



POINT GREY

# FlyCapture 2.0

## API Programming Reference

Revised October 29, 2009

**Point Grey Research Inc.**

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

## **Software Warranty**

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

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

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

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

## Software License Agreement

The FlyCapture® 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

<b>1</b>	<b>Namespace Index</b>	<b>1</b>
1.1	Namespace List . . . . .	1
<b>2</b>	<b>Class Index</b>	<b>3</b>
2.1	Class List . . . . .	3
<b>3</b>	<b>File Index</b>	<b>5</b>
3.1	File List . . . . .	5
<b>4</b>	<b>Namespace Documentation</b>	<b>7</b>
4.1	FlyCapture2 Namespace Reference . . . . .	7
4.1.1	Typedef Documentation . . . . .	16
4.1.1.1	AsyncCommandCallback . . . . .	16
4.1.1.2	BusEventCallback . . . . .	17
4.1.1.3	CallbackHandle . . . . .	17
4.1.1.4	ImageEventCallback . . . . .	17
4.1.1.5	TriggerDelay . . . . .	17
4.1.1.6	TriggerDelayInfo . . . . .	17
4.1.2	Enumeration Type Documentation . . . . .	17
4.1.2.1	BandwidthAllocation . . . . .	17
4.1.2.2	BayerTileFormat . . . . .	18
4.1.2.3	BusCallbackType . . . . .	18
4.1.2.4	BusSpeed . . . . .	18
4.1.2.5	ByteOrder . . . . .	19
4.1.2.6	ColorProcessingAlgorithm . . . . .	19
4.1.2.7	ErrorType . . . . .	19
4.1.2.8	FrameRate . . . . .	20
4.1.2.9	GrabMode . . . . .	21
4.1.2.10	GrabTimeout . . . . .	21

4.1.2.11	ImageFileFormat	21
4.1.2.12	InterfaceType	22
4.1.2.13	Mode	22
4.1.2.14	OSType	23
4.1.2.15	PixelFormat	23
4.1.2.16	PropertyType	24
4.1.2.17	VideoMode	25
4.1.3	Variable Documentation	25
4.1.3.1	sk_maxStringLength	25
<b>5</b>	<b>Class Documentation</b>	<b>27</b>
5.1	AVIOption Struct Reference	27
5.1.1	Detailed Description	27
5.1.2	Constructor & Destructor Documentation	27
5.1.2.1	AVIOption	27
5.1.3	Member Data Documentation	27
5.1.3.1	frameRate	27
5.1.3.2	reserved	28
5.2	AVIRecorder Class Reference	29
5.2.1	Detailed Description	29
5.2.2	Constructor & Destructor Documentation	29
5.2.2.1	AVIRecorder	29
5.2.2.2	~AVIRecorder	29
5.2.3	Member Function Documentation	29
5.2.3.1	AVIAppend	29
5.2.3.2	AVIClose	30
5.2.3.3	AVIOpen	30
5.3	BusManager Class Reference	31
5.3.1	Detailed Description	31
5.3.2	Constructor & Destructor Documentation	31
5.3.2.1	BusManager	31
5.3.2.2	~BusManager	32
5.3.3	Member Function Documentation	32
5.3.3.1	GetCameraFromIndex	32
5.3.3.2	GetCameraFromSerialNumber	32
5.3.3.3	GetCameraSerialNumberFromIndex	32
5.3.3.4	GetNumOfCameras	33

5.3.3.5	GetTopology . . . . .	33
5.3.3.6	RegisterCallback . . . . .	33
5.3.3.7	UnregisterCallback . . . . .	34
5.4	Camera Class Reference . . . . .	35
5.4.1	Detailed Description . . . . .	40
5.4.2	Constructor & Destructor Documentation . . . . .	40
5.4.2.1	Camera . . . . .	40
5.4.2.2	~Camera . . . . .	40
5.4.3	Member Function Documentation . . . . .	40
5.4.3.1	Connect . . . . .	40
5.4.3.2	Disconnect . . . . .	40
5.4.3.3	EnableLUT . . . . .	41
5.4.3.4	FireSoftwareTrigger . . . . .	41
5.4.3.5	GetActiveLUTBank . . . . .	41
5.4.3.6	GetCameraInfo . . . . .	41
5.4.3.7	GetConfiguration . . . . .	42
5.4.3.8	GetEmbeddedImageInfo . . . . .	42
5.4.3.9	GetFormat7Configuration . . . . .	42
5.4.3.10	GetFormat7Info . . . . .	43
5.4.3.11	GetGPIOPinDirection . . . . .	43
5.4.3.12	GetLUTBankInfo . . . . .	43
5.4.3.13	GetLUTChannel . . . . .	44
5.4.3.14	GetLUTInfo . . . . .	44
5.4.3.15	GetMemoryChannel . . . . .	44
5.4.3.16	GetMemoryChannelInfo . . . . .	45
5.4.3.17	GetProperty . . . . .	45
5.4.3.18	GetPropertyInfo . . . . .	46
5.4.3.19	GetRegisterString . . . . .	46
5.4.3.20	GetStrobe . . . . .	46
5.4.3.21	GetStrobeInfo . . . . .	47
5.4.3.22	GetTriggerDelay . . . . .	47
5.4.3.23	GetTriggerDelayInfo . . . . .	47
5.4.3.24	GetTriggerMode . . . . .	48
5.4.3.25	GetTriggerModeInfo . . . . .	48
5.4.3.26	GetVideoModeAndFrameRate . . . . .	48
5.4.3.27	GetVideoModeAndFrameRateInfo . . . . .	49

5.4.3.28	IsConnected	49
5.4.3.29	ReadRegister	49
5.4.3.30	ReadRegisterBlock	50
5.4.3.31	RestoreFromMemoryChannel	50
5.4.3.32	RetrieveBuffer	50
5.4.3.33	SaveToMemoryChannel	51
5.4.3.34	SetActiveLUTBank	51
5.4.3.35	SetConfiguration	51
5.4.3.36	SetEmbeddedImageInfo	52
5.4.3.37	SetFormat7Configuration	52
5.4.3.38	SetFormat7Configuration	52
5.4.3.39	SetGPIOPinDirection	53
5.4.3.40	SetLUTChannel	53
5.4.3.41	SetProperty	53
5.4.3.42	SetStrobe	54
5.4.3.43	SetTriggerDelay	54
5.4.3.44	SetTriggerMode	55
5.4.3.45	SetUserBuffers	55
5.4.3.46	SetVideoModeAndFrameRate	55
5.4.3.47	StartCapture	56
5.4.3.48	StartSyncCapture	56
5.4.3.49	StopCapture	57
5.4.3.50	ValidateFormat7Settings	57
5.4.3.51	WaitForBufferEvent	57
5.4.3.52	WriteRegister	58
5.4.3.53	WriteRegisterBlock	58
5.5	CameraControlDlg Class Reference	59
5.5.1	Detailed Description	59
5.5.2	Constructor & Destructor Documentation	59
5.5.2.1	CameraControlDlg	59
5.5.2.2	~CameraControlDlg	59
5.5.3	Member Function Documentation	60
5.5.3.1	Connect	60
5.5.3.2	Disconnect	60
5.5.3.3	Hide	60
5.5.3.4	IsVisible	60

5.5.3.5	Show	60
5.6	CameraInfo Struct Reference	61
5.6.1	Detailed Description	62
5.6.2	Constructor & Destructor Documentation	62
5.6.2.1	CameraInfo	62
5.6.3	Member Data Documentation	62
5.6.3.1	bayerTileFormat	62
5.6.3.2	configROM	62
5.6.3.3	driverName	62
5.6.3.4	firmwareBuildTime	62
5.6.3.5	firmwareVersion	63
5.6.3.6	iidcVer	63
5.6.3.7	interfaceType	63
5.6.3.8	isColorCamera	63
5.6.3.9	maximumBusSpeed	63
5.6.3.10	modelName	63
5.6.3.11	reserved	63
5.6.3.12	sensorInfo	63
5.6.3.13	sensorResolution	63
5.6.3.14	serialNumber	63
5.6.3.15	vendorName	63
5.7	CameraSelectionDlg Class Reference	64
5.7.1	Detailed Description	64
5.7.2	Constructor & Destructor Documentation	64
5.7.2.1	CameraSelectionDlg	64
5.7.2.2	~CameraSelectionDlg	64
5.7.3	Member Function Documentation	64
5.7.3.1	ShowModal	64
5.8	ConfigROM Struct Reference	65
5.8.1	Detailed Description	66
5.8.2	Constructor & Destructor Documentation	66
5.8.2.1	ConfigROM	66
5.8.3	Member Data Documentation	66
5.8.3.1	chipIdHi	66
5.8.3.2	chipIdLo	66
5.8.3.3	nodeVendorId	66



5.8.3.4	pszKeyword	66
5.8.3.5	reserved	66
5.8.3.6	unitSpecId	66
5.8.3.7	unitSubSWVer	66
5.8.3.8	unitSWVer	66
5.8.3.9	vendorUniqueInfo_0	66
5.8.3.10	vendorUniqueInfo_1	67
5.8.3.11	vendorUniqueInfo_2	67
5.8.3.12	vendorUniqueInfo_3	67
5.9	DCAMFormats Struct Reference	68
5.9.1	Member Data Documentation	68
5.9.1.1	numFormats	68
5.9.1.2	videoModes	68
5.10	EmbeddedImageInfo Struct Reference	69
5.10.1	Detailed Description	69
5.10.2	Member Data Documentation	70
5.10.2.1	brightness	70
5.10.2.2	exposure	70
5.10.2.3	frameCounter	70
5.10.2.4	gain	70
5.10.2.5	GPIOPinState	70
5.10.2.6	ROIPosition	70
5.10.2.7	shutter	70
5.10.2.8	strobePattern	70
5.10.2.9	timestamp	70
5.10.2.10	whiteBalance	70
5.11	EmbeddedImageInfoProperty Struct Reference	71
5.11.1	Detailed Description	71
5.11.2	Constructor & Destructor Documentation	71
5.11.2.1	EmbeddedImageInfoProperty	71
5.11.3	Member Data Documentation	71
5.11.3.1	available	71
5.11.3.2	onOff	71
5.12	Error Class Reference	72
5.12.1	Detailed Description	73
5.12.2	Constructor & Destructor Documentation	73

5.12.2.1	Error	73
5.12.2.2	Error	73
5.12.2.3	~Error	73
5.12.3	Member Function Documentation	73
5.12.3.1	CollectSupportInformation	73
5.12.3.2	GetBuildDate	73
5.12.3.3	GetCause	74
5.12.3.4	GetDescription	74
5.12.3.5	GetFilename	74
5.12.3.6	GetLine	74
5.12.3.7	GetType	74
5.12.3.8	operator!=	74
5.12.3.9	operator!=	74
5.12.3.10	operator=	75
5.12.3.11	operator==	75
5.12.3.12	operator==	75
5.12.3.13	PrintErrorTrace	75
5.12.4	Friends And Related Function Documentation	75
5.12.4.1	InternalError	75
5.13	FC2Config Struct Reference	76
5.13.1	Detailed Description	76
5.13.2	Constructor & Destructor Documentation	76
5.13.2.1	FC2Config	76
5.13.3	Member Data Documentation	76
5.13.3.1	asyncBusSpeed	76
5.13.3.2	bandwidthAllocation	77
5.13.3.3	grabMode	77
5.13.3.4	grabTimeout	77
5.13.3.5	isochBusSpeed	77
5.13.3.6	numBuffers	77
5.13.3.7	numImageNotifications	77
5.13.3.8	reserved	77
5.14	FC2Version Struct Reference	78
5.14.1	Detailed Description	78
5.14.2	Member Data Documentation	78
5.14.2.1	build	78

5.14.2.2	major	78
5.14.2.3	minor	78
5.14.2.4	type	78
5.15	Format7ImageSettings Struct Reference	79
5.15.1	Detailed Description	79
5.15.2	Constructor & Destructor Documentation	79
5.15.2.1	Format7ImageSettings	79
5.15.3	Member Data Documentation	79
5.15.3.1	height	79
5.15.3.2	mode	80
5.15.3.3	offsetX	80
5.15.3.4	offsetY	80
5.15.3.5	pixelFormat	80
5.15.3.6	reserved	80
5.15.3.7	width	80
5.16	Format7Info Struct Reference	81
5.16.1	Detailed Description	82
5.16.2	Constructor & Destructor Documentation	82
5.16.2.1	Format7Info	82
5.16.3	Member Data Documentation	82
5.16.3.1	imageHStepSize	82
5.16.3.2	imageVStepSize	82
5.16.3.3	maxHeight	82
5.16.3.4	maxPacketSize	82
5.16.3.5	maxWidth	82
5.16.3.6	minPacketSize	82
5.16.3.7	mode	82
5.16.3.8	offsetHStepSize	82
5.16.3.9	offsetVStepSize	82
5.16.3.10	packetSize	83
5.16.3.11	percentage	83
5.16.3.12	pixelFormatBitField	83
5.16.3.13	reserved	83
5.17	Format7PacketInfo Struct Reference	84
5.17.1	Detailed Description	84
5.17.2	Constructor & Destructor Documentation	84

5.17.2.1	Format7PacketInfo	84
5.17.3	Member Data Documentation	84
5.17.3.1	maxBytesPerPacket	84
5.17.3.2	recommendedBytesPerPacket	84
5.17.3.3	reserved	84
5.17.3.4	unitBytesPerPacket	85
5.18	Image Class Reference	86
5.18.1	Detailed Description	88
5.18.2	Constructor & Destructor Documentation	89
5.18.2.1	Image	89
5.18.2.2	Image	89
5.18.2.3	Image	89
5.18.2.4	Image	89
5.18.2.5	Image	90
5.18.2.6	~Image	90
5.18.3	Member Function Documentation	90
5.18.3.1	CalculateStatistics	90
5.18.3.2	Convert	90
5.18.3.3	Convert	90
5.18.3.4	DeepCopy	91
5.18.3.5	DetermineBitsPerPixel	91
5.18.3.6	GetBayerTileFormat	91
5.18.3.7	GetBitsPerPixel	91
5.18.3.8	GetColorProcessing	92
5.18.3.9	GetCols	92
5.18.3.10	GetData	92
5.18.3.11	GetData	92
5.18.3.12	GetDataSize	92
5.18.3.13	GetDefaultColorProcessing	92
5.18.3.14	GetDefaultOutputFormat	93
5.18.3.15	GetDimensions	93
5.18.3.16	GetMetadata	93
5.18.3.17	GetPixelFormat	93
5.18.3.18	GetRows	93
5.18.3.19	GetStride	94
5.18.3.20	GetTimeStamp	94

5.18.3.21	operator()	94
5.18.3.22	operator=	94
5.18.3.23	operator[]	94
5.18.3.24	ReleaseBuffer	95
5.18.3.25	Save	95
5.18.3.26	Save	95
5.18.3.27	Save	95
5.18.3.28	Save	96
5.18.3.29	Save	96
5.18.3.30	Save	96
5.18.3.31	Save	96
5.18.3.32	SetColorProcessing	97
5.18.3.33	SetData	97
5.18.3.34	SetDefaultColorProcessing	97
5.18.3.35	SetDefaultOutputFormat	98
5.18.3.36	SetDimensions	98
5.18.4	Friends And Related Function Documentation	98
5.18.4.1	Iso	98
5.19	ImageMetadata Struct Reference	99
5.19.1	Detailed Description	99
5.19.2	Constructor & Destructor Documentation	100
5.19.2.1	ImageMetadata	100
5.19.3	Member Data Documentation	100
5.19.3.1	embeddedBrightness	100
5.19.3.2	embeddedExposure	100
5.19.3.3	embeddedFrameCounter	100
5.19.3.4	embeddedGain	100
5.19.3.5	embeddedGPIOPinState	100
5.19.3.6	embeddedROIPosition	100
5.19.3.7	embeddedShutter	100
5.19.3.8	embeddedStrobePattern	100
5.19.3.9	embeddedTimeStamp	100
5.19.3.10	embeddedWhiteBalance	100
5.19.3.11	reserved	101
5.20	ImageStatistics Class Reference	102
5.20.1	Detailed Description	103

5.20.2	Member Enumeration Documentation	103
5.20.2.1	StatisticsChannel	103
5.20.3	Constructor & Destructor Documentation	104
5.20.3.1	ImageStatistics	104
5.20.3.2	~ImageStatistics	104
5.20.3.3	ImageStatistics	104
5.20.4	Member Function Documentation	104
5.20.4.1	DisableAll	104
5.20.4.2	EnableAll	104
5.20.4.3	EnableGreyOnly	104
5.20.4.4	EnableHSLOnly	104
5.20.4.5	EnableRGBOnly	104
5.20.4.6	GetChannelStatus	105
5.20.4.7	GetHistogram	105
5.20.4.8	GetMean	105
5.20.4.9	GetNumPixelValues	105
5.20.4.10	GetPixelValueRange	106
5.20.4.11	GetRange	106
5.20.4.12	GetStatistics	106
5.20.4.13	operator=	107
5.20.4.14	SetChannelStatus	107
5.20.5	Friends And Related Function Documentation	107
5.20.5.1	ImageStatsCalculator	107
5.21	JPEGOption Struct Reference	108
5.21.1	Detailed Description	108
5.21.2	Constructor & Destructor Documentation	108
5.21.2.1	JPEGOption	108
5.21.3	Member Data Documentation	108
5.21.3.1	progressive	108
5.21.3.2	quality	108
5.21.3.3	reserved	108
5.22	JPG2Option Struct Reference	109
5.22.1	Detailed Description	109
5.22.2	Constructor & Destructor Documentation	109
5.22.2.1	JPG2Option	109
5.22.3	Member Data Documentation	109

5.22.3.1	quality	109
5.22.3.2	reserved	109
5.23	LUTData Struct Reference	110
5.23.1	Detailed Description	110
5.23.2	Constructor & Destructor Documentation	110
5.23.2.1	LUTData	110
5.23.3	Member Data Documentation	110
5.23.3.1	enabled	110
5.23.3.2	inputBitDepth	110
5.23.3.3	numBanks	111
5.23.3.4	numChannels	111
5.23.3.5	numEntries	111
5.23.3.6	outputBitDepth	111
5.23.3.7	reserved	111
5.23.3.8	supported	111
5.24	PGMOption Struct Reference	112
5.24.1	Detailed Description	112
5.24.2	Constructor & Destructor Documentation	112
5.24.2.1	PGMOption	112
5.24.3	Member Data Documentation	112
5.24.3.1	binaryFile	112
5.24.3.2	reserved	112
5.25	PGRGuid Class Reference	113
5.25.1	Detailed Description	113
5.25.2	Constructor & Destructor Documentation	113
5.25.2.1	PGRGuid	113
5.25.2.2	PGRGuid	113
5.25.3	Member Function Documentation	113
5.25.3.1	operator!=	113
5.25.3.2	operator=	113
5.25.3.3	operator==	114
5.25.4	Member Data Documentation	114
5.25.4.1	value	114
5.26	PNGOption Struct Reference	115
5.26.1	Detailed Description	115
5.26.2	Constructor & Destructor Documentation	115

5.26.2.1	PNGOption	115
5.26.3	Member Data Documentation	115
5.26.3.1	compressionLevel	115
5.26.3.2	interlaced	115
5.26.3.3	reserved	115
5.27	PPMOption Struct Reference	116
5.27.1	Detailed Description	116
5.27.2	Constructor & Destructor Documentation	116
5.27.2.1	PPMOption	116
5.27.3	Member Data Documentation	116
5.27.3.1	binaryFile	116
5.27.3.2	reserved	116
5.28	Property Struct Reference	117
5.28.1	Detailed Description	117
5.28.2	Constructor & Destructor Documentation	117
5.28.2.1	Property	117
5.28.3	Member Data Documentation	117
5.28.3.1	absControl	117
5.28.3.2	absValue	118
5.28.3.3	autoManualMode	118
5.28.3.4	onePush	118
5.28.3.5	onOff	118
5.28.3.6	present	118
5.28.3.7	reserved	118
5.28.3.8	type	118
5.28.3.9	valueA	118
5.28.3.10	valueB	118
5.29	PropertyInfo Struct Reference	119
5.29.1	Detailed Description	120
5.29.2	Constructor & Destructor Documentation	120
5.29.2.1	PropertyInfo	120
5.29.3	Member Data Documentation	120
5.29.3.1	absMax	120
5.29.3.2	absMin	120
5.29.3.3	absValSupported	120
5.29.3.4	autoSupported	120



5.29.3.5	manualSupported	120
5.29.3.6	max	120
5.29.3.7	min	120
5.29.3.8	onePushSupported	120
5.29.3.9	onOffSupported	120
5.29.3.10	present	120
5.29.3.11	pUnitAbbr	121
5.29.3.12	pUnits	121
5.29.3.13	readOutSupported	121
5.29.3.14	reserved	121
5.29.3.15	type	121
5.30	StrobeControl Struct Reference	122
5.30.1	Detailed Description	122
5.30.2	Constructor & Destructor Documentation	122
5.30.2.1	StrobeControl	122
5.30.3	Member Data Documentation	122
5.30.3.1	delay	122
5.30.3.2	duration	122
5.30.3.3	onOff	122
5.30.3.4	polarity	123
5.30.3.5	reserved	123
5.30.3.6	source	123
5.31	StrobeInfo Struct Reference	124
5.31.1	Detailed Description	124
5.31.2	Constructor & Destructor Documentation	124
5.31.2.1	StrobeInfo	124
5.31.3	Member Data Documentation	124
5.31.3.1	maxValue	124
5.31.3.2	minValue	124
5.31.3.3	onOffSupported	125
5.31.3.4	polaritySupported	125
5.31.3.5	present	125
5.31.3.6	readOutSupported	125
5.31.3.7	reserved	125
5.31.3.8	source	125
5.32	SystemInfo Struct Reference	126

5.32.1 Detailed Description . . . . .	126
5.32.2 Member Data Documentation . . . . .	126
5.32.2.1 byteOrder . . . . .	126
5.32.2.2 cpuDescription . . . . .	127
5.32.2.3 driverList . . . . .	127
5.32.2.4 gpuDescription . . . . .	127
5.32.2.5 libraryList . . . . .	127
5.32.2.6 numCpuCores . . . . .	127
5.32.2.7 osDescription . . . . .	127
5.32.2.8 osType . . . . .	127
5.32.2.9 reserved . . . . .	127
5.32.2.10 screenHeight . . . . .	127
5.32.2.11 screenWidth . . . . .	127
5.32.2.12 sysMemSize . . . . .	127
5.33 TIFFOption Struct Reference . . . . .	128
5.33.1 Detailed Description . . . . .	128
5.33.2 Member Enumeration Documentation . . . . .	128
5.33.2.1 CompressionMethod . . . . .	128
5.33.3 Constructor & Destructor Documentation . . . . .	129
5.33.3.1 TIFFOption . . . . .	129
5.33.4 Member Data Documentation . . . . .	129
5.33.4.1 compression . . . . .	129
5.33.4.2 reserved . . . . .	129
5.34 TimeStamp Struct Reference . . . . .	130
5.34.1 Detailed Description . . . . .	130
5.34.2 Constructor & Destructor Documentation . . . . .	130
5.34.2.1 TimeStamp . . . . .	130
5.34.3 Member Data Documentation . . . . .	130
5.34.3.1 cycleCount . . . . .	130
5.34.3.2 cycleOffset . . . . .	130
5.34.3.3 cycleSeconds . . . . .	130
5.34.3.4 microSeconds . . . . .	130
5.34.3.5 reserved . . . . .	131
5.34.3.6 seconds . . . . .	131
5.35 TopologyNode Class Reference . . . . .	132
5.35.1 Detailed Description . . . . .	133

5.35.2	Member Enumeration Documentation	133
5.35.2.1	NodeType	133
5.35.2.2	PortType	133
5.35.3	Constructor & Destructor Documentation	133
5.35.3.1	TopologyNode	133
5.35.3.2	TopologyNode	133
5.35.3.3	~TopologyNode	133
5.35.3.4	TopologyNode	134
5.35.4	Member Function Documentation	134
5.35.4.1	AddChild	134
5.35.4.2	AddPort	134
5.35.4.3	AssignGuidToNode	134
5.35.4.4	GetChild	134
5.35.4.5	GetDeviceId	134
5.35.4.6	GetGuid	134
5.35.4.7	GetInterfaceType	134
5.35.4.8	GetNodeType	135
5.35.4.9	GetNumChildren	135
5.35.4.10	GetNumPorts	135
5.35.4.11	GetPortType	135
5.35.4.12	operator=	135
5.36	TriggerMode Struct Reference	136
5.36.1	Detailed Description	136
5.36.2	Constructor & Destructor Documentation	136
5.36.2.1	TriggerMode	136
5.36.3	Member Data Documentation	136
5.36.3.1	mode	136
5.36.3.2	onOff	136
5.36.3.3	parameter	136
5.36.3.4	polarity	136
5.36.3.5	reserved	136
5.36.3.6	source	137
5.37	TriggerModeInfo Struct Reference	138
5.37.1	Detailed Description	138
5.37.2	Constructor & Destructor Documentation	138
5.37.2.1	TriggerModeInfo	138

5.37.3	Member Data Documentation	138
5.37.3.1	modeMask	138
5.37.3.2	onOffSupported	138
5.37.3.3	polaritySupported	138
5.37.3.4	present	139
5.37.3.5	readOutSupported	139
5.37.3.6	reserved	139
5.37.3.7	softwareTriggerSupported	139
5.37.3.8	sourceMask	139
5.37.3.9	valueReadable	139
5.38	Utilities Class Reference	140
5.38.1	Detailed Description	140
5.38.2	Member Function Documentation	140
5.38.2.1	GetLibraryVersion	140
5.38.2.2	GetSystemInfo	140
5.38.2.3	LaunchBrowser	140
5.38.2.4	LaunchCommand	141
5.38.2.5	LaunchCommandAsync	141
5.38.2.6	LaunchHelp	141
5.39	VideoModes Struct Reference	142
5.39.1	Member Data Documentation	142
5.39.1.1	frameRate	142
5.39.1.2	height	142
5.39.1.3	videoMode	142
5.39.1.4	width	142
<b>6</b>	<b>File Documentation</b>	<b>143</b>
6.1	AVIRecorder.h File Reference	143
6.2	BusManager.h File Reference	144
6.3	Camera.h File Reference	145
6.4	Error.h File Reference	146
6.5	FlyCapture2.h File Reference	147
6.6	FlyCapture2Defs.h File Reference	148
6.6.1	Define Documentation	154
6.6.1.1	FULL_32BIT_VALUE	154
6.6.1.2	NULL	154
6.7	FlyCapture2GUI.h File Reference	155

6.8	FlyCapture2Platform.h File Reference . . . . .	156
6.8.1	Define Documentation . . . . .	156
6.8.1.1	FLYCAPTURE2_API . . . . .	156
6.9	Image.h File Reference . . . . .	157
6.10	ImageStatistics.h File Reference . . . . .	158
6.11	PGRDirectShow.h File Reference . . . . .	159
6.11.1	Function Documentation . . . . .	163
6.11.1.1	GetAbsBrightness . . . . .	163
6.11.1.2	GetAbsBrightnessRange . . . . .	163
6.11.1.3	GetAbsExposure . . . . .	163
6.11.1.4	GetAbsExposureRange . . . . .	163
6.11.1.5	GetAbsGain . . . . .	163
6.11.1.6	GetAbsGainRange . . . . .	163
6.11.1.7	GetAbsGamma . . . . .	163
6.11.1.8	GetAbsGammaRange . . . . .	163
6.11.1.9	GetAbsHue . . . . .	163
6.11.1.10	GetAbsHueRange . . . . .	163
6.11.1.11	GetAbsPan . . . . .	163
6.11.1.12	GetAbsPanRange . . . . .	163
6.11.1.13	GetAbsSaturation . . . . .	163
6.11.1.14	GetAbsSaturationRange . . . . .	163
6.11.1.15	GetAbsSharpness . . . . .	163
6.11.1.16	GetAbsSharpnessRange . . . . .	163
6.11.1.17	GetAbsShutter . . . . .	163
6.11.1.18	GetAbsShutterRange . . . . .	163
6.11.1.19	GetAbsTilt . . . . .	163
6.11.1.20	GetAbsTiltRange . . . . .	163
6.11.1.21	GetAbsWhiteBalance . . . . .	163
6.11.1.22	GetAbsWhiteBalanceRange . . . . .	163
6.11.1.23	GetBrightness . . . . .	163
6.11.1.24	GetBrightnessRange . . . . .	163
6.11.1.25	GetCustomImage . . . . .	163
6.11.1.26	GetCustomImageMode . . . . .	163
6.11.1.27	GetExposure . . . . .	163
6.11.1.28	GetExposureRange . . . . .	163
6.11.1.29	GetGain . . . . .	163

6.11.1.30 GetGainRange . . . . .	163
6.11.1.31 GetGamma . . . . .	163
6.11.1.32 GetGammaRange . . . . .	163
6.11.1.33 GetHue . . . . .	163
6.11.1.34 GetHueRange . . . . .	163
6.11.1.35 GetImageFormat . . . . .	163
6.11.1.36 GetOutputVerticalFlip . . . . .	163
6.11.1.37 GetPan . . . . .	163
6.11.1.38 GetPanRange . . . . .	163
6.11.1.39 GetSaturation . . . . .	163
6.11.1.40 GetSaturationRange . . . . .	163
6.11.1.41 GetSharpness . . . . .	163
6.11.1.42 GetSharpnessRange . . . . .	163
6.11.1.43 GetShutter . . . . .	163
6.11.1.44 GetShutterRange . . . . .	163
6.11.1.45 GetTilt . . . . .	163
6.11.1.46 GetTiltRange . . . . .	163
6.11.1.47 GetWhiteBalance . . . . .	163
6.11.1.48 GetWhiteBalanceRange . . . . .	163
6.11.1.49 QueryCustomImage . . . . .	163
6.11.1.50 ReadRegister . . . . .	163
6.11.1.51 SetAbsBrightness . . . . .	163
6.11.1.52 SetAbsExposure . . . . .	163
6.11.1.53 SetAbsGain . . . . .	163
6.11.1.54 SetAbsGamma . . . . .	163
6.11.1.55 SetAbsHue . . . . .	163
6.11.1.56 SetAbsPan . . . . .	163
6.11.1.57 SetAbsSaturation . . . . .	163
6.11.1.58 SetAbsSharpness . . . . .	163
6.11.1.59 SetAbsShutter . . . . .	163
6.11.1.60 SetAbsTilt . . . . .	163
6.11.1.61 SetAbsWhiteBalance . . . . .	163
6.11.1.62 SetBrightness . . . . .	163
6.11.1.63 SetCustomImage . . . . .	163
6.11.1.64 SetCustomImageMode . . . . .	163
6.11.1.65 SetExposure . . . . .	163

---

6.11.1.66 SetGain . . . . .	163
6.11.1.67 SetGamma . . . . .	163
6.11.1.68 SetHue . . . . .	163
6.11.1.69 SetImageFormat . . . . .	163
6.11.1.70 SetOutputVerticalFlip . . . . .	163
6.11.1.71 SetPan . . . . .	163
6.11.1.72 SetSaturation . . . . .	163
6.11.1.73 SetSharpness . . . . .	163
6.11.1.74 SetShutter . . . . .	163
6.11.1.75 SetTilt . . . . .	163
6.11.1.76 SetWhiteBalance . . . . .	163
6.11.1.77 WriteRegister . . . . .	163
6.11.2 Variable Documentation . . . . .	163
6.11.2.1 IID_IFlyCaptureProperties . . . . .	163
6.12 TopologyNode.h File Reference . . . . .	164
6.13 Utilities.h File Reference . . . . .	165

# Chapter 1

## Namespace Index

### 1.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">FlyCapture2</a> . . . . .	7
---------------------------------------	---





# Chapter 2

## Class Index

### 2.1 Class List

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

<a href="#">AVIOption</a> (Options for saving AVI files ) . . . . .	27
<a href="#">AVIRecorder</a> (Functionality for the user to record images to an AVI file ) . . . . .	29
<a href="#">BusManager</a> (Functionality for the user to get an <a href="#">PGRGuid</a> for a desired camera or device easily ) . . . . .	31
<a href="#">Camera</a> (The <a href="#">Camera</a> object represents a physical camera ) . . . . .	35
<a href="#">CameraControlDlg</a> (The <a href="#">CameraControlDlg</a> object represents a GTKmm dialog that provides a graphical interface to a specified camera ) . . . . .	59
<a href="#">CameraInfo</a> ( <a href="#">Camera</a> information ) . . . . .	61
<a href="#">CameraSelectionDlg</a> (The <a href="#">CameraSelectionDlg</a> object represents a GTKmm dialog that provides a graphical interface that lists the number of cameras available to the library ) . . . . .	64
<a href="#">ConfigROM</a> ( <a href="#">Camera</a> configuration ROM ) . . . . .	65
<a href="#">DCAMFormats</a> . . . . .	68
<a href="#">EmbeddedImageInfo</a> (Properties of the possible embedded image information ) . . . . .	69
<a href="#">EmbeddedImageInfoProperty</a> (Properties of a single embedded image info property ) . . . . .	71
<a href="#">Error</a> (The <a href="#">Error</a> object represents an error that is returned from the library ) . . . . .	72
<a href="#">FC2Config</a> (Configuration for a camera ) . . . . .	76
<a href="#">FC2Version</a> (The current version of the library ) . . . . .	78
<a href="#">Format7ImageSettings</a> (Format 7 image settings ) . . . . .	79
<a href="#">Format7Info</a> (Format 7 information for a single mode ) . . . . .	81
<a href="#">Format7PacketInfo</a> (Format 7 packet information ) . . . . .	84
<a href="#">Image</a> (Used to retrieve images from a camera, convert between multiple pixel formats and save images to disk ) . . . . .	86
<a href="#">ImageMetadata</a> (Metadata related to an image ) . . . . .	99
<a href="#">ImageStatistics</a> (The <a href="#">ImageStatistics</a> object represents image statistics for an image ) . . . . .	102
<a href="#">JPEGOption</a> (Options for saving JPEG image ) . . . . .	108
<a href="#">JPG2Option</a> (Options for saving JPEG2000 image ) . . . . .	109
<a href="#">LUTData</a> (Information about the camera's look up table ) . . . . .	110
<a href="#">PGMOption</a> (Options for saving PGM images ) . . . . .	112
<a href="#">PGRGuid</a> (A GUID to the camera ) . . . . .	113
<a href="#">PNGOption</a> (Options for saving PNG images ) . . . . .	115
<a href="#">PPMOption</a> (Options for saving PPM images ) . . . . .	116
<a href="#">Property</a> (A specific camera property ) . . . . .	117
<a href="#">PropertyInfo</a> (Information about a specific camera property ) . . . . .	119
<a href="#">StrobeControl</a> (A camera strobe ) . . . . .	122

<a href="#">StrobeInfo</a> (A camera strobe property ) . . . . .	124
<a href="#">SystemInfo</a> (Description of the system ) . . . . .	126
<a href="#">TIFFOption</a> (Options for saving TIFF images ) . . . . .	128
<a href="#">TimeStamp</a> (Timestamp information ) . . . . .	130
<a href="#">TopologyNode</a> (Topology information that can be used to generate a tree structure of all cameras and devices connected to a computer ) . . . . .	132
<a href="#">TriggerMode</a> (A camera trigger ) . . . . .	136
<a href="#">TriggerModeInfo</a> (Information about a camera trigger property ) . . . . .	138
<a href="#">Utilities</a> (The Utility class is generally used to query for general system information such as operating system, available memory etc ) . . . . .	140
<a href="#">VideoModes</a> . . . . .	142

# Chapter 3

## File Index

### 3.1 File List

Here is a list of all files with brief descriptions:

<a href="#">AVIRecorder.h</a>	143
<a href="#">BusManager.h</a>	144
<a href="#">Camera.h</a>	145
<a href="#">Error.h</a>	146
<a href="#">FlyCapture2.h</a>	147
<a href="#">FlyCapture2Defs.h</a>	148
<a href="#">FlyCapture2GUI.h</a>	155
<a href="#">FlyCapture2Platform.h</a>	156
<a href="#">Image.h</a>	157
<a href="#">ImageStatistics.h</a>	158
<a href="#">PGRDirectShow.h</a>	159
<a href="#">TopologyNode.h</a>	164
<a href="#">Utilities.h</a>	165



# Chapter 4

## Namespace Documentation

### 4.1 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.*

- class [Error](#)

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

- class [PGRGuid](#)

*A GUID to the camera.*

- struct [FC2Version](#)

*The current version of the library.*

- 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 [Format7ImageSettings](#)  
*Format 7 image settings.*
- struct [Format7Info](#)  
*Format 7 information for a single mode.*
- struct [Format7PacketInfo](#)  
*Format 7 packet information.*
- struct [TimeStamp](#)  
*Timestamp information.*
- struct [ConfigROM](#)  
*Camera configuration ROM.*
- struct [CameraInfo](#)  
*Camera information.*
- struct [EmbeddedImageInfoProperty](#)  
*Properties of a single embedded image info property.*
- struct [EmbeddedImageInfo](#)  
*Properties of the possible embedded image information.*
- struct [ImageMetadata](#)  
*Metadata related to an image.*
- struct [LUTData](#)  
*Information about the camera's look up table.*
- struct [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 [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)

*Bus event callback function prototype.*

- typedef void \* [CallbackHandle](#)

*Handle that is returned when registering a callback.*

- typedef void(\* [ImageEventCallback](#) )(class [Image](#) \*pImage, 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\\_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_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\_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\_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,  
    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 `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.*

### 4.1.1 Typedef Documentation

#### 4.1.1.1 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()`.

#### 4.1.1.2 typedef void(\* BusEventCallback)(void \*pParameter)

Bus event callback function prototype.

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

#### 4.1.1.3 typedef void\* CallbackHandle

Handle that is returned when registering a callback.

It is required when unregistering the callback.

#### 4.1.1.4 typedef void(\* ImageEventCallback)(class Image \*pImage, 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.

#### 4.1.1.5 typedef Property TriggerDelay

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

#### 4.1.1.6 typedef PropertyInfo TriggerDelayInfo

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

### 4.1.2 Enumeration Type Documentation

#### 4.1.2.1 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***



#### 4.1.2.2 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*

#### 4.1.2.3 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*

#### 4.1.2.4 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\_S\_FASTEST* The fastest speed available.  
*BUSSPEED\_ANY* Any speed that is available.  
*BUSSPEED\_SPEED\_UNKNOWN* Unknown bus speed.  
*BUSSPEED\_FORCE\_32BITS*

#### 4.1.2.5 enum ByteOrder

Possible byte orders.

**Enumerator:**

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

#### 4.1.2.6 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.  
***COLOR\_PROCESSING\_ALGORITHM\_FORCE\_32BITS***

#### 4.1.2.7 enum ErrorType

The error types returned by functions.

**Enumerator:**

***PGRERROR\_UNDEFINED*** Undefined.  
***PGRERROR\_OK*** Function returned with no errors.  
***PGRERROR\_FAILED*** General failure.  
***PGRERROR\_NOT\_IMPLEMENTED*** Function has not been implemented.  
***PGRERROR\_FAILED\_BUS\_MASTER\_CONNECTION*** Could not connect to Bus Master.  
***PGRERROR\_NOT\_CONNECTED*** Camera has not been connected.  
***PGRERROR\_INIT\_FAILED*** Initialization failed.  
***PGRERROR\_NOT\_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\_FORCE\_32BITS***

#### 4.1.2.8 enum FrameRate

Frame rates in frames per second.

##### Enumerator:

***FRAMERATE\_1\_875*** 1.875 fps.  
***FRAMERATE\_3\_75*** 3.75 fps.  
***FRAMERATE\_7\_5*** 7.5 fps.  
***FRAMERATE\_15*** 15 fps.

**FRAMERATE\_30** 30 fps.

**FRAMERATE\_60** 60 fps.

**FRAMERATE\_120** 120 fps.

**FRAMERATE\_240** 240 fps.

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

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

**FRAMERATE\_FORCE\_32BITS**

#### 4.1.2.9 enum GrabMode

The grab strategy employed during image transfer.

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

##### Enumerator:

**DROP\_FRAMES** Grabs the newest image in the user buffer each time the RetrieveBuffer() function is called.

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

**BUFFER\_FRAMES** Images accumulate in the user buffer, and the oldest image is grabbed for handling before being discarded.

This member can be used to guarantee that each image is seen. However, image processing time must not exceed transmission time from the camera to the buffer. Grabbing blocks if the camera has not finished transmitting the next available image. The buffer size is controlled by the numBuffers parameter in the [FC2Config](#) struct. Note that this mode is the equivalent of flycaptureLockNext in earlier versions of the FlyCapture SDK.

**UNSPECIFIED\_GRAB\_MODE** Unspecified grab mode.

**GRAB\_MODE\_FORCE\_32BITS**

#### 4.1.2.10 enum GrabTimeout

Timeout options for grabbing images.

##### Enumerator:

**TIMEOUT\_INFINITE** Wait indefinitely.

**TIMEOUT\_UNSPECIFIED** Unspecified timeout setting.

**GRAB\_TIMEOUT\_FORCE\_32BITS**

#### 4.1.2.11 enum ImageFileFormat

File formats to be used for saving images to disk.

**Enumerator:**

***FROM\_FILE\_EXT*** Determine file format from file extension.  
***PGM*** Portable gray map.  
***PPM*** Portable pixmap.  
***BMP*** Bitmap.  
***JPEG*** JPEG.  
***JPEG2000*** JPEG 2000.  
***TIFF*** Tagged image file format.  
***PNG*** Portable network graphics.  
***RAW*** Raw data.  
***IMAGE\_FILE\_FORMAT\_FORCE\_32BITS***

**4.1.2.12 enum InterfaceType**

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

**Enumerator:**

***INTERFACE\_IEEE1394*** IEEE-1394 (Includes 1394a and 1394b).  
***INTERFACE\_USB2*** USB 2.0.  
***INTERFACE\_GIGE*** GigE.  
***INTERFACE\_UNKNOWN*** Unknown interface.  
***INTERFACE\_TYPE\_FORCE\_32BITS***

**4.1.2.13 enum Mode**

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

**Enumerator:**

***MODE\_0***  
***MODE\_1***  
***MODE\_2***  
***MODE\_3***  
***MODE\_4***  
***MODE\_5***  
***MODE\_6***  
***MODE\_7***  
***MODE\_8***  
***MODE\_9***  
***MODE\_10***  
***MODE\_11***  
***MODE\_12***

*MODE\_13*

*MODE\_14*

*MODE\_15*

*MODE\_16*

*MODE\_17*

*MODE\_18*

*MODE\_19*

*MODE\_20*

*MODE\_21*

*MODE\_22*

*MODE\_23*

*MODE\_24*

*MODE\_25*

*MODE\_26*

*MODE\_27*

*MODE\_28*

*MODE\_29*

*MODE\_30*

*MODE\_31*

*NUM\_MODES* Number of modes.

*MODE\_FORCE\_32BITS*

#### 4.1.2.14 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*

#### 4.1.2.15 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.

#### 4.1.2.16 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***

#### 4.1.2.17 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***

#### 4.1.3 Variable Documentation

##### 4.1.3.1 const unsigned int sk\_maxStringLength = 512 [static]

The maximum length that is allocated for a string.





# Chapter 5

## Class Documentation

### 5.1 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.*

#### 5.1.1 Detailed Description

Options for saving AVI files.

#### 5.1.2 Constructor & Destructor Documentation

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

#### 5.1.3 Member Data Documentation

##### 5.1.3.1 float [frameRate](#)

Frame rate of the stream.

### 5.1.3.2 unsigned int reserved[256]

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

## 5.2 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.*

### 5.2.1 Detailed Description

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

### 5.2.2 Constructor & Destructor Documentation

#### 5.2.2.1 AVIRecorder ()

Default constructor.

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

Default destructor.

### 5.2.3 Member Function Documentation

#### 5.2.3.1 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.

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

Close the AVI file.

See also:

[AVIOpen\(\)](#)

Returns:

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

### 5.2.3.3 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](#)

## 5.3 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 GetNumOfCameras](#) (unsigned int \*pNumCameras)  
*Gets the number of cameras attached to the PC.*
- 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 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.*

### 5.3.1 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.

### 5.3.2 Constructor & Destructor Documentation

#### 5.3.2.1 [BusManager](#) ()

Default constructor.

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

Default destructor.

## 5.3.3 Member Function Documentation

### 5.3.3.1 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.

### 5.3.3.2 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.

### 5.3.3.3 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.

**5.3.3.4 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.

**5.3.3.5 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.

**5.3.3.6 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.



#### 5.3.3.7 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.

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

- [BusManager.h](#)

## 5.4 Camera Class Reference

The [Camera](#) object represents a physical camera.

### Public Member Functions

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

### Connection and Image Retrieval

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

- virtual [Error Connect](#) ([PGRGuid](#) \*pGuid=NULL)  
*Connects the camera object to the camera specified by the GUID.*
- 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 StartCapture](#) ([ImageEventCallback](#) callbackFn=NULL, 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.*

- static [Error StartSyncCapture](#) (int numCameras, const [Camera](#) \*\*ppCameras, const [ImageEventCallback](#) \*pCallbackFns=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)  
*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.*

## General Purpose Input / Output

These functions deal with general GPIO pin control on 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.*

## Trigger

These functions deal with trigger control on the camera.

- 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.*

## Strobe

These functions deal with strobe control on 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.*

## 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.*

## Look Up Table

These functions deal with Look Up Table control on 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.*

## Memory Channels

These functions deal with memory channel control on 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.*

## Embedded Image Information

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

- 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.*

## Register Operation

These functions deal with register operation on 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 const char \* [GetRegisterString](#) (unsigned int registerVal)  
*Returns a text representation of the register value.*

### 5.4.1 Detailed Description

The [Camera](#) object represents a physical camera.

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

### 5.4.2 Constructor & Destructor Documentation

#### 5.4.2.1 Camera ()

Default constructor.

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

Default destructor.

### 5.4.3 Member Function Documentation

#### 5.4.3.1 virtual Error Connect (PGRGuid \* *pGuid* = NULL) [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.

#### 5.4.3.2 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.

**5.4.3.3 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.

**5.4.3.4 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.

**5.4.3.5 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.

**5.4.3.6 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.



#### 5.4.3.7 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.

#### 5.4.3.8 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.

#### 5.4.3.9 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.

**5.4.3.10 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.

**5.4.3.11 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.

**5.4.3.12 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.

**5.4.3.13 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.

**5.4.3.14 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.

**5.4.3.15 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.

**5.4.3.16 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.

**5.4.3.17 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.

#### 5.4.3.18 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.

#### 5.4.3.19 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.

#### 5.4.3.20 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.

**5.4.3.21 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.

**5.4.3.22 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.

**5.4.3.23 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.

#### 5.4.3.24 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.

#### 5.4.3.25 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.

#### 5.4.3.26 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.

#### 5.4.3.27 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.

#### 5.4.3.28 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.

#### 5.4.3.29 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.

#### 5.4.3.30 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.

#### 5.4.3.31 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.

#### 5.4.3.32 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.

**5.4.3.33 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.

**5.4.3.34 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.

**5.4.3.35 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.

**5.4.3.36 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.

**5.4.3.37 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.

**5.4.3.38 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.

**5.4.3.39 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.

**5.4.3.40 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.

**5.4.3.41 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.

**5.4.3.42** `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.

**5.4.3.43** `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.

#### 5.4.3.44 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.

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

Specify user allocated buffers to use as image data buffers.

##### Parameters:

*pMemBuffers* Pointer to memory buffers to be written to. The size of the data being should be equal to (size \* numBuffers) or larger.  
*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.

#### 5.4.3.46 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.

**5.4.3.47 virtual Error StartCapture (ImageEventCallback callbackFn = NULL, 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 to the callback function.

See also:

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

Returns:

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

**5.4.3.48 static Error StartSyncCapture (int numCameras, const Camera \*\* ppCameras, const ImageEventCallback \* pCallbackFns = 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.

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.

See also:

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

**Returns:**

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

**5.4.3.49 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.

**5.4.3.50 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.

**5.4.3.51 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.

**5.4.3.52 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.

**5.4.3.53 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.

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

- [Camera.h](#)

## 5.5 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](#) ([Camera](#) \*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.*

### 5.5.1 Detailed Description

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

### 5.5.2 Constructor & Destructor Documentation

#### 5.5.2.1 CameraControlDlg ()

Default constructor.

#### 5.5.2.2 ~CameraControlDlg ()

Default destructor.

### 5.5.3 Member Function Documentation

#### 5.5.3.1 void Connect (Camera \* *pCamera*)

Connect dialog to a camera.

**Parameters:**

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

#### 5.5.3.2 void Disconnect ()

Disconnect a connected camera from the dialog.

#### 5.5.3.3 void Hide ()

Hide the dialog.

#### 5.5.3.4 bool IsVisible ()

Get the visibility of the dialog.

**Returns:**

Whether the dialog is visible.

#### 5.5.3.5 void Show ()

Show the dialog.

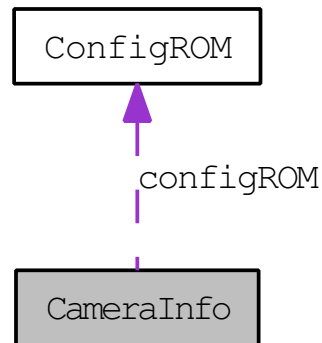
The documentation for this class was generated from the following file:

- [FlyCapture2GUI.h](#)

## 5.6 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.*

- unsigned int [iideVer](#)

*DCAM version.*

- [BusSpeed](#) [maximumBusSpeed](#)

*Maximum bus speed.*

- [BayerTileFormat](#) [bayerTileFormat](#)

*Bayer tile format.*

- [ConfigROM](#) [configROM](#)

*Configuration ROM data.*

- unsigned int [reserved](#) [16]

*Reserved for future use.*

### 5.6.1 Detailed Description

[Camera](#) information.

### 5.6.2 Constructor & Destructor Documentation

#### 5.6.2.1 [CameraInfo](#) () [[inline](#)]

### 5.6.3 Member Data Documentation

#### 5.6.3.1 [BayerTileFormat](#) [bayerTileFormat](#)

Bayer tile format.

#### 5.6.3.2 [ConfigROM](#) [configROM](#)

Configuration ROM data.

#### 5.6.3.3 char [driverName](#)[[sk\\_maxStringLength](#)]

Driver name of driver being used.

#### 5.6.3.4 char [firmwareBuildTime](#)[[sk\\_maxStringLength](#)]

Firmware build time.

**5.6.3.5 char firmwareVersion[sk\_maxStringLength]**

Firmware version of camera.

**5.6.3.6 unsigned int iidcVer**

DCAM version.

**5.6.3.7 InterfaceType interfaceType**

Interface type.

**5.6.3.8 bool isColorCamera**

Flag indicating if this is a color camera.

**5.6.3.9 BusSpeed maximumBusSpeed**

Maximum bus speed.

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

Device model name.

**5.6.3.11 unsigned int reserved[16]**

Reserved for future use.

**5.6.3.12 char sensorInfo[sk\_maxStringLength]**

String detailing the sensor information.

**5.6.3.13 char sensorResolution[sk\_maxStringLength]**

String providing the sensor resolution.

**5.6.3.14 unsigned int serialNumber**

Device serial number.

**5.6.3.15 char vendorName[sk\_maxStringLength]**

Device vendor name.

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

- [FlyCapture2Defs.h](#)

## 5.7 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](#).*

### 5.7.1 Detailed Description

The [CameraSelectionDlg](#) object represents a GTKmm dialog that provides a graphical interface that lists the number of cameras available to the library.

### 5.7.2 Constructor & Destructor Documentation

#### 5.7.2.1 [CameraSelectionDlg](#) ()

Default constructor.

#### 5.7.2.2 [~CameraSelectionDlg](#) ()

Default destructor.

### 5.7.3 Member Function Documentation

#### 5.7.3.1 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](#)

## 5.8 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.*



## 5.8.1 Detailed Description

[Camera](#) configuration ROM.

## 5.8.2 Constructor & Destructor Documentation

### 5.8.2.1 ConfigROM () [inline]

## 5.8.3 Member Data Documentation

### 5.8.3.1 unsigned int chipIdHi

Chip ID (high part).

### 5.8.3.2 unsigned int chipIdLo

Chip ID (low part).

### 5.8.3.3 unsigned int nodeVendorId

Vendor ID of a node.

### 5.8.3.4 char pszKeyword[sk\_maxStringLength]

Keyword.

### 5.8.3.5 unsigned int reserved[16]

Reserved for future use.

### 5.8.3.6 unsigned int unitSpecId

Unit Spec ID, usually 0xa02d.

### 5.8.3.7 unsigned int unitSubSWVer

Unit sub software version.

### 5.8.3.8 unsigned int unitSWVer

Unit software version.

### 5.8.3.9 unsigned int vendorUniqueInfo\_0

Vendor unique info 0.

**5.8.3.10 unsigned int vendorUniqueInfo\_1**

Vendor unique info 1.

**5.8.3.11 unsigned int vendorUniqueInfo\_2**

Vendor unique info 2.

**5.8.3.12 unsigned int vendorUniqueInfo\_3**

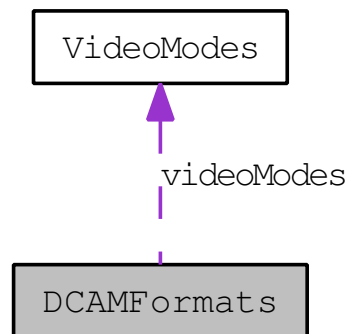
Vendor unique info 3.

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

- [FlyCapture2Defs.h](#)

## 5.9 DCAMFormats Struct Reference

Collaboration diagram for DCAMFormats:



### Public Attributes

- [VideoModes videoModes](#) [FlyCapture2::NUM\_VIDEOMODES]
- unsigned long [numFormats](#)

### 5.9.1 Member Data Documentation

#### 5.9.1.1 unsigned long numFormats

#### 5.9.1.2 VideoModes videoModes[FlyCapture2::NUM\_VIDEOMODES]

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

- [PGRDirectShow.h](#)

## 5.10 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](#)

### 5.10.1 Detailed Description

Properties of the possible embedded image information.

## 5.10.2 Member Data Documentation

- 5.10.2.1 EmbeddedImageInfoProperty brightness
- 5.10.2.2 EmbeddedImageInfoProperty exposure
- 5.10.2.3 EmbeddedImageInfoProperty frameCounter
- 5.10.2.4 EmbeddedImageInfoProperty gain
- 5.10.2.5 EmbeddedImageInfoProperty GPIOPinState
- 5.10.2.6 EmbeddedImageInfoProperty ROIPosition
- 5.10.2.7 EmbeddedImageInfoProperty shutter
- 5.10.2.8 EmbeddedImageInfoProperty strobePattern
- 5.10.2.9 EmbeddedImageInfoProperty timestamp
- 5.10.2.10 EmbeddedImageInfoProperty whiteBalance

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

- [FlyCapture2Defs.h](#)

## 5.11 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.*

### 5.11.1 Detailed Description

Properties of a single embedded image info property.

### 5.11.2 Constructor & Destructor Documentation

#### 5.11.2.1 EmbeddedImageInfoProperty () [inline]

### 5.11.3 Member Data Documentation

#### 5.11.3.1 bool available

Whether this property is available.

#### 5.11.3.2 bool onOff

Whether this property is on or off.

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

- [FlyCapture2Defs.h](#)

## 5.12 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](#)

### 5.12.1 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.

### 5.12.2 Constructor & Destructor Documentation

#### 5.12.2.1 `Error ()`

Default constructor.

#### 5.12.2.2 `Error (const Error & error)`

Copy constructor.

#### 5.12.2.3 `virtual ~Error () [virtual]`

Default destructor.

### 5.12.3 Member Function Documentation

#### 5.12.3.1 `virtual const char* CollectSupportInformation () const [virtual]`

Retrieve the support information.

It is not implemented in this release.

#### Returns:

A string containing support information.

#### 5.12.3.2 `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.



**5.12.3.3 virtual Error GetCause () const** [virtual]

Get the error which caused this error.

**Returns:**

An error object representing the cause of this error.

**5.12.3.4 virtual const char\* GetDescription () const** [virtual]

Retrieve the top level description of the error that occurred.

**Returns:**

A string with the error description.

**5.12.3.5 virtual const char\* GetFilename () const** [virtual]

Retrieve the source filename where the error originated.

**Returns:**

A string with the file name.

**5.12.3.6 virtual unsigned int GetLine () const** [virtual]

Retrieve the line number where the error originated.

**Returns:**

The line number.

**5.12.3.7 virtual ErrorType GetType () const** [virtual]

Retrieve the ErrorType of the error.

**Returns:**

The ErrorType of the error.

**5.12.3.8 virtual bool operator!= (const ErrorType & *errorType*)** [virtual]

Inequality operator.

This overloaded operator compares the ErrorType of the [Error](#) against the specified ErrorType.

**5.12.3.9 virtual bool operator!= (const Error & *error*)** [virtual]

Inequality operator.

**5.12.3.10 virtual Error& operator= (const Error & *error*)** [virtual]

Assignment operator.

**5.12.3.11 virtual bool operator== (const ErrorType & *errorType*)** [virtual]

Equality operator.

This overloaded operator compares the ErrorType of the [Error](#) against the specified ErrorType.

**5.12.3.12 virtual bool operator== (const Error & *error*)** [virtual]

Equality operator.

**5.12.3.13 virtual void PrintErrorTrace () const** [virtual]

Print a formatted log trace to stderr.

**5.12.4 Friends And Related Function Documentation****5.12.4.1 friend class InternalError** [friend]

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

- [Error.h](#)

## 5.13 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](#)  
*This is the number of notifications per image that will be triggered.*
- int [grabTimeout](#)  
*Time in milliseconds that [RetrieveBuffer\(\)](#) 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.*

### 5.13.1 Detailed Description

Configuration for a camera.

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

### 5.13.2 Constructor & Destructor Documentation

#### 5.13.2.1 [FC2Config \(\)](#) [inline]

### 5.13.3 Member Data Documentation

#### 5.13.3.1 [BusSpeed](#) [asyncBusSpeed](#)

Asynchronous bus speed.

### 5.13.3.2 BandwidthAllocation bandwidthAllocation

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

### 5.13.3.3 GrabMode grabMode

Grab mode for the camera.

The default is DROP\_FRAMES.

### 5.13.3.4 int grabTimeout

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

### 5.13.3.5 BusSpeed isochBusSpeed

Isochronous bus speed.

### 5.13.3.6 unsigned int numBuffers

Number of buffers used by the [FlyCapture2](#) library to grab images.

### 5.13.3.7 unsigned int numImageNotifications

This is the number of notifications per image that will be triggered.

The default case is 1 notification at the end of a image. Setting this parameter to 2 will result in notifications after the first packet and at the end of image. Setting this parameter to anything more then 2 will divide the notifications equally throughout the buffer. The maximum number of notifications possible is bufferSize/packetSize since notifications need to land on packet boundaries.

### 5.13.3.8 unsigned int reserved[16]

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

## 5.14 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.*

### 5.14.1 Detailed Description

The current version of the library.

### 5.14.2 Member Data Documentation

#### 5.14.2.1 unsigned int build

Build version number.

#### 5.14.2.2 unsigned int major

Major version number.

#### 5.14.2.3 unsigned int minor

Minor version number.

#### 5.14.2.4 unsigned int type

Type version number.

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

- [FlyCapture2Defs.h](#)

## 5.15 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.*

### 5.15.1 Detailed Description

Format 7 image settings.

### 5.15.2 Constructor & Destructor Documentation

#### 5.15.2.1 [Format7ImageSettings \(\)](#) `[inline]`

### 5.15.3 Member Data Documentation

#### 5.15.3.1 unsigned int height

Height of image.

**5.15.3.2 Mode mode**

Format 7 mode.

**5.15.3.3 unsigned int offsetX**

Horizontal image offset.

**5.15.3.4 unsigned int offsetY**

Vertical image offset.

**5.15.3.5 PixelFormat pixelFormat**

Pixel format of image.

**5.15.3.6 unsigned int reserved[8]**

Reserved for future use.

**5.15.3.7 unsigned int width**

Width of image.

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

- [FlyCapture2Defs.h](#)

## 5.16 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.*



### 5.16.1 Detailed Description

Format 7 information for a single mode.

### 5.16.2 Constructor & Destructor Documentation

#### 5.16.2.1 `Format7Info()` `[inline]`

### 5.16.3 Member Data Documentation

#### 5.16.3.1 `unsigned int imageHStepSize`

Horizontal step size for the image.

#### 5.16.3.2 `unsigned int imageVStepSize`

Vertical step size for the image.

#### 5.16.3.3 `unsigned int maxHeight`

Maximum image height.

#### 5.16.3.4 `unsigned int maxPacketSize`

Maximum packet size in bytes for current mode.

#### 5.16.3.5 `unsigned int maxWidth`

Maximum image width.

#### 5.16.3.6 `unsigned int minPacketSize`

Minimum packet size in bytes for current mode.

#### 5.16.3.7 `Mode mode`

Format 7 mode.

#### 5.16.3.8 `unsigned int offsetHStepSize`

Horizontal step size for the offset.

#### 5.16.3.9 `unsigned int offsetVStepSize`

Vertical step size for the offset.

**5.16.3.10 unsigned int packetSize**

Current packet size in bytes.

**5.16.3.11 float percentage**

Current packet size as a percentage of maximum packet size.

**5.16.3.12 unsigned int pixelFormatBitField**

Supported pixel formats in a bit field.

**5.16.3.13 unsigned int reserved[16]**

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

## 5.17 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.*

### 5.17.1 Detailed Description

Format 7 packet information.

### 5.17.2 Constructor & Destructor Documentation

#### 5.17.2.1 [Format7PacketInfo \(\)](#) `[inline]`

### 5.17.3 Member Data Documentation

#### 5.17.3.1 unsigned int [maxBytesPerPacket](#)

Maximum bytes per packet.

#### 5.17.3.2 unsigned int [recommendedBytesPerPacket](#)

Recommended bytes per packet.

#### 5.17.3.3 unsigned int [reserved](#)[8]

Reserved for future use.

#### 5.17.3.4 unsigned int unitBytesPerPacket

Minimum bytes per packet.

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

- [FlyCapture2Defs.h](#)

## 5.18 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)  
*Converts the current image buffer to the specified output format and stores the result in the specified image.*
- virtual [Error Convert](#) ([Image](#) \*pDestImage)  
*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](#)

### 5.18.1 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.

## 5.18.2 Constructor & Destructor Documentation

### 5.18.2.1 Image ()

Default constructor.

### 5.18.2.2 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.

### 5.18.2.3 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.

### 5.18.2.4 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.



### 5.18.2.5 Image (const Image & *image*)

Copy constructor.

Both images will point to the same image buffer internally.

### 5.18.2.6 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 queued internally.

## 5.18.3 Member Function Documentation

### 5.18.3.1 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:

Metadata associated with the image.

### 5.18.3.2 virtual Error Convert (Image \* *pDestImage*) [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.

### 5.18.3.3 virtual Error Convert (PixelFormat *format*, Image \* *pDestImage*) [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.

**5.18.3.4 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.

**5.18.3.5 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.

**5.18.3.6 virtual BayerTileFormat GetBayerTileFormat () const [virtual]**

Get the Bayer tile format of the image.

**Returns:**

The Bayer tile format.

**5.18.3.7 virtual unsigned int GetBitsPerPixel () const [virtual]**

Get the bits per pixel of the image.

**Returns:**

The bits per pixel.

**5.18.3.8 virtual ColorProcessingAlgorithm GetColorProcessing () const** [virtual]

Get the current color processing algorithm.

See also:

[SetColorProcessing\(\)](#)

Returns:

The current color processing algorithm.

**5.18.3.9 virtual unsigned int GetCols () const** [virtual]

Get the number of columns in the image.

Returns:

The number of columns.

**5.18.3.10 virtual unsigned char\* const GetData () const** [virtual]**5.18.3.11 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.

**5.18.3.12 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.

**5.18.3.13 static ColorProcessingAlgorithm GetDefaultColorProcessing ()** [static]

Get the default color processing algorithm.

See also:

[SetDefaultColorProcessing\(\)](#)

Returns:

The default color processing algorithm.

**5.18.3.14 static PixelFormat GetDefaultOutputFormat () [static]**

Get the default output pixel format.

See also:

[SetDefaultOutputFormat\(\)](#)

**Returns:**

The default pixel format.

**5.18.3.15 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.

**5.18.3.16 virtual ImageMetadata GetMetadata () const [virtual]**

Get the metadata associated with the image.

This includes embedded image information.

**Returns:**

Metadata associated with the image.

**5.18.3.17 virtual PixelFormat GetPixelFormat () const [virtual]**

Get the current pixel format.

**Returns:**

The current pixel format.

**5.18.3.18 virtual unsigned int GetRows () const [virtual]**

Get the number of rows in the image.

**Returns:**

The number of rows.

**5.18.3.19 virtual unsigned int GetStride () const** [virtual]

Get the stride in the image.

**Returns:**

The stride (The number of bytes between rows of the image).

**5.18.3.20 virtual TimeStamp GetTimeStamp () const** [virtual]

Get the timestamp data associated with the image.

**Returns:**

Timestamp data associated with the image.

**5.18.3.21 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.

**5.18.3.22 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.

**5.18.3.23 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.

**5.18.3.24 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.

**5.18.3.25 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.

**5.18.3.26 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.

**5.18.3.27 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.

**5.18.3.28 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.

**5.18.3.29 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.

**5.18.3.30 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.

**5.18.3.31 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.

**5.18.3.32 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.

**5.18.3.33 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.

**5.18.3.34 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.



### 5.18.3.35 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.

### 5.18.3.36 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.

## 5.18.4 Friends And Related Function Documentation

### 5.18.4.1 friend class Iso [friend]

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

- [Image.h](#)

## 5.19 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.*

#### 5.19.1 Detailed Description

Metadata related to an image.

## 5.19.2 Constructor & Destructor Documentation

### 5.19.2.1 ImageMetadata () `[inline]`

## 5.19.3 Member Data Documentation

### 5.19.3.1 unsigned int embeddedBrightness

Embedded brightness.

### 5.19.3.2 unsigned int embeddedExposure

Embedded exposure.

### 5.19.3.3 unsigned int embeddedFrameCounter

Embedded frame counter.

### 5.19.3.4 unsigned int embeddedGain

Embedded gain.

### 5.19.3.5 unsigned int embeddedGPIOPinState

Embedded GPIO pin state.

### 5.19.3.6 unsigned int embeddedROIPosition

Embedded ROI position.

### 5.19.3.7 unsigned int embeddedShutter

Embedded shutter.

### 5.19.3.8 unsigned int embeddedStrobePattern

Embedded strobe pattern.

### 5.19.3.9 unsigned int embeddedTimeStamp

Embedded timestamp.

### 5.19.3.10 unsigned int embeddedWhiteBalance

Embedded white balance.

**5.19.3.11 unsigned int reserved[31]**

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

## 5.20 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](#)

### 5.20.1 Detailed Description

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

### 5.20.2 Member Enumeration Documentation

#### 5.20.2.1 enum StatisticsChannel

Channels that allow statistics to be calculated.

**Enumerator:**

***GREY***  
***RED***  
***GREEN***  
***BLUE***  
***HUE***  
***SATURATION***  
***LIGHTNESS***  
***NUM\_STATISTICS\_CHANNELS***

### 5.20.3 Constructor & Destructor Documentation

#### 5.20.3.1 ImageStatistics ()

Default constructor.

#### 5.20.3.2 virtual ~ImageStatistics () [virtual]

Default destructor.

#### 5.20.3.3 ImageStatistics (const ImageStatistics & *other*)

Copy constructor.

### 5.20.4 Member Function Documentation

#### 5.20.4.1 Error DisableAll ()

Disable all channels.

**Returns:**

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

#### 5.20.4.2 Error EnableAll ()

Enable all channels.

**Returns:**

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

#### 5.20.4.3 Error EnableGreyOnly ()

Enable only the grey channel.

**Returns:**

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

#### 5.20.4.4 Error EnableHSLOnly ()

Enable only the HSL channels.

**Returns:**

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

#### 5.20.4.5 Error EnableRGBOnly ()

Enable only the RGB channels.

**Returns:**

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

**5.20.4.6 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.

**5.20.4.7 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.

**5.20.4.8 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.

**5.20.4.9 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.



#### 5.20.4.10 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.

#### 5.20.4.11 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.

#### 5.20.4.12 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.

#### 5.20.4.13 ImageStatistics& operator= (const ImageStatistics & *other*)

Assignment operator.

**Parameters:**

*other* The [ImageStatistics](#) object to copy from.

#### 5.20.4.14 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.

### 5.20.5 Friends And Related Function Documentation

#### 5.20.5.1 friend class ImageStatsCalculator [friend]

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

- [ImageStatistics.h](#)

## 5.21 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.*

### 5.21.1 Detailed Description

Options for saving JPEG image.

### 5.21.2 Constructor & Destructor Documentation

#### 5.21.2.1 JPEGOption () [inline]

### 5.21.3 Member Data Documentation

#### 5.21.3.1 bool progressive

Whether to save as a progressive JPEG file.

#### 5.21.3.2 unsigned int quality

JPEG image quality in range (0-100).

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

#### 5.21.3.3 unsigned int reserved[16]

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

## 5.22 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.*

### 5.22.1 Detailed Description

Options for saving JPEG2000 image.

### 5.22.2 Constructor & Destructor Documentation

#### 5.22.2.1 [JPG2Option](#) () `[inline]`

### 5.22.3 Member Data Documentation

#### 5.22.3.1 unsigned int [quality](#)

JPEG saving quality in range (1-512).

#### 5.22.3.2 unsigned int [reserved](#)[16]

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

## 5.23 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.*

### 5.23.1 Detailed Description

Information about the camera's look up table.

### 5.23.2 Constructor & Destructor Documentation

#### 5.23.2.1 [LUTData \(\)](#) [inline]

### 5.23.3 Member Data Documentation

#### 5.23.3.1 bool [enabled](#)

Flag indicating if LUT is enabled.

#### 5.23.3.2 unsigned int [inputBitDepth](#)

The input bit depth of the LUT.

**5.23.3.3 unsigned int numBanks**

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

**5.23.3.4 unsigned int numChannels**

The number of LUT channels per bank available.

**5.23.3.5 unsigned int numEntries**

The number of entries in the LUT.

**5.23.3.6 unsigned int outputBitDepth**

The output bit depth of the LUT.

**5.23.3.7 unsigned int reserved[8]**

Reserved for future use.

**5.23.3.8 bool supported**

Flag indicating if LUT is supported.

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

- [FlyCapture2Defs.h](#)

## 5.24 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.*

### 5.24.1 Detailed Description

Options for saving PGM images.

### 5.24.2 Constructor & Destructor Documentation

#### 5.24.2.1 PGMOption () [inline]

### 5.24.3 Member Data Documentation

#### 5.24.3.1 bool binaryFile

Whether to save the PPM as a binary file.

#### 5.24.3.2 unsigned int reserved[16]

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

## 5.25 PGRGuid Class Reference

A GUID to the camera.

### Public Member Functions

- **PGRGuid** ()  
*Constructor.*
- **PGRGuid** (const **PGRGuid** &other)  
*Copy constructor.*
- **PGRGuid** & **operator=** (const **PGRGuid** &other)  
*Assignment operator.*
- bool **operator==** (const **PGRGuid** &guid)  
*Equality operator.*
- bool **operator!=** (const **PGRGuid** &guid)  
*Inequality operator.*

### Public Attributes

- unsigned int **value** [4]

#### 5.25.1 Detailed Description

A GUID to the camera.

It is used to uniquely identify a camera.

#### 5.25.2 Constructor & Destructor Documentation

##### 5.25.2.1 **PGRGuid** () [inline]

Constructor.

##### 5.25.2.2 **PGRGuid** (const **PGRGuid** & *other*) [inline]

Copy constructor.

#### 5.25.3 Member Function Documentation

##### 5.25.3.1 bool **operator!=** (const **PGRGuid** & *guid*) [inline]

Inequality operator.

##### 5.25.3.2 **PGRGuid**& **operator=** (const **PGRGuid** & *other*) [inline]

Assignment operator.



### 5.25.3.3 `bool operator==(const PGRGuid & guid)` `[inline]`

Equality operator.

## 5.25.4 Member Data Documentation

### 5.25.4.1 `unsigned int value[4]`

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

- [FlyCapture2Defs.h](#)

## 5.26 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.*

### 5.26.1 Detailed Description

Options for saving PNG images.

### 5.26.2 Constructor & Destructor Documentation

#### 5.26.2.1 PNGOption () [inline]

### 5.26.3 Member Data Documentation

#### 5.26.3.1 unsigned int compressionLevel

Compression level (0-9).

0 is no compression, 9 is best compression.

#### 5.26.3.2 bool interlaced

Whether to save the PNG as interlaced.

#### 5.26.3.3 unsigned int reserved[16]

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

## 5.27 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.*

### 5.27.1 Detailed Description

Options for saving PPM images.

### 5.27.2 Constructor & Destructor Documentation

#### 5.27.2.1 PPMOption () [inline]

### 5.27.3 Member Data Documentation

#### 5.27.3.1 bool binaryFile

Whether to save the PPM as a binary file.

#### 5.27.3.2 unsigned int reserved[16]

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

## 5.28 Property Struct Reference

A specific camera property.

### Public Member Functions

- [Property](#) ()

### Public Attributes

- [PropertyType](#) type  
*Property info type (e.g.*
- 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.*

### 5.28.1 Detailed Description

A specific camera property.

### 5.28.2 Constructor & Destructor Documentation

#### 5.28.2.1 [Property](#) () [inline]

### 5.28.3 Member Data Documentation

#### 5.28.3.1 bool [absControl](#)

Flag controlling absolute mode.

**5.28.3.2 float absValue**

Floating point value.

**5.28.3.3 bool autoManualMode**

Flag controlling auto.

**5.28.3.4 bool onePush**

Flag controlling one push.

**5.28.3.5 bool onOff**

Flag controlling on/off.

**5.28.3.6 bool present**

Flag indicating if the property is present.

**5.28.3.7 unsigned int reserved[8]**

Reserved for future use.

**5.28.3.8 PropertyType type**

[Property](#) info type (e.g. BRIGHTNESS).

**5.28.3.9 unsigned int valueA**

Value A (integer).

**5.28.3.10 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](#)

## 5.29 PropertyInfo Struct Reference

Information about a specific camera property.

### Public Member Functions

- [PropertyInfo \(\)](#)

### 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.*

### 5.29.1 Detailed Description

Information about a specific camera property.

This structure is also used as the TriggerDelayInfo structure.

### 5.29.2 Constructor & Destructor Documentation

#### 5.29.2.1 PropertyInfo () [inline]

### 5.29.3 Member Data Documentation

#### 5.29.3.1 float absMax

Maximum value (as a floating point value).

#### 5.29.3.2 float absMin

Minimum value (as a floating point value).

#### 5.29.3.3 bool absValSupported

Flag indicating if absolute mode is supported.

#### 5.29.3.4 bool autoSupported

Flag indicating if auto is supported.

#### 5.29.3.5 bool manualSupported

Flag indicating if manual is supported.

#### 5.29.3.6 unsigned int max

Maximum value (as an integer).

#### 5.29.3.7 unsigned int min

Minimum value (as an integer).

#### 5.29.3.8 bool onePushSupported

Flag indicating if one push is supported.

#### 5.29.3.9 bool onOffSupported

Flag indicating if on/off is supported.

#### 5.29.3.10 bool present

Flag indicating if the property is present.

**5.29.3.11 char pUnitAbbr[sk\_maxStringLength]**

Abbreviated textual description of units.

**5.29.3.12 char pUnits[sk\_maxStringLength]**

Textual description of units.

**5.29.3.13 bool readOutSupported**

Flag indicating if property value can be read out.

**5.29.3.14 unsigned int reserved[8]**

Reserved for future use.

**5.29.3.15 PropertyType type**

[Property](#) info type.

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

- [FlyCapture2Defs.h](#)



## 5.30 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.*

### 5.30.1 Detailed Description

A camera strobe.

### 5.30.2 Constructor & Destructor Documentation

#### 5.30.2.1 [StrobeControl \(\)](#) `[inline]`

### 5.30.3 Member Data Documentation

#### 5.30.3.1 float [delay](#)

Signal delay (in ms).

#### 5.30.3.2 float [duration](#)

Signal duration (in ms).

#### 5.30.3.3 bool [onOff](#)

Flag controlling on/off.

**5.30.3.4 unsigned int polarity**

Signal polarity.

**5.30.3.5 unsigned int reserved[8]**

Reserved for future use.

**5.30.3.6 unsigned int source**

Source value.

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

- [FlyCapture2Defs.h](#)

## 5.31 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.*

### 5.31.1 Detailed Description

A camera strobe property.

### 5.31.2 Constructor & Destructor Documentation

#### 5.31.2.1 [StrobeInfo](#) () [inline]

### 5.31.3 Member Data Documentation

#### 5.31.3.1 float [maxValue](#)

Maximum value.

#### 5.31.3.2 float [minValue](#)

Minimum value.

**5.31.3.3 bool onOffSupported**

Flag indicating if on/off is supported.

**5.31.3.4 bool polaritySupported**

Flag indicating if polarity is supported.

**5.31.3.5 bool present**

Presence of strobe.

**5.31.3.6 bool readOutSupported**

Flag indicating if strobe value can be read out.

**5.31.3.7 unsigned int reserved[8]**

Reserved for future use.

**5.31.3.8 unsigned int source**

Source value.

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

- [FlyCapture2Defs.h](#)

## 5.32 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.*

### 5.32.1 Detailed Description

Description of the system.

### 5.32.2 Member Data Documentation

#### 5.32.2.1 ByteOrder `byteOrder`

Byte order of the system.

**5.32.2.2 char cpuDescription[sk\_maxStringLength]**

Detailed description of the CPU.

**5.32.2.3 char driverList[sk\_maxStringLength]**

List of drivers used.

**5.32.2.4 char gpuDescription[sk\_maxStringLength]**

Detailed description of the GPU.

**5.32.2.5 char libraryList[sk\_maxStringLength]**

List of libraries used.

**5.32.2.6 size\_t numCpuCores**

Number of cores on all CPUs on the system.

**5.32.2.7 char osDescription[sk\_maxStringLength]**

Detailed description of the operating system.

**5.32.2.8 OSType osType**

Operating system type as described by OSType.

**5.32.2.9 unsigned int reserved[16]**

Reserved for future use.

**5.32.2.10 size\_t screenHeight**

Screen resolution height in pixels.

**5.32.2.11 size\_t screenWidth**

Screen resolution width in pixels.

**5.32.2.12 size\_t sysMemSize**

Amount of memory available on the system.

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

- [Utilities.h](#)

## 5.33 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.
```

#### 5.33.1 Detailed Description

Options for saving TIFF images.

#### 5.33.2 Member Enumeration Documentation

##### 5.33.2.1 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.
```

### 5.33.3 Constructor & Destructor Documentation

#### 5.33.3.1 TIFFOption () [inline]

### 5.33.4 Member Data Documentation

#### 5.33.4.1 CompressionMethod compression

Compression method to use for encoding TIFF images.

#### 5.33.4.2 unsigned int reserved[16]

Reserved for future use.

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

\* [FlyCapture2Defs.h](#)



## 5.34 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.*

### 5.34.1 Detailed Description

Timestamp information.

### 5.34.2 Constructor & Destructor Documentation

#### 5.34.2.1 [TimeStamp](#) () [inline]

### 5.34.3 Member Data Documentation

#### 5.34.3.1 unsigned int [cycleCount](#)

1394 cycle time count.

#### 5.34.3.2 unsigned int [cycleOffset](#)

1394 cycle time offset.

#### 5.34.3.3 unsigned int [cycleSeconds](#)

1394 cycle time seconds.

#### 5.34.3.4 unsigned int [microSeconds](#)

Microseconds.

**5.34.3.5 unsigned int reserved[8]**

Reserved for future use.

**5.34.3.6 long long seconds**

Seconds.

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

- \* [FlyCapture2Defs.h](#)

## 5.35 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.*

### 5.35.1 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.

### 5.35.2 Member Enumeration Documentation

#### 5.35.2.1 enum NodeType

Type of node.

**Enumerator:**

**COMPUTER**  
**BUS**  
**CAMERA**  
**NODE**

#### 5.35.2.2 enum PortType

Possible states of a port on a node.

**Enumerator:**

**NOT\_CONNECTED**  
**CONNECTED\_TO\_PARENT**  
**CONNECTED\_TO\_CHILD**

### 5.35.3 Constructor & Destructor Documentation

#### 5.35.3.1 TopologyNode ()

Default constructor.

#### 5.35.3.2 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.

#### 5.35.3.3 virtual ~TopologyNode () [virtual]

Default destructor.

#### 5.35.3.4 TopologyNode (const TopologyNode & other)

Copy constructor.

### 5.35.4 Member Function Documentation

#### 5.35.4.1 virtual void AddChild (TopologyNode *childNode*) [virtual]

Add the specified [TopologyNode](#) as a child of the node.

**Parameters:**

*childNode* The [TopologyNode](#) to add.

#### 5.35.4.2 virtual void AddPort (PortType *childPort*) [virtual]

Add the specified PortType as a port of the node.

**Parameters:**

*childPort* The port to add.

#### 5.35.4.3 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.

#### 5.35.4.4 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.

#### 5.35.4.5 virtual int GetDeviceId () [virtual]

Get the device ID associated with the node.

**Returns:**

Device ID of the node.

#### 5.35.4.6 virtual PGRGuid GetGuid () [virtual]

Get the [PGRGuid](#) associated with the node.

**Returns:**

[PGRGuid](#) of the node.

#### 5.35.4.7 virtual InterfaceType GetInterfaceType () [virtual]

Get the interface type associated with the node.

**Returns:**

Interface type of the node.

**5.35.4.8 virtual NodeType GetNodeType () [virtual]**

Get the node type associated with the node.

**Returns:**

Node type of the node.

**5.35.4.9 virtual unsigned int GetNumChildren () [virtual]**

Get the number of child nodes.

**Returns:**

Number of child nodes.

**5.35.4.10 virtual unsigned int GetNumPorts () [virtual]**

Get the number of ports.

**Returns:**

Number of ports.

**5.35.4.11 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.

**5.35.4.12 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](#)

## 5.36 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.*

### 5.36.1 Detailed Description

A camera trigger.

### 5.36.2 Constructor & Destructor Documentation

#### 5.36.2.1 [TriggerMode](#) () [inline]

### 5.36.3 Member Data Documentation

#### 5.36.3.1 unsigned int mode

Mode value.

#### 5.36.3.2 bool onOff

Flag controlling on/off.

#### 5.36.3.3 unsigned int parameter

Parameter value.

#### 5.36.3.4 unsigned int polarity

Polarity value.

#### 5.36.3.5 unsigned int reserved[8]

Reserved for future use.

#### 5.36.3.6 unsigned int source

Source value.

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

- [FlyCapture2Defs.h](#)



## 5.37 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.*

### 5.37.1 Detailed Description

Information about a camera trigger property.

### 5.37.2 Constructor & Destructor Documentation

#### 5.37.2.1 [TriggerModeInfo \(\)](#) [inline]

### 5.37.3 Member Data Documentation

#### 5.37.3.1 unsigned int [modeMask](#)

Mode mask.

#### 5.37.3.2 bool [onOffSupported](#)

Flag indicating if on/off is supported.

#### 5.37.3.3 bool [polaritySupported](#)

Flag indicating if polarity is supported.

**5.37.3.4 bool present**

Presence of trigger mode.

**5.37.3.5 bool readOutSupported**

Flag indicating if trigger value can be read out.

**5.37.3.6 unsigned int reserved[8]**

Reserved for future use.

**5.37.3.7 bool softwareTriggerSupported**

Flag indicating if software trigger is supported.

**5.37.3.8 unsigned int sourceMask**

Source mask.

**5.37.3.9 bool valueReadable**

Flag indicating if the value is readable.

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

- [FlyCapture2Defs.h](#)

## 5.38 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.*

### 5.38.1 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.

### 5.38.2 Member Function Documentation

#### 5.38.2.1 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.

#### 5.38.2.2 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.

#### 5.38.2.3 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.

**5.38.2.4 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.

**5.38.2.5 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.

**5.38.2.6 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](#)

## 5.39 VideoModes Struct Reference

### Public Attributes

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

### 5.39.1 Member Data Documentation

#### 5.39.1.1 [FlyCapture2::FrameRate](#) frameRate

#### 5.39.1.2 unsigned long height

#### 5.39.1.3 [FlyCapture2::VideoMode](#) videoMode

#### 5.39.1.4 unsigned long width

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

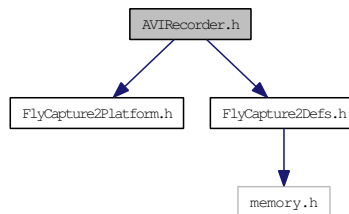
- [PGRDirectShow.h](#)

## Chapter 6

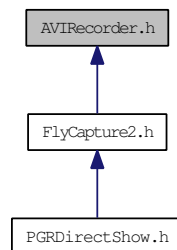
# File Documentation

### 6.1 AVIRecorder.h File Reference

Include dependency graph for AVIRecorder.h:



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



### Classes

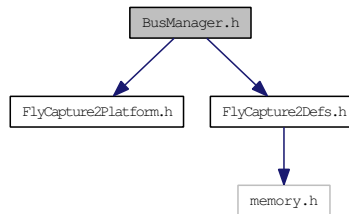
- class [AVIRecorder](#)  
*The [AVIRecorder](#) class provides the functionality for the user to record images to an AVI file.*

### Namespaces

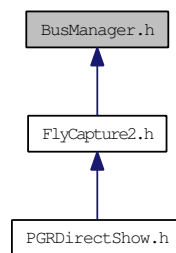
- namespace [FlyCapture2](#)

## 6.2 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

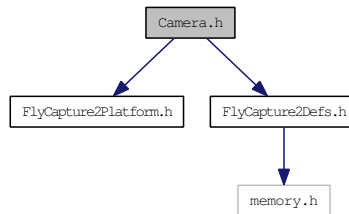
- namespace [FlyCapture2](#)

### Typedefs

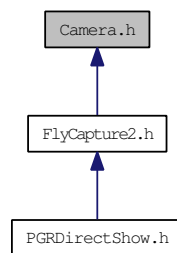
- typedef void(\* [BusEventCallback](#) )(void \*pParameter)  
*Bus event callback function prototype.*
- typedef void \* [CallbackHandle](#)  
*Handle that is returned when registering a callback.*

## 6.3 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.*

### Namespaces

- namespace [FlyCapture2](#)

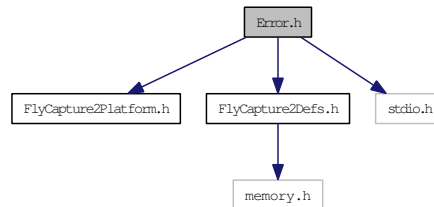
### Typedefs

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

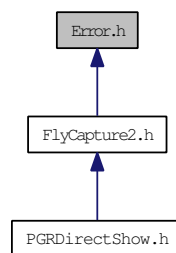


## 6.4 Error.h File Reference

Include dependency graph for Error.h:



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



### Classes

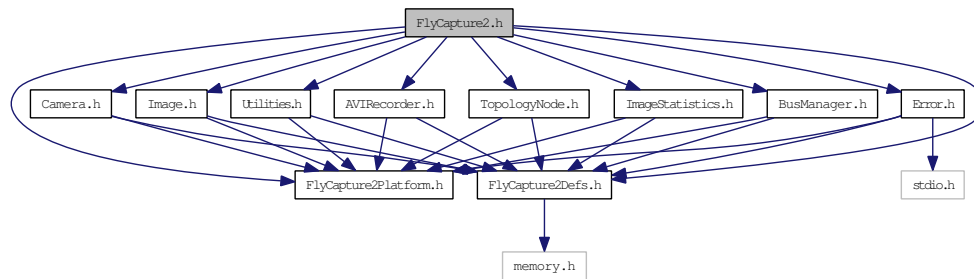
- class [Error](#)  
*The [Error](#) object represents an error that is returned from the library.*

### Namespaces

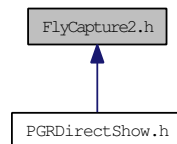
- namespace [FlyCapture2](#)

## 6.5 FlyCapture2.h File Reference

Include dependency graph for FlyCapture2.h:

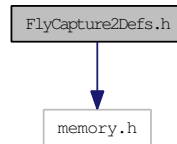


This graph shows which files directly or indirectly include this file:

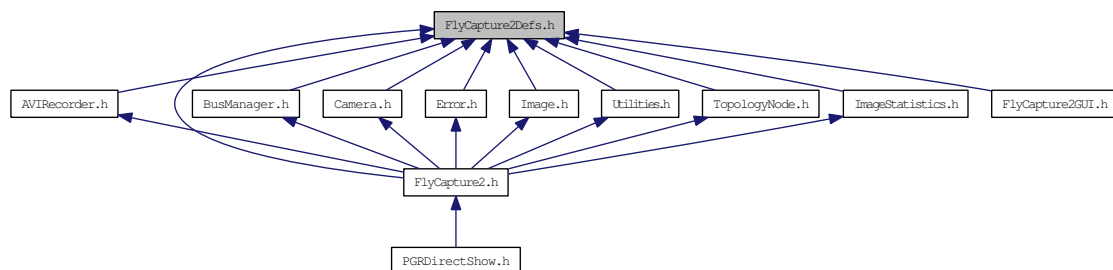


## 6.6 FlyCapture2Defs.h File Reference

Include dependency graph for FlyCapture2Defs.h:



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



### Classes

- class [PGRGuid](#)  
A GUID to the camera.
- struct [FC2Version](#)  
The current version of the library.
- 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 [Format7ImageSettings](#)  
Format 7 image settings.
- struct [Format7Info](#)  
Format 7 information for a single mode.
- struct [Format7PacketInfo](#)  
Format 7 packet information.

- struct [TimeStamp](#)  
*Timestamp information.*
- struct [ConfigROM](#)  
*Camera configuration ROM.*
- struct [CameraInfo](#)  
*Camera information.*
- struct [EmbeddedImageInfoProperty](#)  
*Properties of a single embedded image info property.*
- struct [EmbeddedImageInfo](#)  
*Properties of the possible embedded image information.*
- struct [ImageMetadata](#)  
*Metadata related to an image.*
- struct [LUTData](#)  
*Information about the camera's look up table.*
- struct [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_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_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\_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_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,
    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.*

## Variables

```

· static const unsigned int sk_maxStringLength = 512

```

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

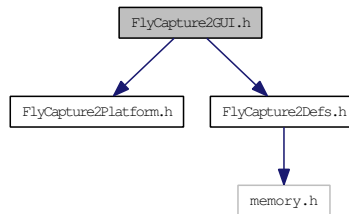
### 6.6.1 Define Documentation

**6.6.1.1 #define FULL\_32BIT\_VALUE 0x7FFFFFFF**

**6.6.1.2 #define NULL 0**

## 6.7 FlyCapture2GUI.h File Reference

Include dependency graph for FlyCapture2GUI.h:



### Classes

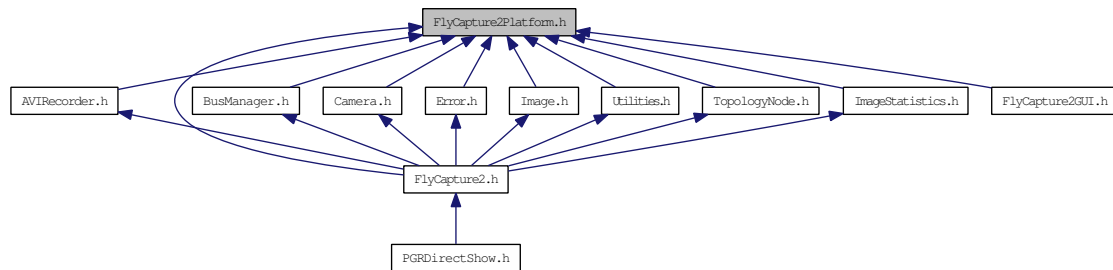
- class [CameraControlDlg](#)  
*The [CameraControlDlg](#) object represents a GTKmm dialog that provides a graphical interface to a specified camera.*
- class [CameraSelectionDlg](#)  
*The [CameraSelectionDlg](#) object represents a GTKmm dialog that provides a graphical interface that lists the number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

### Namespaces

- namespace [FlyCapture2](#)

## 6.8 FlyCapture2Platform.h File Reference

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



### Defines

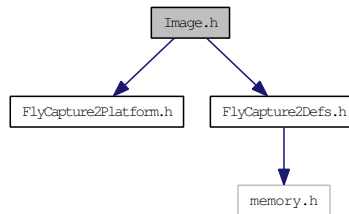
· #define [FLYCAPTURE2\\_API](#)

#### 6.8.1 Define Documentation

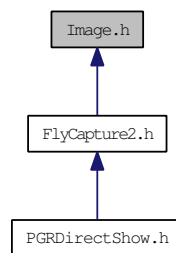
##### 6.8.1.1 #define FLYCAPTURE2\_API

## 6.9 Image.h File Reference

Include dependency graph for Image.h:



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



### Classes

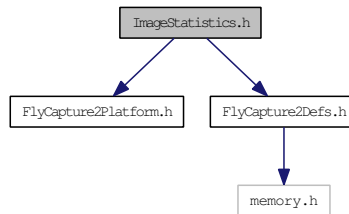
- class [Image](#)  
*The [Image](#) class is used to retrieve images from a camera, convert between multiple pixel formats and save images to disk.*

### Namespaces

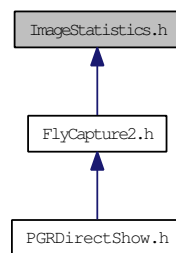
- namespace [FlyCapture2](#)

## 6.10 ImageStatistics.h File Reference

Include dependency graph for ImageStatistics.h:



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



### Classes

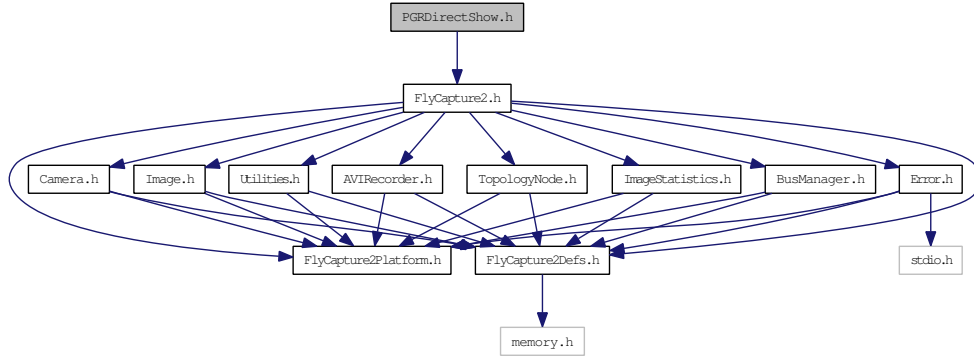
- class [ImageStatistics](#)  
*The [ImageStatistics](#) object represents image statistics for an image.*

### Namespaces

- namespace [FlyCapture2](#)

## 6.11 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
- STDMETHOD() [SetAbsSharpness](#) (float lSharpness, bool bAuto=false)=0
- STDMETHOD() [GetAbsSharpnessRange](#) (float \*pMin, float \*pMax, bool \*pbAutoSupported=NULL, const char \*\*pUnits=NULL)=0

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

signed int uiTop, unsigned int uiWidth, unsigned int uiHeight, unsigned int uiPixelFormat)=0

- STDMETHODCALLTYPE [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}}





### 6.11.1 Function Documentation

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

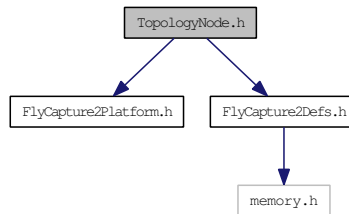
**6.11.1.21** **STDMETHOD()** GetAbsWhiteBalance (float \* *plWhiteBalance*, bool \* *pbAuto* = NULL) [pure virtual]

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

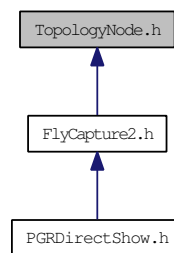
**6.11.1.23** **STDMETHOD()** GetBrightness (long \* *pBrightness*, bool \* *pbAuto* = NULL) [pure virtual]

## 6.12 TopologyNode.h File Reference

Include dependency graph for TopologyNode.h:



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



### Classes

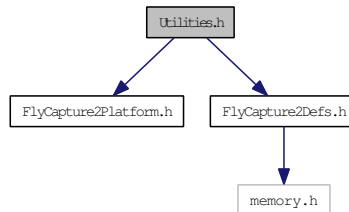
- class [TopologyNode](#)  
*The [TopologyNode](#) class contains topology information that can be used to generate a tree structure of all cameras and devices connected to a computer.*

### Namespaces

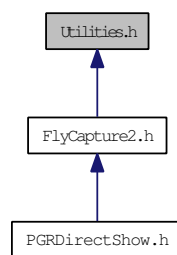
- namespace [FlyCapture2](#)

## 6.13 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, [29](#)
- ~BusManager
  - FlyCapture2::BusManager, [31](#)
- ~Camera
  - FlyCapture2::Camera, [40](#)
- ~CameraControlDlg
  - FlyCapture2::CameraControlDlg, [59](#)
- ~CameraSelectionDlg
  - FlyCapture2::CameraSelectionDlg, [64](#)
- ~Error
  - FlyCapture2::Error, [73](#)
- ~Image
  - FlyCapture2::Image, [90](#)
- ~ImageStatistics
  - FlyCapture2::ImageStatistics, [104](#)
- ~TopologyNode
  - FlyCapture2::TopologyNode, [133](#)
- absControl
  - FlyCapture2::Property, [117](#)
- absMax
  - FlyCapture2::PropertyInfo, [120](#)
- absMin
  - FlyCapture2::PropertyInfo, [120](#)
- absValSupported
  - FlyCapture2::PropertyInfo, [120](#)
- absValue
  - FlyCapture2::Property, [117](#)
- AddChild
  - FlyCapture2::TopologyNode, [134](#)
- AddPort
  - FlyCapture2::TopologyNode, [134](#)
- ADOBE\_DEFLATE
  - FlyCapture2::TIFFOption, [128](#)
- ARRIVAL
  - FlyCapture2, [18](#)
- AssignGuidToNode
  - FlyCapture2::TopologyNode, [134](#)
- asyncBusSpeed
  - FlyCapture2::FC2Config, [76](#)
- AsyncCommandCallback
  - FlyCapture2, [16](#)
- AUTO\_EXPOSURE
  - FlyCapture2, [24](#)
- autoManualMode
  - FlyCapture2::Property, [118](#)
- autoSupported
  - FlyCapture2::PropertyInfo, [120](#)
- available
  - FlyCapture2::EmbeddedImageInfoProperty, [71](#)
- AVIAppend
  - FlyCapture2::AVIRecorder, [29](#)
- AVIClose
  - FlyCapture2::AVIRecorder, [29](#)
- AVIOpen
  - FlyCapture2::AVIRecorder, [30](#)
- AVIOption
  - FlyCapture2::AVIOption, [27](#)
- AVIRecorder
  - FlyCapture2::AVIRecorder, [29](#)
- AVIRecorder.h, [143](#)
- BANDWIDTH\_ALLOCATION\_FORCE\_32BITS
  - FlyCapture2, [17](#)
- BANDWIDTH\_ALLOCATION\_OFF
  - FlyCapture2, [17](#)
- BANDWIDTH\_ALLOCATION\_ON
  - FlyCapture2, [17](#)
- BANDWIDTH\_ALLOCATION\_UNSPECIFIED
  - FlyCapture2, [17](#)
- BANDWIDTH\_ALLOCATION\_UNSUPPORTED
  - FlyCapture2, [17](#)
- BandwidthAllocation
  - FlyCapture2, [17](#)
- bandwidthAllocation
  - FlyCapture2::FC2Config, [76](#)
- BayerTileFormat
  - FlyCapture2, [17](#)
- bayerTileFormat
  - FlyCapture2::CameraInfo, [62](#)
- BGGR
  - FlyCapture2, [18](#)
- binaryFile
  - FlyCapture2::PGMOption, [112](#)
  - FlyCapture2::PPMOption, [116](#)
- BLUE
  - FlyCapture2::ImageStatistics, [103](#)
- BMP
  - FlyCapture2, [22](#)

- BRIGHTNESS
  - FlyCapture2, [24](#)
- brightness
  - FlyCapture2::EmbeddedImageInfo, [70](#)
- BT\_FORCE\_32BITS
  - FlyCapture2, [18](#)
- BUFFER\_FRAMES
  - FlyCapture2, [21](#)
- build
  - FlyCapture2::FC2Version, [78](#)
- BUS
  - FlyCapture2::TopologyNode, [133](#)
- BUS\_RESET
  - FlyCapture2, [18](#)
- BusCallbackType
  - FlyCapture2, [18](#)
- BusEventCallback
  - FlyCapture2, [16](#)
- BusManager
  - FlyCapture2::BusManager, [31](#)
- BusManager.h, [144](#)
- BusSpeed
  - FlyCapture2, [18](#)
- BUSSPEED\_ANY
  - FlyCapture2, [18](#)
- BUSSPEED\_FORCE\_32BITS
  - FlyCapture2, [18](#)
- BUSSPEED\_S100
  - FlyCapture2, [18](#)
- BUSSPEED\_S1600
  - FlyCapture2, [18](#)
- BUSSPEED\_S200
  - FlyCapture2, [18](#)
- BUSSPEED\_S3200
  - FlyCapture2, [18](#)
- BUSSPEED\_S400
  - FlyCapture2, [18](#)
- BUSSPEED\_S480
  - FlyCapture2, [18](#)
- BUSSPEED\_S800
  - FlyCapture2, [18](#)
- BUSSPEED\_S\_FASTEST
  - FlyCapture2, [18](#)
- BUSSPEED\_SPEED\_UNKNOWN
  - FlyCapture2, [18](#)
- BYTE\_ORDER\_BIG\_ENDIAN
  - FlyCapture2, [19](#)
- BYTE\_ORDER\_FORCE\_32BITS
  - FlyCapture2, [19](#)
- BYTE\_ORDER\_LITTLE\_ENDIAN
  - FlyCapture2, [19](#)
- ByteOrder
  - FlyCapture2, [18](#)
- byteOrder
  - FlyCapture2::SystemInfo, [126](#)
- CalculateStatistics
  - FlyCapture2::Image, [90](#)
- CALLBACK\_TYPE\_FORCE\_32BITS
  - FlyCapture2, [18](#)
- CallbackHandle
  - FlyCapture2, [17](#)
- CAMERA
  - FlyCapture2::TopologyNode, [133](#)
- Camera
  - FlyCapture2::Camera, [40](#)
- Camera.h, [145](#)
- CameraControlDlg
  - FlyCapture2::CameraControlDlg, [59](#)
- CameraInfo
  - FlyCapture2::CameraInfo, [62](#)
- CameraSelectionDlg
  - FlyCapture2::CameraSelectionDlg, [64](#)
- CCITTFAX3
  - FlyCapture2::TIFFOption, [128](#)
- CCITTFAX4
  - FlyCapture2::TIFFOption, [128](#)
- chipIdHi
  - FlyCapture2::ConfigROM, [66](#)
- chipIdLo
  - FlyCapture2::ConfigROM, [66](#)
- CollectSupportInformation
  - FlyCapture2::Error, [73](#)
- COLOR\_PROCESSING\_ALGORITHM\_FORCE\_32BITS
  - FlyCapture2, [19](#)
- ColorProcessingAlgorithm
  - FlyCapture2, [19](#)
- compression
  - FlyCapture2::TIFFOption, [129](#)
- compressionLevel
  - FlyCapture2::PNGOption, [115](#)
- CompressionMethod
  - FlyCapture2::TIFFOption, [128](#)
- COMPUTER
  - FlyCapture2::TopologyNode, [133](#)
- ConfigROM
  - FlyCapture2::ConfigROM, [66](#)
- configROM
  - FlyCapture2::CameraInfo, [62](#)
- Connect
  - FlyCapture2::Camera, [40](#)
  - FlyCapture2::CameraControlDlg, [60](#)
- CONNECTED\_TO\_CHILD
  - FlyCapture2::TopologyNode, [133](#)
- CONNECTED\_TO\_PARENT
  - FlyCapture2::TopologyNode, [133](#)
- Convert
  - FlyCapture2::Image, [90](#)

- cpuDescription
  - FlyCapture2::SystemInfo, 126
- cycleCount
  - FlyCapture2::TimeStamp, 130
- cycleOffset
  - FlyCapture2::TimeStamp, 130
- cycleSeconds
  - FlyCapture2::TimeStamp, 130
- DCAMFormats, 68
  - numFormats, 68
  - videoModes, 68
- DeepCopy
  - FlyCapture2::Image, 91
- DEFAULT
  - FlyCapture2, 19
- DEFLATE
  - FlyCapture2::TIFFOption, 128
- delay
  - FlyCapture2::StrobeControl, 122
- DetermineBitsPerPixel
  - FlyCapture2::Image, 91
- DisableAll
  - FlyCapture2::ImageStatistics, 104
- Disconnect
  - FlyCapture2::Camera, 40
  - FlyCapture2::CameraControlDlg, 60
- driverList
  - FlyCapture2::SystemInfo, 127
- driverName
  - FlyCapture2::CameraInfo, 62
- DROP\_FRAMES
  - FlyCapture2, 21
- duration
  - FlyCapture2::StrobeControl, 122
- EDGE\_SENSING
  - FlyCapture2, 19
- embeddedBrightness
  - FlyCapture2::ImageMetadata, 100
- embeddedExposure
  - FlyCapture2::ImageMetadata, 100
- embeddedFrameCounter
  - FlyCapture2::ImageMetadata, 100
- embeddedGain
  - FlyCapture2::ImageMetadata, 100
- embeddedGPIOPinState
  - FlyCapture2::ImageMetadata, 100
- EmbeddedImageInfoProperty
  - FlyCapture2::EmbeddedImageInfoProperty, 71
- embeddedROIPosition
  - FlyCapture2::ImageMetadata, 100
- embeddedShutter
  - FlyCapture2::ImageMetadata, 100
- embeddedStrobePattern
  - FlyCapture2::ImageMetadata, 100
- embeddedTimeStamp
  - FlyCapture2::ImageMetadata, 100
- embeddedWhiteBalance
  - FlyCapture2::ImageMetadata, 100
- EnableAll
  - FlyCapture2::ImageStatistics, 104
- enabled
  - FlyCapture2::LUTData, 110
- EnableGreyOnly
  - FlyCapture2::ImageStatistics, 104
- EnableHSLOnly
  - FlyCapture2::ImageStatistics, 104
- EnableLUT
  - FlyCapture2::Camera, 40
- EnableRGBOnly
  - FlyCapture2::ImageStatistics, 104
- Error
  - FlyCapture2::Error, 73
- Error.h, 146
- ErrorType
  - FlyCapture2, 19
- exposure
  - FlyCapture2::EmbeddedImageInfo, 70
- FC2Config
  - FlyCapture2::FC2Config, 76
- FireSoftwareTrigger
  - FlyCapture2::Camera, 41
- firmwareBuildTime
  - FlyCapture2::CameraInfo, 62
- firmwareVersion
  - FlyCapture2::CameraInfo, 62
- FlyCapture2, 7
  - ARRIVAL, 18
  - AsyncCommandCallback, 16
  - AUTO\_EXPOSURE, 24
  - BANDWIDTH\_ALLOCATION\_FORCE\_32BITS, 17
  - BANDWIDTH\_ALLOCATION\_OFF, 17
  - BANDWIDTH\_ALLOCATION\_ON, 17
  - BANDWIDTH\_ALLOCATION\_UNSPECIFIED, 17
  - BANDWIDTH\_ALLOCATION\_UNSUPPORTED, 17
  - BandwidthAllocation, 17
  - BayerTileFormat, 17
  - BGGR, 18
  - BMP, 22
  - BRIGHTNESS, 24
  - BT\_FORCE\_32BITS, 18
  - BUFFER\_FRAMES, 21
  - BUS\_RESET, 18
  - BusCallbackType, 18
  - BusEventCallback, 16
  - BusSpeed, 18



- BUSSPEED\_ANY, 18
- BUSSPEED\_FORCE\_32BITS, 18
- BUSSPEED\_S100, 18
- BUSSPEED\_S1600, 18
- BUSSPEED\_S200, 18
- BUSSPEED\_S3200, 18
- BUSSPEED\_S400, 18
- BUSSPEED\_S480, 18
- BUSSPEED\_S800, 18
- BUSSPEED\_S\_FASTEST, 18
- BUSSPEED\_SPEED\_UNKNOWN, 18
- BYTE\_ORDER\_BIG\_ENDIAN, 19
- BYTE\_ORDER\_FORCE\_32BITS, 19
- BYTE\_ORDER\_LITTLE\_ENDIAN, 19
- ByteOrder, 18
- CALLBACK\_TYPE\_FORCE\_32BITS, 18
- CallbackHandle, 17
- COLOR\_PROCESSING\_ALGORITHM\_FORCE\_32BITS, 19
- ColorProcessingAlgorithm, 19
- DEFAULT, 19
- DROP\_FRAMES, 21
- EDGE\_SENSING, 19
- ErrorType, 19
- FOCUS, 24
- FRAME\_RATE, 24
- FrameRate, 20
- FRAMERATE\_120, 21
- FRAMERATE\_15, 20
- FRAMERATE\_1\_875, 20
- FRAMERATE\_240, 21
- FRAMERATE\_30, 20
- FRAMERATE\_3\_75, 20
- FRAMERATE\_60, 21
- FRAMERATE\_7\_5, 20
- FRAMERATE\_FORCE\_32BITS, 21
- FRAMERATE\_FORMAT7, 21
- FROM\_FILE\_EXT, 22
- GAIN, 24
- GAMMA, 24
- GBRG, 18
- GRAB\_MODE\_FORCE\_32BITS, 21
- GRAB\_TIMEOUT\_FORCE\_32BITS, 21
- GrabMode, 21
- GrabTimeout, 21
- GRBG, 18
- HQ\_LINEAR, 19
- HUE, 24
- IMAGE\_FILE\_FORMAT\_FORCE\_32BITS, 22
- ImageEventCallback, 17
- ImageFileFormat, 21
- INTERFACE\_GIGE, 22
- INTERFACE\_IEEE1394, 22
- INTERFACE\_TYPE\_FORCE\_32BITS, 22
- INTERFACE\_UNKNOWN, 22
- INTERFACE\_USB2, 22
- InterfaceType, 22
- IRIS, 24
- JPEG, 22
- JPEG2000, 22
- LINUX\_X64, 23
- LINUX\_X86, 23
- MAC, 23
- Mode, 22
- MODE\_0, 22
- MODE\_1, 22
- MODE\_10, 22
- MODE\_11, 22
- MODE\_12, 22
- MODE\_13, 22
- MODE\_14, 23
- MODE\_15, 23
- MODE\_16, 23
- MODE\_17, 23
- MODE\_18, 23
- MODE\_19, 23
- MODE\_2, 22
- MODE\_20, 23
- MODE\_21, 23
- MODE\_22, 23
- MODE\_23, 23
- MODE\_24, 23
- MODE\_25, 23
- MODE\_26, 23
- MODE\_27, 23
- MODE\_28, 23
- MODE\_29, 23
- MODE\_3, 22
- MODE\_30, 23
- MODE\_31, 23
- MODE\_4, 22
- MODE\_5, 22
- MODE\_6, 22
- MODE\_7, 22
- MODE\_8, 22
- MODE\_9, 22
- MODE\_FORCE\_32BITS, 23
- NEAREST\_NEIGHBOR, 19
- NO\_COLOR\_PROCESSING, 19
- NONE, 18
- NUM\_FRAMERATES, 21
- NUM\_MODES, 23
- NUM\_PIXEL\_FORMATS, 24
- NUM\_VIDEOMODES, 25
- OSType, 23
- OSTYPE\_FORCE\_32BITS, 23
- PAN, 24
- PGM, 22
- PGRERROR\_BUFFER\_TOO\_SMALL, 20

- PGRERROR\_BUS\_MASTER\_FAILED, 20
- PGRERROR\_FAILED, 19
- PGRERROR\_FAILED\_BUS\_MASTER\_CONNECTION, 19
- PGRERROR\_FAILED\_GUID, 20
- PGRERROR\_FORCE\_32BITS, 20
- PGRERROR\_IIDC\_FAILED, 20
- PGRERROR\_IMAGE\_CONVERSION\_FAILED, 20
- PGRERROR\_IMAGE\_LIBRARY\_FAILURE, 20
- PGRERROR\_INIT\_FAILED, 19
- PGRERROR\_INVALID\_BUS\_MANAGER, 19
- PGRERROR\_INVALID\_GENERATION, 20
- PGRERROR\_INVALID\_MODE, 20
- PGRERROR\_INVALID\_PACKET\_SIZE, 20
- PGRERROR\_INVALID\_PARAMETER, 19
- PGRERROR\_INVALID\_SETTINGS, 19
- PGRERROR\_ISOCH\_ALREADY\_STARTED, 20
- PGRERROR\_ISOCH\_BANDWIDTH\_EXCEEDED, 20
- PGRERROR\_ISOCH\_FAILED, 20
- PGRERROR\_ISOCH\_NOT\_STARTED, 20
- PGRERROR\_ISOCH\_RETRIEVE\_BUFFER\_FAILED, 20
- PGRERROR\_ISOCH\_START\_FAILED, 20
- PGRERROR\_ISOCH\_STOP\_FAILED, 20
- PGRERROR\_ISOCH\_SYNC\_FAILED, 20
- PGRERROR\_LOW\_LEVEL\_FAILURE, 19
- PGRERROR\_LUT\_FAILED, 20
- PGRERROR\_MEMORY\_ALLOCATION\_FAILED, 19
- PGRERROR\_NOT\_CONNECTED, 19
- PGRERROR\_NOT\_FOUND, 20
- PGRERROR\_NOT\_IMPLEMENTED, 19
- PGRERROR\_NOT\_IN\_FORMAT7, 20
- PGRERROR\_NOT\_INITIALIZED, 19
- PGRERROR\_NOT\_SUPPORTED, 20
- PGRERROR\_OK, 19
- PGRERROR\_PROPERTY\_FAILED, 20
- PGRERROR\_PROPERTY\_NOT\_PRESENT, 20
- PGRERROR\_READ\_REGISTER\_FAILED, 20
- PGRERROR\_REGISTER\_FAILED, 20
- PGRERROR\_STROBE\_FAILED, 20
- PGRERROR\_TIMEOUT, 20
- PGRERROR\_TRIGGER\_FAILED, 20
- PGRERROR\_UNDEFINED, 19
- PGRERROR\_WRITE\_REGISTER\_FAILED, 20
- PIXEL\_FORMAT\_411YUV8, 23
- 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, 23
- 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, 23
- PNG, 22
- PPM, 22
- PROPERTY\_TYPE\_FORCE\_32BITS, 24
- PropertyType, 24
- RAW, 22
- REMOVAL, 18
- RGBB, 18
- RIGOROUS, 19
- SATURATION, 24
- SHARPNESS, 24
- SHUTTER, 24
- sk\_maxStringLength, 25
- TEMPERATURE, 24
- TILE, 22
- TILT, 24
- TIMEOUT\_INFINITE, 21
- TIMEOUT\_UNSPECIFIED, 21
- TRIGGER\_DELAY, 24
- TRIGGER\_MODE, 24
- TriggerDelay, 17
- TriggerDelayInfo, 17
- UNKNOWN\_OS, 23
- UNSPECIFIED\_GRAB\_MODE, 21
- UNSPECIFIED\_PIXEL\_FORMAT, 24
- UNSPECIFIED\_PROPERTY\_TYPE, 24
- VideoMode, 24
- VIDEOMODE\_1024x768RGB, 25
- VIDEOMODE\_1024x768Y16, 25
- VIDEOMODE\_1024x768Y8, 25
- VIDEOMODE\_1024x768YUV422, 25
- VIDEOMODE\_1280x960RGB, 25
- VIDEOMODE\_1280x960Y16, 25
- VIDEOMODE\_1280x960Y8, 25
- VIDEOMODE\_1280x960YUV422, 25
- VIDEOMODE\_1600x1200RGB, 25
- VIDEOMODE\_1600x1200Y16, 25
- VIDEOMODE\_1600x1200Y8, 25
- VIDEOMODE\_1600x1200YUV422, 25
- 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, 25
- VIDEOMODE\_FORMAT7, 25
- WHITE\_BALANCE, 24
- WINDOWS\_X64, 23
- WINDOWS\_X86, 23
- ZOOM, 24
- FlyCapture2.h, 147
- FlyCapture2::AVIOption, 27
  - AVIOption, 27
  - frameRate, 27
  - reserved, 27
- FlyCapture2::AVIRecorder, 29
  - ~AVIRecorder, 29
  - AVIAppend, 29
  - AVIClose, 29
  - AVIOpen, 30
  - AVIRecorder, 29
- FlyCapture2::BusManager, 31
  - ~BusManager, 31
  - BusManager, 31
  - GetCameraFromIndex, 32
  - GetCameraFromSerialNumber, 32
  - GetCameraSerialNumberFromIndex, 32
  - GetNumOfCameras, 33
  - GetTopology, 33
  - RegisterCallback, 33
  - UnregisterCallback, 33
- FlyCapture2::Camera, 35
  - ~Camera, 40
  - Camera, 40
  - Connect, 40
  - Disconnect, 40
  - EnableLUT, 40
  - FireSoftwareTrigger, 41
  - GetActiveLUTBank, 41
  - GetCameraInfo, 41
  - GetConfiguration, 41
  - GetEmbeddedImageInfo, 42
  - GetFormat7Configuration, 42
  - GetFormat7Info, 42
  - GetGPIOPinDirection, 43
  - GetLUTBankInfo, 43
  - GetLUTChannel, 44
  - GetLUTInfo, 44
  - GetMemoryChannel, 44
  - GetMemoryChannelInfo, 45
  - GetProperty, 45
  - GetPropertyInfo, 45
  - GetRegisterString, 46
  - GetStrobe, 46
  - GetStrobeInfo, 46
  - GetTriggerDelay, 47
  - GetTriggerDelayInfo, 47
  - GetTriggerMode, 47
  - GetTriggerModeInfo, 48
  - GetVideoModeAndFrameRate, 48
  - GetVideoModeAndFrameRateInfo, 49
  - IsConnected, 49
  - ReadRegister, 49
  - ReadRegisterBlock, 50
  - RestoreFromMemoryChannel, 50
  - RetrieveBuffer, 50
  - SaveToMemoryChannel, 51
  - SetActiveLUTBank, 51
  - SetConfiguration, 51
  - SetEmbeddedImageInfo, 51
  - SetFormat7Configuration, 52
  - SetGPIOPinDirection, 52
  - SetLUTChannel, 53
  - SetProperty, 53
  - SetStrobe, 54
  - SetTriggerDelay, 54
  - SetTriggerMode, 54
  - SetUserBuffers, 55
  - SetVideoModeAndFrameRate, 55
  - StartCapture, 56
  - StartSyncCapture, 56
  - StopCapture, 57
  - ValidateFormat7Settings, 57
  - WaitForBufferEvent, 57
  - WriteRegister, 58
  - WriteRegisterBlock, 58
- FlyCapture2::CameraControlDlg, 59
  - ~CameraControlDlg, 59
  - CameraControlDlg, 59
  - Connect, 60
  - Disconnect, 60
  - Hide, 60
  - IsVisible, 60
  - Show, 60
- FlyCapture2::CameraInfo, 61
  - bayerTileFormat, 62
  - CameraInfo, 62
  - configROM, 62
  - driverName, 62
  - firmwareBuildTime, 62
  - firmwareVersion, 62
  - iidcVer, 63
  - interfaceType, 63
  - isColorCamera, 63
  - maximumBusSpeed, 63
  - modelName, 63
  - reserved, 63
  - sensorInfo, 63
  - sensorResolution, 63
  - serialNumber, 63
  - vendorName, 63

- FlyCapture2::CameraSelectionDlg, 64
  - ~CameraSelectionDlg, 64
  - CameraSelectionDlg, 64
  - ShowModal, 64
- FlyCapture2::ConfigROM, 65
  - chipIdHi, 66
  - chipIdLo, 66
  - ConfigROM, 66
  - nodeVendorId, 66
  - pszKeyword, 66
  - reserved, 66
  - unitSpecId, 66
  - unitSubSWVer, 66
  - unitSWVer, 66
  - vendorUniqueInfo\_0, 66
  - vendorUniqueInfo\_1, 66
  - vendorUniqueInfo\_2, 67
  - vendorUniqueInfo\_3, 67
- FlyCapture2::EmbeddedImageInfo, 69
  - brightness, 70
  - exposure, 70
  - frameCounter, 70
  - gain, 70
  - GPIOPinState, 70
  - ROIPosition, 70
  - shutter, 70
  - strobePattern, 70
  - timestamp, 70
  - whiteBalance, 70
- FlyCapture2::EmbeddedImageInfoProperty, 71
  - available, 71
  - EmbeddedImageInfoProperty, 71
  - onOff, 71
- FlyCapture2::Error, 72
  - ~Error, 73
  - CollectSupportInformation, 73
  - Error, 73
  - GetBuildDate, 73
  - GetCause, 73
  - GetDescription, 74
  - GetFilename, 74
  - GetLine, 74
  - GetType, 74
  - InternalError, 75
  - operator=, 74
  - operator==, 75
  - PrintErrorTrace, 75
- FlyCapture2::FC2Config, 76
  - asyncBusSpeed, 76
  - bandwidthAllocation, 76
  - FC2Config, 76
  - grabMode, 77
  - grabTimeout, 77
  - isochBusSpeed, 77
  - numBuffers, 77
  - numImageNotifications, 77
  - reserved, 77
- FlyCapture2::FC2Version, 78
  - build, 78
  - major, 78
  - minor, 78
  - type, 78
- FlyCapture2::Format7ImageSettings, 79
  - Format7ImageSettings, 79
  - height, 79
  - mode, 79
  - offsetX, 80
  - offsetY, 80
  - pixelFormat, 80
  - reserved, 80
  - width, 80
- FlyCapture2::Format7Info, 81
  - Format7Info, 82
  - imageHStepSize, 82
  - imageVStepSize, 82
  - maxHeight, 82
  - maxPacketSize, 82
  - maxWidth, 82
  - minPacketSize, 82
  - mode, 82
  - offsetHStepSize, 82
  - offsetVStepSize, 82
  - packetSize, 82
  - percentage, 83
  - pixelFormatBitField, 83
  - reserved, 83
- FlyCapture2::Format7PacketInfo, 84
  - Format7PacketInfo, 84
  - maxBytesPerPacket, 84
  - recommendedBytesPerPacket, 84
  - reserved, 84
  - unitBytesPerPacket, 84
- FlyCapture2::Image, 86
  - ~Image, 90
  - CalculateStatistics, 90
  - Convert, 90
  - DeepCopy, 91
  - DetermineBitsPerPixel, 91
  - GetBayerTileFormat, 91
  - GetBitsPerPixel, 91
  - GetColorProcessing, 91
  - GetCols, 92
  - GetData, 92
  - GetDataSize, 92
  - GetDefaultColorProcessing, 92
  - GetDefaultOutputFormat, 92
  - GetDimensions, 93
  - GetMetadata, 93

- GetPixelFormat, 93
- GetRows, 93
- GetStride, 93
- GetTimeStamp, 94
- Image, 89
- Iso, 98
- operator(), 94
- operator=, 94
- ReleaseBuffer, 94
- Save, 95, 96
- SetColorProcessing, 96
- SetData, 97
- SetDefaultColorProcessing, 97
- SetDefaultOutputFormat, 97
- SetDimensions, 98
- FlyCapture2::ImageMetadata, 99
  - embeddedBrightness, 100
  - embeddedExposure, 100
  - embeddedFrameCounter, 100
  - embeddedGain, 100
  - embeddedGPIOPinState, 100
  - embeddedROIPosition, 100
  - embeddedShutter, 100
  - embeddedStrobePattern, 100
  - embeddedTimeStamp, 100
  - embeddedWhiteBalance, 100
  - ImageMetadata, 100
  - reserved, 100
- FlyCapture2::ImageStatistics, 102
  - ~ImageStatistics, 104
  - BLUE, 103
  - DisableAll, 104
  - EnableAll, 104
  - EnableGreyOnly, 104
  - EnableHSLOnly, 104
  - EnableRGBOnly, 104
  - GetChannelStatus, 104
  - GetHistogram, 105
  - GetMean, 105
  - GetNumPixelValues, 105
  - GetPixelValueRange, 105
  - GetRange, 106
  - GetStatistics, 106
  - GREEN, 103
  - GREY, 103
  - HUE, 103
  - ImageStatistics, 104
  - ImageStatsCalculator, 107
  - LIGHTNESS, 103
  - NUM\_STATISTICS\_CHANNELS, 103
  - operator=, 106
  - RED, 103
  - SATURATION, 103
  - SetChannelStatus, 107
  - StatisticsChannel, 103
- FlyCapture2::JPEGOOption, 108
  - JPEGOOption, 108
  - progressive, 108
  - quality, 108
  - reserved, 108
- FlyCapture2::JPG2Option, 109
  - JPG2Option, 109
  - quality, 109
  - reserved, 109
- FlyCapture2::LUTData, 110
  - enabled, 110
  - inputBitDepth, 110
  - LUTData, 110
  - numBanks, 110
  - numChannels, 111
  - numEntries, 111
  - outputBitDepth, 111
  - reserved, 111
  - supported, 111
- FlyCapture2::PGMOption, 112
  - binaryFile, 112
  - PGMOption, 112
  - reserved, 112
- FlyCapture2::PGRGuid, 113
  - operator=, 113
  - operator==, 113
  - PGRGuid, 113
  - value, 114
- FlyCapture2::PNGOption, 115
  - compressionLevel, 115
  - interlaced, 115
  - PNGOption, 115
  - reserved, 115
- FlyCapture2::PPMOption, 116
  - binaryFile, 116
  - PPMOption, 116
  - reserved, 116
- FlyCapture2::Property, 117
  - absControl, 117
  - absValue, 117
  - autoManualMode, 118
  - onePush, 118
  - onOff, 118
  - present, 118
  - Property, 117
  - reserved, 118
  - type, 118
  - valueA, 118
  - valueB, 118
- FlyCapture2::PropertyInfo, 119
  - absMax, 120
  - absMin, 120
  - absValSupported, 120

- autoSupported, 120
- manualSupported, 120
- max, 120
- min, 120
- onePushSupported, 120
- onOffSupported, 120
- present, 120
- PropertyInfo, 120
- pUnitAbbr, 120
- pUnits, 121
- readOutSupported, 121
- reserved, 121
- type, 121
- FlyCapture2::StrobeControl, 122
  - delay, 122
  - duration, 122
  - onOff, 122
  - polarity, 122
  - reserved, 123
  - source, 123
  - StrobeControl, 122
- FlyCapture2::StrobeInfo, 124
  - maxValue, 124
  - minValue, 124
  - onOffSupported, 124
  - polaritySupported, 125
  - present, 125
  - readOutSupported, 125
  - reserved, 125
  - source, 125
  - StrobeInfo, 124
- FlyCapture2::SystemInfo, 126
  - byteOrder, 126
  - cpuDescription, 126
  - driverList, 127
  - gpuDescription, 127
  - libraryList, 127
  - numCpuCores, 127
  - osDescription, 127
  - osType, 127
  - reserved, 127
  - screenHeight, 127
  - screenWidth, 127
  - sysMemSize, 127
- FlyCapture2::TIFFOption, 128
  - ADOBE\_DEFLATE, 128
  - CCITTFAX3, 128
  - CCITTFAX4, 128
  - compression, 129
  - CompressionMethod, 128
  - DEFLATE, 128
  - JPEG, 128
  - LZW, 128
  - NONE, 128
  - PACKBITS, 128
  - reserved, 129
  - TIFFOption, 129
- FlyCapture2::TimeStamp, 130
  - cycleCount, 130
  - cycleOffset, 130
  - cycleSeconds, 130
  - microSeconds, 130
  - reserved, 130
  - seconds, 131
  - TimeStamp, 130
- FlyCapture2::TopologyNode, 132
  - ~TopologyNode, 133
  - AddChild, 134
  - AddPort, 134
  - AssignGuidToNode, 134
  - BUS, 133
  - CAMERA, 133
  - COMPUTER, 133
  - CONNECTED\_TO\_CHILD, 133
  - CONNECTED\_TO\_PARENT, 133
  - GetChild, 134
  - GetDeviceId, 134
  - GetGuid, 134
  - GetInterfaceType, 134
  - GetNodeType, 134
  - GetNumChildren, 135
  - GetNumPorts, 135
  - GetPortType, 135
  - NODE, 133
  - NodeType, 133
  - NOT\_CONNECTED, 133
  - operator=, 135
  - PortType, 133
  - TopologyNode, 133
- FlyCapture2::TriggerMode, 136
  - mode, 136
  - onOff, 136
  - parameter, 136
  - polarity, 136
  - reserved, 136
  - source, 136
  - TriggerMode, 136
- FlyCapture2::TriggerModeInfo, 138
  - modeMask, 138
  - onOffSupported, 138
  - polaritySupported, 138
  - present, 138
  - readOutSupported, 139
  - reserved, 139
  - softwareTriggerSupported, 139
  - sourceMask, 139
  - TriggerModeInfo, 138
  - valueReadable, 139

- FlyCapture2::Utilities, 140
  - GetLibraryVersion, 140
  - GetSystemInfo, 140
  - LaunchBrowser, 140
  - LaunchCommand, 141
  - LaunchCommandAsync, 141
  - LaunchHelp, 141
- FLYCAPTURE2\_API
  - FlyCapture2Platform.h, 156
- FlyCapture2Defs.h, 148
  - FULL\_32BIT\_VALUE, 154
  - NULL, 154
- FlyCapture2GUI.h, 155
- FlyCapture2Platform.h, 156
  - FLYCAPTURE2\_API, 156
- FOCUS
  - FlyCapture2, 24
- Format7ImageSettings
  - FlyCapture2::Format7ImageSettings, 79
- Format7Info
  - FlyCapture2::Format7Info, 82
- Format7PacketInfo
  - FlyCapture2::Format7PacketInfo, 84
- FRAME\_RATE
  - FlyCapture2, 24
- frameCounter
  - FlyCapture2::EmbeddedImageInfo, 70
- FrameRate
  - FlyCapture2, 20
- frameRate
  - FlyCapture2::AVIOption, 27
  - VideoModes, 142
- FRAMERATE\_120
  - FlyCapture2, 21
- FRAMERATE\_15
  - FlyCapture2, 20
- FRAMERATE\_1\_875
  - FlyCapture2, 20
- FRAMERATE\_240
  - FlyCapture2, 21
- FRAMERATE\_30
  - FlyCapture2, 20
- FRAMERATE\_3\_75
  - FlyCapture2, 20
- FRAMERATE\_60
  - FlyCapture2, 21
- FRAMERATE\_7\_5
  - FlyCapture2, 20
- FRAMERATE\_FORCE\_32BITS
  - FlyCapture2, 21
- FRAMERATE\_FORMAT7
  - FlyCapture2, 21
- FROM\_FILE\_EXT
  - FlyCapture2, 22
- FULL\_32BIT\_VALUE
  - FlyCapture2Defs.h, 154
- GAIN
  - FlyCapture2, 24
- gain
  - FlyCapture2::EmbeddedImageInfo, 70
- GAMMA
  - FlyCapture2, 24
- GBRG
  - FlyCapture2, 18
- GetAbsBrightness
  - PGRDirectShow.h, 163
- GetAbsBrightnessRange
  - PGRDirectShow.h, 163
- GetAbsExposure
  - PGRDirectShow.h, 163
- GetAbsExposureRange
  - PGRDirectShow.h, 163
- GetAbsGain
  - PGRDirectShow.h, 163
- GetAbsGainRange
  - PGRDirectShow.h, 163
- GetAbsGamma
  - PGRDirectShow.h, 163
- GetAbsGammaRange
  - PGRDirectShow.h, 163
- GetAbsHue
  - PGRDirectShow.h, 163
- GetAbsHueRange
  - PGRDirectShow.h, 163
- GetAbsPan
  - PGRDirectShow.h, 163
- GetAbsPanRange
  - PGRDirectShow.h, 163
- GetAbsSaturation
  - PGRDirectShow.h, 163
- GetAbsSaturationRange
  - PGRDirectShow.h, 163
- GetAbsSharpness
  - PGRDirectShow.h, 163
- GetAbsSharpnessRange
  - PGRDirectShow.h, 163
- GetAbsShutter
  - PGRDirectShow.h, 163
- GetAbsShutterRange
  - PGRDirectShow.h, 163
- GetAbsTilt
  - PGRDirectShow.h, 163
- GetAbsTiltRange
  - PGRDirectShow.h, 163
- GetAbsWhiteBalance
  - PGRDirectShow.h, 163
- GetAbsWhiteBalanceRange



- PGRDirectShow.h, 163
- GetActiveLUTBank
  - FlyCapture2::Camera, 41
- GetBayerTileFormat
  - FlyCapture2::Image, 91
- GetBitsPerPixel
  - FlyCapture2::Image, 91
- GetBrightness
  - PGRDirectShow.h, 163
- GetBrightnessRange
  - PGRDirectShow.h, 163
- GetBuildDate
  - FlyCapture2::Error, 73
- GetCameraFromIndex
  - FlyCapture2::BusManager, 32
- GetCameraFromSerialNumber
  - FlyCapture2::BusManager, 32
- GetCameraInfo
  - FlyCapture2::Camera, 41
- GetCameraSerialNumberFromIndex
  - FlyCapture2::BusManager, 32
- GetCause
  - FlyCapture2::Error, 73
- GetChannelStatus
  - FlyCapture2::ImageStatistics, 104
- GetChild
  - FlyCapture2::TopologyNode, 134
- GetColorProcessing
  - FlyCapture2::Image, 91
- GetCols
  - FlyCapture2::Image, 92
- GetConfiguration
  - FlyCapture2::Camera, 41
- GetCustomImage
  - PGRDirectShow.h, 163
- GetCustomImageMode
  - PGRDirectShow.h, 163
- GetData
  - FlyCapture2::Image, 92
- GetDataSize
  - FlyCapture2::Image, 92
- GetDefaultColorProcessing
  - FlyCapture2::Image, 92
- GetDefaultOutputFormat
  - FlyCapture2::Image, 92
- GetDescription
  - FlyCapture2::Error, 74
- GetDeviceId
  - FlyCapture2::TopologyNode, 134
- GetDimensions
  - FlyCapture2::Image, 93
- GetEmbeddedImageInfo
  - FlyCapture2::Camera, 42
- GetExposure
  - PGRDirectShow.h, 163
- GetExposureRange
  - PGRDirectShow.h, 163
- GetFilename
  - FlyCapture2::Error, 74
- GetFormat7Configuration
  - FlyCapture2::Camera, 42
- GetFormat7Info
  - FlyCapture2::Camera, 42
- GetGain
  - PGRDirectShow.h, 163
- GetGainRange
  - PGRDirectShow.h, 163
- GetGamma
  - PGRDirectShow.h, 163
- GetGammaRange
  - PGRDirectShow.h, 163
- GetGPIOPinDirection
  - FlyCapture2::Camera, 43
- GetGuid
  - FlyCapture2::TopologyNode, 134
- GetHistogram
  - FlyCapture2::ImageStatistics, 105
- GetHue
  - PGRDirectShow.h, 163
- GetHueRange
  - PGRDirectShow.h, 163
- GetImageFormat
  - PGRDirectShow.h, 163
- GetInterfaceType
  - FlyCapture2::TopologyNode, 134
- GetLibraryVersion
  - FlyCapture2::Utilities, 140
- GetLine
  - FlyCapture2::Error, 74
- GetLUTBankInfo
  - FlyCapture2::Camera, 43
- GetLUTChannel
  - FlyCapture2::Camera, 44
- GetLUTInfo
  - FlyCapture2::Camera, 44
- GetMean
  - FlyCapture2::ImageStatistics, 105
- GetMemoryChannel
  - FlyCapture2::Camera, 44
- GetMemoryChannelInfo
  - FlyCapture2::Camera, 45
- GetMetadata
  - FlyCapture2::Image, 93
- GetNodeType
  - FlyCapture2::TopologyNode, 134
- GetNumChildren
  - FlyCapture2::TopologyNode, 135
- GetNumOfCameras



- FlyCapture2::BusManager, 33
- GetNumPixelValues
  - FlyCapture2::ImageStatistics, 105
- GetNumPorts
  - FlyCapture2::TopologyNode, 135
- GetOutputVerticalFlip
  - PGRDirectShow.h, 163
- GetPan
  - PGRDirectShow.h, 163
- GetPanRange
  - PGRDirectShow.h, 163
- GetPixelFormat
  - FlyCapture2::Image, 93
- GetPixelValueRange
  - FlyCapture2::ImageStatistics, 105
- GetPortType
  - FlyCapture2::TopologyNode, 135
- GetProperty
  - FlyCapture2::Camera, 45
- GetPropertyInfo
  - FlyCapture2::Camera, 45
- GetRange
  - FlyCapture2::ImageStatistics, 106
- GetRegisterString
  - FlyCapture2::Camera, 46
- GetRows
  - FlyCapture2::Image, 93
- GetSaturation
  - PGRDirectShow.h, 163
- GetSaturationRange
  - PGRDirectShow.h, 163
- GetSharpness
  - PGRDirectShow.h, 163
- GetSharpnessRange
  - PGRDirectShow.h, 163
- GetShutter
  - PGRDirectShow.h, 163
- GetShutterRange
  - PGRDirectShow.h, 163
- GetStatistics
  - FlyCapture2::ImageStatistics, 106
- GetStride
  - FlyCapture2::Image, 93
- GetStrobe
  - FlyCapture2::Camera, 46
- GetStrobeInfo
  - FlyCapture2::Camera, 46
- GetSystemInfo
  - FlyCapture2::Utilities, 140
- GetTilt
  - PGRDirectShow.h, 163
- GetTiltRange
  - PGRDirectShow.h, 163
- GetTimeStamp
  - FlyCapture2::Image, 94
- GetTopology
  - FlyCapture2::BusManager, 33
- GetTriggerDelay
  - FlyCapture2::Camera, 47
- GetTriggerDelayInfo
  - FlyCapture2::Camera, 47
- GetTriggerMode
  - FlyCapture2::Camera, 47
- GetTriggerModeInfo
  - FlyCapture2::Camera, 48
- GetType
  - FlyCapture2::Error, 74
- GetVideoModeAndFrameRate
  - FlyCapture2::Camera, 48
- GetVideoModeAndFrameRateInfo
  - FlyCapture2::Camera, 49
- GetWhiteBalance
  - PGRDirectShow.h, 163
- GetWhiteBalanceRange
  - PGRDirectShow.h, 163
- GPIOPinState
  - FlyCapture2::EmbeddedImageInfo, 70
- gpuDescription
  - FlyCapture2::SystemInfo, 127
- GRAB\_MODE\_FORCE\_32BITS
  - FlyCapture2, 21
- GRAB\_TIMEOUT\_FORCE\_32BITS
  - FlyCapture2, 21
- GrabMode
  - FlyCapture2, 21
- grabMode
  - FlyCapture2::FC2Config, 77
- GrabTimeout
  - FlyCapture2, 21
- grabTimeout
  - FlyCapture2::FC2Config, 77
- GRBG
  - FlyCapture2, 18
- GREEN
  - FlyCapture2::ImageStatistics, 103
- GREY
  - FlyCapture2::ImageStatistics, 103
- height
  - FlyCapture2::Format7ImageSettings, 79
  - VideoModes, 142
- Hide
  - FlyCapture2::CameraControlDlg, 60
- HQ\_LINEAR
  - FlyCapture2, 19
- HUE
  - FlyCapture2, 24
  - FlyCapture2::ImageStatistics, 103

- IID\_IFlyCaptureProperties
  - PGRDirectShow.h, 163
- iideVer
  - FlyCapture2::CameraInfo, 63
- Image
  - FlyCapture2::Image, 89
- Image.h, 157
- IMAGE\_FILE\_FORMAT\_FORCE\_32BITS
  - FlyCapture2, 22
- ImageEventCallback
  - FlyCapture2, 17
- ImageFileFormat
  - FlyCapture2, 21
- imageHStepSize
  - FlyCapture2::Format7Info, 82
- ImageMetadata
  - FlyCapture2::ImageMetadata, 100
- ImageStatistics
  - FlyCapture2::ImageStatistics, 104
- ImageStatistics.h, 158
- ImageStatsCalculator
  - FlyCapture2::ImageStatistics, 107
- imageVStepSize
  - FlyCapture2::Format7Info, 82
- inputBitDepth
  - FlyCapture2::LUTData, 110
- INTERFACE\_GIGE
  - FlyCapture2, 22
- INTERFACE\_IEEE1394
  - FlyCapture2, 22
- INTERFACE\_TYPE\_FORCE\_32BITS
  - FlyCapture2, 22
- INTERFACE\_UNKNOWN
  - FlyCapture2, 22
- INTERFACE\_USB2
  - FlyCapture2, 22
- InterfaceType
  - FlyCapture2, 22
- interfaceType
  - FlyCapture2::CameraInfo, 63
- interlaced
  - FlyCapture2::PNGOption, 115
- InternalError
  - FlyCapture2::Error, 75
- IRIS
  - FlyCapture2, 24
- isColorCamera
  - FlyCapture2::CameraInfo, 63
- IsConnected
  - FlyCapture2::Camera, 49
- Iso
  - FlyCapture2::Image, 98
- isochBusSpeed
  - FlyCapture2::FC2Config, 77
- IsVisible
  - FlyCapture2::CameraControlDlg, 60
- JPEG
  - FlyCapture2, 22
  - FlyCapture2::TIFFOption, 128
- JPEG2000
  - FlyCapture2, 22
- JPEGOption
  - FlyCapture2::JPEGOption, 108
- JPG2Option
  - FlyCapture2::JPG2Option, 109
- LaunchBrowser
  - FlyCapture2::Utilities, 140
- LaunchCommand
  - FlyCapture2::Utilities, 141
- LaunchCommandAsync
  - FlyCapture2::Utilities, 141
- LaunchHelp
  - FlyCapture2::Utilities, 141
- libraryList
  - FlyCapture2::SystemInfo, 127
- LIGHTNESS
  - FlyCapture2::ImageStatistics, 103
- LINUX\_X64
  - FlyCapture2, 23
- LINUX\_X86
  - FlyCapture2, 23
- LUTData
  - FlyCapture2::LUTData, 110
- LZW
  - FlyCapture2::TIFFOption, 128
- MAC
  - FlyCapture2, 23
- major
  - FlyCapture2::FC2Version, 78
- manualSupported
  - FlyCapture2::PropertyInfo, 120
- max
  - FlyCapture2::PropertyInfo, 120
- maxBytesPerPacket
  - FlyCapture2::Format7PacketInfo, 84
- maxHeight
  - FlyCapture2::Format7Info, 82
- maximumBusSpeed
  - FlyCapture2::CameraInfo, 63
- maxPacketSize
  - FlyCapture2::Format7Info, 82
- maxValue
  - FlyCapture2::StrobeInfo, 124
- maxWidth
  - FlyCapture2::Format7Info, 82

- microSeconds
  - FlyCapture2::TimeStamp, [130](#)
- min
  - FlyCapture2::PropertyInfo, [120](#)
- minor
  - FlyCapture2::FC2Version, [78](#)
- minPacketSize
  - FlyCapture2::Format7Info, [82](#)
- minValue
  - FlyCapture2::StrobeInfo, [124](#)
- Mode
  - FlyCapture2, [22](#)
- mode
  - FlyCapture2::Format7ImageSettings, [79](#)
  - FlyCapture2::Format7Info, [82](#)
  - FlyCapture2::TriggerMode, [136](#)
- MODE\_0
  - FlyCapture2, [22](#)
- MODE\_1
  - FlyCapture2, [22](#)
- MODE\_10
  - FlyCapture2, [22](#)
- MODE\_11
  - FlyCapture2, [22](#)
- MODE\_12
  - FlyCapture2, [22](#)
- MODE\_13
  - FlyCapture2, [22](#)
- MODE\_14
  - FlyCapture2, [23](#)
- MODE\_15
  - FlyCapture2, [23](#)
- MODE\_16
  - FlyCapture2, [23](#)
- MODE\_17
  - FlyCapture2, [23](#)
- MODE\_18
  - FlyCapture2, [23](#)
- MODE\_19
  - FlyCapture2, [23](#)
- MODE\_2
  - FlyCapture2, [22](#)
- MODE\_20
  - FlyCapture2, [23](#)
- MODE\_21
  - FlyCapture2, [23](#)
- MODE\_22
  - FlyCapture2, [23](#)
- MODE\_23
  - FlyCapture2, [23](#)
- MODE\_24
  - FlyCapture2, [23](#)
- MODE\_25
  - FlyCapture2, [23](#)
- MODE\_26
  - FlyCapture2, [23](#)
- MODE\_27
  - FlyCapture2, [23](#)
- MODE\_28
  - FlyCapture2, [23](#)
- MODE\_29
  - FlyCapture2, [23](#)
- MODE\_3
  - FlyCapture2, [22](#)
- MODE\_30
  - FlyCapture2, [23](#)
- MODE\_31
  - FlyCapture2, [23](#)
- MODE\_4
  - FlyCapture2, [22](#)
- MODE\_5
  - FlyCapture2, [22](#)
- MODE\_6
  - FlyCapture2, [22](#)
- MODE\_7
  - FlyCapture2, [22](#)
- MODE\_8
  - FlyCapture2, [22](#)
- MODE\_9
  - FlyCapture2, [22](#)
- MODE\_FORCE\_32BITS
  - FlyCapture2, [23](#)
- modelName
  - FlyCapture2::CameraInfo, [63](#)
- modeMask
  - FlyCapture2::TriggerModeInfo, [138](#)
- NEAREST\_NEIGHBOR
  - FlyCapture2, [19](#)
- NO\_COLOR\_PROCESSING
  - FlyCapture2, [19](#)
- NODE
  - FlyCapture2::TopologyNode, [133](#)
- NodeType
  - FlyCapture2::TopologyNode, [133](#)
- nodeVendorId
  - FlyCapture2::ConfigROM, [66](#)
- NONE
  - FlyCapture2, [18](#)
  - FlyCapture2::TIFFOption, [128](#)
- NOT\_CONNECTED
  - FlyCapture2::TopologyNode, [133](#)
- NULL
  - FlyCapture2Defs.h, [154](#)
- NUM\_FRAMERATES
  - FlyCapture2, [21](#)
- NUM\_MODES
  - FlyCapture2, [23](#)

- NUM\_PIXEL\_FORMATS
  - FlyCapture2, [24](#)
- NUM\_STATISTICS\_CHANNELS
  - FlyCapture2::ImageStatistics, [103](#)
- NUM\_VIDEOMODES
  - FlyCapture2, [25](#)
- numBanks
  - FlyCapture2::LUTData, [110](#)
- numBuffers
  - FlyCapture2::FC2Config, [77](#)
- numChannels
  - FlyCapture2::LUTData, [111](#)
- numCpuCores
  - FlyCapture2::SystemInfo, [127](#)
- numEntries
  - FlyCapture2::LUTData, [111](#)
- numFormats
  - DCAMFormats, [68](#)
- numImageNotifications
  - FlyCapture2::FC2Config, [77](#)
- offsetHStepSize
  - FlyCapture2::Format7Info, [82](#)
- offsetVStepSize
  - FlyCapture2::Format7Info, [82](#)
- offsetX
  - FlyCapture2::Format7ImageSettings, [80](#)
- offsetY
  - FlyCapture2::Format7ImageSettings, [80](#)
- onePush
  - FlyCapture2::Property, [118](#)
- onePushSupported
  - FlyCapture2::PropertyInfo, [120](#)
- onOff
  - FlyCapture2::EmbeddedImageInfoProperty, [71](#)
  - FlyCapture2::Property, [118](#)
  - FlyCapture2::StrobeControl, [122](#)
  - FlyCapture2::TriggerMode, [136](#)
- onOffSupported
  - FlyCapture2::PropertyInfo, [120](#)
  - FlyCapture2::StrobeInfo, [124](#)
  - FlyCapture2::TriggerModeInfo, [138](#)
- operator()
  - FlyCapture2::Image, [94](#)
- operator=
  - FlyCapture2::Error, [74](#)
  - FlyCapture2::Image, [94](#)
  - FlyCapture2::ImageStatistics, [106](#)
  - FlyCapture2::PGRGuid, [113](#)
  - FlyCapture2::TopologyNode, [135](#)
- operator==
  - FlyCapture2::Error, [75](#)
  - FlyCapture2::PGRGuid, [113](#)
- osDescription
  - FlyCapture2::SystemInfo, [127](#)
- OSType
  - FlyCapture2, [23](#)
- osType
  - FlyCapture2::SystemInfo, [127](#)
- OSTYPE\_FORCE\_32BITS
  - FlyCapture2, [23](#)
- outputBitDepth
  - FlyCapture2::LUTData, [111](#)
- PACKBITS
  - FlyCapture2::TIFFOption, [128](#)
- packetSize
  - FlyCapture2::Format7Info, [82](#)
- PAN
  - FlyCapture2, [24](#)
- parameter
  - FlyCapture2::TriggerMode, [136](#)
- percentage
  - FlyCapture2::Format7Info, [83](#)
- PGM
  - FlyCapture2, [22](#)
- PGMOption
  - FlyCapture2::PGMOption, [112](#)
- PGRDirectShow.h, [159](#)
  - GetAbsBrightness, [163](#)
  - GetAbsBrightnessRange, [163](#)
  - GetAbsExposure, [163](#)
  - GetAbsExposureRange, [163](#)
  - GetAbsGain, [163](#)
  - GetAbsGainRange, [163](#)
  - GetAbsGamma, [163](#)
  - GetAbsGammaRange, [163](#)
  - GetAbsHue, [163](#)
  - GetAbsHueRange, [163](#)
  - GetAbsPan, [163](#)
  - GetAbsPanRange, [163](#)
  - GetAbsSaturation, [163](#)
  - GetAbsSaturationRange, [163](#)
  - GetAbsSharpness, [163](#)
  - GetAbsSharpnessRange, [163](#)
  - GetAbsShutter, [163](#)
  - GetAbsShutterRange, [163](#)
  - GetAbsTilt, [163](#)
  - GetAbsTiltRange, [163](#)
  - GetAbsWhiteBalance, [163](#)
  - GetAbsWhiteBalanceRange, [163](#)
  - GetBrightness, [163](#)
  - GetBrightnessRange, [163](#)
  - GetCustomImage, [163](#)
  - GetCustomImageMode, [163](#)
  - GetExposure, [163](#)
  - GetExposureRange, [163](#)
  - GetGain, [163](#)

GetGainRange, 163  
 GetGamma, 163  
 GetGammaRange, 163  
 GetHue, 163  
 GetHueRange, 163  
 GetImageFormat, 163  
 GetOutputVerticalFlip, 163  
 GetPan, 163  
 GetPanRange, 163  
 GetSaturation, 163  
 GetSaturationRange, 163  
 GetSharpness, 163  
 GetSharpnessRange, 163  
 GetShutter, 163  
 GetShutterRange, 163  
 GetTilt, 163  
 GetTiltRange, 163  
 GetWhiteBalance, 163  
 GetWhiteBalanceRange, 163  
 IID\_IFlyCaptureProperties, 163  
 QueryCustomImage, 163  
 ReadRegister, 163  
 SetAbsBrightness, 163  
 SetAbsExposure, 163  
 SetAbsGain, 163  
 SetAbsGamma, 163  
 SetAbsHue, 163  
 SetAbsPan, 163  
 SetAbsSaturation, 163  
 SetAbsSharpness, 163  
 SetAbsShutter, 163  
 SetAbsTilt, 163  
 SetAbsWhiteBalance, 163  
 SetBrightness, 163  
 SetCustomImage, 163  
 SetCustomImageMode, 163  
 SetExposure, 163  
 SetGain, 163  
 SetGamma, 163  
 SetHue, 163  
 SetImageFormat, 163  
 SetOutputVerticalFlip, 163  
 SetPan, 163  
 SetSaturation, 163  
 SetSharpness, 163  
 SetShutter, 163  
 SetTilt, 163  
 SetWhiteBalance, 163  
 WriteRegister, 163  
 PGRERROR\_BUFFER\_TOO\_SMALL  
     FlyCapture2, 20  
 PGRERROR\_BUS\_MASTER\_FAILED  
     FlyCapture2, 20  
 PGRERROR\_FAILED

    FlyCapture2, 19  
 PGRERROR\_FAILED\_BUS\_MASTER\_CONNECTION  
     FlyCapture2, 19  
 PGRERROR\_FAILED\_GUID  
     FlyCapture2, 20  
 PGRERROR\_FORCE\_32BITS  
     FlyCapture2, 20  
 PGRERROR\_IIDC\_FAILED  
     FlyCapture2, 20  
 PGRERROR\_IMAGE\_CONVERSION\_FAILED  
     FlyCapture2, 20  
 PGRERROR\_IMAGE\_LIBRARY\_FAILURE  
     FlyCapture2, 20  
 PGRERROR\_INIT\_FAILED  
     FlyCapture2, 19  
 PGRERROR\_INVALID\_BUS\_MANAGER  
     FlyCapture2, 19  
 PGRERROR\_INVALID\_GENERATION  
     FlyCapture2, 20  
 PGRERROR\_INVALID\_MODE  
     FlyCapture2, 20  
 PGRERROR\_INVALID\_PACKET\_SIZE  
     FlyCapture2, 20  
 PGRERROR\_INVALID\_PARAMETER  
     FlyCapture2, 19  
 PGRERROR\_INVALID\_SETTINGS  
     FlyCapture2, 19  
 PGRERROR\_ISOCH\_ALREADY\_STARTED  
     FlyCapture2, 20  
 PGRERROR\_ISOCH\_BANDWIDTH\_EXCEEDED  
     FlyCapture2, 20  
 PGRERROR\_ISOCH\_FAILED  
     FlyCapture2, 20  
 PGRERROR\_ISOCH\_NOT\_STARTED  
     FlyCapture2, 20  
 PGRERROR\_ISOCH\_RETRIEVE\_BUFFER\_FAILED  
     FlyCapture2, 20  
 PGRERROR\_ISOCH\_START\_FAILED  
     FlyCapture2, 20  
 PGRERROR\_ISOCH\_STOP\_FAILED  
     FlyCapture2, 20  
 PGRERROR\_ISOCH\_SYNC\_FAILED  
     FlyCapture2, 20  
 PGRERROR\_LOW\_LEVEL\_FAILURE  
     FlyCapture2, 19  
 PGRERROR\_LUT\_FAILED  
     FlyCapture2, 20  
 PGRERROR\_MEMORY\_ALLOCATION\_FAILED  
     FlyCapture2, 19  
 PGRERROR\_NOT\_CONNECTED  
     FlyCapture2, 19  
 PGRERROR\_NOT\_FOUND  
     FlyCapture2, 20  
 PGRERROR\_NOT\_IMPLEMENTED

- FlyCapture2, [19](#)
- PGRError\_NOT\_IN\_FORMAT7
  - FlyCapture2, [20](#)
- PGRError\_NOT\_INITIALIZED
  - FlyCapture2, [19](#)
- PGRError\_NOT\_SUPPORTED
  - FlyCapture2, [20](#)
- PGRError\_OK
  - FlyCapture2, [19](#)
- PGRError\_PROPERTY\_FAILED
  - FlyCapture2, [20](#)
- PGRError\_PROPERTY\_NOT\_PRESENT
  - FlyCapture2, [20](#)
- PGRError\_READ\_REGISTER\_FAILED
  - FlyCapture2, [20](#)
- PGRError\_REGISTER\_FAILED
  - FlyCapture2, [20](#)
- PGRError\_STROBE\_FAILED
  - FlyCapture2, [20](#)
- PGRError\_TIMEOUT
  - FlyCapture2, [20](#)
- PGRError\_TRIGGER\_FAILED
  - FlyCapture2, [20](#)
- PGRError\_UNDEFINED
  - FlyCapture2, [19](#)
- PGRError\_WRITE\_REGISTER\_FAILED
  - FlyCapture2, [20](#)
- PGRGuid
  - FlyCapture2::PGRGuid, [113](#)
- PIXEL\_FORMAT\_411YUV8
  - FlyCapture2, [23](#)
- PIXEL\_FORMAT\_422YUV8
  - FlyCapture2, [24](#)
- PIXEL\_FORMAT\_444YUV8
  - FlyCapture2, [24](#)
- PIXEL\_FORMAT\_BGR
  - FlyCapture2, [24](#)
- PIXEL\_FORMAT\_BGRU
  - FlyCapture2, [24](#)
- PIXEL\_FORMAT\_MONO12
  - FlyCapture2, [24](#)
- PIXEL\_FORMAT\_MONO16
  - FlyCapture2, [24](#)
- PIXEL\_FORMAT\_MONO8
  - FlyCapture2, [23](#)
- PIXEL\_FORMAT\_RAW12
  - FlyCapture2, [24](#)
- PIXEL\_FORMAT\_RAW16
  - FlyCapture2, [24](#)
- PIXEL\_FORMAT\_RAW8
  - FlyCapture2, [24](#)
- PIXEL\_FORMAT\_RGB
  - FlyCapture2, [24](#)
- PIXEL\_FORMAT\_RGB16
  - FlyCapture2, [24](#)
- PIXEL\_FORMAT\_RGB8
  - FlyCapture2, [24](#)
- PIXEL\_FORMAT\_RGBU
  - FlyCapture2, [24](#)
- PIXEL\_FORMAT\_S\_MONO16
  - FlyCapture2, [24](#)
- PIXEL\_FORMAT\_S\_RGB16
  - FlyCapture2, [24](#)
- PixelFormat
  - FlyCapture2, [23](#)
- pixelFormat
  - FlyCapture2::Format7ImageSettings, [80](#)
- pixelFormatBitField
  - FlyCapture2::Format7Info, [83](#)
- PNG
  - FlyCapture2, [22](#)
- PNGOption
  - FlyCapture2::PNGOption, [115](#)
- polarity
  - FlyCapture2::StrobeControl, [122](#)
  - FlyCapture2::TriggerMode, [136](#)
- polaritySupported
  - FlyCapture2::StrobeInfo, [125](#)
  - FlyCapture2::TriggerModeInfo, [138](#)
- PortType
  - FlyCapture2::TopologyNode, [133](#)
- PPM
  - FlyCapture2, [22](#)
- PPMOption
  - FlyCapture2::PPMOption, [116](#)
- present
  - FlyCapture2::Property, [118](#)
  - FlyCapture2::PropertyInfo, [120](#)
  - FlyCapture2::StrobeInfo, [125](#)
  - FlyCapture2::TriggerModeInfo, [138](#)
- PrintErrorTrace
  - FlyCapture2::Error, [75](#)
- progressive
  - FlyCapture2::JPEGOption, [108](#)
- Property
  - FlyCapture2::Property, [117](#)
- PROPERTY\_TYPE\_FORCE\_32BITS
  - FlyCapture2, [24](#)
- PropertyInfo
  - FlyCapture2::PropertyInfo, [120](#)
- PropertyType
  - FlyCapture2, [24](#)
- pszKeyword
  - FlyCapture2::ConfigROM, [66](#)
- pUnitAbbr
  - FlyCapture2::PropertyInfo, [120](#)
- pUnits
  - FlyCapture2::PropertyInfo, [121](#)

- quality
  - FlyCapture2::JPEGOOption, 108
  - FlyCapture2::JPG2Option, 109
- QueryCustomImage
  - PGRDirectShow.h, 163
- RAW
  - FlyCapture2, 22
- readOutSupported
  - FlyCapture2::PropertyInfo, 121
  - FlyCapture2::StrobeInfo, 125
  - FlyCapture2::TriggerModeInfo, 139
- ReadRegister
  - FlyCapture2::Camera, 49
  - PGRDirectShow.h, 163
- ReadRegisterBlock
  - FlyCapture2::Camera, 50
- recommendedBytesPerPacket
  - FlyCapture2::Format7PacketInfo, 84
- RED
  - FlyCapture2::ImageStatistics, 103
- RegisterCallback
  - FlyCapture2::BusManager, 33
- ReleaseBuffer
  - FlyCapture2::Image, 94
- REMOVAL
  - FlyCapture2, 18
- reserved
  - FlyCapture2::AVIOOption, 27
  - FlyCapture2::CameraInfo, 63
  - FlyCapture2::ConfigROM, 66
  - FlyCapture2::FC2Config, 77
  - FlyCapture2::Format7ImageSettings, 80
  - FlyCapture2::Format7Info, 83
  - FlyCapture2::Format7PacketInfo, 84
  - FlyCapture2::ImageMetadata, 100
  - FlyCapture2::JPEGOOption, 108
  - FlyCapture2::JPG2Option, 109
  - FlyCapture2::LUTData, 111
  - FlyCapture2::PGMOption, 112
  - FlyCapture2::PNGOption, 115
  - FlyCapture2::PPMOption, 116
  - FlyCapture2::Property, 118
  - FlyCapture2::PropertyInfo, 121
  - FlyCapture2::StrobeControl, 123
  - FlyCapture2::StrobeInfo, 125
  - FlyCapture2::SystemInfo, 127
  - FlyCapture2::TIFFOption, 129
  - FlyCapture2::TimeStamp, 130
  - FlyCapture2::TriggerMode, 136
  - FlyCapture2::TriggerModeInfo, 139
- RestoreFromMemoryChannel
  - FlyCapture2::Camera, 50
- RetrieveBuffer
  - FlyCapture2::Camera, 50
- RGBB
  - FlyCapture2, 18
- RIGOROUS
  - FlyCapture2, 19
- ROIPosition
  - FlyCapture2::EmbeddedImageInfo, 70
- SATURATION
  - FlyCapture2, 24
  - FlyCapture2::ImageStatistics, 103
- Save
  - FlyCapture2::Image, 95, 96
- SaveToMemoryChannel
  - FlyCapture2::Camera, 51
- screenHeight
  - FlyCapture2::SystemInfo, 127
- screenWidth
  - FlyCapture2::SystemInfo, 127
- seconds
  - FlyCapture2::TimeStamp, 131
- sensorInfo
  - FlyCapture2::CameraInfo, 63
- sensorResolution
  - FlyCapture2::CameraInfo, 63
- serialNumber
  - FlyCapture2::CameraInfo, 63
- SetAbsBrightness
  - PGRDirectShow.h, 163
- SetAbsExposure
  - PGRDirectShow.h, 163
- SetAbsGain
  - PGRDirectShow.h, 163
- SetAbsGamma
  - PGRDirectShow.h, 163
- SetAbsHue
  - PGRDirectShow.h, 163
- SetAbsPan
  - PGRDirectShow.h, 163
- SetAbsSaturation
  - PGRDirectShow.h, 163
- SetAbsSharpness
  - PGRDirectShow.h, 163
- SetAbsShutter
  - PGRDirectShow.h, 163
- SetAbsTilt
  - PGRDirectShow.h, 163
- SetAbsWhiteBalance
  - PGRDirectShow.h, 163
- SetActiveLUTBank
  - FlyCapture2::Camera, 51
- SetBrightness
  - PGRDirectShow.h, 163
- SetChannelStatus



- FlyCapture2::ImageStatistics, 107
- SetColorProcessing
  - FlyCapture2::Image, 96
- SetConfiguration
  - FlyCapture2::Camera, 51
- SetCustomImage
  - PGRDirectShow.h, 163
- SetCustomImageMode
  - PGRDirectShow.h, 163
- SetData
  - FlyCapture2::Image, 97
- SetDefaultColorProcessing
  - FlyCapture2::Image, 97
- SetDefaultOutputFormat
  - FlyCapture2::Image, 97
- SetDimensions
  - FlyCapture2::Image, 98
- SetEmbeddedImageInfo
  - FlyCapture2::Camera, 51
- SetExposure
  - PGRDirectShow.h, 163
- SetFormat7Configuration
  - FlyCapture2::Camera, 52
- SetGain
  - PGRDirectShow.h, 163
- SetGamma
  - PGRDirectShow.h, 163
- SetGPIOPinDirection
  - FlyCapture2::Camera, 52
- SetHue
  - PGRDirectShow.h, 163
- SetImageFormat
  - PGRDirectShow.h, 163
- SetLUTChannel
  - FlyCapture2::Camera, 53
- SetOutputVerticalFlip
  - PGRDirectShow.h, 163
- SetPan
  - PGRDirectShow.h, 163
- SetProperty
  - FlyCapture2::Camera, 53
- SetSaturation
  - PGRDirectShow.h, 163
- SetSharpness
  - PGRDirectShow.h, 163
- SetShutter
  - PGRDirectShow.h, 163
- SetStrobe
  - FlyCapture2::Camera, 54
- SetTilt
  - PGRDirectShow.h, 163
- SetTriggerDelay
  - FlyCapture2::Camera, 54
- SetTriggerMode
  - FlyCapture2::Camera, 54
- SetUserBuffers
  - FlyCapture2::Camera, 55
- SetVideoModeAndFrameRate
  - FlyCapture2::Camera, 55
- SetWhiteBalance
  - PGRDirectShow.h, 163
- SHARPNESS
  - FlyCapture2, 24
- Show
  - FlyCapture2::CameraControlDlg, 60
- ShowModal
  - FlyCapture2::CameraSelectionDlg, 64
- SHUTTER
  - FlyCapture2, 24
- shutter
  - FlyCapture2::EmbeddedImageInfo, 70
- sk\_maxStringLength
  - FlyCapture2, 25
- softwareTriggerSupported
  - FlyCapture2::TriggerModeInfo, 139
- source
  - FlyCapture2::StrobeControl, 123
  - FlyCapture2::StrobeInfo, 125
  - FlyCapture2::TriggerMode, 136
- sourceMask
  - FlyCapture2::TriggerModeInfo, 139
- StartCapture
  - FlyCapture2::Camera, 56
- StartSyncCapture
  - FlyCapture2::Camera, 56
- StatisticsChannel
  - FlyCapture2::ImageStatistics, 103
- StopCapture
  - FlyCapture2::Camera, 57
- StrobeControl
  - FlyCapture2::StrobeControl, 122
- StrobeInfo
  - FlyCapture2::StrobeInfo, 124
- strobePattern
  - FlyCapture2::EmbeddedImageInfo, 70
- supported
  - FlyCapture2::LUTData, 111
- sysMemSize
  - FlyCapture2::SystemInfo, 127
- TEMPERATURE
  - FlyCapture2, 24
- TIFF
  - FlyCapture2, 22
- TIFFOption
  - FlyCapture2::TIFFOption, 129
- TILT
  - FlyCapture2, 24



- TIMEOUT\_INFINITE
  - FlyCapture2, [21](#)
- TIMEOUT\_UNSPECIFIED
  - FlyCapture2, [21](#)
- TimeStamp
  - FlyCapture2::TimeStamp, [130](#)
- timestamp
  - FlyCapture2::EmbeddedImageInfo, [70](#)
- TopologyNode
  - FlyCapture2::TopologyNode, [133](#)
- TopologyNode.h, [164](#)
- TRIGGER\_DELAY
  - FlyCapture2, [24](#)
- TRIGGER\_MODE
  - FlyCapture2, [24](#)
- TriggerDelay
  - FlyCapture2, [17](#)
- TriggerDelayInfo
  - FlyCapture2, [17](#)
- TriggerMode
  - FlyCapture2::TriggerMode, [136](#)
- TriggerModeInfo
  - FlyCapture2::TriggerModeInfo, [138](#)
- type
  - FlyCapture2::FC2Version, [78](#)
  - FlyCapture2::Property, [118](#)
  - FlyCapture2::PropertyInfo, [121](#)
- unitBytesPerPacket
  - FlyCapture2::Format7PacketInfo, [84](#)
- unitSpecId
  - FlyCapture2::ConfigROM, [66](#)
- unitSubSWVer
  - FlyCapture2::ConfigROM, [66](#)
- unitSWVer
  - FlyCapture2::ConfigROM, [66](#)
- UNKNOWN\_OS
  - FlyCapture2, [23](#)
- UnregisterCallback
  - FlyCapture2::BusManager, [33](#)
- UNSPECIFIED\_GRAB\_MODE
  - FlyCapture2, [21](#)
- UNSPECIFIED\_PIXEL\_FORMAT
  - FlyCapture2, [24](#)
- UNSPECIFIED\_PROPERTY\_TYPE
  - FlyCapture2, [24](#)
- Utilities.h, [165](#)
- ValidateFormat7Settings
  - FlyCapture2::Camera, [57](#)
- value
  - FlyCapture2::PGRGuid, [114](#)
- valueA
  - FlyCapture2::Property, [118](#)
- valueB
  - FlyCapture2::Property, [118](#)
- valueReadable
  - FlyCapture2::TriggerModeInfo, [139](#)
- vendorName
  - FlyCapture2::CameraInfo, [63](#)
- vendorUniqueInfo\_0
  - FlyCapture2::ConfigROM, [66](#)
- vendorUniqueInfo\_1
  - FlyCapture2::ConfigROM, [66](#)
- vendorUniqueInfo\_2
  - FlyCapture2::ConfigROM, [67](#)
- vendorUniqueInfo\_3
  - FlyCapture2::ConfigROM, [67](#)
- VideoMode
  - FlyCapture2, [24](#)
- videoMode
  - VideoModes, [142](#)
- VIDEOMODE\_1024x768RGB
  - FlyCapture2, [25](#)
- VIDEOMODE\_1024x768Y16
  - FlyCapture2, [25](#)
- VIDEOMODE\_1024x768Y8
  - FlyCapture2, [25](#)
- VIDEOMODE\_1024x768YUV422
  - FlyCapture2, [25](#)
- VIDEOMODE\_1280x960RGB
  - FlyCapture2, [25](#)
- VIDEOMODE\_1280x960Y16
  - FlyCapture2, [25](#)
- VIDEOMODE\_1280x960Y8
  - FlyCapture2, [25](#)
- VIDEOMODE\_1280x960YUV422
  - FlyCapture2, [25](#)
- VIDEOMODE\_1600x1200RGB
  - FlyCapture2, [25](#)
- VIDEOMODE\_1600x1200Y16
  - FlyCapture2, [25](#)
- VIDEOMODE\_1600x1200Y8
  - FlyCapture2, [25](#)
- VIDEOMODE\_1600x1200YUV422
  - FlyCapture2, [25](#)
- VIDEOMODE\_160x120YUV444
  - FlyCapture2, [25](#)
- VIDEOMODE\_320x240YUV422
  - FlyCapture2, [25](#)
- VIDEOMODE\_640x480RGB
  - FlyCapture2, [25](#)
- VIDEOMODE\_640x480Y16
  - FlyCapture2, [25](#)
- VIDEOMODE\_640x480Y8
  - FlyCapture2, [25](#)
- VIDEOMODE\_640x480YUV411
  - FlyCapture2, [25](#)

- VIDEOMODE\_640x480YUV422
  - FlyCapture2, [25](#)
- VIDEOMODE\_800x600RGB
  - FlyCapture2, [25](#)
- VIDEOMODE\_800x600Y16
  - FlyCapture2, [25](#)
- VIDEOMODE\_800x600Y8
  - FlyCapture2, [25](#)
- VIDEOMODE\_800x600YUV422
  - FlyCapture2, [25](#)
- VIDEOMODE\_FORCE\_32BITS
  - FlyCapture2, [25](#)
- VIDEOMODE\_FORMAT7
  - FlyCapture2, [25](#)
- VideoModes, [142](#)
  - frameRate, [142](#)
  - height, [142](#)
  - videoMode, [142](#)
  - width, [142](#)
- videoModes
  - DCAMFormats, [68](#)
- WaitForBufferEvent
  - FlyCapture2::Camera, [57](#)
- WHITE\_BALANCE
  - FlyCapture2, [24](#)
- whiteBalance
  - FlyCapture2::EmbeddedImageInfo, [70](#)
- width
  - FlyCapture2::Format7ImageSettings, [80](#)
  - VideoModes, [142](#)
- WINDOWS\_X64
  - FlyCapture2, [23](#)
- WINDOWS\_X86
  - FlyCapture2, [23](#)
- WriteRegister
  - FlyCapture2::Camera, [58](#)
  - PGRDirectShow.h, [163](#)
- WriteRegisterBlock
  - FlyCapture2::Camera, [58](#)
- ZOOM
  - FlyCapture2, [24](#)