

FlyCapture2

2.11.3.0

Generated by Doxygen 1.7.5

Wed Jul 5 2017 12:30:18



# 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>Class 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	Variable Documentation . . . . .	15
7.1.1.1	sk_maxNumPorts . . . . .	15
7.1.1.2	sk_maxStringLength . . . . .	15
7.2	Enumerations . . . . .	16
7.2.1	Enumeration Type Documentation . . . . .	18
7.2.1.1	BandwidthAllocation . . . . .	18
7.2.1.2	BayerTileFormat . . . . .	19
7.2.1.3	BusCallbackType . . . . .	19
7.2.1.4	BusSpeed . . . . .	19

7.2.1.5	ColorProcessingAlgorithm	20
7.2.1.6	DriverType	20
7.2.1.7	ErrorType	21
7.2.1.8	FrameRate	22
7.2.1.9	GrabMode	23
7.2.1.10	GrabTimeout	23
7.2.1.11	ImageFileFormat	24
7.2.1.12	InterfaceType	24
7.2.1.13	Mode	24
7.2.1.14	PCleBusSpeed	25
7.2.1.15	PixelFormat	26
7.2.1.16	PropertyType	26
7.2.1.17	VideoMode	27
7.3	GigE specific enumerations	29
7.3.1	Detailed Description	29
7.3.2	Enumeration Type Documentation	29
7.3.2.1	GigEPropertyType	29
7.4	Structures	30
7.4.1	Typedef Documentation	31
7.4.1.1	TriggerDelay	31
7.4.1.2	TriggerDelayInfo	32
7.5	GigE specific structures	33
7.5.1	Detailed Description	33
7.6	IIDC specific structures	34
7.6.1	Detailed Description	34
7.7	Image saving structures.	35
7.7.1	Detailed Description	36
7.7.2	Typedef Documentation	36
7.7.2.1	CameraEventCallback	36
<b>8</b>	<b>Namespace Documentation</b>	<b>37</b>
8.1	FlyCap3CameraControl Namespace Reference	37
8.2	FlyCapture2 Namespace Reference	37
8.2.1	Typedef Documentation	43

8.2.1.1	<a href="#">AsyncCommandCallback</a>	43
8.2.1.2	<a href="#">BusEventCallback</a>	43
8.2.1.3	<a href="#">CallbackHandle</a>	43
8.2.1.4	<a href="#">ImageEventCallback</a>	44
8.2.2	<a href="#">Enumeration Type Documentation</a>	44
8.2.2.1	<a href="#">ByteOrder</a>	44
8.2.2.2	<a href="#">OSType</a>	44
8.3	<a href="#">MultiSyncLibrary Namespace Reference</a>	44
8.3.1	<a href="#">Enumeration Type Documentation</a>	45
8.3.1.1	<a href="#">PGRSyncError</a>	45
8.3.1.2	<a href="#">PGRSyncMessage</a>	45
<b>9</b>	<b><a href="#">Class Documentation</a></b>	<b>47</b>
9.1	<a href="#">AVIOption Struct Reference</a>	47
9.1.1	<a href="#">Detailed Description</a>	47
9.1.2	<a href="#">Constructor &amp; Destructor Documentation</a>	47
9.1.2.1	<a href="#">AVIOption</a>	47
9.1.3	<a href="#">Member Data Documentation</a>	47
9.1.3.1	<a href="#">frameRate</a>	47
9.1.3.2	<a href="#">reserved</a>	48
9.2	<a href="#">AVIRecorder Class Reference</a>	48
9.2.1	<a href="#">Detailed Description</a>	48
9.2.2	<a href="#">Constructor &amp; Destructor Documentation</a>	48
9.2.2.1	<a href="#">AVIRecorder</a>	48
9.2.2.2	<a href="#">~AVIRecorder</a>	48
9.2.3	<a href="#">Member Function Documentation</a>	49
9.2.3.1	<a href="#">AVIAppend</a>	49
9.2.3.2	<a href="#">AVIClose</a>	49
9.2.3.3	<a href="#">AVIOpen</a>	49
9.2.3.4	<a href="#">AVIOpen</a>	50
9.2.3.5	<a href="#">AVIOpen</a>	50
9.3	<a href="#">BMPOption Struct Reference</a>	51
9.3.1	<a href="#">Detailed Description</a>	51
9.3.2	<a href="#">Constructor &amp; Destructor Documentation</a>	51

9.3.2.1	BMPOption . . . . .	51
9.3.3	Member Data Documentation . . . . .	51
9.3.3.1	indexedColor_8bit . . . . .	51
9.3.3.2	reserved . . . . .	51
9.4	BusManager Class Reference . . . . .	51
9.4.1	Detailed Description . . . . .	53
9.4.2	Constructor & Destructor Documentation . . . . .	53
9.4.2.1	BusManager . . . . .	53
9.4.2.2	~BusManager . . . . .	53
9.4.3	Member Function Documentation . . . . .	53
9.4.3.1	DiscoverGigECameras . . . . .	54
9.4.3.2	FireBusReset . . . . .	54
9.4.3.3	ForceAllIPAddressesAutomatically . . . . .	54
9.4.3.4	ForceAllIPAddressesAutomatically . . . . .	55
9.4.3.5	ForceIPAddressToCamera . . . . .	55
9.4.3.6	GetCameraFromIndex . . . . .	55
9.4.3.7	GetCameraFromIPAddress . . . . .	56
9.4.3.8	GetCameraFromSerialNumber . . . . .	56
9.4.3.9	GetCameraSerialNumberFromIndex . . . . .	57
9.4.3.10	GetDeviceFromIndex . . . . .	57
9.4.3.11	GetInterfaceTypeFromGuid . . . . .	57
9.4.3.12	GetNumOfCameras . . . . .	58
9.4.3.13	GetNumOfDevices . . . . .	58
9.4.3.14	GetTopology . . . . .	58
9.4.3.15	GetUsbLinkInfo . . . . .	59
9.4.3.16	GetUsbPortStatus . . . . .	59
9.4.3.17	IsCameraControlable . . . . .	59
9.4.3.18	ReadPhyRegister . . . . .	59
9.4.3.19	RegisterCallback . . . . .	60
9.4.3.20	RescanBus . . . . .	60
9.4.3.21	UnregisterCallback . . . . .	61
9.4.3.22	WritePhyRegister . . . . .	61
9.5	Camera Class Reference . . . . .	61
9.5.1	Detailed Description . . . . .	66

9.5.2	Constructor & Destructor Documentation . . . . .	66
9.5.2.1	Camera . . . . .	66
9.5.2.2	~Camera . . . . .	66
9.5.3	Member Function Documentation . . . . .	66
9.5.3.1	Connect . . . . .	66
9.5.3.2	DeregisterAllEvents . . . . .	67
9.5.3.3	DeregisterEvent . . . . .	67
9.5.3.4	Disconnect . . . . .	67
9.5.3.5	EnableLUT . . . . .	67
9.5.3.6	FireSoftwareTrigger . . . . .	68
9.5.3.7	GetActiveLUTBank . . . . .	68
9.5.3.8	GetCameraInfo . . . . .	68
9.5.3.9	GetConfiguration . . . . .	68
9.5.3.10	GetCycleTime . . . . .	69
9.5.3.11	GetEmbeddedImageInfo . . . . .	69
9.5.3.12	GetFormat7Configuration . . . . .	70
9.5.3.13	GetFormat7Info . . . . .	70
9.5.3.14	GetGPIOPinDirection . . . . .	71
9.5.3.15	GetLUTBankInfo . . . . .	71
9.5.3.16	GetLUTChannel . . . . .	71
9.5.3.17	GetLUTInfo . . . . .	72
9.5.3.18	GetMemoryChannel . . . . .	72
9.5.3.19	GetMemoryChannelInfo . . . . .	73
9.5.3.20	GetProperty . . . . .	73
9.5.3.21	GetPropertyInfo . . . . .	74
9.5.3.22	GetRegisterString . . . . .	74
9.5.3.23	GetStats . . . . .	75
9.5.3.24	GetStrobe . . . . .	75
9.5.3.25	GetStrobeInfo . . . . .	75
9.5.3.26	GetTriggerDelay . . . . .	75
9.5.3.27	GetTriggerDelayInfo . . . . .	76
9.5.3.28	GetTriggerMode . . . . .	76
9.5.3.29	GetTriggerModeInfo . . . . .	77
9.5.3.30	GetVideoModeAndFrameRate . . . . .	77

9.5.3.31	GetVideoModeAndFrameRateInfo . . . . .	78
9.5.3.32	IsConnected . . . . .	78
9.5.3.33	ReadRegister . . . . .	79
9.5.3.34	ReadRegisterBlock . . . . .	79
9.5.3.35	RegisterAllEvents . . . . .	79
9.5.3.36	RegisterEvent . . . . .	80
9.5.3.37	ResetStats . . . . .	80
9.5.3.38	RestoreFromMemoryChannel . . . . .	80
9.5.3.39	RetrieveBuffer . . . . .	80
9.5.3.40	SaveToMemoryChannel . . . . .	81
9.5.3.41	SetActiveLUTBank . . . . .	81
9.5.3.42	SetCallback . . . . .	81
9.5.3.43	SetConfiguration . . . . .	82
9.5.3.44	SetEmbeddedImageInfo . . . . .	82
9.5.3.45	SetFormat7Configuration . . . . .	83
9.5.3.46	SetFormat7Configuration . . . . .	83
9.5.3.47	SetGPIOPinDirection . . . . .	84
9.5.3.48	SetLUTChannel . . . . .	84
9.5.3.49	SetProperty . . . . .	85
9.5.3.50	SetStrobe . . . . .	85
9.5.3.51	SetTriggerDelay . . . . .	86
9.5.3.52	SetTriggerMode . . . . .	86
9.5.3.53	SetUserBuffers . . . . .	87
9.5.3.54	SetVideoModeAndFrameRate . . . . .	87
9.5.3.55	StartCapture . . . . .	88
9.5.3.56	StartSyncCapture . . . . .	88
9.5.3.57	StopCapture . . . . .	88
9.5.3.58	ValidateFormat7Settings . . . . .	89
9.5.3.59	WaitForBufferEvent . . . . .	89
9.5.3.60	WriteRegister . . . . .	90
9.5.3.61	WriteRegisterBlock . . . . .	90
9.6	CameraBase Class Reference . . . . .	91
9.6.1	Detailed Description . . . . .	95
9.6.2	Constructor & Destructor Documentation . . . . .	95



9.6.2.1	CameraBase	95
9.6.2.2	~CameraBase	96
9.6.3	Member Function Documentation	96
9.6.3.1	Connect	96
9.6.3.2	DeregisterAllEvents	96
9.6.3.3	DeregisterEvent	96
9.6.3.4	Disconnect	96
9.6.3.5	EnableLUT	97
9.6.3.6	FireSoftwareTrigger	97
9.6.3.7	GetActiveLUTBank	97
9.6.3.8	GetCameraInfo	98
9.6.3.9	GetConfiguration	98
9.6.3.10	GetCycleTime	98
9.6.3.11	GetEmbeddedImageInfo	99
9.6.3.12	GetGPIOPinDirection	99
9.6.3.13	GetLUTBankInfo	100
9.6.3.14	GetLUTChannel	100
9.6.3.15	GetLUTInfo	101
9.6.3.16	GetMemoryChannel	101
9.6.3.17	GetMemoryChannelInfo	101
9.6.3.18	GetProperty	102
9.6.3.19	GetPropertyInfo	102
9.6.3.20	GetRegisterString	103
9.6.3.21	GetStats	103
9.6.3.22	GetStrobe	103
9.6.3.23	GetStrobeInfo	104
9.6.3.24	GetTriggerDelay	104
9.6.3.25	GetTriggerDelayInfo	105
9.6.3.26	GetTriggerMode	105
9.6.3.27	GetTriggerModelInfo	106
9.6.3.28	IsConnected	106
9.6.3.29	ReadRegister	106
9.6.3.30	ReadRegisterBlock	107
9.6.3.31	RegisterAllEvents	107

9.6.3.32	RegisterEvent	107
9.6.3.33	ResetStats	107
9.6.3.34	RestoreFromMemoryChannel	108
9.6.3.35	RetrieveBuffer	108
9.6.3.36	SaveToMemoryChannel	109
9.6.3.37	SetActiveLUTBank	109
9.6.3.38	SetCallback	109
9.6.3.39	SetConfiguration	110
9.6.3.40	SetEmbeddedImageInfo	110
9.6.3.41	SetGPIOPinDirection	111
9.6.3.42	SetLUTChannel	111
9.6.3.43	SetProperty	112
9.6.3.44	SetStrobe	112
9.6.3.45	SetTriggerDelay	113
9.6.3.46	SetTriggerMode	113
9.6.3.47	SetUserBuffers	114
9.6.3.48	StartCapture	114
9.6.3.49	StartSyncCapture	115
9.6.3.50	StopCapture	116
9.6.3.51	WaitForBufferEvent	116
9.6.3.52	WriteRegister	116
9.6.3.53	WriteRegisterBlock	117
9.6.4	Member Data Documentation	117
9.6.4.1	m_pCameraData	117
9.7	CameraControlDlg Class Reference	118
9.7.1	Detailed Description	118
9.7.2	Constructor & Destructor Documentation	118
9.7.2.1	CameraControlDlg	118
9.7.2.2	~CameraControlDlg	119
9.7.3	Member Function Documentation	119
9.7.3.1	Connect	119
9.7.3.2	Disconnect	119
9.7.3.3	Hide	119
9.7.3.4	IsVisible	119

9.7.3.5	SetTitle	119
9.7.3.6	Show	119
9.7.3.7	Show	120
9.7.3.8	ShowModal	120
9.7.3.9	ShowModal	120
9.8	CameraInfo Struct Reference	120
9.8.1	Detailed Description	122
9.8.2	Constructor & Destructor Documentation	122
9.8.2.1	CameraInfo	122
9.8.3	Member Data Documentation	122
9.8.3.1	applicationIPAddress	122
9.8.3.2	applicationPort	122
9.8.3.3	bayerTileFormat	122
9.8.3.4	busNumber	123
9.8.3.5	ccpStatus	123
9.8.3.6	configROM	123
9.8.3.7	defaultGateway	123
9.8.3.8	driverName	123
9.8.3.9	driverType	123
9.8.3.10	firmwareBuildTime	123
9.8.3.11	firmwareVersion	123
9.8.3.12	gigEMajorVersion	123
9.8.3.13	gigEMinorVersion	123
9.8.3.14	iidcVer	124
9.8.3.15	interfaceType	124
9.8.3.16	ipAddress	124
9.8.3.17	isColorCamera	124
9.8.3.18	macAddress	124
9.8.3.19	maximumBusSpeed	124
9.8.3.20	modelName	124
9.8.3.21	nodeNumber	124
9.8.3.22	pcieBusSpeed	124
9.8.3.23	reserved	124
9.8.3.24	sensorInfo	125

9.8.3.25	sensorResolution	125
9.8.3.26	serialNumber	125
9.8.3.27	subnetMask	125
9.8.3.28	userDefinedName	125
9.8.3.29	vendorName	125
9.8.3.30	xmlURL1	125
9.8.3.31	xmlURL2	125
9.9	CameraSelectionDlg Class Reference	125
9.9.1	Detailed Description	126
9.9.2	Constructor & Destructor Documentation	126
9.9.2.1	CameraSelectionDlg	126
9.9.2.2	~CameraSelectionDlg	126
9.9.3	Member Function Documentation	126
9.9.3.1	SetTitle	126
9.9.3.2	ShowModal	126
9.10	CameraStats Struct Reference	127
9.10.1	Detailed Description	128
9.10.2	Constructor & Destructor Documentation	128
9.10.2.1	CameraStats	128
9.10.3	Member Data Documentation	128
9.10.3.1	cameraCurrents	128
9.10.3.2	cameraPowerUp	128
9.10.3.3	cameraVoltages	128
9.10.3.4	imageCorrupt	128
9.10.3.5	imageDriverDropped	128
9.10.3.6	imageDropped	128
9.10.3.7	imageXmitFailed	128
9.10.3.8	numCurrents	128
9.10.3.9	numResendPacketsReceived	129
9.10.3.10	numResendPacketsRequested	129
9.10.3.11	numVoltages	129
9.10.3.12	portErrors	129
9.10.3.13	regReadFailed	129
9.10.3.14	regWriteFailed	129

9.10.3.15 reserved	129
9.10.3.16 temperature	129
9.10.3.17 timeSinceBusReset	129
9.10.3.18 timeSinceInitialization	129
9.10.3.19 timeStamp	129
9.11 ConfigROM Struct Reference	129
9.11.1 Detailed Description	130
9.11.2 Constructor & Destructor Documentation	130
9.11.2.1 ConfigROM	130
9.11.3 Member Data Documentation	130
9.11.3.1 chipIdHi	130
9.11.3.2 chipIdLo	130
9.11.3.3 nodeVendorId	131
9.11.3.4 pszKeyword	131
9.11.3.5 reserved	131
9.11.3.6 unitSpecId	131
9.11.3.7 unitSubSWVer	131
9.11.3.8 unitSWVer	131
9.11.3.9 vendorUniqueInfo_0	131
9.11.3.10 vendorUniqueInfo_1	131
9.11.3.11 vendorUniqueInfo_2	131
9.11.3.12 vendorUniqueInfo_3	131
9.12 EmbeddedImageInfo Struct Reference	132
9.12.1 Detailed Description	132
9.12.2 Member Data Documentation	133
9.12.2.1 brightness	133
9.12.2.2 exposure	133
9.12.2.3 frameCounter	133
9.12.2.4 gain	133
9.12.2.5 GPIOPinState	133
9.12.2.6 ROIPosition	133
9.12.2.7 shutter	133
9.12.2.8 strobePattern	133
9.12.2.9 timestamp	133

9.12.2.10 whiteBalance . . . . .	133
9.13 EmbeddedImageInfoProperty Struct Reference . . . . .	133
9.13.1 Detailed Description . . . . .	133
9.13.2 Constructor & Destructor Documentation . . . . .	134
9.13.2.1 EmbeddedImageInfoProperty . . . . .	134
9.13.3 Member Data Documentation . . . . .	134
9.13.3.1 available . . . . .	134
9.13.3.2 onOff . . . . .	134
9.14 Error Class Reference . . . . .	134
9.14.1 Detailed Description . . . . .	135
9.14.2 Constructor & Destructor Documentation . . . . .	135
9.14.2.1 Error . . . . .	135
9.14.2.2 Error . . . . .	135
9.14.2.3 ~Error . . . . .	135
9.14.3 Member Function Documentation . . . . .	136
9.14.3.1 CollectSupportInformation . . . . .	136
9.14.3.2 GetBuildDate . . . . .	136
9.14.3.3 GetCause . . . . .	136
9.14.3.4 GetDescription . . . . .	136
9.14.3.5 GetFilename . . . . .	136
9.14.3.6 GetLine . . . . .	137
9.14.3.7 GetType . . . . .	137
9.14.3.8 operator!= . . . . .	137
9.14.3.9 operator!= . . . . .	137
9.14.3.10 operator= . . . . .	137
9.14.3.11 operator== . . . . .	137
9.14.3.12 operator== . . . . .	137
9.14.3.13 PrintErrorTrace . . . . .	137
9.14.4 Friends And Related Function Documentation . . . . .	138
9.14.4.1 InternalError . . . . .	138
9.15 EventCallbackData Struct Reference . . . . .	138
9.15.1 Member Data Documentation . . . . .	138
9.15.1.1 EventData . . . . .	138
9.15.1.2 EventDataSize . . . . .	138

9.15.1.3	EventID	139
9.15.1.4	EventName	139
9.15.1.5	EventTimestamp	139
9.15.1.6	EventUserData	139
9.15.1.7	EventUserDataSize	139
9.16	EventOptions Struct Reference	139
9.16.1	Detailed Description	140
9.16.2	Member Data Documentation	140
9.16.2.1	EventCallbackFcn	140
9.16.2.2	EventName	140
9.16.2.3	EventUserData	140
9.16.2.4	EventUserDataSize	140
9.17	FC2Config Struct Reference	140
9.17.1	Detailed Description	141
9.17.2	Constructor & Destructor Documentation	141
9.17.2.1	FC2Config	141
9.17.3	Member Data Documentation	141
9.17.3.1	asyncBusSpeed	141
9.17.3.2	bandwidthAllocation	141
9.17.3.3	grabMode	142
9.17.3.4	grabTimeout	142
9.17.3.5	highPerformanceRetrieveBuffer	142
9.17.3.6	isochBusSpeed	142
9.17.3.7	minNumImageNotifications	142
9.17.3.8	numBuffers	142
9.17.3.9	numImageNotifications	142
9.17.3.10	registerTimeout	143
9.17.3.11	registerTimeoutRetries	143
9.17.3.12	reserved	143
9.18	FC2Version Struct Reference	143
9.18.1	Detailed Description	144
9.18.2	Member Data Documentation	144
9.18.2.1	build	144
9.18.2.2	major	144

9.18.2.3	minor	144
9.18.2.4	type	144
9.19	FlyCapture3ApiGuiWrapper Class Reference	144
9.19.1	Constructor & Destructor Documentation	145
9.19.1.1	FlyCapture3ApiGuiWrapper	145
9.19.1.2	~FlyCapture3ApiGuiWrapper	145
9.19.2	Member Function Documentation	145
9.19.2.1	ConnectGUILibrary	145
9.19.2.2	DisconnectGUILibrary	145
9.19.2.3	GetControlNameList	145
9.19.2.4	GetDialogNameList	145
9.19.2.5	GetNumDialogs	145
9.19.2.6	GetNumOfControls	145
9.19.2.7	ShowCameraSelectionDialog	145
9.19.2.8	ShowDialogByIndex	145
9.19.2.9	ShowDialogByName	145
9.19.2.10	ShowPropertyGridDialog	145
9.20	Format7ImageSettings Struct Reference	145
9.20.1	Detailed Description	146
9.20.2	Constructor & Destructor Documentation	146
9.20.2.1	Format7ImageSettings	146
9.20.3	Member Data Documentation	146
9.20.3.1	height	146
9.20.3.2	mode	146
9.20.3.3	offsetX	146
9.20.3.4	offsetY	146
9.20.3.5	pixelFormat	146
9.20.3.6	reserved	147
9.20.3.7	width	147
9.21	Format7Info Struct Reference	147
9.21.1	Detailed Description	148
9.21.2	Constructor & Destructor Documentation	148
9.21.2.1	Format7Info	148
9.21.3	Member Data Documentation	148



9.21.3.1	<a href="#">imageHStepSize</a>	148
9.21.3.2	<a href="#">imageVStepSize</a>	148
9.21.3.3	<a href="#">maxHeight</a>	148
9.21.3.4	<a href="#">maxPacketSize</a>	148
9.21.3.5	<a href="#">maxWidth</a>	148
9.21.3.6	<a href="#">minPacketSize</a>	148
9.21.3.7	<a href="#">mode</a>	149
9.21.3.8	<a href="#">offsetHStepSize</a>	149
9.21.3.9	<a href="#">offsetVStepSize</a>	149
9.21.3.10	<a href="#">packetSize</a>	149
9.21.3.11	<a href="#">percentage</a>	149
9.21.3.12	<a href="#">pixelFormatBitField</a>	149
9.21.3.13	<a href="#">reserved</a>	149
9.21.3.14	<a href="#">vendorPixelFormatBitField</a>	149
9.22	<a href="#">Format7PacketInfo Struct Reference</a>	149
9.22.1	<a href="#">Detailed Description</a>	150
9.22.2	<a href="#">Constructor &amp; Destructor Documentation</a>	150
9.22.2.1	<a href="#">Format7PacketInfo</a>	150
9.22.3	<a href="#">Member Data Documentation</a>	150
9.22.3.1	<a href="#">maxBytesPerPacket</a>	150
9.22.3.2	<a href="#">recommendedBytesPerPacket</a>	150
9.22.3.3	<a href="#">reserved</a>	150
9.22.3.4	<a href="#">unitBytesPerPacket</a>	150
9.23	<a href="#">GCCamera Class Reference</a>	151
9.23.1	<a href="#">Constructor &amp; Destructor Documentation</a>	155
9.23.1.1	<a href="#">GCCamera</a>	155
9.23.1.2	<a href="#">~GCCamera</a>	155
9.23.2	<a href="#">Member Function Documentation</a>	155
9.23.2.1	<a href="#">Connect</a>	155
9.23.2.2	<a href="#">Connect</a>	155
9.23.2.3	<a href="#">Disconnect</a>	155
9.23.2.4	<a href="#">EnableLUT</a>	155
9.23.2.5	<a href="#">FireSoftwareTrigger</a>	156
9.23.2.6	<a href="#">GCCamera::GetXML</a>	156

9.23.2.7	GetActiveLUTBank	156
9.23.2.8	GetCameraInfo	156
9.23.2.9	GetConfiguration	157
9.23.2.10	GetCycleTime	157
9.23.2.11	GetEmbeddedImageInfo	157
9.23.2.12	GetGPIOPinDirection	158
9.23.2.13	GetInterfaceType	158
9.23.2.14	GetLUTBankInfo	158
9.23.2.15	GetLUTChannel	159
9.23.2.16	GetLUTInfo	159
9.23.2.17	GetMemoryChannel	160
9.23.2.18	GetMemoryChannelInfo	160
9.23.2.19	GetNodeMap	161
9.23.2.20	GetProperty	161
9.23.2.21	GetPropertyInfo	161
9.23.2.22	GetRegisterString	162
9.23.2.23	GetStats	162
9.23.2.24	GetStrobe	162
9.23.2.25	GetStrobeInfo	162
9.23.2.26	GetTriggerDelay	163
9.23.2.27	GetTriggerDelayInfo	163
9.23.2.28	GetTriggerMode	164
9.23.2.29	GetTriggerModelInfo	164
9.23.2.30	IsConnected	165
9.23.2.31	ReadGVCPMemory	165
9.23.2.32	ReadGVCPRegister	165
9.23.2.33	ReadGVCPRegisterBlock	165
9.23.2.34	ReadRegister	165
9.23.2.35	ReadRegisterBlock	166
9.23.2.36	ResetStats	166
9.23.2.37	RestoreFromMemoryChannel	166
9.23.2.38	RetrieveBuffer	167
9.23.2.39	SaveToMemoryChannel	167
9.23.2.40	SetActiveLUTBank	168

9.23.2.41 SetCallback . . . . .	168
9.23.2.42 SetCamera . . . . .	169
9.23.2.43 SetCamera . . . . .	169
9.23.2.44 SetConfiguration . . . . .	169
9.23.2.45 SetEmbeddedImageInfo . . . . .	169
9.23.2.46 SetGPIOPinDirection . . . . .	169
9.23.2.47 SetLUTChannel . . . . .	170
9.23.2.48 SetProperty . . . . .	170
9.23.2.49 SetStrobe . . . . .	171
9.23.2.50 SetTriggerDelay . . . . .	171
9.23.2.51 SetTriggerMode . . . . .	172
9.23.2.52 SetUserBuffers . . . . .	172
9.23.2.53 StartCapture . . . . .	173
9.23.2.54 StartSyncCapture . . . . .	174
9.23.2.55 StopCapture . . . . .	174
9.23.2.56 TestGainNode . . . . .	174
9.23.2.57 WaitForBufferEvent . . . . .	174
9.23.2.58 WriteGVCPMemory . . . . .	175
9.23.2.59 WriteGVCPRegister . . . . .	175
9.23.2.60 WriteGVCPRegisterBlock . . . . .	175
9.23.2.61 WriteRegister . . . . .	175
9.23.2.62 WriteRegisterBlock . . . . .	175
9.23.3 Member Data Documentation . . . . .	176
9.23.3.1 m_busMgr . . . . .	176
9.24 GigECamera Class Reference . . . . .	176
9.24.1 Detailed Description . . . . .	182
9.24.2 Constructor & Destructor Documentation . . . . .	182
9.24.2.1 GigECamera . . . . .	182
9.24.2.2 ~GigECamera . . . . .	182
9.24.3 Member Function Documentation . . . . .	182
9.24.3.1 Connect . . . . .	182
9.24.3.2 DeregisterAllEvents . . . . .	182
9.24.3.3 DeregisterEvent . . . . .	182
9.24.3.4 Disconnect . . . . .	182

9.24.3.5 DiscoverGigEPacketSize . . . . .	183
9.24.3.6 EnableLUT . . . . .	183
9.24.3.7 FireSoftwareTrigger . . . . .	183
9.24.3.8 GetActiveLUTBank . . . . .	184
9.24.3.9 GetCameraInfo . . . . .	184
9.24.3.10 GetConfiguration . . . . .	184
9.24.3.11 GetCycleTime . . . . .	185
9.24.3.12 GetEmbeddedImageInfo . . . . .	185
9.24.3.13 GetGigEConfig . . . . .	185
9.24.3.14 GetGigEImageBinningSettings . . . . .	186
9.24.3.15 GetGigEImageSettings . . . . .	186
9.24.3.16 GetGigEImageSettingsInfo . . . . .	186
9.24.3.17 GetGigEImagingMode . . . . .	187
9.24.3.18 GetGigEProperty . . . . .	187
9.24.3.19 GetGigEStreamChannelInfo . . . . .	187
9.24.3.20 GetGPIOPinDirection . . . . .	188
9.24.3.21 GetLUTBankInfo . . . . .	188
9.24.3.22 GetLUTChannel . . . . .	189
9.24.3.23 GetLUTInfo . . . . .	189
9.24.3.24 GetMemoryChannel . . . . .	190
9.24.3.25 GetMemoryChannelInfo . . . . .	190
9.24.3.26 GetNumStreamChannels . . . . .	191
9.24.3.27 GetProperty . . . . .	191
9.24.3.28 GetPropertyInfo . . . . .	191
9.24.3.29 GetRegisterString . . . . .	192
9.24.3.30 GetStats . . . . .	192
9.24.3.31 GetStrobe . . . . .	192
9.24.3.32 GetStrobeInfo . . . . .	193
9.24.3.33 GetTriggerDelay . . . . .	193
9.24.3.34 GetTriggerDelayInfo . . . . .	194
9.24.3.35 GetTriggerMode . . . . .	194
9.24.3.36 GetTriggerModelInfo . . . . .	195
9.24.3.37 IsConnected . . . . .	195
9.24.3.38 QueryGigEImagingMode . . . . .	195

9.24.3.39 ReadGVCPMemory . . . . .	196
9.24.3.40 ReadGVCPRegister . . . . .	196
9.24.3.41 ReadGVCPRegisterBlock . . . . .	196
9.24.3.42 ReadRegister . . . . .	197
9.24.3.43 ReadRegisterBlock . . . . .	197
9.24.3.44 RegisterAllEvents . . . . .	198
9.24.3.45 RegisterEvent . . . . .	198
9.24.3.46 ResetStats . . . . .	198
9.24.3.47 RestoreFromMemoryChannel . . . . .	198
9.24.3.48 RetrieveBuffer . . . . .	198
9.24.3.49 SaveToMemoryChannel . . . . .	199
9.24.3.50 SetActiveLUTBank . . . . .	199
9.24.3.51 SetCallback . . . . .	200
9.24.3.52 SetConfiguration . . . . .	200
9.24.3.53 SetEmbeddedImageInfo . . . . .	200
9.24.3.54 SetGigEConfig . . . . .	201
9.24.3.55 SetGigEImageBinningSettings . . . . .	201
9.24.3.56 SetGigEImageSettings . . . . .	201
9.24.3.57 SetGigEImagingMode . . . . .	202
9.24.3.58 SetGigEProperty . . . . .	202
9.24.3.59 SetGigEStreamChannelInfo . . . . .	202
9.24.3.60 SetGPIOPinDirection . . . . .	203
9.24.3.61 SetLUTChannel . . . . .	203
9.24.3.62 SetProperty . . . . .	204
9.24.3.63 SetStrobe . . . . .	204
9.24.3.64 SetTriggerDelay . . . . .	205
9.24.3.65 SetTriggerMode . . . . .	205
9.24.3.66 SetUserBuffers . . . . .	206
9.24.3.67 StartCapture . . . . .	206
9.24.3.68 StartSyncCapture . . . . .	207
9.24.3.69 StopCapture . . . . .	207
9.24.3.70 WaitForBufferEvent . . . . .	208
9.24.3.71 WriteGVCPMemory . . . . .	208
9.24.3.72 WriteGVCPRegister . . . . .	208

9.24.3.73 WriteGVCPRegisterBlock . . . . .	209
9.24.3.74 WriteRegister . . . . .	209
9.24.3.75 WriteRegisterBlock . . . . .	209
9.25 GigEConfig Struct Reference . . . . .	210
9.25.1 Detailed Description . . . . .	210
9.25.2 Constructor & Destructor Documentation . . . . .	211
9.25.2.1 GigEConfig . . . . .	211
9.25.3 Member Data Documentation . . . . .	211
9.25.3.1 enablePacketResend . . . . .	211
9.25.3.2 registerTimeout . . . . .	211
9.25.3.3 registerTimeoutRetries . . . . .	211
9.26 GigEImageSettings Struct Reference . . . . .	211
9.26.1 Detailed Description . . . . .	212
9.26.2 Constructor & Destructor Documentation . . . . .	212
9.26.2.1 GigEImageSettings . . . . .	212
9.26.3 Member Data Documentation . . . . .	212
9.26.3.1 height . . . . .	212
9.26.3.2 offsetX . . . . .	212
9.26.3.3 offsetY . . . . .	212
9.26.3.4 pixelFormat . . . . .	212
9.26.3.5 reserved . . . . .	212
9.26.3.6 width . . . . .	212
9.27 GigEImageSettingsInfo Struct Reference . . . . .	213
9.27.1 Detailed Description . . . . .	213
9.27.2 Constructor & Destructor Documentation . . . . .	213
9.27.2.1 GigEImageSettingsInfo . . . . .	213
9.27.3 Member Data Documentation . . . . .	213
9.27.3.1 imageHStepSize . . . . .	213
9.27.3.2 imageVStepSize . . . . .	214
9.27.3.3 maxHeight . . . . .	214
9.27.3.4 maxWidth . . . . .	214
9.27.3.5 offsetHStepSize . . . . .	214
9.27.3.6 offsetVStepSize . . . . .	214
9.27.3.7 pixelFormatBitField . . . . .	214

9.27.3.8 reserved	214
9.27.3.9 vendorPixelFormatBitField	214
9.28 GigEProperty Struct Reference	214
9.28.1 Detailed Description	215
9.28.2 Member Data Documentation	215
9.28.2.1 isReadable	215
9.28.2.2 isWritable	215
9.28.2.3 max	215
9.28.2.4 min	215
9.28.2.5 propType	215
9.28.2.6 value	216
9.29 GigEStreamChannel Struct Reference	216
9.29.1 Detailed Description	217
9.29.2 Constructor & Destructor Documentation	217
9.29.2.1 GigEStreamChannel	217
9.29.3 Member Data Documentation	217
9.29.3.1 destinationIpAddress	217
9.29.3.2 doNotFragment	217
9.29.3.3 hostPort	217
9.29.3.4 interPacketDelay	217
9.29.3.5 networkInterfaceIndex	217
9.29.3.6 packetSize	217
9.29.3.7 sourcePort	218
9.30 H264Option Struct Reference	218
9.30.1 Detailed Description	218
9.30.2 Constructor & Destructor Documentation	218
9.30.2.1 H264Option	218
9.30.3 Member Data Documentation	218
9.30.3.1 bitrate	218
9.30.3.2 frameRate	219
9.30.3.3 height	219
9.30.3.4 reserved	219
9.30.3.5 width	219
9.31 Image Class Reference	219

9.31.1 Detailed Description . . . . .	222
9.31.2 Constructor & Destructor Documentation . . . . .	222
9.31.2.1 Image . . . . .	222
9.31.2.2 Image . . . . .	222
9.31.2.3 Image . . . . .	223
9.31.2.4 Image . . . . .	223
9.31.2.5 Image . . . . .	223
9.31.2.6 Image . . . . .	224
9.31.2.7 ~Image . . . . .	224
9.31.3 Member Function Documentation . . . . .	224
9.31.3.1 CalculateStatistics . . . . .	224
9.31.3.2 Convert . . . . .	224
9.31.3.3 Convert . . . . .	225
9.31.3.4 DeepCopy . . . . .	225
9.31.3.5 DetermineBitsPerPixel . . . . .	225
9.31.3.6 GetBayerTileFormat . . . . .	225
9.31.3.7 GetBitsPerPixel . . . . .	226
9.31.3.8 GetBlockId . . . . .	226
9.31.3.9 GetColorProcessing . . . . .	226
9.31.3.10 GetCols . . . . .	226
9.31.3.11 GetData . . . . .	227
9.31.3.12 GetData . . . . .	227
9.31.3.13 GetDataSize . . . . .	227
9.31.3.14 GetDefaultColorProcessing . . . . .	227
9.31.3.15 GetDefaultOutputFormat . . . . .	227
9.31.3.16 GetDimensions . . . . .	228
9.31.3.17 GetMetadata . . . . .	228
9.31.3.18 GetPixelFormat . . . . .	228
9.31.3.19 GetReceivedDataSize . . . . .	228
9.31.3.20 GetRows . . . . .	229
9.31.3.21 GetStride . . . . .	229
9.31.3.22 GetTimeStamp . . . . .	229
9.31.3.23 operator() . . . . .	229
9.31.3.24 operator= . . . . .	229



9.31.3.25 operator[]	230
9.31.3.26 ReleaseBuffer	230
9.31.3.27 Save	230
9.31.3.28 Save	230
9.31.3.29 Save	231
9.31.3.30 Save	231
9.31.3.31 Save	231
9.31.3.32 Save	232
9.31.3.33 Save	232
9.31.3.34 Save	232
9.31.3.35 SetBlockId	233
9.31.3.36 SetColorProcessing	233
9.31.3.37 SetData	233
9.31.3.38 SetDefaultColorProcessing	233
9.31.3.39 SetDefaultOutputFormat	234
9.31.3.40 SetDimensions	234
9.31.4 Friends And Related Function Documentation	235
9.31.4.1 Iso	235
9.32 ImageMetadata Struct Reference	235
9.32.1 Detailed Description	236
9.32.2 Constructor & Destructor Documentation	236
9.32.2.1 ImageMetadata	236
9.32.3 Member Data Documentation	236
9.32.3.1 embeddedBrightness	236
9.32.3.2 embeddedExposure	236
9.32.3.3 embeddedFrameCounter	236
9.32.3.4 embeddedGain	236
9.32.3.5 embeddedGPIOPinState	236
9.32.3.6 embeddedROIPosition	237
9.32.3.7 embeddedShutter	237
9.32.3.8 embeddedStrobePattern	237
9.32.3.9 embeddedTimeStamp	237
9.32.3.10 embeddedWhiteBalance	237
9.32.3.11 reserved	237

9.33	ImageStatistics Class Reference	237
9.33.1	Detailed Description	239
9.33.2	Member Enumeration Documentation	239
9.33.2.1	StatisticsChannel	239
9.33.3	Constructor & Destructor Documentation	239
9.33.3.1	ImageStatistics	239
9.33.3.2	~ImageStatistics	239
9.33.3.3	ImageStatistics	239
9.33.4	Member Function Documentation	239
9.33.4.1	DisableAll	239
9.33.4.2	EnableAll	240
9.33.4.3	EnableGreyOnly	240
9.33.4.4	EnableHSLOnly	240
9.33.4.5	EnableRGBOnly	240
9.33.4.6	GetChannelStatus	240
9.33.4.7	GetHistogram	241
9.33.4.8	GetMean	241
9.33.4.9	GetNumPixelValues	241
9.33.4.10	GetPixelValueRange	242
9.33.4.11	GetRange	242
9.33.4.12	GetStatistics	242
9.33.4.13	operator=	243
9.33.4.14	SetChannelStatus	243
9.33.5	Friends And Related Function Documentation	243
9.33.5.1	ImageStatsCalculator	243
9.34	Internal Class Reference	244
9.34.1	Member Function Documentation	244
9.34.1.1	GetInternal	244
9.35	IPAddress Struct Reference	244
9.35.1	Detailed Description	244
9.35.2	Constructor & Destructor Documentation	245
9.35.2.1	IPAddress	245
9.35.2.2	IPAddress	245
9.35.3	Member Function Documentation	245

9.35.3.1	operator!=	245
9.35.3.2	operator==	245
9.35.4	Member Data Documentation	245
9.35.4.1	octets	245
9.36	JPEGOption Struct Reference	245
9.36.1	Detailed Description	246
9.36.2	Constructor & Destructor Documentation	246
9.36.2.1	JPEGOption	246
9.36.3	Member Data Documentation	246
9.36.3.1	progressive	246
9.36.3.2	quality	246
9.36.3.3	reserved	246
9.37	JPG2Option Struct Reference	246
9.37.1	Detailed Description	247
9.37.2	Constructor & Destructor Documentation	247
9.37.2.1	JPG2Option	247
9.37.3	Member Data Documentation	247
9.37.3.1	quality	247
9.37.3.2	reserved	247
9.38	LUTData Struct Reference	247
9.38.1	Detailed Description	248
9.38.2	Constructor & Destructor Documentation	248
9.38.2.1	LUTData	248
9.38.3	Member Data Documentation	248
9.38.3.1	enabled	248
9.38.3.2	inputBitDepth	248
9.38.3.3	numBanks	248
9.38.3.4	numChannels	248
9.38.3.5	numEntries	248
9.38.3.6	outputBitDepth	249
9.38.3.7	reserved	249
9.38.3.8	supported	249
9.39	MACAddress Struct Reference	249
9.39.1	Detailed Description	249

9.39.2	Constructor & Destructor Documentation	249
9.39.2.1	MACAddress	249
9.39.2.2	MACAddress	250
9.39.3	Member Function Documentation	250
9.39.3.1	operator!=	250
9.39.3.2	operator==	250
9.39.4	Member Data Documentation	250
9.39.4.1	octets	250
9.40	MJPGOption Struct Reference	250
9.40.1	Detailed Description	250
9.40.2	Constructor & Destructor Documentation	251
9.40.2.1	MJPGOption	251
9.40.3	Member Data Documentation	251
9.40.3.1	frameRate	251
9.40.3.2	quality	251
9.40.3.3	reserved	251
9.41	NodeMap Class Reference	251
9.41.1	Constructor & Destructor Documentation	251
9.41.1.1	NodeMap	251
9.41.1.2	~NodeMap	251
9.41.2	Member Function Documentation	252
9.41.2.1	_GetDeviceName	252
9.41.2.2	_GetNode	252
9.41.2.3	_GetNodes	252
9.41.2.4	_InvalidateNodes	252
9.41.2.5	_Poll	252
9.42	PGMOption Struct Reference	252
9.42.1	Detailed Description	253
9.42.2	Constructor & Destructor Documentation	253
9.42.2.1	PGMOption	253
9.42.3	Member Data Documentation	253
9.42.3.1	binaryFile	253
9.42.3.2	reserved	253
9.43	PGRGuid Class Reference	253

9.43.1 Detailed Description . . . . .	253
9.43.2 Constructor & Destructor Documentation . . . . .	254
9.43.2.1 PGRGuid . . . . .	254
9.43.3 Member Function Documentation . . . . .	254
9.43.3.1 operator!= . . . . .	254
9.43.3.2 operator== . . . . .	254
9.43.4 Member Data Documentation . . . . .	254
9.43.4.1 value . . . . .	254
9.44 PNGOption Struct Reference . . . . .	254
9.44.1 Detailed Description . . . . .	255
9.44.2 Constructor & Destructor Documentation . . . . .	255
9.44.2.1 PNGOption . . . . .	255
9.44.3 Member Data Documentation . . . . .	255
9.44.3.1 compressionLevel . . . . .	255
9.44.3.2 interlaced . . . . .	255
9.44.3.3 reserved . . . . .	255
9.45 PPMOption Struct Reference . . . . .	255
9.45.1 Detailed Description . . . . .	256
9.45.2 Constructor & Destructor Documentation . . . . .	256
9.45.2.1 PPMOption . . . . .	256
9.45.3 Member Data Documentation . . . . .	256
9.45.3.1 binaryFile . . . . .	256
9.45.3.2 reserved . . . . .	256
9.46 Property Struct Reference . . . . .	256
9.46.1 Detailed Description . . . . .	257
9.46.2 Constructor & Destructor Documentation . . . . .	257
9.46.2.1 Property . . . . .	257
9.46.2.2 Property . . . . .	257
9.46.3 Member Data Documentation . . . . .	257
9.46.3.1 absControl . . . . .	257
9.46.3.2 absValue . . . . .	257
9.46.3.3 autoManualMode . . . . .	258
9.46.3.4 onePush . . . . .	258
9.46.3.5 onOff . . . . .	258

9.46.3.6	present	258
9.46.3.7	reserved	258
9.46.3.8	type	258
9.46.3.9	valueA	258
9.46.3.10	valueB	258
9.47	PropertyInfo Struct Reference	258
9.47.1	Detailed Description	259
9.47.2	Constructor & Destructor Documentation	260
9.47.2.1	PropertyInfo	260
9.47.2.2	PropertyInfo	260
9.47.3	Member Data Documentation	260
9.47.3.1	absMax	260
9.47.3.2	absMin	260
9.47.3.3	absValSupported	260
9.47.3.4	autoSupported	260
9.47.3.5	manualSupported	260
9.47.3.6	max	260
9.47.3.7	min	260
9.47.3.8	onePushSupported	260
9.47.3.9	onOffSupported	260
9.47.3.10	present	261
9.47.3.11	pUnitAbbr	261
9.47.3.12	pUnits	261
9.47.3.13	readOutSupported	261
9.47.3.14	reserved	261
9.47.3.15	type	261
9.48	PtGreyVideoEncoderOptions Struct Reference	261
9.48.1	Detailed Description	262
9.48.2	Member Data Documentation	262
9.48.2.1	BR	262
9.48.2.2	dstHeight	262
9.48.2.3	dstWidth	262
9.48.2.4	filename	262
9.48.2.5	fmt	262

9.48.2.6	FPS_DEN	262
9.48.2.7	FPS_NUM	262
9.48.2.8	frames	262
9.48.2.9	GOB	262
9.48.2.10	mode	262
9.48.2.11	srcHeight	262
9.48.2.12	srcWidth	262
9.48.2.13	stride	262
9.49	StrobeControl Struct Reference	262
9.49.1	Detailed Description	263
9.49.2	Constructor & Destructor Documentation	263
9.49.2.1	StrobeControl	263
9.49.3	Member Data Documentation	263
9.49.3.1	delay	263
9.49.3.2	duration	263
9.49.3.3	onOff	263
9.49.3.4	polarity	264
9.49.3.5	reserved	264
9.49.3.6	source	264
9.50	StrobeInfo Struct Reference	264
9.50.1	Detailed Description	265
9.50.2	Constructor & Destructor Documentation	265
9.50.2.1	StrobeInfo	265
9.50.3	Member Data Documentation	265
9.50.3.1	maxValue	265
9.50.3.2	minValue	265
9.50.3.3	onOffSupported	265
9.50.3.4	polaritySupported	265
9.50.3.5	present	265
9.50.3.6	readOutSupported	265
9.50.3.7	reserved	265
9.50.3.8	source	265
9.51	SyncManager Class Reference	266
9.51.1	Constructor & Destructor Documentation	266

9.51.1.1	<a href="#">SyncManager</a>	266
9.51.1.2	<a href="#">~SyncManager</a>	266
9.51.2	<a href="#">Member Function Documentation</a>	266
9.51.2.1	<a href="#">DisableCrossPCsynchronization</a>	266
9.51.2.2	<a href="#">EnableCrossPCsynchronization</a>	266
9.51.2.3	<a href="#">GetSyncStatus</a>	266
9.51.2.4	<a href="#">GetTimeSinceSynced</a>	266
9.51.2.5	<a href="#">IsTimingBusConnected</a>	266
9.51.2.6	<a href="#">QueryCrossPCsynchronizationSetting</a>	266
9.51.2.7	<a href="#">RescanMasterTimingBus</a>	266
9.51.2.8	<a href="#">Start</a>	266
9.51.2.9	<a href="#">Stop</a>	266
9.52	<a href="#">SystemInfo Struct Reference</a>	267
9.52.1	<a href="#">Detailed Description</a>	267
9.52.2	<a href="#">Member Data Documentation</a>	267
9.52.2.1	<a href="#">byteOrder</a>	267
9.52.2.2	<a href="#">cpuDescription</a>	268
9.52.2.3	<a href="#">driverList</a>	268
9.52.2.4	<a href="#">gpuDescription</a>	268
9.52.2.5	<a href="#">libraryList</a>	268
9.52.2.6	<a href="#">numCpuCores</a>	268
9.52.2.7	<a href="#">osDescription</a>	268
9.52.2.8	<a href="#">osType</a>	268
9.52.2.9	<a href="#">reserved</a>	268
9.52.2.10	<a href="#">screenHeight</a>	268
9.52.2.11	<a href="#">screenWidth</a>	268
9.52.2.12	<a href="#">sysMemSize</a>	269
9.53	<a href="#">TIFFOption Struct Reference</a>	269
9.53.1	<a href="#">Detailed Description</a>	269
9.53.2	<a href="#">Member Enumeration Documentation</a>	269
9.53.2.1	<a href="#">CompressionMethod</a>	269
9.53.3	<a href="#">Constructor &amp; Destructor Documentation</a>	270
9.53.3.1	<a href="#">TIFFOption</a>	270
9.53.4	<a href="#">Member Data Documentation</a>	270



9.53.4.1	compression	270
9.53.4.2	reserved	270
9.54	TimeStamp Struct Reference	270
9.54.1	Detailed Description	271
9.54.2	Constructor & Destructor Documentation	271
9.54.2.1	TimeStamp	271
9.54.3	Member Data Documentation	271
9.54.3.1	cycleCount	271
9.54.3.2	cycleOffset	271
9.54.3.3	cycleSeconds	271
9.54.3.4	microSeconds	271
9.54.3.5	reserved	271
9.54.3.6	seconds	271
9.55	TopologyNode Class Reference	272
9.55.1	Detailed Description	273
9.55.2	Member Enumeration Documentation	273
9.55.2.1	NodeType	273
9.55.2.2	PortType	273
9.55.3	Constructor & Destructor Documentation	273
9.55.3.1	TopologyNode	273
9.55.3.2	TopologyNode	274
9.55.3.3	~TopologyNode	274
9.55.3.4	TopologyNode	274
9.55.4	Member Function Documentation	274
9.55.4.1	AddChild	274
9.55.4.2	AddPortType	274
9.55.4.3	AssignGuidToNode	274
9.55.4.4	AssignGuidToNode	275
9.55.4.5	GetChild	275
9.55.4.6	GetDeviceId	275
9.55.4.7	GetGuid	276
9.55.4.8	GetInterfaceType	276
9.55.4.9	GetNodeType	276
9.55.4.10	GetNumChildren	276

9.55.4.11	GetNumPorts	276
9.55.4.12	GetPortType	276
9.55.4.13	operator=	277
9.56	TriggerMode Struct Reference	277
9.56.1	Detailed Description	278
9.56.2	Constructor & Destructor Documentation	278
9.56.2.1	TriggerMode	278
9.56.3	Member Data Documentation	278
9.56.3.1	mode	278
9.56.3.2	onOff	278
9.56.3.3	parameter	278
9.56.3.4	polarity	278
9.56.3.5	reserved	278
9.56.3.6	source	278
9.57	TriggerModelInfo Struct Reference	278
9.57.1	Detailed Description	279
9.57.2	Constructor & Destructor Documentation	279
9.57.2.1	TriggerModelInfo	279
9.57.3	Member Data Documentation	279
9.57.3.1	modeMask	279
9.57.3.2	onOffSupported	279
9.57.3.3	polaritySupported	280
9.57.3.4	present	280
9.57.3.5	readOutSupported	280
9.57.3.6	reserved	280
9.57.3.7	softwareTriggerSupported	280
9.57.3.8	sourceMask	280
9.57.3.9	valueReadable	280
9.58	Utilities Class Reference	280
9.58.1	Detailed Description	281
9.58.2	Member Function Documentation	281
9.58.2.1	CheckDriver	281
9.58.2.2	GetDriverDeviceName	281
9.58.2.3	GetLibraryVersion	282

9.58.2.4	GetSystemInfo	282
9.58.2.5	LaunchBrowser	282
9.58.2.6	LaunchCommand	283
9.58.2.7	LaunchCommandAsync	283
9.58.2.8	LaunchHelp	283
<b>10</b>	<b>File Documentation</b>	<b>285</b>
10.1	AVIRecorder.h File Reference	285
10.2	BusManager.h File Reference	285
10.3	Camera.h File Reference	286
10.4	CameraBase.h File Reference	286
10.5	Error.h File Reference	286
10.6	FlyCapture2.h File Reference	287
10.7	FlyCapture2Defs.h File Reference	287
10.7.1	Define Documentation	292
10.7.1.1	FULL_32BIT_VALUE	292
10.7.1.2	NULL	292
10.8	FlyCapture2GUI.h File Reference	292
10.9	FlyCapture2Platform.h File Reference	292
10.9.1	Define Documentation	292
10.9.1.1	FLYCAPTURE2_API	292
10.9.1.2	FLYCAPTURE2_LOCAL	292
10.10	FlyCapture3ApiGuiWrapper.h File Reference	292
10.10.1	Define Documentation	293
10.10.1.1	WRAPPER_API	293
10.11	GCCamera.h File Reference	293
10.12	GigECamera.h File Reference	293
10.13	Image.h File Reference	294
10.14	ImageStatistics.h File Reference	294
10.15	Internal.h File Reference	294
10.16	Licensing.dox File Reference	294
10.17	MultiSyncLibrary.h File Reference	294
10.18	MultiSyncLibraryDefs.h File Reference	294
10.19	MultiSyncLibraryPlatform.h File Reference	295

10.19.1 Define Documentation . . . . .	295
10.19.1.1 MULTISYNCLIBRARY_API . . . . .	295
10.19.1.2 MULTISYNCLIBRARY_LOCAL . . . . .	295
10.20NodeMap.h File Reference . . . . .	295
10.21PtGreyVideoEncoder.h File Reference . . . . .	296
10.21.1 Detailed Description . . . . .	297
10.21.2 Define Documentation . . . . .	297
10.21.2.1 FALSE . . . . .	297
10.21.2.2 FILENAME_MAX . . . . .	297
10.21.2.3 PTGREY_BR_MIN . . . . .	297
10.21.2.4 PTGREY_BUF_LEN . . . . .	297
10.21.2.5 TRUE . . . . .	297
10.21.3 Typedef Documentation . . . . .	297
10.21.3.1 BOOL . . . . .	297
10.21.4 Enumeration Type Documentation . . . . .	297
10.21.4.1 PtGreyVideoCodecFormat . . . . .	297
10.21.4.2 PtGreyVideoScalerMode . . . . .	297
10.21.5 Function Documentation . . . . .	298
10.21.5.1 PtGreyCreateVideoEncoder . . . . .	298
10.21.5.2 PtGreyDestroyVideoEncoder . . . . .	298
10.21.5.3 PtGreyWriteFrame . . . . .	298
10.22PtGreyVideoEncoderPlatform.h File Reference . . . . .	298
10.22.1 Define Documentation . . . . .	298
10.22.1.1 PTGREY_VIDEO_ENCODER_API . . . . .	298
10.23TopologyNode.h File Reference . . . . .	298
10.24Utilities.h File Reference . . . . .	298

## **Chapter 1**

# **Software Licensing Information**

Component	License
<a href="#">FlyCapture2</a>	Copyright © 2017 FLIR Integrated Imaging Solutions, Inc. All Rights Reserved. This software is the confidential and proprietary information of FLIR Integrated Imaging Solutions, Inc. ("Confidential Information"). You shall not disclose such Confidential Information and shall use it only in accordance with the terms of the license agreement you entered into with FLIR Integrated Imaging Solutions, Inc. (FLIR). FLIR MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THE SOFTWARE, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. FLIR SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.
AdapterList	The Code Project Open License (CPOL) <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>
Generated on Wed Jul 5 2017 12:30:18 for FlyCapture2 by Doxygen	

Table 1.1: License table

# Chapter 2

## Module Index

### 2.1 Modules

Here is a list of all modules:

Global constants . . . . .	15
Enumerations . . . . .	16
GigE specific enumerations . . . . .	29
Structures . . . . .	30
GigE specific structures . . . . .	33
IIDC specific structures . . . . .	34
Image saving structures. . . . .	35





# Chapter 3

## Namespace Index

### 3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

<a href="#">FlyCap3CameraControl</a>	37
<a href="#">FlyCapture2</a>	37
<a href="#">MultiSyncLibrary</a>	44



## Chapter 4

# Class Index

### 4.1 Class Hierarchy

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

AVIOption . . . . .	47
AVIRecorder . . . . .	48
BMPOption . . . . .	51
BusManager . . . . .	51
CameraBase . . . . .	91
Camera . . . . .	61
GCCamera . . . . .	151
GigECamera . . . . .	176
CameraControlDlg . . . . .	118
CameraInfo . . . . .	120
CameraSelectionDlg . . . . .	125
CameraStats . . . . .	127
ConfigROM . . . . .	129
EmbeddedImageInfo . . . . .	132
EmbeddedImageInfoProperty . . . . .	133
Error . . . . .	134
EventCallbackData . . . . .	138
EventOptions . . . . .	139
FC2Config . . . . .	140
FC2Version . . . . .	143
FlyCapture3ApiGuiWrapper . . . . .	144
Format7ImageSettings . . . . .	145
Format7Info . . . . .	147
Format7PacketInfo . . . . .	149
GigEConfig . . . . .	210
GigEImageSettings . . . . .	211
GigEImageSettingsInfo . . . . .	213
GigEProperty . . . . .	214
GigEStreamChannel . . . . .	216

H264Option . . . . .	218
Image . . . . .	219
ImageMetadata . . . . .	235
ImageStatistics . . . . .	237
Internal . . . . .	244
IPAddress . . . . .	244
JPEGOption . . . . .	245
JPG2Option . . . . .	246
LUTData . . . . .	247
MACAddress . . . . .	249
MJPGOption . . . . .	250
NodeMap . . . . .	251
PGMOption . . . . .	252
PGRGuid . . . . .	253
PNGOption . . . . .	254
PPMOption . . . . .	255
Property . . . . .	256
PropertyInfo . . . . .	258
PtGreyVideoEncoderOptions . . . . .	261
StrobeControl . . . . .	262
StrobeInfo . . . . .	264
SyncManager . . . . .	266
SystemInfo . . . . .	267
TIFFOption . . . . .	269
TimeStamp . . . . .	270
TopologyNode . . . . .	272
TriggerMode . . . . .	277
TriggerModelInfo . . . . .	278
Utilities . . . . .	280

## 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 . . . . .	47
<a href="#">AVIRecorder</a>	Functionality for the user to record images to an AVI file . . . . .	48
<a href="#">BMPOption</a>	Options for saving Bitmap image . . . . .	51
<a href="#">BusManager</a>	Functionality for the user to get an <a href="#">PGRGuid</a> for a desired camera or device easily . . . . .	51
<a href="#">Camera</a>	The <a href="#">Camera</a> object represents a physical camera that uses the IIDC register set . . . . .	61
<a href="#">CameraBase</a>	Abstract base class that defines a general interface to a camera . .	91
<a href="#">CameraControlDlg</a>	The <a href="#">CameraControlDlg</a> object represents a dialog that provides a graphical interface to a specified camera . . . . .	118
<a href="#">CameraInfo</a>	<a href="#">Camera</a> information . . . . .	120
<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 . . . . .	125
<a href="#">CameraStats</a>	<a href="#">Camera</a> diagnostic information . . . . .	127
<a href="#">ConfigROM</a>	<a href="#">Camera</a> configuration ROM . . . . .	129
<a href="#">EmbeddedImageInfo</a>	Properties of the possible embedded image information . . . . .	132

<a href="#">EmbeddedImageInfoProperty</a>	
Properties of a single embedded image info property . . . . .	133
<a href="#">Error</a>	
The <a href="#">Error</a> object represents an error that is returned from the library	134
<a href="#">EventCallbackData</a> . . . . .	138
<a href="#">EventOptions</a>	
Options for enabling device event registration . . . . .	139
<a href="#">FC2Config</a>	
Configuration for a camera . . . . .	140
<a href="#">FC2Version</a>	
The current version of the library . . . . .	143
<a href="#">FlyCapture3ApiGuiWrapper</a> . . . . .	144
<a href="#">Format7ImageSettings</a>	
Format 7 image settings . . . . .	145
<a href="#">Format7Info</a>	
Format 7 information for a single mode . . . . .	147
<a href="#">Format7PacketInfo</a>	
Format 7 packet information . . . . .	149
<a href="#">GCCamera</a> . . . . .	151
<a href="#">GigECamera</a>	
The <a href="#">GigECamera</a> object represents a physical Gigabit Ethernet camera . . . . .	176
<a href="#">GigEConfig</a>	
Configuration for a GigE camera . . . . .	210
<a href="#">GigEImageSettings</a>	
Image settings for a GigE camera . . . . .	211
<a href="#">GigEImageSettingsInfo</a>	
Format 7 information for a single mode . . . . .	213
<a href="#">GigEProperty</a>	
A GigE property . . . . .	214
<a href="#">GigEStreamChannel</a>	
Information about a single GigE stream channel . . . . .	216
<a href="#">H264Option</a>	
Options for saving H264 files . . . . .	218
<a href="#">Image</a>	
Used to retrieve images from a camera, convert between multiple pixel formats and save images to disk . . . . .	219
<a href="#">ImageMetadata</a>	
Metadata related to an image . . . . .	235
<a href="#">ImageStatistics</a>	
The <a href="#">ImageStatistics</a> object represents image statistics for an image .	237
<a href="#">Internal</a> . . . . .	244
<a href="#">IPAddress</a>	
IPv4 address . . . . .	244
<a href="#">JPEGOption</a>	
Options for saving JPEG image . . . . .	245
<a href="#">JPG2Option</a>	
Options for saving JPEG2000 image . . . . .	246
<a href="#">LUTData</a>	
Information about the camera's look up table . . . . .	247

<a href="#">MACAddress</a>	
MAC address . . . . .	249
<a href="#">MJPGOption</a>	
Options for saving MJPG files . . . . .	250
<a href="#">NodeMap</a> . . . . .	251
<a href="#">PGMOption</a>	
Options for saving PGM images . . . . .	252
<a href="#">PGRGuid</a>	
A GUID to the camera . . . . .	253
<a href="#">PNGOption</a>	
Options for saving PNG images . . . . .	254
<a href="#">PPMOption</a>	
Options for saving PPM images . . . . .	255
<a href="#">Property</a>	
A specific camera property . . . . .	256
<a href="#">PropertyInfo</a>	
Information about a specific camera property . . . . .	258
<a href="#">PtGreyVideoEncoderOptions</a>	
Encoding options provided by the user . . . . .	261
<a href="#">StrobeControl</a>	
A camera strobe . . . . .	262
<a href="#">StrobeInfo</a>	
A camera strobe property . . . . .	264
<a href="#">SyncManager</a> . . . . .	266
<a href="#">SystemInfo</a>	
Description of the system . . . . .	267
<a href="#">TIFFOption</a>	
Options for saving TIFF images . . . . .	269
<a href="#">TimeStamp</a>	
Timestamp information . . . . .	270
<a href="#">TopologyNode</a>	
Topology information that can be used to generate a tree structure of all cameras and devices connected to a computer . . . . .	272
<a href="#">TriggerMode</a>	
A camera trigger . . . . .	277
<a href="#">TriggerModelInfo</a>	
Information about a camera trigger property . . . . .	278
<a href="#">Utilities</a>	
The Utility class is generally used to query for general system infor- mation such as operating system, available memory etc . . . . .	280





## Chapter 6

# File Index

### 6.1 File List

Here is a list of all files with brief descriptions:

<a href="#">AVIRecorder.h</a>	285
<a href="#">BusManager.h</a>	285
<a href="#">Camera.h</a>	286
<a href="#">CameraBase.h</a>	286
<a href="#">Error.h</a>	286
<a href="#">FlyCapture2.h</a>	287
<a href="#">FlyCapture2Defs.h</a>	287
<a href="#">FlyCapture2GUI.h</a>	292
<a href="#">FlyCapture2Platform.h</a>	292
<a href="#">FlyCapture3ApiGuiWrapper.h</a>	292
<a href="#">GCCamera.h</a>	293
<a href="#">GigECamera.h</a>	293
<a href="#">Image.h</a>	294
<a href="#">ImageStatistics.h</a>	294
<a href="#">Internal.h</a>	294
<a href="#">MultiSyncLibrary.h</a>	294
<a href="#">MultiSyncLibraryDefs.h</a>	294
<a href="#">MultiSyncLibraryPlatform.h</a>	295
<a href="#">NodeMap.h</a>	295
<a href="#">PtGreyVideoEncoder.h</a>	
The API functions within this library are re-entrant, as each video encoding session is expected to use a unique handle acquire via	
<a href="#">PtGreyCreateVideoEncoder()</a>	296
<a href="#">PtGreyVideoEncoderPlatform.h</a>	298
<a href="#">TopologyNode.h</a>	298
<a href="#">Utilities.h</a>	298



## 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 Variable Documentation

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

The maximum number of ports one device can have.

7.1.1.2 `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_100BA`

```
SE_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_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`, `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 Enumeration Type Documentation

### 7.2.1.1 enum BandwidthAllocation

Bandwidth allocation options for 1394 devices.

Enumerator:

**BANDWIDTH\_ALLOCATION\_OFF** Do not allocate bandwidth.

**BANDWIDTH\_ALLOCATION\_ON** Allocate bandwidth. This is the default setting.

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

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

**BANDWIDTH\_ALLOCATION\_FORCE\_32BITS**

## 7.2.1.2 enum BayerTileFormat

Bayer tile formats.

Enumerator:

**NONE** No bayer tile format.  
**RGGB** Red-Green-Green-Blue.  
**GRBG** Green-Red-Blue-Green.  
**GBRG** Green-Blue-Red-Green.  
**BGGR** Blue-Green-Green-Red.  
**BT\_FORCE\_32BITS**

## 7.2.1.3 enum BusCallbackType

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

Enumerator:

**BUS\_RESET** Register for all bus events.  
**ARRIVAL** Register for arrivals only.  
**REMOVAL** Register for removals only.  
**CALLBACK\_TYPE\_FORCE\_32BITS**

## 7.2.1.4 enum BusSpeed

Bus speeds.

Enumerator:

**BUSSPEED\_S100** 100Mbps/sec.  
**BUSSPEED\_S200** 200Mbps/sec.  
**BUSSPEED\_S400** 400Mbps/sec.  
**BUSSPEED\_S480** 480Mbps/sec. Only for 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**

#### 7.2.1.5 enum ColorProcessingAlgorithm

Color processing algorithms.

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

Enumerator:

**DEFAULT** Default method.

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

**NEAREST\_NEIGHBOR** Fastest but lowest quality. Equivalent to FLYCAPTURE-NEAREST\_NEIGHBOR\_FAST in FlyCapture.

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

**HQ\_LINEAR** 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**

#### 7.2.1.6 enum DriverType

Types of low level drivers that flycapture uses.

Enumerator:

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

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

**DRIVER\_1394\_JUUU** 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** PGRUsCam.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**

#### 7.2.1.7 enum ErrorType

The error types returned by functions.

Enumerator:

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

**PGRERROR\_IIDC\_FAILED** IIDC failure.  
**PGRERROR\_STROBE\_FAILED** Strobe failure.  
**PGRERROR\_TRIGGER\_FAILED** Trigger failure.  
**PGRERROR\_PROPERTY\_FAILED** [Property](#) failure.  
**PGRERROR\_PROPERTY\_NOT\_PRESENT** [Property](#) is not present.  
**PGRERROR\_REGISTER\_FAILED** Register access failed.  
**PGRERROR\_READ\_REGISTER\_FAILED** Register read failed.  
**PGRERROR\_WRITE\_REGISTER\_FAILED** Register write failed.  
**PGRERROR\_ISOCH\_FAILED** Isochronous failure.  
**PGRERROR\_ISOCH\_ALREADY\_STARTED** Isochronous transfer has already been started.  
**PGRERROR\_ISOCH\_NOT\_STARTED** Isochronous transfer has not been started.  
**PGRERROR\_ISOCH\_START\_FAILED** Isochronous start failed.  
**PGRERROR\_ISOCH\_RETRIEVE\_BUFFER\_FAILED** Isochronous retrieve buffer failed.  
**PGRERROR\_ISOCH\_STOP\_FAILED** Isochronous stop failed.  
**PGRERROR\_ISOCH\_SYNC\_FAILED** Isochronous image synchronization failed.  
**PGRERROR\_ISOCH\_BANDWIDTH\_EXCEEDED** Isochronous bandwidth exceeded.  
**PGRERROR\_IMAGE\_CONVERSION\_FAILED** [Image](#) conversion failed.  
**PGRERROR\_IMAGE\_LIBRARY\_FAILURE** [Image](#) library failure.  
**PGRERROR\_BUFFER\_TOO\_SMALL** Buffer is too small.  
**PGRERROR\_IMAGE\_CONSISTENCY\_ERROR** There is an image consistency error.  
**PGRERROR\_INCOMPATIBLE\_DRIVER** The installed driver is not compatible with the library.  
**PGRERROR\_FORCE\_32BITS**

#### 7.2.1.8 enum FrameRate

Frame rates in frames per second.

Enumerator:

**FRAMERATE\_1\_875** 1.875 fps.  
**FRAMERATE\_3\_75** 3.75 fps.  
**FRAMERATE\_7\_5** 7.5 fps.  
**FRAMERATE\_15** 15 fps.  
**FRAMERATE\_30** 30 fps.

**FRAMERATE\_60** 60 fps.

**FRAMERATE\_120** 120 fps.

**FRAMERATE\_240** 240 fps.

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

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

**FRAMERATE\_FORCE\_32BITS**

#### 7.2.1.9 enum GrabMode

The grab strategy employed during image transfer.

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

Enumerator:

**DROP\_FRAMES** Grabs the newest image in the user buffer each time the - RetrieveBuffer() function is called. Older images are dropped instead of accumulating in the user buffer. Grabbing blocks if the camera has not finished transmitting the next available image. If the camera is transmitting images faster than the application can grab them, images may be dropped and only the most recent image is stored for grabbing. Note that this mode is the equivalent of flycaptureLockLatest in earlier versions of the FlyCapture SDK.

**BUFFER\_FRAMES** Images accumulate in the user buffer, and the oldest image is grabbed for handling before being discarded. This member can be used to guarantee that each image is seen. However, image processing time must not exceed transmission time from the camera to the buffer. Grabbing blocks if the camera has not finished transmitting the next available image. The buffer size is controlled by the numBuffers parameter in the [FC2Config](#) struct. Note that this mode is the equivalent of flycaptureLockNext in earlier versions of the FlyCapture SDK.

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

**GRAB\_MODE\_FORCE\_32BITS**

#### 7.2.1.10 enum GrabTimeout

Timeout options for grabbing images.

Enumerator:

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

**TIMEOUT\_INFINITE** Wait indefinitely.

**TIMEOUT\_UNSPECIFIED** Unspecified timeout setting.

**GRAB\_TIMEOUT\_FORCE\_32BITS**

### 7.2.1.11 enum ImageFileFormat

File formats to be used for saving images to disk.

Enumerator:

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

### 7.2.1.12 enum InterfaceType

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

Enumerator:

**INTERFACE\_IEEE1394** IEEE-1394 (Includes 1394a and 1394b).  
**INTERFACE\_USB2** USB 2.0.  
**INTERFACE\_USB3** USB 3.0.  
**INTERFACE\_GIGE** GigE.  
**INTERFACE\_UNKNOWN** Unknown interface.  
**INTERFACE\_TYPE\_FORCE\_32BITS**

### 7.2.1.13 enum Mode

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

Enumerator:

**MODE\_0**  
**MODE\_1**  
**MODE\_2**  
**MODE\_3**  
**MODE\_4**  
**MODE\_5**

**MODE\_6**  
**MODE\_7**  
**MODE\_8**  
**MODE\_9**  
**MODE\_10**  
**MODE\_11**  
**MODE\_12**  
**MODE\_13**  
**MODE\_14**  
**MODE\_15**  
**MODE\_16**  
**MODE\_17**  
**MODE\_18**  
**MODE\_19**  
**MODE\_20**  
**MODE\_21**  
**MODE\_22**  
**MODE\_23**  
**MODE\_24**  
**MODE\_25**  
**MODE\_26**  
**MODE\_27**  
**MODE\_28**  
**MODE\_29**  
**MODE\_30**  
**MODE\_31**  
**NUM\_MODES** Number of modes.  
**MODE\_FORCE\_32BITS**

#### 7.2.1.14 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.

## 7.2.1.15 enum PixelFormat

Pixel formats available for Format7 modes.

Enumerator:

**PIXEL\_FORMAT\_MONO8** 8 bits of mono information.  
**PIXEL\_FORMAT\_411YUV8** YUV 4:1:1.  
**PIXEL\_FORMAT\_422YUV8** YUV 4:2:2.  
**PIXEL\_FORMAT\_444YUV8** YUV 4:4:4.  
**PIXEL\_FORMAT\_RGB8** R = G = B = 8 bits.  
**PIXEL\_FORMAT\_MONO16** 16 bits of mono information.  
**PIXEL\_FORMAT\_RGB16** R = G = B = 16 bits.  
**PIXEL\_FORMAT\_S\_MONO16** 16 bits of signed mono information.  
**PIXEL\_FORMAT\_S\_RGB16** R = G = B = 16 bits signed.  
**PIXEL\_FORMAT\_RAW8** 8 bit raw data output of sensor.  
**PIXEL\_FORMAT\_RAW16** 16 bit raw data output of sensor.  
**PIXEL\_FORMAT\_MONO12** 12 bits of mono information.  
**PIXEL\_FORMAT\_RAW12** 12 bit raw data output of sensor.  
**PIXEL\_FORMAT\_BGR** 24 bit BGR.  
**PIXEL\_FORMAT\_BGRU** 32 bit BGRU.  
**PIXEL\_FORMAT\_RGB** 24 bit RGB.  
**PIXEL\_FORMAT\_RGBU** 32 bit RGBU.  
**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.

## 7.2.1.16 enum PropertyType

[Camera](#) properties.

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

Enumerator:

**BRIGHTNESS** Brightness.  
**AUTO\_EXPOSURE** Auto exposure.  
**SHARPNESS** Sharpness.  
**WHITE\_BALANCE** White balance.  
**HUE** Hue.

**SATURATION** Saturation.  
**GAMMA** Gamma.  
**IRIS** Iris.  
**FOCUS** Focus.  
**ZOOM** Zoom.  
**PAN** Pan.  
**TILT** Tilt.  
**SHUTTER** Shutter.  
**GAIN** Gain.  
**TRIGGER\_MODE** Trigger mode.  
**TRIGGER\_DELAY** Trigger delay.  
**FRAME\_RATE** Frame rate.  
**TEMPERATURE** Temperature.  
**UNSPECIFIED\_PROPERTY\_TYPE** Unspecified property type.  
**PROPERTY\_TYPE\_FORCE\_32BITS**

#### 7.2.1.17 enum VideoMode

DCAM video modes.

Enumerator:

**VIDEOMODE\_160x120YUV444** 160x120 YUV444.  
**VIDEOMODE\_320x240YUV422** 320x240 YUV422.  
**VIDEOMODE\_640x480YUV411** 640x480 YUV411.  
**VIDEOMODE\_640x480YUV422** 640x480 YUV422.  
**VIDEOMODE\_640x480RGB** 640x480 24-bit RGB.  
**VIDEOMODE\_640x480Y8** 640x480 8-bit.  
**VIDEOMODE\_640x480Y16** 640x480 16-bit.  
**VIDEOMODE\_800x600YUV422** 800x600 YUV422.  
**VIDEOMODE\_800x600RGB** 800x600 RGB.  
**VIDEOMODE\_800x600Y8** 800x600 8-bit.  
**VIDEOMODE\_800x600Y16** 800x600 16-bit.  
**VIDEOMODE\_1024x768YUV422** 1024x768 YUV422.  
**VIDEOMODE\_1024x768RGB** 1024x768 RGB.  
**VIDEOMODE\_1024x768Y8** 1024x768 8-bit.  
**VIDEOMODE\_1024x768Y16** 1024x768 16-bit.  
**VIDEOMODE\_1280x960YUV422** 1280x960 YUV422.  
**VIDEOMODE\_1280x960RGB** 1280x960 RGB.

**VIDEOMODE\_1280x960Y8** 1280x960 8-bit.  
**VIDEOMODE\_1280x960Y16** 1280x960 16-bit.  
**VIDEOMODE\_1600x1200YUV422** 1600x1200 YUV422.  
**VIDEOMODE\_1600x1200RGB** 1600x1200 RGB.  
**VIDEOMODE\_1600x1200Y8** 1600x1200 8-bit.  
**VIDEOMODE\_1600x1200Y16** 1600x1200 16-bit.  
**VIDEOMODE\_FORMAT7** Custom video mode for Format7 functionality.  
**NUM\_VIDEOMODES** Number of possible video modes.  
**VIDEOMODE\_FORCE\_32BITS**



## 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 enum `GigEPropertyType`

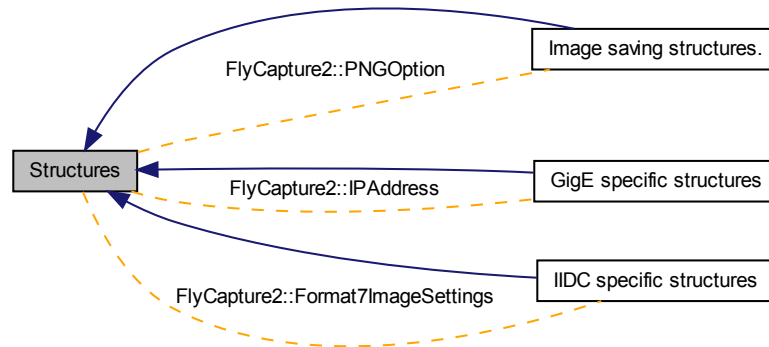
Possible properties that can be queried from the camera.

Enumerator:

**`HEARTBEAT`**  
**`HEARTBEAT_TIMEOUT`**  
**`PACKET_SIZE`**  
**`PACKET_DELAY`**

## 7.4 Structures

Collaboration diagram for Structures:



### 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.*

## Modules

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

## Typedefs

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

### 7.4.1 Typedef Documentation

#### 7.4.1.1 typedef Property TriggerDelay

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

#### 7.4.1.2 `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 [MJPGOption](#)  
*Options for saving MJPG files.*
- struct [H264Option](#)  
*Options for saving H264 files.*
- struct [AVIOption](#)  
*Options for saving AVI files.*
- 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 `typedef void(* CameraEventCallback)(void *data)`



## Chapter 8

# Namespace Documentation

### 8.1 FlyCap3CameraControl Namespace Reference

#### Classes

- class [FlyCapture3ApiGuiWrapper](#)

### 8.2 FlyCapture2 Namespace Reference

#### Classes

- class [AVIRecorder](#)  
*The [AVIRecorder](#) class provides the functionality for the user to record images to an AVI file.*
- class [BusManager](#)  
*The [BusManager](#) class provides the functionality for the user to get an [PGRGuid](#) for a desired camera or device easily.*
- class [Camera](#)  
*The [Camera](#) object represents a physical camera that uses the IIDC register set.*
- class [CameraBase](#)  
*The [CameraBase](#) class is an abstract base class that defines a general interface to a camera.*
- class [Error](#)  
*The [Error](#) object represents an error that is returned from the library.*
- struct [FC2Version](#)  
*The current version of the library.*
- class [PGRGuid](#)  
*A GUID to the camera.*
- struct [IPAddress](#)  
*IPv4 address.*

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

- Metadata related to an image.*
- struct [LUTData](#)
  - Information about the camera's look up table.*
- struct [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 [MJPGOption](#)
  - Options for saving MJPG files.*
- struct [H264Option](#)
  - Options for saving H264 files.*
- struct [AVIOption](#)
  - Options for saving AVI files.*
- struct [EventOptions](#)
  - Options for enabling device event registration.*
- struct [EventCallbackData](#)
- 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.*
- class [GCCamera](#)
- class [GigECamera](#)
  - The [GigECamera](#) object represents a physical Gigabit Ethernet camera.*
- class [Image](#)
  - The [Image](#) class is used to retrieve images from a camera, convert between multiple pixel formats and save images to disk.*
- class [ImageStatistics](#)
  - The [ImageStatistics](#) object represents image statistics for an image.*
- class [Internal](#)
- class [NodeMap](#)

- class [TopologyNode](#)

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

- struct [SystemInfo](#)

Description of the system.

- class [Utilities](#)

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

## Typedefs

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

Bus event callback function prototype.

- typedef void \* [CallbackHandle](#)

Handle that is returned when registering a callback.

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

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 \*pUser-Data)

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, VIDEOMOD-
E_1280x960RGB, VIDEOMODE_1280x960Y8, VIDEOMODE_1280x960Y16,
VIDEOMODE_1600x1200YUV422, VIDEOMODE_1600x1200RGB, VIDEOM-
ODE_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 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 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 typedef void\* CallbackHandle

Handle that is returned when registering a callback.

It is required when unregistering the callback.

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

[Image](#) event callback function prototype.

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

## 8.2.2 Enumeration Type Documentation

### 8.2.2.1 `enum ByteOrder`

Possible byte orders.

Enumerator:

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

### 8.2.2.2 `enum OSType`

Possible operating systems.

Enumerator:

***WINDOWS\_X86*** All Windows 32-bit variants.  
***WINDOWS\_X64*** All Windows 64-bit variants.  
***LINUX\_X86*** All Linux 32-bit variants.  
***LINUX\_X64*** All Linux 32-bit variants.  
***MAC*** Mac OSX.  
***UNKNOWN\_OS*** Unknown operating system.  
***OSTYPE\_FORCE\_32BITS***

## 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 enum `PGRSyncError`

Enumerator:

**`PGRSyncError_OK`**  
**`PGRSyncError_FAILED`**  
**`PGRSyncError_ALREADY_STARTED`**  
**`PGRSyncError_ALREADY_STOPPED`**  
**`PGRSyncError_CAMERA_NOT_FOUND`**  
**`PGRSyncError_UNKNOWN_ERROR`**

#### 8.3.1.2 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](#) ( ) `[inline]`

#### 9.1.3 Member Data Documentation

##### 9.1.3.1 float [frameRate](#)

Frame rate of the stream.

### 9.1.3.2 unsigned int reserved[256]

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

## 9.2 AVIRecorder Class Reference

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

### Public Member Functions

- [AVIRecorder](#) ()  
*Default constructor.*
- virtual [~AVIRecorder](#) ()  
*Default destructor.*
- virtual [Error AVIOpen](#) (const char \*pFileName, [AVIOption](#) \*pOption)  
*Open an AVI file in preparation for writing Images to disk.*
- virtual [Error AVIOpen](#) (const char \*pFileName, [MJPGOption](#) \*pOption)  
*Open an MJPEG AVI file in preparation for writing Images to disk.*
- virtual [Error AVIOpen](#) (const char \*pFileName, [H264Option](#) \*pOption)  
*Open an H264 MP4 file in preparation for writing Images to disk.*
- virtual [Error AVIAppend](#) ([Image](#) \*pImage)  
*Append an image to the AVI/MP4 file.*
- virtual [Error AVIClose](#) ()  
*Close the AVI/MP4 file.*

### 9.2.1 Detailed Description

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

### 9.2.2 Constructor & Destructor Documentation

#### 9.2.2.1 AVIRecorder ( )

Default constructor.

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

Default destructor.

### 9.2.3 Member Function Documentation

#### 9.2.3.1 virtual Error AVIAppend ( 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.

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

Close the AVI/MP4 file.

##### See also

[AVIOpen\(\)](#)

##### Returns

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

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

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

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

##### Parameters

<i>pFileName</i>	The filename of the AVI file.
<i>pOption</i>	Options to apply to the AVI file.

##### See also

[AVIClose\(\)](#)

##### Returns

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

#### 9.2.3.4 virtual Error AVIOpen ( 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

[AVIClose\(\)](#)  
[MJPGOption](#)

##### Returns

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

#### 9.2.3.5 virtual Error AVIOpen ( const char \* *pFileName*, H264Option \* *pOption* ) [virtual]

Open an H264 MP4 file in preparation for writing Images to disk.

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

##### Parameters

<i>pFileName</i>	The filename of the MP4 file.
<i>pOption</i>	H264 options to apply to the MP4 file.

##### See also

[AVIClose\(\)](#)  
[H264Option](#)

##### Returns

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

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

- [AVIRecorder.h](#)

## 9.3 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.*

#### 9.3.1 Detailed Description

Options for saving Bitmap image.

#### 9.3.2 Constructor & Destructor Documentation

##### 9.3.2.1 [BMPOption](#) ( ) [inline]

#### 9.3.3 Member Data Documentation

##### 9.3.3.1 bool [indexedColor\\_8bit](#)

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

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

## 9.4 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.*

#### 9.4.1 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.

#### 9.4.2 Constructor & Destructor Documentation

##### 9.4.2.1 [BusManager](#) ( )

Default constructor.

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

Default destructor.

#### 9.4.3 Member Function Documentation

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

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

This is useful in situations where 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>gigE-Cameras</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.

#### 9.4.3.2 virtual Error FireBusReset ( PGRGuid \* *pGuid* ) [virtual]

Fire a bus reset.

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

##### Parameters

<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.

#### 9.4.3.3 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.

#### 9.4.3.4 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.

#### 9.4.3.5 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>default-Gateway</i>	Default gateway to set on the camera.

##### Returns

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

#### 9.4.3.6 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.

9.4.3.7 **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.

9.4.3.8 **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>serial-Number</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.

9.4.3.9 **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>pSerial-Number</i>	Serial number of camera.

#### Returns

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

9.4.3.10 **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.

9.4.3.11 **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>pInterface-Type</i>	The interface type of the <a href="#">PGRGuid</a> .

**Returns**

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

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

Gets the number of cameras attached to the PC.

**Parameters**

<i>pNum-Cameras</i>	The number of cameras attached.
---------------------	---------------------------------

**Returns**

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

**9.4.3.13 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>pNum-Devices</i>	The number of devices found.
---------------------	------------------------------

**Returns**

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

**9.4.3.14 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.

9.4.3.15 `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>	PGRGuid of the device to read from.
<i>pValue</i>	Value read from the card register.

#### Returns

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

9.4.3.16 `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>	PGRGuid of the device to read from.
<i>pValue</i>	Value read from the card register.

#### Returns

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

9.4.3.17 `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>	PGRGuid of the camera
<i>pControllable</i>	Indicates whether camera is controllable

#### Returns

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

9.4.3.18 `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.

9.4.3.19 **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>busEvent-Callback</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>pCallback-Handle</i>	Unique callback handle used for unregistering callback.

#### See also

[UnregisterCallback\(\)](#)

#### Returns

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

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

Force a rescan of the buses.

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

#### Returns

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



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

Unregister a callback function.

Parameters

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

See also

[RegisterCallback\(\)](#)

Returns

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

9.4.3.22 **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.

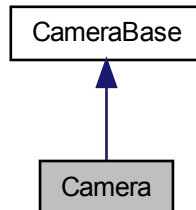
The documentation for this class was generated from the following file:

- [BusManager.h](#)

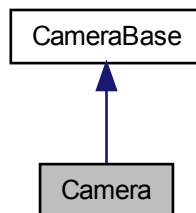
## 9.5 Camera Class Reference

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 \*p-CallbackData=NULL)  
*Sets the callback data to be used on completion of image transfer.*
- virtual [Error StartCapture](#) ([ImageEventCallback](#) callbackFn=NULL, const void \*p-CallbackData=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)

*Returns a text representation of the register value.*

### DCAM Formats

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

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

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

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

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

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

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

## Format7

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

- virtual [Error GetFormat7Info](#) ([Format7Info](#) \*pInfo, bool \*pSupported)  
*Retrieve the availability of Format7 custom image mode and the camera capabilities for the specified Format7 mode.*
- virtual [Error ValidateFormat7Settings](#) (const [Format7ImageSettings](#) \*pImageSettings, bool \*pSettingsAreValid, [Format7PacketInfo](#) \*pPacketInfo)  
*Validates [Format7ImageSettings](#) structure and returns valid packet size information if the image settings are valid.*
- virtual [Error GetFormat7Configuration](#) ([Format7ImageSettings](#) \*pImageSettings, unsigned int \*pPacketSize, float \*pPercentage)  
*Get the current Format7 configuration from the camera.*
- virtual [Error SetFormat7Configuration](#) (const [Format7ImageSettings](#) \*pImageSettings, unsigned int packetSize)  
*Set the current Format7 configuration to the camera.*
- virtual [Error SetFormat7Configuration](#) (const [Format7ImageSettings](#) \*pImageSettings, float percentSpeed)  
*Set the current Format7 configuration to the camera.*

### 9.5.1 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.

### 9.5.2 Constructor & Destructor Documentation

#### 9.5.2.1 [Camera](#) ( )

Default constructor.

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

Default destructor.

### 9.5.3 Member Function Documentation

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

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

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

Implements [CameraBase](#).

#### 9.5.3.2 virtual Error DeregisterAllEvents ( void ) [virtual]

Implements [CameraBase](#).

#### 9.5.3.3 virtual Error DeregisterEvent ( EventOptions \* pOpts ) [virtual]

Implements [CameraBase](#).

#### 9.5.3.4 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](#).

#### 9.5.3.5 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](#).

### 9.5.3.6 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](#).

### 9.5.3.7 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](#).

### 9.5.3.8 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](#).

### 9.5.3.9 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](#).

9.5.3.10 `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](#).

9.5.3.11 `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](#).

9.5.3.12 **virtual Error GetFormat7Configuration ( Format7ImageSettings \* *plmageSettings*, 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>plmageSettings</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.

9.5.3.13 **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.

9.5.3.14 **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](#).

9.5.3.15 **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>pRead-Supported</i>	Whether reading from the bank is supported.
<i>pWrite-Supported</i>	Whether writing to the bank is supported.

#### Returns

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

Implements [CameraBase](#).

9.5.3.16 **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](#).

### 9.5.3.17 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](#).

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

Retrieve the current memory channel from the camera.

## Parameters

<i>pCurrent-Channel</i>	Current memory channel.
-------------------------	-------------------------

## See also

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

## Returns

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

Implements [CameraBase](#).

**9.5.3.19** `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>pNum-Channels</i>	Number of memory channels supported.
----------------------	--------------------------------------

## See also

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

## Returns

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

Implements [CameraBase](#).

**9.5.3.20** `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](#).

**9.5.3.21** `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](#).

**9.5.3.22** `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.

Reimplemented from [CameraBase](#).

9.5.3.23 **virtual Error GetStats ( CameraStats \* *pStats* )** [virtual]

Implements [CameraBase](#).

9.5.3.24 **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](#).

9.5.3.25 **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](#).

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

Retrieve current trigger delay settings from the camera.

## Parameters

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

## See also

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

## Returns

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

Implements [CameraBase](#).

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

Retrieve trigger delay information from the camera.

## Parameters

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

## See also

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

## Returns

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

Implements [CameraBase](#).

9.5.3.28 **virtual Error GetTriggerMode ( TriggerMode \* *pTriggerMode* )** [virtual]

Retrieve current trigger settings from the camera.



## Parameters

<i>pTrigger-Mode</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](#).

**9.5.3.29** `virtual Error GetTriggerModelInfo ( TriggerModelInfo * pTriggerModelInfo )`  
[virtual]

Retrieve trigger information from the camera.

## Parameters

<i>pTrigger-ModelInfo</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](#).

**9.5.3.30** `virtual Error GetVideoModeAndFrameRate ( VideoMode * pVideoMode, FrameRate * pFrameRate )` [virtual]

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

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

## Parameters

<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.

**9.5.3.31** `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.

**9.5.3.32** `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](#).

9.5.3.33 **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](#).

9.5.3.34 **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](#).

9.5.3.35 **virtual Error RegisterAllEvents ( EventOptions \* *pOpts* )** [virtual]

Implements [CameraBase](#).

9.5.3.36 virtual Error RegisterEvent ( EventOptions \* *pOpts* ) [virtual]

Implements [CameraBase](#).

9.5.3.37 virtual Error ResetStats ( ) [virtual]

Implements [CameraBase](#).

9.5.3.38 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](#).

9.5.3.39 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](#).

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

Save the current settings to the specied 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](#).

9.5.3.41 `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](#).

9.5.3.42 `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>pCallback-Data</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](#).

**9.5.3.43** `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](#).

**9.5.3.44** `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](#).

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

Set the current Format7 configuration to the camera.

**Parameters**

<i>pImage- Settings</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.

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

Set the current Format7 configuration to the camera.

**Parameters**

<i>pImage- Settings</i>	<a href="#">Image</a> settings to be written to the camera.
<i>percent- Speed</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.

9.5.3.47 `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](#).

9.5.3.48 `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](#).



9.5.3.49 `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](#).

9.5.3.50 `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>pStrobe-Control</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](#).

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

Set the specified trigger delay settings to the camera.

#### Parameters

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

#### See also

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

#### Returns

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

Implements [CameraBase](#).

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

Set the specified trigger settings to the camera.

#### Parameters

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

#### See also

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

#### Returns

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

Implements [CameraBase](#).

**9.5.3.53** `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](#).

**9.5.3.54** `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.

```
9.5.3.55 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](#).

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

```
9.5.3.57 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](#).

**9.5.3.58** `virtual Error ValidateFormat7Settings ( const Format7ImageSettings *  
 plmageSettings, 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>plmage-Settings</i>	Structure containing the image settings.
<i>pSettings-AreValid</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.

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

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

## Parameters

<i>plmage</i>	Pointer to <a href="#">Image</a> object to store image data.
<i>event-Number</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](#).

9.5.3.60 `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](#).

9.5.3.61 `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](#).

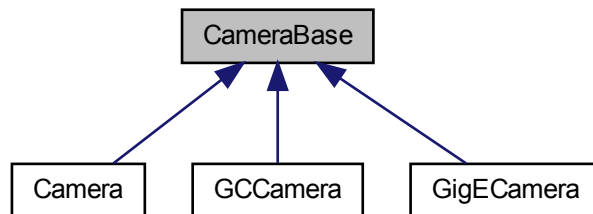
The documentation for this class was generated from the following file:

- [Camera.h](#)

## 9.6 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 \*p-CallbackData=NULL)=0  
*Sets the callback data to be used on completion of image transfer.*
- virtual [Error StartCapture](#) ([ImageEventCallback](#) callbackFn=NULL, const void \*p-CallbackData=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 event-Number)=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 \*\*p-CallbackDataArray=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.*

### 9.6.1 Detailed Description

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

### 9.6.2 Constructor & Destructor Documentation

#### 9.6.2.1 [CameraBase](#) ( ) [inline]

Default constructor.

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

Default destructor.

### 9.6.3 Member Function Documentation

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

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

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

#### Parameters

<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](#).

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

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

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

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

9.6.3.4 `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](#).

**9.6.3.5 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](#).

**9.6.3.6 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](#).

**9.6.3.7 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](#).

#### 9.6.3.8 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](#).

#### 9.6.3.9 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](#).

#### 9.6.3.10 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](#).

**9.6.3.11** `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](#).

**9.6.3.12** `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](#).

9.6.3.13 `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>pRead-Supported</i>	Whether reading from the bank is supported.
<i>pWrite-Supported</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](#).

9.6.3.14 `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](#).



**9.6.3.15** 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 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.

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

**9.6.3.16** 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](#).

**9.6.3.17** 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>pNum-Channels</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](#).

#### 9.6.3.18 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](#).

#### 9.6.3.19 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](#).

9.6.3.20 `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.

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

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

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

9.6.3.22 `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](#).

9.6.3.23 `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](#).

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

Retrieve current trigger delay settings from the camera.

## Parameters

<i>pTrigger-Delay</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](#).

**9.6.3.25** `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](#).

**9.6.3.26** `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](#).

**9.6.3.27** `virtual Error GetTriggerModelInfo ( TriggerModelInfo * pTriggerModelInfo )`  
[pure virtual]

Retrieve trigger information from the camera.

**Parameters**

<i>pTriggerModelInfo</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](#).

**9.6.3.28** `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](#).

**9.6.3.29** `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](#).

**9.6.3.30** `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](#).

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

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

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

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

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

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

**9.6.3.34** `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](#).

**9.6.3.35** `virtual Error RetrieveBuffer ( Image * plimage )` [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>plimage</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](#).



9.6.3.36 **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](#).

9.6.3.37 **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](#).

9.6.3.38 **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](#).

9.6.3.39 **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](#).

9.6.3.40 **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](#).

9.6.3.41 `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](#).

9.6.3.42 `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](#).

**9.6.3.43** `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](#).

**9.6.3.44** `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](#).

9.6.3.45 `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](#).

9.6.3.46 `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](#).

**9.6.3.47** `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  $((\text{unsigned int})(\text{bufferSize} + \text{packetSize} - 1) / \text{packetSize}) * \text{packetSize}$ . 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>pMem- Buffers</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](#).

**9.6.3.48** `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>pCallback-Data</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](#).

```
9.6.3.49 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. Note: The use of this function with GigE Cameras is not supported.

**Parameters**

<i>num-Cameras</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>pCallback-Fns</i>	Array of callback functions for each camera.
<i>pCallback-DataArray</i>	Array of callback data pointers.

**See also**

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

**Returns**

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

**9.6.3.50 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](#).

**9.6.3.51 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>event-Number</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](#).

**9.6.3.52 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](#).

**9.6.3.53** `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](#).

## 9.6.4 Member Data Documentation

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

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

- [CameraBase.h](#)

## 9.7 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.*

### 9.7.1 Detailed Description

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

### 9.7.2 Constructor & Destructor Documentation

#### 9.7.2.1 CameraControlDlg ( )

Default constructor.

### 9.7.2.2 ~CameraControlDlg ( )

Default destructor.

## 9.7.3 Member Function Documentation

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

Connect dialog to a camera.

#### Parameters

<i>pCamera</i>	<a href="#">Camera</a> object to connect the dialog to.
----------------	---

### 9.7.3.2 void Disconnect ( )

Disconnect a connected camera from the dialog.

### 9.7.3.3 void Hide ( )

Hide the dialog.

### 9.7.3.4 bool IsVisible ( )

Get the visibility of the dialog.

#### Returns

Whether the dialog is visible.

### 9.7.3.5 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.
--------------	--

### 9.7.3.6 void Show ( )

Show the dialog.

#### 9.7.3.7 void Show ( void \* *pParent* )

Show the dialog.

#### 9.7.3.8 void ShowModal ( )

Show the modal dialog.

#### 9.7.3.9 void ShowModal ( void \* *pParent* )

Show the modal dialog.

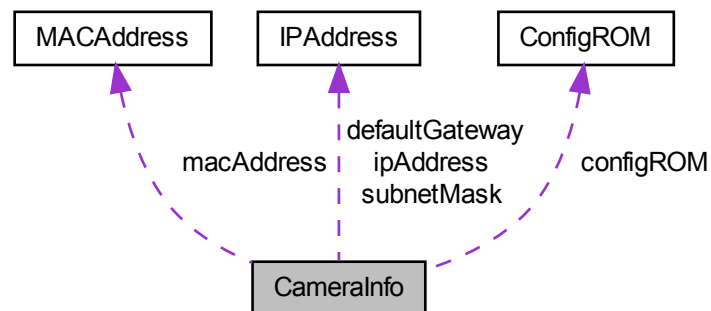
The documentation for this class was generated from the following file:

- [FlyCapture2GUI.h](#)

## 9.8 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](#)  
*PCIe Bus Speed, set to PCIE\_BUSSPEED\_UNKNOWN for unsupported drivers.*
- unsigned int [reserved](#) [16]  
*Reserved for future use.*

## IIDC specific information

- unsigned int [iicdVer](#)  
*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.*

**9.8.1 Detailed Description**

[Camera](#) information.

**9.8.2 Constructor & Destructor Documentation****9.8.2.1 [CameraInfo](#) ( ) [[inline](#)]****9.8.3 Member Data Documentation****9.8.3.1 unsigned int [applicationIPAddress](#)**

Local Application IP Address.

**9.8.3.2 unsigned int [applicationPort](#)**

Local Application port.

**9.8.3.3 [BayerTileFormat](#) [bayerTileFormat](#)**

Bayer tile format.

**9.8.3.4 unsigned short busNumber**

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

**9.8.3.5 unsigned int ccpStatus**

Status/Content of CCP register.

**9.8.3.6 ConfigROM configROM**

Configuration ROM data.

**9.8.3.7 IPAddress defaultGateway**

Default gateway.

**9.8.3.8 char driverName[sk\_maxStringLength]**

Driver name of driver being used.

**9.8.3.9 DriverType driverType**

Driver type.

**9.8.3.10 char firmwareBuildTime[sk\_maxStringLength]**

Firmware build time.

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

Firmware version of camera.

**9.8.3.12 unsigned int gigEMajorVersion**

GigE Vision version.

**9.8.3.13 unsigned int gigEMinorVersion**

GigE Vision minor version.

**9.8.3.14 unsigned int iidcVer**

DCAM version.

**9.8.3.15 InterfaceType interfaceType**

Interface type.

**9.8.3.16 IPAddress ipAddress**

IP address.

**9.8.3.17 bool isColorCamera**

Flag indicating if this is a color camera.

**9.8.3.18 MACAddress macAddress**

MAC address.

**9.8.3.19 BusSpeed maximumBusSpeed**

Maximum bus speed.

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

Device model name.

**9.8.3.21 unsigned short nodeNumber**

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

**9.8.3.22 PCIeBusSpeed pcieBusSpeed**

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

**9.8.3.23 unsigned int reserved[16]**

Reserved for future use.



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

String detailing the sensor information.

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

String providing the sensor resolution.

**9.8.3.26 unsigned int serialNumber**

Device serial number.

**9.8.3.27 IPAddress subnetMask**

Subnet mask.

**9.8.3.28 char userDefinedName[sk\_maxStringLength]**

User defined name.

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

Device vendor name.

**9.8.3.30 char xmlURL1[sk\_maxStringLength]**

XML URL 1.

**9.8.3.31 char xmlURL2[sk\_maxStringLength]**

XML URL 2.

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

- [FlyCapture2Defs.h](#)

## 9.9 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.*

### 9.9.1 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.

### 9.9.2 Constructor & Destructor Documentation

#### 9.9.2.1 [CameraSelectionDlg](#) ( )

Default constructor.

#### 9.9.2.2 [~CameraSelectionDlg](#) ( )

Default destructor.

### 9.9.3 Member Function Documentation

#### 9.9.3.1 void [SetTitle](#) ( const char \* *title* )

Set the window title.

##### Parameters

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

#### 9.9.3.2 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 <a href="#">PGRGuid</a> array.

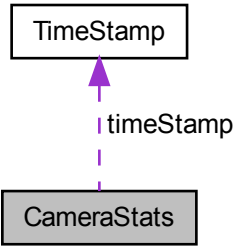
The documentation for this class was generated from the following file:

- [FlyCapture2GUI.h](#)

## 9.10 CameraStats Struct Reference

[Camera](#) diagnostic information.

Collaboration diagram for CameraStats:



### Public Member Functions

- [CameraStats](#) ()

### Public Attributes

- unsigned int [imageDropped](#)
- unsigned int [imageCorrupt](#)
- unsigned int [imageXmitFailed](#)
- unsigned int [imageDriverDropped](#)
- unsigned int [regReadFailed](#)
- unsigned int [regWriteFailed](#)
- unsigned int [portErrors](#)
- bool [cameraPowerUp](#)

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

### 9.10.1 Detailed Description

[Camera](#) diagnostic information.

### 9.10.2 Constructor & Destructor Documentation

#### 9.10.2.1 [CameraStats](#) ( ) [\[inline\]](#)

### 9.10.3 Member Data Documentation

#### 9.10.3.1 float [cameraCurrents](#)[8]

#### 9.10.3.2 bool [cameraPowerUp](#)

#### 9.10.3.3 float [cameraVoltages](#)[8]

#### 9.10.3.4 unsigned int [imageCorrupt](#)

#### 9.10.3.5 unsigned int [imageDriverDropped](#)

#### 9.10.3.6 unsigned int [imageDropped](#)

#### 9.10.3.7 unsigned int [imageXmitFailed](#)

#### 9.10.3.8 unsigned int [numCurrents](#)

The number of current registers available.

0: the values in [cameraCurrents](#)[] are invalid.

9.10.3.9 unsigned int `numResendPacketsReceived`

9.10.3.10 unsigned int `numResendPacketsRequested`

9.10.3.11 unsigned int `numVoltages`

The number of voltage registers available.

0: the values in `cameraVoltages[]` are invalid.

9.10.3.12 unsigned int `portErrors`

9.10.3.13 unsigned int `regReadFailed`

9.10.3.14 unsigned int `regWriteFailed`

9.10.3.15 unsigned int `reserved[16]`

Reserved for future use.

9.10.3.16 unsigned int `temperature`

9.10.3.17 unsigned int `timeSinceBusReset`

9.10.3.18 unsigned int `timeSinceInitialization`

9.10.3.19 `TimeStamp` `timeStamp`

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

- [FlyCapture2Defs.h](#)

## 9.11 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.*

### 9.11.1 Detailed Description

[Camera](#) configuration ROM.

### 9.11.2 Constructor & Destructor Documentation

#### 9.11.2.1 [ConfigROM](#) ( ) [[inline](#)]

### 9.11.3 Member Data Documentation

#### 9.11.3.1 unsigned int [chipIdHi](#)

Chip ID (high part).

#### 9.11.3.2 unsigned int [chipIdLo](#)

Chip ID (low part).

**9.11.3.3 unsigned int nodeVendorId**

Vendor ID of a node.

**9.11.3.4 char pszKeyword[sk\_maxStringLength]**

Keyword.

**9.11.3.5 unsigned int reserved[16]**

Reserved for future use.

**9.11.3.6 unsigned int unitSpecId**

Unit Spec ID, usually 0xa02d.

**9.11.3.7 unsigned int unitSubSWVer**

Unit sub software version.

**9.11.3.8 unsigned int unitSWVer**

Unit software version.

**9.11.3.9 unsigned int vendorUniqueInfo\_0**

Vendor unique info 0.

**9.11.3.10 unsigned int vendorUniqueInfo\_1**

Vendor unique info 1.

**9.11.3.11 unsigned int vendorUniqueInfo\_2**

Vendor unique info 2.

**9.11.3.12 unsigned int vendorUniqueInfo\_3**

Vendor unique info 3.

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

- [FlyCapture2Defs.h](#)

## 9.12 EmbeddedImageInfo Struct Reference

Properties of the possible embedded image information.

Collaboration diagram for EmbeddedImageInfo:



### Public Attributes

- [EmbeddedImageInfoProperty timestamp](#)
- [EmbeddedImageInfoProperty gain](#)
- [EmbeddedImageInfoProperty shutter](#)
- [EmbeddedImageInfoProperty brightness](#)
- [EmbeddedImageInfoProperty exposure](#)
- [EmbeddedImageInfoProperty whiteBalance](#)
- [EmbeddedImageInfoProperty frameCounter](#)
- [EmbeddedImageInfoProperty strobePattern](#)
- [EmbeddedImageInfoProperty GPIOPinState](#)
- [EmbeddedImageInfoProperty ROIPosition](#)

### 9.12.1 Detailed Description

Properties of the possible embedded image information.



### 9.12.2 Member Data Documentation

- 9.12.2.1 EmbeddedImageInfoProperty brightness
- 9.12.2.2 EmbeddedImageInfoProperty exposure
- 9.12.2.3 EmbeddedImageInfoProperty frameCounter
- 9.12.2.4 EmbeddedImageInfoProperty gain
- 9.12.2.5 EmbeddedImageInfoProperty GPIOPinState
- 9.12.2.6 EmbeddedImageInfoProperty ROIPosition
- 9.12.2.7 EmbeddedImageInfoProperty shutter
- 9.12.2.8 EmbeddedImageInfoProperty strobePattern
- 9.12.2.9 EmbeddedImageInfoProperty timestamp
- 9.12.2.10 EmbeddedImageInfoProperty whiteBalance

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

- [FlyCapture2Defs.h](#)

## 9.13 EmbeddedImageInfoProperty Struct Reference

Properties of a single embedded image info property.

### Public Member Functions

- [EmbeddedImageInfoProperty \(\)](#)

### Public Attributes

- bool [available](#)  
*Whether this property is available.*
- bool [onOff](#)  
*Whether this property is on or off.*

### 9.13.1 Detailed Description

Properties of a single embedded image info property.

### 9.13.2 Constructor & Destructor Documentation

#### 9.13.2.1 EmbeddedImageInfoProperty ( ) [inline]

### 9.13.3 Member Data Documentation

#### 9.13.3.1 bool available

Whether this property is available.

#### 9.13.3.2 bool onOff

Whether this property is on or off.

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

- [FlyCapture2Defs.h](#)

## 9.14 Error Class Reference

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

### Public Member Functions

- [Error](#) ()  
*Default constructor.*
- [Error](#) (const [Error](#) &error)  
*Copy constructor.*
- virtual [~Error](#) ()  
*Default destructor.*
- virtual [Error](#) & [operator=](#) (const [Error](#) &error)  
*Assignment operator.*
- virtual bool [operator==](#) (const [Error](#) &error) 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](#)

### 9.14.1 Detailed Description

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

Overloaded operators allow comparisons against other [Error](#) objects or the `ErrorType` enumeration.

### 9.14.2 Constructor & Destructor Documentation

#### 9.14.2.1 `Error ( )`

Default constructor.

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

Copy constructor.

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

Default destructor.

### 9.14.3 Member Function Documentation

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

Retrieve the support information.

It is not implemented in this release.

#### Returns

A string containing support information.

9.14.3.2 `virtual const char* GetBuildDate ( ) const [virtual]`

Retrieve the build date of the file where the error originated.

#### Returns

A string with the build date and time.

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

Get the error which caused this error.

#### Returns

An error object representing the cause of this error.

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

Retrieve the top level description of the error that occurred.

#### Returns

A string with the error description.

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

Retrieve the source filename where the error originated.

#### Returns

A string with the file name.

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

Retrieve the line number where the error originated.

**Returns**

The line number.

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

Retrieve the ErrorType of the error.

**Returns**

The ErrorType of the error.

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

Inequality operator.

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

Inequality operator.

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

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

Assignment operator.

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

Equality operator.

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

Equality operator.

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

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

Print a formatted log trace to stderr.

### 9.14.4 Friends And Related Function Documentation

#### 9.14.4.1 friend class `InternalError` [`friend`]

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

- [Error.h](#)

## 9.15 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.*

### 9.15.1 Member Data Documentation

#### 9.15.1.1 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.

#### 9.15.1.2 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.

#### 9.15.1.3 long long unsigned EventID

The device register which EventName maps to.

Provides an alternate means of indexing into different event types.

#### 9.15.1.4 const char\* EventName

The event name used to register the event.

Provided so the user knows which event triggered the callback.

#### 9.15.1.5 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 timestamps if there is a need to map event timestamps to specific images, if applicable.

#### 9.15.1.6 void\* EventUserData

Pointer to the user-supplied data struct.

#### 9.15.1.7 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](#)

## 9.16 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.*

### 9.16.1 Detailed Description

Options for enabling device event registration.

### 9.16.2 Member Data Documentation

#### 9.16.2.1 CameraEventCallback EventCallbackFcn

Callback function pointer.

#### 9.16.2.2 const char\* EventName

Event name to register.

#### 9.16.2.3 const void\* EventUserData

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

#### 9.16.2.4 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](#)

## 9.17 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.*

### 9.17.1 Detailed Description

Configuration for a camera.

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

### 9.17.2 Constructor & Destructor Documentation

#### 9.17.2.1 [FC2Config](#)( ) [inline]

### 9.17.3 Member Data Documentation

#### 9.17.3.1 [BusSpeed](#) [asyncBusSpeed](#)

Asynchronous bus speed.

#### 9.17.3.2 [BandwidthAllocation](#) [bandwidthAllocation](#)

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

### 9.17.3.3 **GrabMode grabMode**

Grab mode for the camera.

The default is DROP\_FRAMES.

### 9.17.3.4 **int grabTimeout**

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

### 9.17.3.5 **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.

### 9.17.3.6 **BusSpeed isochBusSpeed**

Isochronous bus speed.

### 9.17.3.7 **unsigned int minNumImageNotifications**

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

Read-only value.

### 9.17.3.8 **unsigned int numBuffers**

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

### 9.17.3.9 **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.

#### 9.17.3.10 unsigned int registerTimeout

Register read/write timeout value, in microseconds.

The default value is dependent on the interface type.

#### 9.17.3.11 unsigned int registerTimeoutRetries

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

The default value is 0.

#### 9.17.3.12 unsigned int reserved[16]

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

## 9.18 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.*

### 9.18.1 Detailed Description

The current version of the library.

### 9.18.2 Member Data Documentation

#### 9.18.2.1 unsigned int build

Build version number.

#### 9.18.2.2 unsigned int major

Major version number.

#### 9.18.2.3 unsigned int minor

Minor version number.

#### 9.18.2.4 unsigned int type

Type version number.

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

- [FlyCapture2Defs.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 WRAPPER\_API FlyCapture3ApiGuiWrapper ( void )

9.19.1.2 WRAPPER\_API ~FlyCapture3ApiGuiWrapper ( void )

### 9.19.2 Member Function Documentation

9.19.2.1 WRAPPER\_API void ConnectGUILibrary ( FlyCapture2::GCCamera & camera )

9.19.2.2 WRAPPER\_API void DisconnectGUILibrary ( )

9.19.2.3 WRAPPER\_API std::list<std::string> GetControlNameList ( )

9.19.2.4 WRAPPER\_API std::list<std::string> GetDialogNameList ( )

9.19.2.5 WRAPPER\_API int GetNumDialogs ( )

9.19.2.6 WRAPPER\_API int GetNumOfControls ( )

9.19.2.7 WRAPPER\_API void ShowCameraSelectionDialog ( )

9.19.2.8 WRAPPER\_API void ShowDialogByIndex ( int index )

9.19.2.9 WRAPPER\_API void ShowDialogByName ( std::string dialogName )

9.19.2.10 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](#) ( ) `[inline]`

### 9.20.3 Member Data Documentation

#### 9.20.3.1 unsigned int [height](#)

Height of image.

#### 9.20.3.2 Mode [mode](#)

Format 7 mode.

#### 9.20.3.3 unsigned int [offsetX](#)

Horizontal image offset.

#### 9.20.3.4 unsigned int [offsetY](#)

Vertical image offset.

#### 9.20.3.5 [PixelFormat](#) [pixelFormat](#)

Pixel format of image.

## 9.20.3.6 unsigned int reserved[8]

Reserved for future use.

## 9.20.3.7 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](#) ( ) [inline]

### 9.21.3 Member Data Documentation

#### 9.21.3.1 unsigned int [imageHStepSize](#)

Horizontal step size for the image.

#### 9.21.3.2 unsigned int [imageVStepSize](#)

Vertical step size for the image.

#### 9.21.3.3 unsigned int [maxHeight](#)

Maximum image height.

#### 9.21.3.4 unsigned int [maxPacketSize](#)

Maximum packet size in bytes for current mode.

#### 9.21.3.5 unsigned int [maxWidth](#)

Maximum image width.

#### 9.21.3.6 unsigned int [minPacketSize](#)

Minimum packet size in bytes for current mode.



### 9.21.3.7 Mode mode

Format 7 mode.

### 9.21.3.8 unsigned int offsetHStepSize

Horizontal step size for the offset.

### 9.21.3.9 unsigned int offsetVStepSize

Vertical step size for the offset.

### 9.21.3.10 unsigned int packetSize

Current packet size in bytes.

### 9.21.3.11 float percentage

Current packet size as a percentage of maximum packet size.

### 9.21.3.12 unsigned int pixelFormatBitField

Supported pixel formats in a bit field.

### 9.21.3.13 unsigned int reserved[16]

Reserved for future use.

### 9.21.3.14 unsigned int vendorPixelFormatBitField

Vendor unique pixel formats in a bit field.

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

- [FlyCapture2Defs.h](#)

## 9.22 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 ( )` `[inline]`

### 9.22.3 Member Data Documentation

#### 9.22.3.1 unsigned int `maxBytesPerPacket`

Maximum bytes per packet.

#### 9.22.3.2 unsigned int `recommendedBytesPerPacket`

Recommended bytes per packet.

#### 9.22.3.3 unsigned int `reserved`[8]

Reserved for future use.

#### 9.22.3.4 unsigned int `unitBytesPerPacket`

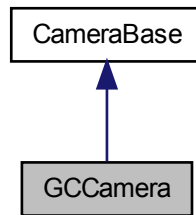
Minimum bytes per packet.

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

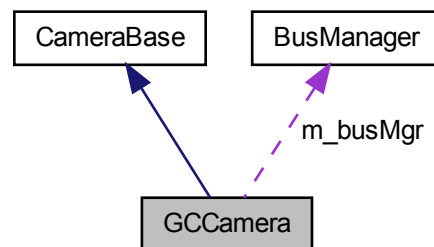
- [FlyCapture2Defs.h](#)

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

### Protected Member Functions

- void [TestGainNode](#) ()

## Protected Attributes

- [BusManager m\\_busMgr](#)

## 9.23.1 Constructor & Destructor Documentation

9.23.1.1 `GCCamera ( void )`

9.23.1.2 `virtual ~GCCamera ( void )` `[virtual]`

## 9.23.2 Member Function Documentation

9.23.2.1 `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 `Error Connect ( PGRGuid * pGuid = NULL, const char * filepath = NULL )`

9.23.2.3 `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 `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** `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** `std::string GCCamera::GetXML ( )`

**9.23.2.7** `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** `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 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 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 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** `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** `InterfaceType GetInterfaceType ( )`

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

Retrieve the current memory channel from the camera.

**Parameters**

<i>pCurrent-Channel</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** `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>pNum-Channels</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 `::GenApi::INodeMap* GetNodeMap ( )`

9.23.2.20 `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 `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** `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.

Reimplemented from [CameraBase](#).

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

Implements [CameraBase](#).

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

Retrieve current trigger delay settings from the camera.

## Parameters

<i>pTrigger-Delay</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** `virtual Error GetTriggerDelayInfo ( TriggerDelayInfo * pTriggerDelayInfo ) [virtual]`

Retrieve trigger delay information from the camera.

## Parameters

<i>pTrigger-DelayInfo</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** `virtual Error GetTriggerMode ( TriggerMode * pTriggerMode )` `[virtual]`

Retrieve current trigger settings from the camera.

## Parameters

<i>pTrigger-Mode</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** `virtual Error GetTriggerModelInfo ( TriggerModelInfo * pTriggerModelInfo )`  
`[virtual]`

Retrieve trigger information from the camera.

## Parameters

<i>pTrigger-ModelInfo</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** 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** virtual `Error ReadGVCPMemory ( unsigned int address, unsigned char * pBuffer, unsigned int length )` [virtual]**9.23.2.32** virtual `Error ReadGVCPRegister ( unsigned int address, unsigned int * pValue )` [virtual]**9.23.2.33** virtual `Error ReadGVCPRegisterBlock ( unsigned int address, unsigned int * pBuffer, unsigned int length )` [virtual]**9.23.2.34** 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** `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** `virtual Error ResetStats ( )` `[virtual]`

Implements [CameraBase](#).

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

Save the current settings to the specfied 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 **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 **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 **Error** SetCamera ( CameraBase \* *camera* )

9.23.2.43 **Error** SetCamera ( CameraBase \* *camera*, const char \* *filepath* = NULL )

9.23.2.44 **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 **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 **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** `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](#).

**9.23.2.48** `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 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>pStrobe- Control</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 virtual Error SetTriggerDelay ( const TriggerDelay * pTriggerDelay, bool broadcast  
= false ) [virtual]
```

Set the specified trigger delay settings to the camera.

## Parameters

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

## See also

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

## Returns

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

Implements [CameraBase](#).

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

Set the specified trigger settings to the camera.

## Parameters

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

## See also

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

## Returns

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

Implements [CameraBase](#).

9.23.2.52 **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>pMem- Buffers</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** `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>pCallback- Data</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** `static Error StartSyncCapture ( unsigned int numCameras, const GigECamera **  
ppCameras, const ImageEventCallback * pCallbackFns = NULL, const void **  
pCallbackDataArray = NULL ) [static]`

**9.23.2.55** `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** `void TestGainNode ( ) [protected]`

**9.23.2.57** `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>event- Number</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 `virtual Error WriteGVCPMemory ( unsigned int address, const unsigned char *  
pBuffer, unsigned int length ) [virtual]`

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

9.23.2.60 `virtual Error WriteGVCPRegisterBlock ( unsigned int address, const unsigned int *  
pBuffer, unsigned int length ) [virtual]`

9.23.2.61 `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 `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 **BusManager m\_busMgr** [protected]

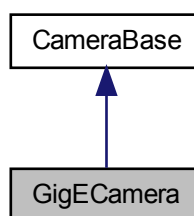
The documentation for this class was generated from the following file:

- [GCCamera.h](#)

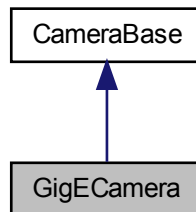
## 9.24 GigECamera Class Reference

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 \*p-CallbackData=NULL)  
*Sets the callback data to be used on completion of image transfer.*
- virtual [Error StartCapture](#) ([ImageEventCallback](#) callbackFn=NULL, const void \*p-CallbackData=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 [ImageEventCallback](#) \*pCallbackFns=NULL, const void \*\*p-CallbackDataArray=NULL)  
*StartSyncCapture() with GigE Cameras is not supported.*
- static const char \* [GetRegisterString](#) (unsigned int registerVal)  
*Returns a text representation of the register value.*

## GVCP Register Operation

These functions deal with GVCP register operation on the camera.

- virtual [Error WriteGVCPRegister](#) (unsigned int address, unsigned int value, bool broadcast=false)  
*Write a GVCP register.*
- virtual [Error ReadGVCPRegister](#) (unsigned int address, unsigned int \*pValue)  
*Read a GVCP register.*
- virtual [Error WriteGVCPRegisterBlock](#) (unsigned int address, const unsigned int \*pBuffer, unsigned int length)  
*Write a GVCP register block.*
- virtual [Error ReadGVCPRegisterBlock](#) (unsigned int address, unsigned int \*p-Buffer, unsigned int length)  
*Read a GVCP register block.*
- virtual [Error WriteGVCPMemory](#) (unsigned int address, const unsigned char \*p-Buffer, 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.*

### 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 ( )

Default constructor.

#### 9.24.2.2 virtual ~GigECamera ( ) [virtual]

Default destructor.

### 9.24.3 Member Function Documentation

#### 9.24.3.1 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 virtual Error DeregisterAllEvents ( void ) [virtual]

Implements [CameraBase](#).

#### 9.24.3.3 virtual Error DeregisterEvent ( EventOptions \* *pOpts* ) [virtual]

Implements [CameraBase](#).

#### 9.24.3.4 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 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 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 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 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 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 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 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 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 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** `virtual Error GetGigEImageBinningSettings ( unsigned int * horzBinningValue,  
unsigned int * vertBinningValue )` [virtual]

Get the current binning settings on the camera.

## Parameters

<i>horz-Binning-Value</i>	Current horizontal binning value.
<i>vert-Binning-Value</i>	Current vertical binning value.

## Returns

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

**9.24.3.15** `virtual Error GetGigEImageSettings ( GigEImageSettings * pImageSettings )`  
[virtual]

Get the current image settings on the camera.

## Parameters

<i>pImage-Settings</i>	Current image settings on camera.
------------------------	-----------------------------------

## Returns

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

**9.24.3.16** `virtual Error GetGigEImageSettingsInfo ( GigEImageSettingsInfo * pInfo )`  
[virtual]

Get information about the image settings possible on the camera.

## Parameters

<i>pInfo</i>	Image settings information.
--------------	-----------------------------

## Returns

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

9.24.3.17 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 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 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 **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 **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>pRead-Supported</i>	Whether reading from the bank is supported.
<i>pWrite-Supported</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 **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 **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](#).

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

Retrieve the current memory channel from the camera.

**Parameters**

<i>pCurrent-Channel</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 **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>pNum-Channels</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 **virtual Error GetNumStreamChannels ( unsigned int \* *numChannels* )**  
[virtual]

Get the number of stream channels present on the camera.

#### Parameters

<i>num-Channels</i>	Number of stream channels present.
---------------------	------------------------------------

#### Returns

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

9.24.3.27 **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 **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 `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.

Reimplemented from [CameraBase](#).

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

Implements [CameraBase](#).

9.24.3.31 `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>pStrobe-Control</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 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 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 **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**

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

**Returns**

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

Implements [CameraBase](#).

9.24.3.35 **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 virtual **Error** GetTriggerModelInfo ( **TriggerModelInfo** \* *pTriggerModelInfo* )  
[virtual]

Retrieve trigger information from the camera.

#### Parameters

<i>pTriggerModelInfo</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 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 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 **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 **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 **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 **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 **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 **virtual Error RegisterAllEvents ( EventOptions \* *pOpts* )** [virtual]

Implements [CameraBase](#).

9.24.3.45 **virtual Error RegisterEvent ( EventOptions \* *pOpts* )** [virtual]

Implements [CameraBase](#).

9.24.3.46 **virtual Error ResetStats ( )** [virtual]

Implements [CameraBase](#).

9.24.3.47 **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 **virtual Error RetrieveBuffer ( Image \* *plmage* )** [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>plmage</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 **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 **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 **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>pCallback-Data</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 **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 **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 **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 **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>horz-Binning-Value</i>	Horizontal binning value.
<i>vert-Binning-Value</i>	Vertical binning value.

Returns

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

9.24.3.56 **virtual Error SetGigEImageSettings ( const GigEImageSettings \* *pImageSettings* )** `[virtual]`

Set the image settings specified to the camera.

## Parameters

<i>pImage-Settings</i>	Image settings to set to camera.
------------------------	----------------------------------

## Returns

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

#### 9.24.3.57 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 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 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 **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 **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** `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** `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>pStrobe-Control</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** `virtual Error SetTriggerDelay ( const TriggerDelay * pTriggerDelay, bool broadcast = false ) [virtual]`

Set the specified trigger delay settings to the camera.

**Parameters**

<i>pTrigger-Delay</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** `virtual Error SetTriggerMode ( const TriggerMode * pTriggerMode, bool broadcast = false ) [virtual]`

Set the specified trigger settings to the camera.

**Parameters**

<i>pTrigger-Mode</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** `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>pMem- Buffers</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** `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 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 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 **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>event-Number</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 **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 **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 `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 `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 `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 ( ) [inline]

### 9.25.3 Member Data Documentation

#### 9.25.3.1 bool enablePacketResend

Turn on/off packet resend functionality.

#### 9.25.3.2 unsigned int registerTimeout

Register read/write timeout value, in microseconds.  
The default value is dependent on the interface type.

#### 9.25.3.3 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 ( )` `[inline]`

### 9.26.3 Member Data Documentation

#### 9.26.3.1 unsigned int height

Height of image.

#### 9.26.3.2 unsigned int offsetX

Horizontal image offset.

#### 9.26.3.3 unsigned int offsetY

Vertical image offset.

#### 9.26.3.4 `PixelFormat pixelFormat`

Pixel format of image.

#### 9.26.3.5 unsigned int reserved[8]

Reserved for future use.

#### 9.26.3.6 unsigned int width

Width of image.

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

- [FlyCapture2Defs.h](#)



## 9.27 GigElImageSettingsInfo Struct Reference

Format 7 information for a single mode.

### Public Member Functions

- [GigElImageSettingsInfo](#) ()

### 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 [GigElImageSettingsInfo](#) ( ) `[inline]`

### 9.27.3 Member Data Documentation

#### 9.27.3.1 unsigned int [imageHStepSize](#)

Horizontal step size for the image.

**9.27.3.2 unsigned int imageVStepSize**

Vertical step size for the image.

**9.27.3.3 unsigned int maxHeight**

Maximum image height.

**9.27.3.4 unsigned int maxWidth**

Maximum image width.

**9.27.3.5 unsigned int offsetHStepSize**

Horizontal step size for the offset.

**9.27.3.6 unsigned int offsetVStepSize**

Vertical step size for the offset.

**9.27.3.7 unsigned int pixelFormatBitField**

Supported pixel formats in a bit field.

**9.27.3.8 unsigned int reserved[16]**

Reserved for future use.

**9.27.3.9 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 bool isReadable

Whether the property is readable.

If this is false, then no other value in this structure is valid.

#### 9.28.2.2 bool isWritable

Whether the property is writable.

#### 9.28.2.3 unsigned int max

Maximum value.

#### 9.28.2.4 unsigned int min

Minimum value.

#### 9.28.2.5 GigEPropertyType propType

The type of property.

#### 9.28.2.6 unsigned int value

Current value.

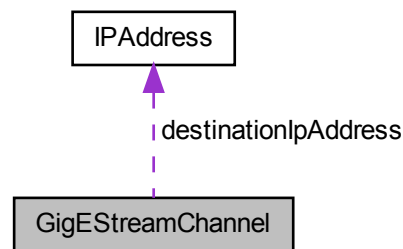
The documentation for this struct was generated from the following file:

- [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( )` `[inline]`

### 9.29.3 Member Data Documentation

#### 9.29.3.1 `IPAddress destinationIpAddress`

Destination IP address.

It can be a multicast or unicast address.

#### 9.29.3.2 `bool doNotFragment`

Disable IP fragmentation of packets.

#### 9.29.3.3 `unsigned int hostPort`

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

#### 9.29.3.4 `unsigned int interPacketDelay`

Inter packet delay, in timestamp counter units.

#### 9.29.3.5 `unsigned int networkInterfaceIndex`

Network interface index used (or to use).

#### 9.29.3.6 `unsigned int packetSize`

Packet size, in bytes.

#### 9.29.3.7 unsigned int sourcePort

Source UDP port of the stream channel.

Read only.

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

- [FlyCapture2Defs.h](#)

## 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 ( ) [inline]

#### 9.30.3 Member Data Documentation

##### 9.30.3.1 unsigned int bitrate

Bitrate to encode at.

### 9.30.3.2 float frameRate

Frame rate of the stream.

### 9.30.3.3 unsigned int height

Height of source image.

### 9.30.3.4 unsigned int reserved[256]

Reserved for future use.

### 9.30.3.5 unsigned int width

Width of source image.

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

- [FlyCapture2Defs.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](#) pixelFormat, [BayerTileFormat](#) bayerFormat)

*Sets the dimensions of the [Image](#) object.*

- virtual [Error SetData](#) (const unsigned char \*pData, unsigned int dataSize)

*Set the data of the [Image](#) object.*

- virtual [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](#) ( )

Default constructor.

#### 9.31.2.2 [Image](#) ( unsigned int *rows*, unsigned int *cols*, unsigned int *stride*, unsigned char \* *pData*, unsigned int *dataSize*, [PixelFormat](#) *format*, [BayerTileFormat](#) *bayerFormat* = NONE )

Construct an [Image](#) object with the specified arguments.

Ownership of the image buffer is not transferred to the [Image](#) object. It is the user's responsibility to delete the buffer when it is no longer in use.

#### Parameters

<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** ( 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>received-DataSize</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** ( 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** ( 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 ( const Image & image )

Copy constructor.

Both images will point to the same image buffer internally.

### 9.31.2.7 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 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 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** 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** virtual **Error** DeepCopy ( const **Image** \* *plmage* ) [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>plmage</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** 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** virtual **BayerTileFormat** GetBayerTileFormat ( ) const [virtual]

Get the Bayer tile format of the image.

**Returns**

The Bayer tile format.

**9.31.3.7** `virtual unsigned int GetBitsPerPixel ( ) const` `[virtual]`

Get the bits per pixel of the image.

**Returns**

The bits per pixel.

**9.31.3.8** `virtual unsigned int GetBlockId ( )` `[virtual]`

get the block id of the [Image](#) object.

**Returns**

The blockId assigned to the image.

**9.31.3.9** `virtual ColorProcessingAlgorithm GetColorProcessing ( ) const`  
`[virtual]`

Get the current color processing algorithm.

**See also**

[SetColorProcessing\(\)](#)

**Returns**

The current color processing algorithm.

**9.31.3.10** `virtual unsigned int GetCols ( ) const` `[virtual]`

Get the number of columns in the image.

**Returns**

The number of columns.

**9.31.3.11** virtual unsigned char\* GetData ( ) [virtual]

Get a pointer to the data associated with the image.

This function is considered unsafe. The pointer returned could be invalidated if the buffer is resized or released. The pointer may also be invalidated if the [Image](#) object is passed to [Camera::RetrieveBuffer\(\)](#). It is recommended that a [Image::DeepCopy\(\)](#) be performed if a separate copy of the [Image](#) data is required for further processing.

**Returns**

A pointer to the image data.

**9.31.3.12** virtual unsigned char\* const GetData ( ) const [virtual]**9.31.3.13** 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** static ColorProcessingAlgorithm GetDefaultColorProcessing ( ) [static]

Get the default color processing algorithm.

**See also**

[SetDefaultColorProcessing\(\)](#)

**Returns**

The default color processing algorithm.

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

Get the default output pixel format.

**See also**

[SetDefaultOutputFormat\(\)](#)

**Returns**

The default pixel format.

**9.31.3.16** `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** `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** `virtual PixelFormat GetPixelFormat ( ) const` [virtual]

Get the current pixel format.

#### Returns

The current pixel format.

**9.31.3.19** `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 `virtual unsigned int GetRows ( ) const [virtual]`

Get the number of rows in the image.

**Returns**

The number of rows.

9.31.3.21 `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 `virtual TimeStamp GetTimeStamp ( ) const [virtual]`

Get the timestamp data associated with the image.

**Returns**

Timestamp data associated with the image.

9.31.3.23 `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 `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 `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 `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 `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 `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 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 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 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** `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** `virtual Error Save ( const char * pFilename, JPEG2Option * 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** `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 **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 **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 **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 **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>default-Method</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 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 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 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 ( ) [inline]`

### 9.32.3 Member Data Documentation

#### 9.32.3.1 unsigned int `embeddedBrightness`

Embedded brightness.

#### 9.32.3.2 unsigned int `embeddedExposure`

Embedded exposure.

#### 9.32.3.3 unsigned int `embeddedFrameCounter`

Embedded frame counter.

#### 9.32.3.4 unsigned int `embeddedGain`

Embedded gain.

#### 9.32.3.5 unsigned int `embeddedGPIOPinState`

Embedded GPIO pin state.



#### 9.32.3.6 unsigned int embeddedROIPosition

Embedded ROI position.

#### 9.32.3.7 unsigned int embeddedShutter

Embedded shutter.

#### 9.32.3.8 unsigned int embeddedStrobePattern

Embedded strobe pattern.

#### 9.32.3.9 unsigned int embeddedTimeStamp

Embedded timestamp.

#### 9.32.3.10 unsigned int embeddedWhiteBalance

Embedded white balance.

#### 9.32.3.11 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 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 ( )

Default constructor.

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

Default destructor.

#### 9.33.3.3 ImageStatistics ( const ImageStatistics & other )

Copy constructor.

### 9.33.4 Member Function Documentation

#### 9.33.4.1 Error DisableAll ( )

Disable all channels.

Returns

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

#### 9.33.4.2 Error EnableAll ( )

Enable all channels.

##### Returns

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

#### 9.33.4.3 Error EnableGreyOnly ( )

Enable only the grey channel.

##### Returns

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

#### 9.33.4.4 Error EnableHSLOnly ( )

Enable only the HSL channels.

##### Returns

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

#### 9.33.4.5 Error EnableRGBOnly ( )

Enable only the RGB channels.

##### Returns

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

#### 9.33.4.6 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 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 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 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 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 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 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>pPixelValue-Min</i>	The minimum pixel value.
<i>pPixelValue-Max</i>	The maximum pixel value.
<i>pNumPixel-Values</i>	The number of unique pixel values.
<i>pPixelValue-Mean</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 ImageStatistics& operator= ( const ImageStatistics & other )**

Assignment operator.

**Parameters**

<i>other</i>	The <a href="#">ImageStatistics</a> object to copy from.
--------------	--

**9.33.4.14 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 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 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 ( )` `[inline]`

9.35.2.2 `IPAddress ( unsigned int ipAddressVal )` `[inline]`

### 9.35.3 Member Function Documentation

9.35.3.1 `bool operator!= ( const IPAddress & address )` `[inline]`

Inequality operator.

9.35.3.2 `bool operator== ( const IPAddress & address ) const` `[inline]`

Equality operator.

### 9.35.4 Member Data Documentation

9.35.4.1 `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 ( )` `[inline]`

### 9.36.3 Member Data Documentation

#### 9.36.3.1 `bool progressive`

Whether to save as a progressive JPEG file.

#### 9.36.3.2 `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 `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 ( ) [inline]

#### 9.37.3 Member Data Documentation

##### 9.37.3.1 unsigned int quality

JPEG saving quality in range (1-512).

##### 9.37.3.2 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 ( )` `[inline]`

### 9.38.3 Member Data Documentation

#### 9.38.3.1 `bool enabled`

Flag indicating if LUT is enabled.

#### 9.38.3.2 `unsigned int inputBitDepth`

The input bit depth of the LUT.

#### 9.38.3.3 `unsigned int numBanks`

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

#### 9.38.3.4 `unsigned int numChannels`

The number of LUT channels per bank available.

#### 9.38.3.5 `unsigned int numEntries`

The number of entries in the LUT.

#### 9.38.3.6 unsigned int outputBitDepth

The output bit depth of the LUT.

#### 9.38.3.7 unsigned int reserved[8]

Reserved for future use.

#### 9.38.3.8 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](#) ( ) [inline]

9.39.2.2 **MACAddress** ( unsigned int *macAddressValHigh*, unsigned int *macAddressValLow* )  
[inline]

### 9.39.3 Member Function Documentation

9.39.3.1 bool operator!= ( const **MACAddress** & *address* ) [inline]

Inequality operator.

9.39.3.2 bool operator== ( const **MACAddress** & *address* ) const [inline]

Equality operator.

### 9.39.4 Member Data Documentation

9.39.4.1 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** ( ) `[inline]`

## 9.40.3 Member Data Documentation

9.40.3.1 **float** `frameRate`

Frame rate of the stream.

9.40.3.2 **unsigned int** `quality`

[Image](#) quality (1-100)

9.40.3.3 **unsigned int** `reserved[256]`

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

- [FlyCapture2Defs.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** ( GenApi::CNodeMapRef \* *ref* )

9.41.1.2 **virtual ~NodeMap** ( void ) `[virtual]`

### 9.41.2 Member Function Documentation

#### 9.41.2.1 GenlCam::gcstring \_GetDeviceName ( )

Get device name.

#### 9.41.2.2 INode\* \_GetNode ( const GenlCam::gcstring & key )

Retrieves the node from the central map by name.

#### 9.41.2.3 void \_GetNodes ( NodeList.t & Nodes )

Retrieves all nodes in the node map.

#### 9.41.2.4 void \_InvalidateNodes ( ) const

Invalidates all nodes.

#### 9.41.2.5 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 ( ) [inline]

### 9.42.3 Member Data Documentation

#### 9.42.3.1 bool binaryFile

Whether to save the PPM as a binary file.

#### 9.42.3.2 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 ( ) [inline]

Constructor.

### 9.43.3 Member Function Documentation

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

Inequality operator.

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

Equality operator.

### 9.43.4 Member Data Documentation

#### 9.43.4.1 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 ( ) `[inline]`

### 9.44.3 Member Data Documentation

#### 9.44.3.1 unsigned int `compressionLevel`

Compression level (0-9).

0 is no compression, 9 is best compression.

#### 9.44.3.2 bool `interlaced`

Whether to save the PNG as interlaced.

#### 9.44.3.3 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 ( )` `[inline]`

### 9.45.3 Member Data Documentation

#### 9.45.3.1 `bool binaryFile`

Whether to save the PPM as a binary file.

#### 9.45.3.2 `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 ( )` [inline]

9.46.2.2 `Property ( PropertyType propType )` [inline]

### 9.46.3 Member Data Documentation

#### 9.46.3.1 bool `absControl`

Flag controlling absolute mode (real world units) or non-absolute mode (camera internal units).

#### 9.46.3.2 float `absValue`

Floating point value.

Used to configure properties in absolute mode.

#### 9.46.3.3 bool **autoManualMode**

Flag controlling auto.

#### 9.46.3.4 bool **onePush**

Flag controlling one push.

#### 9.46.3.5 bool **onOff**

Flag controlling on/off.

#### 9.46.3.6 bool **present**

Flag indicating if the property is present.

#### 9.46.3.7 unsigned int **reserved[8]**

Reserved for future use.

#### 9.46.3.8 **PropertyType** type

[Property](#) info type.

#### 9.46.3.9 unsigned int **valueA**

Value A (integer).

Used to configure properties in non-absolute mode.

#### 9.46.3.10 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** ( ) `[inline]`

9.47.2.2 **PropertyInfo** ( **PropertyType** *propType* ) `[inline]`

## 9.47.3 Member Data Documentation

9.47.3.1 **float absMax**

Maximum value (as a floating point value).

9.47.3.2 **float absMin**

Minimum value (as a floating point value).

9.47.3.3 **bool absValSupported**

Flag indicating if absolute mode is supported.

9.47.3.4 **bool autoSupported**

Flag indicating if auto is supported.

9.47.3.5 **bool manualSupported**

Flag indicating if manual is supported.

9.47.3.6 **unsigned int max**

Maximum value (as an integer).

9.47.3.7 **unsigned int min**

Minimum value (as an integer).

9.47.3.8 **bool onePushSupported**

Flag indicating if one push is supported.

9.47.3.9 **bool onOffSupported**

Flag indicating if on/off is supported.



#### 9.47.3.10 bool present

Flag indicating if the property is present.

#### 9.47.3.11 char pUnitAbbr[sk\_maxStringLength]

Abbreviated textual description of units.

#### 9.47.3.12 char pUnits[sk\_maxStringLength]

Textual description of units.

#### 9.47.3.13 bool readOutSupported

Flag indicating if property value can be read out.

#### 9.47.3.14 unsigned int reserved[8]

Reserved for future use.

#### 9.47.3.15 PropertyType type

[Property](#) info type.

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

- [FlyCapture2Defs.h](#)

## 9.48 PtGreyVideoEncoderOptions Struct Reference

Encoding options provided by the user.

### Public Attributes

- enum [PtGreyVideoCodecFormat](#) fmt
- enum [PtGreyVideoScalerMode](#) mode
- char [filename](#) [PTGREY\_BUF\_LEN]
- int [srcHeight](#)
- int [srcWidth](#)
- int [dstHeight](#)
- int [dstWidth](#)
- int [FPS\\_NUM](#)
- int [FPS\\_DEN](#)

- int [GOB](#)
- int [frames](#)
- int [BR](#)
- int [stride](#)

### 9.48.1 Detailed Description

Encoding options provided by the user.

### 9.48.2 Member Data Documentation

9.48.2.1 int [BR](#)

9.48.2.2 int [dstHeight](#)

9.48.2.3 int [dstWidth](#)

9.48.2.4 char [filename](#)[PTGREY\_BUF\_LEN]

9.48.2.5 enum [PtGreyVideoCodecFormat](#) [fmt](#)

9.48.2.6 int [FPS\\_DEN](#)

9.48.2.7 int [FPS\\_NUM](#)

9.48.2.8 int [frames](#)

9.48.2.9 int [GOB](#)

9.48.2.10 enum [PtGreyVideoScalerMode](#) [mode](#)

9.48.2.11 int [srcHeight](#)

9.48.2.12 int [srcWidth](#)

9.48.2.13 int [stride](#)

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

- [PtGreyVideoEncoder.h](#)

## 9.49 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.*

### 9.49.1 Detailed Description

A camera strobe.

### 9.49.2 Constructor & Destructor Documentation

9.49.2.1 [StrobeControl](#) ( ) [inline]

### 9.49.3 Member Data Documentation

9.49.3.1 float [delay](#)

Signal delay (in ms).

9.49.3.2 float [duration](#)

Signal duration (in ms).

9.49.3.3 bool [onOff](#)

Flag controlling on/off.

#### 9.49.3.4 unsigned int polarity

Signal polarity.

#### 9.49.3.5 unsigned int reserved[8]

Reserved for future use.

#### 9.49.3.6 unsigned int source

Source value.

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

- [FlyCapture2Defs.h](#)

## 9.50 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.*

### 9.50.1 Detailed Description

A camera strobe property.

### 9.50.2 Constructor & Destructor Documentation

#### 9.50.2.1 StrobelInfo ( ) [inline]

### 9.50.3 Member Data Documentation

#### 9.50.3.1 float maxVal

Maximum value.

#### 9.50.3.2 float minVal

Minimum value.

#### 9.50.3.3 bool onOffSupported

Flag indicating if on/off is supported.

#### 9.50.3.4 bool polaritySupported

Flag indicating if polarity is supported.

#### 9.50.3.5 bool present

Presence of strobe.

#### 9.50.3.6 bool readOutSupported

Flag indicating if strobe value can be read out.

#### 9.50.3.7 unsigned int reserved[8]

Reserved for future use.

#### 9.50.3.8 unsigned int source

Source value.

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

- [FlyCapture2Defs.h](#)

## 9.51 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](#) ()

### 9.51.1 Constructor & Destructor Documentation

9.51.1.1 MULTISYNCLIBRARY\_API [SyncManager](#) ( )

9.51.1.2 MULTISYNCLIBRARY\_API [~SyncManager](#) ( )

### 9.51.2 Member Function Documentation

9.51.2.1 MULTISYNCLIBRARY\_API bool [DisableCrossPCSynchronization](#) ( )

9.51.2.2 MULTISYNCLIBRARY\_API bool [EnableCrossPCSynchronization](#) ( )

9.51.2.3 MULTISYNCLIBRARY\_API [PGRSyncMessage GetSyncStatus](#) ( )

9.51.2.4 MULTISYNCLIBRARY\_API double [GetTimeSinceSynced](#) ( )

9.51.2.5 MULTISYNCLIBRARY\_API bool [IsTimingBusConnected](#) ( )

9.51.2.6 MULTISYNCLIBRARY\_API bool [QueryCrossPCSynchronizationSetting](#) ( )

9.51.2.7 MULTISYNCLIBRARY\_API [PGRSyncError RescanMasterTimingBus](#) ( )

9.51.2.8 MULTISYNCLIBRARY\_API [PGRSyncError Start](#) ( )

9.51.2.9 MULTISYNCLIBRARY\_API [PGRSyncError Stop](#) ( )

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

- [MultiSyncLibraryDefs.h](#)

## 9.52 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.*

### 9.52.1 Detailed Description

Description of the system.

### 9.52.2 Member Data Documentation

#### 9.52.2.1 ByteOrder byteOrder

Byte order of the system.

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

Detailed description of the CPU.

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

List of drivers used.

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

Detailed description of the GPU.

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

List of libraries used.

**9.52.2.6 size\_t numCpuCores**

Number of cores on all CPUs on the system.

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

Detailed description of the operating system.

**9.52.2.8 OSType osType**

Operating system type as described by OSType.

**9.52.2.9 unsigned int reserved[16]**

Reserved for future use.

**9.52.2.10 size\_t screenHeight**

Screen resolution height in pixels.

**9.52.2.11 size\_t screenWidth**

Screen resolution width in pixels.



9.52.2.12 `size_t sysMemSize`

Amount of memory available on the system.

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

- [Utilities.h](#)

## 9.53 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.*

#### 9.53.1 Detailed Description

Options for saving TIFF images.

#### 9.53.2 Member Enumeration Documentation

##### 9.53.2.1 enum [CompressionMethod](#)

Enumerator:

- NONE*** Save without any compression.
- PACKBITS*** Save using PACKBITS compression.
- DEFLATE*** Save using DEFLATE compression (ZLIB compression).
- ADOBE\_DEFLATE*** Save using ADOBE DEFLATE compression.

**CCITTFAX3** Save using CCITT Group 3 fax encoding. This is only valid for 1-bit images only. Default to LZW for other bit depths.

**CCITTFAX4** Save using CCITT Group 4 fax encoding. This is only valid for 1-bit images only. Default to LZW for other bit depths.

**LZW** Save using LZW compression.

**JPEG** Save using JPEG compression. This is only valid for 8-bit greyscale and 24-bit only. Default to LZW for other bit depths.

### 9.53.3 Constructor & Destructor Documentation

9.53.3.1 **TIFFOption**( ) `[inline]`

### 9.53.4 Member Data Documentation

#### 9.53.4.1 **CompressionMethod** compression

Compression method to use for encoding TIFF images.

#### 9.53.4.2 **unsigned int reserved**[16]

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

## 9.54 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.*

### 9.54.1 Detailed Description

Timestamp information.

### 9.54.2 Constructor & Destructor Documentation

#### 9.54.2.1 TimeStamp ( ) [inline]

### 9.54.3 Member Data Documentation

#### 9.54.3.1 unsigned int cycleCount

1394 cycle time count.

#### 9.54.3.2 unsigned int cycleOffset

1394 cycle time offset.

#### 9.54.3.3 unsigned int cycleSeconds

1394 cycle time seconds.

#### 9.54.3.4 unsigned int microSeconds

Microseconds.

#### 9.54.3.5 unsigned int reserved[8]

Reserved for future use.

#### 9.54.3.6 long long seconds

Seconds.

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

- [FlyCapture2Defs.h](#)

## 9.55 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.*

### 9.55.1 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.

### 9.55.2 Member Enumeration Documentation

#### 9.55.2.1 enum NodeType

Type of node.

Enumerator:

**COMPUTER**

**BUS**

**CAMERA**

**NODE**

#### 9.55.2.2 enum PortType

Possible states of a port on a node.

Enumerator:

**NOT\_CONNECTED**

**CONNECTED\_TO\_PARENT**

**CONNECTED\_TO\_CHILD**

### 9.55.3 Constructor & Destructor Documentation

#### 9.55.3.1 TopologyNode ( )

Default constructor.

### 9.55.3.2 **TopologyNode** ( *PGRGuid* *guid*, int *deviceld*, *NodeType* *nodeType*, *InterfaceType* *interfaceType* )

Constructor.

#### Parameters

<i>guid</i>	The <a href="#">PGRGuid</a> of the node (if applicable).
<i>deviceld</i>	Device ID of the node.
<i>nodeType</i>	Type of the node.
<i>interface-Type</i>	Interface type of the node.

### 9.55.3.3 **virtual** ~**TopologyNode** ( ) [virtual]

Default destructor.

### 9.55.3.4 **TopologyNode** ( const **TopologyNode** & *other* )

Copy constructor.

## 9.55.4 Member Function Documentation

### 9.55.4.1 **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.
------------------	--

### 9.55.4.2 **virtual** void **AddPortType** ( **PortType** *childPort* ) [virtual]

Add the specified **PortType** as a port of the node.

#### Parameters

<i>childPort</i>	The port to add.
------------------	------------------

### 9.55.4.3 **virtual** bool **AssignGuidToNode** ( *PGRGuid* *guid*, int *deviceld* ) [virtual]

Assign a [PGRGuid](#) and device ID to the node.

## Parameters

<i>guid</i>	<a href="#">PGRGuid</a> to be assigned.
<i>deviceld</i>	Device ID to be assigned.

## Returns

Whether the data was successfully set to the node.

9.55.4.4 `virtual bool AssignGuidToNode ( PGRGuid guid, int deviceld, 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>deviceld</i>	Device ID to be assigned.
<i>nodeType</i>	NodeType to be assigned

## Returns

Whether the data was successfully set to the node.

9.55.4.5 `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.

9.55.4.6 `virtual int GetDeviceld ( )` [virtual]

Get the device ID associated with the node.

## Returns

Device ID of the node.

#### 9.55.4.7 virtual **PGRGuid** GetGuid ( ) [virtual]

Get the **PGRGuid** associated with the node.

##### Returns

**PGRGuid** of the node.

#### 9.55.4.8 virtual **InterfaceType** GetInterfaceType ( ) [virtual]

Get the interface type associated with the node.

##### Returns

Interface type of the node.

#### 9.55.4.9 virtual **NodeType** GetNodeType ( ) [virtual]

Get the node type associated with the node.

##### Returns

Node type of the node.

#### 9.55.4.10 virtual unsigned int GetNumChildren ( ) [virtual]

Get the number of child nodes.

##### Returns

Number of child nodes.

#### 9.55.4.11 virtual unsigned int GetNumPorts ( ) [virtual]

Get the number of ports.

##### Returns

Number of ports.

#### 9.55.4.12 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.

9.55.4.13 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](#)

## 9.56 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.*

### 9.56.1 Detailed Description

A camera trigger.

### 9.56.2 Constructor & Destructor Documentation

#### 9.56.2.1 `TriggerMode ( )` `[inline]`

### 9.56.3 Member Data Documentation

#### 9.56.3.1 `unsigned int mode`

Mode value.

#### 9.56.3.2 `bool onOff`

Flag controlling on/off.

#### 9.56.3.3 `unsigned int parameter`

Parameter value.

#### 9.56.3.4 `unsigned int polarity`

Polarity value.

#### 9.56.3.5 `unsigned int reserved[8]`

Reserved for future use.

#### 9.56.3.6 `unsigned int source`

Source value.

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

- [FlyCapture2Defs.h](#)

## 9.57 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.*

### 9.57.1 Detailed Description

Information about a camera trigger property.

### 9.57.2 Constructor & Destructor Documentation

#### 9.57.2.1 [TriggerModelInfo](#) ( ) `[inline]`

### 9.57.3 Member Data Documentation

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

Mode mask.

#### 9.57.3.2 bool [onOffSupported](#)

Flag indicating if on/off is supported.

### 9.57.3.3 bool `polaritySupported`

Flag indicating if polarity is supported.

### 9.57.3.4 bool `present`

Presence of trigger mode.

### 9.57.3.5 bool `readOutSupported`

Flag indicating if trigger value can be read out.

### 9.57.3.6 unsigned int `reserved[8]`

Reserved for future use.

### 9.57.3.7 bool `softwareTriggerSupported`

Flag indicating if software trigger is supported.

### 9.57.3.8 unsigned int `sourceMask`

Source mask.

### 9.57.3.9 bool `valueReadable`

Flag indicating if the value is readable.

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

- [FlyCapture2Defs.h](#)

## 9.58 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 \*pUserData)  
*Execute a command in the terminal.*

### 9.58.1 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.

### 9.58.2 Member Function Documentation

#### 9.58.2.1 static Error CheckDriver ( const [PGRGuid](#) \* guid ) [static]

Check for driver compatibility for the given camera guid.

##### Parameters

<a href="#">guid</a>	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.

#### 9.58.2.2 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.

**9.58.2.3 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.

**9.58.2.4 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.

**9.58.2.5 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.

**9.58.2.6 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.

**9.58.2.7 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.

**9.58.2.8 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 AVIRecorder.h File Reference

#### Classes

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

#### Namespaces

- namespace [FlyCapture2](#)

### 10.2 BusManager.h File Reference

#### Classes

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

#### Namespaces

- namespace [FlyCapture2](#)

#### Typedefs

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

*Bus event callback function prototype.*

- typedef void \* [CallbackHandle](#)

*Handle that is returned when registering a callback.*

## 10.3 Camera.h File Reference

### Classes

- class [Camera](#)

*The [Camera](#) object represents a physical camera that uses the I2C register set.*

### Namespaces

- namespace [FlyCapture2](#)

## 10.4 CameraBase.h File Reference

### Classes

- class [CameraBase](#)

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

### Namespaces

- namespace [FlyCapture2](#)

### Typedefs

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

*[Image](#) event callback function prototype.*

## 10.5 Error.h File Reference

### Classes

- class [Error](#)

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

## Namespaces

- namespace [FlyCapture2](#)

## 10.6 FlyCapture2.h File Reference

## 10.7 FlyCapture2Defs.h File Reference

### 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 [MJPGOption](#)  
*Options for saving MJPG files.*
- struct [H264Option](#)  
*Options for saving H264 files.*
- struct [AVIOption](#)  
*Options for saving AVI files.*
- struct [EventOptions](#)  
*Options for enabling device event registration.*
- struct [EventCallbackData](#)

## Namespaces

- namespace [FlyCapture2](#)

## Defines

- #define [NULL](#) 0
- #define [FULL\\_32BIT\\_VALUE](#) 0x7FFFFFFF

## Typedefs

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

- 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.*

## 10.7.1 Define Documentation

10.7.1.1 `#define FULL_32BIT_VALUE 0x7FFFFFFF`

10.7.1.2 `#define NULL 0`

## 10.8 FlyCapture2GUI.h File Reference

### 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

- namespace [FlyCapture2](#)

## 10.9 FlyCapture2Platform.h File Reference

### Defines

- `#define FLYCAPTURE2_API __attribute__((visibility ("default")))`
- `#define FLYCAPTURE2_LOCAL __attribute__((visibility ("hidden")))`

## 10.9.1 Define Documentation

10.9.1.1 `#define FLYCAPTURE2_API __attribute__((visibility ("default")))`

10.9.1.2 `#define FLYCAPTURE2_LOCAL __attribute__((visibility ("hidden")))`

## 10.10 FlyCapture3ApiGuiWrapper.h File Reference



## Classes

- class [FlyCapture3ApiGuiWrapper](#)

## Namespaces

- namespace [FlyCapture2](#)
- namespace [FlyCap3CameraControl](#)

## Defines

- #define [WRAPPER\\_API](#) \_\_declspec(dllimport)

### 10.10.1 Define Documentation

10.10.1.1 #define WRAPPER\_API \_\_declspec(dllimport)

## 10.11 GCCamera.h File Reference

## Classes

- class [GCCamera](#)

## Namespaces

- namespace [FlyCapture2](#)

## 10.12 GigECamera.h File Reference

## Classes

- class [GigECamera](#)

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

## Namespaces

- namespace [FlyCapture2](#)

## 10.13 Image.h File Reference

### Classes

- class [Image](#)

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

### Namespaces

- namespace [FlyCapture2](#)

## 10.14 ImageStatistics.h File Reference

### Classes

- class [ImageStatistics](#)

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

### Namespaces

- namespace [FlyCapture2](#)

## 10.15 Internal.h File Reference

### Classes

- class [Internal](#)

### Namespaces

- namespace [FlyCapture2](#)

## 10.16 Licensing.dox File Reference

## 10.17 MultiSyncLibrary.h File Reference

## 10.18 MultiSyncLibraryDefs.h File Reference

## Classes

- class [SyncManager](#)

## Namespaces

- namespace [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](#) }

## 10.19 MultiSyncLibraryPlatform.h File Reference

### Defines

- #define [MULTISYNCLIBRARY\\_API](#) \_\_attribute\_\_((visibility ("default")))
- #define [MULTISYNCLIBRARY\\_LOCAL](#) \_\_attribute\_\_((visibility ("hidden")))

### 10.19.1 Define Documentation

10.19.1.1 #define [MULTISYNCLIBRARY\\_API](#) \_\_attribute\_\_((visibility ("default")))

10.19.1.2 #define [MULTISYNCLIBRARY\\_LOCAL](#) \_\_attribute\_\_((visibility ("hidden")))

## 10.20 NodeMap.h File Reference

### Classes

- class [NodeMap](#)

### Namespaces

- namespace [FlyCapture2](#)

## 10.21 PtGreyVideoEncoder.h File Reference

The API functions within this library are re-entrant, as each video encoding session is expected to use a unique handle acquire via [PtGreyCreateVideoEncoder\(\)](#).

### Classes

- struct [PtGreyVideoEncoderOptions](#)  
*Encoding options provided by the user.*

### Defines

- #define [FALSE](#) 0
- #define [TRUE](#) 1
- #define [FILENAME\\_MAX](#) (4096)
- #define [PTGREY\\_BUF\\_LEN](#) (FILENAME\_MAX)
- #define [PTGREY\\_BR\\_MIN](#) (10)

### Typedefs

- typedef int [BOOL](#)

### Enumerations

- enum [PtGreyVideoCodecFormat](#) { [PTGREY\\_VIDEO\\_UNCOMPRESSED](#) = 0, [PTGREY\\_VIDEO\\_MJPEG](#), [PTGREY\\_VIDEO\\_H264](#), [PTGREY\\_VIDEO\\_MAX](#) }  
*Video formats supported for conversion.*
- enum [PtGreyVideoScalerMode](#) { [PTGREY\\_SCALER\\_FAST\\_BILINEAR](#) = 0, [PTGREY\\_SCALER\\_BILINEAR](#), [PTGREY\\_SCALER\\_BICUBIC](#), [PTGREY\\_SCALER\\_MAX](#) }  
*Image conversion algorithms supported for re-scaling images.*

### Functions

- [PTGREY\\_VIDEO\\_ENCODER\\_API](#) int [PtGreyCreateVideoEncoder](#) (struct [PtGreyHandle](#) \*\*pHandle, struct [PtGreyVideoEncoderOptions](#) \*pOpts)
- [PTGREY\\_VIDEO\\_ENCODER\\_API](#) int [PtGreyDestroyVideoEncoder](#) (struct [PtGreyHandle](#) \*\*pHandle)
- [PTGREY\\_VIDEO\\_ENCODER\\_API](#) int [PtGreyWriteFrame](#) (struct [PtGreyHandle](#) \*pHandle, const unsigned char \*pFrame)

### 10.21.1 Detailed Description

The API functions within this library are re-entrant, as each video encoding session is expected to use a unique handle acquire via [PtGreyCreateVideoEncoder\(\)](#). The code is not thread safe, as there is not internal mutex to prevent the user from attempting to close an encoding session while in progress. The library can be completely thread safe by wrapping each API call for a given handle with a unique mutex.

### 10.21.2 Define Documentation

10.21.2.1 `#define FALSE 0`

10.21.2.2 `#define FILENAME_MAX (4096)`

10.21.2.3 `#define PTGREY_BR_MIN (10)`

10.21.2.4 `#define PTGREY_BUF_LEN (FILENAME_MAX)`

10.21.2.5 `#define TRUE 1`

### 10.21.3 Typedef Documentation

10.21.3.1 `typedef int BOOL`

### 10.21.4 Enumeration Type Documentation

10.21.4.1 `enum PtGreyVideoCodecFormat`

Video formats supported for conversion.

Enumerator:

***PTGREY\_VIDEO\_UNCOMPRESSED***  
***PTGREY\_VIDEO\_MJPG***  
***PTGREY\_VIDEO\_H264***  
***PTGREY\_VIDEO\_MAX***

10.21.4.2 `enum PtGreyVideoScalerMode`

Image conversion algorithms supported for re-scaling images.

Enumerator:

***PTGREY\_SCALER\_FAST\_BILINEAR***  
***PTGREY\_SCALER\_BILINEAR***  
***PTGREY\_SCALER\_BICUBIC***  
***PTGREY\_SCALER\_MAX***

### 10.21.5 Function Documentation

10.21.5.1 PTGREY\_VIDEO\_ENCODER\_API int PtGreyCreateVideoEncoder ( struct PtGreyHandle  
\*\* *pHandle*, struct PtGreyVideoEncoderOptions \* *pOpts* )

10.21.5.2 PTGREY\_VIDEO\_ENCODER\_API int PtGreyDestroyVideoEncoder ( struct PtGreyHandle  
\*\* *pHandle* )

10.21.5.3 PTGREY\_VIDEO\_ENCODER\_API int PtGreyWriteFrame ( struct PtGreyHandle \*  
*pHandle*, const unsigned char \* *pFrame* )

## 10.22 PtGreyVideoEncoderPlatform.h File Reference

### Defines

- `#define` [PTGREY\\_VIDEO\\_ENCODER\\_API](#)

### 10.22.1 Define Documentation

10.22.1.1 `#define` [PTGREY\\_VIDEO\\_ENCODER\\_API](#)

## 10.23 TopologyNode.h File Reference

### Classes

- class [TopologyNode](#)

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

### Namespaces

- namespace [FlyCapture2](#)

## 10.24 Utilities.h File Reference

### Classes

- struct [SystemInfo](#)

*Description of the system.*

- class [Utilities](#)

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

## Namespaces

- namespace [FlyCapture2](#)

## Typedefs

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

*Async command callback function prototype.*

## Enumerations

- enum [OSType](#) { [WINDOWS\\_X86](#), [WINDOWS\\_X64](#), [LINUX\\_X86](#), [LINUX\\_X64](#), [MAC](#), [UNKNOWN\\_OS](#), [OSTYPE\\_FORCE\\_32BITS](#) = [FULL\\_32BIT\\_VALUE](#) }

*Possible operating systems.*

- enum [ByteOrder](#) { [BYTE\\_ORDER\\_LITTLE\\_ENDIAN](#), [BYTE\\_ORDER\\_BIG\\_ENDIAN](#), [BYTE\\_ORDER\\_FORCE\\_32BITS](#) = [FULL\\_32BIT\\_VALUE](#) }

*Possible byte orders.*

# Index

- ~AVIRecorder
  - FlyCapture2::AVIRecorder, [48](#)
- ~BusManager
  - FlyCapture2::BusManager, [53](#)
- ~Camera
  - FlyCapture2::Camera, [66](#)
- ~CameraBase
  - FlyCapture2::CameraBase, [95](#)
- ~CameraControlDlg
  - FlyCapture2::CameraControlDlg, [118](#)
- ~CameraSelectionDlg
  - FlyCapture2::CameraSelectionDlg, [126](#)
- ~Error
  - FlyCapture2::Error, [135](#)
- ~FlyCapture3ApiGuiWrapper
  - FlyCap3CameraControl::FlyCapture3-ApiGuiWrapper, [145](#)
- ~GCCamera
  - FlyCapture2::GCCamera, [155](#)
- ~GigECamera
  - FlyCapture2::GigECamera, [182](#)
- ~Image
  - FlyCapture2::Image, [224](#)
- ~ImageStatistics
  - FlyCapture2::ImageStatistics, [239](#)
- ~NodeMap
  - FlyCapture2::NodeMap, [251](#)
- ~SyncManager
  - MultiSyncLibrary::SyncManager, [266](#)
- ~TopologyNode
  - FlyCapture2::TopologyNode, [274](#)
- ADOBE\_DEFLATE
  - FlyCapture2::TIFFOption, [269](#)
- ARRIVAL
  - Enumerations, [19](#)
- AUTO\_EXPOSURE
  - Enumerations, [26](#)
- BANDWIDTH\_ALLOCATION\_FORCE\_32BITS
  - Enumerations, [18](#)
- BANDWIDTH\_ALLOCATION\_OFF
  - Enumerations, [18](#)
- BANDWIDTH\_ALLOCATION\_ON
  - Enumerations, [18](#)
- BANDWIDTH\_ALLOCATION\_UNSPECIFIED
  - Enumerations, [18](#)
- BANDWIDTH\_ALLOCATION\_UNSUPPORTED
  - Enumerations, [18](#)
- BGGR
  - Enumerations, [19](#)
- BLUE
  - FlyCapture2::ImageStatistics, [239](#)
- BMP
  - Enumerations, [24](#)
- BRIGHTNESS
  - Enumerations, [26](#)
- BT\_FORCE\_32BITS
  - Enumerations, [19](#)
- BUFFER\_FRAMES
  - Enumerations, [23](#)
- BUS
  - FlyCapture2::TopologyNode, [273](#)
- BUSSPEED\_10000BASE\_T
  - Enumerations, [19](#)
- BUSSPEED\_1000BASE\_T
  - Enumerations, [19](#)
- BUSSPEED\_100BASE\_T
  - Enumerations, [19](#)
- BUSSPEED\_10BASE\_T
  - Enumerations, [19](#)
- BUSSPEED\_ANY
  - Enumerations, [20](#)
- BUSSPEED\_FORCE\_32BITS
  - Enumerations, [20](#)
- BUSSPEED\_S100
  - Enumerations, [19](#)
- BUSSPEED\_S1600
  - Enumerations, [19](#)



- BUSSPEED\_S200
  - Enumerations, [19](#)
- BUSSPEED\_S3200
  - Enumerations, [19](#)
- BUSSPEED\_S400
  - Enumerations, [19](#)
- BUSSPEED\_S480
  - Enumerations, [19](#)
- BUSSPEED\_S5000
  - Enumerations, [19](#)
- BUSSPEED\_S800
  - Enumerations, [19](#)
- BUSSPEED\_SPEED\_UNKNOWN
  - Enumerations, [20](#)
- BUSSPEED\_S\_FASTEST
  - Enumerations, [20](#)
- BUS\_RESET
  - Enumerations, [19](#)
- BYTE\_ORDER\_BIG\_ENDIAN
  - FlyCapture2, [44](#)
- BYTE\_ORDER\_FORCE\_32BITS
  - FlyCapture2, [44](#)
- BYTE\_ORDER\_LITTLE\_ENDIAN
  - FlyCapture2, [44](#)
- CALLBACK\_TYPE\_FORCE\_32BITS
  - Enumerations, [19](#)
- CAMERA
  - FlyCapture2::TopologyNode, [273](#)
- CCITTFAX3
  - FlyCapture2::TIFFOption, [269](#)
- CCITTFAX4
  - FlyCapture2::TIFFOption, [270](#)
- COLOR\_PROCESSING\_ALGORITHM\_FORCE\_32BITS
  - Enumerations, [20](#)
- COMPUTER
  - FlyCapture2::TopologyNode, [273](#)
- CONNECTED\_TO\_CHILD
  - FlyCapture2::TopologyNode, [273](#)
- CONNECTED\_TO\_PARENT
  - FlyCapture2::TopologyNode, [273](#)
- DEFAULT
  - Enumerations, [20](#)
- DEFLATE
  - FlyCapture2::TIFFOption, [269](#)
- DIRECTIONAL\_FILTER
  - Enumerations, [20](#)
- DRIVER\_1394\_CAM
  - Enumerations, [20](#)
- DRIVER\_1394\_JUJU
  - Enumerations, [20](#)
- DRIVER\_1394\_PRO
  - Enumerations, [20](#)
- DRIVER\_1394\_RAW1394
  - Enumerations, [20](#)
- DRIVER\_1394\_VIDEO1394
  - Enumerations, [20](#)
- DRIVER\_FORCE\_32BITS
  - Enumerations, [21](#)
- DRIVER\_GIGE\_FILTER
  - Enumerations, [21](#)
- DRIVER\_GIGE\_LWF
  - Enumerations, [21](#)
- DRIVER\_GIGE\_NONE
  - Enumerations, [21](#)
- DRIVER\_GIGE\_PRO
  - Enumerations, [21](#)
- DRIVER\_UNKNOWN
  - Enumerations, [21](#)
- DRIVER\_USB3\_PRO
  - Enumerations, [21](#)
- DRIVER\_USB\_CAM
  - Enumerations, [20](#)
- DRIVER\_USB\_NONE
  - Enumerations, [20](#)
- DROP\_FRAMES
  - Enumerations, [23](#)
- EDGE\_SENSING
  - Enumerations, [20](#)
- Enumerations
  - ARRIVAL, [19](#)
  - AUTO\_EXPOSURE, [26](#)
  - BANDWIDTH\_ALLOCATION\_FORCE\_32BITS, [18](#)
  - BANDWIDTH\_ALLOCATION\_OFF, [18](#)
  - BANDWIDTH\_ALLOCATION\_ON, [18](#)
  - BANDWIDTH\_ALLOCATION\_UNSPECIFIED, [18](#)
  - BANDWIDTH\_ALLOCATION\_UNSUPPORTED, [18](#)
  - BGGR, [19](#)
  - BMP, [24](#)
  - BRIGHTNESS, [26](#)
  - BT\_FORCE\_32BITS, [19](#)
  - BUFFER\_FRAMES, [23](#)
  - BUSSPEED\_10000BASE\_T, [19](#)
  - BUSSPEED\_1000BASE\_T, [19](#)
  - BUSSPEED\_100BASE\_T, [19](#)

BUSSPEED\_10BASE\_T, 19  
 BUSSPEED\_ANY, 20  
 BUSSPEED\_FORCE\_32BITS, 20  
 BUSSPEED\_S100, 19  
 BUSSPEED\_S1600, 19  
 BUSSPEED\_S200, 19  
 BUSSPEED\_S3200, 19  
 BUSSPEED\_S400, 19  
 BUSSPEED\_S480, 19  
 BUSSPEED\_S5000, 19  
 BUSSPEED\_S800, 19  
 BUSSPEED\_SPEED\_UNKNOWN, 20  
 BUSSPEED\_S\_FASTEST, 20  
 BUS\_RESET, 19  
 CALLBACK\_TYPE\_FORCE\_32BITS, 19  
 COLOR\_PROCESSING\_ALGORITHM\_FORCE\_32BITS, 20  
 DEFAULT, 20  
 DIRECTIONAL\_FILTER, 20  
 DRIVER\_1394\_CAM, 20  
 DRIVER\_1394\_JUJU, 20  
 DRIVER\_1394\_PRO, 20  
 DRIVER\_1394\_RAW1394, 20  
 DRIVER\_1394\_VIDEO1394, 20  
 DRIVER\_FORCE\_32BITS, 21  
 DRIVER\_GIGE\_FILTER, 21  
 DRIVER\_GIGE\_LWF, 21  
 DRIVER\_GIGE\_NONE, 21  
 DRIVER\_GIGE\_PRO, 21  
 DRIVER\_UNKNOWN, 21  
 DRIVER\_USB3\_PRO, 21  
 DRIVER\_USB\_CAM, 20  
 DRIVER\_USB\_NONE, 20  
 DROP\_FRAMES, 23  
 EDGE\_SENSING, 20  
 FOCUS, 27  
 FRAMERATE\_120, 23  
 FRAMERATE\_15, 22  
 FRAMERATE\_1\_875, 22  
 FRAMERATE\_240, 23  
 FRAMERATE\_30, 22  
 FRAMERATE\_3\_75, 22  
 FRAMERATE\_60, 22  
 FRAMERATE\_7\_5, 22  
 FRAMERATE\_FORCE\_32BITS, 23  
 FRAMERATE\_FORMAT7, 23  
 FRAME\_RATE, 27  
 FROM\_FILE\_EXT, 24  
 GAIN, 27  
 GAMMA, 27  
 GBRG, 19  
 GRAB\_MODE\_FORCE\_32BITS, 23  
 GRAB\_TIMEOUT\_FORCE\_32BITS, 23  
 GRBG, 19  
 HQ\_LINEAR, 20  
 HUE, 26  
 IMAGE\_FILE\_FORMAT\_FORCE\_32BITS, 24  
 INTERFACE\_GIGE, 24  
 INTERFACE\_IEEE1394, 24  
 INTERFACE\_TYPE\_FORCE\_32BITS, 24  
 INTERFACE\_UNKNOWN, 24  
 INTERFACE\_USB2, 24  
 INTERFACE\_USB3, 24  
 IPP, 20  
 IRIS, 27  
 JPEG, 24  
 JPEG2000, 24  
 MODE\_0, 24  
 MODE\_1, 24  
 MODE\_10, 25  
 MODE\_11, 25  
 MODE\_12, 25  
 MODE\_13, 25  
 MODE\_14, 25  
 MODE\_15, 25  
 MODE\_16, 25  
 MODE\_17, 25  
 MODE\_18, 25  
 MODE\_19, 25  
 MODE\_2, 24  
 MODE\_20, 25  
 MODE\_21, 25  
 MODE\_22, 25  
 MODE\_23, 25  
 MODE\_24, 25  
 MODE\_25, 25  
 MODE\_26, 25  
 MODE\_27, 25  
 MODE\_28, 25  
 MODE\_29, 25  
 MODE\_3, 24  
 MODE\_30, 25  
 MODE\_31, 25  
 MODE\_4, 24  
 MODE\_5, 24

- MODE\_6, [24](#)
- MODE\_7, [25](#)
- MODE\_8, [25](#)
- MODE\_9, [25](#)
- MODE\_FORCE\_32BITS, [25](#)
- NEAREST\_NEIGHBOR, [20](#)
- NONE, [19](#)
- NO\_COLOR\_PROCESSING, [20](#)
- NUM\_FRAMERATES, [23](#)
- NUM\_MODES, [25](#)
- NUM\_PIXEL\_FORMATS, [26](#)
- NUM\_VIDEOMODES, [28](#)
- PAN, [27](#)
- PCIE\_BUSSPEED\_2\_5, [25](#)
- PCIE\_BUSSPEED\_5\_0, [25](#)
- PCIE\_BUSSPEED\_FORCE\_32BITS, [25](#)
- PCIE\_BUSSPEED\_UNKNOWN, [25](#)
- PGM, [24](#)
- PGRERROR\_BUFFER\_TOO\_SMALL, [22](#)
- PGRERROR\_BUS\_MASTER\_FAILED, [21](#)
- PGRERROR\_FAILED, [21](#)
- PGRERROR\_FAILED\_BUS\_MASTER\_CONNECTION, [21](#)
- PGRERROR\_FAILED\_GUID, [21](#)
- PGRERROR\_FORCE\_32BITS, [22](#)
- PGRERROR\_IIDC\_FAILED, [21](#)
- PGRERROR\_IMAGE\_CONSISTENCY\_ERROR, [22](#)
- PGRERROR\_IMAGE\_CONVERSION\_FAILED, [22](#)
- PGRERROR\_IMAGE\_LIBRARY\_FAILURE, [22](#)
- PGRERROR\_INCOMPATIBLE\_DRIVER, [22](#)
- PGRERROR\_INIT\_FAILED, [21](#)
- PGRERROR\_INVALID\_BUS\_MANAGER, [21](#)
- PGRERROR\_INVALID\_GENERATION, [21](#)
- PGRERROR\_INVALID\_MODE, [21](#)
- PGRERROR\_INVALID\_PACKET\_SIZE, [21](#)
- PGRERROR\_INVALID\_PARAMETER, [21](#)
- PGRERROR\_INVALID\_SETTINGS, [21](#)
- PGRERROR\_ISOCH\_ALREADY\_STARTED, [22](#)
- PGRERROR\_ISOCH\_BANDWIDTH\_EXCEEDED, [22](#)
- PGRERROR\_ISOCH\_FAILED, [22](#)
- PGRERROR\_ISOCH\_NOT\_STARTED, [22](#)
- PGRERROR\_ISOCH\_RETRIEVE\_BUFFER\_FAILED, [22](#)
- PGRERROR\_ISOCH\_START\_FAILED, [22](#)
- PGRERROR\_ISOCH\_STOP\_FAILED, [22](#)
- PGRERROR\_ISOCH\_SYNC\_FAILED, [22](#)
- PGRERROR\_LOW\_LEVEL\_FAILURE, [21](#)
- PGRERROR\_LUT\_FAILED, [21](#)
- PGRERROR\_MEMORY\_ALLOCATION\_FAILED, [21](#)
- PGRERROR\_NOT\_CONNECTED, [21](#)
- PGRERROR\_NOT\_FOUND, [21](#)
- PGRERROR\_NOT\_IMPLEMENTED, [21](#)
- PGRERROR\_NOT\_INITIALIZED, [21](#)
- PGRERROR\_NOT\_IN\_FORMAT7, [21](#)
- PGRERROR\_NOT\_SUPPORTED, [21](#)
- PGRERROR\_OK, [21](#)
- PGRERROR\_PROPERTY\_FAILED, [22](#)
- PGRERROR\_PROPERTY\_NOT\_PRESENT, [22](#)
- PGRERROR\_READ\_REGISTER\_FAILED, [22](#)
- PGRERROR\_REGISTER\_FAILED, [22](#)
- PGRERROR\_STROBE\_FAILED, [22](#)
- PGRERROR\_TIMEOUT, [21](#)
- PGRERROR\_TRIGGER\_FAILED, [22](#)
- PGRERROR\_UNDEFINED, [21](#)
- PGRERROR\_WRITE\_REGISTER\_FAILED, [22](#)
- PIXEL\_FORMAT\_411YUV8, [26](#)
- PIXEL\_FORMAT\_422YUV8, [26](#)

PIXEL\_FORMAT\_422YUV8\_JPEG, [26](#)  
 PIXEL\_FORMAT\_444YUV8, [26](#)  
 PIXEL\_FORMAT\_BGR, [26](#)  
 PIXEL\_FORMAT\_BGR16, [26](#)  
 PIXEL\_FORMAT\_BGRU, [26](#)  
 PIXEL\_FORMAT\_BGRU16, [26](#)  
 PIXEL\_FORMAT\_MONO12, [26](#)  
 PIXEL\_FORMAT\_MONO16, [26](#)  
 PIXEL\_FORMAT\_MONO8, [26](#)  
 PIXEL\_FORMAT\_RAW12, [26](#)  
 PIXEL\_FORMAT\_RAW16, [26](#)  
 PIXEL\_FORMAT\_RAW8, [26](#)  
 PIXEL\_FORMAT\_RGB, [26](#)  
 PIXEL\_FORMAT\_RGB16, [26](#)  
 PIXEL\_FORMAT\_RGB8, [26](#)  
 PIXEL\_FORMAT\_RGBU, [26](#)  
 PIXEL\_FORMAT\_S\_MONO16, [26](#)  
 PIXEL\_FORMAT\_S\_RGB16, [26](#)  
 PNG, [24](#)  
 PPM, [24](#)  
 PROPERTY\_TYPE\_FORCE\_32BIT-  
     S, [27](#)  
 RAW, [24](#)  
 REMOVAL, [19](#)  
 RGGB, [19](#)  
 RIGOROUS, [20](#)  
 SATURATION, [26](#)  
 SHARPNESS, [26](#)  
 SHUTTER, [27](#)  
 TEMPERATURE, [27](#)  
 TIFF, [24](#)  
 TILT, [27](#)  
 TIMEOUT\_INFINITE, [23](#)  
 TIMEOUT\_NONE, [23](#)  
 TIMEOUT\_UNSPECIFIED, [23](#)  
 TRIGGER\_DELAY, [27](#)  
 TRIGGER\_MODE, [27](#)  
 UNSPECIFIED\_GRAB\_MODE, [23](#)  
 UNSPECIFIED\_PIXEL\_FORMAT, [26](#)  
 UNSPECIFIED\_PROPERTY\_TYPE, [27](#)  
 VIDEOMODE\_1024x768RGB, [27](#)  
 VIDEOMODE\_1024x768Y16, [27](#)  
 VIDEOMODE\_1024x768Y8, [27](#)  
 VIDEOMODE\_1024x768YUV422, [27](#)  
 VIDEOMODE\_1280x960RGB, [27](#)  
 VIDEOMODE\_1280x960Y16, [28](#)  
 VIDEOMODE\_1280x960Y8, [27](#)  
 VIDEOMODE\_1280x960YUV422, [27](#)  
 VIDEOMODE\_1600x1200RGB, [28](#)  
 VIDEOMODE\_1600x1200Y16, [28](#)  
 VIDEOMODE\_1600x1200Y8, [28](#)  
 VIDEOMODE\_1600x1200YUV422, [28](#)  
 VIDEOMODE\_160x120YUV444, [27](#)  
 VIDEOMODE\_320x240YUV422, [27](#)  
 VIDEOMODE\_640x480RGB, [27](#)  
 VIDEOMODE\_640x480Y16, [27](#)  
 VIDEOMODE\_640x480Y8, [27](#)  
 VIDEOMODE\_640x480YUV411, [27](#)  
 VIDEOMODE\_640x480YUV422, [27](#)  
 VIDEOMODE\_800x600RGB, [27](#)  
 VIDEOMODE\_800x600Y16, [27](#)  
 VIDEOMODE\_800x600Y8, [27](#)  
 VIDEOMODE\_800x600YUV422, [27](#)  
 VIDEOMODE\_FORCE\_32BITS, [28](#)  
 VIDEOMODE\_FORMAT7, [28](#)  
 WEIGHTED\_DIRECTIONAL\_FILTER, [20](#)  
 WHITE\_BALANCE, [26](#)  
 ZOOM, [27](#)  
 FOCUS  
     Enumerations, [27](#)  
 FRAMERATE\_120  
     Enumerations, [23](#)  
 FRAMERATE\_15  
     Enumerations, [22](#)  
 FRAMERATE\_1\_875  
     Enumerations, [22](#)  
 FRAMERATE\_240  
     Enumerations, [23](#)  
 FRAMERATE\_30  
     Enumerations, [22](#)  
 FRAMERATE\_3\_75  
     Enumerations, [22](#)  
 FRAMERATE\_60  
     Enumerations, [22](#)  
 FRAMERATE\_7\_5  
     Enumerations, [22](#)  
 FRAMERATE\_FORCE\_32BITS  
     Enumerations, [23](#)  
 FRAMERATE\_FORMAT7  
     Enumerations, [23](#)  
 FRAME\_RATE  
     Enumerations, [27](#)  
 FROM\_FILE\_EXT

- Enumerations, [24](#)
- FlyCapture2
  - BYTE\_ORDER\_BIG\_ENDIAN, [44](#)
  - BYTE\_ORDER\_FORCE\_32BITS, [44](#)
  - BYTE\_ORDER\_LITTLE\_ENDIAN, [44](#)
  - LINUX\_X64, [44](#)
  - LINUX\_X86, [44](#)
  - MAC, [44](#)
  - OSTYPE\_FORCE\_32BITS, [44](#)
  - UNKNOWN\_OS, [44](#)
  - WINDOWS\_X64, [44](#)
  - WINDOWS\_X86, [44](#)
- FlyCapture2::ImageStatistics
  - BLUE, [239](#)
  - GREEN, [239](#)
  - GREY, [239](#)
  - HUE, [239](#)
  - LIGHTNESS, [239](#)
  - NUM\_STATISTICS\_CHANNELS, [239](#)
  - RED, [239](#)
  - SATURATION, [239](#)
- FlyCapture2::TIFFOption
  - ADOBE\_DEFLATE, [269](#)
  - CCITTFAX3, [269](#)
  - CCITTFAX4, [270](#)
  - DEFLATE, [269](#)
  - JPEG, [270](#)
  - LZW, [270](#)
  - NONE, [269](#)
  - PACKBITS, [269](#)
- FlyCapture2::TopologyNode
  - BUS, [273](#)
  - CAMERA, [273](#)
  - COMPUTER, [273](#)
  - CONNECTED\_TO\_CHILD, [273](#)
  - CONNECTED\_TO\_PARENT, [273](#)
  - NODE, [273](#)
  - NOT\_CONNECTED, [273](#)
- GAIN
  - Enumerations, [27](#)
- GAMMA
  - Enumerations, [27](#)
- GBRG
  - Enumerations, [19](#)
- GRAB\_MODE\_FORCE\_32BITS
  - Enumerations, [23](#)
- GRAB\_TIMEOUT\_FORCE\_32BITS
  - Enumerations, [23](#)
- GRBG
  - Enumerations, [19](#)
- GREEN
  - FlyCapture2::ImageStatistics, [239](#)
- GREY
  - FlyCapture2::ImageStatistics, [239](#)
- GigE specific enumerations
  - HEARTBEAT, [29](#)
  - HEARTBEAT\_TIMEOUT, [29](#)
  - PACKET\_DELAY, [29](#)
  - PACKET\_SIZE, [29](#)
- HEARTBEAT
  - GigE specific enumerations, [29](#)
- HEARTBEAT\_TIMEOUT
  - GigE specific enumerations, [29](#)
- HQ\_LINEAR
  - Enumerations, [20](#)
- HUE
  - Enumerations, [26](#)
  - FlyCapture2::ImageStatistics, [239](#)
- IMAGE\_FILE\_FORMAT\_FORCE\_32BITS
  - Enumerations, [24](#)
- INTERFACE\_GIGE
  - Enumerations, [24](#)
- INTERFACE\_IEEE1394
  - Enumerations, [24](#)
- INTERFACE\_TYPE\_FORCE\_32BITS
  - Enumerations, [24](#)
- INTERFACE\_UNKNOWN
  - Enumerations, [24](#)
- INTERFACE\_USB2
  - Enumerations, [24](#)
- INTERFACE\_USB3
  - Enumerations, [24](#)
- IPP
  - Enumerations, [20](#)
- IRIS
  - Enumerations, [27](#)
- JPEG
  - Enumerations, [24](#)
  - FlyCapture2::TIFFOption, [270](#)
- JPEG2000
  - Enumerations, [24](#)
- LIGHTNESS
  - FlyCapture2::ImageStatistics, [239](#)
- LINUX\_X64
  - FlyCapture2, [44](#)
- LINUX\_X86
  - FlyCapture2, [44](#)

- LZW
  - FlyCapture2::TIFFOption, [270](#)
- MAC
  - FlyCapture2, [44](#)
- MODE\_0
  - Enumerations, [24](#)
- MODE\_1
  - Enumerations, [24](#)
- MODE\_10
  - Enumerations, [25](#)
- MODE\_11
  - Enumerations, [25](#)
- MODE\_12
  - Enumerations, [25](#)
- MODE\_13
  - Enumerations, [25](#)
- MODE\_14
  - Enumerations, [25](#)
- MODE\_15
  - Enumerations, [25](#)
- MODE\_16
  - Enumerations, [25](#)
- MODE\_17
  - Enumerations, [25](#)
- MODE\_18
  - Enumerations, [25](#)
- MODE\_19
  - Enumerations, [25](#)
- MODE\_2
  - Enumerations, [24](#)
- MODE\_20
  - Enumerations, [25](#)
- MODE\_21
  - Enumerations, [25](#)
- MODE\_22
  - Enumerations, [25](#)
- MODE\_23
  - Enumerations, [25](#)
- MODE\_24
  - Enumerations, [25](#)
- MODE\_25
  - Enumerations, [25](#)
- MODE\_26
  - Enumerations, [25](#)
- MODE\_27
  - Enumerations, [25](#)
- MODE\_28
  - Enumerations, [25](#)
- MODE\_29
  - Enumerations, [25](#)
- MODE\_3
  - Enumerations, [24](#)
- MODE\_30
  - Enumerations, [25](#)
- MODE\_31
  - Enumerations, [25](#)
- MODE\_4
  - Enumerations, [24](#)
- MODE\_5
  - Enumerations, [24](#)
- MODE\_6
  - Enumerations, [24](#)
- MODE\_7
  - Enumerations, [25](#)
- MODE\_8
  - Enumerations, [25](#)
- MODE\_9
  - Enumerations, [25](#)
- MODE\_FORCE\_32BITS
  - Enumerations, [25](#)
- MultiSyncLibrary
  - PGRSyncError\_ALREADY\_STARTED, [45](#)
  - PGRSyncError\_ALREADY\_STOPPED, [45](#)
  - PGRSyncError\_CAMERA\_NOT\_FOUND, [45](#)
  - PGRSyncError\_FAILED, [45](#)
  - PGRSyncError\_OK, [45](#)
  - PGRSyncError\_UNKNOWN\_ERROR, [45](#)
  - PGRSyncMessage\_BUS\_RESET, [45](#)
  - PGRSyncMessage\_DEVICE\_ERROR, [45](#)
  - PGRSyncMessage\_NOMASTER, [45](#)
  - PGRSyncMessage\_NOT\_ENOUGH\_DEVICES, [45](#)
  - PGRSyncMessage\_NOT\_INITIALIZED, [45](#)
  - PGRSyncMessage\_OK, [45](#)
  - PGRSyncMessage\_STARTED, [45](#)
  - PGRSyncMessage\_STOPPED, [45](#)
  - PGRSyncMessage\_SYNCING, [45](#)
  - PGRSyncMessage\_THREAD\_ERROR, [45](#)
  - PGRSyncMessage\_UNKNOWN\_ERROR, [45](#)
- NEAREST\_NEIGHBOR
  - Enumerations, [20](#)

- NODE
  - FlyCapture2::TopologyNode, [273](#)
- NONE
  - Enumerations, [19](#)
  - FlyCapture2::TIFFOption, [269](#)
- NOT\_CONNECTED
  - FlyCapture2::TopologyNode, [273](#)
- NO\_COLOR\_PROCESSING
  - Enumerations, [20](#)
- NUM\_FRAMERATES
  - Enumerations, [23](#)
- NUM\_MODES
  - Enumerations, [25](#)
- NUM\_PIXEL\_FORMATS
  - Enumerations, [26](#)
- NUM\_STATISTICS\_CHANNELS
  - FlyCapture2::ImageStatistics, [239](#)
- NUM\_VIDEOMODES
  - Enumerations, [28](#)
- OSTYPE\_FORCE\_32BITS
  - FlyCapture2, [44](#)
- PACKBITS
  - FlyCapture2::TIFFOption, [269](#)
- PACKET\_DELAY
  - GigE specific enumerations, [29](#)
- PACKET\_SIZE
  - GigE specific enumerations, [29](#)
- PAN
  - Enumerations, [27](#)
- PCIE\_BUSSPEED\_2\_5
  - Enumerations, [25](#)
- PCIE\_BUSSPEED\_5\_0
  - Enumerations, [25](#)
- PCIE\_BUSSPEED\_FORCE\_32BITS
  - Enumerations, [25](#)
- PCIE\_BUSSPEED\_UNKNOWN
  - Enumerations, [25](#)
- PGM
  - Enumerations, [24](#)
- PGRERROR\_BUFFER\_TOO\_SMALL
  - Enumerations, [22](#)
- PGRERROR\_BUS\_MASTER\_FAILED
  - Enumerations, [21](#)
- PGRERROR\_FAILED
  - Enumerations, [21](#)
- PGRERROR\_FAILED\_BUS\_MASTER\_CONNECTION
  - Enumerations, [21](#)
- PGRERROR\_FAILED\_GUID
  - Enumerations, [21](#)
- PGRERROR\_FORCE\_32BITS
  - Enumerations, [22](#)
- PGRERROR\_IIDC\_FAILED
  - Enumerations, [21](#)
- PGRERROR\_IMAGE\_CONSISTENCY\_ERROR
  - Enumerations, [22](#)
- PGRERROR\_IMAGE\_CONVERSION\_FAILED
  - Enumerations, [22](#)
- PGRERROR\_IMAGE\_LIBRARY\_FAILURE
  - Enumerations, [22](#)
- PGRERROR\_INCOMPATIBLE\_DRIVER
  - Enumerations, [22](#)
- PGRERROR\_INIT\_FAILED
  - Enumerations, [21](#)
- PGRERROR\_INVALID\_BUS\_MANAGER
  - Enumerations, [21](#)
- PGRERROR\_INVALID\_GENERATION
  - Enumerations, [21](#)
- PGRERROR\_INVALID\_MODE
  - Enumerations, [21](#)
- PGRERROR\_INVALID\_PACKET\_SIZE
  - Enumerations, [21](#)
- PGRERROR\_INVALID\_PARAMETER
  - Enumerations, [21](#)
- PGRERROR\_INVALID\_SETTINGS
  - Enumerations, [21](#)
- PGRERROR\_ISOCH\_ALREADY\_STARTED
  - Enumerations, [22](#)
- PGRERROR\_ISOCH\_BANDWIDTH\_EXCEEDED
  - Enumerations, [22](#)
- PGRERROR\_ISOCH\_FAILED
  - Enumerations, [22](#)
- PGRERROR\_ISOCH\_NOT\_STARTED
  - Enumerations, [22](#)
- PGRERROR\_ISOCH\_RETRIEVE\_BUFFER\_FAILED
  - Enumerations, [22](#)
- PGRERROR\_ISOCH\_START\_FAILED
  - Enumerations, [22](#)
- PGRERROR\_ISOCH\_STOP\_FAILED
  - Enumerations, [22](#)
- PGRERROR\_ISOCH\_SYNC\_FAILED
  - Enumerations, [22](#)
- PGRERROR\_LOW\_LEVEL\_FAILURE
  - Enumerations, [21](#)

- PGRERROR\_LUT\_FAILED
  - Enumerations, [21](#)
- PGRERROR\_MEMORY\_ALLOCATION\_FAILED
  - Enumerations, [21](#)
- PGRERROR\_NOT\_CONNECTED
  - Enumerations, [21](#)
- PGRERROR\_NOT\_FOUND
  - Enumerations, [21](#)
- PGRERROR\_NOT\_IMPLEMENTED
  - Enumerations, [21](#)
- PGRERROR\_NOT\_INITIALIZED
  - Enumerations, [21](#)
- PGRERROR\_NOT\_IN\_FORMAT7
  - Enumerations, [21](#)
- PGRERROR\_NOT\_SUPPORTED
  - Enumerations, [21](#)
- PGRERROR\_OK
  - Enumerations, [21](#)
- PGRERROR\_PROPERTY\_FAILED
  - Enumerations, [22](#)
- PGRERROR\_PROPERTY\_NOT\_PRESENT
  - Enumerations, [22](#)
- PGRERROR\_READ\_REGISTER\_FAILED
  - Enumerations, [22](#)
- PGRERROR\_REGISTER\_FAILED
  - Enumerations, [22](#)
- PGRERROR\_STROBE\_FAILED
  - Enumerations, [22](#)
- PGRERROR\_TIMEOUT
  - Enumerations, [21](#)
- PGRERROR\_TRIGGER\_FAILED
  - Enumerations, [22](#)
- PGRERROR\_UNDEFINED
  - Enumerations, [21](#)
- PGRERROR\_WRITE\_REGISTER\_FAILED
  - Enumerations, [22](#)
- PGRSyncError\_ALREADY\_STARTED
  - MultiSyncLibrary, [45](#)
- PGRSyncError\_ALREADY\_STOPPED
  - MultiSyncLibrary, [45](#)
- PGRSyncError\_CAMERA\_NOT\_FOUND
  - MultiSyncLibrary, [45](#)
- PGRSyncError\_FAILED
  - MultiSyncLibrary, [45](#)
- PGRSyncError\_OK
  - MultiSyncLibrary, [45](#)
- PGRSyncError\_UNKNOWN\_ERROR
  - MultiSyncLibrary, [45](#)
- PGRSyncMessage\_BUS\_RESET
  - MultiSyncLibrary, [45](#)
- PGRSyncMessage\_DEVICE\_ERROR
  - MultiSyncLibrary, [45](#)
- PGRSyncMessage\_NOMASTER
  - MultiSyncLibrary, [45](#)
- PGRSyncMessage\_NOT\_ENOUGH\_DEVICES
  - MultiSyncLibrary, [45](#)
- PGRSyncMessage\_NOT\_INITIALIZED
  - MultiSyncLibrary, [45](#)
- PGRSyncMessage\_OK
  - MultiSyncLibrary, [45](#)
- PGRSyncMessage\_STARTED
  - MultiSyncLibrary, [45](#)
- PGRSyncMessage\_STOPPED
  - MultiSyncLibrary, [45](#)
- PGRSyncMessage\_SYNCING
  - MultiSyncLibrary, [45](#)
- PGRSyncMessage\_THREAD\_ERROR
  - MultiSyncLibrary, [45](#)
- PGRSyncMessage\_UNKNOWN\_ERROR
  - MultiSyncLibrary, [45](#)
- PIXEL\_FORMAT\_411YUV8
  - Enumerations, [26](#)
- PIXEL\_FORMAT\_422YUV8
  - Enumerations, [26](#)
- PIXEL\_FORMAT\_422YUV8\_JPEG
  - Enumerations, [26](#)
- PIXEL\_FORMAT\_444YUV8
  - Enumerations, [26](#)
- PIXEL\_FORMAT\_BGR
  - Enumerations, [26](#)
- PIXEL\_FORMAT\_BGR16
  - Enumerations, [26](#)
- PIXEL\_FORMAT\_BGRU
  - Enumerations, [26](#)
- PIXEL\_FORMAT\_BGRU16
  - Enumerations, [26](#)
- PIXEL\_FORMAT\_MONO12
  - Enumerations, [26](#)
- PIXEL\_FORMAT\_MONO16
  - Enumerations, [26](#)
- PIXEL\_FORMAT\_MONO8
  - Enumerations, [26](#)
- PIXEL\_FORMAT\_RAW12
  - Enumerations, [26](#)
- PIXEL\_FORMAT\_RAW16
  - Enumerations, [26](#)



- Enumerations, [26](#)
- PIXEL\_FORMAT\_RAW8
  - Enumerations, [26](#)
- PIXEL\_FORMAT\_RGB
  - Enumerations, [26](#)
- PIXEL\_FORMAT\_RGB16
  - Enumerations, [26](#)
- PIXEL\_FORMAT\_RGB8
  - Enumerations, [26](#)
- PIXEL\_FORMAT\_RGBU
  - Enumerations, [26](#)
- PIXEL\_FORMAT\_S\_MONO16
  - Enumerations, [26](#)
- PIXEL\_FORMAT\_S\_RGB16
  - Enumerations, [26](#)
- PNG
  - Enumerations, [24](#)
- PPM
  - Enumerations, [24](#)
- PROPERTY\_TYPE\_FORCE\_32BITS
  - Enumerations, [27](#)
- PTGREY\_SCALER\_BICUBIC
  - PtGreyVideoEncoder.h, [297](#)
- PTGREY\_SCALER\_BILINEAR
  - PtGreyVideoEncoder.h, [297](#)
- PTGREY\_SCALER\_FAST\_BILINEAR
  - PtGreyVideoEncoder.h, [297](#)
- PTGREY\_SCALER\_MAX
  - PtGreyVideoEncoder.h, [297](#)
- PTGREY\_VIDEO\_H264
  - PtGreyVideoEncoder.h, [297](#)
- PTGREY\_VIDEO\_MAX
  - PtGreyVideoEncoder.h, [297](#)
- PTGREY\_VIDEO\_MJPG
  - PtGreyVideoEncoder.h, [297](#)
- PTGREY\_VIDEO\_UNCOMPRESSED
  - PtGreyVideoEncoder.h, [297](#)
- PtGreyVideoEncoder.h
  - PTGREY\_SCALER\_BICUBIC, [297](#)
  - PTGREY\_SCALER\_BILINEAR, [297](#)
  - PTGREY\_SCALER\_FAST\_BILINEAR, [297](#)
  - PTGREY\_SCALER\_MAX, [297](#)
  - PTGREY\_VIDEO\_H264, [297](#)
  - PTGREY\_VIDEO\_MAX, [297](#)
  - PTGREY\_VIDEO\_MJPG, [297](#)
  - PTGREY\_VIDEO\_UNCOMPRESSED, [297](#)
- RAW
  - Enumerations, [24](#)
- RED
  - FlyCapture2::ImageStatistics, [239](#)
- REMOVAL
  - Enumerations, [19](#)
- RGGB
  - Enumerations, [19](#)
- RIGOROUS
  - Enumerations, [20](#)
- SATURATION
  - Enumerations, [26](#)
  - FlyCapture2::ImageStatistics, [239](#)
- SHARPNESS
  - Enumerations, [26](#)
- SHUTTER
  - Enumerations, [27](#)
- TEMPERATURE
  - Enumerations, [27](#)
- TIFF
  - Enumerations, [24](#)
- TILT
  - Enumerations, [27](#)
- TIMEOUT\_INFINITE
  - Enumerations, [23](#)
- TIMEOUT\_NONE
  - Enumerations, [23](#)
- TIMEOUT\_UNSPECIFIED
  - Enumerations, [23](#)
- TRIGGER\_DELAY
  - Enumerations, [27](#)
- TRIGGER\_MODE
  - Enumerations, [27](#)
- UNKNOWN\_OS
  - FlyCapture2, [44](#)
- UNSPECIFIED\_GRAB\_MODE
  - Enumerations, [23](#)
- UNSPECIFIED\_PIXEL\_FORMAT
  - Enumerations, [26](#)
- UNSPECIFIED\_PROPERTY\_TYPE
  - Enumerations, [27](#)
- VIDEOMODE\_1024x768RGB
  - Enumerations, [27](#)
- VIDEOMODE\_1024x768Y16
  - Enumerations, [27](#)
- VIDEOMODE\_1024x768Y8
  - Enumerations, [27](#)
- VIDEOMODE\_1024x768YUV422
  - Enumerations, [27](#)
- VIDEOMODE\_1280x960RGB
  - Enumerations, [27](#)
- VIDEOMODE\_1280x960Y16

- Enumerations, [28](#)
- VIDEOMODE\_1280x960Y8
  - Enumerations, [27](#)
- VIDEOMODE\_1280x960YUV422
  - Enumerations, [27](#)
- VIDEOMODE\_1600x1200RGB
  - Enumerations, [28](#)
- VIDEOMODE\_1600x1200Y16
  - Enumerations, [28](#)
- VIDEOMODE\_1600x1200Y8
  - Enumerations, [28](#)
- VIDEOMODE\_1600x1200YUV422
  - Enumerations, [28](#)
- VIDEOMODE\_160x120YUV444
  - Enumerations, [27](#)
- VIDEOMODE\_320x240YUV422
  - Enumerations, [27](#)
- VIDEOMODE\_640x480RGB
  - Enumerations, [27](#)
- VIDEOMODE\_640x480Y16
  - Enumerations, [27](#)
- VIDEOMODE\_640x480Y8
  - Enumerations, [27](#)
- VIDEOMODE\_640x480YUV411
  - Enumerations, [27](#)
- VIDEOMODE\_640x480YUV422
  - Enumerations, [27](#)
- VIDEOMODE\_800x600RGB
  - Enumerations, [27](#)
- VIDEOMODE\_800x600Y16
  - Enumerations, [27](#)
- VIDEOMODE\_800x600Y8
  - Enumerations, [27](#)
- VIDEOMODE\_800x600YUV422
  - Enumerations, [27](#)
- VIDEOMODE\_FORCE\_32BITS
  - Enumerations, [28](#)
- VIDEOMODE\_FORMAT7
  - Enumerations, [28](#)
- WEIGHTED\_DIRECTIONAL\_FILTER
  - Enumerations, [20](#)
- WHITE\_BALANCE
  - Enumerations, [26](#)
- WINDOWS\_X64
  - FlyCapture2, [44](#)
- WINDOWS\_X86
  - FlyCapture2, [44](#)
- ZOOM
  - Enumerations, [27](#)
- AVIAppend
  - FlyCapture2::AVIRecorder, [49](#)
- AVIClose
  - FlyCapture2::AVIRecorder, [49](#)
- AVIOpen
  - FlyCapture2::AVIRecorder, [49](#), [50](#)
- AVIOption, [47](#)
  - FlyCapture2::AVIOption, [47](#)
- AVIRecorder, [48](#)
  - FlyCapture2::AVIRecorder, [48](#)
- AVIRecorder.h, [285](#)
- AddChild
  - FlyCapture2::TopologyNode, [274](#)
- AddPortType
  - FlyCapture2::TopologyNode, [274](#)
- AssignGuidToNode
  - FlyCapture2::TopologyNode, [274](#), [275](#)
- AsyncCommandCallback
  - FlyCapture2, [43](#)
- BMPOption, [51](#)
  - FlyCapture2::BMPOption, [51](#)
- BOOL
  - PtGreyVideoEncoder.h, [297](#)
- BR
  - PtGreyVideoEncoderOptions, [262](#)
- BandwidthAllocation
  - Enumerations, [18](#)
- BayerTileFormat
  - Enumerations, [18](#)
- BusCallbackType
  - Enumerations, [19](#)
- BusEventCallback
  - FlyCapture2, [43](#)
- BusManager, [51](#)
  - FlyCapture2::BusManager, [53](#)
- BusManager.h, [285](#)
- BusSpeed
  - Enumerations, [19](#)
- ByteOrder
  - FlyCapture2, [44](#)
- CalculateStatistics
  - FlyCapture2::Image, [224](#)
- CallbackHandle
  - FlyCapture2, [43](#)
- Camera, [61](#)
  - FlyCapture2::Camera, [66](#)
- Camera.h, [286](#)
- CameraBase, [91](#)
  - FlyCapture2::CameraBase, [95](#)
- CameraBase.h, [286](#)

- CameraControlDlg, 118
  - FlyCapture2::CameraControlDlg, 118
- CameraEventCallback
  - Image saving structures., 36
- CameraInfo, 120
  - FlyCapture2::CameraInfo, 122
- CameraSelectionDlg, 125
  - FlyCapture2::CameraSelectionDlg, 126
- CameraStats, 127
  - FlyCapture2::CameraStats, 128
- CheckDriver
  - FlyCapture2::Utilities, 281
- CollectSupportInformation
  - FlyCapture2::Error, 136
- ColorProcessingAlgorithm
  - Enumerations, 20
- CompressionMethod
  - FlyCapture2::TIFFOption, 269
- ConfigROM, 129
  - FlyCapture2::ConfigROM, 130
- Connect
  - FlyCapture2::Camera, 66
  - FlyCapture2::CameraBase, 96
  - FlyCapture2::CameraControlDlg, 119
  - FlyCapture2::GCCamera, 155
  - FlyCapture2::GigECamera, 182
- ConnectGUILibrary
  - FlyCap3CameraControl::FlyCapture3-ApiGuiWrapper, 145
- Convert
  - FlyCapture2::Image, 224
- DeepCopy
  - FlyCapture2::Image, 225
- DeregisterAllEvents
  - FlyCapture2::Camera, 67
  - FlyCapture2::CameraBase, 96
  - FlyCapture2::GigECamera, 182
- DeregisterEvent
  - FlyCapture2::Camera, 67
  - FlyCapture2::CameraBase, 96
  - FlyCapture2::GigECamera, 182
- DetermineBitsPerPixel
  - FlyCapture2::Image, 225
- DisableAll
  - FlyCapture2::ImageStatistics, 239
- DisableCrossPCSSynchronization
  - MultiSyncLibrary::SyncManager, 266
- Disconnect
  - FlyCapture2::Camera, 67
  - FlyCapture2::CameraBase, 96
  - FlyCapture2::CameraControlDlg, 119
  - FlyCapture2::GCCamera, 155
  - FlyCapture2::GigECamera, 182
- DisconnectGUILibrary
  - FlyCap3CameraControl::FlyCapture3-ApiGuiWrapper, 145
- DiscoverGigECameras
  - FlyCapture2::BusManager, 53
- DiscoverGigEPacketSize
  - FlyCapture2::GigECamera, 183
- DriverType
  - Enumerations, 20
- EmbeddedImageInfo, 132
- EmbeddedImageInfoProperty, 133
  - FlyCapture2::EmbeddedImageInfo-Property, 134
- EnableAll
  - FlyCapture2::ImageStatistics, 239
- EnableCrossPCSSynchronization
  - MultiSyncLibrary::SyncManager, 266
- EnableGreyOnly
  - FlyCapture2::ImageStatistics, 240
- EnableHSLOnly
  - FlyCapture2::ImageStatistics, 240
- EnableLUT
  - FlyCapture2::Camera, 67
  - FlyCapture2::CameraBase, 97
  - FlyCapture2::GCCamera, 155
  - FlyCapture2::GigECamera, 183
- EnableRGBOnly
  - FlyCapture2::ImageStatistics, 240
- Enumerations, 16
  - BandwidthAllocation, 18
  - BayerTileFormat, 18
  - BusCallbackType, 19
  - BusSpeed, 19
  - ColorProcessingAlgorithm, 20
  - DriverType, 20
  - ErrorType, 21
  - FrameRate, 22
  - GrabMode, 23
  - GrabTimeout, 23
  - ImageFileFormat, 23
  - InterfaceType, 24
  - Mode, 24
  - PCleBusSpeed, 25

- PixelFormat, 25
- PropertyType, 26
- VideoMode, 27
- Error, 134
  - FlyCapture2::Error, 135
- Error.h, 286
- ErrorType
  - Enumerations, 21
- EventCallbackData, 138
- EventCallbackFcn
  - FlyCapture2::EventOptions, 140
- EventData
  - FlyCapture2::EventCallbackData, 138
- EventDataSize
  - FlyCapture2::EventCallbackData, 138
- EventID
  - FlyCapture2::EventCallbackData, 138
- EventName
  - FlyCapture2::EventCallbackData, 139
  - FlyCapture2::EventOptions, 140
- EventOptions, 139
- EventTimestamp
  - FlyCapture2::EventCallbackData, 139
- EventUserData
  - FlyCapture2::EventCallbackData, 139
  - FlyCapture2::EventOptions, 140
- EventUserDataSize
  - FlyCapture2::EventCallbackData, 139
  - FlyCapture2::EventOptions, 140
- FALSE
  - PtGreyVideoEncoder.h, 297
- FC2Config, 140
  - FlyCapture2::FC2Config, 141
- FC2Version, 143
- FILENAME\_MAX
  - PtGreyVideoEncoder.h, 297
- FLYCAPTURE2\_API
  - FlyCapture2Platform.h, 292
- FLYCAPTURE2\_LOCAL
  - FlyCapture2Platform.h, 292
- FPS\_DEN
  - PtGreyVideoEncoderOptions, 262
- FPS\_NUM
  - PtGreyVideoEncoderOptions, 262
- FULL\_32BIT\_VALUE
  - FlyCapture2Defs.h, 292
- FireBusReset
  - FlyCapture2::BusManager, 54
- FireSoftwareTrigger
  - FlyCapture2::Camera, 67
  - FlyCapture2::CameraBase, 97
  - FlyCapture2::GCCamera, 156
  - FlyCapture2::GigECamera, 183
- FlyCap3CameraControl, 37
- FlyCap3CameraControl::FlyCapture3ApiGuiWrapper
  - ~FlyCapture3ApiGuiWrapper, 145
- ConnectGUILibrary, 145
- DisconnectGUILibrary, 145
- FlyCapture3ApiGuiWrapper, 145
- GetControlNameList, 145
- GetDialogNameList, 145
- GetNumDialogs, 145
- GetNumOfControls, 145
- ShowCameraSelectionDialog, 145
- ShowDialogByIndex, 145
- ShowDialogByName, 145
- ShowPropertyGridDialog, 145
- FlyCapture2, 37
  - AsyncCommandCallback, 43
  - BusEventCallback, 43
  - ByteOrder, 44
  - CallbackHandle, 43
  - ImageEventCallback, 43
  - OSType, 44
- FlyCapture2.h, 287
- FlyCapture2::AVIOption
  - AVIOption, 47
  - frameRate, 47
  - reserved, 47
- FlyCapture2::AVIRecorder
  - ~AVIRecorder, 48
  - AVIAppend, 49
  - AVIClose, 49
  - AVIOpen, 49, 50
  - AVIRecorder, 48
- FlyCapture2::BMPOption
  - BMPOption, 51
  - indexedColor\_8bit, 51
  - reserved, 51
- FlyCapture2::BusManager
  - ~BusManager, 53
  - BusManager, 53

- DiscoverGigECameras, [53](#)
- FireBusReset, [54](#)
- ForceAllIPAddressesAutomatically, [54](#)
- ForceIPAddressToCamera, [55](#)
- GetCameraFromIPAddress, [56](#)
- GetCameraFromIndex, [55](#)
- GetCameraFromSerialNumber, [56](#)
- GetCameraSerialNumberFromIndex, [56](#)
- GetDeviceFromIndex, [57](#)
- GetInterfaceTypeFromGuid, [57](#)
- GetNumOfCameras, [58](#)
- GetNumOfDevices, [58](#)
- GetTopology, [58](#)
- GetUsbLinkInfo, [58](#)
- GetUsbPortStatus, [59](#)
- IsCameraControlable, [59](#)
- ReadPhyRegister, [59](#)
- RegisterCallback, [60](#)
- RescanBus, [60](#)
- UnregisterCallback, [60](#)
- WritePhyRegister, [61](#)
- FlyCapture2::Camera
  - ~Camera, [66](#)
  - Camera, [66](#)
  - Connect, [66](#)
  - DeregisterAllEvents, [67](#)
  - DeregisterEvent, [67](#)
  - Disconnect, [67](#)
  - EnableLUT, [67](#)
  - FireSoftwareTrigger, [67](#)
  - GetActiveLUTBank, [68](#)
  - GetCameraInfo, [68](#)
  - GetConfiguration, [68](#)
  - GetCycleTime, [69](#)
  - GetEmbeddedImageInfo, [69](#)
  - GetFormat7Configuration, [69](#)
  - GetFormat7Info, [70](#)
  - GetGPIOPinDirection, [70](#)
  - GetLUTBankInfo, [71](#)
  - GetLUTChannel, [71](#)
  - GetLUTInfo, [72](#)
  - GetMemoryChannel, [72](#)
  - GetMemoryChannelInfo, [73](#)
  - GetProperty, [73](#)
  - GetPropertyInfo, [74](#)
  - GetRegisterString, [74](#)
  - GetStats, [74](#)
  - GetStrobe, [75](#)
  - GetStrobeInfo, [75](#)
  - GetTriggerDelay, [75](#)
  - GetTriggerDelayInfo, [76](#)
  - GetTriggerMode, [76](#)
  - GetTriggerModeInfo, [77](#)
  - GetVideoModeAndFrameRate, [77](#)
  - GetVideoModeAndFrameRateInfo, [78](#)
  - IsConnected, [78](#)
  - ReadRegister, [78](#)
  - ReadRegisterBlock, [79](#)
  - RegisterAllEvents, [79](#)
  - RegisterEvent, [79](#)
  - ResetStats, [80](#)
  - RestoreFromMemoryChannel, [80](#)
  - RetrieveBuffer, [80](#)
  - SaveToMemoryChannel, [81](#)
  - SetActiveLUTBank, [81](#)
  - SetCallback, [81](#)
  - SetConfiguration, [82](#)
  - SetEmbeddedImageInfo, [82](#)
  - SetFormat7Configuration, [83](#)
  - SetGPIOPinDirection, [83](#)
  - SetLUTChannel, [84](#)
  - SetProperty, [84](#)
  - SetStrobe, [85](#)
  - SetTriggerDelay, [85](#)
  - SetTriggerMode, [86](#)
  - SetUserBuffers, [86](#)
  - SetVideoModeAndFrameRate, [87](#)
  - StartCapture, [88](#)
  - StartSyncCapture, [88](#)
  - StopCapture, [88](#)
  - ValidateFormat7Settings, [89](#)
  - WaitForBufferEvent, [89](#)
  - WriteRegister, [90](#)
  - WriteRegisterBlock, [90](#)
- FlyCapture2::CameraBase
  - ~CameraBase, [95](#)
  - CameraBase, [95](#)
  - Connect, [96](#)
  - DeregisterAllEvents, [96](#)
  - DeregisterEvent, [96](#)
  - Disconnect, [96](#)
  - EnableLUT, [97](#)
  - FireSoftwareTrigger, [97](#)
  - GetActiveLUTBank, [97](#)
  - GetCameraInfo, [98](#)
  - GetConfiguration, [98](#)
  - GetCycleTime, [98](#)

- GetEmbeddedImageInfo, 99
- GetGPIOPinDirection, 99
- GetLUTBankInfo, 100
- GetLUTChannel, 100
- GetLUTInfo, 100
- GetMemoryChannel, 101
- GetMemoryChannelInfo, 101
- GetProperty, 102
- GetPropertyInfo, 102
- GetRegisterString, 103
- GetStats, 103
- GetStrobe, 103
- GetStrobeInfo, 104
- GetTriggerDelay, 104
- GetTriggerDelayInfo, 105
- GetTriggerMode, 105
- GetTriggerModeInfo, 106
- IsConnected, 106
- ReadRegister, 106
- ReadRegisterBlock, 107
- RegisterAllEvents, 107
- RegisterEvent, 107
- ResetStats, 107
- RestoreFromMemoryChannel, 107
- RetrieveBuffer, 108
- SaveToMemoryChannel, 108
- SetActiveLUTBank, 109
- SetCallback, 109
- SetConfiguration, 110
- SetEmbeddedImageInfo, 110
- SetGPIOPinDirection, 110
- SetLUTChannel, 111
- SetProperty, 111
- SetStrobe, 112
- SetTriggerDelay, 112
- SetTriggerMode, 113
- SetUserBuffers, 113
- StartCapture, 114
- StartSyncCapture, 115
- StopCapture, 116
- WaitForBufferEvent, 116
- WriteRegister, 116
- WriteRegisterBlock, 117
- m\_pCameraData, 117
- FlyCapture2::CameraControlDlg
  - ~CameraControlDlg, 118
  - CameraControlDlg, 118
  - Connect, 119
  - Disconnect, 119
  - Hide, 119
  - IsVisible, 119
  - SetTitle, 119
  - Show, 119
  - ShowModal, 120
- FlyCapture2::CameraInfo
  - CameraInfo, 122
  - applicationIPAddress, 122
  - applicationPort, 122
  - bayerTileFormat, 122
  - busNumber, 122
  - ccpStatus, 123
  - configROM, 123
  - defaultGateway, 123
  - driverName, 123
  - driverType, 123
  - firmwareBuildTime, 123
  - firmwareVersion, 123
  - gigEMajorVersion, 123
  - gigEMinorVersion, 123
  - iidcVer, 123
  - interfaceType, 124
  - ipAddress, 124
  - isColorCamera, 124
  - macAddress, 124
  - maximumBusSpeed, 124
  - modelName, 124
  - nodeNumber, 124
  - pcieBusSpeed, 124
  - reserved, 124
  - sensorInfo, 124
  - sensorResolution, 125
  - serialNumber, 125
  - subnetMask, 125
  - userDefinedName, 125
  - vendorName, 125
  - xmlURL1, 125
  - xmlURL2, 125
- FlyCapture2::CameraSelectionDlg
  - ~CameraSelectionDlg, 126
  - CameraSelectionDlg, 126
  - SetTitle, 126
  - ShowModal, 126
- FlyCapture2::CameraStats
  - CameraStats, 128
  - cameraCurrents, 128
  - cameraPowerUp, 128
  - cameraVoltages, 128
  - imageCorrupt, 128
  - imageDriverDropped, 128
  - imageDropped, 128

- imageXmitFailed, [128](#)
- numCurrents, [128](#)
- numResendPacketsReceived, [128](#)
- numResendPacketsRequested, [129](#)
- numVoltages, [129](#)
- portErrors, [129](#)
- regReadFailed, [129](#)
- regWriteFailed, [129](#)
- reserved, [129](#)
- temperature, [129](#)
- timeSinceBusReset, [129](#)
- timeSinceInitialization, [129](#)
- timeStamp, [129](#)
- FlyCapture2::ConfigROM
  - ConfigROM, [130](#)
  - chipIdHi, [130](#)
  - chipIdLo, [130](#)
  - nodeVendorId, [130](#)
  - pszKeyword, [131](#)
  - reserved, [131](#)
  - unitSWVer, [131](#)
  - unitSpecId, [131](#)
  - unitSubSWVer, [131](#)
  - vendorUniqueInfo\_0, [131](#)
  - vendorUniqueInfo\_1, [131](#)
  - vendorUniqueInfo\_2, [131](#)
  - vendorUniqueInfo\_3, [131](#)
- FlyCapture2::EmbeddedImageInfo
  - GPIOPinState, [133](#)
  - ROIPosition, [133](#)
  - brightness, [133](#)
  - exposure, [133](#)
  - frameCounter, [133](#)
  - gain, [133](#)
  - shutter, [133](#)
  - strobePattern, [133](#)
  - timestamp, [133](#)
  - whiteBalance, [133](#)
- FlyCapture2::EmbeddedImageInfo-Property
  - EmbeddedImageInfoProperty, [134](#)
  - available, [134](#)
  - onOff, [134](#)
- FlyCapture2::Error
  - ~Error, [135](#)
  - CollectSupportInformation, [136](#)
  - Error, [135](#)
  - GetBuildDate, [136](#)
  - GetCause, [136](#)
  - GetDescription, [136](#)
  - GetFilename, [136](#)
  - GetLine, [136](#)
  - GetType, [137](#)
  - InternalError, [138](#)
  - PrintErrorTrace, [137](#)
  - operator=, [137](#)
  - operator==, [137](#)
- FlyCapture2::EventCallbackData
  - EventData, [138](#)
  - EventDataSize, [138](#)
  - EventID, [138](#)
  - EventName, [139](#)
  - EventTimestamp, [139](#)
  - EventUserData, [139](#)
  - EventUserDataSize, [139](#)
- FlyCapture2::EventOptions
  - EventCallbackFcn, [140](#)
  - EventName, [140](#)
  - EventUserData, [140](#)
  - EventUserDataSize, [140](#)
- FlyCapture2::FC2Config
  - FC2Config, [141](#)
  - asyncBusSpeed, [141](#)
  - bandwidthAllocation, [141](#)
  - grabMode, [141](#)
  - grabTimeout, [142](#)
  - highPerformanceRetrieveBuffer, [142](#)
  - isochBusSpeed, [142](#)
  - minNumImageNotifications, [142](#)
  - numBuffers, [142](#)
  - numImageNotifications, [142](#)
  - registerTimeout, [143](#)
  - registerTimeoutRetries, [143](#)
  - reserved, [143](#)
- FlyCapture2::FC2Version
  - build, [144](#)
  - major, [144](#)
  - minor, [144](#)
  - type, [144](#)
- FlyCapture2::Format7ImageSettings
  - Format7ImageSettings, [146](#)
  - height, [146](#)
  - mode, [146](#)
  - offsetX, [146](#)
  - offsetY, [146](#)
  - pixelFormat, [146](#)
  - reserved, [146](#)
  - width, [147](#)
- FlyCapture2::Format7Info
  - Format7Info, [148](#)

- imageHStepSize, [148](#)
- imageVStepSize, [148](#)
- maxHeight, [148](#)
- maxPacketSize, [148](#)
- maxWidth, [148](#)
- minPacketSize, [148](#)
- mode, [148](#)
- offsetHStepSize, [149](#)
- offsetVStepSize, [149](#)
- packetSize, [149](#)
- percentage, [149](#)
- pixelFormatBitField, [149](#)
- reserved, [149](#)
- vendorPixelFormatBitField, [149](#)
- FlyCapture2::Format7PacketInfo
  - Format7PacketInfo, [150](#)
  - maxBytesPerPacket, [150](#)
  - recommendedBytesPerPacket, [150](#)
  - reserved, [150](#)
  - unitBytesPerPacket, [150](#)
- FlyCapture2::GCCamera
  - ~GCCamera, [155](#)
  - Connect, [155](#)
  - Disconnect, [155](#)
  - EnableLUT, [155](#)
  - FireSoftwareTrigger, [156](#)
  - GCCamera, [155](#)
  - GCCamera::GetXML, [156](#)
  - GetActiveLUTBank, [156](#)
  - GetCameraInfo, [156](#)
  - GetConfiguration, [157](#)
  - GetCycleTime, [157](#)
  - GetEmbeddedImageInfo, [157](#)
  - GetGPIOPinDirection, [158](#)
  - GetInterfaceType, [158](#)
  - GetLUTBankInfo, [158](#)
  - GetLUTChannel, [159](#)
  - GetLUTInfo, [159](#)
  - GetMemoryChannel, [160](#)
  - GetMemoryChannelInfo, [160](#)
  - GetNodeMap, [161](#)
  - GetProperty, [161](#)
  - GetPropertyInfo, [161](#)
  - GetRegisterString, [162](#)
  - GetStats, [162](#)
  - GetStrobe, [162](#)
  - GetStrobeInfo, [162](#)
  - GetTriggerDelay, [163](#)
  - GetTriggerDelayInfo, [163](#)
  - GetTriggerMode, [164](#)
  - GetTriggerModeInfo, [164](#)
  - IsConnected, [165](#)
  - ReadGVCPMemory, [165](#)
  - ReadGVCPRegister, [165](#)
  - ReadGVCPRegisterBlock, [165](#)
  - ReadRegister, [165](#)
  - ReadRegisterBlock, [166](#)
  - ResetStats, [166](#)
  - RestoreFromMemoryChannel, [166](#)
  - RetrieveBuffer, [167](#)
  - SaveToMemoryChannel, [167](#)
  - SetActiveLUTBank, [168](#)
  - SetCallback, [168](#)
  - SetCamera, [168](#), [169](#)
  - SetConfiguration, [169](#)
  - SetEmbeddedImageInfo, [169](#)
  - SetGPIOPinDirection, [169](#)
  - SetLUTChannel, [170](#)
  - SetProperty, [170](#)
  - SetStrobe, [171](#)
  - SetTriggerDelay, [171](#)
  - SetTriggerMode, [172](#)
  - SetUserBuffers, [172](#)
  - StartCapture, [173](#)
  - StartSyncCapture, [174](#)
  - StopCapture, [174](#)
  - TestGainNode, [174](#)
  - WaitForBufferEvent, [174](#)
  - WriteGVCPMemory, [175](#)
  - WriteGVCPRegister, [175](#)
  - WriteGVCPRegisterBlock, [175](#)
  - WriteRegister, [175](#)
  - WriteRegisterBlock, [175](#)
  - m\_busMgr, [176](#)
- FlyCapture2::GigECamera
  - ~GigECamera, [182](#)
  - Connect, [182](#)
  - DeregisterAllEvents, [182](#)
  - DeregisterEvent, [182](#)
  - Disconnect, [182](#)
  - DiscoverGigEPacketSize, [183](#)
  - EnableLUT, [183](#)
  - FireSoftwareTrigger, [183](#)
  - GetActiveLUTBank, [184](#)
  - GetCameraInfo, [184](#)
  - GetConfiguration, [184](#)
  - GetCycleTime, [185](#)
  - GetEmbeddedImageInfo, [185](#)
  - GetGPIOPinDirection, [188](#)
  - GetGigEConfig, [185](#)



- GetGigElImageBinningSettings, 186
- GetGigElImageSettings, 186
- GetGigElImageSettingsInfo, 186
- GetGigElImagingMode, 187
- GetGigEProperty, 187
- GetGigEStreamChannelInfo, 187
- GetLUTBankInfo, 188
- GetLUTChannel, 188
- GetLUTInfo, 189
- GetMemoryChannel, 189
- GetMemoryChannelInfo, 190
- GetNumStreamChannels, 190
- GetProperty, 191
- GetPropertyInfo, 191
- GetRegisterString, 192
- GetStats, 192
- GetStrobe, 192
- GetStrobeInfo, 193
- GetTriggerDelay, 193
- GetTriggerDelayInfo, 193
- GetTriggerMode, 194
- GetTriggerModeInfo, 194
- GigECamera, 182
- IsConnected, 195
- QueryGigElImagingMode, 195
- ReadGVCPMemory, 196
- ReadGVCPRegister, 196
- ReadGVCPRegisterBlock, 196
- ReadRegister, 197
- ReadRegisterBlock, 197
- RegisterAllEvents, 197
- RegisterEvent, 198
- ResetStats, 198
- RestoreFromMemoryChannel, 198
- RetrieveBuffer, 198
- SaveToMemoryChannel, 199
- SetActiveLUTBank, 199
- SetCallback, 199
- SetConfiguration, 200
- SetEmbeddedImageInfo, 200
- SetGPIOPinDirection, 203
- SetGigEConfig, 201
- SetGigElImageBinningSettings, 201
- SetGigElImageSettings, 201
- SetGigElImagingMode, 202
- SetGigEProperty, 202
- SetGigEStreamChannelInfo, 202
- SetLUTChannel, 203
- SetProperty, 204
- SetStrobe, 204
- SetTriggerDelay, 205
- SetTriggerMode, 205
- SetUserBuffers, 206
- StartCapture, 206
- StartSyncCapture, 207
- StopCapture, 207
- WaitForBufferEvent, 207
- WriteGVCPMemory, 208
- WriteGVCPRegister, 208
- WriteGVCPRegisterBlock, 209
- WriteRegister, 209
- WriteRegisterBlock, 209
- FlyCapture2::GigEConfig
  - GigEConfig, 211
  - enablePacketResend, 211
  - registerTimeout, 211
  - registerTimeoutRetries, 211
- FlyCapture2::GigElImageSettings
  - GigElImageSettings, 212
  - height, 212
  - offsetX, 212
  - offsetY, 212
  - pixelFormat, 212
  - reserved, 212
  - width, 212
- FlyCapture2::GigElImageSettingsInfo
  - GigElImageSettingsInfo, 213
  - imageHStepSize, 213
  - imageVStepSize, 213
  - maxHeight, 214
  - maxWidth, 214
  - offsetHStepSize, 214
  - offsetVStepSize, 214
  - pixelFormatBitField, 214
  - reserved, 214
  - vendorPixelFormatBitField, 214
- FlyCapture2::GigEProperty
  - isReadable, 215
  - isWritable, 215
  - max, 215
  - min, 215
  - propType, 215
  - value, 215
- FlyCapture2::GigEStreamChannel
  - GigEStreamChannel, 217
  - destinationIpAddress, 217
  - doNotFragment, 217
  - hostPort, 217
  - interPacketDelay, 217
  - networkInterfaceIndex, 217

- packetSize, 217
- sourcePort, 217
- FlyCapture2::H264Option
  - H264Option, 218
  - bitrate, 218
  - frameRate, 218
  - height, 219
  - reserved, 219
  - width, 219
- FlyCapture2::IPAddress
  - IPAddress, 245
  - octets, 245
  - operator==, 245
- FlyCapture2::Image
  - ~Image, 224
  - CalculateStatistics, 224
  - Convert, 224
  - DeepCopy, 225
  - DetermineBitsPerPixel, 225
  - GetBayerTileFormat, 225
  - GetBitsPerPixel, 226
  - GetBlockId, 226
  - GetColorProcessing, 226
  - GetCols, 226
  - GetData, 226, 227
  - GetDataSize, 227
  - GetDefaultColorProcessing, 227
  - GetDefaultOutputFormat, 227
  - GetDimensions, 227
  - GetMetadata, 228
  - GetPixelFormat, 228
  - GetReceivedDataSize, 228
  - GetRows, 228
  - GetStride, 229
  - GetTimeStamp, 229
  - Image, 222, 223
  - Iso, 235
  - ReleaseBuffer, 230
  - Save, 230–232
  - SetBlockId, 232
  - SetColorProcessing, 233
  - SetData, 233
  - SetDefaultColorProcessing, 233
  - SetDefaultOutputFormat, 234
  - SetDimensions, 234
  - operator(), 229
  - operator=, 229
- FlyCapture2::ImageMetadata
  - ImageMetadata, 236
  - embeddedBrightness, 236
  - embeddedExposure, 236
  - embeddedFrameCounter, 236
  - embeddedGPIOPinState, 236
  - embeddedGain, 236
  - embeddedROIPosition, 236
  - embeddedShutter, 237
  - embeddedStrobePattern, 237
  - embeddedTimeStamp, 237
  - embeddedWhiteBalance, 237
  - reserved, 237
- FlyCapture2::ImageStatistics
  - ~ImageStatistics, 239
  - DisableAll, 239
  - EnableAll, 239
  - EnableGreyOnly, 240
  - EnableHSLOnly, 240
  - EnableRGBOnly, 240
  - GetChannelStatus, 240
  - GetHistogram, 241
  - GetMean, 241
  - GetNumPixelValues, 241
  - GetPixelValueRange, 241
  - GetRange, 242
  - GetStatistics, 242
  - ImageStatistics, 239
  - ImageStatsCalculator, 243
  - SetChannelStatus, 243
  - StatisticsChannel, 239
  - operator=, 243
- FlyCapture2::Internal
  - GetInternal, 244
- FlyCapture2::JPEGOption
  - JPEGOption, 246
  - progressive, 246
  - quality, 246
  - reserved, 246
- FlyCapture2::JPG2Option
  - JPG2Option, 247
  - quality, 247
  - reserved, 247
- FlyCapture2::LUTData
  - LUTData, 248
  - enabled, 248
  - inputBitDepth, 248
  - numBanks, 248
  - numChannels, 248
  - numEntries, 248
  - outputBitDepth, 248
  - reserved, 249
  - supported, 249

- FlyCapture2::MACAddress
  - MACAddress, [249](#)
  - octets, [250](#)
  - operator==, [250](#)
- FlyCapture2::MJPGOption
  - MJPGOption, [251](#)
  - frameRate, [251](#)
  - quality, [251](#)
  - reserved, [251](#)
- FlyCapture2::NodeMap
  - ~NodeMap, [251](#)
  - NodeMap, [251](#)
  - \_GetDeviceName, [252](#)
  - \_GetNode, [252](#)
  - \_GetNodes, [252](#)
  - \_InvalidateNodes, [252](#)
  - \_Poll, [252](#)
- FlyCapture2::PGMOption
  - PGMOption, [253](#)
  - binaryFile, [253](#)
  - reserved, [253](#)
- FlyCapture2::PGRGuid
  - PGRGuid, [254](#)
  - operator==, [254](#)
  - value, [254](#)
- FlyCapture2::PNGOption
  - PNGOption, [255](#)
  - compressionLevel, [255](#)
  - interlaced, [255](#)
  - reserved, [255](#)
- FlyCapture2::PPMOption
  - PPMOption, [256](#)
  - binaryFile, [256](#)
  - reserved, [256](#)
- FlyCapture2::Property
  - Property, [257](#)
  - absControl, [257](#)
  - absValue, [257](#)
  - autoManualMode, [257](#)
  - onOff, [258](#)
  - onePush, [258](#)
  - present, [258](#)
  - reserved, [258](#)
  - type, [258](#)
  - valueA, [258](#)
  - valueB, [258](#)
- FlyCapture2::PropertyInfo
  - PropertyInfo, [260](#)
  - absMax, [260](#)
  - absMin, [260](#)
  - absValSupported, [260](#)
  - autoSupported, [260](#)
  - manualSupported, [260](#)
  - max, [260](#)
  - min, [260](#)
  - onOffSupported, [260](#)
  - onePushSupported, [260](#)
  - pUnitAbbr, [261](#)
  - pUnits, [261](#)
  - present, [260](#)
  - readOutSupported, [261](#)
  - reserved, [261](#)
  - type, [261](#)
- FlyCapture2::StrobeControl
  - StrobeControl, [263](#)
  - delay, [263](#)
  - duration, [263](#)
  - onOff, [263](#)
  - polarity, [263](#)
  - reserved, [264](#)
  - source, [264](#)
- FlyCapture2::StrobeInfo
  - StrobeInfo, [265](#)
  - maxValue, [265](#)
  - minValue, [265](#)
  - onOffSupported, [265](#)
  - polaritySupported, [265](#)
  - present, [265](#)
  - readOutSupported, [265](#)
  - reserved, [265](#)
  - source, [265](#)
- FlyCapture2::SystemInfo
  - byteOrder, [267](#)
  - cpuDescription, [267](#)
  - driverList, [268](#)
  - gpuDescription, [268](#)
  - libraryList, [268](#)
  - numCpuCores, [268](#)
  - osDescription, [268](#)
  - osType, [268](#)
  - reserved, [268](#)
  - screenHeight, [268](#)
  - screenWidth, [268](#)
  - sysMemSize, [268](#)
- FlyCapture2::TIFFOption
  - CompressionMethod, [269](#)
  - TIFFOption, [270](#)
  - compression, [270](#)
  - reserved, [270](#)
- FlyCapture2::TimeStamp

- TimeStamp, [271](#)
  - cycleCount, [271](#)
  - cycleOffset, [271](#)
  - cycleSeconds, [271](#)
  - microSeconds, [271](#)
  - reserved, [271](#)
  - seconds, [271](#)
- FlyCapture2::TopologyNode
  - ~TopologyNode, [274](#)
  - AddChild, [274](#)
  - AddPortType, [274](#)
  - AssignGuidToNode, [274](#), [275](#)
  - GetChild, [275](#)
  - GetDeviceld, [275](#)
  - GetGuid, [275](#)
  - GetInterfaceType, [276](#)
  - GetNodeType, [276](#)
  - GetNumChildren, [276](#)
  - GetNumPorts, [276](#)
  - GetPortType, [276](#)
  - NodeType, [273](#)
  - PortType, [273](#)
  - TopologyNode, [273](#), [274](#)
  - operator=, [277](#)
- FlyCapture2::TriggerMode
  - TriggerMode, [278](#)
  - mode, [278](#)
  - onOff, [278](#)
  - parameter, [278](#)
  - polarity, [278](#)
  - reserved, [278](#)
  - source, [278](#)
- FlyCapture2::TriggerModelInfo
  - TriggerModelInfo, [279](#)
  - modeMask, [279](#)
  - onOffSupported, [279](#)
  - polaritySupported, [279](#)
  - present, [280](#)
  - readOutSupported, [280](#)
  - reserved, [280](#)
  - softwareTriggerSupported, [280](#)
  - sourceMask, [280](#)
  - valueReadable, [280](#)
- FlyCapture2::Utilities
  - CheckDriver, [281](#)
  - GetDriverDeviceName, [281](#)
  - GetLibraryVersion, [282](#)
  - GetSystemInfo, [282](#)
  - LaunchBrowser, [282](#)
  - LaunchCommand, [282](#)
  - LaunchCommandAsync, [283](#)
  - LaunchHelp, [283](#)
- FlyCapture2Defs.h, [287](#)
  - FULL\_32BIT\_VALUE, [292](#)
  - NULL, [292](#)
- FlyCapture2GUI.h, [292](#)
- FlyCapture2Platform.h, [292](#)
  - FLYCAPTURE2\_API, [292](#)
  - FLYCAPTURE2\_LOCAL, [292](#)
- FlyCapture3ApiGuiWrapper, [144](#)
  - FlyCap3CameraControl::FlyCapture3-  
ApiGuiWrapper, [145](#)
- FlyCapture3ApiGuiWrapper.h, [292](#)
  - WRAPPER\_API, [293](#)
- ForceAllIPAddressesAutomatically
  - FlyCapture2::BusManager, [54](#)
- ForceIPAddressToCamera
  - FlyCapture2::BusManager, [55](#)
- Format7ImageSettings, [145](#)
  - FlyCapture2::Format7ImageSettings,  
[146](#)
- Format7Info, [147](#)
  - FlyCapture2::Format7Info, [148](#)
- Format7PacketInfo, [149](#)
  - FlyCapture2::Format7PacketInfo,  
[150](#)
- FrameRate
  - Enumerations, [22](#)
- GCCamera, [151](#)
  - FlyCapture2::GCCamera, [155](#)
- GCCamera.h, [293](#)
- GCCamera::GetXML
  - FlyCapture2::GCCamera, [156](#)
- GOB
  - PtGreyVideoEncoderOptions, [262](#)
- GPIOPinState
  - FlyCapture2::EmbeddedImageInfo,  
[133](#)
- GetActiveLUTBank
  - FlyCapture2::Camera, [68](#)
  - FlyCapture2::CameraBase, [97](#)
  - FlyCapture2::GCCamera, [156](#)
  - FlyCapture2::GigECamera, [184](#)
- GetBayerTileFormat
  - FlyCapture2::Image, [225](#)
- GetBitsPerPixel
  - FlyCapture2::Image, [226](#)
- GetBlockId
  - FlyCapture2::Image, [226](#)
- GetBuildDate

- FlyCapture2::Error, [136](#)
- GetCameraFromIPAddress
  - FlyCapture2::BusManager, [56](#)
- GetCameraFromIndex
  - FlyCapture2::BusManager, [55](#)
- GetCameraFromSerialNumber
  - FlyCapture2::BusManager, [56](#)
- GetCameraInfo
  - FlyCapture2::Camera, [68](#)
  - FlyCapture2::CameraBase, [98](#)
  - FlyCapture2::GCCamera, [156](#)
  - FlyCapture2::GigECamera, [184](#)
- GetCameraSerialNumberFromIndex
  - FlyCapture2::BusManager, [56](#)
- GetCause
  - FlyCapture2::Error, [136](#)
- GetChannelStatus
  - FlyCapture2::ImageStatistics, [240](#)
- GetChild
  - FlyCapture2::TopologyNode, [275](#)
- GetColorProcessing
  - FlyCapture2::Image, [226](#)
- GetCols
  - FlyCapture2::Image, [226](#)
- GetConfiguration
  - FlyCapture2::Camera, [68](#)
  - FlyCapture2::CameraBase, [98](#)
  - FlyCapture2::GCCamera, [157](#)
  - FlyCapture2::GigECamera, [184](#)
- GetControlNameList
  - FlyCap3CameraControl::FlyCapture3-  
ApiGuiWrapper, [145](#)
- GetCycleTime
  - FlyCapture2::Camera, [69](#)
  - FlyCapture2::CameraBase, [98](#)
  - FlyCapture2::GCCamera, [157](#)
  - FlyCapture2::GigECamera, [185](#)
- GetData
  - FlyCapture2::Image, [226](#), [227](#)
- GetDataSize
  - FlyCapture2::Image, [227](#)
- GetDefaultColorProcessing
  - FlyCapture2::Image, [227](#)
- GetDefaultOutputFormat
  - FlyCapture2::Image, [227](#)
- GetDescription
  - FlyCapture2::Error, [136](#)
- GetDeviceFromIndex
  - FlyCapture2::BusManager, [57](#)
- GetDeviceId
  - FlyCapture2::TopologyNode, [275](#)
- GetDialogNameList
  - FlyCap3CameraControl::FlyCapture3-  
ApiGuiWrapper, [145](#)
- GetDimensions
  - FlyCapture2::Image, [227](#)
- GetDriverDeviceName
  - FlyCapture2::Utilities, [281](#)
- GetEmbeddedImageInfo
  - FlyCapture2::Camera, [69](#)
  - FlyCapture2::CameraBase, [99](#)
  - FlyCapture2::GCCamera, [157](#)
  - FlyCapture2::GigECamera, [185](#)
- GetFilename
  - FlyCapture2::Error, [136](#)
- GetFormat7Configuration
  - FlyCapture2::Camera, [69](#)
- GetFormat7Info
  - FlyCapture2::Camera, [70](#)
- GetGPIOPinDirection
  - FlyCapture2::Camera, [70](#)
  - FlyCapture2::CameraBase, [99](#)
  - FlyCapture2::GCCamera, [158](#)
  - FlyCapture2::GigECamera, [188](#)
- GetGigEConfig
  - FlyCapture2::GigECamera, [185](#)
- GetGigEImageBinningSettings
  - FlyCapture2::GigECamera, [186](#)
- GetGigEImageSettings
  - FlyCapture2::GigECamera, [186](#)
- GetGigEImageSettingsInfo
  - FlyCapture2::GigECamera, [186](#)
- GetGigEImagingMode
  - FlyCapture2::GigECamera, [187](#)
- GetGigEProperty
  - FlyCapture2::GigECamera, [187](#)
- GetGigEStreamChannelInfo
  - FlyCapture2::GigECamera, [187](#)
- GetGuid
  - FlyCapture2::TopologyNode, [275](#)
- GetHistogram
  - FlyCapture2::ImageStatistics, [241](#)
- GetInterfaceType
  - FlyCapture2::GCCamera, [158](#)
  - FlyCapture2::TopologyNode, [276](#)
- GetInterfaceTypeFromGuid
  - FlyCapture2::BusManager, [57](#)
- GetInternal
  - FlyCapture2::Internal, [244](#)
- GetLUTBankInfo

- FlyCapture2::Camera, [71](#)
- FlyCapture2::CameraBase, [100](#)
- FlyCapture2::GCCamera, [158](#)
- FlyCapture2::GigECamera, [188](#)
- GetLUTChannel
  - FlyCapture2::Camera, [71](#)
  - FlyCapture2::CameraBase, [100](#)
  - FlyCapture2::GCCamera, [159](#)
  - FlyCapture2::GigECamera, [188](#)
- GetLUTInfo
  - FlyCapture2::Camera, [72](#)
  - FlyCapture2::CameraBase, [100](#)
  - FlyCapture2::GCCamera, [159](#)
  - FlyCapture2::GigECamera, [189](#)
- GetLibraryVersion
  - FlyCapture2::Utilities, [282](#)
- GetLine
  - FlyCapture2::Error, [136](#)
- GetMean
  - FlyCapture2::ImageStatistics, [241](#)
- GetMemoryChannel
  - FlyCapture2::Camera, [72](#)
  - FlyCapture2::CameraBase, [101](#)
  - FlyCapture2::GCCamera, [160](#)
  - FlyCapture2::GigECamera, [189](#)
- GetMemoryChannelInfo
  - FlyCapture2::Camera, [73](#)
  - FlyCapture2::CameraBase, [101](#)
  - FlyCapture2::GCCamera, [160](#)
  - FlyCapture2::GigECamera, [190](#)
- GetMetadata
  - FlyCapture2::Image, [228](#)
- GetNodeMap
  - FlyCapture2::GCCamera, [161](#)
- GetNodeType
  - FlyCapture2::TopologyNode, [276](#)
- GetNumChildren
  - FlyCapture2::TopologyNode, [276](#)
- GetNumDialogs
  - FlyCap3CameraControl::FlyCapture3-  
ApiGuiWrapper, [145](#)
- GetNumOfCameras
  - FlyCapture2::BusManager, [58](#)
- GetNumOfControls
  - FlyCap3CameraControl::FlyCapture3-  
ApiGuiWrapper, [145](#)
- GetNumOfDevices
  - FlyCapture2::BusManager, [58](#)
- GetNumPixelValues
  - FlyCapture2::ImageStatistics, [241](#)
- GetNumPorts
  - FlyCapture2::TopologyNode, [276](#)
- GetNumStreamChannels
  - FlyCapture2::GigECamera, [190](#)
- GetPixelFormat
  - FlyCapture2::Image, [228](#)
- GetPixelValueRange
  - FlyCapture2::ImageStatistics, [241](#)
- GetPortType
  - FlyCapture2::TopologyNode, [276](#)
- GetProperty
  - FlyCapture2::Camera, [73](#)
  - FlyCapture2::CameraBase, [102](#)
  - FlyCapture2::GCCamera, [161](#)
  - FlyCapture2::GigECamera, [191](#)
- GetPropertyInfo
  - FlyCapture2::Camera, [74](#)
  - FlyCapture2::CameraBase, [102](#)
  - FlyCapture2::GCCamera, [161](#)
  - FlyCapture2::GigECamera, [191](#)
- GetRange
  - FlyCapture2::ImageStatistics, [242](#)
- GetReceivedDataSize
  - FlyCapture2::Image, [228](#)
- GetRegisterString
  - FlyCapture2::Camera, [74](#)
  - FlyCapture2::CameraBase, [103](#)
  - FlyCapture2::GCCamera, [162](#)
  - FlyCapture2::GigECamera, [192](#)
- GetRows
  - FlyCapture2::Image, [228](#)
- GetStatistics
  - FlyCapture2::ImageStatistics, [242](#)
- GetStats
  - FlyCapture2::Camera, [74](#)
  - FlyCapture2::CameraBase, [103](#)
  - FlyCapture2::GCCamera, [162](#)
  - FlyCapture2::GigECamera, [192](#)
- GetStride
  - FlyCapture2::Image, [229](#)
- GetStrobe
  - FlyCapture2::Camera, [75](#)
  - FlyCapture2::CameraBase, [103](#)
  - FlyCapture2::GCCamera, [162](#)
  - FlyCapture2::GigECamera, [192](#)
- GetStrobeInfo
  - FlyCapture2::Camera, [75](#)
  - FlyCapture2::CameraBase, [104](#)
  - FlyCapture2::GCCamera, [162](#)
  - FlyCapture2::GigECamera, [193](#)

- GetSyncStatus
  - MultiSyncLibrary::SyncManager, 266
- GetSystemInfo
  - FlyCapture2::Utilities, 282
- GetTimeSinceSynced
  - MultiSyncLibrary::SyncManager, 266
- GetTimeStamp
  - FlyCapture2::Image, 229
- GetTopology
  - FlyCapture2::BusManager, 58
- GetTriggerDelay
  - FlyCapture2::Camera, 75
  - FlyCapture2::CameraBase, 104
  - FlyCapture2::GCCamera, 163
  - FlyCapture2::GigECamera, 193
- GetTriggerDelayInfo
  - FlyCapture2::Camera, 76
  - FlyCapture2::CameraBase, 105
  - FlyCapture2::GCCamera, 163
  - FlyCapture2::GigECamera, 193
- GetTriggerMode
  - FlyCapture2::Camera, 76
  - FlyCapture2::CameraBase, 105
  - FlyCapture2::GCCamera, 164
  - FlyCapture2::GigECamera, 194
- GetTriggerModelInfo
  - FlyCapture2::Camera, 77
  - FlyCapture2::CameraBase, 106
  - FlyCapture2::GCCamera, 164
  - FlyCapture2::GigECamera, 194
- GetType
  - FlyCapture2::Error, 137
- GetUsbLinkInfo
  - FlyCapture2::BusManager, 58
- GetUsbPortStatus
  - FlyCapture2::BusManager, 59
- GetVideoModeAndFrameRate
  - FlyCapture2::Camera, 77
- GetVideoModeAndFrameRateInfo
  - FlyCapture2::Camera, 78
- GigE specific enumerations, 29
  - GigEPropertyType, 29
- GigE specific structures, 33
- GigECamera, 176
  - FlyCapture2::GigECamera, 182
- GigECamera.h, 293
- GigEConfig, 210
  - FlyCapture2::GigEConfig, 211
- GigEImageSettings, 211
  - FlyCapture2::GigEImageSettings, 212
- GigEImageSettingsInfo, 213
  - FlyCapture2::GigEImageSettings-Info, 213
- GigEProperty, 214
- GigEPropertyType
  - GigE specific enumerations, 29
- GigEStreamChannel, 216
  - FlyCapture2::GigEStreamChannel, 217
- Global constants, 15
  - sk\_maxNumPorts, 15
  - sk\_maxStringLength, 15
- GrabMode
  - Enumerations, 23
- GrabTimeout
  - Enumerations, 23
- H264Option, 218
  - FlyCapture2::H264Option, 218
- Hide
  - FlyCapture2::CameraControlDlg, 119
- IIDC specific structures, 34
- IPAddress, 244
  - FlyCapture2::IPAddress, 245
- Image, 219
  - FlyCapture2::Image, 222, 223
- Image saving structures., 35
  - CameraEventCallback, 36
- Image.h, 294
- ImageEventCallback
  - FlyCapture2, 43
- ImageFileFormat
  - Enumerations, 23
- ImageMetadata, 235
  - FlyCapture2::ImageMetadata, 236
- ImageStatistics, 237
  - FlyCapture2::ImageStatistics, 239
- ImageStatistics.h, 294
- ImageStatsCalculator
  - FlyCapture2::ImageStatistics, 243
- InterfaceType
  - Enumerations, 24
- Internal, 244
- Internal.h, 294
- InternalError
  - FlyCapture2::Error, 138
- IsCameraControlable
  - FlyCapture2::BusManager, 59



- IsConnected
  - FlyCapture2::Camera, [78](#)
  - FlyCapture2::CameraBase, [106](#)
  - FlyCapture2::GCCamera, [165](#)
  - FlyCapture2::GigECamera, [195](#)
- IsTimingBusConnected
  - MultiSyncLibrary::SyncManager, [266](#)
- IsVisible
  - FlyCapture2::CameraControlDlg, [119](#)
- Iso
  - FlyCapture2::Image, [235](#)
- JPEGOption, [245](#)
  - FlyCapture2::JPEGOption, [246](#)
- JPG2Option, [246](#)
  - FlyCapture2::JPG2Option, [247](#)
- LUTData, [247](#)
  - FlyCapture2::LUTData, [248](#)
- LaunchBrowser
  - FlyCapture2::Utilities, [282](#)
- LaunchCommand
  - FlyCapture2::Utilities, [282](#)
- LaunchCommandAsync
  - FlyCapture2::Utilities, [283](#)
- LaunchHelp
  - FlyCapture2::Utilities, [283](#)
- Licensing.dox, [294](#)
- MACAddress, [249](#)
  - FlyCapture2::MACAddress, [249](#)
- MJPGOption, [250](#)
  - FlyCapture2::MJPGOption, [251](#)
- Mode
  - Enumerations, [24](#)
- MultiSyncLibrary, [44](#)
  - PGRSyncError, [45](#)
  - PGRSyncMessage, [45](#)
- MultiSyncLibrary.h, [294](#)
- MultiSyncLibrary::SyncManager
  - ~SyncManager, [266](#)
  - DisableCrossPCsSynchronization, [266](#)
  - EnableCrossPCsSynchronization, [266](#)
  - GetSyncStatus, [266](#)
  - GetTimeSinceSynced, [266](#)
  - IsTimingBusConnected, [266](#)
  - QueryCrossPCsSynchronization-Setting, [266](#)
  - RescanMasterTimingBus, [266](#)
  - Start, [266](#)
  - Stop, [266](#)
  - SyncManager, [266](#)
- MultiSyncLibraryDefs.h, [294](#)
- MultiSyncLibraryPlatform.h, [295](#)
- NULL
  - FlyCapture2Defs.h, [292](#)
- NodeMap, [251](#)
  - FlyCapture2::NodeMap, [251](#)
- NodeMap.h, [295](#)
- NodeType
  - FlyCapture2::TopologyNode, [273](#)
- OSType
  - FlyCapture2, [44](#)
- PCleBusSpeed
  - Enumerations, [25](#)
- PGMOption, [252](#)
  - FlyCapture2::PGMOption, [253](#)
- PGRGuid, [253](#)
  - FlyCapture2::PGRGuid, [254](#)
- PGRSyncError
  - MultiSyncLibrary, [45](#)
- PGRSyncMessage
  - MultiSyncLibrary, [45](#)
- PNGOption, [254](#)
  - FlyCapture2::PNGOption, [255](#)
- PPMOption, [255](#)
  - FlyCapture2::PPMOption, [256](#)
- PTGREY\_BR\_MIN
  - PtGreyVideoEncoder.h, [297](#)
- PTGREY\_BUF\_LEN
  - PtGreyVideoEncoder.h, [297](#)
- PixelFormat
  - Enumerations, [25](#)
- PortType
  - FlyCapture2::TopologyNode, [273](#)
- PrintErrorTrace
  - FlyCapture2::Error, [137](#)
- Property, [256](#)
  - FlyCapture2::Property, [257](#)
- PropertyInfo, [258](#)
  - FlyCapture2::PropertyInfo, [260](#)
- PropertyType
  - Enumerations, [26](#)
- PtGreyCreateVideoEncoder
  - PtGreyVideoEncoder.h, [298](#)
- PtGreyDestroyVideoEncoder
  - PtGreyVideoEncoder.h, [298](#)
- PtGreyVideoCodecFormat
  - PtGreyVideoEncoder.h, [297](#)
- PtGreyVideoEncoder.h, [296](#)
- BOOL, [297](#)



- FALSE, [297](#)
- FILENAME\_MAX, [297](#)
- PTGREY\_BR\_MIN, [297](#)
- PTGREY\_BUF\_LEN, [297](#)
- PtGreyCreateVideoEncoder, [298](#)
- PtGreyDestroyVideoEncoder, [298](#)
- PtGreyVideoCodecFormat, [297](#)
- PtGreyVideoScalerMode, [297](#)
- PtGreyWriteFrame, [298](#)
- TRUE, [297](#)
- PtGreyVideoEncoderOptions, [261](#)
  - BR, [262](#)
  - FPS\_DEN, [262](#)
  - FPS\_NUM, [262](#)
  - GOB, [262](#)
  - dstHeight, [262](#)
  - dstWidth, [262](#)
  - filename, [262](#)
  - fmt, [262](#)
  - frames, [262](#)
  - mode, [262](#)
  - srcHeight, [262](#)
  - srcWidth, [262](#)
  - stride, [262](#)
- PtGreyVideoEncoderPlatform.h, [298](#)
- PtGreyVideoScalerMode
  - PtGreyVideoEncoder.h, [297](#)
- PtGreyWriteFrame
  - PtGreyVideoEncoder.h, [298](#)
- QueryCrossPCsynchronizationSetting
  - MultiSyncLibrary::SyncManager, [266](#)
- QueryGigEImagingMode
  - FlyCapture2::GigECamera, [195](#)
- ROIPosition
  - FlyCapture2::EmbeddedImageInfo, [133](#)
- ReadGVCPMemory
  - FlyCapture2::GCCamera, [165](#)
  - FlyCapture2::GigECamera, [196](#)
- ReadGVCPRegister
  - FlyCapture2::GCCamera, [165](#)
  - FlyCapture2::GigECamera, [196](#)
- ReadGVCPRegisterBlock
  - FlyCapture2::GCCamera, [165](#)
  - FlyCapture2::GigECamera, [196](#)
- ReadPhyRegister
  - FlyCapture2::BusManager, [59](#)
- ReadRegister
  - FlyCapture2::Camera, [78](#)
  - FlyCapture2::CameraBase, [106](#)
  - FlyCapture2::GCCamera, [165](#)
  - FlyCapture2::GigECamera, [197](#)
- ReadRegisterBlock
  - FlyCapture2::Camera, [79](#)
  - FlyCapture2::CameraBase, [107](#)
  - FlyCapture2::GCCamera, [166](#)
  - FlyCapture2::GigECamera, [197](#)
- RegisterAllEvents
  - FlyCapture2::Camera, [79](#)
  - FlyCapture2::CameraBase, [107](#)
  - FlyCapture2::GigECamera, [197](#)
- RegisterCallback
  - FlyCapture2::BusManager, [60](#)
- RegisterEvent
  - FlyCapture2::Camera, [79](#)
  - FlyCapture2::CameraBase, [107](#)
  - FlyCapture2::GigECamera, [198](#)
- ReleaseBuffer
  - FlyCapture2::Image, [230](#)
- RescanBus
  - FlyCapture2::BusManager, [60](#)
- RescanMasterTimingBus
  - MultiSyncLibrary::SyncManager, [266](#)
- ResetStats
  - FlyCapture2::Camera, [80](#)
  - FlyCapture2::CameraBase, [107](#)
  - FlyCapture2::GCCamera, [166](#)
  - FlyCapture2::GigECamera, [198](#)
- RestoreFromMemoryChannel
  - FlyCapture2::Camera, [80](#)
  - FlyCapture2::CameraBase, [107](#)
  - FlyCapture2::GCCamera, [166](#)
  - FlyCapture2::GigECamera, [198](#)
- RetrieveBuffer
  - FlyCapture2::Camera, [80](#)
  - FlyCapture2::CameraBase, [108](#)
  - FlyCapture2::GCCamera, [167](#)
  - FlyCapture2::GigECamera, [198](#)
- Save
  - FlyCapture2::Image, [230–232](#)
- SaveToMemoryChannel
  - FlyCapture2::Camera, [81](#)
  - FlyCapture2::CameraBase, [108](#)
  - FlyCapture2::GCCamera, [167](#)
  - FlyCapture2::GigECamera, [199](#)
- SetActiveLUTBank
  - FlyCapture2::Camera, [81](#)
  - FlyCapture2::CameraBase, [109](#)
  - FlyCapture2::GCCamera, [168](#)
  - FlyCapture2::GigECamera, [199](#)

- SetBlockId
  - FlyCapture2::Image, [232](#)
- SetCallback
  - FlyCapture2::Camera, [81](#)
  - FlyCapture2::CameraBase, [109](#)
  - FlyCapture2::GCCamera, [168](#)
  - FlyCapture2::GigECamera, [199](#)
- SetCamera
  - FlyCapture2::GCCamera, [168](#), [169](#)
- SetChannelStatus
  - FlyCapture2::ImageStatistics, [243](#)
- SetColorProcessing
  - FlyCapture2::Image, [233](#)
- SetConfiguration
  - FlyCapture2::Camera, [82](#)
  - FlyCapture2::CameraBase, [110](#)
  - FlyCapture2::GCCamera, [169](#)
  - FlyCapture2::GigECamera, [200](#)
- SetData
  - FlyCapture2::Image, [233](#)
- SetDefaultColorProcessing
  - FlyCapture2::Image, [233](#)
- SetDefaultOutputFormat
  - FlyCapture2::Image, [234](#)
- SetDimensions
  - FlyCapture2::Image, [234](#)
- SetEmbeddedImageInfo
  - FlyCapture2::Camera, [82](#)
  - FlyCapture2::CameraBase, [110](#)
  - FlyCapture2::GCCamera, [169](#)
  - FlyCapture2::GigECamera, [200](#)
- SetFormat7Configuration
  - FlyCapture2::Camera, [83](#)
- SetGPIOPinDirection
  - FlyCapture2::Camera, [83](#)
  - FlyCapture2::CameraBase, [110](#)
  - FlyCapture2::GCCamera, [169](#)
  - FlyCapture2::GigECamera, [203](#)
- SetGigEConfig
  - FlyCapture2::GigECamera, [201](#)
- SetGigEImageBinningSettings
  - FlyCapture2::GigECamera, [201](#)
- SetGigEImageSettings
  - FlyCapture2::GigECamera, [201](#)
- SetGigEImagingMode
  - FlyCapture2::GigECamera, [202](#)
- SetGigEProperty
  - FlyCapture2::GigECamera, [202](#)
- SetGigEStreamChannelInfo
  - FlyCapture2::GigECamera, [202](#)
- SetLUTChannel
  - FlyCapture2::Camera, [84](#)
  - FlyCapture2::CameraBase, [111](#)
  - FlyCapture2::GCCamera, [170](#)
  - FlyCapture2::GigECamera, [203](#)
- SetProperty
  - FlyCapture2::Camera, [84](#)
  - FlyCapture2::CameraBase, [111](#)
  - FlyCapture2::GCCamera, [170](#)
  - FlyCapture2::GigECamera, [204](#)
- SetStrobe
  - FlyCapture2::Camera, [85](#)
  - FlyCapture2::CameraBase, [112](#)
  - FlyCapture2::GCCamera, [171](#)
  - FlyCapture2::GigECamera, [204](#)
- SetTitle
  - FlyCapture2::CameraControlDlg, [119](#)
  - FlyCapture2::CameraSelectionDlg, [126](#)
- SetTriggerDelay
  - FlyCapture2::Camera, [85](#)
  - FlyCapture2::CameraBase, [112](#)
  - FlyCapture2::GCCamera, [171](#)
  - FlyCapture2::GigECamera, [205](#)
- SetTriggerMode
  - FlyCapture2::Camera, [86](#)
  - FlyCapture2::CameraBase, [113](#)
  - FlyCapture2::GCCamera, [172](#)
  - FlyCapture2::GigECamera, [205](#)
- SetUserBuffers
  - FlyCapture2::Camera, [86](#)
  - FlyCapture2::CameraBase, [113](#)
  - FlyCapture2::GCCamera, [172](#)
  - FlyCapture2::GigECamera, [206](#)
- SetVideoModeAndFrameRate
  - FlyCapture2::Camera, [87](#)
- Show
  - FlyCapture2::CameraControlDlg, [119](#)
- ShowCameraSelectionDialog
  - FlyCap3CameraControl::FlyCapture3-ApiGuiWrapper, [145](#)
- ShowDialogByIndex
  - FlyCap3CameraControl::FlyCapture3-ApiGuiWrapper, [145](#)
- ShowDialogByName
  - FlyCap3CameraControl::FlyCapture3-ApiGuiWrapper, [145](#)
- ShowModal

- FlyCapture2::CameraControlDlg, 120
- FlyCapture2::CameraSelectionDlg, 126
- ShowPropertyGridDialog
  - FlyCap3CameraControl::FlyCapture3-ApiGuiWrapper, 145
- Start
  - MultiSyncLibrary::SyncManager, 266
- StartCapture
  - FlyCapture2::Camera, 88
  - FlyCapture2::CameraBase, 114
  - FlyCapture2::GCCamera, 173
  - FlyCapture2::GigECamera, 206
- StartSyncCapture
  - FlyCapture2::Camera, 88
  - FlyCapture2::CameraBase, 115
  - FlyCapture2::GCCamera, 174
  - FlyCapture2::GigECamera, 207
- StatisticsChannel
  - FlyCapture2::ImageStatistics, 239
- Stop
  - MultiSyncLibrary::SyncManager, 266
- StopCapture
  - FlyCapture2::Camera, 88
  - FlyCapture2::CameraBase, 116
  - FlyCapture2::GCCamera, 174
  - FlyCapture2::GigECamera, 207
- StrobeControl, 262
  - FlyCapture2::StrobeControl, 263
- StrobeInfo, 264
  - FlyCapture2::StrobeInfo, 265
- Structures, 30
  - TriggerDelay, 31
  - TriggerDelayInfo, 31
- SyncManager, 266
  - MultiSyncLibrary::SyncManager, 266
- SystemInfo, 267
- TIFFOption, 269
  - FlyCapture2::TIFFOption, 270
- TRUE
  - PtGreyVideoEncoder.h, 297
- TestGainNode
  - FlyCapture2::GCCamera, 174
- TimeStamp, 270
  - FlyCapture2::TimeStamp, 271
- TopologyNode, 272
  - FlyCapture2::TopologyNode, 273, 274
- TopologyNode.h, 298
- TriggerDelay
  - Structures, 31
- TriggerDelayInfo
  - Structures, 31
- TriggerMode, 277
  - FlyCapture2::TriggerMode, 278
- TriggerModelInfo, 278
  - FlyCapture2::TriggerModelInfo, 279
- UnregisterCallback
  - FlyCapture2::BusManager, 60
- Utilities, 280
- Utilities.h, 298
- ValidateFormat7Settings
  - FlyCapture2::Camera, 89
- VideoMode
  - Enumerations, 27
- WRAPPER\_API
  - FlyCapture3ApiGuiWrapper.h, 293
- WaitForBufferEvent
  - FlyCapture2::Camera, 89
  - FlyCapture2::CameraBase, 116
  - FlyCapture2::GCCamera, 174
  - FlyCapture2::GigECamera, 207
- WriteGVCPMemory
  - FlyCapture2::GCCamera, 175
  - FlyCapture2::GigECamera, 208
- WriteGVCPRegister
  - FlyCapture2::GCCamera, 175
  - FlyCapture2::GigECamera, 208
- WriteGVCPRegisterBlock
  - FlyCapture2::GCCamera, 175
  - FlyCapture2::GigECamera, 209
- WritePhyRegister
  - FlyCapture2::BusManager, 61
- WriteRegister
  - FlyCapture2::Camera, 90
  - FlyCapture2::CameraBase, 116
  - FlyCapture2::GCCamera, 175
  - FlyCapture2::GigECamera, 209
- WriteRegisterBlock
  - FlyCapture2::Camera, 90
  - FlyCapture2::CameraBase, 117
  - FlyCapture2::GCCamera, 175
  - FlyCapture2::GigECamera, 209
- \_GetDeviceName
  - FlyCapture2::NodeMap, 252
- \_GetNode
  - FlyCapture2::NodeMap, 252
- \_GetNodes
  - FlyCapture2::NodeMap, 252

- [\\_InvalidateNodes](#)
  - [FlyCapture2::NodeMap, 252](#)
- [\\_Poll](#)
  - [FlyCapture2::NodeMap, 252](#)
- [absControl](#)
  - [FlyCapture2::Property, 257](#)
- [absMax](#)
  - [FlyCapture2::PropertyInfo, 260](#)
- [absMin](#)
  - [FlyCapture2::PropertyInfo, 260](#)
- [absValSupported](#)
  - [FlyCapture2::PropertyInfo, 260](#)
- [absValue](#)
  - [FlyCapture2::Property, 257](#)
- [applicationIPAddress](#)
  - [FlyCapture2::CameraInfo, 122](#)
- [applicationPort](#)
  - [FlyCapture2::CameraInfo, 122](#)
- [asyncBusSpeed](#)
  - [FlyCapture2::FC2Config, 141](#)
- [autoManualMode](#)
  - [FlyCapture2::Property, 257](#)
- [autoSupported](#)
  - [FlyCapture2::PropertyInfo, 260](#)
- [available](#)
  - [FlyCapture2::EmbeddedImageInfo-Property, 134](#)
- [bandwidthAllocation](#)
  - [FlyCapture2::FC2Config, 141](#)
- [bayerTileFormat](#)
  - [FlyCapture2::CameraInfo, 122](#)
- [binaryFile](#)
  - [FlyCapture2::PGMOption, 253](#)
  - [FlyCapture2::PPMOption, 256](#)
- [bitrate](#)
  - [FlyCapture2::H264Option, 218](#)
- [brightness](#)
  - [FlyCapture2::EmbeddedImageInfo, 133](#)
- [build](#)
  - [FlyCapture2::FC2Version, 144](#)
- [busNumber](#)
  - [FlyCapture2::CameraInfo, 122](#)
- [byteOrder](#)
  - [FlyCapture2::SystemInfo, 267](#)
- [cameraCurrents](#)
  - [FlyCapture2::CameraStats, 128](#)
- [cameraPowerUp](#)
  - [FlyCapture2::CameraStats, 128](#)
- [cameraVoltages](#)
  - [FlyCapture2::CameraStats, 128](#)
- [ccpStatus](#)
  - [FlyCapture2::CameraInfo, 123](#)
- [chipIdHi](#)
  - [FlyCapture2::ConfigROM, 130](#)
- [chipIdLo](#)
  - [FlyCapture2::ConfigROM, 130](#)
- [compression](#)
  - [FlyCapture2::TIFFOption, 270](#)
- [compressionLevel](#)
  - [FlyCapture2::PNGOption, 255](#)
- [configROM](#)
  - [FlyCapture2::CameraInfo, 123](#)
- [cpuDescription](#)
  - [FlyCapture2::SystemInfo, 267](#)
- [cycleCount](#)
  - [FlyCapture2::TimeStamp, 271](#)
- [cycleOffset](#)
  - [FlyCapture2::TimeStamp, 271](#)
- [cycleSeconds](#)
  - [FlyCapture2::TimeStamp, 271](#)
- [defaultGateway](#)
  - [FlyCapture2::CameraInfo, 123](#)
- [delay](#)
  - [FlyCapture2::StrobeControl, 263](#)
- [destinationIpAddress](#)
  - [FlyCapture2::GigEStreamChannel, 217](#)
- [doNotFragment](#)
  - [FlyCapture2::GigEStreamChannel, 217](#)
- [driverList](#)
  - [FlyCapture2::SystemInfo, 268](#)
- [driverName](#)
  - [FlyCapture2::CameraInfo, 123](#)
- [driverType](#)
  - [FlyCapture2::CameraInfo, 123](#)
- [dstHeight](#)
  - [PtGreyVideoEncoderOptions, 262](#)
- [dstWidth](#)
  - [PtGreyVideoEncoderOptions, 262](#)
- [duration](#)
  - [FlyCapture2::StrobeControl, 263](#)
- [embeddedBrightness](#)
  - [FlyCapture2::ImageMetadata, 236](#)

- embeddedExposure
  - FlyCapture2::ImageMetadata, [236](#)
- embeddedFrameCounter
  - FlyCapture2::ImageMetadata, [236](#)
- embeddedGPIOPinState
  - FlyCapture2::ImageMetadata, [236](#)
- embeddedGain
  - FlyCapture2::ImageMetadata, [236](#)
- embeddedROIPosition
  - FlyCapture2::ImageMetadata, [236](#)
- embeddedShutter
  - FlyCapture2::ImageMetadata, [237](#)
- embeddedStrobePattern
  - FlyCapture2::ImageMetadata, [237](#)
- embeddedTimeStamp
  - FlyCapture2::ImageMetadata, [237](#)
- embeddedWhiteBalance
  - FlyCapture2::ImageMetadata, [237](#)
- enablePacketResend
  - FlyCapture2::GigEConfig, [211](#)
- enabled
  - FlyCapture2::LUTData, [248](#)
- exposure
  - FlyCapture2::EmbeddedImageInfo, [133](#)
- filename
  - PtGreyVideoEncoderOptions, [262](#)
- firmwareBuildTime
  - FlyCapture2::CameraInfo, [123](#)
- firmwareVersion
  - FlyCapture2::CameraInfo, [123](#)
- fmt
  - PtGreyVideoEncoderOptions, [262](#)
- frameCounter
  - FlyCapture2::EmbeddedImageInfo, [133](#)
- frameRate
  - FlyCapture2::AVIOption, [47](#)
  - FlyCapture2::H264Option, [218](#)
  - FlyCapture2::MJPGOption, [251](#)
- frames
  - PtGreyVideoEncoderOptions, [262](#)
- gain
  - FlyCapture2::EmbeddedImageInfo, [133](#)
- gigEMajorVersion
  - FlyCapture2::CameraInfo, [123](#)
- gigEMinorVersion
  - FlyCapture2::CameraInfo, [123](#)
- gpuDescription
  - FlyCapture2::SystemInfo, [268](#)
- grabMode
  - FlyCapture2::FC2Config, [141](#)
- grabTimeout
  - FlyCapture2::FC2Config, [142](#)
- height
  - FlyCapture2::Format7ImageSettings, [146](#)
  - FlyCapture2::GigEImageSettings, [212](#)
  - FlyCapture2::H264Option, [219](#)
- highPerformanceRetrieveBuffer
  - FlyCapture2::FC2Config, [142](#)
- hostPort
  - FlyCapture2::GigEStreamChannel, [217](#)
- iidcVer
  - FlyCapture2::CameraInfo, [123](#)
- imageCorrupt
  - FlyCapture2::CameraStats, [128](#)
- imageDriverDropped
  - FlyCapture2::CameraStats, [128](#)
- imageDropped
  - FlyCapture2::CameraStats, [128](#)
- imageHStepSize
  - FlyCapture2::Format7Info, [148](#)
  - FlyCapture2::GigEImageSettings-Info, [213](#)
- imageVStepSize
  - FlyCapture2::Format7Info, [148](#)
  - FlyCapture2::GigEImageSettings-Info, [213](#)
- imageXmitFailed
  - FlyCapture2::CameraStats, [128](#)
- indexedColor\_8bit
  - FlyCapture2::BMPOption, [51](#)
- inputBitDepth
  - FlyCapture2::LUTData, [248](#)
- interPacketDelay
  - FlyCapture2::GigEStreamChannel, [217](#)
- interfaceType
  - FlyCapture2::CameraInfo, [124](#)
- interlaced
  - FlyCapture2::PNGOption, [255](#)
- ipAddress

- FlyCapture2::CameraInfo, [124](#)
- isColorCamera
  - FlyCapture2::CameraInfo, [124](#)
- isReadable
  - FlyCapture2::GigEProperty, [215](#)
- isWritable
  - FlyCapture2::GigEProperty, [215](#)
- isochBusSpeed
  - FlyCapture2::FC2Config, [142](#)
- libraryList
  - FlyCapture2::SystemInfo, [268](#)
- m\_busMgr
  - FlyCapture2::GCCamera, [176](#)
- m\_pCameraData
  - FlyCapture2::CameraBase, [117](#)
- macAddress
  - FlyCapture2::CameraInfo, [124](#)
- major
  - FlyCapture2::FC2Version, [144](#)
- manualSupported
  - FlyCapture2::PropertyInfo, [260](#)
- max
  - FlyCapture2::GigEProperty, [215](#)
  - FlyCapture2::PropertyInfo, [260](#)
- maxBytesPerPacket
  - FlyCapture2::Format7PacketInfo, [150](#)
- maxHeight
  - FlyCapture2::Format7Info, [148](#)
  - FlyCapture2::GigEImageSettings-Info, [214](#)
- maxPacketSize
  - FlyCapture2::Format7Info, [148](#)
- maxValue
  - FlyCapture2::StrobeInfo, [265](#)
- maxWidth
  - FlyCapture2::Format7Info, [148](#)
  - FlyCapture2::GigEImageSettings-Info, [214](#)
- maximumBusSpeed
  - FlyCapture2::CameraInfo, [124](#)
- microSeconds
  - FlyCapture2::TimeStamp, [271](#)
- min
  - FlyCapture2::GigEProperty, [215](#)
  - FlyCapture2::PropertyInfo, [260](#)
- minNumImageNotifications
  - FlyCapture2::FC2Config, [142](#)
- minPacketSize
  - FlyCapture2::Format7Info, [148](#)
- minValue
  - FlyCapture2::StrobeInfo, [265](#)
- minor
  - FlyCapture2::FC2Version, [144](#)
- mode
  - FlyCapture2::Format7ImageSettings, [146](#)
  - FlyCapture2::Format7Info, [148](#)
  - FlyCapture2::TriggerMode, [278](#)
  - PtGreyVideoEncoderOptions, [262](#)
- modeMask
  - FlyCapture2::TriggerModelInfo, [279](#)
- modelName
  - FlyCapture2::CameraInfo, [124](#)
- networkInterfaceIndex
  - FlyCapture2::GigEStreamChannel, [217](#)
- nodeNumber
  - FlyCapture2::CameraInfo, [124](#)
- nodeVendorId
  - FlyCapture2::ConfigROM, [130](#)
- numBanks
  - FlyCapture2::LUTData, [248](#)
- numBuffers
  - FlyCapture2::FC2Config, [142](#)
- numChannels
  - FlyCapture2::LUTData, [248](#)
- numCpuCores
  - FlyCapture2::SystemInfo, [268](#)
- numCurrents
  - FlyCapture2::CameraStats, [128](#)
- numEntries
  - FlyCapture2::LUTData, [248](#)
- numImageNotifications
  - FlyCapture2::FC2Config, [142](#)
- numResendPacketsReceived
  - FlyCapture2::CameraStats, [128](#)
- numResendPacketsRequested
  - FlyCapture2::CameraStats, [129](#)
- numVoltages
  - FlyCapture2::CameraStats, [129](#)
- octets
  - FlyCapture2::IPAddress, [245](#)
  - FlyCapture2::MACAddress, [250](#)
- offsetHStepSize
  - FlyCapture2::Format7Info, [149](#)

- FlyCapture2::GigEImageSettings-Info, [214](#)
- offsetVStepSize
  - FlyCapture2::Format7Info, [149](#)
  - FlyCapture2::GigEImageSettings-Info, [214](#)
- offsetX
  - FlyCapture2::Format7ImageSettings, [146](#)
  - FlyCapture2::GigEImageSettings, [212](#)
- offsetY
  - FlyCapture2::Format7ImageSettings, [146](#)
  - FlyCapture2::GigEImageSettings, [212](#)
- onOff
  - FlyCapture2::EmbeddedImageInfo-Property, [134](#)
  - FlyCapture2::Property, [258](#)
  - FlyCapture2::StrobeControl, [263](#)
  - FlyCapture2::TriggerMode, [278](#)
- onOffSupported
  - FlyCapture2::PropertyInfo, [260](#)
  - FlyCapture2::StrobeInfo, [265](#)
  - FlyCapture2::TriggerModelInfo, [279](#)
- onePush
  - FlyCapture2::Property, [258](#)
- onePushSupported
  - FlyCapture2::PropertyInfo, [260](#)
- operator()
  - FlyCapture2::Image, [229](#)
- operator=
  - FlyCapture2::Error, [137](#)
  - FlyCapture2::Image, [229](#)
  - FlyCapture2::ImageStatistics, [243](#)
  - FlyCapture2::TopologyNode, [277](#)
- operator==
  - FlyCapture2::Error, [137](#)
  - FlyCapture2::IPAddress, [245](#)
  - FlyCapture2::MACAddress, [250](#)
  - FlyCapture2::PGRGuid, [254](#)
- osDescription
  - FlyCapture2::SystemInfo, [268](#)
- osType
  - FlyCapture2::SystemInfo, [268](#)
- outputBitDepth
  - FlyCapture2::LUTData, [248](#)
- pUnitAbbr
  - FlyCapture2::PropertyInfo, [261](#)
- pUnits
  - FlyCapture2::PropertyInfo, [261](#)
- packetSize
  - FlyCapture2::Format7Info, [149](#)
  - FlyCapture2::GigEStreamChannel, [217](#)
- parameter
  - FlyCapture2::TriggerMode, [278](#)
- pcieBusSpeed
  - FlyCapture2::CameraInfo, [124](#)
- percentage
  - FlyCapture2::Format7Info, [149](#)
- pixelFormat
  - FlyCapture2::Format7ImageSettings, [146](#)
  - FlyCapture2::GigEImageSettings, [212](#)
- pixelFormatBitField
  - FlyCapture2::Format7Info, [149](#)
  - FlyCapture2::GigEImageSettings-Info, [214](#)
- polarity
  - FlyCapture2::StrobeControl, [263](#)
  - FlyCapture2::TriggerMode, [278](#)
- polaritySupported
  - FlyCapture2::StrobeInfo, [265](#)
  - FlyCapture2::TriggerModelInfo, [279](#)
- portErrors
  - FlyCapture2::CameraStats, [129](#)
- present
  - FlyCapture2::Property, [258](#)
  - FlyCapture2::PropertyInfo, [260](#)
  - FlyCapture2::StrobeInfo, [265](#)
  - FlyCapture2::TriggerModelInfo, [280](#)
- progressive
  - FlyCapture2::JPEGOption, [246](#)
- propType
  - FlyCapture2::GigEProperty, [215](#)
- pszKeyword
  - FlyCapture2::ConfigROM, [131](#)
- quality
  - FlyCapture2::JPEGOption, [246](#)
  - FlyCapture2::JPG2Option, [247](#)
  - FlyCapture2::MJPGOption, [251](#)
- readOutSupported
  - FlyCapture2::PropertyInfo, [261](#)
  - FlyCapture2::StrobeInfo, [265](#)



- FlyCapture2::TriggerModelInfo, [280](#)
- recommendedBytesPerPacket
  - FlyCapture2::Format7PacketInfo, [150](#)
- regReadFailed
  - FlyCapture2::CameraStats, [129](#)
- regWriteFailed
  - FlyCapture2::CameraStats, [129](#)
- registerTimeout
  - FlyCapture2::FC2Config, [143](#)
  - FlyCapture2::GigEConfig, [211](#)
- registerTimeoutRetries
  - FlyCapture2::FC2Config, [143](#)
  - FlyCapture2::GigEConfig, [211](#)
- reserved
  - FlyCapture2::AVIOption, [47](#)
  - FlyCapture2::BMPOption, [51](#)
  - FlyCapture2::CameraInfo, [124](#)
  - FlyCapture2::CameraStats, [129](#)
  - FlyCapture2::ConfigROM, [131](#)
  - FlyCapture2::FC2Config, [143](#)
  - FlyCapture2::Format7ImageSettings, [146](#)
  - FlyCapture2::Format7Info, [149](#)
  - FlyCapture2::Format7PacketInfo, [150](#)
  - FlyCapture2::GigEImageSettings, [212](#)
  - FlyCapture2::GigEImageSettings-Info, [214](#)
  - FlyCapture2::H264Option, [219](#)
  - FlyCapture2::ImageMetadata, [237](#)
  - FlyCapture2::JPEGOption, [246](#)
  - FlyCapture2::JPG2Option, [247](#)
  - FlyCapture2::LUTData, [249](#)
  - FlyCapture2::MJPGOption, [251](#)
  - FlyCapture2::PGMOption, [253](#)
  - FlyCapture2::PNGOption, [255](#)
  - FlyCapture2::PPMOption, [256](#)
  - FlyCapture2::Property, [258](#)
  - FlyCapture2::PropertyInfo, [261](#)
  - FlyCapture2::StrobeControl, [264](#)
  - FlyCapture2::StrobeInfo, [265](#)
  - FlyCapture2::SystemInfo, [268](#)
  - FlyCapture2::TIFFOption, [270](#)
  - FlyCapture2::TimeStamp, [271](#)
  - FlyCapture2::TriggerMode, [278](#)
  - FlyCapture2::TriggerModelInfo, [280](#)
- screenHeight
  - FlyCapture2::SystemInfo, [268](#)
- screenWidth
  - FlyCapture2::SystemInfo, [268](#)
- seconds
  - FlyCapture2::TimeStamp, [271](#)
- sensorInfo
  - FlyCapture2::CameraInfo, [124](#)
- sensorResolution
  - FlyCapture2::CameraInfo, [125](#)
- serialNumber
  - FlyCapture2::CameraInfo, [125](#)
- shutter
  - FlyCapture2::EmbeddedImageInfo, [133](#)
- sk\_maxNumPorts
  - Global constants, [15](#)
- sk\_maxStringLength
  - Global constants, [15](#)
- softwareTriggerSupported
  - FlyCapture2::TriggerModelInfo, [280](#)
- source
  - FlyCapture2::StrobeControl, [264](#)
  - FlyCapture2::StrobeInfo, [265](#)
  - FlyCapture2::TriggerMode, [278](#)
- sourceMask
  - FlyCapture2::TriggerModelInfo, [280](#)
- sourcePort
  - FlyCapture2::GigEStreamChannel, [217](#)
- srcHeight
  - PtGreyVideoEncoderOptions, [262](#)
- srcWidth
  - PtGreyVideoEncoderOptions, [262](#)
- stride
  - PtGreyVideoEncoderOptions, [262](#)
- strobePattern
  - FlyCapture2::EmbeddedImageInfo, [133](#)
- subnetMask
  - FlyCapture2::CameraInfo, [125](#)
- supported
  - FlyCapture2::LUTData, [249](#)
- sysMemSize
  - FlyCapture2::SystemInfo, [268](#)
- temperature
  - FlyCapture2::CameraStats, [129](#)
- timeSinceBusReset
  - FlyCapture2::CameraStats, [129](#)
- timeSinceInitialization



- FlyCapture2::CameraStats, [129](#)
- timeStamp
  - FlyCapture2::CameraStats, [129](#)
- timestamp
  - FlyCapture2::EmbeddedImageInfo, [133](#)
- type
  - FlyCapture2::FC2Version, [144](#)
  - FlyCapture2::Property, [258](#)
  - FlyCapture2::PropertyInfo, [261](#)
- unitBytesPerPacket
  - FlyCapture2::Format7PacketInfo, [150](#)
- unitSWVer
  - FlyCapture2::ConfigROM, [131](#)
- unitSpecId
  - FlyCapture2::ConfigROM, [131](#)
- unitSubSWVer
  - FlyCapture2::ConfigROM, [131](#)
- userDefinedName
  - FlyCapture2::CameraInfo, [125](#)
- value
  - FlyCapture2::GigEProperty, [215](#)
  - FlyCapture2::PGRGuid, [254](#)
- valueA
  - FlyCapture2::Property, [258](#)
- valueB
  - FlyCapture2::Property, [258](#)
- valueReadable
  - FlyCapture2::TriggerModelInfo, [280](#)
- vendorName
  - FlyCapture2::CameraInfo, [125](#)
- vendorPixelFormatBitField
  - FlyCapture2::Format7Info, [149](#)
  - FlyCapture2::GigEImageSettings-Info, [214](#)
- vendorUniqueInfo\_0
  - FlyCapture2::ConfigROM, [131](#)
- vendorUniqueInfo\_1
  - FlyCapture2::ConfigROM, [131](#)
- vendorUniqueInfo\_2
  - FlyCapture2::ConfigROM, [131](#)
- vendorUniqueInfo\_3
  - FlyCapture2::ConfigROM, [131](#)
- whiteBalance
  - FlyCapture2::EmbeddedImageInfo, [133](#)
- width
  - FlyCapture2::Format7ImageSettings, [147](#)
  - FlyCapture2::GigEImageSettings, [212](#)
  - FlyCapture2::H264Option, [219](#)
- xmlURL1
  - FlyCapture2::CameraInfo, [125](#)
- xmlURL2
  - FlyCapture2::CameraInfo, [125](#)