

FlyCapture2

2.13.3.31

Generated by Doxygen 1.8.13



# Contents

<b>1</b>	<b>Software Licensing Information</b>	<b>1</b>
<b>2</b>	<b>Module Index</b>	<b>3</b>
2.1	Modules . . . . .	3
<b>3</b>	<b>Namespace Index</b>	<b>5</b>
3.1	Namespace List . . . . .	5
<b>4</b>	<b>Hierarchical Index</b>	<b>7</b>
4.1	Class Hierarchy . . . . .	7
<b>5</b>	<b>Class Index</b>	<b>9</b>
5.1	Class List . . . . .	9
<b>6</b>	<b>File Index</b>	<b>13</b>
6.1	File List . . . . .	13
<b>7</b>	<b>Module Documentation</b>	<b>15</b>
7.1	Global constants . . . . .	15
7.1.1	Detailed Description . . . . .	15
7.1.2	Variable Documentation . . . . .	15
7.1.2.1	sk_maxNumPorts . . . . .	15
7.1.2.2	sk_maxStringLength . . . . .	15
7.2	Enumerations . . . . .	16
7.2.1	Detailed Description . . . . .	20
7.2.2	Enumeration Type Documentation . . . . .	20

7.2.2.1	BandwidthAllocation	21
7.2.2.2	BayerTileFormat	21
7.2.2.3	BusCallbackType	21
7.2.2.4	BusSpeed	22
7.2.2.5	ColorProcessingAlgorithm	22
7.2.2.6	DriverType	23
7.2.2.7	ErrorType	23
7.2.2.8	FrameRate	24
7.2.2.9	GrabMode	25
7.2.2.10	GrabTimeout	25
7.2.2.11	ImageFileFormat	26
7.2.2.12	InterfaceType	26
7.2.2.13	Mode	26
7.2.2.14	PCleBusSpeed	27
7.2.2.15	PixelFormat	28
7.2.2.16	PropertyType	28
7.2.2.17	VideoMode	29
7.3	GigE specific enumerations	31
7.3.1	Detailed Description	31
7.3.2	Enumeration Type Documentation	31
7.3.2.1	GigEPropertyType	31
7.4	Structures	32
7.4.1	Detailed Description	33
7.4.2	Typedef Documentation	33
7.4.2.1	TriggerDelay	33
7.4.2.2	TriggerDelayInfo	33
7.5	GigE specific structures	34
7.5.1	Detailed Description	34
7.6	IIDC specific structures	35
7.6.1	Detailed Description	35
7.7	Image saving structures.	36
7.7.1	Detailed Description	36
7.7.2	Typedef Documentation	36
7.7.2.1	CameraEventCallback	36
7.8	Video saving structures.	37
7.8.1	Detailed Description	37

<b>8</b>	<b>Namespace Documentation</b>	<b>39</b>
8.1	FlyCap3CameraControl Namespace Reference	39
8.2	FlyCapture2 Namespace Reference	39
8.2.1	Typedef Documentation	47
8.2.1.1	AsyncCommandCallback	47
8.2.1.2	BusEventCallback	47
8.2.1.3	CallbackHandle	47
8.2.1.4	ImageEventCallback	48
8.2.2	Enumeration Type Documentation	48
8.2.2.1	ByteOrder	48
8.2.2.2	OSType	48
8.3	MultiSyncLibrary Namespace Reference	48
8.3.1	Enumeration Type Documentation	49
8.3.1.1	PGRSyncError	49
8.3.1.2	PGRSyncMessage	49
<b>9</b>	<b>Class Documentation</b>	<b>51</b>
9.1	AVIOption Struct Reference	51
9.1.1	Detailed Description	51
9.1.2	Constructor & Destructor Documentation	51
9.1.2.1	AVIOption()	51
9.1.3	Member Data Documentation	52
9.1.3.1	frameRate	52
9.1.3.2	reserved	52
9.2	BMPOption Struct Reference	52
9.2.1	Detailed Description	52
9.2.2	Constructor & Destructor Documentation	52
9.2.2.1	BMPOption()	53
9.2.3	Member Data Documentation	53
9.2.3.1	indexedColor_8bit	53
9.2.3.2	reserved	53

9.3	BusManager Class Reference	53
9.3.1	Detailed Description	54
9.3.2	Constructor & Destructor Documentation	55
9.3.2.1	BusManager()	55
9.3.2.2	~BusManager()	55
9.3.3	Member Function Documentation	55
9.3.3.1	DiscoverGigECameras()	55
9.3.3.2	FireBusReset()	56
9.3.3.3	ForceAllIPAddressesAutomatically() [1/2]	56
9.3.3.4	ForceAllIPAddressesAutomatically() [2/2]	56
9.3.3.5	ForceIPAddressToCamera()	57
9.3.3.6	GetCameraFromIndex()	57
9.3.3.7	GetCameraFromIPAddress()	58
9.3.3.8	GetCameraFromSerialNumber()	58
9.3.3.9	GetCameraSerialNumberFromIndex()	58
9.3.3.10	GetDeviceFromIndex()	59
9.3.3.11	GetInterfaceTypeFromGuid()	59
9.3.3.12	GetNumOfCameras()	60
9.3.3.13	GetNumOfDevices()	60
9.3.3.14	GetTopology()	60
9.3.3.15	GetUsbLinkInfo()	61
9.3.3.16	GetUsbPortStatus()	61
9.3.3.17	IsCameraControlable()	63
9.3.3.18	ReadPhyRegister()	63
9.3.3.19	RegisterCallback()	64
9.3.3.20	RescanBus()	64
9.3.3.21	UnregisterCallback()	64
9.3.3.22	WritePhyRegister()	65
9.4	Camera Class Reference	65
9.4.1	Detailed Description	69

9.4.2	Constructor & Destructor Documentation	69
9.4.2.1	Camera()	70
9.4.2.2	~Camera()	70
9.4.3	Member Function Documentation	70
9.4.3.1	Connect()	70
9.4.3.2	DeregisterAllEvents()	70
9.4.3.3	DeregisterEvent()	70
9.4.3.4	Disconnect()	71
9.4.3.5	EnableLUT()	71
9.4.3.6	FireSoftwareTrigger()	71
9.4.3.7	GetActiveLUTBank()	72
9.4.3.8	GetCameraInfo()	72
9.4.3.9	GetConfiguration()	73
9.4.3.10	GetCycleTime()	73
9.4.3.11	GetEmbeddedImageInfo()	73
9.4.3.12	GetFormat7Configuration()	74
9.4.3.13	GetFormat7Info()	75
9.4.3.14	GetGPIOPinDirection()	75
9.4.3.15	GetLUTBankInfo()	76
9.4.3.16	GetLUTChannel()	76
9.4.3.17	GetLUTInfo()	77
9.4.3.18	GetMemoryChannel()	77
9.4.3.19	GetMemoryChannelInfo()	78
9.4.3.20	GetProperty()	78
9.4.3.21	GetPropertyInfo()	79
9.4.3.22	GetRegisterString()	79
9.4.3.23	GetStats()	80
9.4.3.24	GetStrobe()	80
9.4.3.25	GetStrobeInfo()	80
9.4.3.26	GetTriggerDelay()	81

9.4.3.27	GetTriggerDelayInfo()	81
9.4.3.28	GetTriggerMode()	82
9.4.3.29	GetTriggerModeInfo()	82
9.4.3.30	GetVideoModeAndFrameRate()	83
9.4.3.31	GetVideoModeAndFrameRateInfo()	84
9.4.3.32	IsConnected()	84
9.4.3.33	ReadRegister()	84
9.4.3.34	ReadRegisterBlock()	85
9.4.3.35	RegisterAllEvents()	85
9.4.3.36	RegisterEvent()	86
9.4.3.37	ResetStats()	86
9.4.3.38	RestoreFromMemoryChannel()	86
9.4.3.39	RetrieveBuffer()	86
9.4.3.40	SaveToMemoryChannel()	87
9.4.3.41	SetActiveLUTBank()	87
9.4.3.42	SetCallback()	88
9.4.3.43	SetConfiguration()	88
9.4.3.44	SetEmbeddedImageInfo()	89
9.4.3.45	SetFormat7Configuration() [1/2]	89
9.4.3.46	SetFormat7Configuration() [2/2]	90
9.4.3.47	SetGPIOPinDirection()	90
9.4.3.48	SetLUTChannel()	91
9.4.3.49	SetProperty()	92
9.4.3.50	SetStrobe()	92
9.4.3.51	SetTriggerDelay()	93
9.4.3.52	SetTriggerMode()	93
9.4.3.53	SetUserBuffers()	94
9.4.3.54	SetVideoModeAndFrameRate()	95
9.4.3.55	StartCapture()	95
9.4.3.56	StartSyncCapture()	96



9.4.3.57	StopCapture()	96
9.4.3.58	ValidateFormat7Settings()	96
9.4.3.59	WaitForBufferEvent()	97
9.4.3.60	WriteRegister()	97
9.4.3.61	WriteRegisterBlock()	98
9.5	CameraBase Class Reference	99
9.5.1	Detailed Description	102
9.5.2	Constructor & Destructor Documentation	102
9.5.2.1	CameraBase()	102
9.5.2.2	~CameraBase()	103
9.5.3	Member Function Documentation	103
9.5.3.1	Connect()	103
9.5.3.2	DeregisterAllEvents()	103
9.5.3.3	DeregisterEvent()	104
9.5.3.4	Disconnect()	104
9.5.3.5	EnableLUT()	104
9.5.3.6	FireSoftwareTrigger()	105
9.5.3.7	GetActiveLUTBank()	105
9.5.3.8	GetCameraInfo()	106
9.5.3.9	GetConfiguration()	106
9.5.3.10	GetCycleTime()	106
9.5.3.11	GetEmbeddedImageInfo()	107
9.5.3.12	GetGPIOPinDirection()	107
9.5.3.13	GetLUTBankInfo()	108
9.5.3.14	GetLUTChannel()	108
9.5.3.15	GetLUTInfo()	109
9.5.3.16	GetMemoryChannel()	109
9.5.3.17	GetMemoryChannelInfo()	110
9.5.3.18	GetProperty()	110
9.5.3.19	GetPropertyInfo()	111

9.5.3.20	<a href="#">GetRegisterString()</a>	111
9.5.3.21	<a href="#">GetStats()</a>	112
9.5.3.22	<a href="#">GetStrobe()</a>	112
9.5.3.23	<a href="#">GetStrobeInfo()</a>	112
9.5.3.24	<a href="#">GetTriggerDelay()</a>	113
9.5.3.25	<a href="#">GetTriggerDelayInfo()</a>	113
9.5.3.26	<a href="#">GetTriggerMode()</a>	114
9.5.3.27	<a href="#">GetTriggerModeInfo()</a>	114
9.5.3.28	<a href="#">IsConnected()</a>	115
9.5.3.29	<a href="#">ReadRegister()</a>	115
9.5.3.30	<a href="#">ReadRegisterBlock()</a>	116
9.5.3.31	<a href="#">RegisterAllEvents()</a>	116
9.5.3.32	<a href="#">RegisterEvent()</a>	117
9.5.3.33	<a href="#">ResetStats()</a>	117
9.5.3.34	<a href="#">RestoreFromMemoryChannel()</a>	117
9.5.3.35	<a href="#">RetrieveBuffer()</a>	117
9.5.3.36	<a href="#">SaveToMemoryChannel()</a>	118
9.5.3.37	<a href="#">SetActiveLUTBank()</a>	118
9.5.3.38	<a href="#">SetCallback()</a>	119
9.5.3.39	<a href="#">SetConfiguration()</a>	119
9.5.3.40	<a href="#">SetEmbeddedImageInfo()</a>	120
9.5.3.41	<a href="#">SetGPIOPinDirection()</a>	120
9.5.3.42	<a href="#">SetLUTChannel()</a>	121
9.5.3.43	<a href="#">SetProperty()</a>	122
9.5.3.44	<a href="#">SetStrobe()</a>	122
9.5.3.45	<a href="#">SetTriggerDelay()</a>	123
9.5.3.46	<a href="#">SetTriggerMode()</a>	123
9.5.3.47	<a href="#">SetUserBuffers()</a>	124
9.5.3.48	<a href="#">StartCapture()</a>	125
9.5.3.49	<a href="#">StartSyncCapture()</a>	125

9.5.3.50	StopCapture()	126
9.5.3.51	WaitForBufferEvent()	126
9.5.3.52	WriteRegister()	127
9.5.3.53	WriteRegisterBlock()	128
9.5.4	Member Data Documentation	128
9.5.4.1	m_pCameraData	128
9.6	CameraControlDlg Class Reference	128
9.6.1	Detailed Description	129
9.6.2	Constructor & Destructor Documentation	129
9.6.2.1	CameraControlDlg()	129
9.6.2.2	~CameraControlDlg()	129
9.6.3	Member Function Documentation	130
9.6.3.1	Connect()	130
9.6.3.2	Disconnect()	130
9.6.3.3	Hide()	130
9.6.3.4	IsVisible()	130
9.6.3.5	SetTitle()	130
9.6.3.6	Show() [1/2]	131
9.6.3.7	Show() [2/2]	131
9.6.3.8	ShowModal() [1/2]	131
9.6.3.9	ShowModal() [2/2]	131
9.7	CameraInfo Struct Reference	132
9.7.1	Detailed Description	133
9.7.2	Constructor & Destructor Documentation	134
9.7.2.1	CameraInfo()	134
9.7.3	Member Data Documentation	134
9.7.3.1	applicationIPAddress	134
9.7.3.2	applicationPort	134
9.7.3.3	bayerTileFormat	134
9.7.3.4	busNumber	134

9.7.3.5	ccpStatus	135
9.7.3.6	configROM	135
9.7.3.7	defaultGateway	135
9.7.3.8	driverName	135
9.7.3.9	driverType	135
9.7.3.10	firmwareBuildTime	135
9.7.3.11	firmwareVersion	136
9.7.3.12	gigEMajorVersion	136
9.7.3.13	gigEMinorVersion	136
9.7.3.14	iidcVer	136
9.7.3.15	interfaceType	136
9.7.3.16	ipAddress	136
9.7.3.17	isColorCamera	137
9.7.3.18	macAddress	137
9.7.3.19	maximumBusSpeed	137
9.7.3.20	modelName	137
9.7.3.21	nodeNumber	137
9.7.3.22	pcieBusSpeed	137
9.7.3.23	reserved	138
9.7.3.24	sensorInfo	138
9.7.3.25	sensorResolution	138
9.7.3.26	serialNumber	138
9.7.3.27	subnetMask	138
9.7.3.28	userDefinedName	138
9.7.3.29	vendorName	139
9.7.3.30	xmlURL1	139
9.7.3.31	xmlURL2	139
9.8	CameraSelectionDlg Class Reference	139
9.8.1	Detailed Description	139
9.8.2	Constructor & Destructor Documentation	140

9.8.2.1	CameraSelectionDlg()	140
9.8.2.2	~CameraSelectionDlg()	140
9.8.3	Member Function Documentation	140
9.8.3.1	SetTitle()	140
9.8.3.2	ShowModal()	140
9.9	CameraStats Struct Reference	141
9.9.1	Detailed Description	142
9.9.2	Constructor & Destructor Documentation	142
9.9.2.1	CameraStats()	142
9.9.3	Member Data Documentation	142
9.9.3.1	cameraCurrents	142
9.9.3.2	cameraPowerUp	142
9.9.3.3	cameraVoltages	142
9.9.3.4	imageCorrupt	142
9.9.3.5	imageDriverDropped	142
9.9.3.6	imageDropped	143
9.9.3.7	imageXmitFailed	143
9.9.3.8	numCurrents	143
9.9.3.9	numResendPacketsReceived	143
9.9.3.10	numResendPacketsRequested	143
9.9.3.11	numVoltages	143
9.9.3.12	portErrors	143
9.9.3.13	regReadFailed	144
9.9.3.14	regWriteFailed	144
9.9.3.15	reserved	144
9.9.3.16	temperature	144
9.9.3.17	timeSinceBusReset	144
9.9.3.18	timeSinceInitialization	144
9.9.3.19	timeStamp	144
9.10	ConfigROM Struct Reference	145

9.10.1 Detailed Description . . . . .	145
9.10.2 Constructor & Destructor Documentation . . . . .	145
9.10.2.1 ConfigROM() . . . . .	145
9.10.3 Member Data Documentation . . . . .	146
9.10.3.1 chipIdHi . . . . .	146
9.10.3.2 chipIdLo . . . . .	146
9.10.3.3 nodeVendorId . . . . .	146
9.10.3.4 pszKeyword . . . . .	146
9.10.3.5 reserved . . . . .	146
9.10.3.6 unitSpecId . . . . .	147
9.10.3.7 unitSubSWVer . . . . .	147
9.10.3.8 unitSWVer . . . . .	147
9.10.3.9 vendorUniqueInfo_0 . . . . .	147
9.10.3.10 vendorUniqueInfo_1 . . . . .	147
9.10.3.11 vendorUniqueInfo_2 . . . . .	147
9.10.3.12 vendorUniqueInfo_3 . . . . .	148
9.11 EmbeddedImageInfo Struct Reference . . . . .	148
9.11.1 Detailed Description . . . . .	149
9.11.2 Member Data Documentation . . . . .	149
9.11.2.1 brightness . . . . .	149
9.11.2.2 exposure . . . . .	149
9.11.2.3 frameCounter . . . . .	149
9.11.2.4 gain . . . . .	149
9.11.2.5 GPIOPinState . . . . .	149
9.11.2.6 ROIPosition . . . . .	149
9.11.2.7 shutter . . . . .	150
9.11.2.8 strobePattern . . . . .	150
9.11.2.9 timestamp . . . . .	150
9.11.2.10 whiteBalance . . . . .	150
9.12 EmbeddedImageInfoProperty Struct Reference . . . . .	150

9.12.1 Detailed Description . . . . .	150
9.12.2 Constructor & Destructor Documentation . . . . .	151
9.12.2.1 EmbeddedImageInfoProperty() . . . . .	151
9.12.3 Member Data Documentation . . . . .	151
9.12.3.1 available . . . . .	151
9.12.3.2 onOff . . . . .	151
9.13 Error Class Reference . . . . .	151
9.13.1 Detailed Description . . . . .	152
9.13.2 Constructor & Destructor Documentation . . . . .	152
9.13.2.1 Error() [1/2] . . . . .	153
9.13.2.2 Error() [2/2] . . . . .	153
9.13.2.3 ~Error() . . . . .	153
9.13.3 Member Function Documentation . . . . .	153
9.13.3.1 CollectSupportInformation() . . . . .	153
9.13.3.2 GetBuildDate() . . . . .	153
9.13.3.3 GetCause() . . . . .	154
9.13.3.4 GetDescription() . . . . .	154
9.13.3.5 GetFilename() . . . . .	154
9.13.3.6 GetLine() . . . . .	154
9.13.3.7 GetType() . . . . .	155
9.13.3.8 operator!=( ) [1/2] . . . . .	155
9.13.3.9 operator!=( ) [2/2] . . . . .	155
9.13.3.10 operator=( ) . . . . .	155
9.13.3.11 operator==( ) [1/2] . . . . .	155
9.13.3.12 operator==( ) [2/2] . . . . .	156
9.13.3.13 PrintErrorTrace() . . . . .	156
9.13.4 Friends And Related Function Documentation . . . . .	156
9.13.4.1 InternalError . . . . .	156
9.14 EventCallbackData Struct Reference . . . . .	156
9.14.1 Member Data Documentation . . . . .	157

9.14.1.1	EventData	157
9.14.1.2	EventDataSize	157
9.14.1.3	EventID	157
9.14.1.4	EventName	157
9.14.1.5	EventTimestamp	157
9.14.1.6	EventUserData	158
9.14.1.7	EventUserDataSize	158
9.15	EventOptions Struct Reference	158
9.15.1	Detailed Description	158
9.15.2	Member Data Documentation	158
9.15.2.1	EventCallbackFcn	159
9.15.2.2	EventName	159
9.15.2.3	EventUserData	159
9.15.2.4	EventUserDataSize	159
9.16	FC2Config Struct Reference	159
9.16.1	Detailed Description	160
9.16.2	Constructor & Destructor Documentation	160
9.16.2.1	FC2Config()	160
9.16.3	Member Data Documentation	160
9.16.3.1	asyncBusSpeed	161
9.16.3.2	bandwidthAllocation	161
9.16.3.3	grabMode	161
9.16.3.4	grabTimeout	161
9.16.3.5	highPerformanceRetrieveBuffer	161
9.16.3.6	isochBusSpeed	162
9.16.3.7	minNumImageNotifications	162
9.16.3.8	numBuffers	162
9.16.3.9	numImageNotifications	162
9.16.3.10	registerTimeout	163
9.16.3.11	registerTimeoutRetries	163



9.16.3.12 reserved	163
9.17 FC2Version Struct Reference	163
9.17.1 Detailed Description	163
9.17.2 Member Data Documentation	164
9.17.2.1 build	164
9.17.2.2 major	164
9.17.2.3 minor	164
9.17.2.4 type	164
9.18 FlyCapture2Video Class Reference	164
9.18.1 Detailed Description	165
9.18.2 Constructor & Destructor Documentation	165
9.18.2.1 FlyCapture2Video()	165
9.18.2.2 ~FlyCapture2Video()	165
9.18.3 Member Function Documentation	165
9.18.3.1 Append()	165
9.18.3.2 Close()	166
9.18.3.3 Open() [1/3]	166
9.18.3.4 Open() [2/3]	167
9.18.3.5 Open() [3/3]	167
9.18.3.6 SetMaximumFileSize()	168
9.19 FlyCapture3ApiGuiWrapper Class Reference	168
9.19.1 Constructor & Destructor Documentation	169
9.19.1.1 FlyCapture3ApiGuiWrapper()	169
9.19.1.2 ~FlyCapture3ApiGuiWrapper()	169
9.19.2 Member Function Documentation	169
9.19.2.1 ConnectGUILibrary()	169
9.19.2.2 DisconnectGUILibrary()	169
9.19.2.3 GetControlNameList()	169
9.19.2.4 GetDialogNameList()	169
9.19.2.5 GetNumDialogs()	170

9.19.2.6	GetNumOfControls()	170
9.19.2.7	ShowCameraSelectionDialog()	170
9.19.2.8	ShowDialogByIndex()	170
9.19.2.9	ShowDialogByName()	170
9.19.2.10	ShowPropertyGridDialog()	170
9.20	Format7ImageSettings Struct Reference	170
9.20.1	Detailed Description	171
9.20.2	Constructor & Destructor Documentation	171
9.20.2.1	Format7ImageSettings()	171
9.20.3	Member Data Documentation	171
9.20.3.1	height	171
9.20.3.2	mode	172
9.20.3.3	offsetX	172
9.20.3.4	offsetY	172
9.20.3.5	pixelFormat	172
9.20.3.6	reserved	172
9.20.3.7	width	172
9.21	Format7Info Struct Reference	173
9.21.1	Detailed Description	173
9.21.2	Constructor & Destructor Documentation	173
9.21.2.1	Format7Info()	174
9.21.3	Member Data Documentation	174
9.21.3.1	imageHStepSize	174
9.21.3.2	imageVStepSize	174
9.21.3.3	maxHeight	174
9.21.3.4	maxPacketSize	174
9.21.3.5	maxWidth	174
9.21.3.6	minPacketSize	175
9.21.3.7	mode	175
9.21.3.8	offsetHStepSize	175

9.21.3.9	offsetVStepSize	175
9.21.3.10	packetSize	175
9.21.3.11	percentage	175
9.21.3.12	pixelFormatBitField	176
9.21.3.13	reserved	176
9.21.3.14	vendorPixelFormatBitField	176
9.22	Format7PacketInfo Struct Reference	176
9.22.1	Detailed Description	177
9.22.2	Constructor & Destructor Documentation	177
9.22.2.1	Format7PacketInfo()	177
9.22.3	Member Data Documentation	177
9.22.3.1	maxBytesPerPacket	177
9.22.3.2	recommendedBytesPerPacket	177
9.22.3.3	reserved	177
9.22.3.4	unitBytesPerPacket	177
9.23	GCCamera Class Reference	178
9.23.1	Constructor & Destructor Documentation	181
9.23.1.1	GCCamera()	181
9.23.1.2	~GCCamera()	181
9.23.2	Member Function Documentation	181
9.23.2.1	Connect() [1/2]	181
9.23.2.2	Connect() [2/2]	181
9.23.2.3	Disconnect()	182
9.23.2.4	EnableLUT()	182
9.23.2.5	FireSoftwareTrigger()	182
9.23.2.6	GCCamera::GetXML()	183
9.23.2.7	GetActiveLUTBank()	183
9.23.2.8	GetCameraInfo()	183
9.23.2.9	GetConfiguration()	184
9.23.2.10	GetCycleTime()	184

9.23.2.11 GetEmbeddedImageInfo()	185
9.23.2.12 GetGPIOPinDirection()	185
9.23.2.13 GetInterfaceType()	186
9.23.2.14 GetLUTBankInfo()	186
9.23.2.15 GetLUTChannel()	186
9.23.2.16 GetLUTInfo()	187
9.23.2.17 GetMemoryChannel()	188
9.23.2.18 GetMemoryChannelInfo()	188
9.23.2.19 GetNodeMap()	189
9.23.2.20 GetProperty()	189
9.23.2.21 GetPropertyInfo()	189
9.23.2.22 GetRegisterString()	190
9.23.2.23 GetStats()	190
9.23.2.24 GetStrobe()	190
9.23.2.25 GetStrobeInfo()	191
9.23.2.26 GetTriggerDelay()	191
9.23.2.27 GetTriggerDelayInfo()	192
9.23.2.28 GetTriggerMode()	192
9.23.2.29 GetTriggerModelInfo()	193
9.23.2.30 IsConnected()	193
9.23.2.31 ReadGVCPMemory()	194
9.23.2.32 ReadGVCPRegister()	194
9.23.2.33 ReadGVCPRegisterBlock()	194
9.23.2.34 ReadRegister()	194
9.23.2.35 ReadRegisterBlock()	195
9.23.2.36 ResetStats()	195
9.23.2.37 RestoreFromMemoryChannel()	195
9.23.2.38 RetrieveBuffer()	196
9.23.2.39 SaveToMemoryChannel()	196
9.23.2.40 SetActiveLUTBank()	197

9.23.2.41 SetCallback()	197
9.23.2.42 SetCamera() [1/2]	198
9.23.2.43 SetCamera() [2/2]	198
9.23.2.44 SetConfiguration()	198
9.23.2.45 SetEmbeddedImageInfo()	199
9.23.2.46 SetGPIOPinDirection()	199
9.23.2.47 SetLUTChannel()	200
9.23.2.48 SetProperty()	200
9.23.2.49 SetStrobe()	201
9.23.2.50 SetTriggerDelay()	201
9.23.2.51 SetTriggerMode()	202
9.23.2.52 SetUserBuffers()	203
9.23.2.53 StartCapture()	203
9.23.2.54 StartSyncCapture()	204
9.23.2.55 StopCapture()	204
9.23.2.56 TestGainNode()	205
9.23.2.57 WaitForBufferEvent()	205
9.23.2.58 WriteGVCPMemory()	205
9.23.2.59 WriteGVCPRegister()	205
9.23.2.60 WriteGVCPRegisterBlock()	206
9.23.2.61 WriteRegister()	206
9.23.2.62 WriteRegisterBlock()	206
9.23.3 Member Data Documentation	207
9.23.3.1 m_busMgr	207
9.24 GigECamera Class Reference	207
9.24.1 Detailed Description	211
9.24.2 Constructor & Destructor Documentation	212
9.24.2.1 GigECamera()	212
9.24.2.2 ~GigECamera()	212
9.24.3 Member Function Documentation	212

9.24.3.1	Connect()	212
9.24.3.2	DeregisterAllEvents()	212
9.24.3.3	DeregisterEvent()	213
9.24.3.4	Disconnect()	213
9.24.3.5	DiscoverGigEPacketSize()	213
9.24.3.6	EnableLUT()	214
9.24.3.7	FireSoftwareTrigger()	214
9.24.3.8	GetActiveLUTBank()	214
9.24.3.9	GetCameraInfo()	215
9.24.3.10	GetConfiguration()	215
9.24.3.11	GetCycleTime()	216
9.24.3.12	GetEmbeddedImageInfo()	216
9.24.3.13	GetGigEConfig()	217
9.24.3.14	GetGigEImageBinningSettings()	217
9.24.3.15	GetGigEImageSettings()	217
9.24.3.16	GetGigEImageSettingsInfo()	218
9.24.3.17	GetGigEImagingMode()	218
9.24.3.18	GetGigEProperty()	218
9.24.3.19	GetGigEStreamChannelInfo()	219
9.24.3.20	GetGPIOPinDirection()	219
9.24.3.21	GetLUTBankInfo()	220
9.24.3.22	GetLUTChannel()	220
9.24.3.23	GetLUTInfo()	221
9.24.3.24	GetMemoryChannel()	221
9.24.3.25	GetMemoryChannelInfo()	222
9.24.3.26	GetNumStreamChannels()	222
9.24.3.27	GetProperty()	223
9.24.3.28	GetPropertyInfo()	223
9.24.3.29	GetRegisterString()	224
9.24.3.30	GetStats()	224

9.24.3.31 GetStrobe()	224
9.24.3.32 GetStrobeInfo()	225
9.24.3.33 GetTriggerDelay()	225
9.24.3.34 GetTriggerDelayInfo()	226
9.24.3.35 GetTriggerMode()	226
9.24.3.36 GetTriggerModeInfo()	227
9.24.3.37 IsConnected()	228
9.24.3.38 QueryGigEImagingMode()	228
9.24.3.39 ReadGVCPMemory()	228
9.24.3.40 ReadGVCPRegister()	229
9.24.3.41 ReadGVCPRegisterBlock()	229
9.24.3.42 ReadRegister()	229
9.24.3.43 ReadRegisterBlock()	230
9.24.3.44 RegisterAllEvents()	231
9.24.3.45 RegisterEvent()	231
9.24.3.46 ResetStats()	231
9.24.3.47 RestoreFromMemoryChannel()	231
9.24.3.48 RetrieveBuffer()	232
9.24.3.49 SaveToMemoryChannel()	232
9.24.3.50 SetActiveLUTBank()	233
9.24.3.51 SetCallback()	233
9.24.3.52 SetConfiguration()	234
9.24.3.53 SetEmbeddedImageInfo()	234
9.24.3.54 SetGigEConfig()	234
9.24.3.55 SetGigEImageBinningSettings()	235
9.24.3.56 SetGigEImageSettings()	235
9.24.3.57 SetGigEImagingMode()	236
9.24.3.58 SetGigEProperty()	236
9.24.3.59 SetGigEStreamChannelInfo()	236
9.24.3.60 SetGPIOPinDirection()	237

9.24.3.61 SetLUTChannel()	237
9.24.3.62 SetProperty()	238
9.24.3.63 SetStrobe()	239
9.24.3.64 SetTriggerDelay()	239
9.24.3.65 SetTriggerMode()	240
9.24.3.66 SetUserBuffers()	240
9.24.3.67 StartCapture()	241
9.24.3.68 StartSyncCapture()	242
9.24.3.69 StopCapture()	242
9.24.3.70 WaitForBufferEvent()	242
9.24.3.71 WriteGVCPMemory()	243
9.24.3.72 WriteGVCPRegister()	243
9.24.3.73 WriteGVCPRegisterBlock()	244
9.24.3.74 WriteRegister()	244
9.24.3.75 WriteRegisterBlock()	245
9.25 GigEConfig Struct Reference	245
9.25.1 Detailed Description	246
9.25.2 Constructor & Destructor Documentation	246
9.25.2.1 GigEConfig()	246
9.25.3 Member Data Documentation	246
9.25.3.1 enablePacketResend	246
9.25.3.2 registerTimeout	246
9.25.3.3 registerTimeoutRetries	247
9.26 GigEImageSettings Struct Reference	247
9.26.1 Detailed Description	247
9.26.2 Constructor & Destructor Documentation	247
9.26.2.1 GigEImageSettings()	247
9.26.3 Member Data Documentation	248
9.26.3.1 height	248
9.26.3.2 offsetX	248



9.26.3.3	offsetY	248
9.26.3.4	pixelFormat	248
9.26.3.5	reserved	248
9.26.3.6	width	249
9.27	GigEImageSettingsInfo Struct Reference	249
9.27.1	Detailed Description	249
9.27.2	Constructor & Destructor Documentation	249
9.27.2.1	GigEImageSettingsInfo()	250
9.27.3	Member Data Documentation	250
9.27.3.1	imageHStepSize	250
9.27.3.2	imageVStepSize	250
9.27.3.3	maxHeight	250
9.27.3.4	maxWidth	250
9.27.3.5	offsetHStepSize	250
9.27.3.6	offsetVStepSize	251
9.27.3.7	pixelFormatBitField	251
9.27.3.8	reserved	251
9.27.3.9	vendorPixelFormatBitField	251
9.28	GigEProperty Struct Reference	251
9.28.1	Detailed Description	252
9.28.2	Member Data Documentation	252
9.28.2.1	isReadable	252
9.28.2.2	isWritable	252
9.28.2.3	max	252
9.28.2.4	min	252
9.28.2.5	propType	252
9.28.2.6	value	253
9.29	GigEStreamChannel Struct Reference	253
9.29.1	Detailed Description	254
9.29.2	Constructor & Destructor Documentation	254

9.29.2.1	GigEStreamChannel()	254
9.29.3	Member Data Documentation	254
9.29.3.1	destinationIpAddress	254
9.29.3.2	doNotFragment	254
9.29.3.3	hostPort	254
9.29.3.4	interPacketDelay	254
9.29.3.5	networkInterfaceIndex	255
9.29.3.6	packetSize	255
9.29.3.7	sourcePort	255
9.30	H264Option Struct Reference	255
9.30.1	Detailed Description	256
9.30.2	Constructor & Destructor Documentation	256
9.30.2.1	H264Option()	256
9.30.3	Member Data Documentation	256
9.30.3.1	bitrate	256
9.30.3.2	frameRate	256
9.30.3.3	height	256
9.30.3.4	reserved	256
9.30.3.5	width	257
9.31	Image Class Reference	257
9.31.1	Detailed Description	259
9.31.2	Constructor & Destructor Documentation	259
9.31.2.1	Image() [1/6]	259
9.31.2.2	Image() [2/6]	259
9.31.2.3	Image() [3/6]	260
9.31.2.4	Image() [4/6]	260
9.31.2.5	Image() [5/6]	261
9.31.2.6	Image() [6/6]	261
9.31.2.7	~Image()	261
9.31.3	Member Function Documentation	261

9.31.3.1 CalculateStatistics()	262
9.31.3.2 Convert() [1/2]	262
9.31.3.3 Convert() [2/2]	262
9.31.3.4 DeepCopy()	263
9.31.3.5 DetermineBitsPerPixel()	263
9.31.3.6 GetBayerTileFormat()	264
9.31.3.7 GetBitsPerPixel()	264
9.31.3.8 GetBlockId()	264
9.31.3.9 GetColorProcessing()	264
9.31.3.10 GetCols()	265
9.31.3.11 GetData() [1/2]	265
9.31.3.12 GetData() [2/2]	265
9.31.3.13 GetDataSize()	265
9.31.3.14 GetDefaultColorProcessing()	266
9.31.3.15 GetDefaultOutputFormat()	266
9.31.3.16 GetDimensions()	266
9.31.3.17 GetMetadata()	267
9.31.3.18 GetPixelFormat()	267
9.31.3.19 GetReceivedDataSize()	267
9.31.3.20 GetRows()	268
9.31.3.21 GetStride()	268
9.31.3.22 GetTimeStamp()	268
9.31.3.23 operator()()	268
9.31.3.24 operator=()	269
9.31.3.25 operator[]()	269
9.31.3.26 ReleaseBuffer()	269
9.31.3.27 Save() [1/8]	270
9.31.3.28 Save() [2/8]	270
9.31.3.29 Save() [3/8]	270
9.31.3.30 Save() [4/8]	271

9.31.3.31 Save() [5/8]	271
9.31.3.32 Save() [6/8]	271
9.31.3.33 Save() [7/8]	272
9.31.3.34 Save() [8/8]	272
9.31.3.35 SetBlockId()	273
9.31.3.36 SetColorProcessing()	273
9.31.3.37 SetData()	273
9.31.3.38 SetDefaultColorProcessing()	274
9.31.3.39 SetDefaultOutputFormat()	274
9.31.3.40 SetDimensions()	275
9.31.4 Friends And Related Function Documentation	275
9.31.4.1 Iso	275
9.32 ImageMetadata Struct Reference	276
9.32.1 Detailed Description	276
9.32.2 Constructor & Destructor Documentation	276
9.32.2.1 ImageMetadata()	276
9.32.3 Member Data Documentation	277
9.32.3.1 embeddedBrightness	277
9.32.3.2 embeddedExposure	277
9.32.3.3 embeddedFrameCounter	277
9.32.3.4 embeddedGain	277
9.32.3.5 embeddedGPIOPinState	277
9.32.3.6 embeddedROIPosition	278
9.32.3.7 embeddedShutter	278
9.32.3.8 embeddedStrobePattern	278
9.32.3.9 embeddedTimeStamp	278
9.32.3.10 embeddedWhiteBalance	278
9.32.3.11 reserved	278
9.33 ImageStatistics Class Reference	279
9.33.1 Detailed Description	280

9.33.2	Member Enumeration Documentation	280
9.33.2.1	StatisticsChannel	280
9.33.3	Constructor & Destructor Documentation	280
9.33.3.1	ImageStatistics() [1/2]	280
9.33.3.2	~ImageStatistics()	281
9.33.3.3	ImageStatistics() [2/2]	281
9.33.4	Member Function Documentation	281
9.33.4.1	DisableAll()	281
9.33.4.2	EnableAll()	281
9.33.4.3	EnableGreyOnly()	282
9.33.4.4	EnableHSLOnly()	282
9.33.4.5	EnableRGBOnly()	282
9.33.4.6	GetChannelStatus()	282
9.33.4.7	GetHistogram()	283
9.33.4.8	GetMean()	283
9.33.4.9	GetNumPixelValues()	284
9.33.4.10	GetPixelValueRange()	284
9.33.4.11	GetRange()	284
9.33.4.12	GetStatistics()	285
9.33.4.13	operator=()	285
9.33.4.14	SetChannelStatus()	286
9.33.5	Friends And Related Function Documentation	286
9.33.5.1	ImageStatsCalculator	286
9.34	Internal Class Reference	286
9.34.1	Member Function Documentation	287
9.34.1.1	GetInternal()	287
9.35	IPAddress Struct Reference	287
9.35.1	Detailed Description	287
9.35.2	Constructor & Destructor Documentation	287
9.35.2.1	IPAddress() [1/2]	287

9.35.2.2	IPAddress() [2/2]	288
9.35.3	Member Function Documentation	288
9.35.3.1	operator!=(())	288
9.35.3.2	operator==(())	288
9.35.4	Member Data Documentation	288
9.35.4.1	octets	288
9.36	JPEGOption Struct Reference	288
9.36.1	Detailed Description	289
9.36.2	Constructor & Destructor Documentation	289
9.36.2.1	JPEGOption()	289
9.36.3	Member Data Documentation	289
9.36.3.1	progressive	289
9.36.3.2	quality	289
9.36.3.3	reserved	290
9.37	JPG2Option Struct Reference	290
9.37.1	Detailed Description	290
9.37.2	Constructor & Destructor Documentation	290
9.37.2.1	JPG2Option()	290
9.37.3	Member Data Documentation	290
9.37.3.1	quality	291
9.37.3.2	reserved	291
9.38	LUTData Struct Reference	291
9.38.1	Detailed Description	292
9.38.2	Constructor & Destructor Documentation	292
9.38.2.1	LUTData()	292
9.38.3	Member Data Documentation	292
9.38.3.1	enabled	292
9.38.3.2	inputBitDepth	292
9.38.3.3	numBanks	292
9.38.3.4	numChannels	292

9.38.3.5	numEntries . . . . .	293
9.38.3.6	outputBitDepth . . . . .	293
9.38.3.7	reserved . . . . .	293
9.38.3.8	supported . . . . .	293
9.39	MACAddress Struct Reference . . . . .	293
9.39.1	Detailed Description . . . . .	294
9.39.2	Constructor & Destructor Documentation . . . . .	294
9.39.2.1	MACAddress() [1/2] . . . . .	294
9.39.2.2	MACAddress() [2/2] . . . . .	294
9.39.3	Member Function Documentation . . . . .	294
9.39.3.1	operator!=(()) . . . . .	294
9.39.3.2	operator==(()) . . . . .	294
9.39.4	Member Data Documentation . . . . .	295
9.39.4.1	octets . . . . .	295
9.40	MJPGOption Struct Reference . . . . .	295
9.40.1	Detailed Description . . . . .	295
9.40.2	Constructor & Destructor Documentation . . . . .	295
9.40.2.1	MJPGOption() . . . . .	295
9.40.3	Member Data Documentation . . . . .	295
9.40.3.1	frameRate . . . . .	296
9.40.3.2	quality . . . . .	296
9.40.3.3	reserved . . . . .	296
9.41	NodeMap Class Reference . . . . .	296
9.41.1	Constructor & Destructor Documentation . . . . .	296
9.41.1.1	NodeMap() . . . . .	297
9.41.1.2	~NodeMap() . . . . .	297
9.41.2	Member Function Documentation . . . . .	297
9.41.2.1	_GetDeviceName() . . . . .	297
9.41.2.2	_GetNode() . . . . .	297
9.41.2.3	_GetNodes() . . . . .	297

9.41.2.4	<a href="#">_InvalidateNodes()</a>	297
9.41.2.5	<a href="#">_Poll()</a>	298
9.42	<a href="#">PGMOption Struct Reference</a>	298
9.42.1	<a href="#">Detailed Description</a>	298
9.42.2	<a href="#">Constructor &amp; Destructor Documentation</a>	298
9.42.2.1	<a href="#">PGMOption()</a>	298
9.42.3	<a href="#">Member Data Documentation</a>	298
9.42.3.1	<a href="#">binaryFile</a>	299
9.42.3.2	<a href="#">reserved</a>	299
9.43	<a href="#">PGRGuid Class Reference</a>	299
9.43.1	<a href="#">Detailed Description</a>	299
9.43.2	<a href="#">Constructor &amp; Destructor Documentation</a>	299
9.43.2.1	<a href="#">PGRGuid()</a>	300
9.43.3	<a href="#">Member Function Documentation</a>	300
9.43.3.1	<a href="#">operator"!=()</a>	300
9.43.3.2	<a href="#">operator==()</a>	300
9.43.4	<a href="#">Member Data Documentation</a>	300
9.43.4.1	<a href="#">value</a>	300
9.44	<a href="#">PNGOption Struct Reference</a>	300
9.44.1	<a href="#">Detailed Description</a>	301
9.44.2	<a href="#">Constructor &amp; Destructor Documentation</a>	301
9.44.2.1	<a href="#">PNGOption()</a>	301
9.44.3	<a href="#">Member Data Documentation</a>	301
9.44.3.1	<a href="#">compressionLevel</a>	301
9.44.3.2	<a href="#">interlaced</a>	301
9.44.3.3	<a href="#">reserved</a>	302
9.45	<a href="#">PPMOption Struct Reference</a>	302
9.45.1	<a href="#">Detailed Description</a>	302
9.45.2	<a href="#">Constructor &amp; Destructor Documentation</a>	302
9.45.2.1	<a href="#">PPMOption()</a>	302



9.45.3	Member Data Documentation	302
9.45.3.1	binaryFile	303
9.45.3.2	reserved	303
9.46	Property Struct Reference	303
9.46.1	Detailed Description	304
9.46.2	Constructor & Destructor Documentation	304
9.46.2.1	Property() [1/2]	304
9.46.2.2	Property() [2/2]	304
9.46.3	Member Data Documentation	304
9.46.3.1	absControl	304
9.46.3.2	absValue	305
9.46.3.3	autoManualMode	305
9.46.3.4	onePush	305
9.46.3.5	onOff	305
9.46.3.6	present	305
9.46.3.7	reserved	305
9.46.3.8	type	306
9.46.3.9	valueA	306
9.46.3.10	valueB	306
9.47	PropertyInfo Struct Reference	306
9.47.1	Detailed Description	307
9.47.2	Constructor & Destructor Documentation	307
9.47.2.1	PropertyInfo() [1/2]	307
9.47.2.2	PropertyInfo() [2/2]	308
9.47.3	Member Data Documentation	308
9.47.3.1	absMax	308
9.47.3.2	absMin	308
9.47.3.3	absValSupported	308
9.47.3.4	autoSupported	308
9.47.3.5	manualSupported	308

9.47.3.6	max	309
9.47.3.7	min	309
9.47.3.8	onePushSupported	309
9.47.3.9	onOffSupported	309
9.47.3.10	present	309
9.47.3.11	pUnitAbbr	309
9.47.3.12	pUnits	310
9.47.3.13	readOutSupported	310
9.47.3.14	reserved	310
9.47.3.15	type	310
9.48	StrobeControl Struct Reference	310
9.48.1	Detailed Description	311
9.48.2	Constructor & Destructor Documentation	311
9.48.2.1	StrobeControl()	311
9.48.3	Member Data Documentation	311
9.48.3.1	delay	311
9.48.3.2	duration	311
9.48.3.3	onOff	312
9.48.3.4	polarity	312
9.48.3.5	reserved	312
9.48.3.6	source	312
9.49	StrobeInfo Struct Reference	312
9.49.1	Detailed Description	313
9.49.2	Constructor & Destructor Documentation	313
9.49.2.1	StrobeInfo()	313
9.49.3	Member Data Documentation	313
9.49.3.1	maxValue	313
9.49.3.2	minValue	314
9.49.3.3	onOffSupported	314
9.49.3.4	polaritySupported	314

9.49.3.5	present	314
9.49.3.6	readOutSupported	314
9.49.3.7	reserved	314
9.49.3.8	source	315
9.50	SyncManager Class Reference	315
9.50.1	Constructor & Destructor Documentation	315
9.50.1.1	SyncManager()	315
9.50.1.2	~SyncManager()	315
9.50.2	Member Function Documentation	315
9.50.2.1	DisableCrossPCsynchronization()	316
9.50.2.2	EnableCrossPCsynchronization()	316
9.50.2.3	GetSyncStatus()	316
9.50.2.4	GetTimeSinceSynced()	316
9.50.2.5	IsTimingBusConnected()	316
9.50.2.6	QueryCrossPCsynchronizationSetting()	316
9.50.2.7	RescanMasterTimingBus()	316
9.50.2.8	Start()	316
9.50.2.9	Stop()	317
9.51	SystemInfo Struct Reference	317
9.51.1	Detailed Description	317
9.51.2	Member Data Documentation	317
9.51.2.1	byteOrder	318
9.51.2.2	cpuDescription	318
9.51.2.3	driverList	318
9.51.2.4	gpuDescription	318
9.51.2.5	libraryList	318
9.51.2.6	numCpuCores	318
9.51.2.7	osDescription	319
9.51.2.8	osType	319
9.51.2.9	reserved	319

9.51.2.10	screenHeight	319
9.51.2.11	screenWidth	319
9.51.2.12	sysMemSize	319
9.52	TIFFOption Struct Reference	320
9.52.1	Detailed Description	320
9.52.2	Member Enumeration Documentation	320
9.52.2.1	CompressionMethod	320
9.52.3	Constructor & Destructor Documentation	321
9.52.3.1	TIFFOption()	321
9.52.4	Member Data Documentation	321
9.52.4.1	compression	321
9.52.4.2	reserved	321
9.53	TimeStamp Struct Reference	321
9.53.1	Detailed Description	322
9.53.2	Constructor & Destructor Documentation	322
9.53.2.1	TimeStamp()	322
9.53.3	Member Data Documentation	322
9.53.3.1	cycleCount	322
9.53.3.2	cycleOffset	322
9.53.3.3	cycleSeconds	323
9.53.3.4	microSeconds	323
9.53.3.5	reserved	323
9.53.3.6	seconds	323
9.54	TopologyNode Class Reference	323
9.54.1	Detailed Description	324
9.54.2	Member Enumeration Documentation	324
9.54.2.1	NodeType	324
9.54.2.2	PortType	325
9.54.3	Constructor & Destructor Documentation	325
9.54.3.1	TopologyNode() [1/3]	325

9.54.3.2	TopologyNode() [2/3]	325
9.54.3.3	~TopologyNode()	326
9.54.3.4	TopologyNode() [3/3]	326
9.54.4	Member Function Documentation	326
9.54.4.1	AddChild()	326
9.54.4.2	AddPortType()	326
9.54.4.3	AssignGuidToNode() [1/2]	327
9.54.4.4	AssignGuidToNode() [2/2]	327
9.54.4.5	GetChild()	327
9.54.4.6	GetDeviceId()	328
9.54.4.7	GetGuid()	328
9.54.4.8	GetInterfaceType()	328
9.54.4.9	GetNodeType()	329
9.54.4.10	GetNumChildren()	329
9.54.4.11	GetNumPorts()	329
9.54.4.12	GetPortType()	329
9.54.4.13	operator=()	330
9.55	TriggerMode Struct Reference	330
9.55.1	Detailed Description	330
9.55.2	Constructor & Destructor Documentation	331
9.55.2.1	TriggerMode()	331
9.55.3	Member Data Documentation	331
9.55.3.1	mode	331
9.55.3.2	onOff	331
9.55.3.3	parameter	331
9.55.3.4	polarity	331
9.55.3.5	reserved	332
9.55.3.6	source	332
9.56	TriggerModelInfo Struct Reference	332
9.56.1	Detailed Description	333

9.56.2	Constructor & Destructor Documentation . . . . .	333
9.56.2.1	TriggerModelInfo() . . . . .	333
9.56.3	Member Data Documentation . . . . .	333
9.56.3.1	modeMask . . . . .	333
9.56.3.2	onOffSupported . . . . .	333
9.56.3.3	polaritySupported . . . . .	333
9.56.3.4	present . . . . .	333
9.56.3.5	readOutSupported . . . . .	334
9.56.3.6	reserved . . . . .	334
9.56.3.7	softwareTriggerSupported . . . . .	334
9.56.3.8	sourceMask . . . . .	334
9.56.3.9	valueReadable . . . . .	334
9.57	Utilities Class Reference . . . . .	334
9.57.1	Detailed Description . . . . .	335
9.57.2	Member Function Documentation . . . . .	335
9.57.2.1	CheckDriver() . . . . .	335
9.57.2.2	GetDriverDeviceName() . . . . .	336
9.57.2.3	GetLibraryVersion() . . . . .	336
9.57.2.4	GetSystemInfo() . . . . .	336
9.57.2.5	LaunchBrowser() . . . . .	337
9.57.2.6	LaunchCommand() . . . . .	337
9.57.2.7	LaunchCommandAsync() . . . . .	338
9.57.2.8	LaunchHelp() . . . . .	338

<b>10 File Documentation</b>	<b>339</b>
10.1 BusManager.h File Reference	339
10.2 Camera.h File Reference	341
10.3 CameraBase.h File Reference	342
10.4 Error.h File Reference	343
10.5 FlyCapture2.h File Reference	344
10.6 FlyCapture2Defs.h File Reference	345
10.6.1 Macro Definition Documentation	352
10.6.1.1 FULL_32BIT_VALUE	352
10.6.1.2 NULL	352
10.7 FlyCapture2GUI.h File Reference	353
10.8 FlyCapture2Platform.h File Reference	353
10.8.1 Macro Definition Documentation	354
10.8.1.1 FLYCAPTURE2_API	354
10.8.1.2 FLYCAPTURE2_LOCAL	354
10.9 FlyCapture2Video.h File Reference	354
10.10 FlyCapture2VideoDefs.h File Reference	355
10.11 FlyCapture3ApiGuiWrapper.h File Reference	356
10.11.1 Macro Definition Documentation	356
10.11.1.1 WRAPPER_API	356
10.12 GCCamera.h File Reference	357
10.13 GigE Camera.h File Reference	358
10.14 Image.h File Reference	359
10.15 ImageStatistics.h File Reference	360
10.16 Internal.h File Reference	361
10.17 Licensing.dox File Reference	361
10.18 MultiSyncLibrary.h File Reference	361
10.19 MultiSyncLibraryDefs.h File Reference	362
10.20 MultiSyncLibraryPlatform.h File Reference	363
10.20.1 Macro Definition Documentation	363
10.20.1.1 MULTISYNCLIBRARY_API	363
10.20.1.2 MULTISYNCLIBRARY_LOCAL	363
10.21 NodeMap.h File Reference	364
10.22 TopologyNode.h File Reference	364
10.23 Utilities.h File Reference	365
<b>Index</b>	<b>367</b>





## Chapter 1

# Software Licensing Information

Table 1.1 License table

Component	License
FlyCapture2	Copyright © 2017 FLIR Integrated Imaging Solutions, Inc. All Rights Reserved. This software is the confidential and proprietary information of FLIR Integrated Imaging Solutions, Inc. ("↵ Confidential Information"). You shall not disclose such Confidential Information and shall use it only in accordance with the terms of the license agreement you entered into with FLIR Integrated Imaging Solutions, Inc. (FLIR). FLIR MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THE SOFTWARE, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR P↵ URPOSE, OR NON-INFRINGEMENT. FLIR SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.
AdapterList	The Code Project Open License (CPOL) <a href="http://www.codeproject.com/info/cpol10.aspx">http://www.codeproject.↵ com/info/cpol10.aspx</a>
Boost	Boost Software License <a href="http://www.boost.org/users/license.html">http://www.boost.org/users/license.html</a>
FFMPEG	LGPLv2.1 License <a href="https://www.ffmpeg.org/legal.html">https://www.ffmpeg.org/legal.html</a>
FreeImage	FreeImage public license <a href="http://freeimage.sourceforge.↵ net/freeimage-license.txt">http://freeimage.sourceforge.↵ net/freeimage-license.txt</a>
GTK	LGPLv2.1 License <a href="http://www.gnu.org/licenses/old-licenses/lgpl-2.↵ 1.txt">http://www.gnu.org/licenses/old-licenses/lgpl-2.↵ 1.txt</a>
Libusb	LGPLv2.1 License <a href="http://www.gnu.org/licenses/old-licenses/lgpl-2.↵ 1.txt">http://www.gnu.org/licenses/old-licenses/lgpl-2.↵ 1.txt</a>
Libraw1394	LGPLv2.0 License <a href="http://www.gnu.org/licenses/old-licenses/lgpl-2.↵ 0.txt">http://www.gnu.org/licenses/old-licenses/lgpl-2.↵ 0.txt</a>



## Chapter 2

# Module Index

### 2.1 Modules

Here is a list of all modules:

Global constants . . . . .	15
Enumerations . . . . .	16
GigE specific enumerations . . . . .	31
Structures . . . . .	32
GigE specific structures . . . . .	34
IIDC specific structures . . . . .	35
Image saving structures. . . . .	36
Video saving structures. . . . .	37



## Chapter 3

# Namespace Index

### 3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">FlyCap3CameraControl</a>	39
<a href="#">FlyCapture2</a>	39
<a href="#">MultiSyncLibrary</a>	48



## Chapter 4

# Hierarchical Index

### 4.1 Class Hierarchy

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

AVIOption . . . . .	51
BMPOption . . . . .	52
BusManager . . . . .	53
CameraBase . . . . .	99
Camera . . . . .	65
GCCamera . . . . .	178
GigECamera . . . . .	207
CameraControlDlg . . . . .	128
CameraInfo . . . . .	132
CameraSelectionDlg . . . . .	139
CameraStats . . . . .	141
ConfigROM . . . . .	145
EmbeddedImageInfo . . . . .	148
EmbeddedImageInfoProperty . . . . .	150
Error . . . . .	151
EventCallbackData . . . . .	156
EventOptions . . . . .	158
FC2Config . . . . .	159
FC2Version . . . . .	163
FlyCapture2Video . . . . .	164
FlyCapture3ApiGuiWrapper . . . . .	168
Format7ImageSettings . . . . .	170
Format7Info . . . . .	173
Format7PacketInfo . . . . .	176
GigEConfig . . . . .	245
GigEImageSettings . . . . .	247
GigEImageSettingsInfo . . . . .	249
GigEProperty . . . . .	251
GigEStreamChannel . . . . .	253
H264Option . . . . .	255
Image . . . . .	257
ImageMetadata . . . . .	276
ImageStatistics . . . . .	279
Internal . . . . .	286
IPAddress . . . . .	287

JPEGOption . . . . .	288
JPG2Option . . . . .	290
LUTData . . . . .	291
MACAddress . . . . .	293
MJPGOption . . . . .	295
NodeMap . . . . .	296
PGMOption . . . . .	298
PGRGuid . . . . .	299
PNGOption . . . . .	300
PPMOption . . . . .	302
Property . . . . .	303
PropertyInfo . . . . .	306
StrobeControl . . . . .	310
StrobeInfo . . . . .	312
SyncManager . . . . .	315
SystemInfo . . . . .	317
TIFFOption . . . . .	320
TimeStamp . . . . .	321
TopologyNode . . . . .	323
TriggerMode . . . . .	330
TriggerModelInfo . . . . .	332
Utilities . . . . .	334



## Chapter 5

# Class Index

### 5.1 Class List

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

<a href="#">AVIOption</a>	Options for saving AVI files . . . . .	51
<a href="#">BMPOption</a>	Options for saving Bitmap image . . . . .	52
<a href="#">BusManager</a>	Functionality for the user to get an <a href="#">PGRGuid</a> for a desired camera or device easily . . . . .	53
<a href="#">Camera</a>	The <a href="#">Camera</a> object represents a physical camera that uses the IIDC register set . . . . .	65
<a href="#">CameraBase</a>	Abstract base class that defines a general interface to a camera . . . . .	99
<a href="#">CameraControlDlg</a>	The <a href="#">CameraControlDlg</a> object represents a dialog that provides a graphical interface to a specified camera . . . . .	128
<a href="#">CameraInfo</a>	<a href="#">Camera</a> information . . . . .	132
<a href="#">CameraSelectionDlg</a>	The <a href="#">CameraSelectionDlg</a> object represents a dialog that provides a graphical interface that lists the number of cameras available to the library . . . . .	139
<a href="#">CameraStats</a>	<a href="#">Camera</a> diagnostic information . . . . .	141
<a href="#">ConfigROM</a>	<a href="#">Camera</a> configuration ROM . . . . .	145
<a href="#">EmbeddedImageInfo</a>	Properties of the possible embedded image information . . . . .	148
<a href="#">EmbeddedImageInfoProperty</a>	Properties of a single embedded image info property . . . . .	150
<a href="#">Error</a>	The <a href="#">Error</a> object represents an error that is returned from the library . . . . .	151
<a href="#">EventCallbackData</a>	. . . . .	156
<a href="#">EventOptions</a>	Options for enabling device event registration . . . . .	158
<a href="#">FC2Config</a>	Configuration for a camera . . . . .	159
<a href="#">FC2Version</a>	The current version of the library . . . . .	163

<a href="#">FlyCapture2Video</a>	
Functionality for the user to record images to an AVI file . . . . .	164
<a href="#">FlyCapture3ApiGuiWrapper</a> . . . . .	168
<a href="#">Format7ImageSettings</a>	
Format 7 image settings . . . . .	170
<a href="#">Format7Info</a>	
Format 7 information for a single mode . . . . .	173
<a href="#">Format7PacketInfo</a>	
Format 7 packet information . . . . .	176
<a href="#">GCCamera</a> . . . . .	178
<a href="#">GigECamera</a>	
The <a href="#">GigECamera</a> object represents a physical Gigabit Ethernet camera . . . . .	207
<a href="#">GigEConfig</a>	
Configuration for a GigE camera . . . . .	245
<a href="#">GigEImageSettings</a>	
Image settings for a GigE camera . . . . .	247
<a href="#">GigEImageSettingsInfo</a>	
Format 7 information for a single mode . . . . .	249
<a href="#">GigEProperty</a>	
A GigE property . . . . .	251
<a href="#">GigEStreamChannel</a>	
Information about a single GigE stream channel . . . . .	253
<a href="#">H264Option</a>	
Options for saving H264 files . . . . .	255
<a href="#">Image</a>	
Used to retrieve images from a camera, convert between multiple pixel formats and save images to disk . . . . .	257
<a href="#">ImageMetadata</a>	
Metadata related to an image . . . . .	276
<a href="#">ImageStatistics</a>	
The <a href="#">ImageStatistics</a> object represents image statistics for an image . . . . .	279
<a href="#">Internal</a> . . . . .	286
<a href="#">IPAddress</a>	
IPv4 address . . . . .	287
<a href="#">JPEGOption</a>	
Options for saving JPEG image . . . . .	288
<a href="#">JPG2Option</a>	
Options for saving JPEG2000 image . . . . .	290
<a href="#">LUTData</a>	
Information about the camera's look up table . . . . .	291
<a href="#">MACAddress</a>	
MAC address . . . . .	293
<a href="#">MJPEGOption</a>	
Options for saving MJPG files . . . . .	295
<a href="#">NodeMap</a> . . . . .	296
<a href="#">PGMOption</a>	
Options for saving PGM images . . . . .	298
<a href="#">PGRGuid</a>	
A GUID to the camera . . . . .	299
<a href="#">PNGOption</a>	
Options for saving PNG images . . . . .	300
<a href="#">PPMOption</a>	
Options for saving PPM images . . . . .	302
<a href="#">Property</a>	
A specific camera property . . . . .	303
<a href="#">PropertyInfo</a>	
Information about a specific camera property . . . . .	306

<a href="#">StrobeControl</a>	
A camera strobe . . . . .	310
<a href="#">StrobeInfo</a>	
A camera strobe property . . . . .	312
<a href="#">SyncManager</a> . . . . .	315
<a href="#">SystemInfo</a>	
Description of the system . . . . .	317
<a href="#">TIFFOption</a>	
Options for saving TIFF images . . . . .	320
<a href="#">TimeStamp</a>	
Timestamp information . . . . .	321
<a href="#">TopologyNode</a>	
Topology information that can be used to generate a tree structure of all cameras and devices connected to a computer . . . . .	323
<a href="#">TriggerMode</a>	
A camera trigger . . . . .	330
<a href="#">TriggerModelInfo</a>	
Information about a camera trigger property . . . . .	332
<a href="#">Utilities</a>	
The Utility class is generally used to query for general system information such as operating system, available memory etc . . . . .	334



## Chapter 6

# File Index

### 6.1 File List

Here is a list of all files with brief descriptions:

BusManager.h	339
Camera.h	341
CameraBase.h	342
Error.h	343
FlyCapture2.h	344
FlyCapture2Defs.h	345
FlyCapture2GUI.h	353
FlyCapture2Platform.h	353
FlyCapture2Video.h	354
FlyCapture2VideoDefs.h	355
FlyCapture3ApiGuiWrapper.h	356
GCCamera.h	357
GigECamera.h	358
Image.h	359
ImageStatistics.h	360
Internal.h	361
MultiSyncLibrary.h	361
MultiSyncLibraryDefs.h	362
MultiSyncLibraryPlatform.h	363
NodeMap.h	364
TopologyNode.h	364
Utilities.h	365



# Chapter 7

## Module Documentation

### 7.1 Global constants

#### Variables

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

#### 7.1.1 Detailed Description

#### 7.1.2 Variable Documentation

##### 7.1.2.1 `sk_maxNumPorts`

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

The maximum number of ports one device can have.

##### 7.1.2.2 `sk_maxStringLength`

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

The maximum length that is allocated for a string.

## 7.2 Enumerations

### Enumerations

- enum `ErrorType` {  
`PGRERROR_UNDEFINED` = -1,  
`PGRERROR_OK`,  
`PGRERROR_FAILED`,  
`PGRERROR_NOT_IMPLEMENTED`,  
`PGRERROR_FAILED_BUS_MASTER_CONNECTION`,  
`PGRERROR_NOT_CONNECTED`,  
`PGRERROR_INIT_FAILED`,  
`PGRERROR_NOT_INITIALIZED`,  
`PGRERROR_INVALID_PARAMETER`,  
`PGRERROR_INVALID_SETTINGS`,  
`PGRERROR_INVALID_BUS_MANAGER`,  
`PGRERROR_MEMORY_ALLOCATION_FAILED`,  
`PGRERROR_LOW_LEVEL_FAILURE`,  
`PGRERROR_NOT_FOUND`,  
`PGRERROR_FAILED_GUID`,  
`PGRERROR_INVALID_PACKET_SIZE`,  
`PGRERROR_INVALID_MODE`,  
`PGRERROR_NOT_IN_FORMAT7`,  
`PGRERROR_NOT_SUPPORTED`,  
`PGRERROR_TIMEOUT`,  
`PGRERROR_BUS_MASTER_FAILED`,  
`PGRERROR_INVALID_GENERATION`,  
`PGRERROR_LUT_FAILED`,  
`PGRERROR_IIDC_FAILED`,  
`PGRERROR_STROBE_FAILED`,  
`PGRERROR_TRIGGER_FAILED`,  
`PGRERROR_PROPERTY_FAILED`,  
`PGRERROR_PROPERTY_NOT_PRESENT`,  
`PGRERROR_REGISTER_FAILED`,  
`PGRERROR_READ_REGISTER_FAILED`,  
`PGRERROR_WRITE_REGISTER_FAILED`,  
`PGRERROR_ISOCH_FAILED`,  
`PGRERROR_ISOCH_ALREADY_STARTED`,  
`PGRERROR_ISOCH_NOT_STARTED`,  
`PGRERROR_ISOCH_START_FAILED`,  
`PGRERROR_ISOCH_RETRIEVE_BUFFER_FAILED`,  
`PGRERROR_ISOCH_STOP_FAILED`,  
`PGRERROR_ISOCH_SYNC_FAILED`,  
`PGRERROR_ISOCH_BANDWIDTH_EXCEEDED`,  
`PGRERROR_IMAGE_CONVERSION_FAILED`,  
`PGRERROR_IMAGE_LIBRARY_FAILURE`,  
`PGRERROR_BUFFER_TOO_SMALL`,  
`PGRERROR_IMAGE_CONSISTENCY_ERROR`,  
`PGRERROR_INCOMPATIBLE_DRIVER`,  
`PGRERROR_FORCE_32BITS` = `FULL_32BIT_VALUE` }

*The error types returned by functions.*

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

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



- enum `GrabMode` {  
`DROP_FRAMES`,  
`BUFFER_FRAMES`,  
`UNSPECIFIED_GRAB_MODE`,  
`GRAB_MODE_FORCE_32BITS` = `FULL_32BIT_VALUE` }  
*The grab strategy employed during image transfer.*
- enum `GrabTimeout` {  
`TIMEOUT_NONE` = 0,  
`TIMEOUT_INFINITE` = -1,  
`TIMEOUT_UNSPECIFIED` = -2,  
`GRAB_TIMEOUT_FORCE_32BITS` = `FULL_32BIT_VALUE` }  
*Timeout options for grabbing images.*
- enum `BandwidthAllocation` {  
`BANDWIDTH_ALLOCATION_OFF` = 0,  
`BANDWIDTH_ALLOCATION_ON` = 1,  
`BANDWIDTH_ALLOCATION_UNSUPPORTED` = 2,  
`BANDWIDTH_ALLOCATION_UNSPECIFIED` = 3,  
`BANDWIDTH_ALLOCATION_FORCE_32BITS` = `FULL_32BIT_VALUE` }  
*Bandwidth allocation options for 1394 devices.*
- enum `InterfaceType` {  
`INTERFACE_IEEE1394`,  
`INTERFACE_USB2`,  
`INTERFACE_USB3`,  
`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,
    PIXEL_FORMAT_BGR16 = 0x02000001,
    PIXEL_FORMAT_BGRU16 = 0x02000002,
    PIXEL_FORMAT_422YUV8_JPEG = 0x40000001,
    NUM_PIXEL_FORMATS = 20,
    UNSPECIFIED_PIXEL_FORMAT = 0 }

```

*Pixel formats available for Format7 modes.*

```

• enum BusSpeed {
    BUSSPEED_S100,
    BUSSPEED_S200,
    BUSSPEED_S400,
    BUSSPEED_S480,
    BUSSPEED_S800,
    BUSSPEED_S1600,
    BUSSPEED_S3200,
    BUSSPEED_S5000,
    BUSSPEED_10BASE_T,
    BUSSPEED_100BASE_T,
    BUSSPEED_1000BASE_T,
    BUSSPEED_10000BASE_T,
    BUSSPEED_S_FASTEST,
    BUSSPEED_ANY,
    BUSSPEED_SPEED_UNKNOWN = -1,
    BUSSPEED_FORCE_32BITS = FULL_32BIT_VALUE }

```

*Bus speeds.*

```

• enum PCIeBusSpeed {
    PCIE_BUSSPEED_2_5,
    PCIE_BUSSPEED_5_0,
    PCIE_BUSSPEED_UNKNOWN = -1,
    PCIE_BUSSPEED_FORCE_32BITS = FULL_32BIT_VALUE }

```

- enum `DriverType` {  
`DRIVER_1394_CAM`,  
`DRIVER_1394_PRO`,  
`DRIVER_1394_JUUJ`,  
`DRIVER_1394_VIDEO1394`,  
`DRIVER_1394_RAW1394`,  
`DRIVER_USB_NONE`,  
`DRIVER_USB_CAM`,  
`DRIVER_USB3_PRO`,  
`DRIVER_GIGE_NONE`,  
`DRIVER_GIGE_FILTER`,  
`DRIVER_GIGE_PRO`,  
`DRIVER_GIGE_LWF`,  
`DRIVER_UNKNOWN` = -1,  
`DRIVER_FORCE_32BITS` = `FULL_32BIT_VALUE` }

*Types of low level drivers that flycapture uses.*

- enum `ColorProcessingAlgorithm` {  
`DEFAULT`,  
`NO_COLOR_PROCESSING`,  
`NEAREST_NEIGHBOR`,  
`EDGE_SENSING`,  
`HQ_LINEAR`,  
`RIGOROUS`,  
`IPP`,  
`DIRECTIONAL_FILTER`,  
`WEIGHTED_DIRECTIONAL_FILTER`,  
`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.*

### 7.2.1 Detailed Description

### 7.2.2 Enumeration Type Documentation

### 7.2.2.1 BandwidthAllocation

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	

### 7.2.2.2 BayerTileFormat

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	

### 7.2.2.3 BusCallbackType

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	

### 7.2.2.4 BusSpeed

enum `BusSpeed`

Bus speeds.

Enumerator

BUSSPEED_S100	100Mbps/sec.
BUSSPEED_S200	200Mbps/sec.
BUSSPEED_S400	400Mbps/sec.
BUSSPEED_S480	480Mbps/sec. Only for USB2 cameras.
BUSSPEED_S800	800Mbps/sec.
BUSSPEED_S1600	1600Mbps/sec.
BUSSPEED_S3200	3200Mbps/sec.
BUSSPEED_S5000	5000Mbps/sec. Only for USB3 cameras.
BUSSPEED_10BASE_T	10Base-T. Only for GigE Vision cameras.
BUSSPEED_100BASE_T	100Base-T. Only for GigE Vision cameras.
BUSSPEED_1000BASE_T	1000Base-T (Gigabit Ethernet). Only for GigE Vision cameras.
BUSSPEED_10000BASE_T	10000Base-T. Only for GigE Vision cameras.
BUSSPEED_S_FASTEST	The fastest speed available.
BUSSPEED_ANY	Any speed that is available.
BUSSPEED_SPEED_UNKNOWN	Unknown bus speed.
BUSSPEED_FORCE_32BITS	

### 7.2.2.5 ColorProcessingAlgorithm

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	Well-balanced speed and quality.
RIGOROUS	Slowest but produces good results.
IPP	Multithreaded with similar results to edge sensing.
DIRECTIONAL_FILTER	Best quality but much faster than rigorous.
WEIGHTED_DIRECTIONAL_FILTER	Weighted pixel average from different directions.
COLOR_PROCESSING_ALGORITHM_FORCE_↵ 32BITS	

## 7.2.2.6 DriverType

enum [DriverType](#)

Types of low level drivers that flycapture uses.

## Enumerator

DRIVER_1394_CAM	PGRCam.sys.
DRIVER_1394_PRO	PGR1394.sys.
DRIVER_1394_JUUJ	firewire_core.
DRIVER_1394_VIDEO1394	video1394.
DRIVER_1394_RAW1394	raw1394.
DRIVER_USB_NONE	No usb driver used just BSD stack. (Linux only)
DRIVER_USB_CAM	PGRUsbCam.sys.
DRIVER_USB3_PRO	PGRXHCl.sys.
DRIVER_GIGE_NONE	no gige drivers used,MS/BSD stack.
DRIVER_GIGE_FILTER	PGRGigE.sys.
DRIVER_GIGE_PRO	PGRGigEPro.sys.
DRIVER_GIGE_LWF	PgrLwf.sys.
DRIVER_UNKNOWN	Unknown driver type.
DRIVER_FORCE_32BITS	

## 7.2.2.7 ErrorType

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	<a href="#">Camera</a> has not been connected.
PGRERROR_INIT_FAILED	Initialization failed.
PGRERROR_NOT_INITIALIZED	<a href="#">Camera</a> 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.

## Enumerator

PGRERROR_FAILED_GUID	GUID failure.
PGRERROR_INVALID_PACKET_SIZE	Packet size set to camera is invalid.
PGRERROR_INVALID_MODE	Invalid mode has been passed to function.
PGRERROR_NOT_IN_FORMAT7	Error due to not being in Format7.
PGRERROR_NOT_SUPPORTED	This feature is unsupported.
PGRERROR_TIMEOUT	Timeout error.
PGRERROR_BUS_MASTER_FAILED	Bus Master Failure.
PGRERROR_INVALID_GENERATION	Generation Count Mismatch.
PGRERROR_LUT_FAILED	Look Up Table failure.
PGRERROR_IIDC_FAILED	IIDC failure.
PGRERROR_STROBE_FAILED	Strobe failure.
PGRERROR_TRIGGER_FAILED	Trigger failure.
PGRERROR_PROPERTY_FAILED	Property failure.
PGRERROR_PROPERTY_NOT_PRESENT	Property is not present.
PGRERROR_REGISTER_FAILED	Register access failed.
PGRERROR_READ_REGISTER_FAILED	Register read failed.
PGRERROR_WRITE_REGISTER_FAILED	Register write failed.
PGRERROR_ISOCH_FAILED	Isochronous failure.
PGRERROR_ISOCH_ALREADY_STARTED	Isochronous transfer has already been started.
PGRERROR_ISOCH_NOT_STARTED	Isochronous transfer has not been started.
PGRERROR_ISOCH_START_FAILED	Isochronous start failed.
PGRERROR_ISOCH_RETRIEVE_BUFFER_FAILED	Isochronous retrieve buffer failed.
PGRERROR_ISOCH_STOP_FAILED	Isochronous stop failed.
PGRERROR_ISOCH_SYNC_FAILED	Isochronous image synchronization failed.
PGRERROR_ISOCH_BANDWIDTH_EXCEEDED	Isochronous bandwidth exceeded.
PGRERROR_IMAGE_CONVERSION_FAILED	Image conversion failed.
PGRERROR_IMAGE_LIBRARY_FAILURE	Image library failure.
PGRERROR_BUFFER_TOO_SMALL	Buffer is too small.
PGRERROR_IMAGE_CONSISTENCY_ERROR	There is an image consistency error.
PGRERROR_INCOMPATIBLE_DRIVER	The installed driver is not compatible with the library.
PGRERROR_FORCE_32BITS	

## 7.2.2.8 FrameRate

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.



## Enumerator

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	

## 7.2.2.9 GrabMode

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 <a href="#">FC2Config</a> 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	

## 7.2.2.10 GrabTimeout

enum [GrabTimeout](#)

Timeout options for grabbing images.

## Enumerator

TIMEOUT_NONE	Non-blocking wait.
TIMEOUT_INFINITE	Wait indefinitely.
TIMEOUT_UNSPECIFIED	Unspecified timeout setting.
GRAB_TIMEOUT_FORCE_32BITS	

### 7.2.2.11 ImageFileFormat

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	

### 7.2.2.12 InterfaceType

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_USB3	USB 3.0.
INTERFACE_GIGE	GigE.
INTERFACE_UNKNOWN	Unknown interface.
INTERFACE_TYPE_FORCE_32BITS	

### 7.2.2.13 Mode

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	

## 7.2.2.14 PCIeBusSpeed

enum `PCIeBusSpeed`

## Enumerator

PCIE_BUSSPEED_2_5	
PCIE_BUSSPEED_5_0	2.5 Gb/s
PCIE_BUSSPEED_UNKNOWN	5.0 Gb/s
PCIE_BUSSPEED_FORCE_32BITS	Speed is unknown.

### 7.2.2.15 PixelFormat

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.
PIXEL_FORMAT_BGR16	R = G = B = 16 bits.
PIXEL_FORMAT_BGRU16	64 bit BGRU.
PIXEL_FORMAT_422YUV8_JPEG	JPEG compressed stream.
NUM_PIXEL_FORMATS	Number of pixel formats.
UNSPECIFIED_PIXEL_FORMAT	Unspecified pixel format.

### 7.2.2.16 PropertyType

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.

## Enumerator

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	

## 7.2.2.17 VideoMode

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.

## Enumerator

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	

## 7.3 GigE specific enumerations

These enumerations are specific to GigE camera operation only.

### Enumerations

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

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

### 7.3.1 Detailed Description

These enumerations are specific to GigE camera operation only.

### 7.3.2 Enumeration Type Documentation

#### 7.3.2.1 GigEPropertyType

enum `GigEPropertyType`

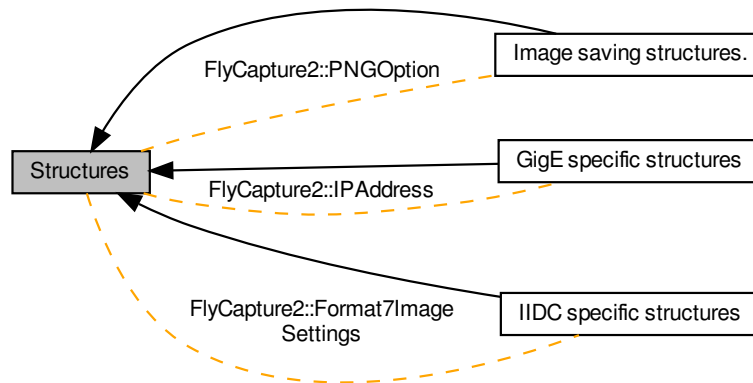
Possible properties that can be queried from the camera.

#### Enumerator

HEARTBEAT	
HEARTBEAT_TIMEOUT	
PACKET_SIZE	
PACKET_DELAY	

## 7.4 Structures

Collaboration diagram for Structures:



### Modules

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

### Classes

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



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

## Typedefs

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

### 7.4.1 Detailed Description

### 7.4.2 Typedef Documentation

#### 7.4.2.1 TriggerDelay

```
typedef Property TriggerDelay
```

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

#### 7.4.2.2 TriggerDelayInfo

```
typedef PropertyInfo TriggerDelayInfo
```

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

## 7.5 GigE specific structures

These structures are specific to GigE camera operation only.

Collaboration diagram for GigE specific structures:



### Classes

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

### 7.5.1 Detailed Description

These structures are specific to GigE camera operation only.

## 7.6 IIDC specific structures

These structures are specific to IIDC camera operation only.

Collaboration diagram for IIDC specific structures:



### Classes

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

### 7.6.1 Detailed Description

These structures are specific to IIDC camera operation only.

## 7.7 Image saving structures.

These structures define various parameters used for saving images.

Collaboration diagram for Image saving structures.:



### Classes

- struct [PNGOption](#)  
*Options for saving PNG images.*
- struct [PPMOption](#)  
*Options for saving PPM images.*
- struct [PGMOption](#)  
*Options for saving PGM images.*
- struct [TIFFOption](#)  
*Options for saving TIFF images.*
- struct [JPEGOption](#)  
*Options for saving JPEG image.*
- struct [JPG2Option](#)  
*Options for saving JPEG2000 image.*
- struct [BMPOption](#)  
*Options for saving Bitmap image.*
- struct [EventOptions](#)  
*Options for enabling device event registration.*
- struct [EventCallbackData](#)

### Typedefs

- typedef void(\* [CameraEventCallback](#)) (void \*data)

#### 7.7.1 Detailed Description

These structures define various parameters used for saving images.

#### 7.7.2 Typedef Documentation

##### 7.7.2.1 CameraEventCallback

```
typedef void(* CameraEventCallback) (void *data)
```

## 7.8 Video saving structures.

These structures define various parameters used for saving videos.

### Classes

- struct [MJPGOption](#)  
*Options for saving MJPG files.*
- struct [H264Option](#)  
*Options for saving H264 files.*
- struct [AVIOption](#)  
*Options for saving AVI files.*

### 7.8.1 Detailed Description

These structures define various parameters used for saving videos.



## Chapter 8

# Namespace Documentation

### 8.1 FlyCap3CameraControl Namespace Reference

#### Classes

- class [FlyCapture3ApiGuiWrapper](#)

### 8.2 FlyCapture2 Namespace Reference

#### Classes

- struct [AVIOption](#)  
*Options for saving AVI files.*
- struct [BMPOption](#)  
*Options for saving Bitmap image.*
- class [BusManager](#)  
*The [BusManager](#) class provides the functionality for the user to get an [PGRGuid](#) for a desired camera or device easily.*
- class [Camera](#)  
*The [Camera](#) object represents a physical camera that uses the IIDC register set.*
- class [CameraBase](#)  
*The [CameraBase](#) class is an abstract base class that defines a general interface to a camera.*
- class [CameraControlDlg](#)  
*The [CameraControlDlg](#) object represents a dialog that provides a graphical interface to a specified camera.*
- struct [CameraInfo](#)  
*[Camera](#) information.*
- class [CameraSelectionDlg](#)  
*The [CameraSelectionDlg](#) object represents a dialog that provides a graphical interface that lists the number of cameras available to the library.*
- struct [CameraStats](#)  
*[Camera](#) diagnostic information.*
- struct [ConfigROM](#)  
*[Camera](#) configuration ROM.*
- struct [EmbeddedImageInfo](#)  
*Properties of the possible embedded image information.*

- struct [EmbeddedImageInfoProperty](#)  
*Properties of a single embedded image info property.*
- class [Error](#)  
*The [Error](#) object represents an error that is returned from the library.*
- struct [EventCallbackData](#)
- struct [EventOptions](#)  
*Options for enabling device event registration.*
- struct [FC2Config](#)  
*Configuration for a camera.*
- struct [FC2Version](#)  
*The current version of the library.*
- class [FlyCapture2Video](#)  
*The [FlyCapture2Video](#) class provides the functionality for the user to record images to an AVI file.*
- struct [Format7ImageSettings](#)  
*Format 7 image settings.*
- struct [Format7Info](#)  
*Format 7 information for a single mode.*
- struct [Format7PacketInfo](#)  
*Format 7 packet information.*
- class [GCCamera](#)
- class [GigECamera](#)  
*The [GigECamera](#) object represents a physical Gigabit Ethernet camera.*
- struct [GigEConfig](#)  
*Configuration for a GigE camera.*
- struct [GigEImageSettings](#)  
*[Image](#) settings for a GigE camera.*
- struct [GigEImageSettingsInfo](#)  
*Format 7 information for a single mode.*
- struct [GigEProperty](#)  
*A GigE property.*
- struct [GigEStreamChannel](#)  
*Information about a single GigE stream channel.*
- struct [H264Option](#)  
*Options for saving H264 files.*
- class [Image](#)  
*The [Image](#) class is used to retrieve images from a camera, convert between multiple pixel formats and save images to disk.*
- struct [ImageMetadata](#)  
*Metadata related to an image.*
- class [ImageStatistics](#)  
*The [ImageStatistics](#) object represents image statistics for an image.*
- class [Internal](#)
- struct [IPAddress](#)  
*IPv4 address.*
- struct [JPEGOption](#)  
*Options for saving JPEG image.*
- struct [JPG2Option](#)  
*Options for saving JPEG2000 image.*
- struct [LUTData](#)  
*Information about the camera's look up table.*
- struct [MACAddress](#)



- MAC address.
- struct [MJPGOption](#)
  - Options for saving MJPG files.
- class [NodeMap](#)
- struct [PGMOption](#)
  - Options for saving PGM images.
- class [PGRGuid](#)
  - A GUID to the camera.
- struct [PNGOption](#)
  - Options for saving PNG images.
- struct [PPMOption](#)
  - Options for saving PPM images.
- struct [Property](#)
  - A specific camera property.
- struct [PropertyInfo](#)
  - Information about a specific camera property.
- struct [StrobeControl](#)
  - A camera strobe.
- struct [StrobeInfo](#)
  - A camera strobe property.
- struct [SystemInfo](#)
  - Description of the system.
- struct [TIFFOption](#)
  - Options for saving TIFF images.
- struct [TimeStamp](#)
  - Timestamp information.
- 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 [TriggerMode](#)
  - A camera trigger.
- struct [TriggerModelInfo](#)
  - Information about a camera trigger property.
- class [Utilities](#)
  - The Utility class is generally used to query for general system information such as operating system, available memory etc.

## Typedefs

- typedef void(\* [BusEventCallback](#)) (void \*pParameter, unsigned int serialNumber)
  - Bus event callback function prototype.
- typedef void \* [CallbackHandle](#)
  - Handle that is returned when registering a callback.
- typedef void(\* [ImageEventCallback](#)) (class [Image](#) \*pImage, const void \*pCallbackData)
  - [Image](#) event callback function prototype.
- typedef [PropertyInfo](#) [TriggerDelayInfo](#)
  - The [TriggerDelayInfo](#) structure is identical to [PropertyInfo](#).
- typedef [Property](#) [TriggerDelay](#)
  - The [TriggerDelay](#) structure is identical to [Property](#).
- typedef void(\* [CameraEventCallback](#)) (void \*data)
- typedef void(\* [AsyncCommandCallback](#)) (class [Error](#) retError, void \*pUserData)
  - Async command callback function prototype.

## Enumerations

- enum `ErrorType` {  
`PGRERROR_UNDEFINED` = -1,  
`PGRERROR_OK`,  
`PGRERROR_FAILED`,  
`PGRERROR_NOT_IMPLEMENTED`,  
`PGRERROR_FAILED_BUS_MASTER_CONNECTION`,  
`PGRERROR_NOT_CONNECTED`,  
`PGRERROR_INIT_FAILED`,  
`PGRERROR_NOT_INITIALIZED`,  
`PGRERROR_INVALID_PARAMETER`,  
`PGRERROR_INVALID_SETTINGS`,  
`PGRERROR_INVALID_BUS_MANAGER`,  
`PGRERROR_MEMORY_ALLOCATION_FAILED`,  
`PGRERROR_LOW_LEVEL_FAILURE`,  
`PGRERROR_NOT_FOUND`,  
`PGRERROR_FAILED_GUID`,  
`PGRERROR_INVALID_PACKET_SIZE`,  
`PGRERROR_INVALID_MODE`,  
`PGRERROR_NOT_IN_FORMAT7`,  
`PGRERROR_NOT_SUPPORTED`,  
`PGRERROR_TIMEOUT`,  
`PGRERROR_BUS_MASTER_FAILED`,  
`PGRERROR_INVALID_GENERATION`,  
`PGRERROR_LUT_FAILED`,  
`PGRERROR_IIDC_FAILED`,  
`PGRERROR_STROBE_FAILED`,  
`PGRERROR_TRIGGER_FAILED`,  
`PGRERROR_PROPERTY_FAILED`,  
`PGRERROR_PROPERTY_NOT_PRESENT`,  
`PGRERROR_REGISTER_FAILED`,  
`PGRERROR_READ_REGISTER_FAILED`,  
`PGRERROR_WRITE_REGISTER_FAILED`,  
`PGRERROR_ISOCH_FAILED`,  
`PGRERROR_ISOCH_ALREADY_STARTED`,  
`PGRERROR_ISOCH_NOT_STARTED`,  
`PGRERROR_ISOCH_START_FAILED`,  
`PGRERROR_ISOCH_RETRIEVE_BUFFER_FAILED`,  
`PGRERROR_ISOCH_STOP_FAILED`,  
`PGRERROR_ISOCH_SYNC_FAILED`,  
`PGRERROR_ISOCH_BANDWIDTH_EXCEEDED`,  
`PGRERROR_IMAGE_CONVERSION_FAILED`,  
`PGRERROR_IMAGE_LIBRARY_FAILURE`,  
`PGRERROR_BUFFER_TOO_SMALL`,  
`PGRERROR_IMAGE_CONSISTENCY_ERROR`,  
`PGRERROR_INCOMPATIBLE_DRIVER`,  
`PGRERROR_FORCE_32BITS` = `FULL_32BIT_VALUE` }

*The error types returned by functions.*

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

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

- enum `GrabMode` {  
`DROP_FRAMES`,  
`BUFFER_FRAMES`,

```
UNSPECIFIED_GRAB_MODE,
GRAB_MODE_FORCE_32BITS = FULL_32BIT_VALUE }
```

*The grab strategy employed during image transfer.*

- enum `GrabTimeout` {  
`TIMEOUT_NONE` = 0,  
`TIMEOUT_INFINITE` = -1,  
`TIMEOUT_UNSPECIFIED` = -2,  
`GRAB_TIMEOUT_FORCE_32BITS` = FULL\_32BIT\_VALUE }

*Timeout options for grabbing images.*

- enum `BandwidthAllocation` {  
`BANDWIDTH_ALLOCATION_OFF` = 0,  
`BANDWIDTH_ALLOCATION_ON` = 1,  
`BANDWIDTH_ALLOCATION_UNSUPPORTED` = 2,  
`BANDWIDTH_ALLOCATION_UNSPECIFIED` = 3,  
`BANDWIDTH_ALLOCATION_FORCE_32BITS` = FULL\_32BIT\_VALUE }

*Bandwidth allocation options for 1394 devices.*

- enum `InterfaceType` {  
`INTERFACE_IEEE1394`,  
`INTERFACE_USB2`,  
`INTERFACE_USB3`,  
`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,
PIXEL_FORMAT_BGR16 = 0x02000001,
PIXEL_FORMAT_BGRU16 = 0x02000002,
PIXEL_FORMAT_422YUV8_JPEG = 0x40000001,
NUM_PIXEL_FORMATS = 20,
UNSPECIFIED_PIXEL_FORMAT = 0 }

```

*Pixel formats available for Format7 modes.*

- enum BusSpeed {
 

```

BUSSPEED_S100,
BUSSPEED_S200,
BUSSPEED_S400,
BUSSPEED_S480,
BUSSPEED_S800,
BUSSPEED_S1600,
BUSSPEED_S3200,
BUSSPEED_S5000,
BUSSPEED_10BASE_T,
BUSSPEED_100BASE_T,
BUSSPEED_1000BASE_T,
BUSSPEED_10000BASE_T,
BUSSPEED_S_FASTEST,
BUSSPEED_ANY,
BUSSPEED_SPEED_UNKNOWN = -1,
BUSSPEED_FORCE_32BITS = FULL_32BIT_VALUE }

```

*Bus speeds.*

- enum PCIeBusSpeed {
 

```

PCIE_BUSSPEED_2_5,
PCIE_BUSSPEED_5_0,
PCIE_BUSSPEED_UNKNOWN = -1,
PCIE_BUSSPEED_FORCE_32BITS = FULL_32BIT_VALUE }

```
- enum DriverType {
 

```

DRIVER_1394_CAM,

```

```

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

```

*Types of low level drivers that flycapture uses.*

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

*Color processing algorithms.*

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

*Bayer tile formats.*

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

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

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

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

- enum `OSType` {  
`WINDOWS_X86`,  
`WINDOWS_X64`,  
`LINUX_X86`,  
`LINUX_X64`,

```
MAC,
UNKNOWN_OS,
OSTYPE_FORCE_32BITS = FULL_32BIT_VALUE }
```

*Possible operating systems.*

- enum `ByteOrder` {  
`BYTE_ORDER_LITTLE_ENDIAN`,  
`BYTE_ORDER_BIG_ENDIAN`,  
`BYTE_ORDER_FORCE_32BITS` = `FULL_32BIT_VALUE` }

*Possible byte orders.*

## Variables

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

## 8.2.1 Typedef Documentation

### 8.2.1.1 AsyncCommandCallback

```
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()`.

### 8.2.1.2 BusEventCallback

```
typedef void(* BusEventCallback) (void *pParameter, unsigned int serialNumber)
```

Bus event callback function prototype.

Defines the syntax of the callback function that is passed into `RegisterCallback()` and `UnregisterCallback()`. It is recommended that minimal handling be performed in this callback as it will block internal processing of bus events until it returns.

### 8.2.1.3 CallbackHandle

```
typedef void* CallbackHandle
```

Handle that is returned when registering a callback.

It is required when unregistering the callback.

#### 8.2.1.4 ImageEventCallback

```
typedef void(* ImageEventCallback) (class Image *pImage, const void *pCallbackData)
```

`Image` event callback function prototype.

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

### 8.2.2 Enumeration Type Documentation

#### 8.2.2.1 ByteOrder

```
enum ByteOrder
```

Possible byte orders.

Enumerator

BYTE_ORDER_LITTLE_ENDIAN	
BYTE_ORDER_BIG_ENDIAN	
BYTE_ORDER_FORCE_32BITS	

#### 8.2.2.2 OSType

```
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	

## 8.3 MultiSyncLibrary Namespace Reference



## Classes

- class [SyncManager](#)

## Enumerations

- enum [PGRSyncError](#) {  
    [PGRSyncError\\_OK](#) = 0,  
    [PGRSyncError\\_FAILED](#),  
    [PGRSyncError\\_ALREADY\\_STARTED](#),  
    [PGRSyncError\\_ALREADY\\_STOPPED](#),  
    [PGRSyncError\\_CAMERA\\_NOT\\_FOUND](#),  
    [PGRSyncError\\_UNKNOWN\\_ERROR](#) }
- enum [PGRSyncMessage](#) {  
    [PGRSyncMessage\\_OK](#) = 0,  
    [PGRSyncMessage\\_STARTED](#),  
    [PGRSyncMessage\\_STOPPED](#),  
    [PGRSyncMessage\\_SYNCING](#),  
    [PGRSyncMessage\\_NOMASTER](#),  
    [PGRSyncMessage\\_THREAD\\_ERROR](#),  
    [PGRSyncMessage\\_DEVICE\\_ERROR](#),  
    [PGRSyncMessage\\_NOT\\_ENOUGH\\_DEVICES](#),  
    [PGRSyncMessage\\_BUS\\_RESET](#),  
    [PGRSyncMessage\\_NOT\\_INITIALIZED](#),  
    [PGRSyncMessage\\_UNKNOWN\\_ERROR](#) }

### 8.3.1 Enumeration Type Documentation

#### 8.3.1.1 PGRSyncError

enum [PGRSyncError](#)

##### Enumerator

<a href="#">PGRSyncError_OK</a>	
<a href="#">PGRSyncError_FAILED</a>	
<a href="#">PGRSyncError_ALREADY_STARTED</a>	
<a href="#">PGRSyncError_ALREADY_STOPPED</a>	
<a href="#">PGRSyncError_CAMERA_NOT_FOUND</a>	
<a href="#">PGRSyncError_UNKNOWN_ERROR</a>	

#### 8.3.1.2 PGRSyncMessage

enum [PGRSyncMessage](#)

## Enumerator

PGRSyncMessage_OK	
PGRSyncMessage_STARTED	
PGRSyncMessage_STOPPED	
PGRSyncMessage_SYNCING	
PGRSyncMessage_NOMASTER	
PGRSyncMessage_THREAD_ERROR	
PGRSyncMessage_DEVICE_ERROR	
PGRSyncMessage_NOT_ENOUGH_DEVICES	
PGRSyncMessage_BUS_RESET	
PGRSyncMessage_NOT_INITIALIZED	
PGRSyncMessage_UNKNOWN_ERROR	

## Chapter 9

# Class Documentation

### 9.1 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.*

#### 9.1.1 Detailed Description

Options for saving AVI files.

#### 9.1.2 Constructor & Destructor Documentation

##### 9.1.2.1 AVIOption()

```
AVIOption ( ) [inline]
```

### 9.1.3 Member Data Documentation

#### 9.1.3.1 `frameRate`

```
float frameRate
```

Frame rate of the stream.

#### 9.1.3.2 `reserved`

```
unsigned int reserved[256]
```

Reserved for future use.

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

- [FlyCapture2VideoDefs.h](#)

## 9.2 BMPOption Struct Reference

Options for saving Bitmap image.

### Public Member Functions

- [BMPOption](#) ()

### Public Attributes

- bool [indexedColor\\_8bit](#)
- unsigned int [reserved](#) [16]

*Reserved for future use.*

#### 9.2.1 Detailed Description

Options for saving Bitmap image.

#### 9.2.2 Constructor & Destructor Documentation

## 9.2.2.1 BMOption()

```
BMOption ( ) [inline]
```

## 9.2.3 Member Data Documentation

## 9.2.3.1 indexedColor\_8bit

```
bool indexedColor_8bit
```

## 9.2.3.2 reserved

```
unsigned int reserved[16]
```

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

## 9.3 BusManager Class Reference

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

## Public Member Functions

- [BusManager](#) ()  
*Default constructor.*
- virtual [~BusManager](#) ()  
*Default destructor.*
- virtual [Error FireBusReset](#) ([PGRGuid](#) \*pGuid)  
*Fire a bus reset.*
- virtual [Error GetNumOfCameras](#) (unsigned int \*pNumCameras)  
*Gets the number of cameras attached to the PC.*
- virtual [Error GetCameraFromIPAddress](#) ([IPAddress](#) ipAddress, [PGRGuid](#) \*pGuid)  
*Gets the [PGRGuid](#) for a camera with the specified IPv4 address.*
- virtual [Error GetCameraFromIndex](#) (unsigned int index, [PGRGuid](#) \*pGuid)  
*Gets the [PGRGuid](#) for a camera on the PC.*
- virtual [Error GetCameraFromSerialNumber](#) (unsigned int serialNumber, [PGRGuid](#) \*pGuid)  
*Gets the [PGRGuid](#) for a camera on the PC.*
- virtual [Error GetCameraSerialNumberFromIndex](#) (unsigned int index, unsigned int \*pSerialNumber)

- Gets the serial number of the camera with the specified index.*
- virtual [Error GetInterfaceTypeFromGuid](#) ([PGRGuid](#) \*pGuid, [InterfaceType](#) \*pInterfaceType)
  - Gets the interface type associated with a [PGRGuid](#).*
- virtual [Error GetNumOfDevices](#) (unsigned int \*pNumDevices)
  - Gets the number of devices.*
- virtual [Error GetDeviceFromIndex](#) (unsigned int index, [PGRGuid](#) \*pGuid)
  - Gets the [PGRGuid](#) for a device.*
- virtual [Error ReadPhyRegister](#) ([PGRGuid](#) guid, unsigned int page, unsigned int port, unsigned int address, unsigned int \*pValue)
  - Read a phy register on the specified device.*
- virtual [Error WritePhyRegister](#) ([PGRGuid](#) guid, unsigned int page, unsigned int port, unsigned int address, unsigned int value)
  - Write a phy register on the specified device.*
- virtual [Error GetUsbLinkInfo](#) ([PGRGuid](#) guid, unsigned int \*pValue)
  - Read usb link info for the port that the specified device is connected to.*
- virtual [Error GetUsbPortStatus](#) ([PGRGuid](#) guid, unsigned int \*pValue)
  - Read usb port status for the port that the specified device is connected to.*
- virtual [Error GetTopology](#) ([TopologyNode](#) \*pNode)
  - Gets the topology information for the PC.*
- virtual [Error RegisterCallback](#) ([BusEventCallback](#) busEventCallback, [BusCallbackType](#) callbackType, void \*pParameter, [CallbackHandle](#) \*pCallbackHandle)
  - Register a callback function that will be called when the specified callback event occurs.*
- virtual [Error UnregisterCallback](#) ([CallbackHandle](#) callbackHandle)
  - Unregister a callback function.*
- virtual [Error RescanBus](#) ()
  - Force a rescan of the buses.*
- [Error IsCameraControlable](#) ([PGRGuid](#) \*pGuid, bool \*pControlable)
  - Query CCP status on camera with corresponding [PGRGuid](#).*

## Static Public Member Functions

- static [Error ForceIPAddressToCamera](#) ([MACAddress](#) macAddress, [IPAddress](#) ipAddress, [IPAddress](#) subnetMask, [IPAddress](#) defaultGateway)
  - Force the camera with the specific MAC address to the specified IP address, subnet mask and default gateway.*
- static [Error ForceAllIPAddressesAutomatically](#) ()
  - Force all cameras on the network to be assigned sequential IP addresses on the same subnet as the network adapters that they are connected to.*
- static [Error ForceAllIPAddressesAutomatically](#) (unsigned int serialNumber)
  - Force a camera on the network to be assigned an IP address on the same subnet as the network adapters that it is connected to.*
- static [Error DiscoverGigECameras](#) ([CameraInfo](#) \*gigECameras, unsigned int \*arraySize)
  - Discover all cameras connected to the network even if they reside on a different subnet.*

### 9.3.1 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.

### 9.3.2 Constructor & Destructor Documentation

#### 9.3.2.1 BusManager()

```
BusManager ( )
```

Default constructor.

#### 9.3.2.2 ~BusManager()

```
virtual ~BusManager ( ) [virtual]
```

Default destructor.

### 9.3.3 Member Function Documentation

#### 9.3.3.1 DiscoverGigECameras()

```
static Error DiscoverGigECameras (
    CameraInfo * gigECameras,
    unsigned int * arraySize ) [static]
```

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

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

##### Parameters

<i>gigECameras</i>	Pointer to an array of <a href="#">CameraInfo</a> structures.
<i>arraySize</i>	Size of the array. Number of discovered cameras is returned in the same value.

##### Returns

An [Error](#) indicating the success or failure of the function. If the error is `PGRERROR_BUFFER_TOO_SMALL` then *arraySize* will contain the minimum size needed for *gigECameras* array.

### 9.3.3.2 FireBusReset()

```
virtual Error FireBusReset (
    PGRGuid * pGuid ) [virtual]
```

Fire a bus reset.

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

#### Parameters

<i>pGuid</i>	<a href="#">PGRGuid</a> of the camera or the device to cause bus reset.
--------------	---

#### Returns

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

### 9.3.3.3 ForceAllIPAddressesAutomatically() [1/2]

```
static Error ForceAllIPAddressesAutomatically ( ) [static]
```

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

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

#### Returns

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

### 9.3.3.4 ForceAllIPAddressesAutomatically() [2/2]

```
static Error ForceAllIPAddressesAutomatically (
    unsigned int serialNumber ) [static]
```

Force a camera on the network to be assigned an IP address on the same subnet as the network adapters that it is connected to.

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

#### Returns

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



#### 9.3.3.5 ForceIPAddressToCamera()

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

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

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

##### Parameters

<i>macAddress</i>	MAC address of the camera.
<i>ipAddress</i>	IP address to set on the camera.
<i>subnetMask</i>	Subnet mask to set on the camera.
<i>defaultGateway</i>	Default gateway to set on the camera.

##### Returns

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

#### 9.3.3.6 GetCameraFromIndex()

```
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

<i>index</i>	Zero based index of camera.
<i>pGuid</i>	Unique <a href="#">PGRGuid</a> for the camera.

##### See also

[GetCameraFromSerialNumber\(\)](#)

##### Returns

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

### 9.3.3.7 GetCameraFromIPAddress()

```
virtual Error GetCameraFromIPAddress (
    IPAddress ipAddress,
    PGRGuid * pGuid ) [virtual]
```

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

#### Parameters

<i>ipAddress</i>	IP address to get GUID for.
<i>pGuid</i>	Unique <a href="#">PGRGuid</a> for the camera.

#### Returns

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

### 9.3.3.8 GetCameraFromSerialNumber()

```
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

<i>serialNumber</i>	Serial number of camera.
<i>pGuid</i>	Unique <a href="#">PGRGuid</a> for the camera.

#### See also

[GetCameraFromIndex\(\)](#)

#### Returns

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

### 9.3.3.9 GetCameraSerialNumberFromIndex()

```
virtual Error GetCameraSerialNumberFromIndex (
    unsigned int index,
    unsigned int * pSerialNumber ) [virtual]
```

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

## Parameters

<i>index</i>	Zero based index of desired camera.
<i>pSerialNumber</i>	Serial number of camera.

## Returns

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

## 9.3.3.10 GetDeviceFromIndex()

```
virtual Error GetDeviceFromIndex (
    unsigned int index,
    PGRGuid * pGuid ) [virtual]
```

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

It uniquely identifies the device specified by the index.

## Parameters

<i>index</i>	Zero based index of device.
<i>pGuid</i>	Unique <a href="#">PGRGuid</a> for the device.

## See also

[GetNumOfDevices\(\)](#)

## Returns

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

## 9.3.3.11 GetInterfaceTypeFromGuid()

```
virtual Error GetInterfaceTypeFromGuid (
    PGRGuid * pGuid,
    InterfaceType * pInterfaceType ) [virtual]
```

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

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

## Parameters

<i>pGuid</i>	The <a href="#">PGRGuid</a> to get the interface for.
<i>pInterfaceType</i>	The interface type of the <a href="#">PGRGuid</a> .

**Returns**

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

**9.3.3.12 GetNumOfCameras()**

```
virtual Error GetNumOfCameras (
    unsigned int * pNumCameras ) [virtual]
```

Gets the number of cameras attached to the PC.

**Parameters**

<i>pNumCameras</i>	The number of cameras attached.
--------------------	---------------------------------

**Returns**

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

**9.3.3.13 GetNumOfDevices()**

```
virtual Error GetNumOfDevices (
    unsigned int * pNumDevices ) [virtual]
```

Gets the number of devices.

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

**Parameters**

<i>pNumDevices</i>	The number of devices found.
--------------------	------------------------------

**Returns**

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

**9.3.3.14 GetTopology()**

```
virtual Error GetTopology (
    TopologyNode * pNode ) [virtual]
```

Gets the topology information for the PC.

## Parameters

<i>pNode</i>	<a href="#">TopologyNode</a> object that will contain the topology information.
--------------	---

## Returns

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

## 9.3.3.15 GetUsbLinkInfo()

```
virtual Error GetUsbLinkInfo (
    PGRGuid guid,
    unsigned int * pValue ) [virtual]
```

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

## Parameters

<i>guid</i>	<a href="#">PGRGuid</a> of the device to read from.
<i>pValue</i>	Value read from the card register.  Bit 15:0 = Link Error Count. Default = 0. This field returns the number of link errors detected by the port. Bit 19:16 = Rx Lane Count. Default = 0. This field that identifies the number of Receive Lanes negotiated by the port. Bit 23:20 = Tx Lane Count. Default = 0. This field that identifies the number of Transmit Lanes negotiated by the port. Bit 31:24 = Reserved.

Refer to XHCI 1.1 section 5.4.10 for Port Link Info:

[eXtensible Host Controller interface for USB xHCI](#)

## Returns

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

## 9.3.3.16 GetUsbPortStatus()

```
virtual Error GetUsbPortStatus (
    PGRGuid guid,
    unsigned int * pValue ) [virtual]
```

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

## Parameters

<i>guid</i>	<a href="#">PGRGuid</a> of the device to read from.
-------------	---

## Parameters

<i>pValue</i>	<p>Value read from the card register.</p> <p>Bit 0 = Current Connect Status. Default = 0. 1 = A device is connected to the port 0 = A device is not connected. This value reflects the current state of the port, and may not correspond directly to the event that caused the Connect Status Change (CSC) bit to be set to 1. Bit 1 = Port Enabled/Disabled. Default = 0. 1 = Enabled. 0 = Disabled. Bit 2 = Reserved. Bit 3 = Over-current Active. Default = 0. 1 = This port currently has an over-current condition. 0 = This port does not have an over-current condition. Bit 4 = Port Reset. Default = 0. 1 = Port Reset signaling is asserted. 0 = Port is not in Reset. Bit 8:5 = Port Link State. Default = RxDetect(5). This field is used to power manage the port and reflects its current link state. Bit 9 = Port Power. Default = 1. This flag reflects a port's logical, power control state. 0 = This port is in the powered-off state. 1 = This port is not in the powered-off state. Bit 13:10 = Port Speed. Default = 0. This field identifies the speed of the connected USB Device. This field is only relevant if a device is connected, in all other cases this field shall indicate Undefined Speed. 0 : Undefined speed 1-15 : Protocol Speed ID (refer to other sections) Bit 15:14 = Port Indicator Control. Default = 0. 0 = Port indicators are off. 1 = Amber. 2 = Green. 3 = Undefined. Bit 16 = Port Link State Write Strobe. Default = 0. When this bit is set to 1 on a write reference to this register, this flag enables writes to the PLS field. Bit 17 = Connect Status Change. Default = 0. 1 = Change in current connect status. 0 = No change. Bit 18 = Port Enabled/Disabled Change. Default = 0. 1 = change in PED. 0 = No change. Bit 19 = Warm Port Reset Change. Default = 0. This bit is set when Warm Reset processing on this port completes. 0 = No change. 1 = Warm Reset complete. Bit 20 = Over current change. Default = 0. This bit shall be set to a 1 when there is a 0 to 1 or 1 to 0 transition of Over-current Active. Bit 21 = Port Reset Change. Default = 0. This flag is set to 1 due to a 1 to 0 transition of Port Reset. Bit 22 = Port Link State Change. Default = 0. This flag is set to 1 due to PLS transitions (refer to document) Bit 23 = Port Config Error Change. Default = 0. This flag indicates that the port failed to configure its link partner. 0 = No change. 1 = Port Config Error detected. Bit 24 = Cold Attach Status. Default = 0. 1 = Far-end receiver terminations were detected in the disconnected state and the root hub port state machine was unable to advance to the enabled state. 0 - This flag is 0 if PP is 0 or for USB2 protocol parts. Bit 25 = Wake on Connect Enable. Default = 0. Writing this bit to a 1 enables to port to be sensitive to device connects as system wake up events. Bit 26 = Wake on Disconnect Enable. Default = 0. Writing this bit to a 1 enables the port to be sensitive to device disconnects as system wake up events. Bit 27 = Wake on Over-current Enable. Default = 0. Writing this bit to a 1 enables the port to be sensitive to over-current conditions as system wake up events. Bit 29:28 = Reserved Bit 30 = Device Removable. This flag indicates if this port has a removable device attached. 1 = Device is non-removable. 0 = Device is removable. Bit 31 = Warm Port Reset. Default = 0. This flag shall always return 0 when read. Refer to document for writing.</p>
---------------	---

Refer to XHCI 1.1 section 5.4.8 for Port Status:

**eXtensible Host Controller interface for USB xHCI**

## Returns

An **Error** indicating the success or failure of the function.

### 9.3.3.17 IsCameraControllable()

```
Error IsCameraControllable (
    PGRGuid * pGuid,
    bool * pControllable )
```

Query CCP status on camera with corresponding [PGRGuid](#).

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

#### Parameters

<i>pGuid</i>	<a href="#">PGRGuid</a> of the camera
<i>pControllable</i>	Indicates whether camera is controllable

#### Returns

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

### 9.3.3.18 ReadPhyRegister()

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

Read a phy register on the specified device.

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

#### Parameters

<i>guid</i>	<a href="#">PGRGuid</a> of the device to read from.
<i>page</i>	Page to read from.
<i>port</i>	Port to read from.
<i>address</i>	Address to read from.
<i>pValue</i>	Value read from the phy register.

#### Returns

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

**9.3.3.19 RegisterCallback()**

```
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**

<i>busEventCallback</i>	Pointer to function that will receive the callback.
<i>callbackType</i>	Type of callback to register for.
<i>pParameter</i>	Callback parameter to be passed to callback.
<i>pCallbackHandle</i>	Unique callback handle used for unregistering callback.

**See also**

[UnregisterCallback\(\)](#)

**Returns**

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

**9.3.3.20 RescanBus()**

```
virtual Error RescanBus ( ) [virtual]
```

Force a rescan of the buses.

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

**Returns**

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

**9.3.3.21 UnregisterCallback()**

```
virtual Error UnregisterCallback (
    CallbackHandle callbackHandle ) [virtual]
```

Unregister a callback function.



## Parameters

<i>callbackHandle</i>	Unique callback handle.
-----------------------	-------------------------

## See also

[RegisterCallback\(\)](#)

## Returns

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

## 9.3.3.22 WritePhyRegister()

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

Write a phy register on the specified device.

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

## Parameters

<i>guid</i>	<a href="#">PGRGuid</a> of the device to write to.
<i>page</i>	Page to write to.
<i>port</i>	Port to write to.
<i>address</i>	Address to write to.
<i>value</i>	Value to write to phy register.

## Returns

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

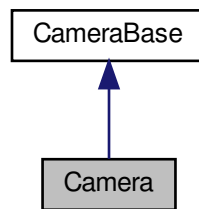
The documentation for this class was generated from the following file:

- [BusManager.h](#)

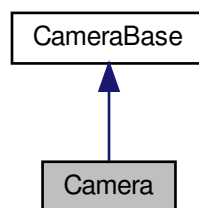
## 9.4 Camera Class Reference

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

Inheritance diagram for Camera:



Collaboration diagram for Camera:



## Public Member Functions

- [Camera](#) ()  
*Default constructor.*
- virtual [~Camera](#) ()  
*Default destructor.*
- virtual [Error Connect](#) ([PGRGuid](#) \*pGuid=NULL)  
*The following functions are inherited from [CameraBase](#).*
- virtual [Error Disconnect](#) ()  
*Disconnects the camera object from the camera.*
- virtual bool [IsConnected](#) ()  
*Checks if the camera object is connected to a physical camera specified by a GUID.*
- virtual [Error SetCallback](#) ([ImageEventCallback](#) callbackFn, const void \*pCallbackData=NULL)  
*Sets the callback data to be used on completion of image transfer.*
- virtual [Error StartCapture](#) ([ImageEventCallback](#) callbackFn=NULL, const void \*pCallbackData=NULL)  
*Starts isochronous image capture.*
- virtual [Error RetrieveBuffer](#) ([Image](#) \*pImage)  
*Retrieves the the next image object containing the next image.*
- virtual [Error StopCapture](#) ()

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

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

*Retrieves the next image event containing the next part of the image.*
- virtual [Error SetUserBuffers](#) (unsigned char \*const pMemBuffers, int size, int numBuffers)

*Specify user allocated buffers to use as image data buffers.*
- virtual [Error GetConfiguration](#) ([FC2Config](#) \*pConfig)

*Get the configuration associated with the camera object.*
- virtual [Error SetConfiguration](#) (const [FC2Config](#) \*pConfig)

*Set the configuration associated with the camera object.*
- virtual [Error GetCameraInfo](#) ([CameraInfo](#) \*pCameraInfo)

*Retrieves information from the camera such as serial number, model name and other camera information.*
- virtual [Error GetPropertyInfo](#) ([PropertyInfo](#) \*pPropInfo)

*Retrieves information about the specified camera property.*
- virtual [Error GetProperty](#) ([Property](#) \*pProp)

*Reads the settings for the specified property from the camera.*
- virtual [Error SetProperty](#) (const [Property](#) \*pProp, bool broadcast=false)

*Writes the settings for the specified property to the camera.*
- virtual [Error GetGPIOPinDirection](#) (unsigned int pin, unsigned int \*pDirection)

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

*Set the GPIO pin direction for the specified pin.*
- virtual [Error GetTriggerModelInfo](#) ([TriggerModelInfo](#) \*pTriggerModelInfo)

*Retrieve trigger information from the camera.*
- virtual [Error GetTriggerMode](#) ([TriggerMode](#) \*pTriggerMode)

*Retrieve current trigger settings from the camera.*
- virtual [Error SetTriggerMode](#) (const [TriggerMode](#) \*pTriggerMode, bool broadcast=false)

*Set the specified trigger settings to the camera.*
- virtual [Error FireSoftwareTrigger](#) (bool broadcast=false)

*Fire the software trigger according to the DCAM specifications.*
- virtual [Error GetTriggerDelayInfo](#) ([TriggerDelayInfo](#) \*pTriggerDelayInfo)

*Retrieve trigger delay information from the camera.*
- virtual [Error GetTriggerDelay](#) ([TriggerDelay](#) \*pTriggerDelay)

*Retrieve current trigger delay settings from the camera.*
- virtual [Error SetTriggerDelay](#) (const [TriggerDelay](#) \*pTriggerDelay, bool broadcast=false)

*Set the specified trigger delay settings to the camera.*
- virtual [Error GetStrobeInfo](#) ([StrobeInfo](#) \*pStrobeInfo)

*Retrieve strobe information from the camera.*
- virtual [Error GetStrobe](#) ([StrobeControl](#) \*pStrobeControl)

*Retrieve current strobe settings from the camera.*
- virtual [Error SetStrobe](#) (const [StrobeControl](#) \*pStrobeControl, bool broadcast=false)

*Set current strobe settings to the camera.*
- virtual [Error GetLUTInfo](#) ([LUTData](#) \*pData)

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

*Query the read/write status of a single LUT bank.*
- virtual [Error GetActiveLUTBank](#) (unsigned int \*pActiveBank)

*Get the LUT bank that is currently being used.*
- virtual [Error SetActiveLUTBank](#) (unsigned int activeBank)

*Set the LUT bank that will be used.*
- virtual [Error EnableLUT](#) (bool on)

*Enable or disable LUT functionality on the camera.*

- virtual [Error GetLUTChannel](#) (unsigned int bank, unsigned int channel, unsigned int sizeEntries, unsigned int \*pEntries)  
*Get the LUT channel settings from the camera.*
- virtual [Error SetLUTChannel](#) (unsigned int bank, unsigned int channel, unsigned int sizeEntries, const unsigned int \*pEntries)  
*Set the LUT channel settings to the camera.*
- virtual [Error GetMemoryChannel](#) (unsigned int \*pCurrentChannel)  
*Retrieve the current memory channel from the camera.*
- virtual [Error SaveToMemoryChannel](#) (unsigned int channel)  
*Save the current settings to the specified current memory channel.*
- virtual [Error RestoreFromMemoryChannel](#) (unsigned int channel)  
*Restore the specified current memory channel.*
- virtual [Error GetMemoryChannelInfo](#) (unsigned int \*pNumChannels)  
*Query the camera for memory channel support.*
- virtual [Error GetEmbeddedImageInfo](#) ([EmbeddedImageInfo](#) \*pInfo)  
*Get the current status of the embedded image information register, as well as the availability of each embedded property.*
- virtual [Error SetEmbeddedImageInfo](#) ([EmbeddedImageInfo](#) \*pInfo)  
*Sets the on/off values of the embedded image information structure to the camera.*
- virtual [Error WriteRegister](#) (unsigned int address, unsigned int value, bool broadcast=false)  
*Write to the specified register on the camera.*
- virtual [Error ReadRegister](#) (unsigned int address, unsigned int \*pValue)  
*Read the specified register from the camera.*
- virtual [Error WriteRegisterBlock](#) (unsigned short addressHigh, unsigned int addressLow, const unsigned int \*pBuffer, unsigned int length)  
*Write to the specified register block on the camera.*
- virtual [Error ReadRegisterBlock](#) (unsigned short addressHigh, unsigned int addressLow, unsigned int \*pBuffer, unsigned int length)  
*Read from the specified register block on the camera.*
- virtual [Error GetCycleTime](#) ([TimeStamp](#) \*timeStamp)  
*Returns a Timestamp struct containing 1394 CYCLE\_TIME information.*
- virtual [Error GetStats](#) ([CameraStats](#) \*pStats)
- virtual [Error ResetStats](#) ()
- virtual [Error RegisterEvent](#) ([EventOptions](#) \*pOpts)
- virtual [Error DeregisterEvent](#) ([EventOptions](#) \*pOpts)
- virtual [Error RegisterAllEvents](#) ([EventOptions](#) \*pOpts)
- virtual [Error DeregisterAllEvents](#) (void)

## Static Public Member Functions

- static [Error StartSyncCapture](#) (unsigned int numCameras, const [Camera](#) \*\*ppCameras, const [ImageEventCallback](#) \*pCallbackFns=NULL, const void \*\*pCallbackDataArray=NULL)
- static const char \* [GetRegisterString](#) (unsigned int registerVal)

## DCAM Formats

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

They are only used for firewire and usb2 cameras.

- 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.*

## Additional Inherited Members

### 9.4.1 Detailed Description

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

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

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

### 9.4.2 Constructor & Destructor Documentation

#### 9.4.2.1 Camera()

`Camera ( )`

Default constructor.

#### 9.4.2.2 ~Camera()

`virtual ~Camera ( ) [virtual]`

Default destructor.

### 9.4.3 Member Function Documentation

#### 9.4.3.1 Connect()

```
virtual Error Connect (
    PGRGuid * pGuid = NULL ) [virtual]
```

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

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

Implements [CameraBase](#).

#### 9.4.3.2 DeregisterAllEvents()

```
virtual Error DeregisterAllEvents (
    void ) [virtual]
```

Implements [CameraBase](#).

#### 9.4.3.3 DeregisterEvent()

```
virtual Error DeregisterEvent (
    EventOptions * pOpts ) [virtual]
```

Implements [CameraBase](#).

#### 9.4.3.4 Disconnect()

```
virtual Error Disconnect ( ) [virtual]
```

Disconnects the camera object from the camera.

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

See also

[Connect\(\)](#)

Returns

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

Implements [CameraBase](#).

#### 9.4.3.5 EnableLUT()

```
virtual Error EnableLUT (
    bool on ) [virtual]
```

Enable or disable LUT functionality on the camera.

Parameters

<i>on</i>	Whether to enable or disable LUT.
-----------	-----------------------------------

See also

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

Returns

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

Implements [CameraBase](#).

#### 9.4.3.6 FireSoftwareTrigger()

```
virtual Error FireSoftwareTrigger (
    bool broadcast = false ) [virtual]
```

Fire the software trigger according to the DCAM specifications.

**Parameters**

<i>broadcast</i>	Whether the action should be broadcast.
------------------	---

**Returns**

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

Implements [CameraBase](#).

**9.4.3.7 GetActiveLUTBank()**

```
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**

<i>pActiveBank</i>	The currently active bank.
--------------------	----------------------------

**Returns**

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

Implements [CameraBase](#).

**9.4.3.8 GetCameraInfo()**

```
virtual Error GetCameraInfo (
    CameraInfo * pCameraInfo ) [virtual]
```

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

**Parameters**

<i>pCameraInfo</i>	Pointer to the camera information structure to be filled.
--------------------	---

**Returns**

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

Implements [CameraBase](#).



#### 9.4.3.9 GetConfiguration()

```
virtual Error GetConfiguration (
    FC2Config * pConfig ) [virtual]
```

Get the configuration associated with the camera object.

##### Parameters

<i>pConfig</i>	Pointer to the configuration structure to be filled.
----------------	--

##### See also

[SetConfiguration\(\)](#)

##### Returns

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

Implements [CameraBase](#).

#### 9.4.3.10 GetCycleTime()

```
virtual Error GetCycleTime (
    TimeStamp * timeStamp ) [virtual]
```

Returns a Timestamp struct containing 1394 CYCLE\_TIME information.

##### Parameters

<i>registerVal</i>	The register value to query.
--------------------	------------------------------

##### Returns

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

Implements [CameraBase](#).

#### 9.4.3.11 GetEmbeddedImageInfo()

```
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**

<i>pInfo</i>	Structure to be filled.
--------------	-------------------------

**See also**

[SetEmbeddedImageInfo\(\)](#)

**Returns**

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

Implements [CameraBase](#).

**9.4.3.12 GetFormat7Configuration()**

```
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**

<i>pImageSettings</i>	Current image settings.
<i>pPacketSize</i>	Current packet size.
<i>pPercentage</i>	Current packet size as a percentage.

**See also**

[GetFormat7Info\(\)](#)  
[ValidateFormat7Settings\(\)](#)  
[SetFormat7Configuration\(\)](#)  
[GetVideoModeAndFrameRate\(\)](#)

**Returns**

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

#### 9.4.3.13 GetFormat7Info()

```
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

<i>pInfo</i>	Structure to be filled with the capabilities of the specified mode and the current state in the specified mode.
<i>pSupported</i>	Whether the specified mode is supported.

##### See also

[ValidateFormat7Settings\(\)](#)  
[GetFormat7Configuration\(\)](#)  
[SetFormat7Configuration\(\)](#)

##### Returns

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

#### 9.4.3.14 GetGPIOPinDirection()

```
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

<i>pin</i>	Pin to get the direction for.
<i>pDirection</i>	Direction of the pin. 0 for input, 1 for output.

##### See also

[SetGPIOPinDirection\(\)](#)

**Returns**

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

Implements [CameraBase](#).

**9.4.3.15 GetLUTBankInfo()**

```
virtual Error GetLUTBankInfo (
    unsigned int bank,
    bool * pReadSupported,
    bool * pWriteSupported ) [virtual]
```

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

**Parameters**

<i>bank</i>	The bank to query.
<i>pReadSupported</i>	Whether reading from the bank is supported.
<i>pWriteSupported</i>	Whether writing to the bank is supported.

**Returns**

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

Implements [CameraBase](#).

**9.4.3.16 GetLUTChannel()**

```
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**

<i>bank</i>	Bank to retrieve.
<i>channel</i>	Channel to retrieve.
<i>sizeEntries</i>	Number of entries in LUT table to read.
<i>pEntries</i>	Array to store LUT entries.

## See also

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

## Returns

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

Implements [CameraBase](#).

## 9.4.3.17 GetLUTInfo()

```
virtual Error GetLUTInfo (
    LUTData * pData ) [virtual]
```

Query if LUT support is available on the camera.

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

## Parameters

<i>pData</i>	The LUT structure to be filled.
--------------	---------------------------------

## See also

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

## Returns

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

Implements [CameraBase](#).

## 9.4.3.18 GetMemoryChannel()

```
virtual Error GetMemoryChannel (
    unsigned int * pCurrentChannel ) [virtual]
```

Retrieve the current memory channel from the camera.

**Parameters**

<i>pCurrentChannel</i>	Current memory channel.
------------------------	-------------------------

**See also**

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

**Returns**

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

Implements [CameraBase](#).

**9.4.3.19 GetMemoryChannelInfo()**

```
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**

<i>pNumChannels</i>	Number of memory channels supported.
---------------------	--------------------------------------

**See also**

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

**Returns**

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

Implements [CameraBase](#).

**9.4.3.20 GetProperty()**

```
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

<i>pProp</i>	Pointer to the <a href="#">Property</a> structure to be filled.
--------------	---

## See also

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

## Returns

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

Implements [CameraBase](#).

9.4.3.21 [GetPropertyInfo\(\)](#)

```
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

<i>pPropInfo</i>	Pointer to the <a href="#">PropertyInfo</a> structure to be filled.
------------------	---

## See also

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

## Returns

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

Implements [CameraBase](#).

9.4.3.22 [GetRegisterString\(\)](#)

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

#### 9.4.3.23 GetStats()

```
virtual Error GetStats (
    CameraStats * pStats ) [virtual]
```

Implements [CameraBase](#).

#### 9.4.3.24 GetStrobe()

```
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

<i>pStrobeControl</i>	Structure to receive strobe settings.
-----------------------	---------------------------------------

##### See also

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

##### Returns

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

Implements [CameraBase](#).

#### 9.4.3.25 GetStrobeInfo()

```
virtual Error GetStrobeInfo (
    StrobeInfo * pStrobeInfo ) [virtual]
```

Retrieve strobe information from the camera.

##### Parameters

<i>pStrobeInfo</i>	Structure to receive strobe information.
--------------------	--



See also

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

Returns

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

Implements [CameraBase](#).

#### 9.4.3.26 GetTriggerDelay()

```
virtual Error GetTriggerDelay (
    TriggerDelay * pTriggerDelay ) [virtual]
```

Retrieve current trigger delay settings from the camera.

Parameters

<i>pTriggerDelay</i>	Structure to receive trigger delay settings.
----------------------	--

See also

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

Returns

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

Implements [CameraBase](#).

#### 9.4.3.27 GetTriggerDelayInfo()

```
virtual Error GetTriggerDelayInfo (
    TriggerDelayInfo * pTriggerDelayInfo ) [virtual]
```

Retrieve trigger delay information from the camera.

Parameters

<i>pTriggerDelayInfo</i>	Structure to receive trigger delay information.
--------------------------	---

**See also**

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

**Returns**

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

Implements [CameraBase](#).

**9.4.3.28 GetTriggerMode()**

```
virtual Error GetTriggerMode (
    TriggerMode * pTriggerMode ) [virtual]
```

Retrieve current trigger settings from the camera.

**Parameters**

<i>pTriggerMode</i>	Structure to receive trigger mode settings.
---------------------	---

**See also**

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

**Returns**

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

Implements [CameraBase](#).

**9.4.3.29 GetTriggerModelInfo()**

```
virtual Error GetTriggerModelInfo (
    TriggerModeInfo * pTriggerModeInfo ) [virtual]
```

Retrieve trigger information from the camera.

## Parameters

<i>pTriggerModeInfo</i>	Structure to receive trigger information.
-------------------------	---

## See also

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

## Returns

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

Implements [CameraBase](#).

## 9.4.3.30 GetVideoModeAndFrameRate()

```
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 FRAME↵RATE\_FORMAT7.

## Parameters

<i>pVideoMode</i>	Current video mode.
<i>pFrameRate</i>	Current frame rate.

## See also

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

## Returns

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

#### 9.4.3.31 GetVideoModeAndFrameRateInfo()

```
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

<i>videoMode</i>	Video mode to check.
<i>frameRate</i>	Frame rate to check.
<i>pSupported</i>	Whether the video mode and frame rate is supported.

##### See also

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

##### Returns

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

#### 9.4.3.32 IsConnected()

```
virtual bool IsConnected ( ) [virtual]
```

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

##### See also

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

##### Returns

Whether [Connect\(\)](#) was called on the camera object.

Implements [CameraBase](#).

#### 9.4.3.33 ReadRegister()

```
virtual Error ReadRegister (
    unsigned int address,
    unsigned int * pValue ) [virtual]
```

Read the specified register from the camera.

## Parameters

<i>address</i>	DCAM address to be read from.
<i>pValue</i>	The value that is read.

## See also

[WriteRegister\(\)](#)

## Returns

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

Implements [CameraBase](#).

#### 9.4.3.34 ReadRegisterBlock()

```
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

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

## See also

[WriteRegisterBlock\(\)](#)

## Returns

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

Implements [CameraBase](#).

#### 9.4.3.35 RegisterAllEvents()

```
virtual Error RegisterAllEvents (
    EventOptions * pOpts ) [virtual]
```

Implements [CameraBase](#).

**9.4.3.36 RegisterEvent()**

```
virtual Error RegisterEvent (
    EventOptions * pOpts ) [virtual]
```

Implements [CameraBase](#).

**9.4.3.37 ResetStats()**

```
virtual Error ResetStats ( ) [virtual]
```

Implements [CameraBase](#).

**9.4.3.38 RestoreFromMemoryChannel()**

```
virtual Error RestoreFromMemoryChannel (
    unsigned int channel ) [virtual]
```

Restore the specified current memory channel.

**Parameters**

<i>channel</i>	Memory channel to restore from.
----------------	---------------------------------

**See also**

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

**Returns**

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

Implements [CameraBase](#).

**9.4.3.39 RetrieveBuffer()**

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

## Parameters

<i>pImage</i>	Pointer to <a href="#">Image</a> object to store image data.
---------------	--

## See also

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

## Returns

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

Implements [CameraBase](#).

## 9.4.3.40 SaveToMemoryChannel()

```
virtual Error SaveToMemoryChannel (
    unsigned int channel ) [virtual]
```

Save the current settings to the specified current memory channel.

## Parameters

<i>channel</i>	Memory channel to save to.
----------------	----------------------------

## See also

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

## Returns

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

Implements [CameraBase](#).

## 9.4.3.41 SetActiveLUTBank()

```
virtual Error SetActiveLUTBank (
    unsigned int activeBank ) [virtual]
```

Set the LUT bank that will be used.

## Parameters

<i>activeBank</i>	The bank to be set as active.
-------------------	-------------------------------

## Returns

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

Implements [CameraBase](#).

## 9.4.3.42 SetCallback()

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

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

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

## Parameters

<i>callbackFn</i>	A function to be called when a new image is received.
<i>pCallbackData</i>	A pointer to data that can be passed to the callback function.

## See also

[StartCapture\(\)](#)

## Returns

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

Implements [CameraBase](#).

## 9.4.3.43 SetConfiguration()

```
virtual Error SetConfiguration (
    const FC2Config * pConfig ) [virtual]
```

Set the configuration associated with the camera object.

## Parameters

<i>pConfig</i>	Pointer to the configuration structure to be used.
----------------	--



See also

[GetConfiguration\(\)](#)

Returns

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

Implements [CameraBase](#).

#### 9.4.3.44 SetEmbeddedImageInfo()

```
virtual Error SetEmbeddedImageInfo (
    EmbeddedImageInfo * pInfo ) [virtual]
```

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

Parameters

<i>pInfo</i>	Structure to be used.
--------------	-----------------------

See also

[GetEmbeddedImageInfo\(\)](#)

Returns

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

Implements [CameraBase](#).

#### 9.4.3.45 SetFormat7Configuration() [1/2]

```
virtual Error SetFormat7Configuration (
    const Format7ImageSettings * pImageSettings,
    unsigned int packetSize ) [virtual]
```

Set the current Format7 configuration to the camera.

Parameters

<i>pImageSettings</i>	<a href="#">Image</a> settings to be written to the camera.
<i>packetSize</i>	Packet size to be written to the camera.

**See also**

[GetFormat7Info\(\)](#)  
[ValidateFormat7Settings\(\)](#)  
[GetFormat7Configuration\(\)](#)

**Returns**

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

**9.4.3.46 SetFormat7Configuration()** [2/2]

```
virtual Error SetFormat7Configuration (
    const Format7ImageSettings * pImageSettings,
    float percentSpeed ) [virtual]
```

Set the current Format7 configuration to the camera.

**Parameters**

<i>pImageSettings</i>	<a href="#">Image</a> settings to be written to the camera.
<i>percentSpeed</i>	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.

**9.4.3.47 SetGPIOPinDirection()**

```
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

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

## See also

[GetGPIOPinDirection\(\)](#)

## Returns

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

Implements [CameraBase](#).

## 9.4.3.48 SetLUTChannel()

```
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

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

## See also

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

## Returns

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

Implements [CameraBase](#).

#### 9.4.3.49 SetProperty()

```
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. Use [GetPropertyInfo\(\)](#) to query which options are available for a specific property.

##### Parameters

<i>pProp</i>	Pointer to the <a href="#">Property</a> structure to be used.
<i>broadcast</i>	Whether the action should be broadcast.

##### See also

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

##### Returns

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

Implements [CameraBase](#).

#### 9.4.3.50 SetStrobe()

```
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

<i>pStrobeControl</i>	Structure providing strobe settings.
<i>broadcast</i>	Whether the action should be broadcast.

##### See also

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

**Returns**

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

Implements [CameraBase](#).

**9.4.3.51 SetTriggerDelay()**

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

Set the specified trigger delay settings to the camera.

**Parameters**

<i>pTriggerDelay</i>	Structure providing trigger delay settings.
<i>broadcast</i>	Whether the action should be broadcast.

**See also**

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

**Returns**

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

Implements [CameraBase](#).

**9.4.3.52 SetTriggerMode()**

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

Set the specified trigger settings to the camera.

**Parameters**

<i>pTriggerMode</i>	Structure providing trigger mode settings.
<i>broadcast</i>	Whether the action should be broadcast.

**See also**

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

**Returns**

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

Implements [CameraBase](#).

**9.4.3.53 SetUserBuffers()**

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

Specify user allocated buffers to use as image data buffers.

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

**Parameters**

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

**See also**

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

**Returns**

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

Implements [CameraBase](#).

## 9.4.3.54 SetVideoModeAndFrameRate()

```
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

<i>videoMode</i>	Video mode to set to camera.
<i>frameRate</i>	Frame rate to set to camera.

## See also

[GetVideoModeAndFrameRateInfo\(\)](#)

[GetVideoModeAndFrameRate\(\)](#)

## Returns

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

## 9.4.3.55 StartCapture()

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

Starts isochronous image capture.

It will use either the current video mode or the most recently set video mode of the camera. The optional callback function parameter is called on completion of image transfer. When a callback function is specified, the grab mode will determine how images are delivered. If the grab mode has not been set, or has been set to DROP\_FRAMES the default behavior is to requeue images for DMA if they have not been delivered by the time the next image transfer completes. If BUFFER\_FRAMES is specified, the next image in the sequence will be delivered. Note that for the BUFFER\_FRAMES case, if delivery does not keep up with the DMA process, images will be lost. The default behavior is to perform DROP\_FRAMES image delivery Alternatively, the callback parameter can be set to NULL and [RetrieveBuffer\(\)](#) can be called as a blocking call to get the image data.

## Parameters

<i>callbackFn</i>	A function to be called when a new image is received.
<i>pCallbackData</i>	A pointer to data that can be passed to the callback function.

**See also**

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

**Returns**

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

Implements [CameraBase](#).

**9.4.3.56 StartSyncCapture()**

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

**9.4.3.57 StopCapture()**

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

Stops isochronous image transfer and cleans up all associated resources.

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

**See also**

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

**Returns**

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

Implements [CameraBase](#).

**9.4.3.58 ValidateFormat7Settings()**

```
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

<i>pImageSettings</i>	Structure containing the image settings.
<i>pSettingsAreValid</i>	Whether the settings are valid.
<i>pPacketInfo</i>	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.

## 9.4.3.59 WaitForBufferEvent()

```
virtual Error WaitForBufferEvent (
    Image * pImage,
    unsigned int eventNumber ) [virtual]
```

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

## Parameters

<i>pImage</i>	Pointer to <a href="#">Image</a> object to store image data.
<i>eventNumber</i>	The event number to wait for.

## See also

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

## Returns

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

Implements [CameraBase](#).

## 9.4.3.60 WriteRegister()

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

Write to the specified register on the camera.

**Parameters**

<i>address</i>	DCAM address to be written to.
<i>value</i>	The value to be written.
<i>broadcast</i>	Whether the action should be broadcast.

**See also**

[ReadRegister\(\)](#)

**Returns**

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

Implements [CameraBase](#).

**9.4.3.61 WriteRegisterBlock()**

```
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**

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

**See also**

[ReadRegisterBlock\(\)](#)

**Returns**

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

Implements [CameraBase](#).

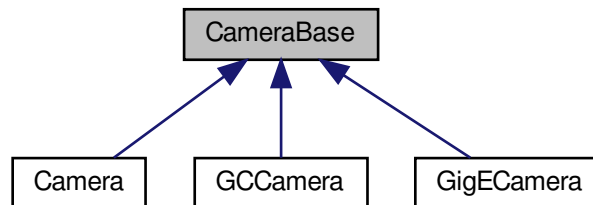
The documentation for this class was generated from the following file:

- [Camera.h](#)

## 9.5 CameraBase Class Reference

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

Inheritance diagram for CameraBase:



### Public Member Functions

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

### Protected Attributes

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

### Connection and Image Retrieval

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

- virtual [Error Connect](#) ([PGRGuid](#) \*pGuid=NULL)=0  
*Connects the camera object to the camera specified by the GUID.*
- virtual [Error Disconnect](#) ()=0  
*Disconnects the camera object from the camera.*
- virtual bool [IsConnected](#) ()=0  
*Checks if the camera object is connected to a physical camera specified by a GUID.*
- virtual [Error SetCallback](#) ([ImageEventCallback](#) callbackFn, const void \*pCallbackData=NULL)=0  
*Sets the callback data to be used on completion of image transfer.*
- virtual [Error StartCapture](#) ([ImageEventCallback](#) callbackFn=NULL, const void \*pCallbackData=NULL)=0  
*Starts isochronous image capture.*
- virtual [Error RetrieveBuffer](#) ([Image](#) \*pImage)=0  
*Retrieves the the next image object containing the next image.*
- virtual [Error StopCapture](#) ()=0  
*Stops isochronous image transfer and cleans up all associated resources.*

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

## Information and Properties

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

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

## General Purpose Input / Output

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

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

## Trigger

These functions deal with trigger control on the camera.

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

## Strobe

These functions deal with strobe control on the camera.

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

## Look Up Table

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

- virtual [Error GetLUTInfo](#) ([LUTData](#) \*pData)=0  
*Query if LUT support is available on the camera.*
- virtual [Error GetLUTBankInfo](#) (unsigned int bank, bool \*pReadSupported, bool \*pWriteSupported)=0  
*Query the read/write status of a single LUT bank.*
- virtual [Error GetActiveLUTBank](#) (unsigned int \*pActiveBank)=0  
*Get the LUT bank that is currently being used.*
- virtual [Error SetActiveLUTBank](#) (unsigned int activeBank)=0  
*Set the LUT bank that will be used.*
- virtual [Error EnableLUT](#) (bool on)=0  
*Enable or disable LUT functionality on the camera.*
- virtual [Error GetLUTChannel](#) (unsigned int bank, unsigned int channel, unsigned int sizeEntries, unsigned int \*pEntries)=0  
*Get the LUT channel settings from the camera.*
- virtual [Error SetLUTChannel](#) (unsigned int bank, unsigned int channel, unsigned int sizeEntries, const unsigned int \*pEntries)=0  
*Set the LUT channel settings to the camera.*

## Memory Channels

These functions deal with memory channel control on the camera.

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

## Embedded Image Information

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

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

## Register Operation

These functions deal with register operation on the camera.

- virtual [Error WriteRegister](#) (unsigned int address, unsigned int value, bool broadcast=false)=0  
*Write to the specified register on the camera.*
- virtual [Error ReadRegister](#) (unsigned int address, unsigned int \*pValue)=0  
*Read the specified register from the camera.*
- virtual [Error WriteRegisterBlock](#) (unsigned short addressHigh, unsigned int addressLow, const unsigned int \*pBuffer, unsigned int length)=0  
*Write to the specified register block on the camera.*
- virtual [Error ReadRegisterBlock](#) (unsigned short addressHigh, unsigned int addressLow, unsigned int \*pBuffer, unsigned int length)=0  
*Read from the specified register block on the camera.*
- virtual [Error GetCycleTime](#) ([TimeStamp](#) \*timeStamp)=0  
*Returns a Timestamp struct containing 1394 CYCLE\_TIME information.*
- virtual [Error GetStats](#) ([CameraStats](#) \*pStats)=0
- virtual [Error ResetStats](#) ()=0
- virtual [Error RegisterEvent](#) ([EventOptions](#) \*pOpts)=0
- virtual [Error DeregisterEvent](#) ([EventOptions](#) \*pOpts)=0
- virtual [Error RegisterAllEvents](#) ([EventOptions](#) \*pOpts)=0
- virtual [Error DeregisterAllEvents](#) (void)=0
- static const char \* [GetRegisterString](#) (unsigned int registerVal)  
*Returns a text representation of the register value.*

### 9.5.1 Detailed Description

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

### 9.5.2 Constructor & Destructor Documentation

#### 9.5.2.1 CameraBase()

```
CameraBase ( ) [inline]
```

Default constructor.

### 9.5.2.2 ~CameraBase()

```
virtual ~CameraBase ( ) [inline], [virtual]
```

Default destructor.

## 9.5.3 Member Function Documentation

### 9.5.3.1 Connect()

```
virtual Error Connect (
    PGRGuid * pGuid = NULL ) [pure virtual]
```

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

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

#### Parameters

<i>pGuid</i>	The unique identifier for a specific camera on the PC.
--------------	--

#### See also

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

#### Returns

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

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

### 9.5.3.2 DeregisterAllEvents()

```
virtual Error DeregisterAllEvents (
    void ) [pure virtual]
```

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

#### 9.5.3.3 DeregisterEvent()

```
virtual Error DeregisterEvent (
    EventOptions * pOpts ) [pure virtual]
```

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

#### 9.5.3.4 Disconnect()

```
virtual Error Disconnect ( ) [pure virtual]
```

Disconnects the camera object from the camera.

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

See also

[Connect\(\)](#)

Returns

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

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

#### 9.5.3.5 EnableLUT()

```
virtual Error EnableLUT (
    bool on ) [pure virtual]
```

Enable or disable LUT functionality on the camera.

Parameters

<i>on</i>	Whether to enable or disable LUT.
-----------	-----------------------------------

See also

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



**Returns**

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

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

**9.5.3.6 FireSoftwareTrigger()**

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

Fire the software trigger according to the DCAM specifications.

**Parameters**

<i>broadcast</i>	Whether the action should be broadcast.
------------------	---

**Returns**

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

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

**9.5.3.7 GetActiveLUTBank()**

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

Get the LUT bank that is currently being used.

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

**Parameters**

<i>pActiveBank</i>	The currently active bank.
--------------------	----------------------------

**Returns**

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

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

#### 9.5.3.8 GetCameraInfo()

```
virtual Error GetCameraInfo (
    CameraInfo * pCameraInfo ) [pure virtual]
```

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

##### Parameters

<i>pCameraInfo</i>	Pointer to the camera information structure to be filled.
--------------------	---

##### Returns

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

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

#### 9.5.3.9 GetConfiguration()

```
virtual Error GetConfiguration (
    FC2Config * pConfig ) [pure virtual]
```

Get the configuration associated with the camera object.

##### Parameters

<i>pConfig</i>	Pointer to the configuration structure to be filled.
----------------	--

##### See also

[SetConfiguration\(\)](#)

##### Returns

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

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

#### 9.5.3.10 GetCycleTime()

```
virtual Error GetCycleTime (
    TimeStamp * timeStamp ) [pure virtual]
```

Returns a Timestamp struct containing 1394 CYCLE\_TIME information.

## Parameters

<i>registerVal</i>	The register value to query.
--------------------	------------------------------

## Returns

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

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

## 9.5.3.11 GetEmbeddedImageInfo()

```
virtual Error GetEmbeddedImageInfo (
    EmbeddedImageInfo * pInfo ) [pure virtual]
```

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

## Parameters

<i>pInfo</i>	Structure to be filled.
--------------	-------------------------

## See also

[SetEmbeddedImageInfo\(\)](#)

## Returns

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

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

## 9.5.3.12 GetGPIOPinDirection()

```
virtual Error GetGPIOPinDirection (
    unsigned int pin,
    unsigned int * pDirection ) [pure virtual]
```

Get the GPIO pin direction for the specified pin.

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

## Parameters

<i>pin</i>	Pin to get the direction for.
<i>pDirection</i>	Direction of the pin. 0 for input, 1 for output.

See also

[SetGPIOPinDirection\(\)](#)

Returns

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

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

#### 9.5.3.13 GetLUTBankInfo()

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

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

Parameters

<i>bank</i>	The bank to query.
<i>pReadSupported</i>	Whether reading from the bank is supported.
<i>pWriteSupported</i>	Whether writing to the bank is supported.

Returns

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

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

#### 9.5.3.14 GetLUTChannel()

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

Get the LUT channel settings from the camera.

Parameters

<i>bank</i>	Bank to retrieve.
<i>channel</i>	Channel to retrieve.
<i>sizeEntries</i>	Number of entries in LUT table to read.
<i>pEntries</i>	Array to store LUT entries.

## See also

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

## Returns

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

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

## 9.5.3.15 GetLUTInfo()

```
virtual Error GetLUTInfo (
    LUTData * pData ) [pure virtual]
```

Query if LUT support is available on the camera.

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

## Parameters

<i>pData</i>	The LUT structure to be filled.
--------------	---------------------------------

## See also

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

## Returns

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

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

## 9.5.3.16 GetMemoryChannel()

```
virtual Error GetMemoryChannel (
    unsigned int * pCurrentChannel ) [pure virtual]
```

Retrieve the current memory channel from the camera.

**Parameters**

<i>pCurrentChannel</i>	Current memory channel.
------------------------	-------------------------

**See also**

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

**Returns**

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

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

**9.5.3.17 GetMemoryChannelInfo()**

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

Query the camera for memory channel support.

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

**Parameters**

<i>pNumChannels</i>	Number of memory channels supported.
---------------------	--------------------------------------

**See also**

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

**Returns**

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

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

**9.5.3.18 GetProperty()**

```
virtual Error GetProperty (
    Property * pProp ) [pure virtual]
```

Reads the settings for the specified property from the camera.

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

## Parameters

<i>pProp</i>	Pointer to the <a href="#">Property</a> structure to be filled.
--------------	---

## See also

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

## Returns

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

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

## 9.5.3.19 GetPropertyInfo()

```
virtual Error GetPropertyInfo (
    PropertyInfo * pPropInfo ) [pure virtual]
```

Retrieves information about the specified camera property.

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

## Parameters

<i>pPropInfo</i>	Pointer to the <a href="#">PropertyInfo</a> structure to be filled.
------------------	---

## See also

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

## Returns

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

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

## 9.5.3.20 GetRegisterString()

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

Returns a text representation of the register value.

**Parameters**

<i>registerVal</i>	The register value to query.
--------------------	------------------------------

**Returns**

The text representation of the register.

**9.5.3.21 GetStats()**

```
virtual Error GetStats (
    CameraStats * pStats ) [pure virtual]
```

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

**9.5.3.22 GetStrobe()**

```
virtual Error GetStrobe (
    StrobeControl * pStrobeControl ) [pure virtual]
```

Retrieve current strobe settings from the camera.

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

**Parameters**

<i>pStrobeControl</i>	Structure to receive strobe settings.
-----------------------	---------------------------------------

**See also**

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

**Returns**

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

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

**9.5.3.23 GetStrobeInfo()**

```
virtual Error GetStrobeInfo (
    StrobeInfo * pStrobeInfo ) [pure virtual]
```

Retrieve strobe information from the camera.



## Parameters

<i>pStrobeInfo</i>	Structure to receive strobe information.
--------------------	--

## See also

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

## Returns

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

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

## 9.5.3.24 GetTriggerDelay()

```
virtual Error GetTriggerDelay (
    TriggerDelay * pTriggerDelay ) [pure virtual]
```

Retrieve current trigger delay settings from the camera.

## Parameters

<i>pTriggerDelay</i>	Structure to receive trigger delay settings.
----------------------	--

## See also

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

## Returns

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

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

## 9.5.3.25 GetTriggerDelayInfo()

```
virtual Error GetTriggerDelayInfo (
    TriggerDelayInfo * pTriggerDelayInfo ) [pure virtual]
```

Retrieve trigger delay information from the camera.

## Parameters

<i>pTriggerDelayInfo</i>	Structure to receive trigger delay information.
--------------------------	---

## See also

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

## Returns

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

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

## 9.5.3.26 GetTriggerMode()

```
virtual Error GetTriggerMode (
    TriggerMode * pTriggerMode ) [pure virtual]
```

Retrieve current trigger settings from the camera.

## Parameters

<i>pTriggerMode</i>	Structure to receive trigger mode settings.
---------------------	---

## See also

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

## Returns

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

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

## 9.5.3.27 GetTriggerModelInfo()

```
virtual Error GetTriggerModelInfo (
    TriggerModeInfo * pTriggerModeInfo ) [pure virtual]
```

Retrieve trigger information from the camera.

## Parameters

<i>pTriggerModeInfo</i>	Structure to receive trigger information.
-------------------------	---

## See also

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

## Returns

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

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

## 9.5.3.28 IsConnected()

```
virtual bool IsConnected ( ) [pure virtual]
```

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

## See also

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

## Returns

Whether [Connect\(\)](#) was called on the camera object.

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

## 9.5.3.29 ReadRegister()

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

Read the specified register from the camera.

**Parameters**

<i>address</i>	DCAM address to be read from.
<i>pValue</i>	The value that is read.

**See also**

[WriteRegister\(\)](#)

**Returns**

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

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

**9.5.3.30 ReadRegisterBlock()**

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

Read from the specified register block on the camera.

**Parameters**

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

**See also**

[WriteRegisterBlock\(\)](#)

**Returns**

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

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

**9.5.3.31 RegisterAllEvents()**

```
virtual Error RegisterAllEvents (
    EventOptions * pOpts ) [pure virtual]
```

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

#### 9.5.3.32 RegisterEvent()

```
virtual Error RegisterEvent (
    EventOptions * pOpts ) [pure virtual]
```

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

#### 9.5.3.33 ResetStats()

```
virtual Error ResetStats ( ) [pure virtual]
```

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

#### 9.5.3.34 RestoreFromMemoryChannel()

```
virtual Error RestoreFromMemoryChannel (
    unsigned int channel ) [pure virtual]
```

Restore the specified current memory channel.

##### Parameters

<i>channel</i>	Memory channel to restore from.
----------------	---------------------------------

##### See also

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

##### Returns

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

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

#### 9.5.3.35 RetrieveBuffer()

```
virtual Error RetrieveBuffer (
    Image * pImage ) [pure virtual]
```

Retrieves the the next image object containing the next image.

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

**Parameters**

<i>pImage</i>	Pointer to <a href="#">Image</a> object to store image data.
---------------	--

**See also**[StartCapture\(\)](#)[StopCapture\(\)](#)**Returns**

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

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

**9.5.3.36 SaveToMemoryChannel()**

```
virtual Error SaveToMemoryChannel (
    unsigned int channel ) [pure virtual]
```

Save the current settings to the specified current memory channel.

**Parameters**

<i>channel</i>	Memory channel to save to.
----------------	----------------------------

**See also**[GetMemoryChannel\(\)](#)[RestoreFromMemoryChannel\(\)](#)[GetMemoryChannelInfo\(\)](#)**Returns**

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

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

**9.5.3.37 SetActiveLUTBank()**

```
virtual Error SetActiveLUTBank (
    unsigned int activeBank ) [pure virtual]
```

Set the LUT bank that will be used.

## Parameters

<i>activeBank</i>	The bank to be set as active.
-------------------	-------------------------------

## Returns

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

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

## 9.5.3.38 SetCallback()

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

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

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

## Parameters

<i>callbackFn</i>	A function to be called when a new image is received.
<i>pCallbackData</i>	A pointer to data that can be passed to the callback function.

## See also

[StartCapture\(\)](#)

## Returns

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

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

## 9.5.3.39 SetConfiguration()

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

Set the configuration associated with the camera object.

## Parameters

<i>pConfig</i>	Pointer to the configuration structure to be used.
----------------	--

See also

[GetConfiguration\(\)](#)

Returns

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

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

#### 9.5.3.40 SetEmbeddedImageInfo()

```
virtual Error SetEmbeddedImageInfo (
    EmbeddedImageInfo * pInfo ) [pure virtual]
```

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

Parameters

<i>pInfo</i>	Structure to be used.
--------------	-----------------------

See also

[GetEmbeddedImageInfo\(\)](#)

Returns

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

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

#### 9.5.3.41 SetGPIOPinDirection()

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

Set the GPIO pin direction for the specified pin.

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



## Parameters

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

## See also

[GetGPIOPinDirection\(\)](#)

## Returns

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

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

## 9.5.3.42 SetLUTChannel()

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

Set the LUT channel settings to the camera.

## Parameters

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

## See also

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

## Returns

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

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

#### 9.5.3.43 SetProperty()

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

Writes the settings for the specified property to the camera.

The property type must be specified in the [Property](#) structure passed into the function in order for the function to succeed. The absControl flag controls whether the absolute or integer value is written to the camera. Use [GetPropertyInfo\(\)](#) to query which options are available for a specific property.

##### Parameters

<i>pProp</i>	Pointer to the <a href="#">Property</a> structure to be used.
<i>broadcast</i>	Whether the action should be broadcast.

##### See also

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

##### Returns

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

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

#### 9.5.3.44 SetStrobe()

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

Set current strobe settings to the camera.

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

##### Parameters

<i>pStrobeControl</i>	Structure providing strobe settings.
<i>broadcast</i>	Whether the action should be broadcast.

##### See also

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

**Returns**

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

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

**9.5.3.45 SetTriggerDelay()**

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

Set the specified trigger delay settings to the camera.

**Parameters**

<i>pTriggerDelay</i>	Structure providing trigger delay settings.
<i>broadcast</i>	Whether the action should be broadcast.

**See also**

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

**Returns**

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

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

**9.5.3.46 SetTriggerMode()**

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

Set the specified trigger settings to the camera.

**Parameters**

<i>pTriggerMode</i>	Structure providing trigger mode settings.
<i>broadcast</i>	Whether the action should be broadcast.

**See also**

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

**Returns**

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

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

**9.5.3.47 SetUserBuffers()**

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

Specify user allocated buffers to use as image data buffers.

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

**Parameters**

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

**See also**

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

**Returns**

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

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

## 9.5.3.48 StartCapture()

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

Starts isochronous image capture.

It will use either the current video mode or the most recently set video mode of the camera. The optional callback function parameter is called on completion of image transfer. When a callback function is specified, the grab mode will determine how images are delivered. If the grab mode has not been set, or has been set to `DROP_FRAMES` the default behavior is to requeue images for DMA if they have not been delivered by the time the next image transfer completes. If `BUFFER_FRAMES` is specified, the next image in the sequence will be delivered. Note that for the `BUFFER_FRAMES` case, if delivery does not keep up with the DMA process, images will be lost. The default behavior is to perform `DROP_FRAMES` image delivery. Alternatively, the callback parameter can be set to `NULL` and [RetrieveBuffer\(\)](#) can be called as a blocking call to get the image data.

## Parameters

<i>callbackFn</i>	A function to be called when a new image is received.
<i>pCallbackData</i>	A pointer to data that can be passed to the callback function.

## See also

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

## Returns

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

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

## 9.5.3.49 StartSyncCapture()

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

Starts isochronous image capture on multiple cameras.

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

## Parameters

<i>numCameras</i>	Number of <a href="#">Camera</a> objects in the ppCameras array.
<i>ppCameras</i>	Array of pointers to <a href="#">Camera</a> objects containing the cameras to be started and synchronized.
<i>pCallbackFns</i>	Array of callback functions for each camera.
<i>pCallbackDataArray</i>	Array of callback data pointers.

## See also

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

## Returns

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

## 9.5.3.50 StopCapture()

```
virtual Error StopCapture ( ) [pure virtual]
```

Stops isochronous image transfer and cleans up all associated resources.

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

## See also

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

## Returns

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

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

## 9.5.3.51 WaitForBufferEvent()

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

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

## Parameters

<i>pImage</i>	Pointer to <a href="#">Image</a> object to store image data.
<i>eventNumber</i>	The event number to wait for.

## See also

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

## Returns

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

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

## 9.5.3.52 WriteRegister()

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

Write to the specified register on the camera.

## Parameters

<i>address</i>	DCAM address to be written to.
<i>value</i>	The value to be written.
<i>broadcast</i>	Whether the action should be broadcast.

## See also

[ReadRegister\(\)](#)

## Returns

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

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

### 9.5.3.53 WriteRegisterBlock()

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

Write to the specified register block on the camera.

#### Parameters

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

#### See also

[ReadRegisterBlock\(\)](#)

#### Returns

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

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

## 9.5.4 Member Data Documentation

### 9.5.4.1 m\_pCameraData

```
CameraData* m_pCameraData [protected]
```

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

- [CameraBase.h](#)

## 9.6 CameraControlDlg Class Reference

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



## Public Member Functions

- [CameraControlDlg](#) ()  
*Default constructor.*
- [~CameraControlDlg](#) ()  
*Default destructor.*
- void [Connect](#) ([CameraBase](#) \*pCamera)  
*Connect dialog to a camera.*
- void [Disconnect](#) ()  
*Disconnect a connected camera from the dialog.*
- void [Show](#) ()  
*Show the dialog.*
- void [Show](#) (void \*pParent)  
*Show the dialog.*
- void [ShowModal](#) ()  
*Show the modal dialog.*
- void [ShowModal](#) (void \*pParent)  
*Show the modal dialog.*
- void [Hide](#) ()  
*Hide the dialog.*
- bool [IsVisible](#) ()  
*Get the visibility of the dialog.*
- void [SetTitle](#) (const char \*title)  
*Change the title of the window.*

### 9.6.1 Detailed Description

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

### 9.6.2 Constructor & Destructor Documentation

#### 9.6.2.1 CameraControlDlg()

```
CameraControlDlg ( )
```

Default constructor.

#### 9.6.2.2 ~CameraControlDlg()

```
~CameraControlDlg ( )
```

Default destructor.

### 9.6.3 Member Function Documentation

#### 9.6.3.1 Connect()

```
void Connect (
    CameraBase * pCamera )
```

Connect dialog to a camera.

##### Parameters

<i>pCamera</i>	Camera object to connect the dialog to.
----------------	---

#### 9.6.3.2 Disconnect()

```
void Disconnect ( )
```

Disconnect a connected camera from the dialog.

#### 9.6.3.3 Hide()

```
void Hide ( )
```

Hide the dialog.

#### 9.6.3.4 isVisible()

```
bool isVisible ( )
```

Get the visibility of the dialog.

##### Returns

Whether the dialog is visible.

#### 9.6.3.5 SetTitle()

```
void SetTitle (
    const char * title )
```

Change the title of the window.

This has to be called after calling [Connect\(\)](#).

## Parameters

<i>title</i>	Null-terminated string representing the title.
--------------	--

**9.6.3.6 Show()** [1/2]

```
void Show ( )
```

Show the dialog.

**9.6.3.7 Show()** [2/2]

```
void Show (
    void * pParent )
```

Show the dialog.

**9.6.3.8 ShowModal()** [1/2]

```
void ShowModal ( )
```

Show the modal dialog.

**9.6.3.9 ShowModal()** [2/2]

```
void ShowModal (
    void * pParent )
```

Show the modal dialog.

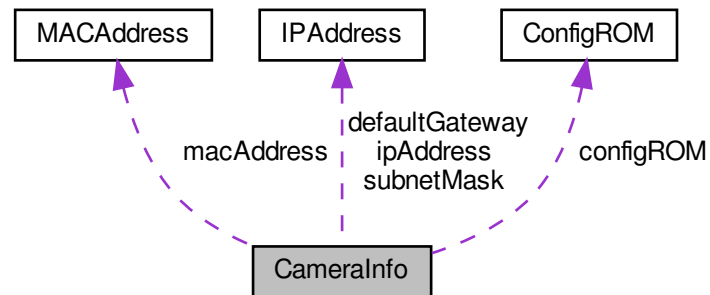
The documentation for this class was generated from the following file:

- [FlyCapture2GUI.h](#)

## 9.7 CameraInfo Struct Reference

Camera information.

Collaboration diagram for CameraInfo:



### Public Member Functions

- [CameraInfo](#) ()

### Public Attributes

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

- [BusSpeed](#) [maximumBusSpeed](#)  
*Maximum bus speed.*
- [BayerTileFormat](#) [bayerTileFormat](#)  
*Bayer tile format.*
- unsigned short [busNumber](#)  
*Bus number, set to 0 for GigE and USB cameras.*
- unsigned short [nodeNumber](#)  
*ieee1394 Node number, set to 0 for GigE and USB cameras*
- [PCleBusSpeed](#) [pcieBusSpeed](#)  
*PCle Bus Speed, set to PCIE\_BUSSPEED\_UNKNOWN for unsupported drivers.*
- unsigned int [reserved](#) [16]  
*Reserved for future use.*

#### IIDC specific information

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

#### GigE specific information

- unsigned int [gigEMajorVersion](#)  
*GigE Vision version.*
- unsigned int [gigEMinorVersion](#)  
*GigE Vision minor version.*
- char [userDefinedName](#) [[sk\\_maxStringLength](#)]  
*User defined name.*
- char [xmlURL1](#) [[sk\\_maxStringLength](#)]  
*XML URL 1.*
- char [xmlURL2](#) [[sk\\_maxStringLength](#)]  
*XML URL 2.*
- [MACAddress](#) [macAddress](#)  
*MAC address.*
- [IPAddress](#) [ipAddress](#)  
*IP address.*
- [IPAddress](#) [subnetMask](#)  
*Subnet mask.*
- [IPAddress](#) [defaultGateway](#)  
*Default gateway.*
- unsigned int [ccpStatus](#)  
*Status/Content of CCP register.*
- unsigned int [applicationIPAddress](#)  
*Local Application IP Address.*
- unsigned int [applicationPort](#)  
*Local Application port.*

### 9.7.1 Detailed Description

[Camera](#) information.

## 9.7.2 Constructor & Destructor Documentation

### 9.7.2.1 CameraInfo()

```
CameraInfo ( ) [inline]
```

## 9.7.3 Member Data Documentation

### 9.7.3.1 applicationIPAddress

```
unsigned int applicationIPAddress
```

Local Application IP Address.

### 9.7.3.2 applicationPort

```
unsigned int applicationPort
```

Local Application port.

### 9.7.3.3 bayerTileFormat

```
BayerTileFormat bayerTileFormat
```

Bayer tile format.

### 9.7.3.4 busNumber

```
unsigned short busNumber
```

Bus number, set to 0 for GigE and USB cameras.

#### 9.7.3.5 ccpStatus

```
unsigned int ccpStatus
```

Status/Content of CCP register.

#### 9.7.3.6 configROM

```
ConfigROM configROM
```

Configuration ROM data.

#### 9.7.3.7 defaultGateway

```
IPAddress defaultGateway
```

Default gateway.

#### 9.7.3.8 driverName

```
char driverName[sk_maxStringLength]
```

Driver name of driver being used.

#### 9.7.3.9 driverType

```
DriverType driverType
```

Driver type.

#### 9.7.3.10 firmwareBuildTime

```
char firmwareBuildTime[sk_maxStringLength]
```

Firmware build time.

#### 9.7.3.11 firmwareVersion

```
char firmwareVersion[sk_maxStringLength]
```

Firmware version of camera.

#### 9.7.3.12 gigEMajorVersion

```
unsigned int gigEMajorVersion
```

GigE Vision version.

#### 9.7.3.13 gigEMinorVersion

```
unsigned int gigEMinorVersion
```

GigE Vision minor version.

#### 9.7.3.14 iidcVer

```
unsigned int iidcVer
```

DCAM version.

#### 9.7.3.15 interfaceType

```
InterfaceType interfaceType
```

Interface type.

#### 9.7.3.16 ipAddress

```
IPAddress ipAddress
```

IP address.



#### 9.7.3.17 isColorCamera

```
bool isColorCamera
```

Flag indicating if this is a color camera.

#### 9.7.3.18 macAddress

```
MACAddress macAddress
```

MAC address.

#### 9.7.3.19 maximumBusSpeed

```
BusSpeed maximumBusSpeed
```

Maximum bus speed.

#### 9.7.3.20 modelName

```
char modelName[sk_maxStringLength]
```

Device model name.

#### 9.7.3.21 nodeNumber

```
unsigned short nodeNumber
```

ieee1394 Node number, set to 0 for GigE and USB cameras

#### 9.7.3.22 pcieBusSpeed

```
PCIEBusSpeed pcieBusSpeed
```

PCIe Bus Speed, set to PCIE\_BUSSPEED\_UNKNOWN for unsupported drivers.

**9.7.3.23 reserved**

```
unsigned int reserved[16]
```

Reserved for future use.

**9.7.3.24 sensorInfo**

```
char sensorInfo[sk_maxStringLength]
```

String detailing the sensor information.

**9.7.3.25 sensorResolution**

```
char sensorResolution[sk_maxStringLength]
```

String providing the sensor resolution.

**9.7.3.26 serialNumber**

```
unsigned int serialNumber
```

Device serial number.

**9.7.3.27 subnetMask**

```
IPAddress subnetMask
```

Subnet mask.

**9.7.3.28 userDefinedName**

```
char userDefinedName[sk_maxStringLength]
```

User defined name.

#### 9.7.3.29 vendorName

```
char vendorName[sk_maxStringLength]
```

Device vendor name.

#### 9.7.3.30 xmlURL1

```
char xmlURL1[sk_maxStringLength]
```

XML URL 1.

#### 9.7.3.31 xmlURL2

```
char xmlURL2[sk_maxStringLength]
```

XML URL 2.

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

- [FlyCapture2Defs.h](#)

## 9.8 CameraSelectionDlg Class Reference

The [CameraSelectionDlg](#) object represents a 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](#).*
- void [SetTitle](#) (const char \*title)  
*Set the window title.*

### 9.8.1 Detailed Description

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

Any GigE cameras that were connected prior to creating a [CameraSelectionDlg](#) will lose CCP after the creation. Consider creating a [CameraSelectionDlg](#) prior to connecting any GigE cameras or calling connect on any outstanding GigE camera.

## 9.8.2 Constructor & Destructor Documentation

### 9.8.2.1 CameraSelectionDlg()

```
CameraSelectionDlg ( )
```

Default constructor.

### 9.8.2.2 ~CameraSelectionDlg()

```
~CameraSelectionDlg ( )
```

Default destructor.

## 9.8.3 Member Function Documentation

### 9.8.3.1 SetTitle()

```
void SetTitle (
    const char * title )
```

Set the window title.

#### Parameters

<i>title</i>	Null-terminated string representing the title.
--------------	--

### 9.8.3.2 ShowModal()

```
void ShowModal (
    bool * pOk,
    PGRGuid * pGuid,
    unsigned int * pSize )
```

Show the [CameraSelectionDlg](#).

#### Parameters

<i>pOk</i>	Whether Ok (true) or Cancel (false) was clicked.
<i>pGuid</i>	Array of PGRGuids containing the selected cameras.
<i>pSize</i>	Size of PGRGuid array.

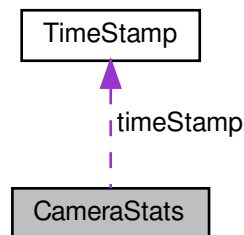
The documentation for this class was generated from the following file:

- [FlyCapture2GUI.h](#)

## 9.9 CameraStats Struct Reference

[Camera](#) diagnostic information.

Collaboration diagram for CameraStats:



### Public Member Functions

- [CameraStats](#) ()

### Public Attributes

- unsigned int [imageDropped](#)
- unsigned int [imageCorrupt](#)
- unsigned int [imageXmitFailed](#)
- unsigned int [imageDriverDropped](#)
- unsigned int [regReadFailed](#)
- unsigned int [regWriteFailed](#)
- unsigned int [portErrors](#)
- bool [cameraPowerUp](#)
- float [cameraVoltages](#) [8]  
*The number of voltage registers available.*
- unsigned int [numVoltages](#)
- float [cameraCurrents](#) [8]  
*The number of current registers available.*
- unsigned int [numCurrents](#)
- unsigned int [temperature](#)
- unsigned int [timeSinceInitialization](#)
- unsigned int [timeSinceBusReset](#)
- [TimeStamp](#) [timeStamp](#)
- unsigned int [numResendPacketsRequested](#)
- unsigned int [numResendPacketsReceived](#)
- unsigned int [reserved](#) [16]  
*Reserved for future use.*

### 9.9.1 Detailed Description

[Camera](#) diagnostic information.

### 9.9.2 Constructor & Destructor Documentation

#### 9.9.2.1 CameraStats()

```
CameraStats ( ) [inline]
```

### 9.9.3 Member Data Documentation

#### 9.9.3.1 cameraCurrents

```
float cameraCurrents[8]
```

#### 9.9.3.2 cameraPowerUp

```
bool cameraPowerUp
```

#### 9.9.3.3 cameraVoltages

```
float cameraVoltages[8]
```

#### 9.9.3.4 imageCorrupt

```
unsigned int imageCorrupt
```

#### 9.9.3.5 imageDriverDropped

```
unsigned int imageDriverDropped
```

#### 9.9.3.6 imageDropped

unsigned int imageDropped

#### 9.9.3.7 imageXmitFailed

unsigned int imageXmitFailed

#### 9.9.3.8 numCurrents

unsigned int numCurrents

The number of current registers available.

0: the values in cameraCurrents[] are invalid.

#### 9.9.3.9 numResendPacketsReceived

unsigned int numResendPacketsReceived

#### 9.9.3.10 numResendPacketsRequested

unsigned int numResendPacketsRequested

#### 9.9.3.11 numVoltages

unsigned int numVoltages

The number of voltage registers available.

0: the values in cameraVoltages[] are invalid.

#### 9.9.3.12 portErrors

unsigned int portErrors

**9.9.3.13 regReadFailed**

```
unsigned int regReadFailed
```

**9.9.3.14 regWriteFailed**

```
unsigned int regWriteFailed
```

**9.9.3.15 reserved**

```
unsigned int reserved[16]
```

Reserved for future use.

**9.9.3.16 temperature**

```
unsigned int temperature
```

**9.9.3.17 timeSinceBusReset**

```
unsigned int timeSinceBusReset
```

**9.9.3.18 timeSinceInitialization**

```
unsigned int timeSinceInitialization
```

**9.9.3.19 timeStamp**

```
TimeStamp timeStamp
```

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

- [FlyCapture2Defs.h](#)



## 9.10 ConfigROM Struct Reference

Camera configuration ROM.

### Public Member Functions

- [ConfigROM](#) ()

### Public Attributes

- unsigned int [nodeVendorId](#)  
*Vendor ID of a node.*
- unsigned int [chipIdHi](#)  
*Chip ID (high part).*
- unsigned int [chipIdLo](#)  
*Chip ID (low part).*
- unsigned int [unitSpecId](#)  
*Unit Spec ID, usually 0xa02d.*
- unsigned int [unitSWVer](#)  
*Unit software version.*
- unsigned int [unitSubSWVer](#)  
*Unit sub software version.*
- unsigned int [vendorUniqueInfo\\_0](#)  
*Vendor unique info 0.*
- unsigned int [vendorUniqueInfo\\_1](#)  
*Vendor unique info 1.*
- unsigned int [vendorUniqueInfo\\_2](#)  
*Vendor unique info 2.*
- unsigned int [vendorUniqueInfo\\_3](#)  
*Vendor unique info 3.*
- char [pszKeyword](#) [[sk\\_maxStringLength](#)]  
*Keyword.*
- unsigned int [reserved](#) [16]  
*Reserved for future use.*

### 9.10.1 Detailed Description

Camera configuration ROM.

### 9.10.2 Constructor & Destructor Documentation

#### 9.10.2.1 ConfigROM()

```
ConfigROM ( ) [inline]
```

### 9.10.3 Member Data Documentation

#### 9.10.3.1 chipIdHi

```
unsigned int chipIdHi
```

Chip ID (high part).

#### 9.10.3.2 chipIdLo

```
unsigned int chipIdLo
```

Chip ID (low part).

#### 9.10.3.3 nodeVendorId

```
unsigned int nodeVendorId
```

Vendor ID of a node.

#### 9.10.3.4 pszKeyword

```
char pszKeyword[sk_maxStringLength]
```

Keyword.

#### 9.10.3.5 reserved

```
unsigned int reserved[16]
```

Reserved for future use.

#### 9.10.3.6 unitSpecId

```
unsigned int unitSpecId
```

Unit Spec ID, usually 0xa02d.

#### 9.10.3.7 unitSubSWVer

```
unsigned int unitSubSWVer
```

Unit sub software version.

#### 9.10.3.8 unitSWVer

```
unsigned int unitSWVer
```

Unit software version.

#### 9.10.3.9 vendorUniqueInfo\_0

```
unsigned int vendorUniqueInfo_0
```

Vendor unique info 0.

#### 9.10.3.10 vendorUniqueInfo\_1

```
unsigned int vendorUniqueInfo_1
```

Vendor unique info 1.

#### 9.10.3.11 vendorUniqueInfo\_2

```
unsigned int vendorUniqueInfo_2
```

Vendor unique info 2.

### 9.10.3.12 vendorUniqueInfo\_3

```
unsigned int vendorUniqueInfo_3
```

Vendor unique info 3.

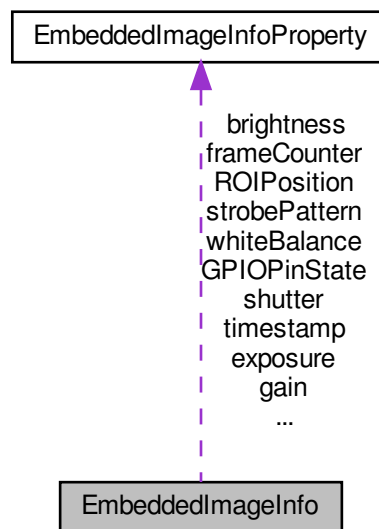
The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 9.11 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](#)

### 9.11.1 Detailed Description

Properties of the possible embedded image information.

### 9.11.2 Member Data Documentation

#### 9.11.2.1 brightness

`EmbeddedImageInfoProperty` brightness

#### 9.11.2.2 exposure

`EmbeddedImageInfoProperty` exposure

#### 9.11.2.3 frameCounter

`EmbeddedImageInfoProperty` frameCounter

#### 9.11.2.4 gain

`EmbeddedImageInfoProperty` gain

#### 9.11.2.5 GPIOPinState

`EmbeddedImageInfoProperty` GPIOPinState

#### 9.11.2.6 ROIPosition

`EmbeddedImageInfoProperty` ROIPosition

#### 9.11.2.7 shutter

[EmbeddedImageInfoProperty](#) shutter

#### 9.11.2.8 strobePattern

[EmbeddedImageInfoProperty](#) strobePattern

#### 9.11.2.9 timestamp

[EmbeddedImageInfoProperty](#) timestamp

#### 9.11.2.10 whiteBalance

[EmbeddedImageInfoProperty](#) whiteBalance

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

- [FlyCapture2Defs.h](#)

## 9.12 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.*

#### 9.12.1 Detailed Description

Properties of a single embedded image info property.

## 9.12.2 Constructor & Destructor Documentation

### 9.12.2.1 EmbeddedImageInfoProperty()

```
EmbeddedImageInfoProperty ( ) [inline]
```

## 9.12.3 Member Data Documentation

### 9.12.3.1 available

```
bool available
```

Whether this property is available.

### 9.12.3.2 onOff

```
bool onOff
```

Whether this property is on or off.

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

- [FlyCapture2Defs.h](#)

## 9.13 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) const  
*Equality operator.*
- virtual bool [operator==](#) (const [ErrorType](#) &errorType) const  
*Equality operator.*
- virtual bool [operator!=](#) (const [Error](#) &error) const  
*Inequality operator.*
- virtual bool [operator!=](#) (const [ErrorType](#) &errorType) const  
*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](#)

### 9.13.1 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.

### 9.13.2 Constructor & Destructor Documentation



#### 9.13.2.1 `Error()` [1/2]

```
Error ( )
```

Default constructor.

#### 9.13.2.2 `Error()` [2/2]

```
Error (
    const Error & error )
```

Copy constructor.

#### 9.13.2.3 `~Error()`

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

Default destructor.

### 9.13.3 Member Function Documentation

#### 9.13.3.1 `CollectSupportInformation()`

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

Retrieve the support information.

It is not implemented in this release.

##### Returns

A string containing support information.

#### 9.13.3.2 `GetBuildDate()`

```
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.

#### 9.13.3.3 GetCause()

```
virtual Error GetCause ( ) const [virtual]
```

Get the error which caused this error.

##### Returns

An error object representing the cause of this error.

#### 9.13.3.4 GetDescription()

```
virtual const char* GetDescription ( ) const [virtual]
```

Retrieve the top level description of the error that occurred.

##### Returns

A string with the error description.

#### 9.13.3.5 GetFilename()

```
virtual const char* GetFilename ( ) const [virtual]
```

Retrieve the source filename where the error originated.

##### Returns

A string with the file name.

#### 9.13.3.6 GetLine()

```
virtual unsigned int GetLine ( ) const [virtual]
```

Retrieve the line number where the error originated.

##### Returns

The line number.

#### 9.13.3.7 GetType()

```
virtual ErrorType GetType ( ) const [virtual]
```

Retrieve the `ErrorType` of the error.

##### Returns

The `ErrorType` of the error.

#### 9.13.3.8 operator!=() [1/2]

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

Inequality operator.

#### 9.13.3.9 operator!=() [2/2]

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

Inequality operator.

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

#### 9.13.3.10 operator=()

```
virtual Error& operator= (
    const Error & error ) [virtual]
```

Assignment operator.

#### 9.13.3.11 operator==( ) [1/2]

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

Equality operator.

#### 9.13.3.12 operator==( ) [2/2]

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

Equality operator.

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

#### 9.13.3.13 PrintErrorTrace()

```
virtual void PrintErrorTrace ( ) const [virtual]
```

Print a formatted log trace to stderr.

### 9.13.4 Friends And Related Function Documentation

#### 9.13.4.1 InternalError

```
friend class InternalError [friend]
```

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

- [Error.h](#)

## 9.14 EventCallbackData Struct Reference

### Public Attributes

- void \* [EventUserData](#)  
*Pointer to the user-supplied data struct.*
- size\_t [EventUserDataSize](#)  
*Size of the user data data supplied to the RegisterEvent() function.*
- const char \* [EventName](#)  
*The event name used to register the event.*
- long long unsigned [EventID](#)  
*The device register which EventName maps to.*
- long long unsigned [EventTimestamp](#)  
*Timestamp indicated the time (as reported by the camera) at which the camera exposure operation completed.*
- void \* [EventData](#)  
*A pointer to additional data pertaining to the event which just trigger the callback function.*
- size\_t [EventDataSize](#)  
*The size of the structure pointed to by EventData.*

### 9.14.1 Member Data Documentation

#### 9.14.1.1 EventData

```
void* EventData
```

A pointer to additional data pertaining to the event which just trigger the callback function.

The data may be of difference sizes or may not even be allocated, depending on the type of event which triggered the callback.

#### 9.14.1.2 EventDataSize

```
size_t EventDataSize
```

The size of the structure pointed to by EventData.

This value should be checked, especially if there are events which can trigger variable- length event data to be returned to the user when the callback function is issued.

#### 9.14.1.3 EventID

```
long long unsigned EventID
```

The device register which EventName maps to.

Provides an alternate means of indexing into different event types.

#### 9.14.1.4 EventName

```
const char* EventName
```

The event name used to register the event.

Provided so the user knows which event triggered the callback.

#### 9.14.1.5 EventTimestamp

```
long long unsigned EventTimestamp
```

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

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

#### 9.14.1.6 EventUserData

```
void* EventUserData
```

Pointer to the user-supplied data struct.

#### 9.14.1.7 EventUserDataSize

```
size_t EventUserDataSize
```

Size of the user data data supplied to the RegisterEvent() function.

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

- [FlyCapture2Defs.h](#)

### 9.15 EventOptions Struct Reference

Options for enabling device event registration.

#### Public Attributes

- [CameraEventCallback EventCallbackFcn](#)  
*Callback function pointer.*
- const char \* [EventName](#)  
*Event name to register.*
- const void \* [EventUserData](#)  
*Pointer to callback data to be passed to the callback function.*
- size\_t [EventUserDataSize](#)  
*Size of the underlying struct passed as eventCallbackData for sanity checks.*

#### 9.15.1 Detailed Description

Options for enabling device event registration.

#### 9.15.2 Member Data Documentation

#### 9.15.2.1 EventCallbackFcn

[CameraEventCallback](#) EventCallbackFcn

Callback function pointer.

#### 9.15.2.2 EventName

`const char* EventName`

Event name to register.

#### 9.15.2.3 EventUserData

`const void* EventUserData`

Pointer to callback data to be passed to the callback function.

#### 9.15.2.4 EventUserDataSize

`size_t EventUserDataSize`

Size of the underlying struct passed as eventCallbackData for sanity checks.

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

- [FlyCapture2Defs.h](#)

## 9.16 FC2Config Struct Reference

Configuration for a camera.

### Public Member Functions

- [FC2Config](#) ()

## Public Attributes

- unsigned int [numBuffers](#)  
*Number of buffers used by the [FlyCapture2](#) library to grab images.*
- unsigned int [numImageNotifications](#)  
*Number of notifications per image.*
- unsigned int [minNumImageNotifications](#)  
*Minimum number of notifications needed for the current image settings on the camera.*
- int [grabTimeout](#)  
*Time in milliseconds that [RetrieveBuffer\(\)](#) and [WaitForBufferEvent\(\)](#) will wait for an image before timing out and returning.*
- [GrabMode](#) [grabMode](#)  
*Grab mode for the camera.*
- bool [highPerformanceRetrieveBuffer](#)  
*This parameter enables [RetrieveBuffer](#) to run in high performance mode.*
- [BusSpeed](#) [isochBusSpeed](#)  
*Isochronous bus speed.*
- [BusSpeed](#) [asyncBusSpeed](#)  
*Asynchronous bus speed.*
- [BandwidthAllocation](#) [bandwidthAllocation](#)  
*Bandwidth allocation flag that tells the camera the bandwidth allocation strategy to employ.*
- unsigned int [registerTimeoutRetries](#)  
*Number of retries to perform when a register read/write timeout is received by the library.*
- unsigned int [registerTimeout](#)  
*Register read/write timeout value, in microseconds.*
- unsigned int [reserved](#) [16]  
*Reserved for future use.*

### 9.16.1 Detailed Description

Configuration for a camera.

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

### 9.16.2 Constructor & Destructor Documentation

#### 9.16.2.1 FC2Config()

```
FC2Config ( ) [inline]
```

### 9.16.3 Member Data Documentation



#### 9.16.3.1 asyncBusSpeed

`BusSpeed` asyncBusSpeed

Asynchronous bus speed.

#### 9.16.3.2 bandwidthAllocation

`BandwidthAllocation` bandwidthAllocation

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

#### 9.16.3.3 grabMode

`GrabMode` grabMode

Grab mode for the camera.

The default is DROP\_FRAMES.

#### 9.16.3.4 grabTimeout

`int` grabTimeout

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

#### 9.16.3.5 highPerformanceRetrieveBuffer

`bool` highPerformanceRetrieveBuffer

This parameter enables RetrieveBuffer to run in high performance mode.

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

#### 9.16.3.6 isochBusSpeed

[BusSpeed](#) isochBusSpeed

Isochronous bus speed.

#### 9.16.3.7 minNumImageNotifications

```
unsigned int minNumImageNotifications
```

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

Read-only value.

#### 9.16.3.8 numBuffers

```
unsigned int numBuffers
```

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

#### 9.16.3.9 numImageNotifications

```
unsigned int numImageNotifications
```

Number of notifications per image.

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

There are 4 general scenarios:

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

Specifying zero for the number of notifications will be ignored (the current value will not be modified).

Note that the event numbers start at 0. Ex. when 3 notifications are used, the three events will be 0, 1 and 2.

#### 9.16.3.10 registerTimeout

```
unsigned int registerTimeout
```

Register read/write timeout value, in microseconds.

The default value is dependent on the interface type.

#### 9.16.3.11 registerTimeoutRetries

```
unsigned int registerTimeoutRetries
```

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

The default value is 0.

#### 9.16.3.12 reserved

```
unsigned int reserved[16]
```

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

## 9.17 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.*

### 9.17.1 Detailed Description

The current version of the library.

## 9.17.2 Member Data Documentation

### 9.17.2.1 build

`unsigned int build`

Build version number.

### 9.17.2.2 major

`unsigned int major`

Major version number.

### 9.17.2.3 minor

`unsigned int minor`

Minor version number.

### 9.17.2.4 type

`unsigned int type`

Type version number.

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

- [FlyCapture2Defs.h](#)

## 9.18 FlyCapture2Video Class Reference

The [FlyCapture2Video](#) class provides the functionality for the user to record images to an AVI file.

## Public Member Functions

- [FlyCapture2Video](#) ()  
*Default constructor.*
- virtual [~FlyCapture2Video](#) ()  
*Default destructor.*
- virtual [Error Open](#) (const char \*pFileName, [AVIOption](#) \*pOption)  
*Open an AVI file in preparation for writing Images to disk.*
- virtual [Error Open](#) (const char \*pFileName, [MJPGOption](#) \*pOption)  
*Open an MJPEG AVI file in preparation for writing Images to disk.*
- virtual [Error Open](#) (const char \*pFileName, [H264Option](#) \*pOption)  
*Open an H.264 video file in preparation for writing Images to disk.*
- virtual [Error Append](#) ([Image](#) \*pImage)  
*Append an image to the AVI/MP4 file.*
- virtual [Error Close](#) ()  
*Close the AVI/MP4 file.*
- virtual void [SetMaximumFileSize](#) (unsigned int size)  
*Set the maximum file size (in megabytes) of a AVI/MP4 file.*

### 9.18.1 Detailed Description

The [FlyCapture2Video](#) class provides the functionality for the user to record images to an AVI file.

### 9.18.2 Constructor & Destructor Documentation

#### 9.18.2.1 FlyCapture2Video()

```
FlyCapture2Video ( )
```

Default constructor.

#### 9.18.2.2 ~FlyCapture2Video()

```
virtual ~FlyCapture2Video ( ) [virtual]
```

Default destructor.

### 9.18.3 Member Function Documentation

#### 9.18.3.1 Append()

```
virtual Error Append (  
    Image * pImage ) [virtual]
```

Append an image to the AVI/MP4 file.

**Parameters**

<i>pImage</i>	The image to append.
---------------	----------------------

**Returns**

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

**9.18.3.2 Close()**

```
virtual Error Close ( ) [virtual]
```

Close the AVI/MP4 file.

**See also**

[Open\(\)](#)

**Returns**

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

**9.18.3.3 Open()** [1/3]

```
virtual Error Open (
    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**

<i>pFileName</i>	The filename of the AVI file.
<i>pOption</i>	Options to apply to the AVI file.

**See also**

[SetMaximumFileSize\(\)](#)  
[Close\(\)](#)

### Returns

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

#### 9.18.3.4 Open() [2/3]

```
virtual Error Open (
    const char * pFileName,
    MJPGOption * pOption ) [virtual]
```

Open an MJPEG 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

<i>pFileName</i>	The filename of the AVI file.
<i>pOption</i>	MJPEG options to apply to the AVI file.

### See also

[SetMaximumFileSize\(\)](#)  
[Close\(\)](#)  
[MJPGOption](#)

### Returns

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

#### 9.18.3.5 Open() [3/3]

```
virtual Error Open (
    const char * pFileName,
    H264Option * pOption ) [virtual]
```

Open an H.264 video file in preparation for writing Images to disk.

If the file extension is not specified, MP4 will be used as the default container. Consult ffmpeg documentation for a list of supported containers.

### Parameters

<i>pFileName</i>	The filename of the video file.
<i>pOption</i>	H.264 options to apply to the video file.

See also

[Close\(\)](#)  
[H264Option](#)

Returns

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

### 9.18.3.6 SetMaximumFileSize()

```
virtual void SetMaximumFileSize (
    unsigned int size ) [virtual]
```

Set the maximum file size (in megabytes) of a AVI/MP4 file.

A new AVI/MP4 file is created automatically when file size limit is reached. Setting a maximum size of 0 indicates no limit on file size.

Parameters

<i>size</i>	The maximum AVI file size in MB.
-------------	----------------------------------

See also

[Append\(\)](#)

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

- [FlyCapture2Video.h](#)

## 9.19 FlyCapture3ApiGuiWrapper Class Reference

### Public Member Functions

- [WRAPPER\\_API FlyCapture3ApiGuiWrapper](#) (void)
- [WRAPPER\\_API ~FlyCapture3ApiGuiWrapper](#) (void)
- [WRAPPER\\_API void ConnectGUILibrary](#) (FlyCapture2::GCCamera &camera)
- [WRAPPER\\_API void DisconnectGUILibrary](#) ()
- [WRAPPER\\_API void ShowPropertyGridDialog](#) ()
- [WRAPPER\\_API void ShowCameraSelectionDialog](#) ()
- [WRAPPER\\_API int GetNumDialogs](#) ()
- [WRAPPER\\_API std::list< std::string > GetDialogNameList](#) ()
- [WRAPPER\\_API void ShowDialogByName](#) (std::string dialogName)
- [WRAPPER\\_API void ShowDialogByIndex](#) (int index)
- [WRAPPER\\_API int GetNumOfControls](#) ()
- [WRAPPER\\_API std::list< std::string > GetControlNameList](#) ()



## 9.19.1 Constructor & Destructor Documentation

### 9.19.1.1 FlyCapture3ApiGuiWrapper()

```
WRAPPER_API FlyCapture3ApiGuiWrapper (
    void )
```

### 9.19.1.2 ~FlyCapture3ApiGuiWrapper()

```
WRAPPER_API ~FlyCapture3ApiGuiWrapper (
    void )
```

## 9.19.2 Member Function Documentation

### 9.19.2.1 ConnectGUILibrary()

```
WRAPPER_API void ConnectGUILibrary (
    FlyCapture2::GCCamera & camera )
```

### 9.19.2.2 DisconnectGUILibrary()

```
WRAPPER_API void DisconnectGUILibrary ( )
```

### 9.19.2.3 GetControlNameList()

```
WRAPPER_API std::list<std::string> GetControlNameList ( )
```

### 9.19.2.4 GetDialogNameList()

```
WRAPPER_API std::list<std::string> GetDialogNameList ( )
```

#### 9.19.2.5 GetNumDialogs()

```
WRAPPER_API int GetNumDialogs ( )
```

#### 9.19.2.6 GetNumOfControls()

```
WRAPPER_API int GetNumOfControls ( )
```

#### 9.19.2.7 ShowCameraSelectionDialog()

```
WRAPPER_API void ShowCameraSelectionDialog ( )
```

#### 9.19.2.8 ShowDialogByIndex()

```
WRAPPER_API void ShowDialogByIndex (
    int index )
```

#### 9.19.2.9 ShowDialogByName()

```
WRAPPER_API void ShowDialogByName (
    std::string dialogName )
```

#### 9.19.2.10 ShowPropertyGridDialog()

```
WRAPPER_API void ShowPropertyGridDialog ( )
```

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

- [FlyCapture3ApiGuiWrapper.h](#)

## 9.20 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.*

### 9.20.1 Detailed Description

Format 7 image settings.

### 9.20.2 Constructor & Destructor Documentation

#### 9.20.2.1 Format7ImageSettings()

```
Format7ImageSettings ( ) [inline]
```

### 9.20.3 Member Data Documentation

#### 9.20.3.1 height

```
unsigned int height
```

Height of image.

#### 9.20.3.2 mode

[Mode](#) mode

Format 7 mode.

#### 9.20.3.3 offsetX

unsigned int offsetX

Horizontal image offset.

#### 9.20.3.4 offsetY

unsigned int offsetY

Vertical image offset.

#### 9.20.3.5 pixelFormat

[PixelFormat](#) pixelFormat

Pixel format of image.

#### 9.20.3.6 reserved

unsigned int reserved[8]

Reserved for future use.

#### 9.20.3.7 width

unsigned int width

Width of image.

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

- [FlyCapture2Defs.h](#)

## 9.21 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 [vendorPixelFormatBitField](#)  
*Vendor unique pixel formats in a bit field.*
- unsigned int [packetSize](#)  
*Current packet size in bytes.*
- unsigned int [minPacketSize](#)  
*Minimum packet size in bytes for current mode.*
- unsigned int [maxPacketSize](#)  
*Maximum packet size in bytes for current mode.*
- float [percentage](#)  
*Current packet size as a percentage of maximum packet size.*
- unsigned int [reserved](#) [16]  
*Reserved for future use.*

### 9.21.1 Detailed Description

Format 7 information for a single mode.

### 9.21.2 Constructor & Destructor Documentation

#### 9.21.2.1 Format7Info()

```
Format7Info ( ) [inline]
```

### 9.21.3 Member Data Documentation

#### 9.21.3.1 imageHStepSize

```
unsigned int imageHStepSize
```

Horizontal step size for the image.

#### 9.21.3.2 imageVStepSize

```
unsigned int imageVStepSize
```

Vertical step size for the image.

#### 9.21.3.3 maxHeight

```
unsigned int maxHeight
```

Maximum image height.

#### 9.21.3.4 maxPacketSize

```
unsigned int maxPacketSize
```

Maximum packet size in bytes for current mode.

#### 9.21.3.5 maxWidth

```
unsigned int maxWidth
```

Maximum image width.

#### 9.21.3.6 minPacketSize

```
unsigned int minPacketSize
```

Minimum packet size in bytes for current mode.

#### 9.21.3.7 mode

```
Mode mode
```

Format 7 mode.

#### 9.21.3.8 offsetHStepSize

```
unsigned int offsetHStepSize
```

Horizontal step size for the offset.

#### 9.21.3.9 offsetVStepSize

```
unsigned int offsetVStepSize
```

Vertical step size for the offset.

#### 9.21.3.10 packetSize

```
unsigned int packetSize
```

Current packet size in bytes.

#### 9.21.3.11 percentage

```
float percentage
```

Current packet size as a percentage of maximum packet size.

#### 9.21.3.12 pixelFormatBitFields

```
unsigned int pixelFormatBitFields
```

Supported pixel formats in a bit field.

#### 9.21.3.13 reserved

```
unsigned int reserved[16]
```

Reserved for future use.

#### 9.21.3.14 vendorPixelFormatBitFields

```
unsigned int vendorPixelFormatBitFields
```

Vendor unique pixel formats in a bit field.

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

- [FlyCapture2Defs.h](#)

## 9.22 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.*



### 9.22.1 Detailed Description

Format 7 packet information.

### 9.22.2 Constructor & Destructor Documentation

#### 9.22.2.1 Format7PacketInfo()

```
Format7PacketInfo ( ) [inline]
```

### 9.22.3 Member Data Documentation

#### 9.22.3.1 maxBytesPerPacket

```
unsigned int maxBytesPerPacket
```

Maximum bytes per packet.

#### 9.22.3.2 recommendedBytesPerPacket

```
unsigned int recommendedBytesPerPacket
```

Recommended bytes per packet.

#### 9.22.3.3 reserved

```
unsigned int reserved[8]
```

Reserved for future use.

#### 9.22.3.4 unitBytesPerPacket

```
unsigned int unitBytesPerPacket
```

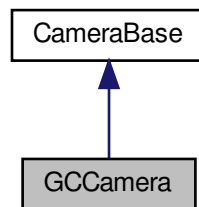
Minimum bytes per packet.

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

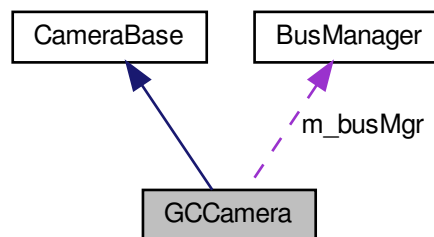
- [FlyCapture2Defs.h](#)

## 9.23 GCCamera Class Reference

Inheritance diagram for GCCamera:



Collaboration diagram for GCCamera:



### Public Member Functions

- [GCCamera](#) (void)
- virtual [~GCCamera](#) (void)
- [::GenApi::INodeMap \\* GetNodeMap](#) ()
- [Error SetCamera](#) ([CameraBase](#) \*camera)
- [Error SetCamera](#) ([CameraBase](#) \*camera, const char \*filepath=[NULL](#))
- [std::string GCCamera::GetXML](#) ()
- virtual [Error WriteGVCPRegister](#) (unsigned int address, unsigned int value, bool broadcast=false)
- virtual [Error ReadGVCPRegister](#) (unsigned int address, unsigned int \*pValue)
- virtual [Error WriteGVCPRegisterBlock](#) (unsigned int address, const unsigned int \*pBuffer, unsigned int length)
- virtual [Error ReadGVCPRegisterBlock](#) (unsigned int address, unsigned int \*pBuffer, unsigned int length)
- virtual [Error WriteGVCPMemory](#) (unsigned int address, const unsigned char \*pBuffer, unsigned int length)
- virtual [Error ReadGVCPMemory](#) (unsigned int address, unsigned char \*pBuffer, unsigned int length)
- virtual [Error Connect](#) ([PGRGuid](#) \*pGuid=[NULL](#))

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

- [Error Connect](#) ([PGRGuid](#) \*pGuid=[NULL](#), const char \*filepath=[NULL](#))

- virtual [Error Disconnect](#) ()  
*Disconnects the camera object from the camera.*
- virtual bool [IsConnected](#) ()  
*Checks if the camera object is connected to a physical camera specified by a GUID.*
- virtual [Error SetCallback](#) ([ImageEventCallback](#) callbackFn, const void \*pCallbackData=NULL)  
*Sets the callback data to be used on completion of image transfer.*
- virtual [Error StartCapture](#) ([ImageEventCallback](#) callbackFn=NULL, const void \*pCallbackData=NULL)  
*Starts isochronous image capture.*
- virtual [Error RetrieveBuffer](#) ([Image](#) \*pImage)  
*Retrieves the the next image object containing the next image.*
- virtual [Error StopCapture](#) ()  
*Stops isochronous image transfer and cleans up all associated resources.*
- virtual [Error WaitForBufferEvent](#) ([Image](#) \*pImage, unsigned int eventNumber)  
*Retrieves the next image event containing the next part of the image.*
- virtual [Error SetUserBuffers](#) (unsigned char \*const pMemBuffers, int size, int numBuffers)  
*Specify user allocated buffers to use as image data buffers.*
- virtual [Error GetConfiguration](#) ([FC2Config](#) \*pConfig)  
*Get the configuration associated with the camera object.*
- virtual [Error SetConfiguration](#) (const [FC2Config](#) \*pConfig)  
*Set the configuration associated with the camera object.*
- virtual [Error GetCameraInfo](#) ([CameraInfo](#) \*pCameraInfo)  
*Retrieves information from the camera such as serial number, model name and other camera information.*
- virtual [Error GetPropertyInfo](#) ([PropertyInfo](#) \*pPropInfo)  
*Retrieves information about the specified camera property.*
- virtual [Error GetProperty](#) ([Property](#) \*pProp)  
*Reads the settings for the specified property from the camera.*
- virtual [Error SetProperty](#) (const [Property](#) \*pProp, bool broadcast=false)  
*Writes the settings for the specified property to the camera.*
- virtual [Error GetGPIOPinDirection](#) (unsigned int pin, unsigned int \*pDirection)  
*Get the GPIO pin direction for the specified pin.*
- virtual [Error SetGPIOPinDirection](#) (unsigned int pin, unsigned int direction, bool broadcast=false)  
*Set the GPIO pin direction for the specified pin.*
- virtual [Error GetTriggerModelInfo](#) ([TriggerModelInfo](#) \*pTriggerModelInfo)  
*Retrieve trigger information from the camera.*
- virtual [Error GetTriggerMode](#) ([TriggerMode](#) \*pTriggerMode)  
*Retrieve current trigger settings from the camera.*
- virtual [Error SetTriggerMode](#) (const [TriggerMode](#) \*pTriggerMode, bool broadcast=false)  
*Set the specified trigger settings to the camera.*
- virtual [Error FireSoftwareTrigger](#) (bool broadcast=false)  
*Fire the software trigger according to the DCAM specifications.*
- virtual [Error GetTriggerDelayInfo](#) ([TriggerDelayInfo](#) \*pTriggerDelayInfo)  
*Retrieve trigger delay information from the camera.*
- virtual [Error GetTriggerDelay](#) ([TriggerDelay](#) \*pTriggerDelay)  
*Retrieve current trigger delay settings from the camera.*
- virtual [Error SetTriggerDelay](#) (const [TriggerDelay](#) \*pTriggerDelay, bool broadcast=false)  
*Set the specified trigger delay settings to the camera.*
- virtual [Error GetStrobeInfo](#) ([StrobeInfo](#) \*pStrobeInfo)  
*Retrieve strobe information from the camera.*
- virtual [Error GetStrobe](#) ([StrobeControl](#) \*pStrobeControl)  
*Retrieve current strobe settings from the camera.*
- virtual [Error SetStrobe](#) (const [StrobeControl](#) \*pStrobeControl, bool broadcast=false)

- Set current strobe settings to the camera.*

  - virtual [Error GetLUTInfo](#) ([LUTData](#) \*pData)
- Query if LUT support is available on the camera.*

  - virtual [Error GetLUTBankInfo](#) (unsigned int bank, bool \*pReadSupported, bool \*pWriteSupported)
- Query the read/write status of a single LUT bank.*

  - virtual [Error GetActiveLUTBank](#) (unsigned int \*pActiveBank)
- Get the LUT bank that is currently being used.*

  - virtual [Error SetActiveLUTBank](#) (unsigned int activeBank)
- Set the LUT bank that will be used.*

  - virtual [Error EnableLUT](#) (bool on)
- Enable or disable LUT functionality on the camera.*

  - virtual [Error GetLUTChannel](#) (unsigned int bank, unsigned int channel, unsigned int sizeEntries, unsigned int \*pEntries)
- Get the LUT channel settings from the camera.*

  - virtual [Error SetLUTChannel](#) (unsigned int bank, unsigned int channel, unsigned int sizeEntries, const unsigned int \*pEntries)
- Set the LUT channel settings to the camera.*

  - virtual [Error GetMemoryChannel](#) (unsigned int \*pCurrentChannel)
- Retrieve the current memory channel from the camera.*

  - virtual [Error SaveToMemoryChannel](#) (unsigned int channel)
- Save the current settings to the specified current memory channel.*

  - virtual [Error RestoreFromMemoryChannel](#) (unsigned int channel)
- Restore the specified current memory channel.*

  - virtual [Error GetMemoryChannelInfo](#) (unsigned int \*pNumChannels)
- Query the camera for memory channel support.*

  - virtual [Error GetEmbeddedImageInfo](#) ([EmbeddedImageInfo](#) \*pInfo)
- Get the current status of the embedded image information register, as well as the availability of each embedded property.*

  - virtual [Error SetEmbeddedImageInfo](#) ([EmbeddedImageInfo](#) \*pInfo)
- Sets the on/off values of the embedded image information structure to the camera.*

  - virtual [Error WriteRegister](#) (unsigned int address, unsigned int value, bool broadcast=false)
- Write to the specified register on the camera.*

  - virtual [Error ReadRegister](#) (unsigned int address, unsigned int \*pValue)
- Read the specified register from the camera.*

  - virtual [Error WriteRegisterBlock](#) (unsigned short addressHigh, unsigned int addressLow, const unsigned int \*pBuffer, unsigned int length)
- Write to the specified register block on the camera.*

  - virtual [Error ReadRegisterBlock](#) (unsigned short addressHigh, unsigned int addressLow, unsigned int \*pBuffer, unsigned int length)
- Read from the specified register block on the camera.*

  - [Error GetCycleTime](#) ([TimeStamp](#) \*timeStamp)
- Returns a Timestamp struct containing 1394 CYCLE\_TIME information.*

  - [InterfaceType GetInterfaceType](#) ()
- virtual [Error GetStats](#) ([CameraStats](#) \*pStats)
  - virtual [Error ResetStats](#) ()

## Static Public Member Functions

- static [Error StartSyncCapture](#) (unsigned int numCameras, const [GigECamera](#) \*\*ppCameras, const [ImageCallback](#) \*pCallbackFns=NULL, const void \*\*pCallbackDataArray=NULL)
- static const char \* [GetRegisterString](#) (unsigned int registerVal)

## Protected Member Functions

- void [TestGainNode](#) ()

## Protected Attributes

- [BusManager](#) `m_busMgr`

## 9.23.1 Constructor & Destructor Documentation

### 9.23.1.1 GCCamera()

```
GCCamera (  
    void )
```

### 9.23.1.2 ~GCCamera()

```
virtual ~GCCamera (  
    void ) [virtual]
```

## 9.23.2 Member Function Documentation

### 9.23.2.1 Connect() [1/2]

```
virtual Error Connect (  
    PGRGuid * pGuid = NULL ) [virtual]
```

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

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

Implements [CameraBase](#).

### 9.23.2.2 Connect() [2/2]

```
Error Connect (  
    PGRGuid * pGuid = NULL,  
    const char * filepath = NULL )
```

### 9.23.2.3 Disconnect()

```
virtual Error Disconnect ( ) [virtual]
```

Disconnects the camera object from the camera.

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

See also

[Connect\(\)](#)

Returns

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

Implements [CameraBase](#).

### 9.23.2.4 EnableLUT()

```
virtual Error EnableLUT (
    bool on ) [virtual]
```

Enable or disable LUT functionality on the camera.

Parameters

<i>on</i>	Whether to enable or disable LUT.
-----------	-----------------------------------

See also

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

Returns

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

Implements [CameraBase](#).

### 9.23.2.5 FireSoftwareTrigger()

```
virtual Error FireSoftwareTrigger (
    bool broadcast = false ) [virtual]
```

Fire the software trigger according to the DCAM specifications.

## Parameters

<i>broadcast</i>	Whether the action should be broadcast.
------------------	---

## Returns

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

Implements [CameraBase](#).

## 9.23.2.6 GCCamera::GetXML()

```
std::string GCCamera::GetXML ( )
```

## 9.23.2.7 GetActiveLUTBank()

```
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

<i>pActiveBank</i>	The currently active bank.
--------------------	----------------------------

## Returns

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

Implements [CameraBase](#).

## 9.23.2.8 GetCameraInfo()

```
virtual Error GetCameraInfo (
    CameraInfo * pCameraInfo ) [virtual]
```

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

## Parameters

<i>pCameraInfo</i>	Pointer to the camera information structure to be filled.
--------------------	---

## Returns

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

Implements [CameraBase](#).

### 9.23.2.9 GetConfiguration()

```
virtual Error GetConfiguration (
    FC2Config * pConfig ) [virtual]
```

Get the configuration associated with the camera object.

## Parameters

<i>pConfig</i>	Pointer to the configuration structure to be filled.
----------------	--

## See also

[SetConfiguration\(\)](#)

## Returns

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

Implements [CameraBase](#).

### 9.23.2.10 GetCycleTime()

```
Error GetCycleTime (
    TimeStamp * timeStamp ) [virtual]
```

Returns a Timestamp struct containing 1394 CYCLE\_TIME information.

## Parameters

<i>registerVal</i>	The register value to query.
--------------------	------------------------------



**Returns**

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

Implements [CameraBase](#).

**9.23.2.11 GetEmbeddedImageInfo()**

```
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**

<i>pInfo</i>	Structure to be filled.
--------------	-------------------------

**See also**

[SetEmbeddedImageInfo\(\)](#)

**Returns**

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

Implements [CameraBase](#).

**9.23.2.12 GetGPIOPinDirection()**

```
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**

<i>pin</i>	Pin to get the direction for.
<i>pDirection</i>	Direction of the pin. 0 for input, 1 for output.

See also

[SetGPIOPinDirection\(\)](#)

Returns

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

Implements [CameraBase](#).

#### 9.23.2.13 GetInterfaceType()

```
InterfaceType GetInterfaceType ( )
```

#### 9.23.2.14 GetLUTBankInfo()

```
virtual Error GetLUTBankInfo (
    unsigned int bank,
    bool * pReadSupported,
    bool * pWriteSupported ) [virtual]
```

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

Parameters

<i>bank</i>	The bank to query.
<i>pReadSupported</i>	Whether reading from the bank is supported.
<i>pWriteSupported</i>	Whether writing to the bank is supported.

Returns

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

Implements [CameraBase](#).

#### 9.23.2.15 GetLUTChannel()

```
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

<i>bank</i>	Bank to retrieve.
<i>channel</i>	Channel to retrieve.
<i>sizeEntries</i>	Number of entries in LUT table to read.
<i>pEntries</i>	Array to store LUT entries.

## See also

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

## Returns

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

Implements [CameraBase](#).

## 9.23.2.16 GetLUTInfo()

```
virtual Error GetLUTInfo (
    LUTData * pData ) [virtual]
```

Query if LUT support is available on the camera.

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

## Parameters

<i>pData</i>	The LUT structure to be filled.
--------------	---------------------------------

## See also

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

## Returns

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

Implements [CameraBase](#).

#### 9.23.2.17 GetMemoryChannel()

```
virtual Error GetMemoryChannel (
    unsigned int * pCurrentChannel ) [virtual]
```

Retrieve the current memory channel from the camera.

##### Parameters

<i>pCurrentChannel</i>	Current memory channel.
------------------------	-------------------------

##### See also

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

##### Returns

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

Implements [CameraBase](#).

#### 9.23.2.18 GetMemoryChannelInfo()

```
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

<i>pNumChannels</i>	Number of memory channels supported.
---------------------	--------------------------------------

##### See also

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

##### Returns

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

Implements [CameraBase](#).

#### 9.23.2.19 GetNodeMap()

```
::GenApi::INodeMap* GetNodeMap ( )
```

#### 9.23.2.20 GetProperty()

```
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

<i>pProp</i>	Pointer to the <a href="#">Property</a> structure to be filled.
--------------	---

##### See also

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

##### Returns

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

Implements [CameraBase](#).

#### 9.23.2.21 GetPropertyInfo()

```
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

<i>pPropInfo</i>	Pointer to the <a href="#">PropertyInfo</a> structure to be filled.
------------------	---

See also

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

Returns

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

Implements [CameraBase](#).

#### 9.23.2.22 GetRegisterString()

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

#### 9.23.2.23 GetStats()

```
virtual Error GetStats (  
    CameraStats * pStats )    [virtual]
```

Implements [CameraBase](#).

#### 9.23.2.24 GetStrobe()

```
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

<i>pStrobeControl</i>	Structure to receive strobe settings.
-----------------------	---------------------------------------

See also

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

#### Returns

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

Implements [CameraBase](#).

#### 9.23.2.25 GetStrobeInfo()

```
virtual Error GetStrobeInfo (
    StrobeInfo * pStrobeInfo ) [virtual]
```

Retrieve strobe information from the camera.

#### Parameters

<i>pStrobeInfo</i>	Structure to receive strobe information.
--------------------	--

#### See also

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

#### Returns

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

Implements [CameraBase](#).

#### 9.23.2.26 GetTriggerDelay()

```
virtual Error GetTriggerDelay (
    TriggerDelay * pTriggerDelay ) [virtual]
```

Retrieve current trigger delay settings from the camera.

#### Parameters

<i>pTriggerDelay</i>	Structure to receive trigger delay settings.
----------------------	--

#### See also

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

**Returns**

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

Implements [CameraBase](#).

**9.23.2.27 GetTriggerDelayInfo()**

```
virtual Error GetTriggerDelayInfo (
    TriggerDelayInfo * pTriggerDelayInfo ) [virtual]
```

Retrieve trigger delay information from the camera.

**Parameters**

<i>pTriggerDelayInfo</i>	Structure to receive trigger delay information.
--------------------------	---

**See also**

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

**Returns**

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

Implements [CameraBase](#).

**9.23.2.28 GetTriggerMode()**

```
virtual Error GetTriggerMode (
    TriggerMode * pTriggerMode ) [virtual]
```

Retrieve current trigger settings from the camera.

**Parameters**

<i>pTriggerMode</i>	Structure to receive trigger mode settings.
---------------------	---

**See also**

[GetTriggerModelInfo\(\)](#)  
[SetTriggerMode\(\)](#)



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

#### Returns

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

Implements [CameraBase](#).

#### 9.23.2.29 GetTriggerModeInfo()

```
virtual Error GetTriggerModeInfo (
    TriggerModeInfo * pTriggerModeInfo ) [virtual]
```

Retrieve trigger information from the camera.

#### Parameters

<i>pTriggerModeInfo</i>	Structure to receive trigger information.
-------------------------	---

#### See also

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

#### Returns

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

Implements [CameraBase](#).

#### 9.23.2.30 IsConnected()

```
virtual bool IsConnected ( ) [virtual]
```

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

#### See also

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

#### Returns

Whether [Connect\(\)](#) was called on the camera object.

Implements [CameraBase](#).

#### 9.23.2.31 ReadGVCPMemory()

```
virtual Error ReadGVCPMemory (
    unsigned int address,
    unsigned char * pBuffer,
    unsigned int length ) [virtual]
```

#### 9.23.2.32 ReadGVCPRegister()

```
virtual Error ReadGVCPRegister (
    unsigned int address,
    unsigned int * pValue ) [virtual]
```

#### 9.23.2.33 ReadGVCPRegisterBlock()

```
virtual Error ReadGVCPRegisterBlock (
    unsigned int address,
    unsigned int * pBuffer,
    unsigned int length ) [virtual]
```

#### 9.23.2.34 ReadRegister()

```
virtual Error ReadRegister (
    unsigned int address,
    unsigned int * pValue ) [virtual]
```

Read the specified register from the camera.

##### Parameters

<i>address</i>	DCAM address to be read from.
<i>pValue</i>	The value that is read.

##### See also

[WriteRegister\(\)](#)

##### Returns

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

Implements [CameraBase](#).

#### 9.23.2.35 ReadRegisterBlock()

```
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

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

##### See also

[WriteRegisterBlock\(\)](#)

##### Returns

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

Implements [CameraBase](#).

#### 9.23.2.36 ResetStats()

```
virtual Error ResetStats ( ) [virtual]
```

Implements [CameraBase](#).

#### 9.23.2.37 RestoreFromMemoryChannel()

```
virtual Error RestoreFromMemoryChannel (
    unsigned int channel ) [virtual]
```

Restore the specified current memory channel.

##### Parameters

<i>channel</i>	Memory channel to restore from.
----------------	---------------------------------

## See also

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

## Returns

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

Implements [CameraBase](#).

### 9.23.2.38 RetrieveBuffer()

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

## Parameters

<i>pImage</i>	Pointer to <a href="#">Image</a> object to store image data.
---------------	--

## See also

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

## Returns

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

Implements [CameraBase](#).

### 9.23.2.39 SaveToMemoryChannel()

```
virtual Error SaveToMemoryChannel (
    unsigned int channel ) [virtual]
```

Save the current settings to the specified current memory channel.

## Parameters

<i>channel</i>	Memory channel to save to.
----------------	----------------------------

## See also

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

## Returns

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

Implements [CameraBase](#).

## 9.23.2.40 SetActiveLUTBank()

```
virtual Error SetActiveLUTBank (  
    unsigned int activeBank ) [virtual]
```

Set the LUT bank that will be used.

## Parameters

<i>activeBank</i>	The bank to be set as active.
-------------------	-------------------------------

## Returns

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

Implements [CameraBase](#).

## 9.23.2.41 SetCallback()

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

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

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

**Parameters**

<i>callbackFn</i>	A function to be called when a new image is received.
<i>pCallbackData</i>	A pointer to data that can be passed to the callback function.

**See also**[StartCapture\(\)](#)**Returns**

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

Implements [CameraBase](#).

**9.23.2.42 SetCamera()** [1/2]

```
Error SetCamera (
    CameraBase * camera )
```

**9.23.2.43 SetCamera()** [2/2]

```
Error SetCamera (
    CameraBase * camera,
    const char * filepath = NULL )
```

**9.23.2.44 SetConfiguration()**

```
virtual Error SetConfiguration (
    const FC2Config * pConfig ) [virtual]
```

Set the configuration associated with the camera object.

**Parameters**

<i>pConfig</i>	Pointer to the configuration structure to be used.
----------------	--

**See also**[GetConfiguration\(\)](#)

#### Returns

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

Implements [CameraBase](#).

##### 9.23.2.45 SetEmbeddedImageInfo()

```
virtual Error SetEmbeddedImageInfo (
    EmbeddedImageInfo * pInfo ) [virtual]
```

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

#### Parameters

<i>pInfo</i>	Structure to be used.
--------------	-----------------------

#### See also

[GetEmbeddedImageInfo\(\)](#)

#### Returns

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

Implements [CameraBase](#).

##### 9.23.2.46 SetGPIOPinDirection()

```
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

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

See also

[GetGPIOPinDirection\(\)](#)

Returns

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

Implements [CameraBase](#).

#### 9.23.2.47 SetLUTChannel()

```
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

<i>bank</i>	Bank to set.
<i>channel</i>	Channel to set.
<i>sizeEntries</i>	Number of entries in LUT table to write. This must be the same size as numEntries returned by <a href="#">GetLutInfo()</a> .
<i>pEntries</i>	Array containing LUT entries to write.

See also

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

Returns

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

Implements [CameraBase](#).

#### 9.23.2.48 SetProperty()

```
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. Use [GetPropertyInfo\(\)](#) to query which options are available for a specific property.



## Parameters

<i>pProp</i>	Pointer to the <a href="#">Property</a> structure to be used.
<i>broadcast</i>	Whether the action should be broadcast.

## See also

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

## Returns

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

Implements [CameraBase](#).

## 9.23.2.49 SetStrobe()

```
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

<i>pStrobeControl</i>	Structure providing strobe settings.
<i>broadcast</i>	Whether the action should be broadcast.

## See also

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

## Returns

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

Implements [CameraBase](#).

## 9.23.2.50 SetTriggerDelay()

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

Set the specified trigger delay settings to the camera.

**Parameters**

<i>pTriggerDelay</i>	Structure providing trigger delay settings.
<i>broadcast</i>	Whether the action should be broadcast.

**See also**

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

**Returns**

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

Implements [CameraBase](#).

**9.23.2.51 SetTriggerMode()**

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

Set the specified trigger settings to the camera.

**Parameters**

<i>pTriggerMode</i>	Structure providing trigger mode settings.
<i>broadcast</i>	Whether the action should be broadcast.

**See also**

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

**Returns**

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

Implements [CameraBase](#).

### 9.23.2.52 SetUserBuffers()

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

Specify user allocated buffers to use as image data buffers.

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

#### Parameters

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

#### See also

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

#### Returns

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

Implements [CameraBase](#).

### 9.23.2.53 StartCapture()

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

Starts isochronous image capture.

It will use either the current video mode or the most recently set video mode of the camera. The optional callback function parameter is called on completion of image transfer. When a callback function is specified, the grab mode will determine how images are delivered. If the grab mode has not been set, or has been set to DROP\_FRAMES the default behavior is to requeue images for DMA if they have not been delivered by the time the next image transfer completes. If BUFFER\_FRAMES is specified, the next image in the sequence will be delivered. Note that for the BUFFER\_FRAMES case, if delivery does not keep up with the DMA process, images will be lost. The default behavior is to perform DROP\_FRAMES image delivery Alternatively, the callback parameter can be set to NULL and [RetrieveBuffer\(\)](#) can be called as a blocking call to get the image data.

## Parameters

<i>callbackFn</i>	A function to be called when a new image is received.
<i>pCallbackData</i>	A pointer to data that can be passed to the callback function.

## See also

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

## Returns

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

Implements [CameraBase](#).

## 9.23.2.54 StartSyncCapture()

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

## 9.23.2.55 StopCapture()

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

Stops isochronous image transfer and cleans up all associated resources.

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

## See also

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

## Returns

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

Implements [CameraBase](#).

#### 9.23.2.56 TestGainNode()

```
void TestGainNode ( ) [protected]
```

#### 9.23.2.57 WaitForBufferEvent()

```
virtual Error WaitForBufferEvent (
    Image * pImage,
    unsigned int eventNumber ) [virtual]
```

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

##### Parameters

<i>pImage</i>	Pointer to <a href="#">Image</a> object to store image data.
<i>eventNumber</i>	The event number to wait for.

##### See also

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

##### Returns

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

Implements [CameraBase](#).

#### 9.23.2.58 WriteGVCPMemory()

```
virtual Error WriteGVCPMemory (
    unsigned int address,
    const unsigned char * pBuffer,
    unsigned int length ) [virtual]
```

#### 9.23.2.59 WriteGVCPRegister()

```
virtual Error WriteGVCPRegister (
    unsigned int address,
    unsigned int value,
    bool broadcast = false ) [virtual]
```

**9.23.2.60 WriteGVCPRegisterBlock()**

```
virtual Error WriteGVCPRegisterBlock (
    unsigned int address,
    const unsigned int * pBuffer,
    unsigned int length ) [virtual]
```

**9.23.2.61 WriteRegister()**

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

Write to the specified register on the camera.

**Parameters**

<i>address</i>	DCAM address to be written to.
<i>value</i>	The value to be written.
<i>broadcast</i>	Whether the action should be broadcast.

**See also**

[ReadRegister\(\)](#)

**Returns**

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

Implements [CameraBase](#).

**9.23.2.62 WriteRegisterBlock()**

```
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**

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

See also

[ReadRegisterBlock\(\)](#)

Returns

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

Implements [CameraBase](#).

### 9.23.3 Member Data Documentation

#### 9.23.3.1 m\_busMgr

[BusManager](#) m\_busMgr [protected]

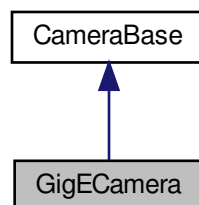
The documentation for this class was generated from the following file:

- [GCCamera.h](#)

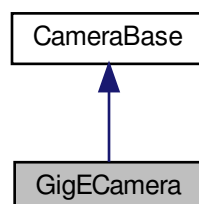
## 9.24 GigECamera Class Reference

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

Inheritance diagram for GigECamera:



Collaboration diagram for GigECamera:



## Public Member Functions

- [GigECamera](#) ()  
*Default constructor.*
- virtual [~GigECamera](#) ()  
*Default destructor.*
- virtual [Error Connect](#) ([PGRGuid](#) \*pGuid=NULL)  
*The following functions are inherited from [CameraBase](#).*
- virtual [Error Disconnect](#) ()  
*Disconnects the camera object from the camera.*
- virtual bool [IsConnected](#) ()  
*Checks if the camera object is connected to a physical camera specified by a GUID.*
- virtual [Error SetCallback](#) ([ImageEventCallback](#) callbackFn, const void \*pCallbackData=NULL)  
*Sets the callback data to be used on completion of image transfer.*
- virtual [Error StartCapture](#) ([ImageEventCallback](#) callbackFn=NULL, const void \*pCallbackData=NULL)  
*Starts isochronous image capture.*
- virtual [Error RetrieveBuffer](#) ([Image](#) \*pImage)  
*Retrieves the the next image object containing the next image.*
- virtual [Error StopCapture](#) ()  
*Stops isochronous image transfer and cleans up all associated resources.*
- virtual [Error WaitForBufferEvent](#) ([Image](#) \*pImage, unsigned int eventNumber)  
*Retrieves the next image event containing the next part of the image.*
- virtual [Error SetUserBuffers](#) (unsigned char \*const pMemBuffers, int size, int numBuffers)  
*Specify user allocated buffers to use as image data buffers.*
- virtual [Error GetConfiguration](#) ([FC2Config](#) \*pConfig)  
*Get the configuration associated with the camera object.*
- virtual [Error SetConfiguration](#) (const [FC2Config](#) \*pConfig)  
*Set the configuration associated with the camera object.*
- virtual [Error GetCameraInfo](#) ([CameraInfo](#) \*pCameraInfo)  
*Retrieves information from the camera such as serial number, model name and other camera information.*
- virtual [Error GetPropertyInfo](#) ([PropertyInfo](#) \*pPropInfo)  
*Retrieves information about the specified camera property.*
- virtual [Error GetProperty](#) ([Property](#) \*pProp)  
*Reads the settings for the specified property from the camera.*
- virtual [Error SetProperty](#) (const [Property](#) \*pProp, bool broadcast=false)  
*Writes the settings for the specified property to the camera.*
- virtual [Error GetGPIOPinDirection](#) (unsigned int pin, unsigned int \*pDirection)  
*Get the GPIO pin direction for the specified pin.*
- virtual [Error SetGPIOPinDirection](#) (unsigned int pin, unsigned int direction, bool broadcast=false)  
*Set the GPIO pin direction for the specified pin.*
- virtual [Error GetTriggerModelInfo](#) ([TriggerModelInfo](#) \*pTriggerModelInfo)  
*Retrieve trigger information from the camera.*
- virtual [Error GetTriggerMode](#) ([TriggerMode](#) \*pTriggerMode)  
*Retrieve current trigger settings from the camera.*
- virtual [Error SetTriggerMode](#) (const [TriggerMode](#) \*pTriggerMode, bool broadcast=false)  
*Set the specified trigger settings to the camera.*
- virtual [Error FireSoftwareTrigger](#) (bool broadcast=false)  
*Fire the software trigger according to the DCAM specifications.*
- virtual [Error GetTriggerDelayInfo](#) ([TriggerDelayInfo](#) \*pTriggerDelayInfo)  
*Retrieve trigger delay information from the camera.*
- virtual [Error GetTriggerDelay](#) ([TriggerDelay](#) \*pTriggerDelay)



- Retrieve current trigger delay settings from the camera.*

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

*Set the specified trigger delay settings to the camera.*
- virtual [Error GetStrobeInfo](#) ([StrobeInfo](#) \*pStrobeInfo)

*Retrieve strobe information from the camera.*
- virtual [Error GetStrobe](#) ([StrobeControl](#) \*pStrobeControl)

*Retrieve current strobe settings from the camera.*
- virtual [Error SetStrobe](#) (const [StrobeControl](#) \*pStrobeControl, bool broadcast=false)

*Set current strobe settings to the camera.*
- virtual [Error GetLUTInfo](#) ([LUTData](#) \*pData)

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

*Query the read/write status of a single LUT bank.*
- virtual [Error GetActiveLUTBank](#) (unsigned int \*pActiveBank)

*Get the LUT bank that is currently being used.*
- virtual [Error SetActiveLUTBank](#) (unsigned int activeBank)

*Set the LUT bank that will be used.*
- virtual [Error EnableLUT](#) (bool on)

*Enable or disable LUT functionality on the camera.*
- virtual [Error GetLUTChannel](#) (unsigned int bank, unsigned int channel, unsigned int sizeEntries, unsigned int \*pEntries)

*Get the LUT channel settings from the camera.*
- virtual [Error SetLUTChannel](#) (unsigned int bank, unsigned int channel, unsigned int sizeEntries, const unsigned int \*pEntries)

*Set the LUT channel settings to the camera.*
- virtual [Error GetMemoryChannel](#) (unsigned int \*pCurrentChannel)

*Retrieve the current memory channel from the camera.*
- virtual [Error SaveToMemoryChannel](#) (unsigned int channel)

*Save the current settings to the specified current memory channel.*
- virtual [Error RestoreFromMemoryChannel](#) (unsigned int channel)

*Restore the specified current memory channel.*
- virtual [Error GetMemoryChannelInfo](#) (unsigned int \*pNumChannels)

*Query the camera for memory channel support.*
- virtual [Error GetEmbeddedImageInfo](#) ([EmbeddedImageInfo](#) \*pInfo)

*Get the current status of the embedded image information register, as well as the availability of each embedded property.*
- virtual [Error SetEmbeddedImageInfo](#) ([EmbeddedImageInfo](#) \*pInfo)

*Sets the on/off values of the embedded image information structure to the camera.*
- virtual [Error WriteRegister](#) (unsigned int address, unsigned int value, bool broadcast=false)

*Write to the specified register on the camera.*
- virtual [Error ReadRegister](#) (unsigned int address, unsigned int \*pValue)

*Read the specified register from the camera.*
- virtual [Error WriteRegisterBlock](#) (unsigned short addressHigh, unsigned int addressLow, const unsigned int \*pBuffer, unsigned int length)

*Write to the specified register block on the camera.*
- virtual [Error ReadRegisterBlock](#) (unsigned short addressHigh, unsigned int addressLow, unsigned int \*pBuffer, unsigned int length)

*Read from the specified register block on the camera.*
- [Error GetCycleTime](#) ([TimeStamp](#) \*timeStamp)

*Returns a TimeStamp struct containing 1394 CYCLE\_TIME information.*
- virtual [Error GetStats](#) ([CameraStats](#) \*pStats)

- virtual [Error ResetStats](#) ()
- virtual [Error RegisterEvent](#) ([EventOptions](#) \*pOpts)
- virtual [Error DeregisterEvent](#) ([EventOptions](#) \*pOpts)
- virtual [Error RegisterAllEvents](#) ([EventOptions](#) \*pOpts)
- virtual [Error DeregisterAllEvents](#) (void)

### Static Public Member Functions

- static [Error StartSyncCapture](#) (unsigned int numCameras, const [GigECamera](#) \*\*ppCameras, const [Image](#)↔  
[EventCallback](#) \*pCallbackFns=NULL, const void \*\*pCallbackdataArray=NULL)  
*StartSyncCapture() with GigE Cameras is not supported.*
- static const char \* [GetRegisterString](#) (unsigned int registerVal)

### GVCP Register Operation

These functions deal with GVCP register operation on the camera.

- virtual [Error WriteGVCPRegister](#) (unsigned int address, unsigned int value, bool broadcast=false)  
*Write a GVCP register.*
- virtual [Error ReadGVCPRegister](#) (unsigned int address, unsigned int \*pValue)  
*Read a GVCP register.*
- virtual [Error WriteGVCPRegisterBlock](#) (unsigned int address, const unsigned int \*pBuffer, unsigned int length)  
*Write a GVCP register block.*
- virtual [Error ReadGVCPRegisterBlock](#) (unsigned int address, unsigned int \*pBuffer, unsigned int length)  
*Read a GVCP register block.*
- virtual [Error WriteGVCPMemory](#) (unsigned int address, const unsigned char \*pBuffer, unsigned int length)  
*Write a GVCP Memory block.*
- virtual [Error ReadGVCPMemory](#) (unsigned int address, unsigned char \*pBuffer, unsigned int length)  
*Read a GVCP memory block.*

### GigE property manipulation

These functions deal with GigE properties.

- virtual [Error GetGigEProperty](#) ([GigEProperty](#) \*pGigEProp)  
*Get the specified GigEProperty.*
- virtual [Error SetGigEProperty](#) (const [GigEProperty](#) \*pGigEProp)  
*Set the specified GigEProperty.*
- virtual [Error DiscoverGigEPacketSize](#) (unsigned int \*packetSize)  
*Discover the largest packet size that works for the network link between the PC and the camera.*

## GigE image settings

These functions deal with GigE image setting.

- virtual [Error QueryGigEImagingMode](#) ([Mode](#) mode, bool \*isSupported)  
*Check if the particular imaging mode is supported by the camera.*
- virtual [Error GetGigEImagingMode](#) ([Mode](#) \*mode)  
*Get the current imaging mode on the camera.*
- virtual [Error SetGigEImagingMode](#) ([Mode](#) mode)  
*Set the current imaging mode to the camera.*
- virtual [Error GetGigEImageSettingsInfo](#) ([GigEImageSettingsInfo](#) \*pInfo)  
*Get information about the image settings possible on the camera.*
- virtual [Error GetGigEImageSettings](#) ([GigEImageSettings](#) \*pImageSettings)  
*Get the current image settings on the camera.*
- virtual [Error SetGigEImageSettings](#) (const [GigEImageSettings](#) \*pImageSettings)  
*Set the image settings specified to the camera.*

## GigE image binning settings

These functions deal with GigE image binning settings.

- virtual [Error GetGigEImageBinningSettings](#) (unsigned int \*horzBinningValue, unsigned int \*vertBinningValue)  
*Get the current binning settings on the camera.*
- virtual [Error SetGigEImageBinningSettings](#) (unsigned int horzBinningValue, unsigned int vertBinningValue)  
*Set the specified binning values to the camera.*

## GigE image stream configuration

These functions deal with GigE image stream configuration.

- virtual [Error GetNumStreamChannels](#) (unsigned int \*numChannels)  
*Get the number of stream channels present on the camera.*
- virtual [Error GetGigEStreamChannelInfo](#) (unsigned int channel, [GigEStreamChannel](#) \*pChannel)  
*Get the stream channel information for the specified channel.*
- virtual [Error SetGigEStreamChannelInfo](#) (unsigned int channel, [GigEStreamChannel](#) \*pChannel)  
*Set the stream channel information for the specified channel.*
- virtual [Error GetGigEConfig](#) ([GigEConfig](#) \*pGigEConfig)  
*Get the current gige config on the camera.*
- virtual [Error SetGigEConfig](#) (const [GigEConfig](#) \*pGigEConfig)  
*Set the gige config specified to the camera.*

## Additional Inherited Members

### 9.24.1 Detailed Description

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

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

Please see [Camera.h](#) for basic functions that this class inherits from.

## 9.24.2 Constructor & Destructor Documentation

### 9.24.2.1 GigECamera()

```
GigECamera ( )
```

Default constructor.

### 9.24.2.2 ~GigECamera()

```
virtual ~GigECamera ( ) [virtual]
```

Default destructor.

## 9.24.3 Member Function Documentation

### 9.24.3.1 Connect()

```
virtual Error Connect (
    PGRGuid * pGuid = NULL ) [virtual]
```

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

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

Implements [CameraBase](#).

### 9.24.3.2 DeregisterAllEvents()

```
virtual Error DeregisterAllEvents (
    void ) [virtual]
```

Implements [CameraBase](#).

#### 9.24.3.3 DeregisterEvent()

```
virtual Error DeregisterEvent (
    EventOptions * pOpts ) [virtual]
```

Implements [CameraBase](#).

#### 9.24.3.4 Disconnect()

```
virtual Error Disconnect ( ) [virtual]
```

Disconnects the camera object from the camera.

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

See also

[Connect\(\)](#)

Returns

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

Implements [CameraBase](#).

#### 9.24.3.5 DiscoverGigEPacketSize()

```
virtual Error DiscoverGigEPacketSize (
    unsigned int * packetSize ) [virtual]
```

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

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

Parameters

<i>packetSize</i>	The maximum packet size supported by the link.
-------------------	--

Returns

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

#### 9.24.3.6 EnableLUT()

```
virtual Error EnableLUT (
    bool on ) [virtual]
```

Enable or disable LUT functionality on the camera.

##### Parameters

<i>on</i>	Whether to enable or disable LUT.
-----------	-----------------------------------

##### See also

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

##### Returns

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

Implements [CameraBase](#).

#### 9.24.3.7 FireSoftwareTrigger()

```
virtual Error FireSoftwareTrigger (
    bool broadcast = false ) [virtual]
```

Fire the software trigger according to the DCAM specifications.

##### Parameters

<i>broadcast</i>	Whether the action should be broadcast.
------------------	---

##### Returns

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

Implements [CameraBase](#).

#### 9.24.3.8 GetActiveLUTBank()

```
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

<i>pActiveBank</i>	The currently active bank.
--------------------	----------------------------

## Returns

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

Implements [CameraBase](#).

## 9.24.3.9 GetCameraInfo()

```
virtual Error GetCameraInfo (
    CameraInfo * pCameraInfo ) [virtual]
```

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

## Parameters

<i>pCameraInfo</i>	Pointer to the camera information structure to be filled.
--------------------	---

## Returns

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

Implements [CameraBase](#).

## 9.24.3.10 GetConfiguration()

```
virtual Error GetConfiguration (
    FC2Config * pConfig ) [virtual]
```

Get the configuration associated with the camera object.

## Parameters

<i>pConfig</i>	Pointer to the configuration structure to be filled.
----------------	--

## See also

[SetConfiguration\(\)](#)

**Returns**

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

Implements [CameraBase](#).

**9.24.3.11 GetCycleTime()**

```
Error GetCycleTime (
    TimeStamp * timeStamp ) [virtual]
```

Returns a Timestamp struct containing 1394 CYCLE\_TIME information.

**Parameters**

<i>registerVal</i>	The register value to query.
--------------------	------------------------------

**Returns**

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

Implements [CameraBase](#).

**9.24.3.12 GetEmbeddedImageInfo()**

```
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**

<i>pInfo</i>	Structure to be filled.
--------------	-------------------------

**See also**

[SetEmbeddedImageInfo\(\)](#)

**Returns**

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

Implements [CameraBase](#).



#### 9.24.3.13 GetGigEConfig()

```
virtual Error GetGigEConfig (
    GigEConfig * pGigEConfig ) [virtual]
```

Get the current gige config on the camera.

##### Parameters

<i>pGigEConfig</i>	Current configuration on camera.
--------------------	----------------------------------

##### Returns

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

#### 9.24.3.14 GetGigEImageBinningSettings()

```
virtual Error GetGigEImageBinningSettings (
    unsigned int * horzBinnningValue,
    unsigned int * vertBinnningValue ) [virtual]
```

Get the current binning settings on the camera.

##### Parameters

<i>horzBinnningValue</i>	Current horizontal binning value.
<i>vertBinnningValue</i>	Current vertical binning value.

##### Returns

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

#### 9.24.3.15 GetGigEImageSettings()

```
virtual Error GetGigEImageSettings (
    GigEImageSettings * pImageSettings ) [virtual]
```

Get the current image settings on the camera.

##### Parameters

<i>pImageSettings</i>	Current image settings on camera.
-----------------------	-----------------------------------

**Returns**

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

**9.24.3.16 GetGigEImageSettingsInfo()**

```
virtual Error GetGigEImageSettingsInfo (
    GigEImageSettingsInfo * pInfo ) [virtual]
```

Get information about the image settings possible on the camera.

**Parameters**

<i>pInfo</i>	<a href="#">Image</a> settings information.
--------------	---

**Returns**

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

**9.24.3.17 GetGigEImagingMode()**

```
virtual Error GetGigEImagingMode (
    Mode * mode ) [virtual]
```

Get the current imaging mode on the camera.

**Parameters**

<i>mode</i>	Current imaging mode on the camera.
-------------	-------------------------------------

**Returns**

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

**9.24.3.18 GetGigEProperty()**

```
virtual Error GetGigEProperty (
    GigEProperty * pGigEProp ) [virtual]
```

Get the specified [GigEProperty](#).

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

## Parameters

<i>pGigEProp</i>	The GigE property to get.
------------------	---------------------------

## Returns

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

## 9.24.3.19 GetGigEStreamChannelInfo()

```
virtual Error GetGigEStreamChannelInfo (
    unsigned int channel,
    GigEStreamChannel * pChannel ) [virtual]
```

Get the stream channel information for the specified channel.

## Parameters

<i>channel</i>	Channel number to use.
<i>pChannel</i>	Stream channel information for the specified channel.

## Returns

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

## 9.24.3.20 GetGPIOPinDirection()

```
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

<i>pin</i>	Pin to get the direction for.
<i>pDirection</i>	Direction of the pin. 0 for input, 1 for output.

## See also

[SetGPIOPinDirection\(\)](#)

**Returns**

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

Implements [CameraBase](#).

**9.24.3.21 GetLUTBankInfo()**

```
virtual Error GetLUTBankInfo (
    unsigned int bank,
    bool * pReadSupported,
    bool * pWriteSupported ) [virtual]
```

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

**Parameters**

<i>bank</i>	The bank to query.
<i>pReadSupported</i>	Whether reading from the bank is supported.
<i>pWriteSupported</i>	Whether writing to the bank is supported.

**Returns**

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

Implements [CameraBase](#).

**9.24.3.22 GetLUTChannel()**

```
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**

<i>bank</i>	Bank to retrieve.
<i>channel</i>	Channel to retrieve.
<i>sizeEntries</i>	Number of entries in LUT table to read.
<i>pEntries</i>	Array to store LUT entries.

## See also

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

## Returns

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

Implements [CameraBase](#).

## 9.24.3.23 GetLUTInfo()

```
virtual Error GetLUTInfo (
    LUTData * pData ) [virtual]
```

Query if LUT support is available on the camera.

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

## Parameters

<i>pData</i>	The LUT structure to be filled.
--------------	---------------------------------

## See also

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

## Returns

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

Implements [CameraBase](#).

## 9.24.3.24 GetMemoryChannel()

```
virtual Error GetMemoryChannel (
    unsigned int * pCurrentChannel ) [virtual]
```

Retrieve the current memory channel from the camera.

## Parameters

<i>pCurrentChannel</i>	Current memory channel.
------------------------	-------------------------

## See also

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

## Returns

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

Implements [CameraBase](#).

## 9.24.3.25 GetMemoryChannelInfo()

```
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

<i>pNumChannels</i>	Number of memory channels supported.
---------------------	--------------------------------------

## See also

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

## Returns

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

Implements [CameraBase](#).

## 9.24.3.26 GetNumStreamChannels()

```
virtual Error GetNumStreamChannels (
    unsigned int * numChannels ) [virtual]
```

Get the number of stream channels present on the camera.

## Parameters

<i>numChannels</i>	Number of stream channels present.
--------------------	------------------------------------

## Returns

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

## 9.24.3.27 GetProperty()

```
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

<i>pProp</i>	Pointer to the <a href="#">Property</a> structure to be filled.
--------------	---

## See also

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

## Returns

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

Implements [CameraBase](#).

## 9.24.3.28 GetPropertyInfo()

```
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**

<i>pPropInfo</i>	Pointer to the <a href="#">PropertyInfo</a> structure to be filled.
------------------	---

**See also**

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

**Returns**

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

Implements [CameraBase](#).

**9.24.3.29 GetRegisterString()**

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

**9.24.3.30 GetStats()**

```
virtual Error GetStats (  
    CameraStats * pStats )    [virtual]
```

Implements [CameraBase](#).

**9.24.3.31 GetStrobe()**

```
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**

<i>pStrobeControl</i>	Structure to receive strobe settings.
-----------------------	---------------------------------------



See also

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

Returns

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

Implements [CameraBase](#).

#### 9.24.3.32 GetStrobeInfo()

```
virtual Error GetStrobeInfo (  
    StrobeInfo * pStrobeInfo ) [virtual]
```

Retrieve strobe information from the camera.

Parameters

<i>pStrobeInfo</i>	Structure to receive strobe information.
--------------------	--

See also

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

Returns

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

Implements [CameraBase](#).

#### 9.24.3.33 GetTriggerDelay()

```
virtual Error GetTriggerDelay (  
    TriggerDelay * pTriggerDelay ) [virtual]
```

Retrieve current trigger delay settings from the camera.

Parameters

<i>pTriggerDelay</i>	Structure to receive trigger delay settings.
----------------------	--

**See also**

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

**Returns**

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

Implements [CameraBase](#).

**9.24.3.34 GetTriggerDelayInfo()**

```
virtual Error GetTriggerDelayInfo (  
    TriggerDelayInfo * pTriggerDelayInfo ) [virtual]
```

Retrieve trigger delay information from the camera.

**Parameters**

<i>pTriggerDelayInfo</i>	Structure to receive trigger delay information.
--------------------------	---

**See also**

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

**Returns**

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

Implements [CameraBase](#).

**9.24.3.35 GetTriggerMode()**

```
virtual Error GetTriggerMode (  
    TriggerMode * pTriggerMode ) [virtual]
```

Retrieve current trigger settings from the camera.

## Parameters

<i>pTriggerMode</i>	Structure to receive trigger mode settings.
---------------------	---

## See also

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

## Returns

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

Implements [CameraBase](#).

## 9.24.3.36 GetTriggerModeInfo()

```
virtual Error GetTriggerModeInfo (
    TriggerModeInfo * pTriggerModeInfo ) [virtual]
```

Retrieve trigger information from the camera.

## Parameters

<i>pTriggerModeInfo</i>	Structure to receive trigger information.
-------------------------	---

## See also

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

## Returns

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

Implements [CameraBase](#).

#### 9.24.3.37 IsConnected()

```
virtual bool IsConnected ( ) [virtual]
```

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

See also

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

Returns

Whether [Connect\(\)](#) was called on the camera object.

Implements [CameraBase](#).

#### 9.24.3.38 QueryGigEImagingMode()

```
virtual Error QueryGigEImagingMode (
    Mode mode,
    bool * isSupported ) [virtual]
```

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

Parameters

<i>mode</i>	The mode to check.
<i>isSupported</i>	Whether the mode is supported.

Returns

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

#### 9.24.3.39 ReadGVCPMemory()

```
virtual Error ReadGVCPMemory (
    unsigned int address,
    unsigned char * pBuffer,
    unsigned int length ) [virtual]
```

Read a GVCP memory block.

Parameters

<i>address</i>	GVCP address to be read from.
<i>pBuffer</i>	Array for data to be read into.
<i>length</i>	Size of array, in quadlets.

**Returns**

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

**9.24.3.40 ReadGVCPRegister()**

```
virtual Error ReadGVCPRegister (
    unsigned int address,
    unsigned int * pValue ) [virtual]
```

Read a GVCP register.

**Parameters**

<i>address</i>	GVCP address to be read from.
<i>pValue</i>	The value that is read.

**Returns**

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

**9.24.3.41 ReadGVCPRegisterBlock()**

```
virtual Error ReadGVCPRegisterBlock (
    unsigned int address,
    unsigned int * pBuffer,
    unsigned int length ) [virtual]
```

Read a GVCP register block.

**Parameters**

<i>address</i>	GVCP address to be read from.
<i>pBuffer</i>	Array for data to be read into.
<i>length</i>	Size of array, in quadlets.

**Returns**

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

**9.24.3.42 ReadRegister()**

```
virtual Error ReadRegister (
    unsigned int address,
    unsigned int * pValue ) [virtual]
```

Read the specified register from the camera.

#### Parameters

<i>address</i>	DCAM address to be read from.
<i>pValue</i>	The value that is read.

#### See also

[WriteRegister\(\)](#)

#### Returns

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

Implements [CameraBase](#).

#### 9.24.3.43 ReadRegisterBlock()

```
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

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

#### See also

[WriteRegisterBlock\(\)](#)

#### Returns

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

Implements [CameraBase](#).

#### 9.24.3.44 RegisterAllEvents()

```
virtual Error RegisterAllEvents (
    EventOptions * pOpts ) [virtual]
```

Implements [CameraBase](#).

#### 9.24.3.45 RegisterEvent()

```
virtual Error RegisterEvent (
    EventOptions * pOpts ) [virtual]
```

Implements [CameraBase](#).

#### 9.24.3.46 ResetStats()

```
virtual Error ResetStats ( ) [virtual]
```

Implements [CameraBase](#).

#### 9.24.3.47 RestoreFromMemoryChannel()

```
virtual Error RestoreFromMemoryChannel (
    unsigned int channel ) [virtual]
```

Restore the specified current memory channel.

##### Parameters

<i>channel</i>	Memory channel to restore from.
----------------	---------------------------------

##### See also

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

##### Returns

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

Implements [CameraBase](#).

#### 9.24.3.48 RetrieveBuffer()

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

##### Parameters

<i>pImage</i>	Pointer to <a href="#">Image</a> object to store image data.
---------------	--

##### See also

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

##### Returns

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

Implements [CameraBase](#).

#### 9.24.3.49 SaveToMemoryChannel()

```
virtual Error SaveToMemoryChannel (
    unsigned int channel ) [virtual]
```

Save the current settings to the specified current memory channel.

##### Parameters

<i>channel</i>	Memory channel to save to.
----------------	----------------------------

##### See also

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



**Returns**

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

Implements [CameraBase](#).

**9.24.3.50 SetActiveLUTBank()**

```
virtual Error SetActiveLUTBank (
    unsigned int activeBank ) [virtual]
```

Set the LUT bank that will be used.

**Parameters**

<i>activeBank</i>	The bank to be set as active.
-------------------	-------------------------------

**Returns**

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

Implements [CameraBase](#).

**9.24.3.51 SetCallback()**

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

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

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

**Parameters**

<i>callbackFn</i>	A function to be called when a new image is received.
<i>pCallbackData</i>	A pointer to data that can be passed to the callback function.

**See also**

[StartCapture\(\)](#)

**Returns**

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

Implements [CameraBase](#).

#### 9.24.3.52 SetConfiguration()

```
virtual Error SetConfiguration (
    const FC2Config * pConfig ) [virtual]
```

Set the configuration associated with the camera object.

##### Parameters

<i>pConfig</i>	Pointer to the configuration structure to be used.
----------------	--

##### See also

[GetConfiguration\(\)](#)

##### Returns

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

Implements [CameraBase](#).

#### 9.24.3.53 SetEmbeddedImageInfo()

```
virtual Error SetEmbeddedImageInfo (
    EmbeddedImageInfo * pInfo ) [virtual]
```

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

##### Parameters

<i>pInfo</i>	Structure to be used.
--------------	-----------------------

##### See also

[GetEmbeddedImageInfo\(\)](#)

##### Returns

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

Implements [CameraBase](#).

#### 9.24.3.54 SetGigEConfig()

```
virtual Error SetGigEConfig (
    const GigEConfig * pGigEConfig ) [virtual]
```

Set the gige config specified to the camera.

## Parameters

<i>pGigEConfig</i>	configuration to set to camera.
--------------------	---------------------------------

## Returns

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

## 9.24.3.55 SetGigEImageBinningSettings()

```
virtual Error SetGigEImageBinningSettings (
    unsigned int horzBinningValue,
    unsigned int vertBinningValue ) [virtual]
```

Set the specified binning values to the camera.

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

## Parameters

<i>horzBinningValue</i>	Horizontal binning value.
<i>vertBinningValue</i>	Vertical binning value.

## Returns

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

## 9.24.3.56 SetGigEImageSettings()

```
virtual Error SetGigEImageSettings (
    const GigEImageSettings * pImageSettings ) [virtual]
```

Set the image settings specified to the camera.

## Parameters

<i>pImageSettings</i>	<a href="#">Image</a> settings to set to camera.
-----------------------	--

## Returns

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

#### 9.24.3.57 SetGigEImagingMode()

```
virtual Error SetGigEImagingMode (  
    Mode mode ) [virtual]
```

Set the current imaging mode to the camera.

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

##### Parameters

<i>mode</i>	Imaging mode to set to the camera.
-------------	------------------------------------

##### Returns

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

#### 9.24.3.58 SetGigEProperty()

```
virtual Error SetGigEProperty (  
    const GigEProperty * pGigEProp ) [virtual]
```

Set the specified [GigEProperty](#).

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

##### Parameters

<i>pGigEProp</i>	The GigE property to set.
------------------	---------------------------

##### Returns

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

#### 9.24.3.59 SetGigEStreamChannelInfo()

```
virtual Error SetGigEStreamChannelInfo (  
    unsigned int channel,  
    GigEStreamChannel * pChannel ) [virtual]
```

Set the stream channel information for the specified channel.

Note that the source UDP port of the stream channel is read-only.

## Parameters

<i>channel</i>	Channel number to use.
<i>pChannel</i>	Stream channel information to use for the specified channel.

## Returns

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

## 9.24.3.60 SetGPIOPinDirection()

```
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

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

## See also

[GetGPIOPinDirection\(\)](#)

## Returns

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

Implements [CameraBase](#).

## 9.24.3.61 SetLUTChannel()

```
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

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

## See also

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

## Returns

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

Implements [CameraBase](#).

## 9.24.3.62 SetProperty()

```
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. Use [GetPropertyInfo\(\)](#) to query which options are available for a specific property.

## Parameters

<i>pProp</i>	Pointer to the <a href="#">Property</a> structure to be used.
<i>broadcast</i>	Whether the action should be broadcast.

## See also

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

## Returns

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

Implements [CameraBase](#).

### 9.24.3.63 SetStrobe()

```
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

<i>pStrobeControl</i>	Structure providing strobe settings.
<i>broadcast</i>	Whether the action should be broadcast.

#### See also

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

#### Returns

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

Implements [CameraBase](#).

### 9.24.3.64 SetTriggerDelay()

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

Set the specified trigger delay settings to the camera.

#### Parameters

<i>pTriggerDelay</i>	Structure providing trigger delay settings.
<i>broadcast</i>	Whether the action should be broadcast.

#### See also

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

**Returns**

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

Implements [CameraBase](#).

**9.24.3.65 SetTriggerMode()**

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

Set the specified trigger settings to the camera.

**Parameters**

<i>pTriggerMode</i>	Structure providing trigger mode settings.
<i>broadcast</i>	Whether the action should be broadcast.

**See also**

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

**Returns**

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

Implements [CameraBase](#).

**9.24.3.66 SetUserBuffers()**

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

Specify user allocated buffers to use as image data buffers.

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



## Parameters

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

## See also

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

## Returns

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

Implements [CameraBase](#).

## 9.24.3.67 StartCapture()

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

Starts isochronous image capture.

It will use either the current video mode or the most recently set video mode of the camera. The optional callback function parameter is called on completion of image transfer. When a callback function is specified, the grab mode will determine how images are delivered. If the grab mode has not been set, or has been set to `DROP_FRAMES` the default behavior is to requeue images for DMA if they have not been delivered by the time the next image transfer completes. If `BUFFER_FRAMES` is specified, the next image in the sequence will be delivered. Note that for the `BUFFER_FRAMES` case, if delivery does not keep up with the DMA process, images will be lost. The default behavior is to perform `DROP_FRAMES` image delivery. Alternatively, the callback parameter can be set to `NULL` and [RetrieveBuffer\(\)](#) can be called as a blocking call to get the image data.

## Parameters

<i>callbackFn</i>	A function to be called when a new image is received.
<i>pCallbackData</i>	A pointer to data that can be passed to the callback function.

## See also

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

**Returns**

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

Implements [CameraBase](#).

**9.24.3.68 StartSyncCapture()**

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

[StartSyncCapture\(\)](#) with GigE Cameras is not supported.

This function has been deprecated and will be removed in a future version of FlyCapture.

**9.24.3.69 StopCapture()**

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

Stops isochronous image transfer and cleans up all associated resources.

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

**See also**

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

**Returns**

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

Implements [CameraBase](#).

**9.24.3.70 WaitForBufferEvent()**

```
virtual Error WaitForBufferEvent (
    Image * pImage,
    unsigned int eventNumber ) [virtual]
```

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

## Parameters

<i>pImage</i>	Pointer to <a href="#">Image</a> object to store image data.
<i>eventNumber</i>	The event number to wait for.

## See also

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

## Returns

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

Implements [CameraBase](#).

## 9.24.3.71 WriteGVCPMemory()

```
virtual Error WriteGVCPMemory (
    unsigned int address,
    const unsigned char * pBuffer,
    unsigned int length ) [virtual]
```

Write a GVCP Memory block.

## Parameters

<i>address</i>	GVCP address to be write to.
<i>pBuffer</i>	Array containing data to be written in increments.
<i>length</i>	Size of array, in quadlets.

## Returns

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

## 9.24.3.72 WriteGVCPRegister()

```
virtual Error WriteGVCPRegister (
    unsigned int address,
    unsigned int value,
    bool broadcast = false ) [virtual]
```

Write a GVCP register.

**Parameters**

<i>address</i>	GVCP address to be written to.
<i>value</i>	The value to be written.
<i>broadcast</i>	Whether the action should be broadcast.

**Returns**

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

**9.24.3.73 WriteGVCPRegisterBlock()**

```
virtual Error WriteGVCPRegisterBlock (
    unsigned int address,
    const unsigned int * pBuffer,
    unsigned int length ) [virtual]
```

Write a GVCP register block.

**Parameters**

<i>address</i>	GVCP address to be write to.
<i>pBuffer</i>	Array containing data to be written.
<i>length</i>	Size of array, in quadlets.

**Returns**

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

**9.24.3.74 WriteRegister()**

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

Write to the specified register on the camera.

**Parameters**

<i>address</i>	DCAM address to be written to.
<i>value</i>	The value to be written.
<i>broadcast</i>	Whether the action should be broadcast.

See also

[ReadRegister\(\)](#)

Returns

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

Implements [CameraBase](#).

#### 9.24.3.75 WriteRegisterBlock()

```
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

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

See also

[ReadRegisterBlock\(\)](#)

Returns

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

Implements [CameraBase](#).

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

- [GigECamera.h](#)

## 9.25 GigEConfig Struct Reference

Configuration for a GigE camera.

## Public Member Functions

- [GigEConfig](#) ()

## Public Attributes

- bool [enablePacketResend](#)  
*Turn on/off packet resend functionality.*
- unsigned int [registerTimeoutRetries](#)  
*Number of retries to perform when a register read/write timeout is received by the library.*
- unsigned int [registerTimeout](#)  
*Register read/write timeout value, in microseconds.*

### 9.25.1 Detailed Description

Configuration for a GigE camera.

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

### 9.25.2 Constructor & Destructor Documentation

#### 9.25.2.1 GigEConfig()

```
GigEConfig ( ) [inline]
```

### 9.25.3 Member Data Documentation

#### 9.25.3.1 enablePacketResend

```
bool enablePacketResend
```

Turn on/off packet resend functionality.

#### 9.25.3.2 registerTimeout

```
unsigned int registerTimeout
```

Register read/write timeout value, in microseconds.

The default value is dependent on the interface type.

### 9.25.3.3 registerTimeoutRetries

```
unsigned int registerTimeoutRetries
```

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

The default value is 0.

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

- [FlyCapture2Defs.h](#)

## 9.26 GigEImageSettings Struct Reference

[Image](#) settings for a GigE camera.

### Public Member Functions

- [GigEImageSettings](#) ()

### Public Attributes

- unsigned int [offsetX](#)  
*Horizontal image offset.*
- unsigned int [offsetY](#)  
*Vertical image offset.*
- unsigned int [width](#)  
*Width of image.*
- unsigned int [height](#)  
*Height of image.*
- [PixelFormat](#) [pixelFormat](#)  
*Pixel format of image.*
- unsigned int [reserved](#) [8]  
*Reserved for future use.*

### 9.26.1 Detailed Description

[Image](#) settings for a GigE camera.

### 9.26.2 Constructor & Destructor Documentation

#### 9.26.2.1 GigEImageSettings()

```
GigEImageSettings ( ) [inline]
```

### 9.26.3 Member Data Documentation

#### 9.26.3.1 height

`unsigned int height`

Height of image.

#### 9.26.3.2 offsetX

`unsigned int offsetX`

Horizontal image offset.

#### 9.26.3.3 offsetY

`unsigned int offsetY`

Vertical image offset.

#### 9.26.3.4 pixelFormat

`PixelFormat pixelFormat`

Pixel format of image.

#### 9.26.3.5 reserved

`unsigned int reserved[8]`

Reserved for future use.



#### 9.26.3.6 width

```
unsigned int width
```

Width of image.

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

- [FlyCapture2Defs.h](#)

## 9.27 GigImageSettingsInfo Struct Reference

Format 7 information for a single mode.

### Public Member Functions

- [GigImageSettingsInfo](#) ()

### Public Attributes

- unsigned int [maxWidth](#)  
*Maximum image width.*
- unsigned int [maxHeight](#)  
*Maximum image height.*
- unsigned int [offsetHStepSize](#)  
*Horizontal step size for the offset.*
- unsigned int [offsetVStepSize](#)  
*Vertical step size for the offset.*
- unsigned int [imageHStepSize](#)  
*Horizontal step size for the image.*
- unsigned int [imageVStepSize](#)  
*Vertical step size for the image.*
- unsigned int [pixelFormatBitField](#)  
*Supported pixel formats in a bit field.*
- unsigned int [vendorPixelFormatBitField](#)  
*Vendor unique pixel formats in a bit field.*
- unsigned int [reserved](#) [16]  
*Reserved for future use.*

### 9.27.1 Detailed Description

Format 7 information for a single mode.

### 9.27.2 Constructor & Destructor Documentation

#### 9.27.2.1 GigEImageSettingsInfo()

```
GigEImageSettingsInfo ( ) [inline]
```

### 9.27.3 Member Data Documentation

#### 9.27.3.1 imageHStepSize

```
unsigned int imageHStepSize
```

Horizontal step size for the image.

#### 9.27.3.2 imageVStepSize

```
unsigned int imageVStepSize
```

Vertical step size for the image.

#### 9.27.3.3 maxHeight

```
unsigned int maxHeight
```

Maximum image height.

#### 9.27.3.4 maxWidth

```
unsigned int maxWidth
```

Maximum image width.

#### 9.27.3.5 offsetHStepSize

```
unsigned int offsetHStepSize
```

Horizontal step size for the offset.

#### 9.27.3.6 offsetVStepSize

`unsigned int offsetVStepSize`

Vertical step size for the offset.

#### 9.27.3.7 pixelFormatBitField

`unsigned int pixelFormatBitField`

Supported pixel formats in a bit field.

#### 9.27.3.8 reserved

`unsigned int reserved[16]`

Reserved for future use.

#### 9.27.3.9 vendorPixelFormatBitField

`unsigned int vendorPixelFormatBitField`

Vendor unique pixel formats in a bit field.

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

- [FlyCapture2Defs.h](#)

## 9.28 GigEProperty Struct Reference

A GigE property.

### Public Attributes

- [GigEPropertyType propType](#)  
*The type of property.*
- bool [isReadable](#)  
*Whether the property is readable.*
- bool [isWritable](#)  
*Whether the property is writable.*
- unsigned int [min](#)  
*Minimum value.*
- unsigned int [max](#)  
*Maximum value.*
- unsigned int [value](#)  
*Current value.*

### 9.28.1 Detailed Description

A GigE property.

### 9.28.2 Member Data Documentation

#### 9.28.2.1 isReadable

`bool isReadable`

Whether the property is readable.

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

#### 9.28.2.2 isWritable

`bool isWritable`

Whether the property is writable.

#### 9.28.2.3 max

`unsigned int max`

Maximum value.

#### 9.28.2.4 min

`unsigned int min`

Minimum value.

#### 9.28.2.5 propType

`GigEPropertyType propType`

The type of property.

## 9.28.2.6 value

unsigned int value

Current value.

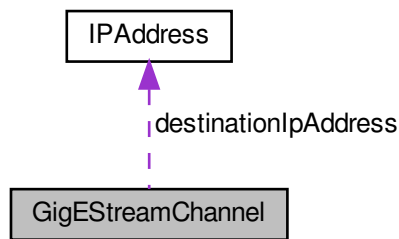
The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

## 9.29 GigEStreamChannel Struct Reference

Information about a single GigE stream channel.

Collaboration diagram for GigEStreamChannel:



### Public Member Functions

- [GigEStreamChannel](#) ()

### Public Attributes

- unsigned int [networkInterfaceIndex](#)  
*Network interface index used (or to use).*
- unsigned int [hostPort](#)  
*Host port on the PC where the camera will send the data stream.*
- bool [doNotFragment](#)  
*Disable IP fragmentation of packets.*
- unsigned int [packetSize](#)  
*Packet size, in bytes.*
- unsigned int [interPacketDelay](#)  
*Inter packet delay, in timestamp counter units.*
- [IPAddress](#) [destinationIpAddress](#)  
*Destination IP address.*
- unsigned int [sourcePort](#)  
*Source UDP port of the stream channel.*

### 9.29.1 Detailed Description

Information about a single GigE stream channel.

### 9.29.2 Constructor & Destructor Documentation

#### 9.29.2.1 GigEStreamChannel()

```
GigEStreamChannel ( ) [inline]
```

### 9.29.3 Member Data Documentation

#### 9.29.3.1 destinationIpAddress

```
IPAddress destinationIpAddress
```

Destination IP address.

It can be a multicast or unicast address.

#### 9.29.3.2 doNotFragment

```
bool doNotFragment
```

Disable IP fragmentation of packets.

#### 9.29.3.3 hostPort

```
unsigned int hostPort
```

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

#### 9.29.3.4 interPacketDelay

```
unsigned int interPacketDelay
```

Inter packet delay, in timestamp counter units.

#### 9.29.3.5 networkInterfaceIndex

```
unsigned int networkInterfaceIndex
```

Network interface index used (or to use).

#### 9.29.3.6 packetSize

```
unsigned int packetSize
```

Packet size, in bytes.

#### 9.29.3.7 sourcePort

```
unsigned int sourcePort
```

Source UDP port of the stream channel.

Read only.

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

- [FlyCapture2Defs.h](#)

## 9.30 H264Option Struct Reference

Options for saving H264 files.

### Public Member Functions

- [H264Option](#) ()

### Public Attributes

- float [frameRate](#)  
*Frame rate of the stream.*
- unsigned int [width](#)  
*Width of source image.*
- unsigned int [height](#)  
*Height of source image.*
- unsigned int [bitrate](#)  
*Bitrate to encode at.*
- unsigned int [reserved](#) [256]  
*Reserved for future use.*

### 9.30.1 Detailed Description

Options for saving H264 files.

### 9.30.2 Constructor & Destructor Documentation

#### 9.30.2.1 H264Option()

```
H264Option ( ) [inline]
```

### 9.30.3 Member Data Documentation

#### 9.30.3.1 bitrate

```
unsigned int bitrate
```

Bitrate to encode at.

#### 9.30.3.2 frameRate

```
float frameRate
```

Frame rate of the stream.

#### 9.30.3.3 height

```
unsigned int height
```

Height of source image.

#### 9.30.3.4 reserved

```
unsigned int reserved[256]
```

Reserved for future use.



## 9.30.3.5 width

```
unsigned int width
```

Width of source image.

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

- [FlyCapture2VideoDefs.h](#)

## 9.31 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 int rows, unsigned int cols, unsigned int stride, unsigned char \*pData, unsigned int dataSize, unsigned int receivedDataSize, [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](#) pixel↔  
Format, [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 [Error SetBlockId](#) (const unsigned int blockId)  
*Set the block id of the [Image](#) object.*
- virtual unsigned int [GetBlockId](#) ()

- get the block id 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 unsigned int [GetReceivedDataSize](#) () const
  - Get the size of the compressed data, 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](#) [Save](#) (const char \*pFilename, [BMPOption](#) \*pOption)
  - Save the image to the specified file name with the options specified.*
  - virtual [Error](#) [Convert](#) ([PixelFormat](#) format, [Image](#) \*pDestImage) const
  - Converts the current image buffer to the specified output format and stores the result in the specified image.*

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

### Static Public Member Functions

- static [Error SetDefaultColorProcessing](#) ([ColorProcessingAlgorithm](#) defaultMethod)  
*Set the default color processing algorithm.*
- static [ColorProcessingAlgorithm GetDefaultColorProcessing](#) ()  
*Get the default color processing algorithm.*
- static [Error SetDefaultOutputFormat](#) ([PixelFormat](#) format)  
*Set the default output pixel format.*
- static [PixelFormat GetDefaultOutputFormat](#) ()  
*Get the default output pixel format.*
- static unsigned int [DetermineBitsPerPixel](#) ([PixelFormat](#) format)  
*Calculate the bits per pixel for the specified pixel format.*

### Friends

- class [Iso](#)

## 9.31.1 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.

## 9.31.2 Constructor & Destructor Documentation

### 9.31.2.1 [Image](#)() [1/6]

[Image](#) ( )

Default constructor.

### 9.31.2.2 [Image](#)() [2/6]

```
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

<i>rows</i>	Rows in the image.
<i>cols</i>	Columns in the image.
<i>stride</i>	Stride of the image buffer.
<i>pData</i>	Pointer to the image buffer.
<i>dataSize</i>	Size of the image buffer.
<i>format</i>	Pixel format.
<i>bayerFormat</i>	Format of the Bayer tiled raw image.

## 9.31.2.3 Image() [3/6]

```
Image (
    unsigned int rows,
    unsigned int cols,
    unsigned int stride,
    unsigned char * pData,
    unsigned int dataSize,
    unsigned int receivedDataSize,
    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

<i>rows</i>	Rows in the image.
<i>cols</i>	Columns in the image.
<i>stride</i>	Stride of the image buffer.
<i>pData</i>	Pointer to the image buffer.
<i>dataSize</i>	Size of the image buffer.
<i>receivedDataSize</i>	Actual size of data.
<i>format</i>	Pixel format.
<i>bayerFormat</i>	Format of the Bayer tiled raw image.

## 9.31.2.4 Image() [4/6]

```
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

<i>pData</i>	Pointer to the image buffer.
<i>dataSize</i>	Size of the image buffer.

## 9.31.2.5 Image() [5/6]

```
Image (
    unsigned int rows,
    unsigned int cols,
    PixelFormat format,
    BayerTileFormat bayerFormat = NONE )
```

Construct an [Image](#) object with the specified arguments.

## Parameters

<i>rows</i>	Rows in the image.
<i>cols</i>	Columns in the image.
<i>format</i>	Pixel format.
<i>bayerFormat</i>	Format of the Bayer tiled raw image.

## 9.31.2.6 Image() [6/6]

```
Image (
    const Image & image )
```

Copy constructor.

Both images will point to the same image buffer internally.

## 9.31.2.7 ~Image()

```
virtual ~Image ( ) [virtual]
```

Default destructor.

The internal image buffer will be released if there are no other [Image](#) objects holding a reference to it. This will also allow the buffer to be requested internally.

## 9.31.3 Member Function Documentation

### 9.31.3.1 CalculateStatistics()

```
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

<i>pStatistics</i>	The <a href="#">ImageStatistics</a> object to hold the statistics.
--------------------	--

#### Returns

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

### 9.31.3.2 Convert() [1/2]

```
virtual Error Convert (
    PixelFormat format,
    Image * pDestImage ) const [virtual]
```

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

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

#### Parameters

<i>format</i>	Output format of the converted image.
<i>pDestImage</i>	Destination image.

#### Returns

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

### 9.31.3.3 Convert() [2/2]

```
virtual Error Convert (
    Image * pDestImage ) const [virtual]
```

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

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

## Parameters

<i>pDestImage</i>	Destination image.
-------------------	--------------------

## Returns

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

## 9.31.3.4 DeepCopy()

```
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

<i>pImage</i>	The <a href="#">Image</a> to copy the data from.
---------------	--

## Returns

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

## 9.31.3.5 DetermineBitsPerPixel()

```
static unsigned int DetermineBitsPerPixel (
    PixelFormat format ) [static]
```

Calculate the bits per pixel for the specified pixel format.

## Parameters

<i>format</i>	The pixel format.
---------------	-------------------

## Returns

The bits per pixel.

#### 9.31.3.6 GetBayerTileFormat()

```
virtual BayerTileFormat GetBayerTileFormat ( ) const [virtual]
```

Get the Bayer tile format of the image.

##### Returns

The Bayer tile format.

#### 9.31.3.7 GetBitsPerPixel()

```
virtual unsigned int GetBitsPerPixel ( ) const [virtual]
```

Get the bits per pixel of the image.

##### Returns

The bits per pixel.

#### 9.31.3.8 GetBlockId()

```
virtual unsigned int GetBlockId ( ) [virtual]
```

get the block id of the [Image](#) object.

##### Returns

The blockId assigned to the image.

#### 9.31.3.9 GetColorProcessing()

```
virtual ColorProcessingAlgorithm GetColorProcessing ( ) const [virtual]
```

Get the current color processing algorithm.

##### See also

[SetColorProcessing\(\)](#)

##### Returns

The current color processing algorithm.



#### 9.31.3.10 GetCols()

```
virtual unsigned int GetCols ( ) const [virtual]
```

Get the number of columns in the image.

##### Returns

The number of columns.

#### 9.31.3.11 GetData() [1/2]

```
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.

#### 9.31.3.12 GetData() [2/2]

```
virtual unsigned char* const GetData ( ) const [virtual]
```

#### 9.31.3.13 GetDataSize()

```
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.

#### 9.31.3.14 GetDefaultColorProcessing()

```
static ColorProcessingAlgorithm GetDefaultColorProcessing ( ) [static]
```

Get the default color processing algorithm.

See also

[SetDefaultColorProcessing\(\)](#)

Returns

The default color processing algorithm.

#### 9.31.3.15 GetDefaultOutputFormat()

```
static PixelFormat GetDefaultOutputFormat ( ) [static]
```

Get the default output pixel format.

See also

[SetDefaultOutputFormat\(\)](#)

Returns

The default pixel format.

#### 9.31.3.16 GetDimensions()

```
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

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

#### 9.31.3.17 GetMetadata()

```
virtual ImageMetadata GetMetadata ( ) const [virtual]
```

Get the metadata associated with the image.

This includes embedded image information.

##### Returns

Metadata associated with the image.

#### 9.31.3.18 GetPixelFormat()

```
virtual PixelFormat GetPixelFormat ( ) const [virtual]
```

Get the current pixel format.

##### Returns

The current pixel format.

#### 9.31.3.19 GetReceivedDataSize()

```
virtual unsigned int GetReceivedDataSize ( ) const [virtual]
```

Get the size of the compressed data, in bytes.

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

##### Returns

The size of the compressed data, in bytes. 0 when camera not sending compressed data.

#### 9.31.3.20 GetRows()

```
virtual unsigned int GetRows ( ) const [virtual]
```

Get the number of rows in the image.

##### Returns

The number of rows.

#### 9.31.3.21 GetStride()

```
virtual unsigned int GetStride ( ) const [virtual]
```

Get the stride in the image.

##### Returns

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

#### 9.31.3.22 GetTimeStamp()

```
virtual TimeStamp GetTimeStamp ( ) const [virtual]
```

Get the timestamp data associated with the image.

##### Returns

Timestamp data associated with the image.

#### 9.31.3.23 operator()( )

```
virtual unsigned char* operator() (
    unsigned int row,
    unsigned int col ) [virtual]
```

Indexing operator.

##### Parameters

<i>row</i>	The row of the pixel to return.
<i>col</i>	The column of the pixel to return.

**Returns**

The address of the specified byte from the image data.

**9.31.3.24 operator=()**

```
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**

<i>image</i>	The image to copy from.
--------------	-------------------------

**9.31.3.25 operator[]()**

```
virtual unsigned char* operator[] (
    unsigned int index ) [virtual]
```

Indexing operator.

**Parameters**

<i>index</i>	The index of the byte to return.
--------------	----------------------------------

**Returns**

The address of the specified byte from the image data.

**9.31.3.26 ReleaseBuffer()**

```
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.

**9.31.3.27 Save()** [1/8]

```
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**

<i>pFilename</i>	Filename to save image with.
<i>format</i>	File format to save in.

**Returns**

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

**9.31.3.28 Save()** [2/8]

```
virtual Error Save (
    const char * pFilename,
    PNGOption * pOption ) [virtual]
```

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

**Parameters**

<i>pFilename</i>	Filename to save image with.
<i>pOption</i>	Options to use while saving image.

**Returns**

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

**9.31.3.29 Save()** [3/8]

```
virtual Error Save (
    const char * pFilename,
    PPMOption * pOption ) [virtual]
```

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

**Parameters**

<i>pFilename</i>	Filename to save image with.
<i>pOption</i>	Options to use while saving image.

**Returns**

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

**9.31.3.30 Save()** [4/8]

```
virtual Error Save (  
    const char * pFilename,  
    PGMOption * pOption ) [virtual]
```

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

**Parameters**

<i>pFilename</i>	Filename to save image with.
<i>pOption</i>	Options to use while saving image.

**Returns**

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

**9.31.3.31 Save()** [5/8]

```
virtual Error Save (  
    const char * pFilename,  
    TIFFOption * pOption ) [virtual]
```

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

**Parameters**

<i>pFilename</i>	Filename to save image with.
<i>pOption</i>	Options to use while saving image.

**Returns**

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

**9.31.3.32 Save()** [6/8]

```
virtual Error Save (  
    const char * pFilename,  
    JPEGOption * pOption ) [virtual]
```

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

**Parameters**

<i>pFilename</i>	Filename to save image with.
<i>pOption</i>	Options to use while saving image.

**Returns**

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

**9.31.3.33 Save()** [7/8]

```
virtual Error Save (  
    const char * pFilename,  
    JPG2Option * pOption ) [virtual]
```

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

**Parameters**

<i>pFilename</i>	Filename to save image with.
<i>pOption</i>	Options to use while saving image.

**Returns**

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

**9.31.3.34 Save()** [8/8]

```
virtual Error Save (  
    const char * pFilename,  
    BMPOption * pOption ) [virtual]
```

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

**Parameters**

<i>pFilename</i>	Filename to save image with.
<i>pOption</i>	Options to use while saving image.

**Returns**

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



#### 9.31.3.35 SetBlockId()

```
virtual Error SetBlockId (
    const unsigned int blockId ) [virtual]
```

Set the block id of the [Image](#) object.

##### Parameters

<i>blockId</i>	The blockId to assign to the image.
----------------	-------------------------------------

#### 9.31.3.36 SetColorProcessing()

```
virtual Error SetColorProcessing (
    ColorProcessingAlgorithm colorProc ) [virtual]
```

Set the color processing algorithm.

This should be set on the input [Image](#) object.

##### Parameters

<i>colorProc</i>	The color processing algorithm to use.
------------------	--

##### See also

[GetColorProcessing\(\)](#)

##### Returns

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

#### 9.31.3.37 SetData()

```
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

<i>pData</i>	Pointer to the image buffer.
<i>dataSize</i>	Size of the image buffer.

## 9.31.3.38 SetDefaultColorProcessing()

```
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

<i>defaultMethod</i>	The color processing algorithm to set.
----------------------	--

## See also

[GetDefaultColorProcessing\(\)](#)

## Returns

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

## 9.31.3.39 SetDefaultOutputFormat()

```
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

<i>format</i>	The output pixel format to set.
---------------	---------------------------------

See also

[GetDefaultOutputFormat\(\)](#)

Returns

The default color processing algorithm.

#### 9.31.3.40 SetDimensions()

```
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

<i>rows</i>	Number of rows to set.
<i>cols</i>	Number of cols to set.
<i>stride</i>	Stride to set.
<i>pixelFormat</i>	Pixel format to set.
<i>bayerFormat</i>	Bayer tile format to set.

See also

[GetDimensions\(\)](#)

Returns

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

### 9.31.4 Friends And Related Function Documentation

#### 9.31.4.1 Iso

```
friend class Iso [friend]
```

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

- [Image.h](#)

## 9.32 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.*

### 9.32.1 Detailed Description

Metadata related to an image.

### 9.32.2 Constructor & Destructor Documentation

#### 9.32.2.1 ImageMetadata()

```
ImageMetadata ( ) [inline]
```

### 9.32.3 Member Data Documentation

#### 9.32.3.1 embeddedBrightness

`unsigned int embeddedBrightness`

Embedded brightness.

#### 9.32.3.2 embeddedExposure

`unsigned int embeddedExposure`

Embedded exposure.

#### 9.32.3.3 embeddedFrameCounter

`unsigned int embeddedFrameCounter`

Embedded frame counter.

#### 9.32.3.4 embeddedGain

`unsigned int embeddedGain`

Embedded gain.

#### 9.32.3.5 embeddedGPIOPinState

`unsigned int embeddedGPIOPinState`

Embedded GPIO pin state.

#### 9.32.3.6 embeddedROIPosition

```
unsigned int embeddedROIPosition
```

Embedded ROI position.

#### 9.32.3.7 embeddedShutter

```
unsigned int embeddedShutter
```

Embedded shutter.

#### 9.32.3.8 embeddedStrobePattern

```
unsigned int embeddedStrobePattern
```

Embedded strobe pattern.

#### 9.32.3.9 embeddedTimeStamp

```
unsigned int embeddedTimeStamp
```

Embedded timestamp.

#### 9.32.3.10 embeddedWhiteBalance

```
unsigned int embeddedWhiteBalance
```

Embedded white balance.

#### 9.32.3.11 reserved

```
unsigned int reserved[31]
```

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

## 9.33 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](#)

### 9.33.1 Detailed Description

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

### 9.33.2 Member Enumeration Documentation

#### 9.33.2.1 StatisticsChannel

enum [StatisticsChannel](#)

Channels that allow statistics to be calculated.

#### Enumerator

GREY	
RED	
GREEN	
BLUE	
HUE	
SATURATION	
LIGHTNESS	
NUM_STATISTICS_CHANNELS	

### 9.33.3 Constructor & Destructor Documentation

#### 9.33.3.1 ImageStatistics() [1/2]

[ImageStatistics](#) ( )

Default constructor.



### 9.33.3.2 ~ImageStatistics()

```
virtual ~ImageStatistics ( ) [virtual]
```

Default destructor.

### 9.33.3.3 ImageStatistics() [2/2]

```
ImageStatistics (
    const ImageStatistics & other )
```

Copy constructor.

## 9.33.4 Member Function Documentation

### 9.33.4.1 DisableAll()

```
Error DisableAll ( )
```

Disable all channels.

#### Returns

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

### 9.33.4.2 EnableAll()

```
Error EnableAll ( )
```

Enable all channels.

#### Returns

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

#### 9.33.4.3 EnableGreyOnly()

```
Error EnableGreyOnly ( )
```

Enable only the grey channel.

##### Returns

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

#### 9.33.4.4 EnableHSLOnly()

```
Error EnableHSLOnly ( )
```

Enable only the HSL channels.

##### Returns

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

#### 9.33.4.5 EnableRGBOnly()

```
Error EnableRGBOnly ( )
```

Enable only the RGB channels.

##### Returns

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

#### 9.33.4.6 GetChannelStatus()

```
Error GetChannelStatus (
    StatisticsChannel channel,
    bool * pEnabled ) const
```

Get the status of a statistics channel.

##### Parameters

<i>channel</i>	The statistics channel.
<i>pEnabled</i>	Whether the channel is enabled.

See also

[SetChannelStatus\(\)](#)

Returns

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

#### 9.33.4.7 GetHistogram()

```
Error GetHistogram (
    StatisticsChannel channel,
    int ** ppHistogram ) const
```

Get the histogram for the image.

Parameters

<i>channel</i>	The statistics channel.
<i>ppHistogram</i>	Pointer to an array containing the histogram.

Returns

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

#### 9.33.4.8 GetMean()

```
Error GetMean (
    StatisticsChannel channel,
    float * pPixelValueMean ) const
```

Get the mean of the image.

Parameters

<i>channel</i>	The statistics channel.
<i>pPixelValueMean</i>	The mean of the image.

Returns

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

#### 9.33.4.9 GetNumPixelValues()

```
Error GetNumPixelValues (
    StatisticsChannel channel,
    unsigned int * pNumPixelValues ) const
```

Get the number of unique pixel values in the image.

##### Parameters

<i>channel</i>	The statistics channel.
<i>pNumPixelValues</i>	The number of unique pixel values.

##### Returns

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

#### 9.33.4.10 GetPixelValueRange()

```
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

<i>channel</i>	The statistics channel.
<i>pPixelValueMin</i>	The minimum pixel value.
<i>pPixelValueMax</i>	The maximum pixel value.

##### Returns

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

#### 9.33.4.11 GetRange()

```
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

<i>channel</i>	The statistics channel.
<i>pMin</i>	The minimum possible value.
<i>pMax</i>	The maximum possible value.

## Returns

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

## 9.33.4.12 GetStatistics()

```
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

<i>channel</i>	The statistics channel.
<i>pRangeMin</i>	The minimum possible value.
<i>pRangeMax</i>	The maximum possible value.
<i>pPixelValueMin</i>	The minimum pixel value.
<i>pPixelValueMax</i>	The maximum pixel value.
<i>pNumPixelValues</i>	The number of unique pixel values.
<i>pPixelValueMean</i>	The mean of the image.
<i>ppHistogram</i>	Pointer to an array containing the histogram.

## Returns

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

## 9.33.4.13 operator=()

```
ImageStatistics& operator= (
    const ImageStatistics & other )
```

Assignment operator.

## Parameters

<i>other</i>	The <a href="#">ImageStatistics</a> object to copy from.
--------------	--

## 9.33.4.14 SetChannelStatus()

```
Error SetChannelStatus (
    StatisticsChannel channel,
    bool enabled )
```

Set the status of a statistics channel.

## Parameters

<i>channel</i>	The statistics channel.
<i>enabled</i>	Whether the channel should be enabled.

## See also

[GetChannelStatus\(\)](#)

## Returns

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

## 9.33.5 Friends And Related Function Documentation

## 9.33.5.1 ImageStatsCalculator

```
friend class ImageStatsCalculator [friend]
```

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

- [ImageStatistics.h](#)

## 9.34 Internal Class Reference

## Static Public Member Functions

- static void \* [GetInternal](#) (unsigned int index)

### 9.34.1 Member Function Documentation

#### 9.34.1.1 GetInternal()

```
static void* GetInternal (
    unsigned int index ) [static]
```

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

- [Internal.h](#)

## 9.35 IPAddress Struct Reference

IPv4 address.

### Public Member Functions

- [IPAddress](#) ()
- [IPAddress](#) (unsigned int ipAddressVal)
- bool [operator==](#) (const [IPAddress](#) &address) const  
*Equality operator.*
- bool [operator!=](#) (const [IPAddress](#) &address)  
*Inequality operator.*

### Public Attributes

- unsigned char [octets](#) [4]

### 9.35.1 Detailed Description

IPv4 address.

### 9.35.2 Constructor & Destructor Documentation

#### 9.35.2.1 [IPAddress](#)() [1/2]

```
IPAddress ( ) [inline]
```

### 9.35.2.2 IPAddress() [2/2]

```
IPAddress (
    unsigned int ipAddressVal ) [inline]
```

## 9.35.3 Member Function Documentation

### 9.35.3.1 operator!=(())

```
bool operator!= (
    const IPAddress & address ) [inline]
```

Inequality operator.

### 9.35.3.2 operator==(())

```
bool operator== (
    const IPAddress & address ) const [inline]
```

Equality operator.

## 9.35.4 Member Data Documentation

### 9.35.4.1 octets

```
unsigned char octets[4]
```

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

- [FlyCapture2Defs.h](#)

## 9.36 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.*

### 9.36.1 Detailed Description

Options for saving JPEG image.

### 9.36.2 Constructor & Destructor Documentation

#### 9.36.2.1 JPEGOption()

```
JPEGOption ( ) [inline]
```

### 9.36.3 Member Data Documentation

#### 9.36.3.1 progressive

```
bool progressive
```

Whether to save as a progressive JPEG file.

#### 9.36.3.2 quality

```
unsigned int quality
```

JPEG image quality in range (0-100).

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

### 9.36.3.3 reserved

```
unsigned int reserved[16]
```

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

## 9.37 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.*

### 9.37.1 Detailed Description

Options for saving JPEG2000 image.

### 9.37.2 Constructor & Destructor Documentation

#### 9.37.2.1 JPG2Option()

```
JPG2Option ( ) [inline]
```

### 9.37.3 Member Data Documentation

### 9.37.3.1 quality

```
unsigned int quality
```

JPEG saving quality in range (1-512).

### 9.37.3.2 reserved

```
unsigned int reserved[16]
```

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

## 9.38 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.*

### 9.38.1 Detailed Description

Information about the camera's look up table.

### 9.38.2 Constructor & Destructor Documentation

#### 9.38.2.1 LUTData()

```
LUTData ( ) [inline]
```

### 9.38.3 Member Data Documentation

#### 9.38.3.1 enabled

```
bool enabled
```

Flag indicating if LUT is enabled.

#### 9.38.3.2 inputBitDepth

```
unsigned int inputBitDepth
```

The input bit depth of the LUT.

#### 9.38.3.3 numBanks

```
unsigned int numBanks
```

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

#### 9.38.3.4 numChannels

```
unsigned int numChannels
```

The number of LUT channels per bank available.

#### 9.38.3.5 numEntries

```
unsigned int numEntries
```

The number of entries in the LUT.

#### 9.38.3.6 outputBitDepth

```
unsigned int outputBitDepth
```

The output bit depth of the LUT.

#### 9.38.3.7 reserved

```
unsigned int reserved[8]
```

Reserved for future use.

#### 9.38.3.8 supported

```
bool supported
```

Flag indicating if LUT is supported.

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

- [FlyCapture2Defs.h](#)

## 9.39 MACAddress Struct Reference

MAC address.

### Public Member Functions

- [MACAddress](#) ()
- [MACAddress](#) (unsigned int macAddressValHigh, unsigned int macAddressValLow)
- bool [operator==](#) (const [MACAddress](#) &address) const  
*Equality operator.*
- bool [operator!=](#) (const [MACAddress](#) &address)  
*Inequality operator.*

## Public Attributes

- unsigned char [octets](#) [6]

### 9.39.1 Detailed Description

MAC address.

### 9.39.2 Constructor & Destructor Documentation

#### 9.39.2.1 [MACAddress\(\)](#) [1/2]

```
MACAddress ( ) [inline]
```

#### 9.39.2.2 [MACAddress\(\)](#) [2/2]

```
MACAddress (  
    unsigned int macAddressValHigh,  
    unsigned int macAddressValLow ) [inline]
```

### 9.39.3 Member Function Documentation

#### 9.39.3.1 [operator!=\(\)](#)

```
bool operator!= (  
    const MACAddress & address ) [inline]
```

Inequality operator.

#### 9.39.3.2 [operator==\(\)](#)

```
bool operator== (  
    const MACAddress & address ) const [inline]
```

Equality operator.

### 9.39.4 Member Data Documentation

#### 9.39.4.1 octets

```
unsigned char octets[6]
```

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

- [FlyCapture2Defs.h](#)

## 9.40 MJPGOption Struct Reference

Options for saving MJPG files.

### Public Member Functions

- [MJPGOption](#) ()

### Public Attributes

- float [frameRate](#)  
*Frame rate of the stream.*
- unsigned int [quality](#)  
*Image quality (1-100)*
- unsigned int [reserved](#) [256]

### 9.40.1 Detailed Description

Options for saving MJPG files.

### 9.40.2 Constructor & Destructor Documentation

#### 9.40.2.1 MJPGOption()

```
MJPGOption ( ) [inline]
```

### 9.40.3 Member Data Documentation

#### 9.40.3.1 frameRate

```
float frameRate
```

Frame rate of the stream.

#### 9.40.3.2 quality

```
unsigned int quality
```

[Image](#) quality (1-100)

#### 9.40.3.3 reserved

```
unsigned int reserved[256]
```

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

- [FlyCapture2VideoDefs.h](#)

## 9.41 NodeMap Class Reference

### Public Member Functions

- [NodeMap](#) (GenApi::CNodeMapRef \*ref)
- virtual [~NodeMap](#) (void)
- GenICam::gcstring [\\_GetDeviceName](#) ()  
*Get device name.*
- void [\\_Poll](#) (int64\_t ElapsedTime)  
*Fires nodes which have a polling time.*
- void [\\_GetNodes](#) (NodeList\_t &Nodes)  
*Retrieves all nodes in the node map.*
- INode \* [\\_GetNode](#) (const GenICam::gcstring &key)  
*Retrieves the node from the central map by name.*
- void [\\_InvalidateNodes](#) () const  
*Invalidates all nodes.*

#### 9.41.1 Constructor & Destructor Documentation



#### 9.41.1.1 NodeMap()

```
NodeMap (
    GenApi::CNodeMapRef * ref )
```

#### 9.41.1.2 ~NodeMap()

```
virtual ~NodeMap (
    void ) [virtual]
```

### 9.41.2 Member Function Documentation

#### 9.41.2.1 \_GetDeviceName()

```
GenICam::gcstring _GetDeviceName ( )
```

Get device name.

#### 9.41.2.2 \_GetNode()

```
INode* _GetNode (
    const GenICam::gcstring & key )
```

Retrieves the node from the central map by name.

#### 9.41.2.3 \_GetNodes()

```
void _GetNodes (
    NodeList_t & Nodes )
```

Retrieves all nodes in the node map.

#### 9.41.2.4 \_InvalidateNodes()

```
void _InvalidateNodes ( ) const
```

Invalidates all nodes.

#### 9.41.2.5 \_Poll()

```
void _Poll (
    int64_t ElapsedTime )
```

Fires nodes which have a polling time.

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

- [NodeMap.h](#)

## 9.42 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.*

#### 9.42.1 Detailed Description

Options for saving PGM images.

#### 9.42.2 Constructor & Destructor Documentation

##### 9.42.2.1 PGMOption()

```
PGMOption ( ) [inline]
```

#### 9.42.3 Member Data Documentation

#### 9.42.3.1 binaryFile

```
bool binaryFile
```

Whether to save the PPM as a binary file.

#### 9.42.3.2 reserved

```
unsigned int reserved[16]
```

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

## 9.43 PGRGuid Class Reference

A GUID to the camera.

### Public Member Functions

- [PGRGuid \(\)](#)  
*Constructor.*
- bool [operator==](#) (const [PGRGuid](#) &guid) const  
*Equality operator.*
- bool [operator!=](#) (const [PGRGuid](#) &guid)  
*Inequality operator.*

### Public Attributes

- unsigned int [value](#) [4]

#### 9.43.1 Detailed Description

A GUID to the camera.

It is used to uniquely identify a camera.

#### 9.43.2 Constructor & Destructor Documentation

#### 9.43.2.1 PGRGuid()

```
PGRGuid ( ) [inline]
```

Constructor.

### 9.43.3 Member Function Documentation

#### 9.43.3.1 operator"!="()

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

Inequality operator.

#### 9.43.3.2 operator==( )

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

Equality operator.

### 9.43.4 Member Data Documentation

#### 9.43.4.1 value

```
unsigned int value[4]
```

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

- [FlyCapture2Defs.h](#)

## 9.44 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.*

### 9.44.1 Detailed Description

Options for saving PNG images.

### 9.44.2 Constructor & Destructor Documentation

#### 9.44.2.1 PNGOption()

```
PNGOption ( ) [inline]
```

### 9.44.3 Member Data Documentation

#### 9.44.3.1 compressionLevel

```
unsigned int compressionLevel
```

Compression level (0-9).

0 is no compression, 9 is best compression.

#### 9.44.3.2 interlaced

```
bool interlaced
```

Whether to save the PNG as interlaced.

#### 9.44.3.3 reserved

```
unsigned int reserved[16]
```

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

## 9.45 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.*

### 9.45.1 Detailed Description

Options for saving PPM images.

### 9.45.2 Constructor & Destructor Documentation

#### 9.45.2.1 PPMOption()

```
PPMOption ( ) [inline]
```

### 9.45.3 Member Data Documentation

### 9.45.3.1 binaryFile

```
bool binaryFile
```

Whether to save the PPM as a binary file.

### 9.45.3.2 reserved

```
unsigned int reserved[16]
```

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

## 9.46 Property Struct Reference

A specific camera property.

### Public Member Functions

- [Property](#) ()
- [Property](#) ([PropertyType](#) propType)

### Public Attributes

- [PropertyType](#) type  
*Property info type.*
- bool [present](#)  
*Flag indicating if the property is present.*
- bool [absControl](#)  
*Flag controlling absolute mode (real world units) or non-absolute mode (camera internal units).*
- 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.*

### 9.46.1 Detailed Description

A specific camera property.

For example, to set the gain to 12dB, set the following values:

- *type* - GAIN
- *absControl* - true
- *onePush* - false
- *onOff* - true
- *autoManualMode* - false
- *absValue* - 12.0

### 9.46.2 Constructor & Destructor Documentation

#### 9.46.2.1 Property() [1/2]

```
Property ( ) [inline]
```

#### 9.46.2.2 Property() [2/2]

```
Property (
    PropertyType propType ) [inline]
```

### 9.46.3 Member Data Documentation

#### 9.46.3.1 absControl

```
bool absControl
```

Flag controlling absolute mode (real world units) or non-absolute mode (camera internal units).



#### 9.46.3.2 absValue

```
float absValue
```

Floating point value.

Used to configure properties in absolute mode.

#### 9.46.3.3 autoManualMode

```
bool autoManualMode
```

Flag controlling auto.

#### 9.46.3.4 onePush

```
bool onePush
```

Flag controlling one push.

#### 9.46.3.5 onOff

```
bool onOff
```

Flag controlling on/off.

#### 9.46.3.6 present

```
bool present
```

Flag indicating if the property is present.

#### 9.46.3.7 reserved

```
unsigned int reserved[8]
```

Reserved for future use.

#### 9.46.3.8 type

[PropertyType](#) type

[Property](#) info type.

#### 9.46.3.9 valueA

unsigned int valueA

Value A (integer).

Used to configure properties in non-absolute mode.

#### 9.46.3.10 valueB

unsigned int valueB

Value B (integer).

For white balance, value B applies to the blue value and value A applies to the red value.

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

- [FlyCapture2Defs.h](#)

## 9.47 PropertyInfo Struct Reference

Information about a specific camera property.

### Public Member Functions

- [PropertyInfo](#) ()
- [PropertyInfo](#) ([PropertyType](#) propType)

## Public Attributes

- [PropertyType](#) type  
*Property info type.*
- bool [present](#)  
*Flag indicating if the property is present.*
- bool [autoSupported](#)  
*Flag indicating if auto is supported.*
- bool [manualSupported](#)  
*Flag indicating if manual is supported.*
- bool [onOffSupported](#)  
*Flag indicating if on/off is supported.*
- bool [onePushSupported](#)  
*Flag indicating if one push is supported.*
- bool [absValSupported](#)  
*Flag indicating if absolute mode is supported.*
- bool [readOutSupported](#)  
*Flag indicating if property value can be read out.*
- unsigned int [min](#)  
*Minimum value (as an integer).*
- unsigned int [max](#)  
*Maximum value (as an integer).*
- float [absMin](#)  
*Minimum value (as a floating point value).*
- float [absMax](#)  
*Maximum value (as a floating point value).*
- char [pUnits](#) [[sk\\_maxStringLength](#)]  
*Textual description of units.*
- char [pUnitAbbr](#) [[sk\\_maxStringLength](#)]  
*Abbreviated textual description of units.*
- unsigned int [reserved](#) [8]  
*Reserved for future use.*

### 9.47.1 Detailed Description

Information about a specific camera property.

This structure is also also used as the TriggerDelayInfo structure.

### 9.47.2 Constructor & Destructor Documentation

#### 9.47.2.1 PropertyInfo() [1/2]

```
PropertyInfo ( ) [inline]
```

#### 9.47.2.2 PropertyInfo() [2/2]

```
PropertyInfo (
    PropertyType propType ) [inline]
```

### 9.47.3 Member Data Documentation

#### 9.47.3.1 absMax

```
float absMax
```

Maximum value (as a floating point value).

#### 9.47.3.2 absMin

```
float absMin
```

Minimum value (as a floating point value).

#### 9.47.3.3 absValSupported

```
bool absValSupported
```

Flag indicating if absolute mode is supported.

#### 9.47.3.4 autoSupported

```
bool autoSupported
```

Flag indicating if auto is supported.

#### 9.47.3.5 manualSupported

```
bool manualSupported
```

Flag indicating if manual is supported.

#### 9.47.3.6 max

`unsigned int max`

Maximum value (as an integer).

#### 9.47.3.7 min

`unsigned int min`

Minimum value (as an integer).

#### 9.47.3.8 onePushSupported

`bool onePushSupported`

Flag indicating if one push is supported.

#### 9.47.3.9 onOffSupported

`bool onOffSupported`

Flag indicating if on/off is supported.

#### 9.47.3.10 present

`bool present`

Flag indicating if the property is present.

#### 9.47.3.11 pUnitAbbr

`char pUnitAbbr[sk_maxStringLength]`

Abbreviated textual description of units.

#### 9.47.3.12 pUnits

```
char pUnits[sk_maxStringLength]
```

Textual description of units.

#### 9.47.3.13 readOutSupported

```
bool readOutSupported
```

Flag indicating if property value can be read out.

#### 9.47.3.14 reserved

```
unsigned int reserved[8]
```

Reserved for future use.

#### 9.47.3.15 type

```
PropertyType type
```

[Property](#) info type.

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

- [FlyCapture2Defs.h](#)

## 9.48 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.*

### 9.48.1 Detailed Description

A camera strobe.

### 9.48.2 Constructor & Destructor Documentation

#### 9.48.2.1 StrobeControl()

```
StrobeControl ( ) [inline]
```

### 9.48.3 Member Data Documentation

#### 9.48.3.1 delay

```
float delay
```

Signal delay (in ms).

#### 9.48.3.2 duration

```
float duration
```

Signal duration (in ms).

#### 9.48.3.3 onOff

`bool onOff`

Flag controlling on/off.

#### 9.48.3.4 polarity

`unsigned int polarity`

Signal polarity.

#### 9.48.3.5 reserved

`unsigned int reserved[8]`

Reserved for future use.

#### 9.48.3.6 source

`unsigned int source`

Source value.

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

- [FlyCapture2Defs.h](#)

## 9.49 StrobelInfo Struct Reference

A camera strobe property.

### Public Member Functions

- [StrobelInfo](#) ()



## 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.*

### 9.49.1 Detailed Description

A camera strobe property.

### 9.49.2 Constructor & Destructor Documentation

#### 9.49.2.1 StrobelInfo()

```
StrobelInfo ( ) [inline]
```

### 9.49.3 Member Data Documentation

#### 9.49.3.1 maxValue

```
float maxValue
```

Maximum value.

**9.49.3.2   minValue**

```
float minValue
```

Minimum value.

**9.49.3.3   onOffSupported**

```
bool onOffSupported
```

Flag indicating if on/off is supported.

**9.49.3.4   polaritySupported**

```
bool polaritySupported
```

Flag indicating if polarity is supported.

**9.49.3.5   present**

```
bool present
```

Presence of strobe.

**9.49.3.6   readOutSupported**

```
bool readOutSupported
```

Flag indicating if strobe value can be read out.

**9.49.3.7   reserved**

```
unsigned int reserved[8]
```

Reserved for future use.

## 9.49.3.8 source

unsigned int source

Source value.

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

- [FlyCapture2Defs.h](#)

## 9.50 SyncManager Class Reference

### Public Member Functions

- [MULTISYNCLIBRARY\\_API SyncManager \(\)](#)
- [MULTISYNCLIBRARY\\_API ~SyncManager \(\)](#)
- [MULTISYNCLIBRARY\\_API PGRSyncError Start \(\)](#)
- [MULTISYNCLIBRARY\\_API PGRSyncError Stop \(\)](#)
- [MULTISYNCLIBRARY\\_API PGRSyncError RescanMasterTimingBus \(\)](#)
- [MULTISYNCLIBRARY\\_API PGRSyncMessage GetSyncStatus \(\)](#)
- [MULTISYNCLIBRARY\\_API double GetTimeSinceSynced \(\)](#)
- [MULTISYNCLIBRARY\\_API bool IsTimingBusConnected \(\)](#)
- [MULTISYNCLIBRARY\\_API bool EnableCrossPCSynchronization \(\)](#)
- [MULTISYNCLIBRARY\\_API bool DisableCrossPCSynchronization \(\)](#)
- [MULTISYNCLIBRARY\\_API bool QueryCrossPCSynchronizationSetting \(\)](#)

### 9.50.1 Constructor & Destructor Documentation

#### 9.50.1.1 SyncManager()

[MULTISYNCLIBRARY\\_API SyncManager \( \)](#)

#### 9.50.1.2 ~SyncManager()

[MULTISYNCLIBRARY\\_API ~SyncManager \( \)](#)

### 9.50.2 Member Function Documentation

#### 9.50.2.1 DisableCrossPCSynchronization()

`MULTISYNCLIBRARY_API bool DisableCrossPCSynchronization ( )`

#### 9.50.2.2 EnableCrossPCSynchronization()

`MULTISYNCLIBRARY_API bool EnableCrossPCSynchronization ( )`

#### 9.50.2.3 GetSyncStatus()

`MULTISYNCLIBRARY_API PGRSyncMessage GetSyncStatus ( )`

#### 9.50.2.4 GetTimeSinceSynced()

`MULTISYNCLIBRARY_API double GetTimeSinceSynced ( )`

#### 9.50.2.5 IsTimingBusConnected()

`MULTISYNCLIBRARY_API bool IsTimingBusConnected ( )`

#### 9.50.2.6 QueryCrossPCSynchronizationSetting()

`MULTISYNCLIBRARY_API bool QueryCrossPCSynchronizationSetting ( )`

#### 9.50.2.7 RescanMasterTimingBus()

`MULTISYNCLIBRARY_API PGRSyncError RescanMasterTimingBus ( )`

#### 9.50.2.8 Start()

`MULTISYNCLIBRARY_API PGRSyncError Start ( )`

## 9.50.2.9 Stop()

```
MULTISYNCLIBRARY_API PGRSyncError Stop ( )
```

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

- [MultiSyncLibraryDefs.h](#)

## 9.51 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.*

### 9.51.1 Detailed Description

Description of the system.

### 9.51.2 Member Data Documentation

#### 9.51.2.1 `byteOrder`

`ByteOrder` `byteOrder`

Byte order of the system.

#### 9.51.2.2 `cpuDescription`

`char` `cpuDescription`[`sk_maxStringLength`]

Detailed description of the CPU.

#### 9.51.2.3 `driverList`

`char` `driverList`[`sk_maxStringLength`]

List of drivers used.

#### 9.51.2.4 `gpuDescription`

`char` `gpuDescription`[`sk_maxStringLength`]

Detailed description of the GPU.

#### 9.51.2.5 `libraryList`

`char` `libraryList`[`sk_maxStringLength`]

List of libraries used.

#### 9.51.2.6 `numCpuCores`

`size_t` `numCpuCores`

Number of cores on all CPUs on the system.

#### 9.51.2.7 osDescription

```
char osDescription[sk_maxStringLength]
```

Detailed description of the operating system.

#### 9.51.2.8 osType

```
OSType osType
```

Operating system type as described by OSType.

#### 9.51.2.9 reserved

```
unsigned int reserved[16]
```

Reserved for future use.

#### 9.51.2.10 screenHeight

```
size_t screenHeight
```

Screen resolution height in pixels.

#### 9.51.2.11 screenWidth

```
size_t screenWidth
```

Screen resolution width in pixels.

#### 9.51.2.12 sysMemSize

```
size_t sysMemSize
```

Amount of memory available on the system.

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

- [Utilities.h](#)

## 9.52 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.*

### 9.52.1 Detailed Description

Options for saving TIFF images.

### 9.52.2 Member Enumeration Documentation

#### 9.52.2.1 CompressionMethod

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.



### 9.52.3 Constructor & Destructor Documentation

#### 9.52.3.1 TIFFOption()

```
TIFFOption ( ) [inline]
```

### 9.52.4 Member Data Documentation

#### 9.52.4.1 compression

```
CompressionMethod compression
```

Compression method to use for encoding TIFF images.

#### 9.52.4.2 reserved

```
unsigned int reserved[16]
```

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

## 9.53 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.*

### 9.53.1 Detailed Description

Timestamp information.

### 9.53.2 Constructor & Destructor Documentation

#### 9.53.2.1 TimeStamp()

```
TimeStamp ( ) [inline]
```

### 9.53.3 Member Data Documentation

#### 9.53.3.1 cycleCount

```
unsigned int cycleCount
```

1394 cycle time count.

#### 9.53.3.2 cycleOffset

```
unsigned int cycleOffset
```

1394 cycle time offset.

#### 9.53.3.3 cycleSeconds

`unsigned int cycleSeconds`

1394 cycle time seconds.

#### 9.53.3.4 microSeconds

`unsigned int microSeconds`

Microseconds.

#### 9.53.3.5 reserved

`unsigned int reserved[8]`

Reserved for future use.

#### 9.53.3.6 seconds

`long long seconds`

Seconds.

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

- [FlyCapture2Defs.h](#)

## 9.54 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 AddPortType ([PortType](#) childPort)  
*Add the specified [PortType](#) as a port of the node.*
- virtual bool AssignGuidToNode ([PGRGuid](#) guid, int deviceId)  
*Assign a [PGRGuid](#) and device ID to the node.*
- virtual bool AssignGuidToNode ([PGRGuid](#) guid, int deviceId, [NodeType](#) nodeType)  
*Assign a [PGRGuid](#), device ID and nodeType to the node.*

### 9.54.1 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.

### 9.54.2 Member Enumeration Documentation

#### 9.54.2.1 NodeType

enum [NodeType](#)

Type of node.

## Enumerator

COMPUTER	
BUS	
CAMERA	
NODE	

## 9.54.2.2 PortType

enum `PortType`

Possible states of a port on a node.

## Enumerator

NOT_CONNECTED	
CONNECTED_TO_PARENT	
CONNECTED_TO_CHILD	

## 9.54.3 Constructor &amp; Destructor Documentation

## 9.54.3.1 TopologyNode() [1/3]

`TopologyNode ( )`

Default constructor.

## 9.54.3.2 TopologyNode() [2/3]

```
TopologyNode (
    PGRGuid guid,
    int deviceId,
    NodeType nodeType,
    InterfaceType interfaceType )
```

Constructor.

## Parameters

<i>guid</i>	The <code>PGRGuid</code> of the node (if applicable).
<i>deviceId</i>	Device ID of the node.
<i>nodeType</i>	Type of the node.
<i>interfaceType</i>	Interface type of the node.

#### 9.54.3.3 ~TopologyNode()

```
virtual ~TopologyNode ( ) [virtual]
```

Default destructor.

#### 9.54.3.4 TopologyNode() [3/3]

```
TopologyNode (
    const TopologyNode & other )
```

Copy constructor.

### 9.54.4 Member Function Documentation

#### 9.54.4.1 AddChild()

```
virtual void AddChild (
    TopologyNode childNode ) [virtual]
```

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

##### Parameters

<i>childNode</i>	The <a href="#">TopologyNode</a> to add.
------------------	--

#### 9.54.4.2 AddPortType()

```
virtual void AddPortType (
    PortType childPort ) [virtual]
```

Add the specified PortType as a port of the node.

##### Parameters

<i>childPort</i>	The port to add.
------------------	------------------

#### 9.54.4.3 AssignGuidToNode() [1/2]

```
virtual bool AssignGuidToNode (
    PGRGuid guid,
    int deviceId ) [virtual]
```

Assign a [PGRGuid](#) and device ID to the node.

##### Parameters

<i>guid</i>	<a href="#">PGRGuid</a> to be assigned.
<i>deviceId</i>	Device ID to be assigned.

##### Returns

Whether the data was successfully set to the node.

#### 9.54.4.4 AssignGuidToNode() [2/2]

```
virtual bool AssignGuidToNode (
    PGRGuid guid,
    int deviceId,
    NodeType nodeType ) [virtual]
```

Assign a [PGRGuid](#), device ID and nodeType to the node.

##### Parameters

<i>guid</i>	<a href="#">PGRGuid</a> to be assigned.
<i>deviceId</i>	Device ID to be assigned.
<i>nodeType</i>	NodeType to be assigned

##### Returns

Whether the data was successfully set to the node.

#### 9.54.4.5 GetChild()

```
virtual TopologyNode GetChild (
    unsigned int position ) [virtual]
```

Get child node located at the specified position.

**Parameters**

<i>position</i>	Position of the node.
-----------------	-----------------------

**Returns**

[TopologyNode](#) at the specified position.

**9.54.4.6 GetDeviceId()**

```
virtual int GetDeviceId ( ) [virtual]
```

Get the device ID associated with the node.

**Returns**

Device ID of the node.

**9.54.4.7 GetGuid()**

```
virtual PGRGuid GetGuid ( ) [virtual]
```

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

**Returns**

[PGRGuid](#) of the node.

**9.54.4.8 GetInterfaceType()**

```
virtual InterfaceType GetInterfaceType ( ) [virtual]
```

Get the interface type associated with the node.

**Returns**

Interface type of the node.



#### 9.54.4.9 GetNodeType()

```
virtual NodeType GetNodeType ( ) [virtual]
```

Get the node type associated with the node.

##### Returns

Node type of the node.

#### 9.54.4.10 GetNumChildren()

```
virtual unsigned int GetNumChildren ( ) [virtual]
```

Get the number of child nodes.

##### Returns

Number of child nodes.

#### 9.54.4.11 GetNumPorts()

```
virtual unsigned int GetNumPorts ( ) [virtual]
```

Get the number of ports.

##### Returns

Number of ports.

#### 9.54.4.12 GetPortType()

```
virtual PortType GetPortType (
    unsigned int position ) [virtual]
```

Get type of port located at the specified position.

##### Parameters

<i>position</i>	Position of the port.
-----------------	-----------------------

### Returns

PortType at the specified position.

#### 9.54.4.13 operator=()

```
virtual TopologyNode& operator= (  
    const TopologyNode & other ) [virtual]
```

Assignment operator.

### Parameters

<i>other</i>	The <a href="#">TopologyNode</a> to copy from.
--------------	--

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

- [TopologyNode.h](#)

## 9.55 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.*

#### 9.55.1 Detailed Description

A camera trigger.

## 9.55.2 Constructor & Destructor Documentation

### 9.55.2.1 TriggerMode()

```
TriggerMode ( ) [inline]
```

## 9.55.3 Member Data Documentation

### 9.55.3.1 mode

```
unsigned int mode
```

Mode value.

### 9.55.3.2 onOff

```
bool onOff
```

Flag controlling on/off.

### 9.55.3.3 parameter

```
unsigned int parameter
```

Parameter value.

### 9.55.3.4 polarity

```
unsigned int polarity
```

Polarity value.

#### 9.55.3.5 reserved

```
unsigned int reserved[8]
```

Reserved for future use.

#### 9.55.3.6 source

```
unsigned int source
```

Source value.

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

- [FlyCapture2Defs.h](#)

## 9.56 TriggerModelInfo Struct Reference

Information about a camera trigger property.

### Public Member Functions

- [TriggerModelInfo](#) ()

### 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.*

### 9.56.1 Detailed Description

Information about a camera trigger property.

### 9.56.2 Constructor & Destructor Documentation

#### 9.56.2.1 TriggerModelInfo()

```
TriggerModelInfo ( ) [inline]
```

### 9.56.3 Member Data Documentation

#### 9.56.3.1 modeMask

```
unsigned int modeMask
```

Mode mask.

#### 9.56.3.2 onOffSupported

```
bool onOffSupported
```

Flag indicating if on/off is supported.

#### 9.56.3.3 polaritySupported

```
bool polaritySupported
```

Flag indicating if polarity is supported.

#### 9.56.3.4 present

```
bool present
```

Presence of trigger mode.

#### 9.56.3.5 readOutSupported

```
bool readOutSupported
```

Flag indicating if trigger value can be read out.

#### 9.56.3.6 reserved

```
unsigned int reserved[8]
```

Reserved for future use.

#### 9.56.3.7 softwareTriggerSupported

```
bool softwareTriggerSupported
```

Flag indicating if software trigger is supported.

#### 9.56.3.8 sourceMask

```
unsigned int sourceMask
```

Source mask.

#### 9.56.3.9 valueReadable

```
bool valueReadable
```

Flag indicating if the value is readable.

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

- [FlyCapture2Defs.h](#)

## 9.57 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 CheckDriver](#) (const [PGRGuid](#) \*guid)  
*Check for driver compatibility for the given camera guid.*
- static [Error GetDriverDeviceName](#) (const [PGRGuid](#) \*guid, std::string &deviceName)  
*Get the driver's name for a device.*
- 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 \*p↵  
UserData)  
*Execute a command in the terminal.*

### 9.57.1 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.

### 9.57.2 Member Function Documentation

#### 9.57.2.1 CheckDriver()

```
static Error CheckDriver (
    const PGRGuid * guid ) [static]
```

Check for driver compatibility for the given camera guid.

#### Parameters

<i>guid</i>	Pointer to the guid of the device to check.
-------------	---

#### Returns

[PGR\\_NO\\_ERROR](#) if the library is compatible with the currently loaded driver, otherwise an error indicating the type of failure.

### 9.57.2.2 GetDriverDeviceName()

```
static Error GetDriverDeviceName (
    const PGRGuid * guid,
    std::string & deviceName ) [static]
```

Get the driver's name for a device.

#### Parameters

<i>guid</i>	Pointer to the guid of the device to check.
<i>deviceName</i>	The device name will be returned in this string

#### Returns

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

### 9.57.2.3 GetLibraryVersion()

```
static Error GetLibraryVersion (
    FC2Version * pVersion ) [static]
```

Get library version.

#### Parameters

<i>pVersion</i>	Structure to receive the library version.
-----------------	---

#### Returns

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

### 9.57.2.4 GetSystemInfo()

```
static Error GetSystemInfo (
    SystemInfo * pSystemInfo ) [static]
```

Get system information.

#### Parameters

<i>pSystemInfo</i>	Structure to receive system information.
--------------------	--



**Returns**

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

**9.57.2.5 LaunchBrowser()**

```
static Error LaunchBrowser (  
    const char * pAddress ) [static]
```

Launch a URL in the system default browser.

**Parameters**

<i>pAddress</i>	URL to open in browser.
-----------------	-------------------------

**Returns**

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

**9.57.2.6 LaunchCommand()**

```
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**

<i>pCommand</i>	Command to execute.
-----------------	---------------------

**See also**

[LaunchCommandAsync\(\)](#)

**Returns**

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

### 9.57.2.7 LaunchCommandAsync()

```
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

<i>pCommand</i>	Command to execute.
<i>pCallback</i>	Callback to fire when command is complete.
<i>pUserData</i>	Data pointer to pass to callback.

#### See also

[LaunchCommand\(\)](#)

#### Returns

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

### 9.57.2.8 LaunchHelp()

```
static Error LaunchHelp (
    const char * pFileName ) [static]
```

Open a CHM file in the system default CHM viewer.

#### Parameters

<i>pFileName</i>	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:

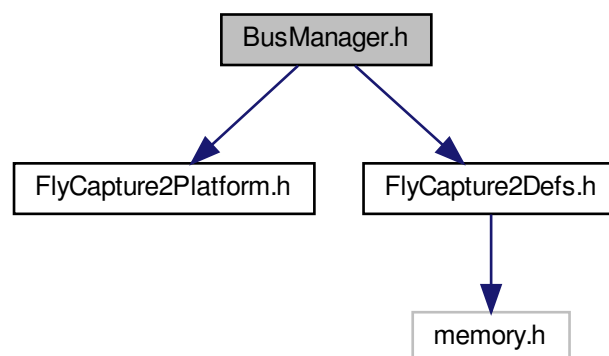
- [Utilities.h](#)

## Chapter 10

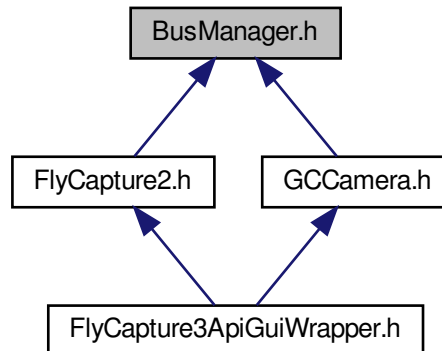
# File Documentation

### 10.1 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

- [FlyCapture2](#)

## Typedefs

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

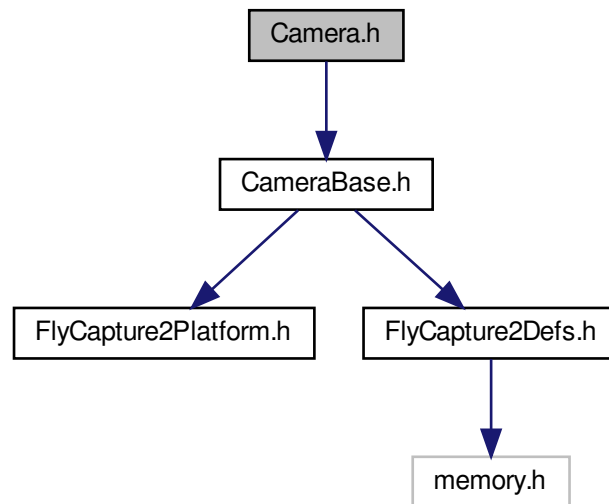
*Bus event callback function prototype.*

- typedef void \* [CallbackHandle](#)

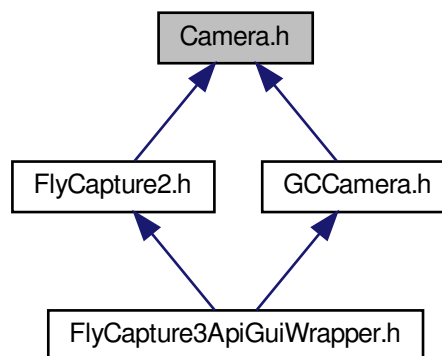
*Handle that is returned when registering a callback.*

## 10.2 Camera.h File Reference

Include dependency graph for Camera.h:



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



### Classes

- class [Camera](#)

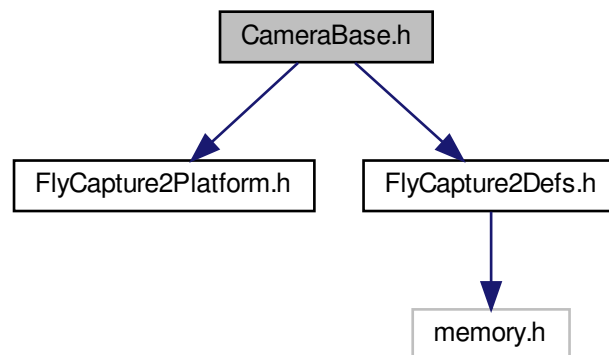
The [Camera](#) object represents a physical camera that uses the IIDC register set.

## Namespaces

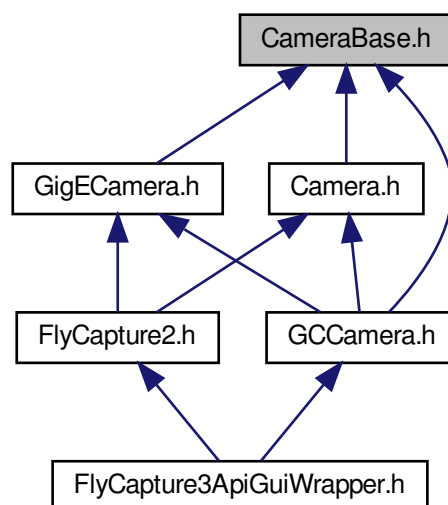
- [FlyCapture2](#)

## 10.3 CameraBase.h File Reference

Include dependency graph for CameraBase.h:



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



## Classes

- class [CameraBase](#)

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

## Namespaces

- [FlyCapture2](#)

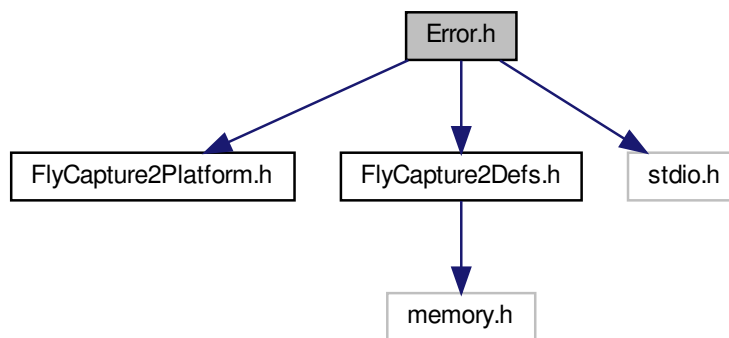
## Typedefs

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

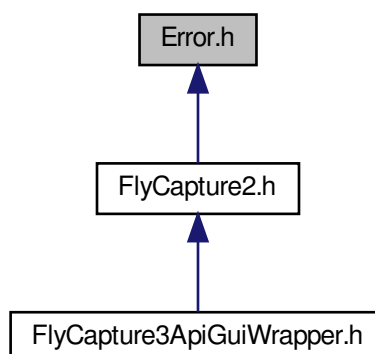
[Image](#) event callback function prototype.

## 10.4 Error.h File Reference

Include dependency graph for Error.h:



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



## Classes

- class [Error](#)

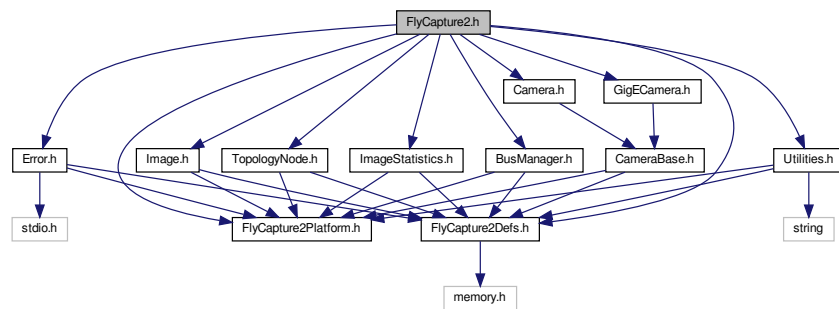
The [Error](#) object represents an error that is returned from the library.

## Namespaces

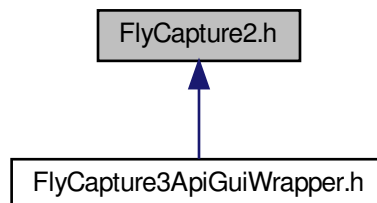
- [FlyCapture2](#)

## 10.5 FlyCapture2.h File Reference

Include dependency graph for FlyCapture2.h:



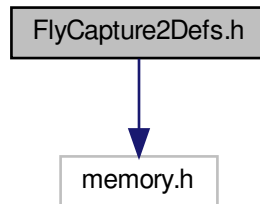
This graph shows which files directly or indirectly include this file:



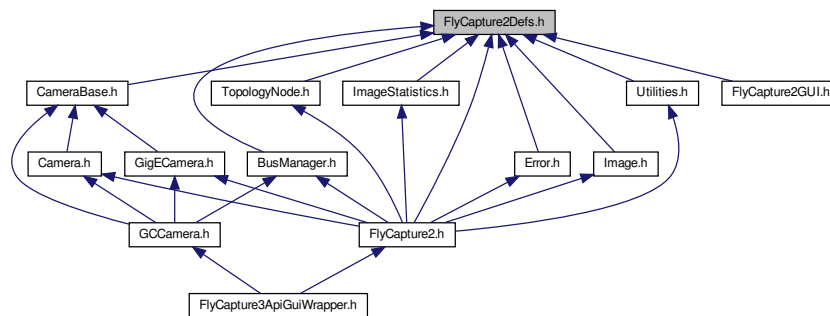


## 10.6 FlyCapture2Defs.h File Reference

Include dependency graph for FlyCapture2Defs.h:



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



### Classes

- struct [FC2Version](#)  
*The current version of the library.*
- class [PGRGuid](#)  
*A GUID to the camera.*
- struct [IPAddress](#)  
*IPv4 address.*
- struct [MACAddress](#)  
*MAC address.*
- struct [GigEProperty](#)  
*A GigE property.*
- struct [GigEStreamChannel](#)  
*Information about a single GigE stream channel.*
- struct [GigEConfig](#)  
*Configuration for a GigE camera.*
- struct [GigEImageSettingsInfo](#)

- Format 7 information for a single mode.*
- struct [GigEImageSettings](#)
  - Image settings for a GigE camera.*
- struct [Format7ImageSettings](#)
  - Format 7 image settings.*
- struct [Format7Info](#)
  - Format 7 information for a single mode.*
- struct [Format7PacketInfo](#)
  - Format 7 packet information.*
- struct [FC2Config](#)
  - Configuration for a camera.*
- struct [PropertyInfo](#)
  - Information about a specific camera property.*
- struct [Property](#)
  - A specific camera property.*
- struct [TriggerModelInfo](#)
  - Information about a camera trigger property.*
- struct [TriggerMode](#)
  - A camera trigger.*
- struct [StrobeInfo](#)
  - A camera strobe property.*
- struct [StrobeControl](#)
  - A camera strobe.*
- struct [TimeStamp](#)
  - Timestamp information.*
- struct [ConfigROM](#)
  - Camera configuration ROM.*
- struct [CameraInfo](#)
  - Camera information.*
- struct [EmbeddedImageInfoProperty](#)
  - Properties of a single embedded image info property.*
- struct [EmbeddedImageInfo](#)
  - Properties of the possible embedded image information.*
- struct [ImageMetadata](#)
  - Metadata related to an image.*
- struct [LUTData](#)
  - Information about the camera's look up table.*
- struct [CameraStats](#)
  - Camera diagnostic information.*
- struct [PNGOption](#)
  - Options for saving PNG images.*
- struct [PPMOption](#)
  - Options for saving PPM images.*
- struct [PGMOption](#)
  - Options for saving PGM images.*
- struct [TIFFOption](#)
  - Options for saving TIFF images.*
- struct [JPEGOption](#)
  - Options for saving JPEG image.*
- struct [JPG2Option](#)
  - Options for saving JPEG2000 image.*

- struct [BMPOption](#)  
*Options for saving Bitmap image.*
- struct [EventOptions](#)  
*Options for enabling device event registration.*
- struct [EventCallbackData](#)

## Namespaces

- [FlyCapture2](#)

## Macros

- `#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](#).*
- typedef void(\* [CameraEventCallback](#))(void \*data)

## Enumerations

- enum [ErrorType](#) {  
    [PGRERROR\\_UNDEFINED](#) = -1,  
    [PGRERROR\\_OK](#),  
    [PGRERROR\\_FAILED](#),  
    [PGRERROR\\_NOT\\_IMPLEMENTED](#),  
    [PGRERROR\\_FAILED\\_BUS\\_MASTER\\_CONNECTION](#),  
    [PGRERROR\\_NOT\\_CONNECTED](#),  
    [PGRERROR\\_INIT\\_FAILED](#),  
    [PGRERROR\\_NOT\\_INITIALIZED](#),  
    [PGRERROR\\_INVALID\\_PARAMETER](#),  
    [PGRERROR\\_INVALID\\_SETTINGS](#),  
    [PGRERROR\\_INVALID\\_BUS\\_MANAGER](#),  
    [PGRERROR\\_MEMORY\\_ALLOCATION\\_FAILED](#),  
    [PGRERROR\\_LOW\\_LEVEL\\_FAILURE](#),  
    [PGRERROR\\_NOT\\_FOUND](#),  
    [PGRERROR\\_FAILED\\_GUID](#),  
    [PGRERROR\\_INVALID\\_PACKET\\_SIZE](#),  
    [PGRERROR\\_INVALID\\_MODE](#),  
    [PGRERROR\\_NOT\\_IN\\_FORMAT7](#),  
    [PGRERROR\\_NOT\\_SUPPORTED](#),  
    [PGRERROR\\_TIMEOUT](#),  
    [PGRERROR\\_BUS\\_MASTER\\_FAILED](#),  
    [PGRERROR\\_INVALID\\_GENERATION](#),  
    [PGRERROR\\_LUT\\_FAILED](#),  
    [PGRERROR\\_IIDC\\_FAILED](#),  
    [PGRERROR\\_STROBE\\_FAILED](#),  
    [PGRERROR\\_TRIGGER\\_FAILED](#),  
}

```

PGRERROR_PROPERTY_FAILED,
PGRERROR_PROPERTY_NOT_PRESENT,
PGRERROR_REGISTER_FAILED,
PGRERROR_READ_REGISTER_FAILED,
PGRERROR_WRITE_REGISTER_FAILED,
PGRERROR_ISOCH_FAILED,
PGRERROR_ISOCH_ALREADY_STARTED,
PGRERROR_ISOCH_NOT_STARTED,
PGRERROR_ISOCH_START_FAILED,
PGRERROR_ISOCH_RETRIEVE_BUFFER_FAILED,
PGRERROR_ISOCH_STOP_FAILED,
PGRERROR_ISOCH_SYNC_FAILED,
PGRERROR_ISOCH_BANDWIDTH_EXCEEDED,
PGRERROR_IMAGE_CONVERSION_FAILED,
PGRERROR_IMAGE_LIBRARY_FAILURE,
PGRERROR_BUFFER_TOO_SMALL,
PGRERROR_IMAGE_CONSISTENCY_ERROR,
PGRERROR_INCOMPATIBLE_DRIVER,
PGRERROR_FORCE_32BITS = FULL_32BIT_VALUE }

```

*The error types returned by functions.*

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

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

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

*The grab strategy employed during image transfer.*

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

*Timeout options for grabbing images.*

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

*Bandwidth allocation options for 1394 devices.*

- enum `InterfaceType` {  
`INTERFACE_IEEE1394`,  
`INTERFACE_USB2`,  
`INTERFACE_USB3`,  
`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_RBGU = 0x40000002,
    PIXEL_FORMAT_BGR16 = 0x02000001,
    PIXEL_FORMAT_BGRU16 = 0x02000002,
    PIXEL_FORMAT_422YUV8_JPEG = 0x40000001,
```

```
NUM_PIXEL_FORMATS = 20,
UNSPECIFIED_PIXEL_FORMAT = 0 }
```

*Pixel formats available for Format7 modes.*

- enum `BusSpeed` {  
`BUSSPEED_S100`,  
`BUSSPEED_S200`,  
`BUSSPEED_S400`,  
`BUSSPEED_S480`,  
`BUSSPEED_S800`,  
`BUSSPEED_S1600`,  
`BUSSPEED_S3200`,  
`BUSSPEED_S5000`,  
`BUSSPEED_10BASE_T`,  
`BUSSPEED_100BASE_T`,  
`BUSSPEED_1000BASE_T`,  
`BUSSPEED_10000BASE_T`,  
`BUSSPEED_S_FASTEST`,  
`BUSSPEED_ANY`,  
`BUSSPEED_SPEED_UNKNOWN` = -1,  
`BUSSPEED_FORCE_32BITS` = `FULL_32BIT_VALUE` }

*Bus speeds.*

- enum `PCleBusSpeed` {  
`PCIE_BUSSPEED_2_5`,  
`PCIE_BUSSPEED_5_0`,  
`PCIE_BUSSPEED_UNKNOWN` = -1,  
`PCIE_BUSSPEED_FORCE_32BITS` = `FULL_32BIT_VALUE` }
- enum `DriverType` {  
`DRIVER_1394_CAM`,  
`DRIVER_1394_PRO`,  
`DRIVER_1394_JUUJ`,  
`DRIVER_1394_VIDEO1394`,  
`DRIVER_1394_RAW1394`,  
`DRIVER_USB_NONE`,  
`DRIVER_USB_CAM`,  
`DRIVER_USB3_PRO`,  
`DRIVER_GIGE_NONE`,  
`DRIVER_GIGE_FILTER`,  
`DRIVER_GIGE_PRO`,  
`DRIVER_GIGE_LWF`,  
`DRIVER_UNKNOWN` = -1,  
`DRIVER_FORCE_32BITS` = `FULL_32BIT_VALUE` }

*Types of low level drivers that flycapture uses.*

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

*Color processing algorithms.*

- enum `BayerTileFormat` {  
`NONE`,  
`RGBB`,

```

GRBG,
GBRG,
BGGR,
BT_FORCE_32BITS = FULL_32BIT_VALUE }

```

*Bayer tile formats.*

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

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

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

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

## Variables

- static const unsigned int `sk_maxStringLength` = 512

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

- static const unsigned int `sk_maxNumPorts` = 32

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

## 10.6.1 Macro Definition Documentation

### 10.6.1.1 FULL\_32BIT\_VALUE

```
#define FULL_32BIT_VALUE 0x7FFFFFFF
```

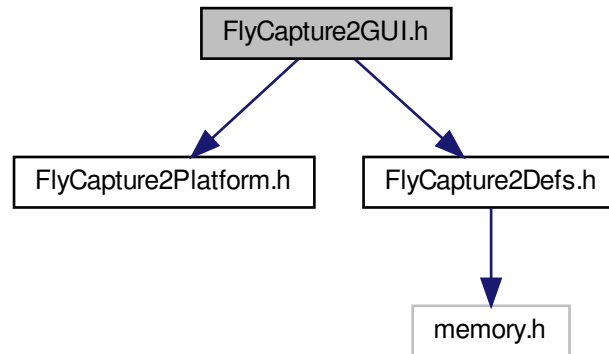
### 10.6.1.2 NULL

```
#define NULL 0
```



## 10.7 FlyCapture2GUI.h File Reference

Include dependency graph for FlyCapture2GUI.h:



### Classes

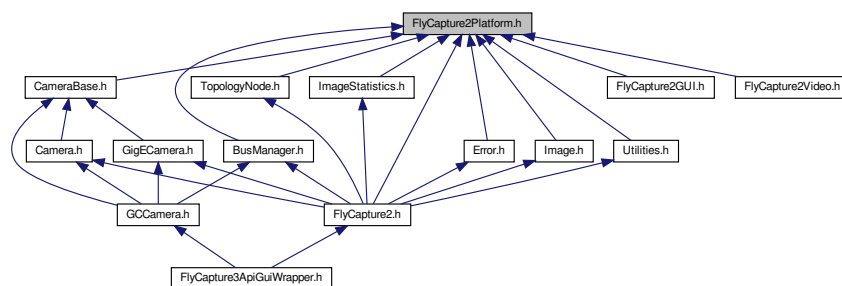
- class [CameraControlDlg](#)  
The *CameraControlDlg* object represents a dialog that provides a graphical interface to a specified camera.
- class [CameraSelectionDlg](#)  
The *CameraSelectionDlg* object represents a dialog that provides a graphical interface that lists the number of cameras available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

### Namespaces

- [FlyCapture2](#)

## 10.8 FlyCapture2Platform.h File Reference

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



## Macros

- `#define FLYCAPTURE2_API __attribute__((visibility("default")))`
- `#define FLYCAPTURE2_LOCAL __attribute__((visibility("hidden")))`

### 10.8.1 Macro Definition Documentation

#### 10.8.1.1 FLYCAPTURE2\_API

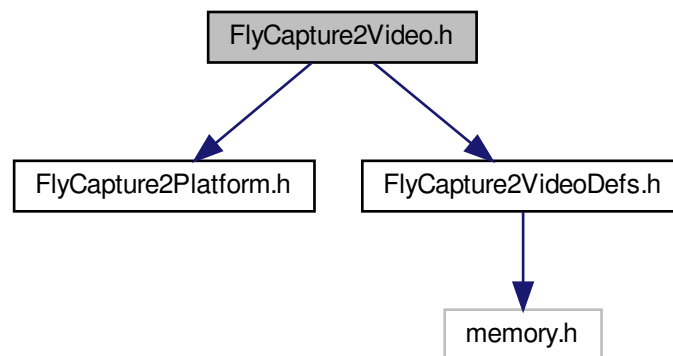
```
#define FLYCAPTURE2_API __attribute__((visibility("default")))
```

#### 10.8.1.2 FLYCAPTURE2\_LOCAL

```
#define FLYCAPTURE2_LOCAL __attribute__((visibility("hidden")))
```

## 10.9 FlyCapture2Video.h File Reference

Include dependency graph for FlyCapture2Video.h:



## Classes

- class `FlyCapture2Video`

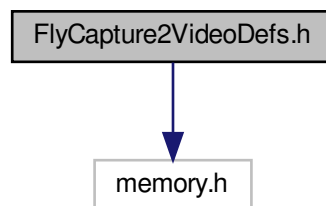
The `FlyCapture2Video` class provides the functionality for the user to record images to an AVI file.

## Namespaces

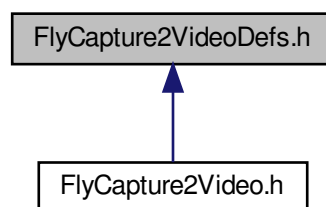
- [FlyCapture2](#)

## 10.10 FlyCapture2VideoDefs.h File Reference

Include dependency graph for FlyCapture2VideoDefs.h:



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



## Classes

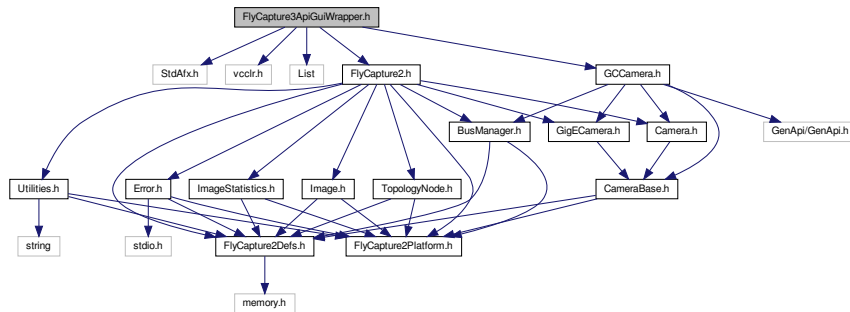
- struct [MJPGOption](#)  
*Options for saving MJPG files.*
- struct [H264Option](#)  
*Options for saving H264 files.*
- struct [AVIOption](#)  
*Options for saving AVI files.*

## Namespaces

- [FlyCapture2](#)

## 10.11 FlyCapture3ApiGuiWrapper.h File Reference

Include dependency graph for FlyCapture3ApiGuiWrapper.h:



### Classes

- class [FlyCapture3ApiGuiWrapper](#)

### Namespaces

- [FlyCapture2](#)
- [FlyCap3CameraControl](#)

### Macros

- `#define WRAPPER\_API __declspec(dllimport)`

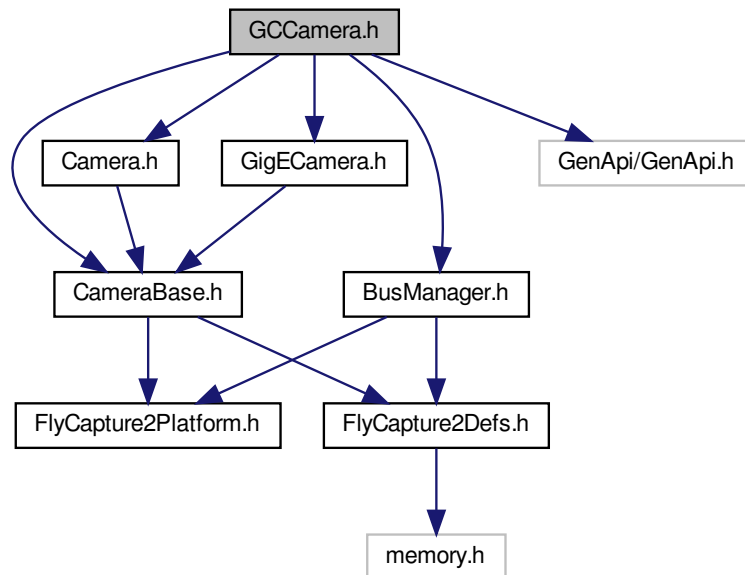
#### 10.11.1 Macro Definition Documentation

##### 10.11.1.1 WRAPPER\_API

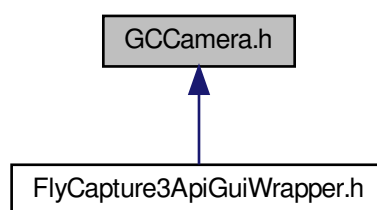
```
#define WRAPPER_API __declspec(dllimport)
```

## 10.12 GCCamera.h File Reference

Include dependency graph for GCCamera.h:



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



### Classes

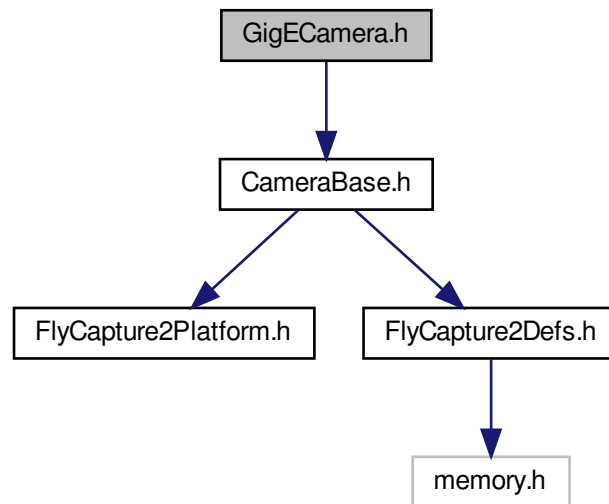
- class [GCCamera](#)

### Namespaces

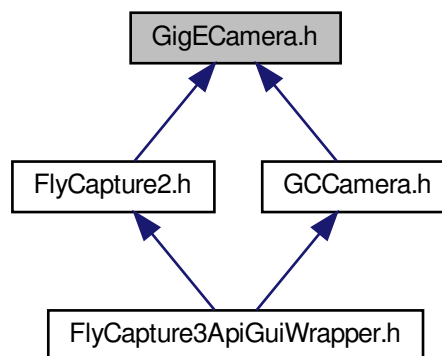
- [FlyCapture2](#)

## 10.13 GigECamera.h File Reference

Include dependency graph for GigECamera.h:



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



### Classes

- class [GigECamera](#)

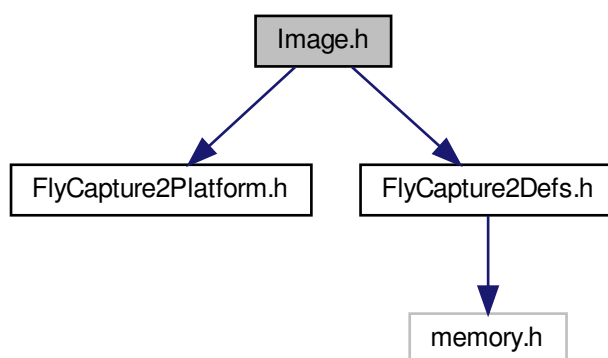
The [GigECamera](#) object represents a physical Gigabit Ethernet camera.

## Namespaces

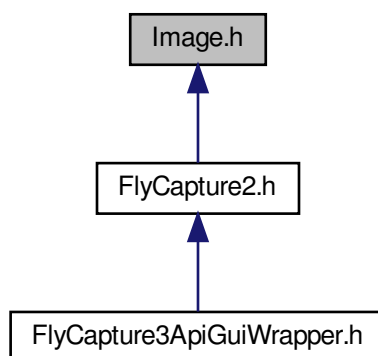
- [FlyCapture2](#)

## 10.14 Image.h File Reference

Include dependency graph for Image.h:



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



## Classes

- class [Image](#)

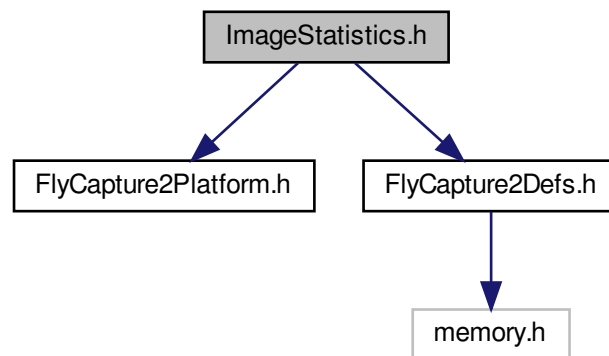
The [Image](#) class is used to retrieve images from a camera, convert between multiple pixel formats and save images to disk.

## Namespaces

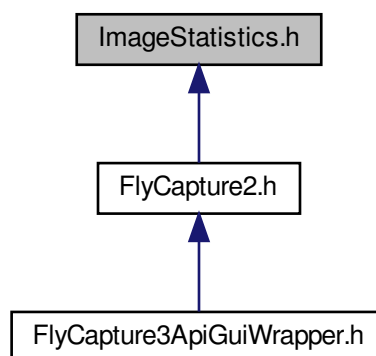
- [FlyCapture2](#)

## 10.15 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

- [FlyCapture2](#)

## 10.16 Internal.h File Reference

### Classes

- class [Internal](#)

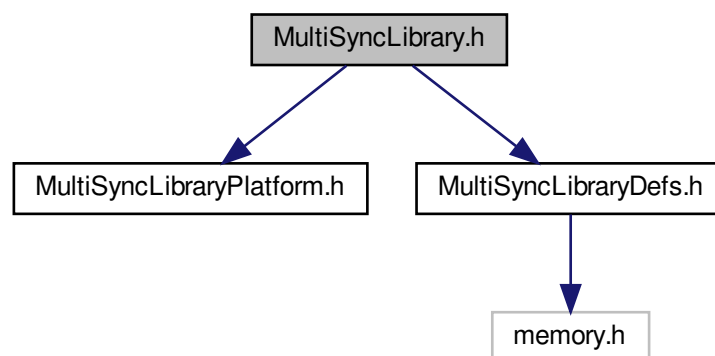
## Namespaces

- [FlyCapture2](#)

## 10.17 Licensing.dox File Reference

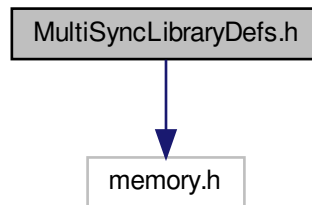
## 10.18 MultiSyncLibrary.h File Reference

Include dependency graph for MultiSyncLibrary.h:

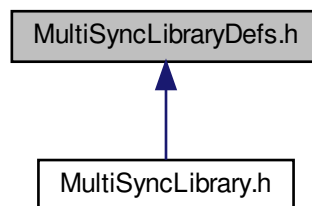


## 10.19 MultiSyncLibraryDefs.h File Reference

Include dependency graph for MultiSyncLibraryDefs.h:



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



### Classes

- class [SyncManager](#)

### Namespaces

- [MultiSyncLibrary](#)

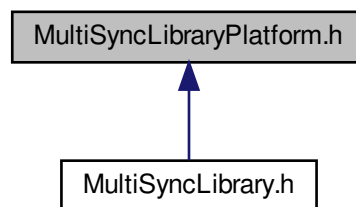
### Enumerations

- enum [PGRSyncError](#) {  
    [PGRSyncError\\_OK](#) = 0,  
    [PGRSyncError\\_FAILED](#),  
    [PGRSyncError\\_ALREADY\\_STARTED](#),  
    [PGRSyncError\\_ALREADY\\_STOPPED](#),  
    [PGRSyncError\\_CAMERA\\_NOT\\_FOUND](#),  
    [PGRSyncError\\_UNKNOWN\\_ERROR](#) }

- enum PGRSyncMessage {  
PGRSyncMessage\_OK = 0,  
PGRSyncMessage\_STARTED,  
PGRSyncMessage\_STOPPED,  
PGRSyncMessage\_SYNCING,  
PGRSyncMessage\_NOMASTER,  
PGRSyncMessage\_THREAD\_ERROR,  
PGRSyncMessage\_DEVICE\_ERROR,  
PGRSyncMessage\_NOT\_ENOUGH\_DEVICES,  
PGRSyncMessage\_BUS\_RESET,  
PGRSyncMessage\_NOT\_INITIALIZED,  
PGRSyncMessage\_UNKNOWN\_ERROR }

## 10.20 MultiSyncLibraryPlatform.h File Reference

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



### Macros

- #define `MULTISYNCLIBRARY_API __attribute__((visibility ("default")))`
- #define `MULTISYNCLIBRARY_LOCAL __attribute__((visibility ("hidden")))`

### 10.20.1 Macro Definition Documentation

#### 10.20.1.1 MULTISYNCLIBRARY\_API

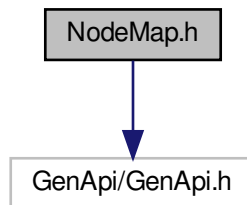
```
#define MULTISYNCLIBRARY_API __attribute__((visibility ("default")))
```

#### 10.20.1.2 MULTISYNCLIBRARY\_LOCAL

```
#define MULTISYNCLIBRARY_LOCAL __attribute__((visibility ("hidden")))
```

## 10.21 NodeMap.h File Reference

Include dependency graph for NodeMap.h:



### Classes

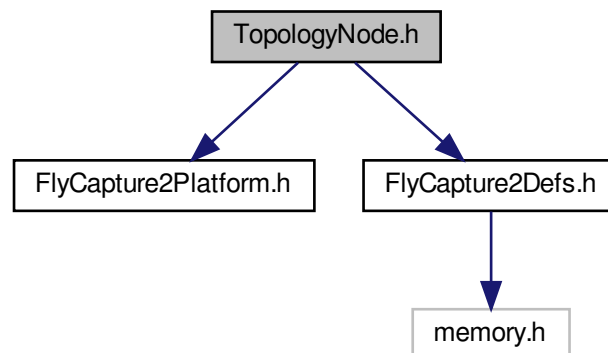
- class [NodeMap](#)

### Namespaces

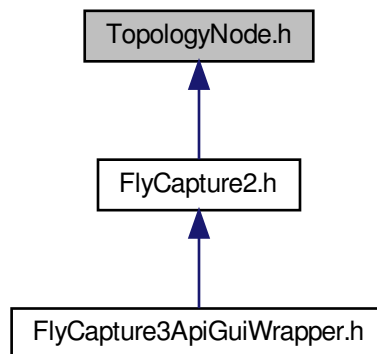
- [FlyCapture2](#)

## 10.22 TopologyNode.h File Reference

Include dependency graph for TopologyNode.h:



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



## Classes

- class [TopologyNode](#)

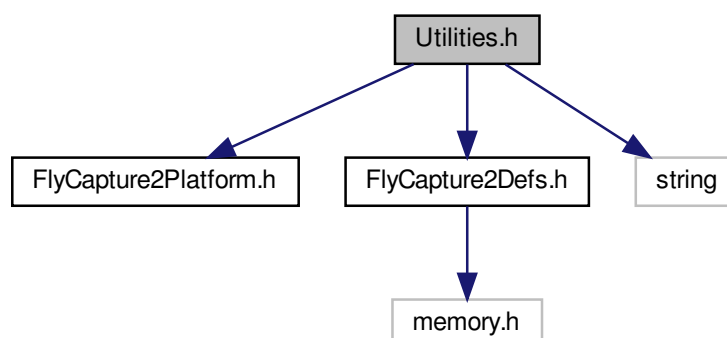
The [TopologyNode](#) class contains topology information that can be used to generate a tree structure of all cameras and devices connected to a computer.

## Namespaces

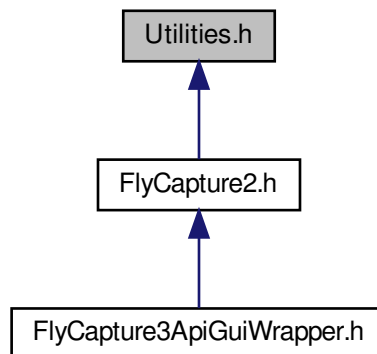
- [FlyCapture2](#)

## 10.23 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

- [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

- \_GetDeviceName
    - FlyCapture2::NodeMap, [297](#)
  - \_GetNode
    - FlyCapture2::NodeMap, [297](#)
  - \_GetNodes
    - FlyCapture2::NodeMap, [297](#)
  - \_InvalidateNodes
    - FlyCapture2::NodeMap, [297](#)
  - \_Poll
    - FlyCapture2::NodeMap, [297](#)
- ~BusManager
  - FlyCapture2::BusManager, [55](#)
- ~Camera
  - FlyCapture2::Camera, [70](#)
- ~CameraBase
  - FlyCapture2::CameraBase, [102](#)
- ~CameraControlDlg
  - FlyCapture2::CameraControlDlg, [129](#)
- ~CameraSelectionDlg
  - FlyCapture2::CameraSelectionDlg, [140](#)
- ~Error
  - FlyCapture2::Error, [153](#)
- ~FlyCapture2Video
  - FlyCapture2::FlyCapture2Video, [165](#)
- ~FlyCapture3ApiGuiWrapper
  - FlyCap3CameraControl::FlyCapture3ApiGui←  
Wrapper, [169](#)
- ~GCCamera
  - FlyCapture2::GCCamera, [181](#)
- ~GigECamera
  - FlyCapture2::GigECamera, [212](#)
- ~Image
  - FlyCapture2::Image, [261](#)
- ~ImageStatistics
  - FlyCapture2::ImageStatistics, [280](#)
- ~NodeMap
  - FlyCapture2::NodeMap, [297](#)
- ~SyncManager
  - MultiSyncLibrary::SyncManager, [315](#)
- ~TopologyNode
  - FlyCapture2::TopologyNode, [326](#)
- AVIOption, [51](#)
  - FlyCapture2::AVIOption, [51](#)
- absControl
  - FlyCapture2::Property, [304](#)
- absMax
  - FlyCapture2::PropertyInfo, [308](#)
- absMin
  - FlyCapture2::PropertyInfo, [308](#)
- absValSupported
  - FlyCapture2::PropertyInfo, [308](#)
- absValue
  - FlyCapture2::Property, [304](#)
- AddChild
  - FlyCapture2::TopologyNode, [326](#)
- AddPortType
  - FlyCapture2::TopologyNode, [326](#)
- Append
  - FlyCapture2::FlyCapture2Video, [165](#)
- applicationIPAddress
  - FlyCapture2::CameraInfo, [134](#)
- applicationPort
  - FlyCapture2::CameraInfo, [134](#)
- AssignGuidToNode
  - FlyCapture2::TopologyNode, [326](#), [327](#)
- asyncBusSpeed
  - FlyCapture2::FC2Config, [160](#)
- AsyncCommandCallback
  - FlyCapture2, [47](#)
- autoManualMode
  - FlyCapture2::Property, [305](#)
- autoSupported
  - FlyCapture2::PropertyInfo, [308](#)
- available
  - FlyCapture2::EmbeddedImageInfoProperty, [151](#)
- BMPOption, [52](#)
  - FlyCapture2::BMPOption, [52](#)
- BandwidthAllocation
  - Enumerations, [20](#)
- bandwidthAllocation
  - FlyCapture2::FC2Config, [161](#)
- BayerTileFormat
  - Enumerations, [21](#)
- bayerTileFormat
  - FlyCapture2::CameraInfo, [134](#)
- binaryFile
  - FlyCapture2::PGMOption, [298](#)
  - FlyCapture2::PPMOption, [302](#)
- bitrate
  - FlyCapture2::H264Option, [256](#)
- brightness
  - FlyCapture2::EmbeddedImageInfo, [149](#)
- build
  - FlyCapture2::FC2Version, [164](#)
- BusCallbackType
  - Enumerations, [21](#)
- BusEventCallback
  - FlyCapture2, [47](#)

- BusManager, [53](#)
  - FlyCapture2::BusManager, [55](#)
- BusManager.h, [339](#)
- busNumber
  - FlyCapture2::CameraInfo, [134](#)
- BusSpeed
  - Enumerations, [21](#)
- ByteOrder
  - FlyCapture2, [48](#)
- byteOrder
  - FlyCapture2::SystemInfo, [317](#)
- CalculateStatistics
  - FlyCapture2::Image, [261](#)
- CallbackHandle
  - FlyCapture2, [47](#)
- Camera, [65](#)
  - FlyCapture2::Camera, [69](#)
- Camera.h, [341](#)
- CameraBase, [99](#)
  - FlyCapture2::CameraBase, [102](#)
- CameraBase.h, [342](#)
- CameraControlDlg, [128](#)
  - FlyCapture2::CameraControlDlg, [129](#)
- cameraCurrents
  - FlyCapture2::CameraStats, [142](#)
- CameraEventCallback
  - Image saving structures., [36](#)
- CameraInfo, [132](#)
  - FlyCapture2::CameraInfo, [134](#)
- cameraPowerUp
  - FlyCapture2::CameraStats, [142](#)
- CameraSelectionDlg, [139](#)
  - FlyCapture2::CameraSelectionDlg, [140](#)
- CameraStats, [141](#)
  - FlyCapture2::CameraStats, [142](#)
- cameraVoltages
  - FlyCapture2::CameraStats, [142](#)
- ccpStatus
  - FlyCapture2::CameraInfo, [134](#)
- CheckDriver
  - FlyCapture2::Utilities, [335](#)
- chipIdHi
  - FlyCapture2::ConfigROM, [146](#)
- chipIdLo
  - FlyCapture2::ConfigROM, [146](#)
- Close
  - FlyCapture2::FlyCapture2Video, [166](#)
- CollectSupportInformation
  - FlyCapture2::Error, [153](#)
- ColorProcessingAlgorithm
  - Enumerations, [22](#)
- compression
  - FlyCapture2::TIFFOption, [321](#)
- compressionLevel
  - FlyCapture2::PNGOption, [301](#)
- CompressionMethod
  - FlyCapture2::TIFFOption, [320](#)
- ConfigROM, [145](#)
  - FlyCapture2::ConfigROM, [145](#)
- configROM
  - FlyCapture2::CameraInfo, [135](#)
- Connect
  - FlyCapture2::Camera, [70](#)
  - FlyCapture2::CameraBase, [103](#)
  - FlyCapture2::CameraControlDlg, [130](#)
  - FlyCapture2::GCCamera, [181](#)
  - FlyCapture2::GigECamera, [212](#)
- ConnectGUILibrary
  - FlyCap3CameraControl::FlyCapture3ApiGui↔  
Wrapper, [169](#)
- Convert
  - FlyCapture2::Image, [262](#)
- cpuDescription
  - FlyCapture2::SystemInfo, [318](#)
- cycleCount
  - FlyCapture2::TimeStamp, [322](#)
- cycleOffset
  - FlyCapture2::TimeStamp, [322](#)
- cycleSeconds
  - FlyCapture2::TimeStamp, [322](#)
- DeepCopy
  - FlyCapture2::Image, [263](#)
- defaultGateway
  - FlyCapture2::CameraInfo, [135](#)
- delay
  - FlyCapture2::StrobeControl, [311](#)
- DeregisterAllEvents
  - FlyCapture2::Camera, [70](#)
  - FlyCapture2::CameraBase, [103](#)
  - FlyCapture2::GigECamera, [212](#)
- DeregisterEvent
  - FlyCapture2::Camera, [70](#)
  - FlyCapture2::CameraBase, [103](#)
  - FlyCapture2::GigECamera, [212](#)
- destinationIpAddress
  - FlyCapture2::GigEStreamChannel, [254](#)
- DetermineBitsPerPixel
  - FlyCapture2::Image, [263](#)
- DisableAll
  - FlyCapture2::ImageStatistics, [281](#)
- DisableCrossPCSSynchronization
  - MultiSyncLibrary::SyncManager, [315](#)
- Disconnect
  - FlyCapture2::Camera, [70](#)
  - FlyCapture2::CameraBase, [104](#)
  - FlyCapture2::CameraControlDlg, [130](#)
  - FlyCapture2::GCCamera, [181](#)
  - FlyCapture2::GigECamera, [213](#)
- DisconnectGUILibrary
  - FlyCap3CameraControl::FlyCapture3ApiGui↔  
Wrapper, [169](#)
- DiscoverGigECameras
  - FlyCapture2::BusManager, [55](#)
- DiscoverGigEPacketSize
  - FlyCapture2::GigECamera, [213](#)
- doNotFragment



- FlyCapture2::GigEStreamChannel, [254](#)
- driverList
  - FlyCapture2::SystemInfo, [318](#)
- driverName
  - FlyCapture2::CameraInfo, [135](#)
- DriverType
  - Enumerations, [23](#)
- driverType
  - FlyCapture2::CameraInfo, [135](#)
- duration
  - FlyCapture2::StrobeControl, [311](#)
- embeddedBrightness
  - FlyCapture2::ImageMetadata, [277](#)
- embeddedExposure
  - FlyCapture2::ImageMetadata, [277](#)
- embeddedFrameCounter
  - FlyCapture2::ImageMetadata, [277](#)
- embeddedGPIOPinState
  - FlyCapture2::ImageMetadata, [277](#)
- embeddedGain
  - FlyCapture2::ImageMetadata, [277](#)
- EmbeddedImageInfo, [148](#)
- EmbeddedImageInfoProperty, [150](#)
  - FlyCapture2::EmbeddedImageInfoProperty, [151](#)
- embeddedROIPosition
  - FlyCapture2::ImageMetadata, [277](#)
- embeddedShutter
  - FlyCapture2::ImageMetadata, [278](#)
- embeddedStrobePattern
  - FlyCapture2::ImageMetadata, [278](#)
- embeddedTimeStamp
  - FlyCapture2::ImageMetadata, [278](#)
- embeddedWhiteBalance
  - FlyCapture2::ImageMetadata, [278](#)
- EnableAll
  - FlyCapture2::ImageStatistics, [281](#)
- EnableCrossPCSSynchronization
  - MultiSyncLibrary::SyncManager, [316](#)
- EnableGreyOnly
  - FlyCapture2::ImageStatistics, [281](#)
- EnableHSLOnly
  - FlyCapture2::ImageStatistics, [282](#)
- EnableLUT
  - FlyCapture2::Camera, [71](#)
  - FlyCapture2::CameraBase, [104](#)
  - FlyCapture2::GCCamera, [182](#)
  - FlyCapture2::GigECamera, [213](#)
- enablePacketResend
  - FlyCapture2::GigEConfig, [246](#)
- EnableRGBOnly
  - FlyCapture2::ImageStatistics, [282](#)
- enabled
  - FlyCapture2::LUTData, [292](#)
- Enumerations, [16](#)
  - BandwidthAllocation, [20](#)
  - BayerTileFormat, [21](#)
  - BusCallbackType, [21](#)
  - BusSpeed, [21](#)
  - ColorProcessingAlgorithm, [22](#)
  - DriverType, [23](#)
  - ErrorType, [23](#)
  - FrameRate, [24](#)
  - GrabMode, [25](#)
  - GrabTimeout, [25](#)
  - ImageFileFormat, [26](#)
  - InterfaceType, [26](#)
  - Mode, [26](#)
  - PCleBusSpeed, [27](#)
  - PixelFormat, [28](#)
  - PropertyType, [28](#)
  - VideoMode, [29](#)
- Error, [151](#)
  - FlyCapture2::Error, [152, 153](#)
- Error.h, [343](#)
- ErrorType
  - Enumerations, [23](#)
- EventCallbackData, [156](#)
- EventCallbackFcn
  - FlyCapture2::EventOptions, [158](#)
- EventData
  - FlyCapture2::EventCallbackData, [157](#)
- EventDataSize
  - FlyCapture2::EventCallbackData, [157](#)
- EventID
  - FlyCapture2::EventCallbackData, [157](#)
- EventName
  - FlyCapture2::EventCallbackData, [157](#)
  - FlyCapture2::EventOptions, [159](#)
- EventOptions, [158](#)
- EventTimestamp
  - FlyCapture2::EventCallbackData, [157](#)
- EventUserData
  - FlyCapture2::EventCallbackData, [157](#)
  - FlyCapture2::EventOptions, [159](#)
- EventUserDataSize
  - FlyCapture2::EventCallbackData, [158](#)
  - FlyCapture2::EventOptions, [159](#)
- exposure
  - FlyCapture2::EmbeddedImageInfo, [149](#)
- FC2Config, [159](#)
  - FlyCapture2::FC2Config, [160](#)
- FC2Version, [163](#)
- FLYCAPTURE2\_API
  - FlyCapture2Platform.h, [354](#)
- FLYCAPTURE2\_LOCAL
  - FlyCapture2Platform.h, [354](#)
- FULL\_32BIT\_VALUE
  - FlyCapture2Defs.h, [352](#)
- FireBusReset
  - FlyCapture2::BusManager, [55](#)
- FireSoftwareTrigger
  - FlyCapture2::Camera, [71](#)
  - FlyCapture2::CameraBase, [105](#)
  - FlyCapture2::GCCamera, [182](#)
  - FlyCapture2::GigECamera, [214](#)
- firmwareBuildTime

- FlyCapture2::CameraInfo, 135
- firmwareVersion
  - FlyCapture2::CameraInfo, 135
- FlyCap3CameraControl, 39
- FlyCap3CameraControl::FlyCapture3ApiGuiWrapper
  - ~FlyCapture3ApiGuiWrapper, 169
  - ConnectGUILibrary, 169
  - DisconnectGUILibrary, 169
  - FlyCapture3ApiGuiWrapper, 169
  - GetControlNameList, 169
  - GetDialogNameList, 169
  - GetNumDialogs, 169
  - GetNumOfControls, 170
  - ShowCameraSelectionDialog, 170
  - ShowDialogByIndex, 170
  - ShowDialogByName, 170
  - ShowPropertyGridDialog, 170
- FlyCapture2, 39
  - AsyncCommandCallback, 47
  - BusEventCallback, 47
  - ByteOrder, 48
  - CallbackHandle, 47
  - ImageEventCallback, 47
  - OSType, 48
- FlyCapture2.h, 344
- FlyCapture2::AVIOption
  - AVIOption, 51
  - frameRate, 52
  - reserved, 52
- FlyCapture2::BMPOption
  - BMPOption, 52
  - indexedColor\_8bit, 53
  - reserved, 53
- FlyCapture2::BusManager
  - ~BusManager, 55
  - BusManager, 55
  - DiscoverGigECameras, 55
  - FireBusReset, 55
  - ForceAllIPAddressesAutomatically, 56
  - ForceIPAddressToCamera, 56
  - GetCameraFromIPAddress, 57
  - GetCameraFromIndex, 57
  - GetCameraFromSerialNumber, 58
  - GetCameraSerialNumberFromIndex, 58
  - GetDeviceFromIndex, 59
  - GetInterfaceTypeFromGuid, 59
  - GetNumOfCameras, 60
  - GetNumOfDevices, 60
  - GetTopology, 60
  - GetUsbLinkInfo, 61
  - GetUsbPortStatus, 61
  - IsCameraControlable, 62
  - ReadPhyRegister, 63
  - RegisterCallback, 63
  - RescanBus, 64
  - UnregisterCallback, 64
  - WritePhyRegister, 65
- FlyCapture2::Camera
  - ~Camera, 70
  - Camera, 69
  - Connect, 70
  - DeregisterAllEvents, 70
  - DeregisterEvent, 70
  - Disconnect, 70
  - EnableLUT, 71
  - FireSoftwareTrigger, 71
  - GetActiveLUTBank, 72
  - GetCameraInfo, 72
  - GetConfiguration, 73
  - GetCycleTime, 73
  - GetEmbeddedImageInfo, 73
  - GetFormat7Configuration, 74
  - GetFormat7Info, 74
  - GetGPIOPinDirection, 75
  - GetLUTBankInfo, 76
  - GetLUTChannel, 76
  - GetLUTInfo, 77
  - GetMemoryChannel, 77
  - GetMemoryChannelInfo, 78
  - GetProperty, 78
  - GetPropertyInfo, 79
  - GetRegisterString, 79
  - GetStats, 79
  - GetStrobe, 80
  - GetStrobeInfo, 80
  - GetTriggerDelay, 81
  - GetTriggerDelayInfo, 81
  - GetTriggerMode, 82
  - GetTriggerModeInfo, 82
  - GetVideoModeAndFrameRate, 83
  - GetVideoModeAndFrameRateInfo, 83
  - IsConnected, 84
  - ReadRegister, 84
  - ReadRegisterBlock, 85
  - RegisterAllEvents, 85
  - RegisterEvent, 85
  - ResetStats, 86
  - RestoreFromMemoryChannel, 86
  - RetrieveBuffer, 86
  - SaveToMemoryChannel, 87
  - SetActiveLUTBank, 87
  - SetCallback, 88
  - SetConfiguration, 88
  - SetEmbeddedImageInfo, 89
  - SetFormat7Configuration, 89, 90
  - SetGPIOPinDirection, 90
  - SetLUTChannel, 91
  - SetProperty, 91
  - SetStrobe, 92
  - SetTriggerDelay, 93
  - SetTriggerMode, 93
  - SetUserBuffers, 94
  - SetVideoModeAndFrameRate, 94
  - StartCapture, 95
  - StartSyncCapture, 96
  - StopCapture, 96

- ValidateFormat7Settings, 96
- WaitForBufferEvent, 97
- WriteRegister, 97
- WriteRegisterBlock, 98
- FlyCapture2::CameraBase
  - ~CameraBase, 102
  - CameraBase, 102
  - Connect, 103
  - DeregisterAllEvents, 103
  - DeregisterEvent, 103
  - Disconnect, 104
  - EnableLUT, 104
  - FireSoftwareTrigger, 105
  - GetActiveLUTBank, 105
  - GetCameraInfo, 105
  - GetConfiguration, 106
  - GetCycleTime, 106
  - GetEmbeddedImageInfo, 107
  - GetGPIOPinDirection, 107
  - GetLUTBankInfo, 108
  - GetLUTChannel, 108
  - GetLUTInfo, 109
  - GetMemoryChannel, 109
  - GetMemoryChannelInfo, 110
  - GetProperty, 110
  - GetPropertyInfo, 111
  - GetRegisterString, 111
  - GetStats, 112
  - GetStrobe, 112
  - GetStrobeInfo, 112
  - GetTriggerDelay, 113
  - GetTriggerDelayInfo, 113
  - GetTriggerMode, 114
  - GetTriggerModeInfo, 114
  - IsConnected, 115
  - m\_pCameraData, 128
  - ReadRegister, 115
  - ReadRegisterBlock, 116
  - RegisterAllEvents, 116
  - RegisterEvent, 116
  - ResetStats, 117
  - RestoreFromMemoryChannel, 117
  - RetrieveBuffer, 117
  - SaveToMemoryChannel, 118
  - SetActiveLUTBank, 118
  - SetCallback, 119
  - SetConfiguration, 119
  - SetEmbeddedImageInfo, 120
  - SetGPIOPinDirection, 120
  - SetLUTChannel, 121
  - SetProperty, 121
  - SetStrobe, 122
  - SetTriggerDelay, 123
  - SetTriggerMode, 123
  - SetUserBuffers, 124
  - StartCapture, 124
  - StartSyncCapture, 125
  - StopCapture, 126
  - WaitForBufferEvent, 126
  - WriteRegister, 127
  - WriteRegisterBlock, 127
- FlyCapture2::CameraControlDlg
  - ~CameraControlDlg, 129
  - CameraControlDlg, 129
  - Connect, 130
  - Disconnect, 130
  - Hide, 130
  - IsVisible, 130
  - SetTitle, 130
  - Show, 131
  - ShowModal, 131
- FlyCapture2::CameraInfo
  - applicationIPAddress, 134
  - applicationPort, 134
  - bayerTileFormat, 134
  - busNumber, 134
  - CameraInfo, 134
  - ccpStatus, 134
  - configROM, 135
  - defaultGateway, 135
  - driverName, 135
  - driverType, 135
  - firmwareBuildTime, 135
  - firmwareVersion, 135
  - gigEMajorVersion, 136
  - gigEMinorVersion, 136
  - iidcVer, 136
  - interfaceType, 136
  - ipAddress, 136
  - isColorCamera, 136
  - macAddress, 137
  - maximumBusSpeed, 137
  - modelName, 137
  - nodeNumber, 137
  - pcieBusSpeed, 137
  - reserved, 137
  - sensorInfo, 138
  - sensorResolution, 138
  - serialNumber, 138
  - subnetMask, 138
  - userDefinedName, 138
  - vendorName, 138
  - xmlURL1, 139
  - xmlURL2, 139
- FlyCapture2::CameraSelectionDlg
  - ~CameraSelectionDlg, 140
  - CameraSelectionDlg, 140
  - SetTitle, 140
  - ShowModal, 140
- FlyCapture2::CameraStats
  - cameraCurrents, 142
  - cameraPowerUp, 142
  - CameraStats, 142
  - cameraVoltages, 142
  - imageCorrupt, 142
  - imageDriverDropped, 142

- imageDropped, [142](#)
- imageXmitFailed, [143](#)
- numCurrents, [143](#)
- numResendPacketsReceived, [143](#)
- numResendPacketsRequested, [143](#)
- numVoltages, [143](#)
- portErrors, [143](#)
- regReadFailed, [143](#)
- regWriteFailed, [144](#)
- reserved, [144](#)
- temperature, [144](#)
- timeSinceBusReset, [144](#)
- timeSinceInitialization, [144](#)
- timeStamp, [144](#)
- FlyCapture2::ConfigROM
  - chipIdHi, [146](#)
  - chipIdLo, [146](#)
  - ConfigROM, [145](#)
  - nodeVendorId, [146](#)
  - pszKeyword, [146](#)
  - reserved, [146](#)
  - unitSWVer, [147](#)
  - unitSpecId, [146](#)
  - unitSubSWVer, [147](#)
  - vendorUniqueInfo\_0, [147](#)
  - vendorUniqueInfo\_1, [147](#)
  - vendorUniqueInfo\_2, [147](#)
  - vendorUniqueInfo\_3, [147](#)
- FlyCapture2::EmbeddedImageInfo
  - brightness, [149](#)
  - exposure, [149](#)
  - frameCounter, [149](#)
  - GPIOPinState, [149](#)
  - gain, [149](#)
  - ROIPosition, [149](#)
  - shutter, [149](#)
  - strobePattern, [150](#)
  - timestamp, [150](#)
  - whiteBalance, [150](#)
- FlyCapture2::EmbeddedImageInfoProperty
  - available, [151](#)
  - EmbeddedImageInfoProperty, [151](#)
  - onOff, [151](#)
- FlyCapture2::Error
  - ~Error, [153](#)
  - CollectSupportInformation, [153](#)
  - Error, [152](#), [153](#)
  - GetBuildDate, [153](#)
  - GetCause, [153](#)
  - GetDescription, [154](#)
  - GetFilename, [154](#)
  - GetLine, [154](#)
  - GetType, [154](#)
  - InternalError, [156](#)
  - operator!=, [155](#)
  - operator=, [155](#)
  - operator==, [155](#)
  - PrintErrorTrace, [156](#)
- FlyCapture2::EventCallbackData
  - EventData, [157](#)
  - EventDataSize, [157](#)
  - EventID, [157](#)
  - EventName, [157](#)
  - EventTimestamp, [157](#)
  - EventUserData, [157](#)
  - EventUserDataSize, [158](#)
- FlyCapture2::EventOptions
  - EventCallbackFcn, [158](#)
  - EventName, [159](#)
  - EventUserData, [159](#)
  - EventUserDataSize, [159](#)
- FlyCapture2::FC2Config
  - asyncBusSpeed, [160](#)
  - bandwidthAllocation, [161](#)
  - FC2Config, [160](#)
  - grabMode, [161](#)
  - grabTimeout, [161](#)
  - highPerformanceRetrieveBuffer, [161](#)
  - isochBusSpeed, [161](#)
  - minNumImageNotifications, [162](#)
  - numBuffers, [162](#)
  - numImageNotifications, [162](#)
  - registerTimeout, [162](#)
  - registerTimeoutRetries, [163](#)
  - reserved, [163](#)
- FlyCapture2::FC2Version
  - build, [164](#)
  - major, [164](#)
  - minor, [164](#)
  - type, [164](#)
- FlyCapture2::FlyCapture2Video
  - ~FlyCapture2Video, [165](#)
  - Append, [165](#)
  - Close, [166](#)
  - FlyCapture2Video, [165](#)
  - Open, [166](#), [167](#)
  - SetMaximumFileSize, [168](#)
- FlyCapture2::Format7ImageSettings
  - Format7ImageSettings, [171](#)
  - height, [171](#)
  - mode, [171](#)
  - offsetX, [172](#)
  - offsetY, [172](#)
  - pixelFormat, [172](#)
  - reserved, [172](#)
  - width, [172](#)
- FlyCapture2::Format7Info
  - Format7Info, [173](#)
  - imageHStepSize, [174](#)
  - imageVStepSize, [174](#)
  - maxHeight, [174](#)
  - maxPacketSize, [174](#)
  - maxWidth, [174](#)
  - minPacketSize, [174](#)
  - mode, [175](#)
  - offsetHStepSize, [175](#)

- offsetVStepSize, [175](#)
- packetSize, [175](#)
- percentage, [175](#)
- pixelFormatBitField, [175](#)
- reserved, [176](#)
- vendorPixelFormatBitField, [176](#)
- FlyCapture2::Format7PacketInfo
  - Format7PacketInfo, [177](#)
  - maxBytesPerPacket, [177](#)
  - recommendedBytesPerPacket, [177](#)
  - reserved, [177](#)
  - unitBytesPerPacket, [177](#)
- FlyCapture2::GCCamera
  - ~GCCamera, [181](#)
  - Connect, [181](#)
  - Disconnect, [181](#)
  - EnableLUT, [182](#)
  - FireSoftwareTrigger, [182](#)
  - GCCamera, [181](#)
  - GCCamera::GetXML, [183](#)
  - GetActiveLUTBank, [183](#)
  - GetCameraInfo, [183](#)
  - GetConfiguration, [184](#)
  - GetCycleTime, [184](#)
  - GetEmbeddedImageInfo, [185](#)
  - GetGPIOPinDirection, [185](#)
  - GetInterfaceType, [186](#)
  - GetLUTBankInfo, [186](#)
  - GetLUTChannel, [186](#)
  - GetLUTInfo, [187](#)
  - GetMemoryChannel, [187](#)
  - GetMemoryChannelInfo, [188](#)
  - GetNodeMap, [188](#)
  - GetProperty, [189](#)
  - GetPropertyInfo, [189](#)
  - GetRegisterString, [190](#)
  - GetStats, [190](#)
  - GetStrobe, [190](#)
  - GetStrobeInfo, [191](#)
  - GetTriggerDelay, [191](#)
  - GetTriggerDelayInfo, [192](#)
  - GetTriggerMode, [192](#)
  - GetTriggerModeInfo, [193](#)
  - IsConnected, [193](#)
  - m\_busMgr, [207](#)
  - ReadGVCPMemory, [193](#)
  - ReadGVCPRegister, [194](#)
  - ReadGVCPRegisterBlock, [194](#)
  - ReadRegister, [194](#)
  - ReadRegisterBlock, [194](#)
  - ResetStats, [195](#)
  - RestoreFromMemoryChannel, [195](#)
  - RetrieveBuffer, [196](#)
  - SaveToMemoryChannel, [196](#)
  - SetActiveLUTBank, [197](#)
  - SetCallback, [197](#)
  - SetCamera, [198](#)
  - SetConfiguration, [198](#)
  - SetEmbeddedImageInfo, [199](#)
  - SetGPIOPinDirection, [199](#)
  - SetLUTChannel, [200](#)
  - SetProperty, [200](#)
  - SetStrobe, [201](#)
  - SetTriggerDelay, [201](#)
  - SetTriggerMode, [202](#)
  - SetUserBuffers, [202](#)
  - StartCapture, [203](#)
  - StartSyncCapture, [204](#)
  - StopCapture, [204](#)
  - TestGainNode, [204](#)
  - WaitForBufferEvent, [205](#)
  - WriteGVCPMemory, [205](#)
  - WriteGVCPRegister, [205](#)
  - WriteGVCPRegisterBlock, [205](#)
  - WriteRegister, [206](#)
  - WriteRegisterBlock, [206](#)
- FlyCapture2::GigECamera
  - ~GigECamera, [212](#)
  - Connect, [212](#)
  - DeregisterAllEvents, [212](#)
  - DeregisterEvent, [212](#)
  - Disconnect, [213](#)
  - DiscoverGigEPacketSize, [213](#)
  - EnableLUT, [213](#)
  - FireSoftwareTrigger, [214](#)
  - GetActiveLUTBank, [214](#)
  - GetCameraInfo, [215](#)
  - GetConfiguration, [215](#)
  - GetCycleTime, [216](#)
  - GetEmbeddedImageInfo, [216](#)
  - GetGPIOPinDirection, [219](#)
  - GetGigEConfig, [216](#)
  - GetGigEImageBinningSettings, [217](#)
  - GetGigEImageSettings, [217](#)
  - GetGigEImageSettingsInfo, [218](#)
  - GetGigEImagingMode, [218](#)
  - GetGigEProperty, [218](#)
  - GetGigEStreamChannelInfo, [219](#)
  - GetLUTBankInfo, [220](#)
  - GetLUTChannel, [220](#)
  - GetLUTInfo, [221](#)
  - GetMemoryChannel, [221](#)
  - GetMemoryChannelInfo, [222](#)
  - GetNumStreamChannels, [222](#)
  - GetProperty, [223](#)
  - GetPropertyInfo, [223](#)
  - GetRegisterString, [224](#)
  - GetStats, [224](#)
  - GetStrobe, [224](#)
  - GetStrobeInfo, [225](#)
  - GetTriggerDelay, [225](#)
  - GetTriggerDelayInfo, [226](#)
  - GetTriggerMode, [226](#)
  - GetTriggerModeInfo, [227](#)
  - GigECamera, [212](#)
  - IsConnected, [227](#)

- QueryGigElmagingMode, [228](#)
- ReadGVCPMemory, [228](#)
- ReadGVCPRegister, [229](#)
- ReadGVCPRegisterBlock, [229](#)
- ReadRegister, [229](#)
- ReadRegisterBlock, [230](#)
- RegisterAllEvents, [230](#)
- RegisterEvent, [231](#)
- ResetStats, [231](#)
- RestoreFromMemoryChannel, [231](#)
- RetrieveBuffer, [231](#)
- SaveToMemoryChannel, [232](#)
- SetActiveLUTBank, [233](#)
- SetCallback, [233](#)
- SetConfiguration, [233](#)
- SetEmbeddedImageInfo, [234](#)
- SetGPIOPinDirection, [237](#)
- SetGigEConfig, [234](#)
- SetGigEImageBinningSettings, [235](#)
- SetGigEImageSettings, [235](#)
- SetGigElmagingMode, [235](#)
- SetGigEProperty, [236](#)
- SetGigEStreamChannelInfo, [236](#)
- SetLUTChannel, [237](#)
- SetProperty, [238](#)
- SetStroke, [238](#)
- SetTriggerDelay, [239](#)
- SetTriggerMode, [240](#)
- SetUserBuffers, [240](#)
- StartCapture, [241](#)
- StartSyncCapture, [242](#)
- StopCapture, [242](#)
- WaitForBufferEvent, [242](#)
- WriteGVCPMemory, [243](#)
- WriteGVCPRegister, [243](#)
- WriteGVCPRegisterBlock, [244](#)
- WriteRegister, [244](#)
- WriteRegisterBlock, [245](#)
- FlyCapture2::GigEConfig
  - enablePacketResend, [246](#)
  - GigEConfig, [246](#)
  - registerTimeout, [246](#)
  - registerTimeoutRetries, [246](#)
- FlyCapture2::GigEImageSettings
  - GigEImageSettings, [247](#)
  - height, [248](#)
  - offsetX, [248](#)
  - offsetY, [248](#)
  - pixelFormat, [248](#)
  - reserved, [248](#)
  - width, [248](#)
- FlyCapture2::GigEImageSettingsInfo
  - GigEImageSettingsInfo, [249](#)
  - imageHStepSize, [250](#)
  - imageVStepSize, [250](#)
  - maxHeight, [250](#)
  - maxWidth, [250](#)
  - offsetHStepSize, [250](#)
  - offsetVStepSize, [250](#)
  - pixelFormatBitField, [251](#)
  - reserved, [251](#)
  - vendorPixelFormatBitField, [251](#)
- FlyCapture2::GigEProperty
  - isReadable, [252](#)
  - isWritable, [252](#)
  - max, [252](#)
  - min, [252](#)
  - propType, [252](#)
  - value, [252](#)
- FlyCapture2::GigEStreamChannel
  - destinationIpAddress, [254](#)
  - doNotFragment, [254](#)
  - GigEStreamChannel, [254](#)
  - hostPort, [254](#)
  - interPacketDelay, [254](#)
  - networkInterfaceIndex, [254](#)
  - packetSize, [255](#)
  - sourcePort, [255](#)
- FlyCapture2::H264Option
  - bitrate, [256](#)
  - frameRate, [256](#)
  - H264Option, [256](#)
  - height, [256](#)
  - reserved, [256](#)
  - width, [256](#)
- FlyCapture2::IPAddress
  - IPAddress, [287](#)
  - octets, [288](#)
  - operator!=, [288](#)
  - operator==, [288](#)
- FlyCapture2::Image
  - ~Image, [261](#)
  - CalculateStatistics, [261](#)
  - Convert, [262](#)
  - DeepCopy, [263](#)
  - DetermineBitsPerPixel, [263](#)
  - GetBayerTileFormat, [263](#)
  - GetBitsPerPixel, [264](#)
  - GetBlockId, [264](#)
  - GetColorProcessing, [264](#)
  - GetCols, [264](#)
  - GetData, [265](#)
  - GetDataSize, [265](#)
  - GetDefaultColorProcessing, [265](#)
  - GetDefaultOutputFormat, [266](#)
  - GetDimensions, [266](#)
  - GetMetadata, [267](#)
  - GetPixelFormat, [267](#)
  - GetReceivedDataSize, [267](#)
  - GetRows, [267](#)
  - GetStride, [268](#)
  - GetTimeStamp, [268](#)
  - Image, [259–261](#)
  - Iso, [275](#)
  - operator(), [268](#)
  - operator=, [269](#)

- operator[], [269](#)
- ReleaseBuffer, [269](#)
- Save, [269–272](#)
- SetBlockId, [272](#)
- SetColorProcessing, [273](#)
- SetData, [273](#)
- SetDefaultColorProcessing, [274](#)
- SetDefaultOutputFormat, [274](#)
- SetDimensions, [275](#)
- FlyCapture2::ImageMetadata
  - embeddedBrightness, [277](#)
  - embeddedExposure, [277](#)
  - embeddedFrameCounter, [277](#)
  - embeddedGPIOPinState, [277](#)
  - embeddedGain, [277](#)
  - embeddedROIPosition, [277](#)
  - embeddedShutter, [278](#)
  - embeddedStrobePattern, [278](#)
  - embeddedTimeStamp, [278](#)
  - embeddedWhiteBalance, [278](#)
  - ImageMetadata, [276](#)
  - reserved, [278](#)
- FlyCapture2::ImageStatistics
  - ~ImageStatistics, [280](#)
  - DisableAll, [281](#)
  - EnableAll, [281](#)
  - EnableGreyOnly, [281](#)
  - EnableHSLOnly, [282](#)
  - EnableRGBOnly, [282](#)
  - GetChannelStatus, [282](#)
  - GetHistogram, [283](#)
  - GetMean, [283](#)
  - GetNumPixelValues, [283](#)
  - GetPixelValueRange, [284](#)
  - GetRange, [284](#)
  - GetStatistics, [285](#)
  - ImageStatistics, [280, 281](#)
  - ImageStatsCalculator, [286](#)
  - operator=, [285](#)
  - SetChannelStatus, [286](#)
  - StatisticsChannel, [280](#)
- FlyCapture2::Internal
  - GetInternal, [287](#)
- FlyCapture2::JPEGOption
  - JPEGOption, [289](#)
  - progressive, [289](#)
  - quality, [289](#)
  - reserved, [289](#)
- FlyCapture2::JPG2Option
  - JPG2Option, [290](#)
  - quality, [290](#)
  - reserved, [291](#)
- FlyCapture2::LUTData
  - enabled, [292](#)
  - inputBitDepth, [292](#)
  - LUTData, [292](#)
  - numBanks, [292](#)
  - numChannels, [292](#)
  - numEntries, [292](#)
  - outputBitDepth, [293](#)
  - reserved, [293](#)
  - supported, [293](#)
- FlyCapture2::MACAddress
  - MACAddress, [294](#)
  - octets, [295](#)
  - operator!=, [294](#)
  - operator==, [294](#)
- FlyCapture2::MJPGOption
  - frameRate, [295](#)
  - MJPGOption, [295](#)
  - quality, [296](#)
  - reserved, [296](#)
- FlyCapture2::NodeMap
  - \_GetDeviceName, [297](#)
  - \_GetNode, [297](#)
  - \_GetNodes, [297](#)
  - \_InvalidateNodes, [297](#)
  - \_Poll, [297](#)
  - ~NodeMap, [297](#)
  - NodeMap, [296](#)
- FlyCapture2::PGMOption
  - binaryFile, [298](#)
  - PGMOption, [298](#)
  - reserved, [299](#)
- FlyCapture2::PGRGuid
  - operator!=, [300](#)
  - operator==, [300](#)
  - PGRGuid, [299](#)
  - value, [300](#)
- FlyCapture2::PNGOption
  - compressionLevel, [301](#)
  - interlaced, [301](#)
  - PNGOption, [301](#)
  - reserved, [301](#)
- FlyCapture2::PPMOption
  - binaryFile, [302](#)
  - PPMOption, [302](#)
  - reserved, [303](#)
- FlyCapture2::Property
  - absControl, [304](#)
  - absValue, [304](#)
  - autoManualMode, [305](#)
  - onOff, [305](#)
  - onePush, [305](#)
  - present, [305](#)
  - Property, [304](#)
  - reserved, [305](#)
  - type, [305](#)
  - valueA, [306](#)
  - valueB, [306](#)
- FlyCapture2::PropertyInfo
  - absMax, [308](#)
  - absMin, [308](#)
  - absValSupported, [308](#)
  - autoSupported, [308](#)
  - manualSupported, [308](#)



- max, 308
- min, 309
- onOffSupported, 309
- onePushSupported, 309
- pUnitAbbr, 309
- pUnits, 309
- present, 309
- PropertyInfo, 307
- readOutSupported, 310
- reserved, 310
- type, 310
- FlyCapture2::StrobeControl
  - delay, 311
  - duration, 311
  - onOff, 311
  - polarity, 312
  - reserved, 312
  - source, 312
  - StrobeControl, 311
- FlyCapture2::StrobeInfo
  - maxValue, 313
  - minValue, 313
  - onOffSupported, 314
  - polaritySupported, 314
  - present, 314
  - readOutSupported, 314
  - reserved, 314
  - source, 314
  - StrobeInfo, 313
- FlyCapture2::SystemInfo
  - byteOrder, 317
  - cpuDescription, 318
  - driverList, 318
  - gpuDescription, 318
  - libraryList, 318
  - numCpuCores, 318
  - osDescription, 318
  - osType, 319
  - reserved, 319
  - screenHeight, 319
  - screenWidth, 319
  - sysMemSize, 319
- FlyCapture2::TIFFOption
  - compression, 321
  - CompressionMethod, 320
  - reserved, 321
  - TIFFOption, 321
- FlyCapture2::TimeStamp
  - cycleCount, 322
  - cycleOffset, 322
  - cycleSeconds, 322
  - microSeconds, 323
  - reserved, 323
  - seconds, 323
  - TimeStamp, 322
- FlyCapture2::TopologyNode
  - ~TopologyNode, 326
  - AddChild, 326
  - AddPortType, 326
  - AssignGuidToNode, 326, 327
  - GetChild, 327
  - GetDeviceId, 328
  - GetGuid, 328
  - GetInterfaceType, 328
  - GetNodeType, 328
  - GetNumChildren, 329
  - GetNumPorts, 329
  - GetPortType, 329
  - NodeType, 324
  - operator=, 330
  - PortType, 325
  - TopologyNode, 325, 326
- FlyCapture2::TriggerMode
  - mode, 331
  - onOff, 331
  - parameter, 331
  - polarity, 331
  - reserved, 331
  - source, 332
  - TriggerMode, 331
- FlyCapture2::TriggerModelInfo
  - modeMask, 333
  - onOffSupported, 333
  - polaritySupported, 333
  - present, 333
  - readOutSupported, 333
  - reserved, 334
  - softwareTriggerSupported, 334
  - sourceMask, 334
  - TriggerModelInfo, 333
  - valueReadable, 334
- FlyCapture2::Utilities
  - CheckDriver, 335
  - GetDriverDeviceName, 335
  - GetLibraryVersion, 336
  - GetSystemInfo, 336
  - LaunchBrowser, 337
  - LaunchCommand, 337
  - LaunchCommandAsync, 337
  - LaunchHelp, 338
- FlyCapture2Defs.h, 345
  - FULL\_32BIT\_VALUE, 352
  - NULL, 352
- FlyCapture2GUI.h, 353
- FlyCapture2Platform.h, 353
  - FLYCAPTURE2\_API, 354
  - FLYCAPTURE2\_LOCAL, 354
- FlyCapture2Video, 164
  - FlyCapture2::FlyCapture2Video, 165
- FlyCapture2Video.h, 354
- FlyCapture2VideoDefs.h, 355
- FlyCapture3ApiGuiWrapper, 168
  - FlyCap3CameraControl::FlyCapture3ApiGui↔  
Wrapper, 169
- FlyCapture3ApiGuiWrapper.h, 356
  - WRAPPER\_API, 356



- ForceAllIPAddressesAutomatically
  - FlyCapture2::BusManager, [56](#)
- ForceIPAddressToCamera
  - FlyCapture2::BusManager, [56](#)
- Format7ImageSettings, [170](#)
  - FlyCapture2::Format7ImageSettings, [171](#)
- Format7Info, [173](#)
  - FlyCapture2::Format7Info, [173](#)
- Format7PacketInfo, [176](#)
  - FlyCapture2::Format7PacketInfo, [177](#)
- frameCounter
  - FlyCapture2::EmbeddedImageInfo, [149](#)
- FrameRate
  - Enumerations, [24](#)
- frameRate
  - FlyCapture2::AVIOption, [52](#)
  - FlyCapture2::H264Option, [256](#)
  - FlyCapture2::MJPGOption, [295](#)
- GCCamera, [178](#)
  - FlyCapture2::GCCamera, [181](#)
- GCCamera.h, [357](#)
- GCCamera::GetXML
  - FlyCapture2::GCCamera, [183](#)
- GPIOPinState
  - FlyCapture2::EmbeddedImageInfo, [149](#)
- gain
  - FlyCapture2::EmbeddedImageInfo, [149](#)
- GetActiveLUTBank
  - FlyCapture2::Camera, [72](#)
  - FlyCapture2::CameraBase, [105](#)
  - FlyCapture2::GCCamera, [183](#)
  - FlyCapture2::GigECamera, [214](#)
- GetBayerTileFormat
  - FlyCapture2::Image, [263](#)
- GetBitsPerPixel
  - FlyCapture2::Image, [264](#)
- GetBlockId
  - FlyCapture2::Image, [264](#)
- GetBuildDate
  - FlyCapture2::Error, [153](#)
- GetCameraFromIPAddress
  - FlyCapture2::BusManager, [57](#)
- GetCameraFromIndex
  - FlyCapture2::BusManager, [57](#)
- GetCameraFromSerialNumber
  - FlyCapture2::BusManager, [58](#)
- GetCameraInfo
  - FlyCapture2::Camera, [72](#)
  - FlyCapture2::CameraBase, [105](#)
  - FlyCapture2::GCCamera, [183](#)
  - FlyCapture2::GigECamera, [215](#)
- GetCameraSerialNumberFromIndex
  - FlyCapture2::BusManager, [58](#)
- GetCause
  - FlyCapture2::Error, [153](#)
- GetChannelStatus
  - FlyCapture2::ImageStatistics, [282](#)
- GetChild
  - FlyCapture2::TopologyNode, [327](#)
- GetColorProcessing
  - FlyCapture2::Image, [264](#)
- GetCols
  - FlyCapture2::Image, [264](#)
- GetConfiguration
  - FlyCapture2::Camera, [73](#)
  - FlyCapture2::CameraBase, [106](#)
  - FlyCapture2::GCCamera, [184](#)
  - FlyCapture2::GigECamera, [215](#)
- GetControlNameList
  - FlyCap3CameraControl::FlyCapture3ApiGui↔  
Wrapper, [169](#)
- GetCycleTime
  - FlyCapture2::Camera, [73](#)
  - FlyCapture2::CameraBase, [106](#)
  - FlyCapture2::GCCamera, [184](#)
  - FlyCapture2::GigECamera, [216](#)
- GetData
  - FlyCapture2::Image, [265](#)
- GetDataSize
  - FlyCapture2::Image, [265](#)
- GetDefaultColorProcessing
  - FlyCapture2::Image, [265](#)
- GetDefaultOutputFormat
  - FlyCapture2::Image, [266](#)
- GetDescription
  - FlyCapture2::Error, [154](#)
- GetDeviceFromIndex
  - FlyCapture2::BusManager, [59](#)
- GetDeviceId
  - FlyCapture2::TopologyNode, [328](#)
- GetDialogNameList
  - FlyCap3CameraControl::FlyCapture3ApiGui↔  
Wrapper, [169](#)
- GetDimensions
  - FlyCapture2::Image, [266](#)
- GetDriverDeviceName
  - FlyCapture2::Utilities, [335](#)
- GetEmbeddedImageInfo
  - FlyCapture2::Camera, [73](#)
  - FlyCapture2::CameraBase, [107](#)
  - FlyCapture2::GCCamera, [185](#)
  - FlyCapture2::GigECamera, [216](#)
- GetFilename
  - FlyCapture2::Error, [154](#)
- GetFormat7Configuration
  - FlyCapture2::Camera, [74](#)
- GetFormat7Info
  - FlyCapture2::Camera, [74](#)
- GetGPIOPinDirection
  - FlyCapture2::Camera, [75](#)
  - FlyCapture2::CameraBase, [107](#)
  - FlyCapture2::GCCamera, [185](#)
  - FlyCapture2::GigECamera, [219](#)
- GetGigEConfig
  - FlyCapture2::GigECamera, [216](#)
- GetGigEImageBinningSettings

- FlyCapture2::GigECamera, [217](#)
- GetGigImageSettings
  - FlyCapture2::GigECamera, [217](#)
- GetGigImageSettingsInfo
  - FlyCapture2::GigECamera, [218](#)
- GetGigImagingMode
  - FlyCapture2::GigECamera, [218](#)
- GetGigEProperty
  - FlyCapture2::GigECamera, [218](#)
- GetGigEStreamChannelInfo
  - FlyCapture2::GigECamera, [219](#)
- GetGuid
  - FlyCapture2::TopologyNode, [328](#)
- GetHistogram
  - FlyCapture2::ImageStatistics, [283](#)
- GetInterfaceType
  - FlyCapture2::GCCamera, [186](#)
  - FlyCapture2::TopologyNode, [328](#)
- GetInterfaceTypeFromGuid
  - FlyCapture2::BusManager, [59](#)
- GetInternal
  - FlyCapture2::Internal, [287](#)
- GetLUTBankInfo
  - FlyCapture2::Camera, [76](#)
  - FlyCapture2::CameraBase, [108](#)
  - FlyCapture2::GCCamera, [186](#)
  - FlyCapture2::GigECamera, [220](#)
- GetLUTChannel
  - FlyCapture2::Camera, [76](#)
  - FlyCapture2::CameraBase, [108](#)
  - FlyCapture2::GCCamera, [186](#)
  - FlyCapture2::GigECamera, [220](#)
- GetLUTInfo
  - FlyCapture2::Camera, [77](#)
  - FlyCapture2::CameraBase, [109](#)
  - FlyCapture2::GCCamera, [187](#)
  - FlyCapture2::GigECamera, [221](#)
- GetLibraryVersion
  - FlyCapture2::Utilities, [336](#)
- GetLine
  - FlyCapture2::Error, [154](#)
- GetMean
  - FlyCapture2::ImageStatistics, [283](#)
- GetMemoryChannel
  - FlyCapture2::Camera, [77](#)
  - FlyCapture2::CameraBase, [109](#)
  - FlyCapture2::GCCamera, [187](#)
  - FlyCapture2::GigECamera, [221](#)
- GetMemoryChannelInfo
  - FlyCapture2::Camera, [78](#)
  - FlyCapture2::CameraBase, [110](#)
  - FlyCapture2::GCCamera, [188](#)
  - FlyCapture2::GigECamera, [222](#)
- GetMetadata
  - FlyCapture2::Image, [267](#)
- GetNodeMap
  - FlyCapture2::GCCamera, [188](#)
- GetNodeType
  - FlyCapture2::TopologyNode, [328](#)
- GetNumChildren
  - FlyCapture2::TopologyNode, [329](#)
- GetNumDialogs
  - FlyCap3CameraControl::FlyCapture3ApiGui↔  
Wrapper, [169](#)
- GetNumOfCameras
  - FlyCapture2::BusManager, [60](#)
- GetNumOfControls
  - FlyCap3CameraControl::FlyCapture3ApiGui↔  
Wrapper, [170](#)
- GetNumOfDevices
  - FlyCapture2::BusManager, [60](#)
- GetNumPixelValues
  - FlyCapture2::ImageStatistics, [283](#)
- GetNumPorts
  - FlyCapture2::TopologyNode, [329](#)
- GetNumStreamChannels
  - FlyCapture2::GigECamera, [222](#)
- GetPixelFormat
  - FlyCapture2::Image, [267](#)
- GetPixelValueRange
  - FlyCapture2::ImageStatistics, [284](#)
- GetPortType
  - FlyCapture2::TopologyNode, [329](#)
- GetProperty
  - FlyCapture2::Camera, [78](#)
  - FlyCapture2::CameraBase, [110](#)
  - FlyCapture2::GCCamera, [189](#)
  - FlyCapture2::GigECamera, [223](#)
- GetPropertyInfo
  - FlyCapture2::Camera, [79](#)
  - FlyCapture2::CameraBase, [111](#)
  - FlyCapture2::GCCamera, [189](#)
  - FlyCapture2::GigECamera, [223](#)
- GetRange
  - FlyCapture2::ImageStatistics, [284](#)
- GetReceivedDataSize
  - FlyCapture2::Image, [267](#)
- GetRegisterString
  - FlyCapture2::Camera, [79](#)
  - FlyCapture2::CameraBase, [111](#)
  - FlyCapture2::GCCamera, [190](#)
  - FlyCapture2::GigECamera, [224](#)
- GetRows
  - FlyCapture2::Image, [267](#)
- GetStatistics
  - FlyCapture2::ImageStatistics, [285](#)
- GetStats
  - FlyCapture2::Camera, [79](#)
  - FlyCapture2::CameraBase, [112](#)
  - FlyCapture2::GCCamera, [190](#)
  - FlyCapture2::GigECamera, [224](#)
- GetStride
  - FlyCapture2::Image, [268](#)
- GetStroke
  - FlyCapture2::Camera, [80](#)
  - FlyCapture2::CameraBase, [112](#)

- FlyCapture2::GCCamera, [190](#)
- FlyCapture2::GigECamera, [224](#)
- GetStrobeInfo
  - FlyCapture2::Camera, [80](#)
  - FlyCapture2::CameraBase, [112](#)
  - FlyCapture2::GCCamera, [191](#)
  - FlyCapture2::GigECamera, [225](#)
- GetSyncStatus
  - MultiSyncLibrary::SyncManager, [316](#)
- GetSystemInfo
  - FlyCapture2::Utilities, [336](#)
- GetTimeSinceSynced
  - MultiSyncLibrary::SyncManager, [316](#)
- GetTimeStamp
  - FlyCapture2::Image, [268](#)
- GetTopology
  - FlyCapture2::BusManager, [60](#)
- GetTriggerDelay
  - FlyCapture2::Camera, [81](#)
  - FlyCapture2::CameraBase, [113](#)
  - FlyCapture2::GCCamera, [191](#)
  - FlyCapture2::GigECamera, [225](#)
- GetTriggerDelayInfo
  - FlyCapture2::Camera, [81](#)
  - FlyCapture2::CameraBase, [113](#)
  - FlyCapture2::GCCamera, [192](#)
  - FlyCapture2::GigECamera, [226](#)
- GetTriggerMode
  - FlyCapture2::Camera, [82](#)
  - FlyCapture2::CameraBase, [114](#)
  - FlyCapture2::GCCamera, [192](#)
  - FlyCapture2::GigECamera, [226](#)
- GetTriggerModelInfo
  - FlyCapture2::Camera, [82](#)
  - FlyCapture2::CameraBase, [114](#)
  - FlyCapture2::GCCamera, [193](#)
  - FlyCapture2::GigECamera, [227](#)
- GetType
  - FlyCapture2::Error, [154](#)
- GetUsbLinkInfo
  - FlyCapture2::BusManager, [61](#)
- GetUsbPortStatus
  - FlyCapture2::BusManager, [61](#)
- GetVideoModeAndFrameRate
  - FlyCapture2::Camera, [83](#)
- GetVideoModeAndFrameRateInfo
  - FlyCapture2::Camera, [83](#)
- GigE specific enumerations, [31](#)
  - GigEPropertyType, [31](#)
- GigE specific structures, [34](#)
- GigECamera, [207](#)
  - FlyCapture2::GigECamera, [212](#)
- GigECamera.h, [358](#)
- GigEConfig, [245](#)
  - FlyCapture2::GigEConfig, [246](#)
- GigEImageSettings, [247](#)
  - FlyCapture2::GigEImageSettings, [247](#)
- GigEImageSettingsInfo, [249](#)
  - FlyCapture2::GigEImageSettingsInfo, [249](#)
- gigEMajorVersion
  - FlyCapture2::CameraInfo, [136](#)
- gigEMinorVersion
  - FlyCapture2::CameraInfo, [136](#)
- GigEProperty, [251](#)
- GigEPropertyType
  - GigE specific enumerations, [31](#)
- GigEStreamChannel, [253](#)
  - FlyCapture2::GigEStreamChannel, [254](#)
- Global constants, [15](#)
  - sk\_maxNumPorts, [15](#)
  - sk\_maxStringLength, [15](#)
- gpuDescription
  - FlyCapture2::SystemInfo, [318](#)
- GrabMode
  - Enumerations, [25](#)
- grabMode
  - FlyCapture2::FC2Config, [161](#)
- GrabTimeout
  - Enumerations, [25](#)
- grabTimeout
  - FlyCapture2::FC2Config, [161](#)
- H264Option, [255](#)
  - FlyCapture2::H264Option, [256](#)
- height
  - FlyCapture2::Format7ImageSettings, [171](#)
  - FlyCapture2::GigEImageSettings, [248](#)
  - FlyCapture2::H264Option, [256](#)
- Hide
  - FlyCapture2::CameraControlDlg, [130](#)
- highPerformanceRetrieveBuffer
  - FlyCapture2::FC2Config, [161](#)
- hostPort
  - FlyCapture2::GigEStreamChannel, [254](#)
- IIDC specific structures, [35](#)
- IPAddress, [287](#)
  - FlyCapture2::IPAddress, [287](#)
- iidcVer
  - FlyCapture2::CameraInfo, [136](#)
- Image, [257](#)
  - FlyCapture2::Image, [259–261](#)
- Image saving structures., [36](#)
  - CameraEventCallback, [36](#)
- Image.h, [359](#)
- imageCorrupt
  - FlyCapture2::CameraStats, [142](#)
- imageDriverDropped
  - FlyCapture2::CameraStats, [142](#)
- imageDropped
  - FlyCapture2::CameraStats, [142](#)
- ImageEventCallback
  - FlyCapture2, [47](#)
- ImageFileFormat
  - Enumerations, [26](#)
- imageHStepSize
  - FlyCapture2::Format7Info, [174](#)

- FlyCapture2::GigEImageSettingsInfo, 250
- ImageMetadata, 276
  - FlyCapture2::ImageMetadata, 276
- ImageStatistics, 279
  - FlyCapture2::ImageStatistics, 280, 281
- ImageStatistics.h, 360
- ImageStatsCalculator
  - FlyCapture2::ImageStatistics, 286
- imageVStepSize
  - FlyCapture2::Format7Info, 174
  - FlyCapture2::GigEImageSettingsInfo, 250
- imageXmitFailed
  - FlyCapture2::CameraStats, 143
- indexedColor\_8bit
  - FlyCapture2::BMPOption, 53
- inputBitDepth
  - FlyCapture2::LUTData, 292
- interPacketDelay
  - FlyCapture2::GigEStreamChannel, 254
- InterfaceType
  - Enumerations, 26
- interfaceType
  - FlyCapture2::CameraInfo, 136
- interlaced
  - FlyCapture2::PNGOption, 301
- Internal, 286
- Internal.h, 361
- InternalError
  - FlyCapture2::Error, 156
- ipAddress
  - FlyCapture2::CameraInfo, 136
- IsCameraControllable
  - FlyCapture2::BusManager, 62
- isColorCamera
  - FlyCapture2::CameraInfo, 136
- IsConnected
  - FlyCapture2::Camera, 84
  - FlyCapture2::CameraBase, 115
  - FlyCapture2::GCCamera, 193
  - FlyCapture2::GigECamera, 227
- isReadable
  - FlyCapture2::GigEProperty, 252
- IsTimingBusConnected
  - MultiSyncLibrary::SyncManager, 316
- IsVisible
  - FlyCapture2::CameraControlDlg, 130
- isWritable
  - FlyCapture2::GigEProperty, 252
- Iso
  - FlyCapture2::Image, 275
- isochBusSpeed
  - FlyCapture2::FC2Config, 161
- JPEGOption, 288
  - FlyCapture2::JPEGOption, 289
- JPG2Option, 290
  - FlyCapture2::JPG2Option, 290
- LUTData, 291
  - FlyCapture2::LUTData, 292
- LaunchBrowser
  - FlyCapture2::Utilities, 337
- LaunchCommand
  - FlyCapture2::Utilities, 337
- LaunchCommandAsync
  - FlyCapture2::Utilities, 337
- LaunchHelp
  - FlyCapture2::Utilities, 338
- libraryList
  - FlyCapture2::SystemInfo, 318
- Licensing.dox, 361
- m\_busMgr
  - FlyCapture2::GCCamera, 207
- m\_pCameraData
  - FlyCapture2::CameraBase, 128
- MACAddress, 293
  - FlyCapture2::MACAddress, 294
- MJPGOption, 295
  - FlyCapture2::MJPGOption, 295
- MULTISYNCLIBRARY\_API
  - MultiSyncLibraryPlatform.h, 363
- MULTISYNCLIBRARY\_LOCAL
  - MultiSyncLibraryPlatform.h, 363
- macAddress
  - FlyCapture2::CameraInfo, 137
- major
  - FlyCapture2::FC2Version, 164
- manualSupported
  - FlyCapture2::PropertyInfo, 308
- max
  - FlyCapture2::GigEProperty, 252
  - FlyCapture2::PropertyInfo, 308
- maxBytesPerPacket
  - FlyCapture2::Format7PacketInfo, 177
- maxHeight
  - FlyCapture2::Format7Info, 174
  - FlyCapture2::GigEImageSettingsInfo, 250
- maxPacketSize
  - FlyCapture2::Format7Info, 174
- maxValue
  - FlyCapture2::StrobeInfo, 313
- maxWidth
  - FlyCapture2::Format7Info, 174
  - FlyCapture2::GigEImageSettingsInfo, 250
- maximumBusSpeed
  - FlyCapture2::CameraInfo, 137
- microSeconds
  - FlyCapture2::TimeStamp, 323
- min
  - FlyCapture2::GigEProperty, 252
  - FlyCapture2::PropertyInfo, 309
- minNumImageNotifications
  - FlyCapture2::FC2Config, 162
- minPacketSize
  - FlyCapture2::Format7Info, 174
- minValue
  - FlyCapture2::StrobeInfo, 313

- minor
  - FlyCapture2::FC2Version, [164](#)
- Mode
  - Enumerations, [26](#)
- mode
  - FlyCapture2::Format7ImageSettings, [171](#)
  - FlyCapture2::Format7Info, [175](#)
  - FlyCapture2::TriggerMode, [331](#)
- modeMask
  - FlyCapture2::TriggerModelInfo, [333](#)
- modelName
  - FlyCapture2::CameraInfo, [137](#)
- MultiSyncLibrary, [48](#)
  - PGRSyncError, [49](#)
  - PGRSyncMessage, [49](#)
- MultiSyncLibrary.h, [361](#)
- MultiSyncLibrary::SyncManager
  - ~SyncManager, [315](#)
  - DisableCrossPCSynchronization, [315](#)
  - EnableCrossPCSynchronization, [316](#)
  - GetSyncStatus, [316](#)
  - GetTimeSinceSynced, [316](#)
  - IsTimingBusConnected, [316](#)
  - QueryCrossPCSynchronizationSetting, [316](#)
  - RescanMasterTimingBus, [316](#)
  - Start, [316](#)
  - Stop, [316](#)
  - SyncManager, [315](#)
- MultiSyncLibraryDefs.h, [362](#)
- MultiSyncLibraryPlatform.h, [363](#)
  - MULTISYNCLIBRARY\_API, [363](#)
  - MULTISYNCLIBRARY\_LOCAL, [363](#)
- NULL
  - FlyCapture2Defs.h, [352](#)
- networkInterfaceIndex
  - FlyCapture2::GigEStreamChannel, [254](#)
- NodeMap, [296](#)
  - FlyCapture2::NodeMap, [296](#)
- NodeMap.h, [364](#)
- nodeNumber
  - FlyCapture2::CameraInfo, [137](#)
- NodeType
  - FlyCapture2::TopologyNode, [324](#)
- nodeVendorId
  - FlyCapture2::ConfigROM, [146](#)
- numBanks
  - FlyCapture2::LUTData, [292](#)
- numBuffers
  - FlyCapture2::FC2Config, [162](#)
- numChannels
  - FlyCapture2::LUTData, [292](#)
- numCpuCores
  - FlyCapture2::SystemInfo, [318](#)
- numCurrents
  - FlyCapture2::CameraStats, [143](#)
- numEntries
  - FlyCapture2::LUTData, [292](#)
- numImageNotifications
  - FlyCapture2::FC2Config, [162](#)
- numResendPacketsReceived
  - FlyCapture2::CameraStats, [143](#)
- numResendPacketsRequested
  - FlyCapture2::CameraStats, [143](#)
- numVoltages
  - FlyCapture2::CameraStats, [143](#)
- OSType
  - FlyCapture2, [48](#)
- octets
  - FlyCapture2::IPAddress, [288](#)
  - FlyCapture2::MACAddress, [295](#)
- offsetHStepSize
  - FlyCapture2::Format7Info, [175](#)
  - FlyCapture2::GigEImageSettingsInfo, [250](#)
- offsetVStepSize
  - FlyCapture2::Format7Info, [175](#)
  - FlyCapture2::GigEImageSettingsInfo, [250](#)
- offsetX
  - FlyCapture2::Format7ImageSettings, [172](#)
  - FlyCapture2::GigEImageSettings, [248](#)
- offsetY
  - FlyCapture2::Format7ImageSettings, [172](#)
  - FlyCapture2::GigEImageSettings, [248](#)
- onOff
  - FlyCapture2::EmbeddedImageInfoProperty, [151](#)
  - FlyCapture2::Property, [305](#)
  - FlyCapture2::StrobeControl, [311](#)
  - FlyCapture2::TriggerMode, [331](#)
- onOffSupported
  - FlyCapture2::PropertyInfo, [309](#)
  - FlyCapture2::StrobeInfo, [314](#)
  - FlyCapture2::TriggerModelInfo, [333](#)
- onePush
  - FlyCapture2::Property, [305](#)
- onePushSupported
  - FlyCapture2::PropertyInfo, [309](#)
- Open
  - FlyCapture2::FlyCapture2Video, [166](#), [167](#)
- operator!=
  - FlyCapture2::Error, [155](#)
  - FlyCapture2::IPAddress, [288](#)
  - FlyCapture2::MACAddress, [294](#)
  - FlyCapture2::PGRGuid, [300](#)
- operator()
  - FlyCapture2::Image, [268](#)
- operator=
  - FlyCapture2::Error, [155](#)
  - FlyCapture2::Image, [269](#)
  - FlyCapture2::ImageStatistics, [285](#)
  - FlyCapture2::TopologyNode, [330](#)
- operator==
  - FlyCapture2::Error, [155](#)
  - FlyCapture2::IPAddress, [288](#)
  - FlyCapture2::MACAddress, [294](#)
  - FlyCapture2::PGRGuid, [300](#)
- operator[]
  - FlyCapture2::Image, [269](#)

- osDescription
  - FlyCapture2::SystemInfo, [318](#)
- osType
  - FlyCapture2::SystemInfo, [319](#)
- outputBitDepth
  - FlyCapture2::LUTData, [293](#)
- PCleBusSpeed
  - Enumerations, [27](#)
- PGMOption, [298](#)
  - FlyCapture2::PGMOption, [298](#)
- PGRGuid, [299](#)
  - FlyCapture2::PGRGuid, [299](#)
- PGRSyncError
  - MultiSyncLibrary, [49](#)
- PGRSyncMessage
  - MultiSyncLibrary, [49](#)
- PNGOption, [300](#)
  - FlyCapture2::PNGOption, [301](#)
- PPMOption, [302](#)
  - FlyCapture2::PPMOption, [302](#)
- pUnitAbbr
  - FlyCapture2::PropertyInfo, [309](#)
- pUnits
  - FlyCapture2::PropertyInfo, [309](#)
- packetSize
  - FlyCapture2::Format7Info, [175](#)
  - FlyCapture2::GigEStreamChannel, [255](#)
- parameter
  - FlyCapture2::TriggerMode, [331](#)
- pcieBusSpeed
  - FlyCapture2::CameraInfo, [137](#)
- percentage
  - FlyCapture2::Format7Info, [175](#)
- PixelFormat
  - Enumerations, [28](#)
- pixelFormat
  - FlyCapture2::Format7ImageSettings, [172](#)
  - FlyCapture2::GigEImageSettings, [248](#)
- pixelFormatBitField
  - FlyCapture2::Format7Info, [175](#)
  - FlyCapture2::GigEImageSettingsInfo, [251](#)
- polarity
  - FlyCapture2::StrobeControl, [312](#)
  - FlyCapture2::TriggerMode, [331](#)
- polaritySupported
  - FlyCapture2::StrobeInfo, [314](#)
  - FlyCapture2::TriggerModelInfo, [333](#)
- portErrors
  - FlyCapture2::CameraStats, [143](#)
- PortType
  - FlyCapture2::TopologyNode, [325](#)
- present
  - FlyCapture2::Property, [305](#)
  - FlyCapture2::PropertyInfo, [309](#)
  - FlyCapture2::StrobeInfo, [314](#)
  - FlyCapture2::TriggerModelInfo, [333](#)
- PrintErrorTrace
  - FlyCapture2::Error, [156](#)
- progressive
  - FlyCapture2::JPEGOption, [289](#)
- propType
  - FlyCapture2::GigEProperty, [252](#)
- Property, [303](#)
  - FlyCapture2::Property, [304](#)
- PropertyInfo, [306](#)
  - FlyCapture2::PropertyInfo, [307](#)
- PropertyType
  - Enumerations, [28](#)
- pszKeyword
  - FlyCapture2::ConfigROM, [146](#)
- quality
  - FlyCapture2::JPEGOption, [289](#)
  - FlyCapture2::JPG2Option, [290](#)
  - FlyCapture2::MJPGOption, [296](#)
- QueryCrossPCSynchronizationSetting
  - MultiSyncLibrary::SyncManager, [316](#)
- QueryGigEImagingMode
  - FlyCapture2::GigECamera, [228](#)
- ROIPosition
  - FlyCapture2::EmbeddedImageInfo, [149](#)
- ReadGVCPMemory
  - FlyCapture2::GCCamera, [193](#)
  - FlyCapture2::GigECamera, [228](#)
- ReadGVCPRegister
  - FlyCapture2::GCCamera, [194](#)
  - FlyCapture2::GigECamera, [229](#)
- ReadGVCPRegisterBlock
  - FlyCapture2::GCCamera, [194](#)
  - FlyCapture2::GigECamera, [229](#)
- readOutSupported
  - FlyCapture2::PropertyInfo, [310](#)
  - FlyCapture2::StrobeInfo, [314](#)
  - FlyCapture2::TriggerModelInfo, [333](#)
- ReadPhyRegister
  - FlyCapture2::BusManager, [63](#)
- ReadRegister
  - FlyCapture2::Camera, [84](#)
  - FlyCapture2::CameraBase, [115](#)
  - FlyCapture2::GCCamera, [194](#)
  - FlyCapture2::GigECamera, [229](#)
- ReadRegisterBlock
  - FlyCapture2::Camera, [85](#)
  - FlyCapture2::CameraBase, [116](#)
  - FlyCapture2::GCCamera, [194](#)
  - FlyCapture2::GigECamera, [230](#)
- recommendedBytesPerPacket
  - FlyCapture2::Format7PacketInfo, [177](#)
- regReadFailed
  - FlyCapture2::CameraStats, [143](#)
- regWriteFailed
  - FlyCapture2::CameraStats, [144](#)
- RegisterAllEvents
  - FlyCapture2::Camera, [85](#)
  - FlyCapture2::CameraBase, [116](#)
  - FlyCapture2::GigECamera, [230](#)



- RegisterCallback
  - FlyCapture2::BusManager, [63](#)
- RegisterEvent
  - FlyCapture2::Camera, [85](#)
  - FlyCapture2::CameraBase, [116](#)
  - FlyCapture2::GigECamera, [231](#)
- registerTimeout
  - FlyCapture2::FC2Config, [162](#)
  - FlyCapture2::GigEConfig, [246](#)
- registerTimeoutRetries
  - FlyCapture2::FC2Config, [163](#)
  - FlyCapture2::GigEConfig, [246](#)
- ReleaseBuffer
  - FlyCapture2::Image, [269](#)
- RescanBus
  - FlyCapture2::BusManager, [64](#)
- RescanMasterTimingBus
  - MultiSyncLibrary::SyncManager, [316](#)
- reserved
  - FlyCapture2::AVIOption, [52](#)
  - FlyCapture2::BMPOption, [53](#)
  - FlyCapture2::CameraInfo, [137](#)
  - FlyCapture2::CameraStats, [144](#)
  - FlyCapture2::ConfigROM, [146](#)
  - FlyCapture2::FC2Config, [163](#)
  - FlyCapture2::Format7ImageSettings, [172](#)
  - FlyCapture2::Format7Info, [176](#)
  - FlyCapture2::Format7PacketInfo, [177](#)
  - FlyCapture2::GigEImageSettings, [248](#)
  - FlyCapture2::GigEImageSettingsInfo, [251](#)
  - FlyCapture2::H264Option, [256](#)
  - FlyCapture2::ImageMetadata, [278](#)
  - FlyCapture2::JPEGOption, [289](#)
  - FlyCapture2::JPG2Option, [291](#)
  - FlyCapture2::LUTData, [293](#)
  - FlyCapture2::MJPGOption, [296](#)
  - FlyCapture2::PGMOption, [299](#)
  - FlyCapture2::PNGOption, [301](#)
  - FlyCapture2::PPMOption, [303](#)
  - FlyCapture2::Property, [305](#)
  - FlyCapture2::PropertyInfo, [310](#)
  - FlyCapture2::StrobeControl, [312](#)
  - FlyCapture2::StrobeInfo, [314](#)
  - FlyCapture2::SystemInfo, [319](#)
  - FlyCapture2::TIFFOption, [321](#)
  - FlyCapture2::TimeStamp, [323](#)
  - FlyCapture2::TriggerMode, [331](#)
  - FlyCapture2::TriggerModelInfo, [334](#)
- ResetStats
  - FlyCapture2::Camera, [86](#)
  - FlyCapture2::CameraBase, [117](#)
  - FlyCapture2::GCCamera, [195](#)
  - FlyCapture2::GigECamera, [231](#)
- RestoreFromMemoryChannel
  - FlyCapture2::Camera, [86](#)
  - FlyCapture2::CameraBase, [117](#)
  - FlyCapture2::GCCamera, [195](#)
  - FlyCapture2::GigECamera, [231](#)
- RetrieveBuffer
  - FlyCapture2::Camera, [86](#)
  - FlyCapture2::CameraBase, [117](#)
  - FlyCapture2::GCCamera, [196](#)
  - FlyCapture2::GigECamera, [231](#)
- Save
  - FlyCapture2::Image, [269–272](#)
- SaveToMemoryChannel
  - FlyCapture2::Camera, [87](#)
  - FlyCapture2::CameraBase, [118](#)
  - FlyCapture2::GCCamera, [196](#)
  - FlyCapture2::GigECamera, [232](#)
- screenHeight
  - FlyCapture2::SystemInfo, [319](#)
- screenWidth
  - FlyCapture2::SystemInfo, [319](#)
- seconds
  - FlyCapture2::TimeStamp, [323](#)
- sensorInfo
  - FlyCapture2::CameraInfo, [138](#)
- sensorResolution
  - FlyCapture2::CameraInfo, [138](#)
- serialNumber
  - FlyCapture2::CameraInfo, [138](#)
- SetActiveLUTBank
  - FlyCapture2::Camera, [87](#)
  - FlyCapture2::CameraBase, [118](#)
  - FlyCapture2::GCCamera, [197](#)
  - FlyCapture2::GigECamera, [233](#)
- SetBlockId
  - FlyCapture2::Image, [272](#)
- SetCallback
  - FlyCapture2::Camera, [88](#)
  - FlyCapture2::CameraBase, [119](#)
  - FlyCapture2::GCCamera, [197](#)
  - FlyCapture2::GigECamera, [233](#)
- SetCamera
  - FlyCapture2::GCCamera, [198](#)
- SetChannelStatus
  - FlyCapture2::ImageStatistics, [286](#)
- SetColorProcessing
  - FlyCapture2::Image, [273](#)
- SetConfiguration
  - FlyCapture2::Camera, [88](#)
  - FlyCapture2::CameraBase, [119](#)
  - FlyCapture2::GCCamera, [198](#)
  - FlyCapture2::GigECamera, [233](#)
- SetData
  - FlyCapture2::Image, [273](#)
- SetDefaultColorProcessing
  - FlyCapture2::Image, [274](#)
- SetDefaultOutputFormat
  - FlyCapture2::Image, [274](#)
- SetDimensions
  - FlyCapture2::Image, [275](#)
- SetEmbeddedImageInfo
  - FlyCapture2::Camera, [89](#)
  - FlyCapture2::CameraBase, [120](#)

- FlyCapture2::GCCamera, [199](#)
- FlyCapture2::GigECamera, [234](#)
- SetFormat7Configuration
  - FlyCapture2::Camera, [89](#), [90](#)
- SetGPIOPinDirection
  - FlyCapture2::Camera, [90](#)
  - FlyCapture2::CameraBase, [120](#)
  - FlyCapture2::GCCamera, [199](#)
  - FlyCapture2::GigECamera, [237](#)
- SetGigEConfig
  - FlyCapture2::GigECamera, [234](#)
- SetGigEImageBinningSettings
  - FlyCapture2::GigECamera, [235](#)
- SetGigEImageSettings
  - FlyCapture2::GigECamera, [235](#)
- SetGigEImagingMode
  - FlyCapture2::GigECamera, [235](#)
- SetGigEProperty
  - FlyCapture2::GigECamera, [236](#)
- SetGigEStreamChannelInfo
  - FlyCapture2::GigECamera, [236](#)
- SetLUTChannel
  - FlyCapture2::Camera, [91](#)
  - FlyCapture2::CameraBase, [121](#)
  - FlyCapture2::GCCamera, [200](#)
  - FlyCapture2::GigECamera, [237](#)
- SetMaximumFileSize
  - FlyCapture2::FlyCapture2Video, [168](#)
- SetProperty
  - FlyCapture2::Camera, [91](#)
  - FlyCapture2::CameraBase, [121](#)
  - FlyCapture2::GCCamera, [200](#)
  - FlyCapture2::GigECamera, [238](#)
- SetStrobe
  - FlyCapture2::Camera, [92](#)
  - FlyCapture2::CameraBase, [122](#)
  - FlyCapture2::GCCamera, [201](#)
  - FlyCapture2::GigECamera, [238](#)
- SetTitle
  - FlyCapture2::CameraControlDlg, [130](#)
  - FlyCapture2::CameraSelectionDlg, [140](#)
- SetTriggerDelay
  - FlyCapture2::Camera, [93](#)
  - FlyCapture2::CameraBase, [123](#)
  - FlyCapture2::GCCamera, [201](#)
  - FlyCapture2::GigECamera, [239](#)
- SetTriggerMode
  - FlyCapture2::Camera, [93](#)
  - FlyCapture2::CameraBase, [123](#)
  - FlyCapture2::GCCamera, [202](#)
  - FlyCapture2::GigECamera, [240](#)
- SetUserBuffers
  - FlyCapture2::Camera, [94](#)
  - FlyCapture2::CameraBase, [124](#)
  - FlyCapture2::GCCamera, [202](#)
  - FlyCapture2::GigECamera, [240](#)
- SetVideoModeAndFrameRate
  - FlyCapture2::Camera, [94](#)
- Show
  - FlyCapture2::CameraControlDlg, [131](#)
- ShowCameraSelectionDialog
  - FlyCap3CameraControl::FlyCapture3ApiGui↔  
Wrapper, [170](#)
- ShowDialogByIndex
  - FlyCap3CameraControl::FlyCapture3ApiGui↔  
Wrapper, [170](#)
- ShowDialogByName
  - FlyCap3CameraControl::FlyCapture3ApiGui↔  
Wrapper, [170](#)
- ShowModal
  - FlyCapture2::CameraControlDlg, [131](#)
  - FlyCapture2::CameraSelectionDlg, [140](#)
- ShowPropertyGridDialog
  - FlyCap3CameraControl::FlyCapture3ApiGui↔  
Wrapper, [170](#)
- shutter
  - FlyCapture2::EmbeddedImageInfo, [149](#)
- sk\_maxNumPorts
  - Global constants, [15](#)
- sk\_maxStringLength
  - Global constants, [15](#)
- softwareTriggerSupported
  - FlyCapture2::TriggerModelInfo, [334](#)
- source
  - FlyCapture2::StrobeControl, [312](#)
  - FlyCapture2::StrobeInfo, [314](#)
  - FlyCapture2::TriggerMode, [332](#)
- sourceMask
  - FlyCapture2::TriggerModelInfo, [334](#)
- sourcePort
  - FlyCapture2::GigEStreamChannel, [255](#)
- Start
  - MultiSyncLibrary::SyncManager, [316](#)
- StartCapture
  - FlyCapture2::Camera, [95](#)
  - FlyCapture2::CameraBase, [124](#)
  - FlyCapture2::GCCamera, [203](#)
  - FlyCapture2::GigECamera, [241](#)
- StartSyncCapture
  - FlyCapture2::Camera, [96](#)
  - FlyCapture2::CameraBase, [125](#)
  - FlyCapture2::GCCamera, [204](#)
  - FlyCapture2::GigECamera, [242](#)
- StatisticsChannel
  - FlyCapture2::ImageStatistics, [280](#)
- Stop
  - MultiSyncLibrary::SyncManager, [316](#)
- StopCapture
  - FlyCapture2::Camera, [96](#)
  - FlyCapture2::CameraBase, [126](#)
  - FlyCapture2::GCCamera, [204](#)
  - FlyCapture2::GigECamera, [242](#)
- StrobeControl, [310](#)
  - FlyCapture2::StrobeControl, [311](#)
- StrobeInfo, [312](#)
  - FlyCapture2::StrobeInfo, [313](#)



- strobePattern
  - FlyCapture2::EmbeddedImageInfo, 150
- Structures, 32
  - TriggerDelay, 33
  - TriggerDelayInfo, 33
- subnetMask
  - FlyCapture2::CameraInfo, 138
- supported
  - FlyCapture2::LUTData, 293
- SyncManager, 315
  - MultiSyncLibrary::SyncManager, 315
- sysMemSize
  - FlyCapture2::SystemInfo, 319
- SystemInfo, 317
- TIFFOption, 320
  - FlyCapture2::TIFFOption, 321
- temperature
  - FlyCapture2::CameraStats, 144
- TestGainNode
  - FlyCapture2::GCCamera, 204
- timeSinceBusReset
  - FlyCapture2::CameraStats, 144
- timeSinceInitialization
  - FlyCapture2::CameraStats, 144
- TimeStamp, 321
  - FlyCapture2::TimeStamp, 322
- timeStamp
  - FlyCapture2::CameraStats, 144
- timestamp
  - FlyCapture2::EmbeddedImageInfo, 150
- TopologyNode, 323
  - FlyCapture2::TopologyNode, 325, 326
- TopologyNode.h, 364
- TriggerDelay
  - Structures, 33
- TriggerDelayInfo
  - Structures, 33
- TriggerMode, 330
  - FlyCapture2::TriggerMode, 331
- TriggerModelInfo, 332
  - FlyCapture2::TriggerModelInfo, 333
- type
  - FlyCapture2::FC2Version, 164
  - FlyCapture2::Property, 305
  - FlyCapture2::PropertyInfo, 310
- unitBytesPerPacket
  - FlyCapture2::Format7PacketInfo, 177
- unitSWVer
  - FlyCapture2::ConfigROM, 147
- unitSpecId
  - FlyCapture2::ConfigROM, 146
- unitSubSWVer
  - FlyCapture2::ConfigROM, 147
- UnregisterCallback
  - FlyCapture2::BusManager, 64
- userDefinedName
  - FlyCapture2::CameraInfo, 138
- Utilities, 334
- Utilities.h, 365
- ValidateFormat7Settings
  - FlyCapture2::Camera, 96
- value
  - FlyCapture2::GigEProperty, 252
  - FlyCapture2::PGRGuid, 300
- valueReadable
  - FlyCapture2::TriggerModelInfo, 334
- valueA
  - FlyCapture2::Property, 306
- valueB
  - FlyCapture2::Property, 306
- vendorName
  - FlyCapture2::CameraInfo, 138
- vendorPixelFormatBitField
  - FlyCapture2::Format7Info, 176
  - FlyCapture2::GigEImageSettingsInfo, 251
- vendorUniqueInfo\_0
  - FlyCapture2::ConfigROM, 147
- vendorUniqueInfo\_1
  - FlyCapture2::ConfigROM, 147
- vendorUniqueInfo\_2
  - FlyCapture2::ConfigROM, 147
- vendorUniqueInfo\_3
  - FlyCapture2::ConfigROM, 147
- Video saving structures., 37
- VideoMode
  - Enumerations, 29
- WRAPPER\_API
  - FlyCapture3ApiGuiWrapper.h, 356
- WaitForBufferEvent
  - FlyCapture2::Camera, 97
  - FlyCapture2::CameraBase, 126
  - FlyCapture2::GCCamera, 205
  - FlyCapture2::GigECamera, 242
- whiteBalance
  - FlyCapture2::EmbeddedImageInfo, 150
- width
  - FlyCapture2::Format7ImageSettings, 172
  - FlyCapture2::GigEImageSettings, 248
  - FlyCapture2::H264Option, 256
- WriteGVCPMemory
  - FlyCapture2::GCCamera, 205
  - FlyCapture2::GigECamera, 243
- WriteGVCPRegister
  - FlyCapture2::GCCamera, 205
  - FlyCapture2::GigECamera, 243
- WriteGVCPRegisterBlock
  - FlyCapture2::GCCamera, 205
  - FlyCapture2::GigECamera, 244
- WritePhyRegister
  - FlyCapture2::BusManager, 65
- WriteRegister
  - FlyCapture2::Camera, 97
  - FlyCapture2::CameraBase, 127
  - FlyCapture2::GCCamera, 206

- FlyCapture2::GigECamera, [244](#)
- WriteRegisterBlock
  - FlyCapture2::Camera, [98](#)
  - FlyCapture2::CameraBase, [127](#)
  - FlyCapture2::GCCamera, [206](#)
  - FlyCapture2::GigECamera, [245](#)
- xmlURL1
  - FlyCapture2::CameraInfo, [139](#)
- xmlURL2
  - FlyCapture2::CameraInfo, [139](#)