2::Camera, 68
 - FlyCapture2::CameraBase, 92
 - FlyCapture2::GigECamera, 151
- GetMemoryChannelInfo
 - FlyCapture2::Camera, 68
 - FlyCapture2::CameraBase, 93
 - FlyCapture2::GigECamera, 151
- GetMetadata
 - FlyCapture2::Image, 187
- GetNodeType
 - FlyCapture2::TopologyNode, 230
- GetNumChildren
 - FlyCapture2::TopologyNode, 230
- GetNumOfCameras
 - FlyCapture2::BusManager, 55
- GetNumOfDevices
 - FlyCapture2::BusManager, 56
- GetNumPixelValues
 - FlyCapture2::ImageStatistics, 199

- GetNumPorts
 - FlyCapture2::TopologyNode, 230
- GetNumStreamChannels
 - FlyCapture2::GigECamera, 152
- GetOutputVerticalFlip
 - PGRDirectShow.h, 262
- GetPan
 - PGRDirectShow.h, 262
- GetPanRange
 - PGRDirectShow.h, 262
- GetPixelFormat
 - FlyCapture2::Image, 187
- GetPixelValueRange
 - FlyCapture2::ImageStatistics, 199
- GetPortType
 - FlyCapture2::TopologyNode, 230
- GetProperty
 - FlyCapture2::Camera, 69
 - FlyCapture2::CameraBase, 93
 - FlyCapture2::GigECamera, 152
- GetPropertyInfo
 - FlyCapture2::Camera, 69
 - FlyCapture2::CameraBase, 93
 - FlyCapture2::GigECamera, 152
- GetRange
 - FlyCapture2::ImageStatistics, 200
- GetRegisterString
 - FlyCapture2::Camera, 70
 - FlyCapture2::CameraBase, 94
 - FlyCapture2::GigECamera, 153
- GetRows
 - FlyCapture2::Image, 187
- GetSaturation
 - PGRDirectShow.h, 262
- GetSaturationRange
 - PGRDirectShow.h, 262
- GetSharpness
 - PGRDirectShow.h, 262
- GetSharpnessRange
 - PGRDirectShow.h, 262
- GetShutter
 - PGRDirectShow.h, 262
- GetShutterRange
 - PGRDirectShow.h, 262
- GetStatistics
 - FlyCapture2::ImageStatistics, 200
- GetStride
 - FlyCapture2::Image, 187
- GetStrobe
 - FlyCapture2::Camera, 70
 - FlyCapture2::CameraBase, 94
 - FlyCapture2::GigECamera, 153
- GetStrobeInfo
 - FlyCapture2::Camera, 70
 - FlyCapture2::CameraBase, 94
 - FlyCapture2::GigECamera, 153
- GetSystemInfo
 - FlyCapture2::Utilities, 235
- GetTilt
 - PGRDirectShow.h, 262
- GetTiltRange
 - PGRDirectShow.h, 262
- GetTimeStamp
 - FlyCapture2::Image, 188
- GetTopology
 - FlyCapture2::BusManager, 56
- GetTriggerDelay
 - FlyCapture2::Camera, 71
 - FlyCapture2::CameraBase, 95
 - FlyCapture2::GigECamera, 154
- GetTriggerDelayInfo
 - FlyCapture2::Camera, 71
 - FlyCapture2::CameraBase, 95
 - FlyCapture2::GigECamera, 154
- GetTriggerMode
 - FlyCapture2::Camera, 71
 - FlyCapture2::CameraBase, 96
 - FlyCapture2::GigECamera, 155
- GetTriggerModeInfo
 - FlyCapture2::Camera, 72
 - FlyCapture2::CameraBase, 96
 - FlyCapture2::GigECamera, 155
- GetType
 - FlyCapture2::Error, 127
- GetVideoModeAndFrameRate
 - FlyCapture2::Camera, 72
- GetVideoModeAndFrameRateInfo
 - FlyCapture2::Camera, 73
- GetWhiteBalance
 - PGRDirectShow.h, 262
- GetWhiteBalanceRange
 - PGRDirectShow.h, 262
- GigE specific enumerations, 27
- GigE specific structures, 31
- GigECamera
 - FlyCapture2::GigECamera, 145
- GigECamera.h, 255
- GigEConfig
 - FlyCapture2::GigEConfig, 169
- GigEEnums
 - GigEPropertyType, 27
 - HEARTBEAT, 27
 - HEARTBEAT_TIMEOUT, 27
 - PACKET_DELAY, 27
 - PACKET_SIZE, 27
- GigEImageSettings
 - FlyCapture2::GigEImageSettings, 170
- GigEImageSettingsInfo

- FlyCapture2::GigEImageSettingsInfo, 172
- gigEMajorVersion
 - FlyCapture2::CameraInfo, 111
- gigEMinorVersion
 - FlyCapture2::CameraInfo, 111
- GigEPropertyType
 - GigEEnums, 27
- GigEStreamChannel
 - FlyCapture2::GigEStreamChannel, 177
- Global constants, 11
- GlobalConstants
 - sk_maxNumPorts, 11
 - sk_maxStringLength, 11
- GPIOPinState
 - FlyCapture2::EmbeddedImageInfo, 123
- gpuDescription
 - FlyCapture2::SystemInfo, 222
- GRAB_MODE_FORCE_32BITS
 - Enumerations, 22
- GRAB_TIMEOUT_FORCE_32BITS
 - Enumerations, 22
- GrabMode
 - Enumerations, 21
- grabMode
 - FlyCapture2::FC2Config, 130
- GrabTimeout
 - Enumerations, 22
- grabTimeout
 - FlyCapture2::FC2Config, 130
- GRBG
 - Enumerations, 18
- GREEN
 - FlyCapture2::ImageStatistics, 197
- GREY
 - FlyCapture2::ImageStatistics, 197
- HEARTBEAT
 - GigEEnums, 27
- HEARTBEAT_TIMEOUT
 - GigEEnums, 27
- height
 - FlyCapture2::Format7ImageSettings, 133
 - FlyCapture2::GigEImageSettings, 170
 - VideoModes, 237
- Hide
 - FlyCapture2::CameraControlDlg, 108
- HostAdapterStats
 - FlyCapture2::HostAdapterStats, 179
- hostPost
 - FlyCapture2::GigEStreamChannel, 177
- HQ_LINEAR
 - Enumerations, 20
- HUE
 - Enumerations, 25
- FlyCapture2::ImageStatistics, 197
- IID_IFlyCaptureProperties
 - PGRDirectShow.h, 262
- IIDC specific structures, 32
- iidcVer
 - FlyCapture2::CameraInfo, 111
- Image
 - FlyCapture2::Image, 183
- Image saving structures., 33
- Image.h, 256
- IMAGE_FILE_FORMAT_FORCE_32BITS
 - Enumerations, 23
- imageCorrupt
 - FlyCapture2::CameraStats, 116
- imageDriverDropped
 - FlyCapture2::CameraStats, 116
- imageDropped
 - FlyCapture2::CameraStats, 116
- ImageEventCallback
 - FlyCapture2, 46
- ImageFileFormat
 - Enumerations, 22
- imageHStepSize
 - FlyCapture2::Format7Info, 136
 - FlyCapture2::GigEImageSettingsInfo, 172
- ImageMetadata
 - FlyCapture2::ImageMetadata, 194
- ImageStatistics
 - FlyCapture2::ImageStatistics, 198
- ImageStatistics.h, 257
- ImageStatsCalculator
 - FlyCapture2::ImageStatistics, 201
- imageVStepSize
 - FlyCapture2::Format7Info, 136
 - FlyCapture2::GigEImageSettingsInfo, 172
- imageXmitFailed
 - FlyCapture2::CameraStats, 116
- inputBitDepth
 - FlyCapture2::LUTData, 205
- INTERFACE_GIGE
 - Enumerations, 23
- INTERFACE_IEEE1394
 - Enumerations, 23
- INTERFACE_TYPE_FORCE_32BITS
 - Enumerations, 23
- INTERFACE_UNKNOWN
 - Enumerations, 23
- INTERFACE_USB2
 - Enumerations, 23
- InterfaceType
 - Enumerations, 23
- interfaceType
 - FlyCapture2::CameraInfo, 111

- interlaced
 - FlyCapture2::PNGOption, 210
- InternalError
 - FlyCapture2::Error, 128
- interPacketDelay
 - FlyCapture2::GigEStreamChannel, 177
- IPAddress
 - FlyCapture2::IPAddress, 202
- ipAddress
 - FlyCapture2::CameraInfo, 112
- IPP
 - Enumerations, 20
- IRIS
 - Enumerations, 25
- isColorCamera
 - FlyCapture2::CameraInfo, 112
- IsConnected
 - FlyCapture2::Camera, 73
 - FlyCapture2::CameraBase, 96
 - FlyCapture2::GigECamera, 155
- Iso
 - FlyCapture2::Image, 192
- isochBusSpeed
 - FlyCapture2::FC2Config, 130
- isReadable
 - FlyCapture2::GigEProperty, 174
- IsVisible
 - FlyCapture2::CameraControlDlg, 108
- isWritable
 - FlyCapture2::GigEProperty, 174
- JPEG
 - Enumerations, 22
 - FlyCapture2::TIFFOption, 223
- JPEG2000
 - Enumerations, 22
- JPEGOption
 - FlyCapture2::JPEGOption, 203
- JPG2Option
 - FlyCapture2::JPG2Option, 204
- LaunchBrowser
 - FlyCapture2::Utilities, 235
- LaunchCommand
 - FlyCapture2::Utilities, 236
- LaunchCommandAsync
 - FlyCapture2::Utilities, 236
- LaunchHelp
 - FlyCapture2::Utilities, 236
- libraryList
 - FlyCapture2::SystemInfo, 222
- LIGHTNESS
 - FlyCapture2::ImageStatistics, 197
- LINUX_X64
 - FlyCapture2, 46
- LINUX_X86
 - FlyCapture2, 46
- LUTData
 - FlyCapture2::LUTData, 205
- LZW
 - FlyCapture2::TIFFOption, 223
- m_pCameraData
 - FlyCapture2::CameraBase, 106
- MAC
 - FlyCapture2, 46
- MACAddress
 - FlyCapture2::MACAddress, 207
- macAddress
 - FlyCapture2::CameraInfo, 112
- major
 - FlyCapture2::FC2Version, 132
- manualSupported
 - FlyCapture2::PropertyInfo, 215
- max
 - FlyCapture2::GigEProperty, 174
 - FlyCapture2::PropertyInfo, 215
- maxBytesPerPacket
 - FlyCapture2::Format7PacketInfo, 138
- maxHeight
 - FlyCapture2::Format7Info, 136
 - FlyCapture2::GigEImageSettingsInfo, 173
- maximumBusSpeed
 - FlyCapture2::CameraInfo, 112
- maxPacketSize
 - FlyCapture2::Format7Info, 136
- maxValue
 - FlyCapture2::StrobeInfo, 219
- maxWidth
 - FlyCapture2::Format7Info, 136
 - FlyCapture2::GigEImageSettingsInfo, 173
- microSeconds
 - FlyCapture2::TimeStamp, 225
- min
 - FlyCapture2::GigEProperty, 174
 - FlyCapture2::PropertyInfo, 215
- minor
 - FlyCapture2::FC2Version, 132
- minPacketSize
 - FlyCapture2::Format7Info, 136
- minValue
 - FlyCapture2::StrobeInfo, 219
- Mode
 - Enumerations, 23
- mode
 - FlyCapture2::Format7ImageSettings, 133
 - FlyCapture2::Format7Info, 136
 - FlyCapture2::TriggerMode, 231

- MODE_0
 - Enumerations, [23](#)
- MODE_1
 - Enumerations, [23](#)
- MODE_10
 - Enumerations, [23](#)
- MODE_11
 - Enumerations, [23](#)
- MODE_12
 - Enumerations, [23](#)
- MODE_13
 - Enumerations, [23](#)
- MODE_14
 - Enumerations, [23](#)
- MODE_15
 - Enumerations, [23](#)
- MODE_16
 - Enumerations, [23](#)
- MODE_17
 - Enumerations, [23](#)
- MODE_18
 - Enumerations, [23](#)
- MODE_19
 - Enumerations, [23](#)
- MODE_2
 - Enumerations, [23](#)
- MODE_20
 - Enumerations, [23](#)
- MODE_21
 - Enumerations, [23](#)
- MODE_22
 - Enumerations, [24](#)
- MODE_23
 - Enumerations, [24](#)
- MODE_24
 - Enumerations, [24](#)
- MODE_25
 - Enumerations, [24](#)
- MODE_26
 - Enumerations, [24](#)
- MODE_27
 - Enumerations, [24](#)
- MODE_28
 - Enumerations, [24](#)
- MODE_29
 - Enumerations, [24](#)
- MODE_3
 - Enumerations, [23](#)
- MODE_30
 - Enumerations, [24](#)
- MODE_31
 - Enumerations, [24](#)
- MODE_4
 - Enumerations, [23](#)
- MODE_5
 - Enumerations, [23](#)
- MODE_6
 - Enumerations, [23](#)
- MODE_7
 - Enumerations, [23](#)
- MODE_8
 - Enumerations, [23](#)
- MODE_9
 - Enumerations, [23](#)
- MODE_FORCE_32BITS
 - Enumerations, [24](#)
- modelName
 - FlyCapture2::CameraInfo, [112](#)
- modeMask
 - FlyCapture2::TriggerModeInfo, [233](#)
- NEAREST_NEIGHBOR
 - Enumerations, [20](#)
- networkInterfaceIndex
 - FlyCapture2::GigEStreamChannel, [177](#)
- NO_COLOR_PROCESSING
 - Enumerations, [20](#)
- NODE
 - FlyCapture2::TopologyNode, [228](#)
- NodeType
 - FlyCapture2::TopologyNode, [228](#)
- nodeVendorId
 - FlyCapture2::ConfigROM, [119](#)
- NONE
 - Enumerations, [18](#)
 - FlyCapture2::TIFFOption, [223](#)
- NOT_CONNECTED
 - FlyCapture2::TopologyNode, [228](#)
- NULL
 - FlyCapture2Defs.h, [252](#)
- NUM_FRAMERATES
 - Enumerations, [21](#)
- NUM_MODES
 - Enumerations, [24](#)
- NUM_PIXEL_FORMATS
 - Enumerations, [24](#)
- NUM_STATISTICS_CHANNELS
 - FlyCapture2::ImageStatistics, [197](#)
- NUM_VIDEOMODES
 - Enumerations, [26](#)
- numBanks
 - FlyCapture2::LUTData, [205](#)
- numBuffers
 - FlyCapture2::FC2Config, [130](#)
- numChannels
 - FlyCapture2::GigEConfig, [169](#)
 - FlyCapture2::LUTData, [206](#)
- numCpuCores

- FlyCapture2::SystemInfo, 222
- numCurrents
 - FlyCapture2::CameraStats, 116
- numEntries
 - FlyCapture2::LUTData, 206
- numFormats
 - DCAMFormats, 121
- numImageNotifications
 - FlyCapture2::FC2Config, 130
- numPorts
 - FlyCapture2::HostAdapterStats, 179
- numVoltages
 - FlyCapture2::CameraStats, 116
- octets
 - FlyCapture2::IPAddress, 202
 - FlyCapture2::MACAddress, 207
- offsetHStepSize
 - FlyCapture2::Format7Info, 136
 - FlyCapture2::GigEImageSettingsInfo, 173
- offsetVStepSize
 - FlyCapture2::Format7Info, 136
 - FlyCapture2::GigEImageSettingsInfo, 173
- offsetX
 - FlyCapture2::Format7ImageSettings, 134
 - FlyCapture2::GigEImageSettings, 170
- offsetY
 - FlyCapture2::Format7ImageSettings, 134
 - FlyCapture2::GigEImageSettings, 170
- onePush
 - FlyCapture2::Property, 213
- onePushSupported
 - FlyCapture2::PropertyInfo, 215
- onOff
 - FlyCapture2::EmbeddedImageInfoProperty, 124
 - FlyCapture2::Property, 213
 - FlyCapture2::StrobeControl, 217
 - FlyCapture2::TriggerMode, 231
- onOffSupported
 - FlyCapture2::PropertyInfo, 215
 - FlyCapture2::StrobeInfo, 219
 - FlyCapture2::TriggerModeInfo, 233
- operator()
 - FlyCapture2::Image, 188
- operator=
 - FlyCapture2::Error, 127
 - FlyCapture2::Image, 188
 - FlyCapture2::ImageStatistics, 200
 - FlyCapture2::TopologyNode, 230
- operator==
 - FlyCapture2::Error, 128
 - FlyCapture2::IPAddress, 202
 - FlyCapture2::MACAddress, 207
 - FlyCapture2::PGRGuid, 209
- osDescription
 - FlyCapture2::SystemInfo, 222
- OSType
 - FlyCapture2, 46
- osType
 - FlyCapture2::SystemInfo, 222
- OSTYPE_FORCE_32BITS
 - FlyCapture2, 46
- outputBitDepth
 - FlyCapture2::LUTData, 206
- PACKBITS
 - FlyCapture2::TIFFOption, 223
- PACKET_DELAY
 - GigEEnums, 27
- PACKET_SIZE
 - GigEEnums, 27
- packetSize
 - FlyCapture2::Format7Info, 136
 - FlyCapture2::GigEStreamChannel, 177
- PAN
 - Enumerations, 25
- parameter
 - FlyCapture2::TriggerMode, 231
- percentage
 - FlyCapture2::Format7Info, 137
- PGM
 - Enumerations, 22
- PGMOption
 - FlyCapture2::PGMOption, 208
- PGRDirectShow.h, 258
 - GetAbsBrightness, 262
 - GetAbsBrightnessRange, 262
 - GetAbsExposure, 262
 - GetAbsExposureRange, 262
 - GetAbsGain, 262
 - GetAbsGainRange, 262
 - GetAbsGamma, 262
 - GetAbsGammaRange, 262
 - GetAbsHue, 262
 - GetAbsHueRange, 262
 - GetAbsPan, 262
 - GetAbsPanRange, 262
 - GetAbsSaturation, 262
 - GetAbsSaturationRange, 262
 - GetAbsSharpness, 262
 - GetAbsSharpnessRange, 262
 - GetAbsShutter, 262
 - GetAbsShutterRange, 262
 - GetAbsTilt, 262
 - GetAbsTiltRange, 262
 - GetAbsWhiteBalance, 262
 - GetAbsWhiteBalanceRange, 262
 - GetBrightness, 262

- GetBrightnessRange, 262
- GetCustomImage, 262
- GetCustomImageMode, 262
- GetExposure, 262
- GetExposureRange, 262
- GetGain, 262
- GetGainRange, 262
- GetGamma, 262
- GetGammaRange, 262
- GetHue, 262
- GetHueRange, 262
- GetImageFormat, 262
- GetOutputVerticalFlip, 262
- GetPan, 262
- GetPanRange, 262
- GetSaturation, 262
- GetSaturationRange, 262
- GetSharpness, 262
- GetSharpnessRange, 262
- GetShutter, 262
- GetShutterRange, 262
- GetTilt, 262
- GetTiltRange, 262
- GetWhiteBalance, 262
- GetWhiteBalanceRange, 262
- IID_IFlyCaptureProperties, 262
- QueryCustomImage, 262
- ReadRegister, 262
- SetAbsBrightness, 262
- SetAbsExposure, 262
- SetAbsGain, 262
- SetAbsGamma, 262
- SetAbsHue, 262
- SetAbsPan, 262
- SetAbsSaturation, 262
- SetAbsSharpness, 262
- SetAbsShutter, 262
- SetAbsTilt, 262
- SetAbsWhiteBalance, 262
- SetBrightness, 262
- SetCustomImage, 262
- SetCustomImageMode, 262
- SetExposure, 262
- SetGain, 262
- SetGamma, 262
- SetHue, 262
- SetImageFormat, 262
- SetOutputVerticalFlip, 262
- SetPan, 262
- SetSaturation, 262
- SetSharpness, 262
- SetShutter, 262
- SetTilt, 262
- SetWhiteBalance, 262
- WriteRegister, 262
- PGRERROR_BUFFER_TOO_SMALL
 - Enumerations, 21
- PGRERROR_BUS_MASTER_FAILED
 - Enumerations, 20
- PGRERROR_FAILED
 - Enumerations, 20
- PGRERROR_FAILED_BUS_MASTER_CONNECTION
 - Enumerations, 20
- PGRERROR_FAILED_GUID
 - Enumerations, 20
- PGRERROR_FORCE_32BITS
 - Enumerations, 21
- PGRERROR_IIDC_FAILED
 - Enumerations, 20
- PGRERROR_IMAGE_CONSISTENCY_ERROR
 - Enumerations, 21
- PGRERROR_IMAGE_CONVERSION_FAILED
 - Enumerations, 21
- PGRERROR_IMAGE_LIBRARY_FAILURE
 - Enumerations, 21
- PGRERROR_INIT_FAILED
 - Enumerations, 20
- PGRERROR_INVALID_BUS_MANAGER
 - Enumerations, 20
- PGRERROR_INVALID_GENERATION
 - Enumerations, 20
- PGRERROR_INVALID_MODE
 - Enumerations, 20
- PGRERROR_INVALID_PACKET_SIZE
 - Enumerations, 20
- PGRERROR_INVALID_PARAMETER
 - Enumerations, 20
- PGRERROR_INVALID_SETTINGS
 - Enumerations, 20
- PGRERROR_ISOCH_ALREADY_STARTED
 - Enumerations, 21
- PGRERROR_ISOCH_BANDWIDTH_EXCEEDED
 - Enumerations, 21
- PGRERROR_ISOCH_FAILED
 - Enumerations, 21
- PGRERROR_ISOCH_NOT_STARTED
 - Enumerations, 21
- PGRERROR_ISOCH_RETRIEVE_BUFFER_FAILED
 - Enumerations, 21
- PGRERROR_ISOCH_START_FAILED
 - Enumerations, 21
- PGRERROR_ISOCH_STOP_FAILED
 - Enumerations, 21
- PGRERROR_ISOCH_SYNC_FAILED
 - Enumerations, 21
- PGRERROR_LOW_LEVEL_FAILURE
 - Enumerations, 20
- PGRERROR_LUT_FAILED

- Enumerations, [20](#)
- PGRERROR_MEMORY_ALLOCATION_FAILED
 - Enumerations, [20](#)
- PGRERROR_NOT_CONNECTED
 - Enumerations, [20](#)
- PGRERROR_NOT_FOUND
 - Enumerations, [20](#)
- PGRERROR_NOT_IMPLEMENTED
 - Enumerations, [20](#)
- PGRERROR_NOT_IN_FORMAT7
 - Enumerations, [20](#)
- PGRERROR_NOT_INITIALIZED
 - Enumerations, [20](#)
- PGRERROR_NOT_SUPPORTED
 - Enumerations, [20](#)
- PGRERROR_OK
 - Enumerations, [20](#)
- PGRERROR_PROPERTY_FAILED
 - Enumerations, [21](#)
- PGRERROR_PROPERTY_NOT_PRESENT
 - Enumerations, [21](#)
- PGRERROR_READ_REGISTER_FAILED
 - Enumerations, [21](#)
- PGRERROR_REGISTER_FAILED
 - Enumerations, [21](#)
- PGRERROR_STROBE_FAILED
 - Enumerations, [20](#)
- PGRERROR_TIMEOUT
 - Enumerations, [20](#)
- PGRERROR_TRIGGER_FAILED
 - Enumerations, [20](#)
- PGRERROR_UNDEFINED
 - Enumerations, [20](#)
- PGRERROR_WRITE_REGISTER_FAILED
 - Enumerations, [21](#)
- PGRGuid
 - FlyCapture2::PGRGuid, [209](#)
- PIXEL_FORMAT_411YUV8
 - Enumerations, [24](#)
- PIXEL_FORMAT_422YUV8
 - Enumerations, [24](#)
- PIXEL_FORMAT_444YUV8
 - Enumerations, [24](#)
- PIXEL_FORMAT_BGR
 - Enumerations, [24](#)
- PIXEL_FORMAT_BGRU
 - Enumerations, [24](#)
- PIXEL_FORMAT_MONO12
 - Enumerations, [24](#)
- PIXEL_FORMAT_MONO16
 - Enumerations, [24](#)
- PIXEL_FORMAT_MONO8
 - Enumerations, [24](#)
- PIXEL_FORMAT_RAW12
 - Enumerations, [24](#)
- PIXEL_FORMAT_RAW16
 - Enumerations, [24](#)
- PIXEL_FORMAT_RAW8
 - Enumerations, [24](#)
- PIXEL_FORMAT_RGB
 - Enumerations, [24](#)
- PIXEL_FORMAT_RGB16
 - Enumerations, [24](#)
- PIXEL_FORMAT_RGB8
 - Enumerations, [24](#)
- PIXEL_FORMAT_RGBU
 - Enumerations, [24](#)
- PIXEL_FORMAT_S_MONO16
 - Enumerations, [24](#)
- PIXEL_FORMAT_S_RGB16
 - Enumerations, [24](#)
- PixelFormat
 - Enumerations, [24](#)
- pixelFormat
 - FlyCapture2::Format7ImageSettings, [134](#)
 - FlyCapture2::GigEImageSettings, [171](#)
- pixelFormatBitField
 - FlyCapture2::Format7Info, [137](#)
 - FlyCapture2::GigEImageSettingsInfo, [173](#)
- PNG
 - Enumerations, [22](#)
- PNGOption
 - FlyCapture2::PNGOption, [210](#)
- polarity
 - FlyCapture2::StrobeControl, [217](#)
 - FlyCapture2::TriggerMode, [231](#)
- polaritySupported
 - FlyCapture2::StrobeInfo, [220](#)
 - FlyCapture2::TriggerModeInfo, [233](#)
- portErrors
 - FlyCapture2::CameraStats, [116](#)
 - FlyCapture2::HostAdapterStats, [179](#)
- PortType
 - FlyCapture2::TopologyNode, [228](#)
- PPM
 - Enumerations, [22](#)
- PPMOption
 - FlyCapture2::PPMOption, [211](#)
- present
 - FlyCapture2::Property, [213](#)
 - FlyCapture2::PropertyInfo, [215](#)
 - FlyCapture2::StrobeInfo, [220](#)
 - FlyCapture2::TriggerModeInfo, [233](#)
- PrintErrorTrace
 - FlyCapture2::Error, [128](#)
- progressive
 - FlyCapture2::JPEGOption, [203](#)
- Property

- FlyCapture2::Property, 213
- PROPERTY_TYPE_FORCE_32BITS
 - Enumerations, 25
- PropertyInfo
 - FlyCapture2::PropertyInfo, 215
- PropertyType
 - Enumerations, 24
- propType
 - FlyCapture2::GigEProperty, 174
- pszKeyword
 - FlyCapture2::ConfigROM, 119
- pUnitAbbr
 - FlyCapture2::PropertyInfo, 216
- pUnits
 - FlyCapture2::PropertyInfo, 216
- quality
 - FlyCapture2::JPEGOption, 203
 - FlyCapture2::JPG2Option, 204
- QueryCustomImage
 - PGRDirectShow.h, 262
- QueryGigEImagingMode
 - FlyCapture2::GigECamera, 156
- RAW
 - Enumerations, 23
- ReadGVCPMemory
 - FlyCapture2::GigECamera, 156
- ReadGVCPRegister
 - FlyCapture2::GigECamera, 156
- ReadGVCPRegisterBlock
 - FlyCapture2::GigECamera, 156
- readOutSupported
 - FlyCapture2::PropertyInfo, 216
 - FlyCapture2::StrobeInfo, 220
 - FlyCapture2::TriggerModeInfo, 234
- ReadPhyRegister
 - FlyCapture2::BusManager, 56
- ReadRegister
 - FlyCapture2::Camera, 73
 - FlyCapture2::CameraBase, 97
 - FlyCapture2::GigECamera, 157
 - PGRDirectShow.h, 262
- ReadRegisterBlock
 - FlyCapture2::Camera, 74
 - FlyCapture2::CameraBase, 97
 - FlyCapture2::GigECamera, 157
- recommendedBytesPerPacket
 - FlyCapture2::Format7PacketInfo, 138
- RED
 - FlyCapture2::ImageStatistics, 197
- RegisterCallback
 - FlyCapture2::BusManager, 56
- regReadFailed
 - FlyCapture2::CameraStats, 116
- regWriteFailed
 - FlyCapture2::CameraStats, 116
- ReleaseBuffer
 - FlyCapture2::Image, 188
- REMOVAL
 - Enumerations, 19
- RescanBus
 - FlyCapture2::BusManager, 57
- reserved
 - FlyCapture2::AVIOption, 47
 - FlyCapture2::CameraInfo, 112
 - FlyCapture2::CameraStats, 116
 - FlyCapture2::ConfigROM, 119
 - FlyCapture2::FC2Config, 130
 - FlyCapture2::Format7ImageSettings, 134
 - FlyCapture2::Format7Info, 137
 - FlyCapture2::Format7PacketInfo, 138
 - FlyCapture2::GigEImageSettings, 171
 - FlyCapture2::GigEImageSettingsInfo, 173
 - FlyCapture2::ImageMetadata, 194
 - FlyCapture2::JPEGOption, 203
 - FlyCapture2::JPG2Option, 204
 - FlyCapture2::LUTData, 206
 - FlyCapture2::PGMOption, 208
 - FlyCapture2::PNGOption, 210
 - FlyCapture2::PPMOption, 211
 - FlyCapture2::Property, 213
 - FlyCapture2::PropertyInfo, 216
 - FlyCapture2::StrobeControl, 218
 - FlyCapture2::StrobeInfo, 220
 - FlyCapture2::SystemInfo, 222
 - FlyCapture2::TIFFOption, 224
 - FlyCapture2::TimeStamp, 225
 - FlyCapture2::TriggerMode, 231
 - FlyCapture2::TriggerModeInfo, 234
- RestoreFromMemoryChannel
 - FlyCapture2::Camera, 74
 - FlyCapture2::CameraBase, 97
 - FlyCapture2::GigECamera, 157
- RetrieveBuffer
 - FlyCapture2::Camera, 74
 - FlyCapture2::CameraBase, 98
 - FlyCapture2::GigECamera, 158
- RGGB
 - Enumerations, 18
- RIGOROUS
 - Enumerations, 20
- ROIPosition
 - FlyCapture2::EmbeddedImageInfo, 123
- SATURATION
 - Enumerations, 25
 - FlyCapture2::ImageStatistics, 197

- Save
 - FlyCapture2::Image, [189](#), [190](#)
- SaveToMemoryChannel
 - FlyCapture2::Camera, [75](#)
 - FlyCapture2::CameraBase, [98](#)
 - FlyCapture2::GigECamera, [158](#)
- screenHeight
 - FlyCapture2::SystemInfo, [222](#)
- screenWidth
 - FlyCapture2::SystemInfo, [222](#)
- seconds
 - FlyCapture2::TimeStamp, [226](#)
- sensorInfo
 - FlyCapture2::CameraInfo, [112](#)
- sensorResolution
 - FlyCapture2::CameraInfo, [112](#)
- serialNumber
 - FlyCapture2::CameraInfo, [112](#)
- SetAbsBrightness
 - PGRDirectShow.h, [262](#)
- SetAbsExposure
 - PGRDirectShow.h, [262](#)
- SetAbsGain
 - PGRDirectShow.h, [262](#)
- SetAbsGamma
 - PGRDirectShow.h, [262](#)
- SetAbsHue
 - PGRDirectShow.h, [262](#)
- SetAbsPan
 - PGRDirectShow.h, [262](#)
- SetAbsSaturation
 - PGRDirectShow.h, [262](#)
- SetAbsSharpness
 - PGRDirectShow.h, [262](#)
- SetAbsShutter
 - PGRDirectShow.h, [262](#)
- SetAbsTilt
 - PGRDirectShow.h, [262](#)
- SetAbsWhiteBalance
 - PGRDirectShow.h, [262](#)
- SetActiveLUTBank
 - FlyCapture2::Camera, [75](#)
 - FlyCapture2::CameraBase, [99](#)
 - FlyCapture2::GigECamera, [159](#)
- SetBrightness
 - PGRDirectShow.h, [262](#)
- SetCallback
 - FlyCapture2::Camera, [75](#)
 - FlyCapture2::CameraBase, [99](#)
 - FlyCapture2::GigECamera, [159](#)
- SetChannelStatus
 - FlyCapture2::ImageStatistics, [201](#)
- SetColorProcessing
 - FlyCapture2::Image, [190](#)
- SetConfiguration
 - FlyCapture2::Camera, [76](#)
 - FlyCapture2::CameraBase, [99](#)
 - FlyCapture2::GigECamera, [159](#)
- SetCustomImage
 - PGRDirectShow.h, [262](#)
- SetCustomImageMode
 - PGRDirectShow.h, [262](#)
- SetData
 - FlyCapture2::Image, [191](#)
- SetDefaultColorProcessing
 - FlyCapture2::Image, [191](#)
- SetDefaultOutputFormat
 - FlyCapture2::Image, [191](#)
- SetDimensions
 - FlyCapture2::Image, [192](#)
- SetEmbeddedImageInfo
 - FlyCapture2::Camera, [76](#)
 - FlyCapture2::CameraBase, [99](#)
 - FlyCapture2::GigECamera, [159](#)
- SetExposure
 - PGRDirectShow.h, [262](#)
- SetFormat7Configuration
 - FlyCapture2::Camera, [76](#), [77](#)
- SetGain
 - PGRDirectShow.h, [262](#)
- SetGamma
 - PGRDirectShow.h, [262](#)
- SetGigEImageBinningSettings
 - FlyCapture2::GigECamera, [160](#)
- SetGigEImageSettings
 - FlyCapture2::GigECamera, [160](#)
- SetGigEImagingMode
 - FlyCapture2::GigECamera, [160](#)
- SetGigEProperty
 - FlyCapture2::GigECamera, [161](#)
- SetGigEStreamChannelInfo
 - FlyCapture2::GigECamera, [161](#)
- SetGPIOPinDirection
 - FlyCapture2::Camera, [77](#)
 - FlyCapture2::CameraBase, [100](#)
 - FlyCapture2::GigECamera, [161](#)
- SetHue
 - PGRDirectShow.h, [262](#)
- SetImageFormat
 - PGRDirectShow.h, [262](#)
- SetLUTChannel
 - FlyCapture2::Camera, [78](#)
 - FlyCapture2::CameraBase, [100](#)
 - FlyCapture2::GigECamera, [162](#)
- SetOutputVerticalFlip
 - PGRDirectShow.h, [262](#)
- SetPan
 - PGRDirectShow.h, [262](#)

- SetProperty
 - FlyCapture2::Camera, 78
 - FlyCapture2::CameraBase, 101
 - FlyCapture2::GigECamera, 162
- SetSaturation
 - PGRDirectShow.h, 262
- SetSharpness
 - PGRDirectShow.h, 262
- SetShutter
 - PGRDirectShow.h, 262
- SetStrobe
 - FlyCapture2::Camera, 79
 - FlyCapture2::CameraBase, 101
 - FlyCapture2::GigECamera, 162
- SetTilt
 - PGRDirectShow.h, 262
- SetTriggerDelay
 - FlyCapture2::Camera, 79
 - FlyCapture2::CameraBase, 102
 - FlyCapture2::GigECamera, 163
- SetTriggerMode
 - FlyCapture2::Camera, 79
 - FlyCapture2::CameraBase, 102
 - FlyCapture2::GigECamera, 163
- SetUserBuffers
 - FlyCapture2::Camera, 80
 - FlyCapture2::CameraBase, 102
 - FlyCapture2::GigECamera, 164
- SetVideoModeAndFrameRate
 - FlyCapture2::Camera, 80
- SetWhiteBalance
 - PGRDirectShow.h, 262
- SHARPNESS
 - Enumerations, 25
- Show
 - FlyCapture2::CameraControlDlg, 108
- ShowModal
 - FlyCapture2::CameraSelectionDlg, 114
- SHUTTER
 - Enumerations, 25
- shutter
 - FlyCapture2::EmbeddedImageInfo, 123
- sk_maxNumPorts
 - GlobalConstants, 11
- sk_maxStringLength
 - GlobalConstants, 11
- softwareTriggerSupported
 - FlyCapture2::TriggerModeInfo, 234
- source
 - FlyCapture2::StrobeControl, 218
 - FlyCapture2::StrobeInfo, 220
 - FlyCapture2::TriggerMode, 231
- sourceMask
 - FlyCapture2::TriggerModeInfo, 234
- sourcePort
 - FlyCapture2::GigEStreamChannel, 177
- StartCapture
 - FlyCapture2::Camera, 81
 - FlyCapture2::CameraBase, 103
 - FlyCapture2::GigECamera, 164
- StartSyncCapture
 - FlyCapture2::Camera, 81
 - FlyCapture2::CameraBase, 103
 - FlyCapture2::GigECamera, 165
- StatisticsChannel
 - FlyCapture2::ImageStatistics, 197
- StopCapture
 - FlyCapture2::Camera, 81
 - FlyCapture2::CameraBase, 104
 - FlyCapture2::GigECamera, 165
- StrobeControl
 - FlyCapture2::StrobeControl, 217
- StrobeInfo
 - FlyCapture2::StrobeInfo, 219
- strobePattern
 - FlyCapture2::EmbeddedImageInfo, 123
- Structures, 28
 - TriggerDelay, 30
 - TriggerDelayInfo, 30
- subnetMask
 - FlyCapture2::CameraInfo, 112
- supported
 - FlyCapture2::LUTData, 206
- sysMemSize
 - FlyCapture2::SystemInfo, 222
- TEMPERATURE
 - Enumerations, 25
- temperature
 - FlyCapture2::CameraStats, 116
- TIFF
 - Enumerations, 22
- TIFFOption
 - FlyCapture2::TIFFOption, 224
- TILT
 - Enumerations, 25
- TIMEOUT_INFINITE
 - Enumerations, 22
- TIMEOUT_NONE
 - Enumerations, 22
- TIMEOUT_UNSPECIFIED
 - Enumerations, 22
- timeSinceBusReset
 - FlyCapture2::CameraStats, 117
- timeSinceInitialization
 - FlyCapture2::CameraStats, 117
- TimeStamp
 - FlyCapture2::TimeStamp, 225

- timeStamp
 - FlyCapture2::CameraStats, 117
- timestamp
 - FlyCapture2::EmbeddedImageInfo, 123
- TopologyNode
 - FlyCapture2::TopologyNode, 228
- TopologyNode.h, 263
- TRIGGER_DELAY
 - Enumerations, 25
- TRIGGER_MODE
 - Enumerations, 25
- TriggerDelay
 - Structures, 30
- TriggerDelayInfo
 - Structures, 30
- TriggerMode
 - FlyCapture2::TriggerMode, 231
- TriggerModeInfo
 - FlyCapture2::TriggerModeInfo, 233
- type
 - FlyCapture2::FC2Version, 132
 - FlyCapture2::Property, 213
 - FlyCapture2::PropertyInfo, 216
- unitBytesPerPacket
 - FlyCapture2::Format7PacketInfo, 138
- unitSpecId
 - FlyCapture2::ConfigROM, 119
- unitSubSWVer
 - FlyCapture2::ConfigROM, 119
- unitSWVer
 - FlyCapture2::ConfigROM, 119
- UNKNOWN_OS
 - FlyCapture2, 46
- UnregisterCallback
 - FlyCapture2::BusManager, 57
- UNSPECIFIED_GRAB_MODE
 - Enumerations, 22
- UNSPECIFIED_PIXEL_FORMAT
 - Enumerations, 24
- UNSPECIFIED_PROPERTY_TYPE
 - Enumerations, 25
- userDefinedName
 - FlyCapture2::CameraInfo, 112
- Utilities.h, 264
- ValidateFormat7Settings
 - FlyCapture2::Camera, 81
- value
 - FlyCapture2::GigEProperty, 175
 - FlyCapture2::PGRGuid, 209
- valueA
 - FlyCapture2::Property, 213
- valueB
 - FlyCapture2::Property, 213
- valueReadable
 - FlyCapture2::TriggerModeInfo, 234
- vendor
 - FlyCapture2::HostAdapterStats, 179
- vendorName
 - FlyCapture2::CameraInfo, 113
- vendorUniqueInfo_0
 - FlyCapture2::ConfigROM, 119
- vendorUniqueInfo_1
 - FlyCapture2::ConfigROM, 119
- vendorUniqueInfo_2
 - FlyCapture2::ConfigROM, 120
- vendorUniqueInfo_3
 - FlyCapture2::ConfigROM, 120
- VideoMode
 - Enumerations, 25
- videoMode
 - VideoModes, 237
- VIDEOMODE_1024x768RGB
 - Enumerations, 26
- VIDEOMODE_1024x768Y16
 - Enumerations, 26
- VIDEOMODE_1024x768Y8
 - Enumerations, 26
- VIDEOMODE_1024x768YUV422
 - Enumerations, 26
- VIDEOMODE_1280x960RGB
 - Enumerations, 26
- VIDEOMODE_1280x960Y16
 - Enumerations, 26
- VIDEOMODE_1280x960Y8
 - Enumerations, 26
- VIDEOMODE_1280x960YUV422
 - Enumerations, 26
- VIDEOMODE_1600x1200RGB
 - Enumerations, 26
- VIDEOMODE_1600x1200Y16
 - Enumerations, 26
- VIDEOMODE_1600x1200Y8
 - Enumerations, 26
- VIDEOMODE_1600x1200YUV422
 - Enumerations, 26
- VIDEOMODE_160x120YUV444
 - Enumerations, 25
- VIDEOMODE_320x240YUV422
 - Enumerations, 25
- VIDEOMODE_640x480RGB
 - Enumerations, 25
- VIDEOMODE_640x480Y16
 - Enumerations, 25
- VIDEOMODE_640x480Y8
 - Enumerations, 25
- VIDEOMODE_640x480YUV411

- Enumerations, [25](#)
- VIDEOMODE_640x480YUV422
 - Enumerations, [25](#)
- VIDEOMODE_800x600RGB
 - Enumerations, [25](#)
- VIDEOMODE_800x600Y16
 - Enumerations, [25](#)
- VIDEOMODE_800x600Y8
 - Enumerations, [25](#)
- VIDEOMODE_800x600YUV422
 - Enumerations, [25](#)
- VIDEOMODE_FORCE_32BITS
 - Enumerations, [26](#)
- VIDEOMODE_FORMAT7
 - Enumerations, [26](#)
- VideoModes, [237](#)
 - frameRate, [237](#)
 - height, [237](#)
 - videoMode, [237](#)
 - width, [237](#)
- videoModes
 - DCAMFormats, [121](#)
- WaitForBufferEvent
 - FlyCapture2::Camera, [82](#)
 - FlyCapture2::CameraBase, [104](#)
 - FlyCapture2::GigECamera, [165](#)
- WHITE_BALANCE
 - Enumerations, [25](#)
- whiteBalance
 - FlyCapture2::EmbeddedImageInfo, [123](#)
- width
 - FlyCapture2::Format7ImageSettings, [134](#)
 - FlyCapture2::GigEImageSettings, [171](#)
 - VideoModes, [237](#)
- WINDOWS_X64
 - FlyCapture2, [46](#)
- WINDOWS_X86
 - FlyCapture2, [46](#)
- WriteGVCPMemory
 - FlyCapture2::GigECamera, [165](#)
- WriteGVCPRegister
 - FlyCapture2::GigECamera, [166](#)
- WriteGVCPRegisterBlock
 - FlyCapture2::GigECamera, [166](#)
- WritePhyRegister
 - FlyCapture2::BusManager, [57](#)
- WriteRegister
 - FlyCapture2::Camera, [82](#)
 - FlyCapture2::CameraBase, [105](#)
 - FlyCapture2::GigECamera, [166](#)
 - PGRDirectShow.h, [262](#)
- WriteRegisterBlock
 - FlyCapture2::Camera, [83](#)
 - FlyCapture2::CameraBase, [105](#)
 - FlyCapture2::GigECamera, [167](#)
- xmlURL1
 - FlyCapture2::CameraInfo, [113](#)
- xmlURL2
 - FlyCapture2::CameraInfo, [113](#)
- ZOOM
 - Enumerations, [25](#)