

FlyCapture2

2.13.3.61

Generated by Doxygen 1.7.5

Wed Apr 3 2019 19:06:42

Contents

1	Software Licensing Information	1
2	Module Index	3
2.1	Modules	3
3	Namespace Index	5
3.1	Namespace List	5
4	Class Index	7
4.1	Class Hierarchy	7
5	Class Index	9
5.1	Class List	9
6	File Index	13
6.1	File List	13
7	Module Documentation	15
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	I IDC specific structures	34
7.6.1	Detailed Description	34
7.7	Image saving structures.	35
7.7.1	Detailed Description	35
7.7.2	Typedef Documentation	36
7.7.2.1	CameraEventCallback	36
7.8	Video saving structures.	37
7.8.1	Detailed Description	37
8	Namespace Documentation	39
8.1	FlyCap3CameraControl Namespace Reference	39

8.2	FlyCapture2 Namespace Reference	39
8.2.1	Typedef Documentation	45
8.2.1.1	AsyncCommandCallback	45
8.2.1.2	BusEventCallback	45
8.2.1.3	CallbackHandle	45
8.2.1.4	ImageEventCallback	46
8.2.2	Enumeration Type Documentation	46
8.2.2.1	ByteOrder	46
8.2.2.2	OSType	46
8.3	MultiSyncLibrary Namespace Reference	46
8.3.1	Enumeration Type Documentation	47
8.3.1.1	PGRSyncError	47
8.3.1.2	PGRSyncMessage	47
9	Class Documentation	49
9.1	AVIOption Struct Reference	49
9.1.1	Detailed Description	49
9.1.2	Constructor & Destructor Documentation	49
9.1.2.1	AVIOption	49
9.1.3	Member Data Documentation	49
9.1.3.1	frameRate	49
9.1.3.2	reserved	50
9.2	BMPOption Struct Reference	50
9.2.1	Detailed Description	50
9.2.2	Constructor & Destructor Documentation	50
9.2.2.1	BMPOption	50
9.2.3	Member Data Documentation	50
9.2.3.1	indexedColor_8bit	50
9.2.3.2	reserved	50
9.3	BusManager Class Reference	51
9.3.1	Detailed Description	52
9.3.2	Constructor & Destructor Documentation	52
9.3.2.1	BusManager	52
9.3.2.2	~BusManager	53

9.3.3	Member Function Documentation	53
9.3.3.1	DiscoverGigECameras	53
9.3.3.2	FireBusReset	53
9.3.3.3	ForceAllIPAddressesAutomatically	53
9.3.3.4	ForceAllIPAddressesAutomatically	54
9.3.3.5	ForceIPAddressToCamera	54
9.3.3.6	GetCameraFromIndex	54
9.3.3.7	GetCameraFromIPAddress	55
9.3.3.8	GetCameraFromSerialNumber	55
9.3.3.9	GetCameraSerialNumberFromIndex	56
9.3.3.10	GetDeviceFromIndex	56
9.3.3.11	GetInterfaceTypeFromGuid	56
9.3.3.12	GetNumOfCameras	57
9.3.3.13	GetNumOfDevices	57
9.3.3.14	GetTopology	57
9.3.3.15	GetUsbLinkInfo	58
9.3.3.16	GetUsbPortStatus	58
9.3.3.17	IsCameraControlable	60
9.3.3.18	ReadPhyRegister	60
9.3.3.19	RegisterCallback	60
9.3.3.20	RescanBus	61
9.3.3.21	UnregisterCallback	61
9.3.3.22	WritePhyRegister	62
9.4	Camera Class Reference	62
9.4.1	Detailed Description	67
9.4.2	Constructor & Destructor Documentation	67
9.4.2.1	Camera	67
9.4.2.2	~Camera	67
9.4.3	Member Function Documentation	67
9.4.3.1	Connect	67
9.4.3.2	DeregisterAllEvents	67
9.4.3.3	DeregisterEvent	67
9.4.3.4	Disconnect	67
9.4.3.5	EnableLUT	68

9.4.3.6	FireSoftwareTrigger	68
9.4.3.7	GetActiveLUTBank	68
9.4.3.8	GetCameraInfo	69
9.4.3.9	GetConfiguration	69
9.4.3.10	GetCycleTime	69
9.4.3.11	GetEmbeddedImageInfo	70
9.4.3.12	GetFormat7Configuration	70
9.4.3.13	GetFormat7Info	71
9.4.3.14	GetGPIOPinDirection	71
9.4.3.15	GetLUTBankInfo	72
9.4.3.16	GetLUTChannel	72
9.4.3.17	GetLUTInfo	73
9.4.3.18	GetMemoryChannel	73
9.4.3.19	GetMemoryChannelInfo	73
9.4.3.20	GetProperty	74
9.4.3.21	GetPropertyInfo	74
9.4.3.22	GetRegisterString	75
9.4.3.23	GetStats	75
9.4.3.24	GetStrobe	75
9.4.3.25	GetStrobeInfo	76
9.4.3.26	GetTriggerDelay	76
9.4.3.27	GetTriggerDelayInfo	77
9.4.3.28	GetTriggerMode	77
9.4.3.29	GetTriggerModeInfo	78
9.4.3.30	GetVideoModeAndFrameRate	78
9.4.3.31	GetVideoModeAndFrameRateInfo	79
9.4.3.32	IsConnected	79
9.4.3.33	ReadRegister	79
9.4.3.34	ReadRegisterBlock	80
9.4.3.35	RegisterAllEvents	80
9.4.3.36	RegisterEvent	80
9.4.3.37	ResetStats	80
9.4.3.38	RestoreFromMemoryChannel	81
9.4.3.39	RetrieveBuffer	81

9.4.3.40	SaveToMemoryChannel	81
9.4.3.41	SetActiveLUTBank	82
9.4.3.42	SetCallback	82
9.4.3.43	SetConfiguration	83
9.4.3.44	SetEmbeddedImageInfo	83
9.4.3.45	SetFormat7Configuration	83
9.4.3.46	SetFormat7Configuration	84
9.4.3.47	SetGPIOPinDirection	84
9.4.3.48	SetLUTChannel	85
9.4.3.49	SetProperty	85
9.4.3.50	SetStrobe	86
9.4.3.51	SetTriggerDelay	86
9.4.3.52	SetTriggerMode	87
9.4.3.53	SetUserBuffers	87
9.4.3.54	SetVideoModeAndFrameRate	88
9.4.3.55	StartCapture	89
9.4.3.56	StartSyncCapture	89
9.4.3.57	StopCapture	89
9.4.3.58	ValidateFormat7Settings	90
9.4.3.59	WaitForBufferEvent	90
9.4.3.60	WriteRegister	91
9.4.3.61	WriteRegisterBlock	91
9.5	CameraBase Class Reference	92
9.5.1	Detailed Description	96
9.5.2	Constructor & Destructor Documentation	96
9.5.2.1	CameraBase	96
9.5.2.2	~CameraBase	97
9.5.3	Member Function Documentation	97
9.5.3.1	Connect	97
9.5.3.2	DeregisterAllEvents	97
9.5.3.3	DeregisterEvent	97
9.5.3.4	Disconnect	97
9.5.3.5	EnableLUT	98
9.5.3.6	FireSoftwareTrigger	98

9.5.3.7	GetActiveLUTBank	98
9.5.3.8	GetCameraInfo	99
9.5.3.9	GetConfiguration	99
9.5.3.10	GetCycleTime	99
9.5.3.11	GetEmbeddedImageInfo	100
9.5.3.12	GetGPIOPinDirection	100
9.5.3.13	GetLUTBankInfo	101
9.5.3.14	GetLUTChannel	101
9.5.3.15	GetLUTInfo	102
9.5.3.16	GetMemoryChannel	102
9.5.3.17	GetMemoryChannelInfo	102
9.5.3.18	GetProperty	103
9.5.3.19	GetPropertyInfo	103
9.5.3.20	GetRegisterString	104
9.5.3.21	GetStats	104
9.5.3.22	GetStrobe	104
9.5.3.23	GetStrobeInfo	105
9.5.3.24	GetTriggerDelay	105
9.5.3.25	GetTriggerDelayInfo	106
9.5.3.26	GetTriggerMode	106
9.5.3.27	GetTriggerModeInfo	107
9.5.3.28	IsConnected	107
9.5.3.29	ReadRegister	107
9.5.3.30	ReadRegisterBlock	108
9.5.3.31	RegisterAllEvents	108
9.5.3.32	RegisterEvent	108
9.5.3.33	ResetStats	108
9.5.3.34	RestoreFromMemoryChannel	109
9.5.3.35	RetrieveBuffer	109
9.5.3.36	SaveToMemoryChannel	110
9.5.3.37	SetActiveLUTBank	110
9.5.3.38	SetCallback	110
9.5.3.39	SetConfiguration	111
9.5.3.40	SetEmbeddedImageInfo	111

9.5.3.41	SetGPIOPinDirection	112
9.5.3.42	SetLUTChannel	112
9.5.3.43	SetProperty	113
9.5.3.44	SetStrobe	113
9.5.3.45	SetTriggerDelay	114
9.5.3.46	SetTriggerMode	114
9.5.3.47	SetUserBuffers	115
9.5.3.48	StartCapture	115
9.5.3.49	StartSyncCapture	116
9.5.3.50	StopCapture	117
9.5.3.51	WaitForBufferEvent	117
9.5.3.52	WriteRegister	117
9.5.3.53	WriteRegisterBlock	118
9.5.4	Member Data Documentation	118
9.5.4.1	m_pCameraData	118
9.6	CameraControlDlg Class Reference	119
9.6.1	Detailed Description	119
9.6.2	Constructor & Destructor Documentation	119
9.6.2.1	CameraControlDlg	119
9.6.2.2	~CameraControlDlg	120
9.6.3	Member Function Documentation	120
9.6.3.1	Connect	120
9.6.3.2	Disconnect	120
9.6.3.3	Hide	120
9.6.3.4	IsVisible	120
9.6.3.5	SetTitle	120
9.6.3.6	Show	120
9.6.3.7	Show	121
9.6.3.8	ShowModal	121
9.6.3.9	ShowModal	121
9.7	CameraInfo Struct Reference	121
9.7.1	Detailed Description	123
9.7.2	Constructor & Destructor Documentation	123
9.7.2.1	CameraInfo	123

9.7.3	Member Data Documentation	123
9.7.3.1	applicationIPAddress	123
9.7.3.2	applicationPort	123
9.7.3.3	bayerTileFormat	123
9.7.3.4	busNumber	124
9.7.3.5	ccpStatus	124
9.7.3.6	configROM	124
9.7.3.7	defaultGateway	124
9.7.3.8	driverName	124
9.7.3.9	driverType	124
9.7.3.10	firmwareBuildTime	124
9.7.3.11	firmwareVersion	124
9.7.3.12	gigEMajorVersion	124
9.7.3.13	gigEMinorVersion	124
9.7.3.14	iidcVer	125
9.7.3.15	interfaceType	125
9.7.3.16	ipAddress	125
9.7.3.17	isColorCamera	125
9.7.3.18	macAddress	125
9.7.3.19	maximumBusSpeed	125
9.7.3.20	modelName	125
9.7.3.21	nodeNumber	125
9.7.3.22	pcieBusSpeed	125
9.7.3.23	reserved	125
9.7.3.24	sensorInfo	126
9.7.3.25	sensorResolution	126
9.7.3.26	serialNumber	126
9.7.3.27	subnetMask	126
9.7.3.28	userDefinedName	126
9.7.3.29	vendorName	126
9.7.3.30	xmlURL1	126
9.7.3.31	xmlURL2	126
9.8	CameraSelectionDlg Class Reference	126
9.8.1	Detailed Description	127

9.8.2	Constructor & Destructor Documentation	127
9.8.2.1	CameraSelectionDlg	127
9.8.2.2	~CameraSelectionDlg	127
9.8.3	Member Function Documentation	127
9.8.3.1	SetTitle	127
9.8.3.2	ShowModal	127
9.9	CameraStats Struct Reference	128
9.9.1	Detailed Description	129
9.9.2	Constructor & Destructor Documentation	129
9.9.2.1	CameraStats	129
9.9.3	Member Data Documentation	129
9.9.3.1	cameraCurrents	129
9.9.3.2	cameraPowerUp	129
9.9.3.3	cameraVoltages	129
9.9.3.4	imageCorrupt	129
9.9.3.5	imageDriverDropped	129
9.9.3.6	imageDropped	129
9.9.3.7	imageXmitFailed	129
9.9.3.8	numCurrents	129
9.9.3.9	numResendPacketsReceived	130
9.9.3.10	numResendPacketsRequested	130
9.9.3.11	numVoltages	130
9.9.3.12	portErrors	130
9.9.3.13	regReadFailed	130
9.9.3.14	regWriteFailed	130
9.9.3.15	reserved	130
9.9.3.16	temperature	130
9.9.3.17	timeSinceBusReset	130
9.9.3.18	timeSinceInitialization	130
9.9.3.19	timeStamp	130
9.10	ConfigROM Struct Reference	130
9.10.1	Detailed Description	131
9.10.2	Constructor & Destructor Documentation	131
9.10.2.1	ConfigROM	131

9.10.3	Member Data Documentation	131
9.10.3.1	chipIdHi	131
9.10.3.2	chipIdLo	131
9.10.3.3	nodeVendorId	132
9.10.3.4	pszKeyword	132
9.10.3.5	reserved	132
9.10.3.6	unitSpecId	132
9.10.3.7	unitSubSWVer	132
9.10.3.8	unitSWVer	132
9.10.3.9	vendorUniqueInfo_0	132
9.10.3.10	vendorUniqueInfo_1	132
9.10.3.11	vendorUniqueInfo_2	132
9.10.3.12	vendorUniqueInfo_3	132
9.11	EmbeddedImageInfo Struct Reference	133
9.11.1	Detailed Description	133
9.11.2	Member Data Documentation	134
9.11.2.1	brightness	134
9.11.2.2	exposure	134
9.11.2.3	frameCounter	134
9.11.2.4	gain	134
9.11.2.5	GPIOPinState	134
9.11.2.6	ROIPosition	134
9.11.2.7	shutter	134
9.11.2.8	strobePattern	134
9.11.2.9	timestamp	134
9.11.2.10	whiteBalance	134
9.12	EmbeddedImageInfoProperty Struct Reference	134
9.12.1	Detailed Description	134
9.12.2	Constructor & Destructor Documentation	135
9.12.2.1	EmbeddedImageInfoProperty	135
9.12.3	Member Data Documentation	135
9.12.3.1	available	135
9.12.3.2	onOff	135
9.13	Error Class Reference	135

9.13.1 Detailed Description	136
9.13.2 Constructor & Destructor Documentation	136
9.13.2.1 Error	136
9.13.2.2 Error	136
9.13.2.3 ~Error	136
9.13.3 Member Function Documentation	137
9.13.3.1 CollectSupportInformation	137
9.13.3.2 GetBuildDate	137
9.13.3.3 GetCause	137
9.13.3.4 GetDescription	137
9.13.3.5 GetFilename	137
9.13.3.6 GetLine	138
9.13.3.7 GetType	138
9.13.3.8 operator!=	138
9.13.3.9 operator!=	138
9.13.3.10 operator=	138
9.13.3.11 operator==	138
9.13.3.12 operator==	138
9.13.3.13 PrintErrorTrace	138
9.13.4 Friends And Related Function Documentation	139
9.13.4.1 InternalError	139
9.14 EventCallbackData Struct Reference	139
9.14.1 Member Data Documentation	139
9.14.1.1 eventData	139
9.14.1.2 eventDataSize	139
9.14.1.3 eventId	140
9.14.1.4 eventName	140
9.14.1.5 eventTimestamp	140
9.14.1.6 eventUserData	140
9.14.1.7 eventUserDataSize	140
9.15 EventOptions Struct Reference	140
9.15.1 Detailed Description	141
9.15.2 Member Data Documentation	141
9.15.2.1 EventCallbackFcn	141

9.15.2.2	EventName	141
9.15.2.3	EventUserData	141
9.15.2.4	EventUserDataSize	141
9.16	FC2Config Struct Reference	141
9.16.1	Detailed Description	142
9.16.2	Constructor & Destructor Documentation	142
9.16.2.1	FC2Config	142
9.16.3	Member Data Documentation	142
9.16.3.1	asyncBusSpeed	142
9.16.3.2	bandwidthAllocation	142
9.16.3.3	grabMode	143
9.16.3.4	grabTimeout	143
9.16.3.5	highPerformanceRetrieveBuffer	143
9.16.3.6	isochBusSpeed	143
9.16.3.7	minNumImageNotifications	143
9.16.3.8	numBuffers	143
9.16.3.9	numImageNotifications	143
9.16.3.10	registerTimeout	144
9.16.3.11	registerTimeoutRetries	144
9.16.3.12	reserved	144
9.17	FC2Version Struct Reference	144
9.17.1	Detailed Description	145
9.17.2	Member Data Documentation	145
9.17.2.1	build	145
9.17.2.2	major	145
9.17.2.3	minor	145
9.17.2.4	type	145
9.18	FlyCapture2Video Class Reference	145
9.18.1	Detailed Description	146
9.18.2	Constructor & Destructor Documentation	146
9.18.2.1	FlyCapture2Video	146
9.18.2.2	~FlyCapture2Video	146
9.18.3	Member Function Documentation	146
9.18.3.1	Append	146

9.18.3.2	Close	146
9.18.3.3	Open	147
9.18.3.4	Open	147
9.18.3.5	Open	148
9.18.3.6	SetMaximumFileSize	148
9.19	FlyCapture3ApiGuiWrapper Class Reference	148
9.19.1	Constructor & Destructor Documentation	149
9.19.1.1	FlyCapture3ApiGuiWrapper	149
9.19.1.2	~FlyCapture3ApiGuiWrapper	149
9.19.2	Member Function Documentation	149
9.19.2.1	ConnectGUILibrary	149
9.19.2.2	DisconnectGUILibrary	149
9.19.2.3	GetControlNameList	149
9.19.2.4	GetDialogNameList	149
9.19.2.5	GetNumDialogs	149
9.19.2.6	GetNumOfControls	149
9.19.2.7	ShowCameraSelectionDialog	149
9.19.2.8	ShowDialogByIndex	149
9.19.2.9	ShowDialogByName	149
9.19.2.10	ShowPropertyGridDialog	149
9.20	Format7ImageSettings Struct Reference	149
9.20.1	Detailed Description	150
9.20.2	Constructor & Destructor Documentation	150
9.20.2.1	Format7ImageSettings	150
9.20.3	Member Data Documentation	150
9.20.3.1	height	150
9.20.3.2	mode	150
9.20.3.3	offsetX	150
9.20.3.4	offsetY	151
9.20.3.5	pixelFormat	151
9.20.3.6	reserved	151
9.20.3.7	width	151
9.21	Format7Info Struct Reference	151
9.21.1	Detailed Description	152

9.21.2	Constructor & Destructor Documentation	152
9.21.2.1	Format7Info	152
9.21.3	Member Data Documentation	152
9.21.3.1	imageHStepSize	152
9.21.3.2	imageVStepSize	152
9.21.3.3	maxHeight	152
9.21.3.4	maxPacketSize	152
9.21.3.5	maxWidth	153
9.21.3.6	minPacketSize	153
9.21.3.7	mode	153
9.21.3.8	offsetHStepSize	153
9.21.3.9	offsetVStepSize	153
9.21.3.10	packetSize	153
9.21.3.11	percentage	153
9.21.3.12	pixelFormatBitField	153
9.21.3.13	reserved	153
9.21.3.14	vendorPixelFormatBitField	153
9.22	Format7PacketInfo Struct Reference	154
9.22.1	Detailed Description	154
9.22.2	Constructor & Destructor Documentation	154
9.22.2.1	Format7PacketInfo	154
9.22.3	Member Data Documentation	154
9.22.3.1	maxBytesPerPacket	154
9.22.3.2	recommendedBytesPerPacket	154
9.22.3.3	reserved	154
9.22.3.4	unitBytesPerPacket	155
9.23	GCCamera Class Reference	155
9.23.1	Constructor & Destructor Documentation	159
9.23.1.1	GCCamera	159
9.23.1.2	~GCCamera	159
9.23.2	Member Function Documentation	159
9.23.2.1	Connect	159
9.23.2.2	Connect	159
9.23.2.3	Disconnect	159

9.23.2.4	EnableLUT	160
9.23.2.5	FireSoftwareTrigger	160
9.23.2.6	GCCamera::GetXML	160
9.23.2.7	GetActiveLUTBank	160
9.23.2.8	GetCameraInfo	161
9.23.2.9	GetConfiguration	161
9.23.2.10	GetCycleTime	161
9.23.2.11	GetEmbeddedImageInfo	162
9.23.2.12	GetGPIOPinDirection	162
9.23.2.13	GetInterfaceType	163
9.23.2.14	GetLUTBankInfo	163
9.23.2.15	GetLUTChannel	163
9.23.2.16	GetLUTInfo	164
9.23.2.17	GetMemoryChannel	164
9.23.2.18	GetMemoryChannelInfo	165
9.23.2.19	GetNodeMap	165
9.23.2.20	GetProperty	165
9.23.2.21	GetPropertyInfo	166
9.23.2.22	GetRegisterString	166
9.23.2.23	GetStats	166
9.23.2.24	GetStrobe	166
9.23.2.25	GetStrobeInfo	167
9.23.2.26	GetTriggerDelay	167
9.23.2.27	GetTriggerDelayInfo	168
9.23.2.28	GetTriggerMode	168
9.23.2.29	GetTriggerModelInfo	169
9.23.2.30	IsConnected	169
9.23.2.31	ReadGVCPMemory	170
9.23.2.32	ReadGVCPRegister	170
9.23.2.33	ReadGVCPRegisterBlock	170
9.23.2.34	ReadRegister	170
9.23.2.35	ReadRegisterBlock	170
9.23.2.36	ResetStats	171
9.23.2.37	RestoreFromMemoryChannel	171

9.23.2.38 RetrieveBuffer	171
9.23.2.39 SaveToMemoryChannel	172
9.23.2.40 SetActiveLUTBank	172
9.23.2.41 SetCallback	173
9.23.2.42 SetCamera	173
9.23.2.43 SetCamera	173
9.23.2.44 SetConfiguration	173
9.23.2.45 SetEmbeddedImageInfo	173
9.23.2.46 SetGPIOPinDirection	174
9.23.2.47 SetLUTChannel	174
9.23.2.48 SetProperty	175
9.23.2.49 SetStrobe	175
9.23.2.50 SetTriggerDelay	176
9.23.2.51 SetTriggerMode	176
9.23.2.52 SetUserBuffers	177
9.23.2.53 StartCapture	177
9.23.2.54 StartSyncCapture	178
9.23.2.55 StopCapture	178
9.23.2.56 TestGainNode	179
9.23.2.57 WaitForBufferEvent	179
9.23.2.58 WriteGVCPMemory	179
9.23.2.59 WriteGVCPRegister	179
9.23.2.60 WriteGVCPRegisterBlock	179
9.23.2.61 WriteRegister	179
9.23.2.62 WriteRegisterBlock	180
9.23.3 Member Data Documentation	180
9.23.3.1 m_busMgr	180
9.24 GigECamera Class Reference	180
9.24.1 Detailed Description	186
9.24.2 Constructor & Destructor Documentation	186
9.24.2.1 GigECamera	186
9.24.2.2 ~GigECamera	186
9.24.3 Member Function Documentation	186
9.24.3.1 Connect	186

9.24.3.2	DeregisterAllEvents	187
9.24.3.3	DeregisterEvent	187
9.24.3.4	Disconnect	187
9.24.3.5	DiscoverGigEPacketSize	187
9.24.3.6	EnableLUT	187
9.24.3.7	FireSoftwareTrigger	188
9.24.3.8	GetActiveLUTBank	188
9.24.3.9	GetCameraInfo	189
9.24.3.10	GetConfiguration	189
9.24.3.11	GetCycleTime	189
9.24.3.12	GetEmbeddedImageInfo	190
9.24.3.13	GetGigEConfig	190
9.24.3.14	GetGigEImageBinningSettings	190
9.24.3.15	GetGigEImageSettings	191
9.24.3.16	GetGigEImageSettingsInfo	191
9.24.3.17	GetGigEImagingMode	191
9.24.3.18	GetGigEProperty	192
9.24.3.19	GetGigEStreamChannelInfo	192
9.24.3.20	GetGPIOPinDirection	192
9.24.3.21	GetLUTBankInfo	193
9.24.3.22	GetLUTChannel	193
9.24.3.23	GetLUTInfo	194
9.24.3.24	GetMemoryChannel	194
9.24.3.25	GetMemoryChannelInfo	195
9.24.3.26	GetNumStreamChannels	195
9.24.3.27	GetProperty	195
9.24.3.28	GetPropertyInfo	196
9.24.3.29	GetRegisterString	196
9.24.3.30	GetStats	197
9.24.3.31	GetStrobe	197
9.24.3.32	GetStrobeInfo	197
9.24.3.33	GetTriggerDelay	198
9.24.3.34	GetTriggerDelayInfo	198
9.24.3.35	GetTriggerMode	199

9.24.3.36 GetTriggerModelInfo	199
9.24.3.37 IsConnected	200
9.24.3.38 QueryGigEImagingMode	200
9.24.3.39 ReadGVCPMemory	200
9.24.3.40 ReadGVCPRegister	201
9.24.3.41 ReadGVCPRegisterBlock	201
9.24.3.42 ReadRegister	201
9.24.3.43 ReadRegisterBlock	202
9.24.3.44 RegisterAllEvents	202
9.24.3.45 RegisterEvent	202
9.24.3.46 ResetStats	202
9.24.3.47 RestoreFromMemoryChannel	202
9.24.3.48 RetrieveBuffer	203
9.24.3.49 SaveToMemoryChannel	203
9.24.3.50 SetActiveLUTBank	204
9.24.3.51 SetCallback	204
9.24.3.52 SetConfiguration	205
9.24.3.53 SetEmbeddedImageInfo	205
9.24.3.54 SetGigEConfig	205
9.24.3.55 SetGigEImageBinningSettings	206
9.24.3.56 SetGigEImageSettings	206
9.24.3.57 SetGigEImagingMode	206
9.24.3.58 SetGigEProperty	207
9.24.3.59 SetGigEStreamChannelInfo	207
9.24.3.60 SetGPIOPinDirection	207
9.24.3.61 SetLUTChannel	208
9.24.3.62 SetProperty	208
9.24.3.63 SetStrobe	209
9.24.3.64 SetTriggerDelay	209
9.24.3.65 SetTriggerMode	210
9.24.3.66 SetUserBuffers	210
9.24.3.67 StartCapture	211
9.24.3.68 StartSyncCapture	212
9.24.3.69 StopCapture	212

9.24.3.70	WaitForBufferEvent	212
9.24.3.71	WriteGVCPMemory	213
9.24.3.72	WriteGVCPRegister	213
9.24.3.73	WriteGVCPRegisterBlock	213
9.24.3.74	WriteRegister	214
9.24.3.75	WriteRegisterBlock	214
9.25	GigEConfig Struct Reference	215
9.25.1	Detailed Description	215
9.25.2	Constructor & Destructor Documentation	215
9.25.2.1	GigEConfig	215
9.25.3	Member Data Documentation	215
9.25.3.1	enablePacketResend	215
9.25.3.2	registerTimeout	215
9.25.3.3	registerTimeoutRetries	216
9.26	GigEImageSettings Struct Reference	216
9.26.1	Detailed Description	216
9.26.2	Constructor & Destructor Documentation	216
9.26.2.1	GigEImageSettings	216
9.26.3	Member Data Documentation	216
9.26.3.1	height	217
9.26.3.2	offsetX	217
9.26.3.3	offsetY	217
9.26.3.4	pixelFormat	217
9.26.3.5	reserved	217
9.26.3.6	width	217
9.27	GigEImageSettingsInfo Struct Reference	217
9.27.1	Detailed Description	218
9.27.2	Constructor & Destructor Documentation	218
9.27.2.1	GigEImageSettingsInfo	218
9.27.3	Member Data Documentation	218
9.27.3.1	imageHStepSize	218
9.27.3.2	imageVStepSize	218
9.27.3.3	maxHeight	218
9.27.3.4	maxWidth	218

9.27.3.5	offsetHStepSize	219
9.27.3.6	offsetVStepSize	219
9.27.3.7	pixelFormatBitField	219
9.27.3.8	reserved	219
9.27.3.9	vendorPixelFormatBitField	219
9.28	GigEProperty Struct Reference	219
9.28.1	Detailed Description	220
9.28.2	Member Data Documentation	220
9.28.2.1	isReadable	220
9.28.2.2	isWritable	220
9.28.2.3	max	220
9.28.2.4	min	220
9.28.2.5	propType	220
9.28.2.6	value	220
9.29	GigEStreamChannel Struct Reference	220
9.29.1	Detailed Description	221
9.29.2	Constructor & Destructor Documentation	222
9.29.2.1	GigEStreamChannel	222
9.29.3	Member Data Documentation	222
9.29.3.1	destinationIpAddress	222
9.29.3.2	doNotFragment	222
9.29.3.3	hostPort	222
9.29.3.4	interPacketDelay	222
9.29.3.5	networkInterfaceIndex	222
9.29.3.6	packetSize	222
9.29.3.7	sourcePort	222
9.30	H264Option Struct Reference	222
9.30.1	Detailed Description	223
9.30.2	Constructor & Destructor Documentation	223
9.30.2.1	H264Option	223
9.30.3	Member Data Documentation	223
9.30.3.1	bitrate	223
9.30.3.2	frameRate	223
9.30.3.3	height	223

9.30.3.4	reserved	223
9.30.3.5	width	224
9.31	Image Class Reference	224
9.31.1	Detailed Description	227
9.31.2	Constructor & Destructor Documentation	227
9.31.2.1	Image	227
9.31.2.2	Image	227
9.31.2.3	Image	227
9.31.2.4	Image	228
9.31.2.5	Image	228
9.31.2.6	Image	228
9.31.2.7	~Image	228
9.31.3	Member Function Documentation	228
9.31.3.1	CalculateStatistics	229
9.31.3.2	Convert	229
9.31.3.3	Convert	229
9.31.3.4	DeepCopy	230
9.31.3.5	DetermineBitsPerPixel	230
9.31.3.6	GetBayerTileFormat	230
9.31.3.7	GetBitsPerPixel	230
9.31.3.8	GetBlockId	230
9.31.3.9	GetColorProcessing	231
9.31.3.10	GetCols	231
9.31.3.11	GetData	231
9.31.3.12	GetData	231
9.31.3.13	GetDataSize	231
9.31.3.14	GetDefaultColorProcessing	232
9.31.3.15	GetDefaultOutputFormat	232
9.31.3.16	GetDimensions	232
9.31.3.17	GetMetadata	232
9.31.3.18	GetPixelFormat	233
9.31.3.19	GetReceivedDataSize	233
9.31.3.20	GetRows	233
9.31.3.21	GetStride	233

9.31.3.22	GetTimeStamp	233
9.31.3.23	operator()	234
9.31.3.24	operator=	234
9.31.3.25	operator[]	234
9.31.3.26	ReleaseBuffer	234
9.31.3.27	Save	235
9.31.3.28	Save	235
9.31.3.29	Save	235
9.31.3.30	Save	235
9.31.3.31	Save	236
9.31.3.32	Save	236
9.31.3.33	Save	236
9.31.3.34	Save	237
9.31.3.35	SetBlockId	237
9.31.3.36	SetColorProcessing	237
9.31.3.37	SetData	238
9.31.3.38	SetDefaultColorProcessing	238
9.31.3.39	SetDefaultOutputFormat	238
9.31.3.40	SetDimensions	239
9.31.4	Friends And Related Function Documentation	239
9.31.4.1	Iso	239
9.32	ImageMetadata Struct Reference	239
9.32.1	Detailed Description	240
9.32.2	Constructor & Destructor Documentation	240
9.32.2.1	ImageMetadata	240
9.32.3	Member Data Documentation	240
9.32.3.1	embeddedBrightness	240
9.32.3.2	embeddedExposure	240
9.32.3.3	embeddedFrameCounter	241
9.32.3.4	embeddedGain	241
9.32.3.5	embeddedGPIOPinState	241
9.32.3.6	embeddedROIPosition	241
9.32.3.7	embeddedShutter	241
9.32.3.8	embeddedStrobePattern	241

9.32.3.9	embeddedTimeStamp	241
9.32.3.10	embeddedWhiteBalance	241
9.32.3.11	reserved	241
9.33	ImageStatistics Class Reference	241
9.33.1	Detailed Description	243
9.33.2	Member Enumeration Documentation	243
9.33.2.1	StatisticsChannel	243
9.33.3	Constructor & Destructor Documentation	243
9.33.3.1	ImageStatistics	243
9.33.3.2	~ImageStatistics	243
9.33.3.3	ImageStatistics	243
9.33.4	Member Function Documentation	244
9.33.4.1	DisableAll	244
9.33.4.2	EnableAll	244
9.33.4.3	EnableGreyOnly	244
9.33.4.4	EnableHSLOnly	244
9.33.4.5	EnableRGBOnly	244
9.33.4.6	GetChannelStatus	245
9.33.4.7	GetHistogram	245
9.33.4.8	GetMean	245
9.33.4.9	GetNumPixelValues	246
9.33.4.10	GetPixelValueRange	246
9.33.4.11	GetRange	246
9.33.4.12	GetStatistics	247
9.33.4.13	operator=	247
9.33.4.14	SetChannelStatus	247
9.33.5	Friends And Related Function Documentation	248
9.33.5.1	ImageStatsCalculator	248
9.34	Internal Class Reference	248
9.34.1	Member Function Documentation	248
9.34.1.1	GetInternal	248
9.35	IPAddress Struct Reference	248
9.35.1	Detailed Description	249
9.35.2	Constructor & Destructor Documentation	249

9.35.2.1	IPAddress	249
9.35.2.2	IPAddress	249
9.35.3	Member Function Documentation	249
9.35.3.1	operator!=	249
9.35.3.2	operator==	249
9.35.4	Member Data Documentation	249
9.35.4.1	octets	249
9.36	JPEGOption Struct Reference	249
9.36.1	Detailed Description	250
9.36.2	Constructor & Destructor Documentation	250
9.36.2.1	JPEGOption	250
9.36.3	Member Data Documentation	250
9.36.3.1	progressive	250
9.36.3.2	quality	250
9.36.3.3	reserved	250
9.37	JPG2Option Struct Reference	251
9.37.1	Detailed Description	251
9.37.2	Constructor & Destructor Documentation	251
9.37.2.1	JPG2Option	251
9.37.3	Member Data Documentation	251
9.37.3.1	quality	251
9.37.3.2	reserved	251
9.38	LUTData Struct Reference	251
9.38.1	Detailed Description	252
9.38.2	Constructor & Destructor Documentation	252
9.38.2.1	LUTData	252
9.38.3	Member Data Documentation	252
9.38.3.1	enabled	252
9.38.3.2	inputBitDepth	252
9.38.3.3	numBanks	253
9.38.3.4	numChannels	253
9.38.3.5	numEntries	253
9.38.3.6	outputBitDepth	253
9.38.3.7	reserved	253

9.38.3.8	supported	253
9.39	MACAddress Struct Reference	253
9.39.1	Detailed Description	254
9.39.2	Constructor & Destructor Documentation	254
9.39.2.1	MACAddress	254
9.39.2.2	MACAddress	254
9.39.3	Member Function Documentation	254
9.39.3.1	operator!=	254
9.39.3.2	operator==	254
9.39.4	Member Data Documentation	254
9.39.4.1	octets	254
9.40	MJPGOption Struct Reference	254
9.40.1	Detailed Description	255
9.40.2	Constructor & Destructor Documentation	255
9.40.2.1	MJPGOption	255
9.40.3	Member Data Documentation	255
9.40.3.1	frameRate	255
9.40.3.2	quality	255
9.40.3.3	reserved	255
9.41	NodeMap Class Reference	255
9.41.1	Constructor & Destructor Documentation	256
9.41.1.1	NodeMap	256
9.41.1.2	~NodeMap	256
9.41.2	Member Function Documentation	256
9.41.2.1	_GetDeviceName	256
9.41.2.2	_GetNode	256
9.41.2.3	_GetNodes	256
9.41.2.4	_InvalidateNodes	256
9.41.2.5	_Poll	256
9.42	PGMOption Struct Reference	256
9.42.1	Detailed Description	257
9.42.2	Constructor & Destructor Documentation	257
9.42.2.1	PGMOption	257
9.42.3	Member Data Documentation	257

9.42.3.1	binaryFile	257
9.42.3.2	reserved	257
9.43	PGRGuid Class Reference	257
9.43.1	Detailed Description	258
9.43.2	Constructor & Destructor Documentation	258
9.43.2.1	PGRGuid	258
9.43.3	Member Function Documentation	258
9.43.3.1	operator!=	258
9.43.3.2	operator==	258
9.43.4	Member Data Documentation	258
9.43.4.1	value	258
9.44	PNGOption Struct Reference	258
9.44.1	Detailed Description	259
9.44.2	Constructor & Destructor Documentation	259
9.44.2.1	PNGOption	259
9.44.3	Member Data Documentation	259
9.44.3.1	compressionLevel	259
9.44.3.2	interlaced	259
9.44.3.3	reserved	259
9.45	PPMOption Struct Reference	259
9.45.1	Detailed Description	260
9.45.2	Constructor & Destructor Documentation	260
9.45.2.1	PPMOption	260
9.45.3	Member Data Documentation	260
9.45.3.1	binaryFile	260
9.45.3.2	reserved	260
9.46	Property Struct Reference	260
9.46.1	Detailed Description	261
9.46.2	Constructor & Destructor Documentation	262
9.46.2.1	Property	262
9.46.2.2	Property	262
9.46.3	Member Data Documentation	262
9.46.3.1	absControl	262
9.46.3.2	absValue	262

9.46.3.3	autoManualMode	262
9.46.3.4	onePush	262
9.46.3.5	onOff	262
9.46.3.6	present	262
9.46.3.7	reserved	262
9.46.3.8	type	262
9.46.3.9	valueA	263
9.46.3.10	valueB	263
9.47	PropertyInfo Struct Reference	263
9.47.1	Detailed Description	264
9.47.2	Constructor & Destructor Documentation	264
9.47.2.1	PropertyInfo	264
9.47.2.2	PropertyInfo	264
9.47.3	Member Data Documentation	264
9.47.3.1	absMax	264
9.47.3.2	absMin	264
9.47.3.3	absValSupported	264
9.47.3.4	autoSupported	264
9.47.3.5	manualSupported	265
9.47.3.6	max	265
9.47.3.7	min	265
9.47.3.8	onePushSupported	265
9.47.3.9	onOffSupported	265
9.47.3.10	present	265
9.47.3.11	pUnitAbbr	265
9.47.3.12	pUnits	265
9.47.3.13	readOutSupported	265
9.47.3.14	reserved	265
9.47.3.15	type	266
9.48	StrobeControl Struct Reference	266
9.48.1	Detailed Description	266
9.48.2	Constructor & Destructor Documentation	266
9.48.2.1	StrobeControl	266
9.48.3	Member Data Documentation	266

9.48.3.1	delay	266
9.48.3.2	duration	267
9.48.3.3	onOff	267
9.48.3.4	polarity	267
9.48.3.5	reserved	267
9.48.3.6	source	267
9.49	StrobeInfo Struct Reference	267
9.49.1	Detailed Description	268
9.49.2	Constructor & Destructor Documentation	268
9.49.2.1	StrobeInfo	268
9.49.3	Member Data Documentation	268
9.49.3.1	maxValue	268
9.49.3.2	minValue	268
9.49.3.3	onOffSupported	268
9.49.3.4	polaritySupported	268
9.49.3.5	present	268
9.49.3.6	readOutSupported	268
9.49.3.7	reserved	269
9.49.3.8	source	269
9.50	SyncManager Class Reference	269
9.50.1	Constructor & Destructor Documentation	269
9.50.1.1	SyncManager	269
9.50.1.2	~SyncManager	269
9.50.2	Member Function Documentation	269
9.50.2.1	DisableCrossPCsynchronization	269
9.50.2.2	EnableCrossPCsynchronization	269
9.50.2.3	GetSyncStatus	269
9.50.2.4	GetTimeSinceSynced	269
9.50.2.5	IsTimingBusConnected	269
9.50.2.6	QueryCrossPCsynchronizationSetting	270
9.50.2.7	RescanMasterTimingBus	270
9.50.2.8	Start	270
9.50.2.9	Stop	270
9.51	SystemInfo Struct Reference	270

9.51.1	Detailed Description	271
9.51.2	Member Data Documentation	271
9.51.2.1	byteOrder	271
9.51.2.2	cpuDescription	271
9.51.2.3	driverList	271
9.51.2.4	gpuDescription	271
9.51.2.5	libraryList	271
9.51.2.6	numCpuCores	271
9.51.2.7	osDescription	271
9.51.2.8	osType	271
9.51.2.9	reserved	271
9.51.2.10	screenHeight	272
9.51.2.11	screenWidth	272
9.51.2.12	sysMemSize	272
9.52	TIFFOption Struct Reference	272
9.52.1	Detailed Description	272
9.52.2	Member Enumeration Documentation	273
9.52.2.1	CompressionMethod	273
9.52.3	Constructor & Destructor Documentation	273
9.52.3.1	TIFFOption	273
9.52.4	Member Data Documentation	273
9.52.4.1	compression	273
9.52.4.2	reserved	273
9.53	TimeStamp Struct Reference	273
9.53.1	Detailed Description	274
9.53.2	Constructor & Destructor Documentation	274
9.53.2.1	TimeStamp	274
9.53.3	Member Data Documentation	274
9.53.3.1	cycleCount	274
9.53.3.2	cycleOffset	274
9.53.3.3	cycleSeconds	274
9.53.3.4	microSeconds	274
9.53.3.5	reserved	275
9.53.3.6	seconds	275

9.54	TopologyNode Class Reference	275
9.54.1	Detailed Description	276
9.54.2	Member Enumeration Documentation	276
9.54.2.1	NodeType	276
9.54.2.2	PortType	277
9.54.3	Constructor & Destructor Documentation	277
9.54.3.1	TopologyNode	277
9.54.3.2	TopologyNode	277
9.54.3.3	~TopologyNode	277
9.54.3.4	TopologyNode	277
9.54.4	Member Function Documentation	277
9.54.4.1	AddChild	277
9.54.4.2	AddPortType	278
9.54.4.3	AssignGuidToNode	278
9.54.4.4	AssignGuidToNode	278
9.54.4.5	GetChild	278
9.54.4.6	GetDeviceId	279
9.54.4.7	GetGuid	279
9.54.4.8	GetInterfaceType	279
9.54.4.9	GetNodeType	279
9.54.4.10	GetNumChildren	279
9.54.4.11	GetNumPorts	280
9.54.4.12	GetPortType	280
9.54.4.13	operator=	280
9.55	TriggerMode Struct Reference	280
9.55.1	Detailed Description	281
9.55.2	Constructor & Destructor Documentation	281
9.55.2.1	TriggerMode	281
9.55.3	Member Data Documentation	281
9.55.3.1	mode	281
9.55.3.2	onOff	281
9.55.3.3	parameter	281
9.55.3.4	polarity	281
9.55.3.5	reserved	281

9.55.3.6	source	282
9.56	TriggerModelInfo Struct Reference	282
9.56.1	Detailed Description	282
9.56.2	Constructor & Destructor Documentation	283
9.56.2.1	TriggerModelInfo	283
9.56.3	Member Data Documentation	283
9.56.3.1	modeMask	283
9.56.3.2	onOffSupported	283
9.56.3.3	polaritySupported	283
9.56.3.4	present	283
9.56.3.5	readOutSupported	283
9.56.3.6	reserved	283
9.56.3.7	softwareTriggerSupported	283
9.56.3.8	sourceMask	283
9.56.3.9	valueReadable	283
9.57	Utilities Class Reference	284
9.57.1	Detailed Description	284
9.57.2	Member Function Documentation	284
9.57.2.1	CheckDriver	284
9.57.2.2	GetDriverDeviceName	285
9.57.2.3	GetLibraryVersion	285
9.57.2.4	GetSystemInfo	285
9.57.2.5	LaunchBrowser	285
9.57.2.6	LaunchCommand	286
9.57.2.7	LaunchCommandAsync	286
9.57.2.8	LaunchHelp	287
10	File Documentation	289
10.1	BusManager.h File Reference	289
10.2	Camera.h File Reference	289
10.3	CameraBase.h File Reference	290
10.4	Error.h File Reference	290
10.5	FlyCapture2.h File Reference	290
10.6	FlyCapture2Defs.h File Reference	290

10.6.1 Define Documentation	295
10.6.1.1 FULL_32BIT_VALUE	295
10.6.1.2 NULL	295
10.7 FlyCapture2GUI.h File Reference	295
10.8 FlyCapture2Platform.h File Reference	296
10.8.1 Define Documentation	296
10.8.1.1 FLYCAPTURE2_API	296
10.8.1.2 FLYCAPTURE2_LOCAL	296
10.9 FlyCapture2Video.h File Reference	296
10.10 FlyCapture2VideoDefs.h File Reference	296
10.11 FlyCapture3ApiGuiWrapper.h File Reference	297
10.11.1 Define Documentation	297
10.11.1.1 WRAPPER_API	297
10.12 GCCamera.h File Reference	297
10.13 GigECamera.h File Reference	298
10.14 Image.h File Reference	298
10.15 ImageStatistics.h File Reference	298
10.16 Internal.h File Reference	298
10.17 Licensing.dox File Reference	299
10.18 MultiSyncLibrary.h File Reference	299
10.19 MultiSyncLibraryDefs.h File Reference	299
10.20 MultiSyncLibraryPlatform.h File Reference	299
10.20.1 Define Documentation	299
10.20.1.1 MULTISYNCLIBRARY_API	299
10.20.1.2 MULTISYNCLIBRARY_LOCAL	300
10.21 NodeMap.h File Reference	300
10.22 TopologyNode.h File Reference	300
10.23 Utilities.h File Reference	300

Chapter 1

Software Licensing Information

Component	License
FlyCapture2	Copyright © 2017 FLIR Integrated Imaging Solutions, Inc. All Rights Reserved. This software is the confidential and proprietary information of FLIR Integrated Imaging Solutions, Inc. ("Confidential Information"). You shall not disclose such Confidential Information and shall use it only in accordance with the terms of the license agreement you entered into with FLIR Integrated Imaging Solutions, Inc. (FLIR). FLIR MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THE SOFTWARE, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR 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) http://www.codeproject.com/info/cpol10.aspx
Boost	Boost Software License http://www.boost.org/users/license.html
FFMPEG	LGPLv2.1 License https://www.ffmpeg.org/legal.html
FreeImage	FreeImage public license http://freeimage.sourceforge.net/freeimage-license.txt
GTK	LGPLv2.1 License http://www.gnu.org/licenses/old-licenses/lgpl-2.1.txt
Libusb	LGPLv2.1 License http://www.gnu.org/licenses/old-licenses/lgpl-2.1.txt
Libraw1394	LGPLv2.0 License http://www.gnu.org/licenses/old-licenses/lgpl-2.0.txt
Generated on Wed Apr 3 2019 19:06:42 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
Video saving structures.	37

Chapter 3

Namespace Index

3.1 Namespace List

Here is a list of all namespaces with brief descriptions:

FlyCap3CameraControl	39
FlyCapture2	39
MultiSyncLibrary	46

Chapter 4

Class Index

4.1 Class Hierarchy

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

AVIOption	49
BMPOption	50
BusManager	51
CameraBase	92
Camera	62
GCCamera	155
GigECamera	180
CameraControlDlg	119
CameraInfo	121
CameraSelectionDlg	126
CameraStats	128
ConfigROM	130
EmbeddedImageInfo	133
EmbeddedImageInfoProperty	134
Error	135
EventCallbackData	139
EventOptions	140
FC2Config	141
FC2Version	144
FlyCapture2Video	145
FlyCapture3ApiGuiWrapper	148
Format7ImageSettings	149
Format7Info	151
Format7PacketInfo	154
GigEConfig	215
GigEImageSettings	216
GigEImageSettingsInfo	217
GigEProperty	219
GigEStreamChannel	220

H264Option	222
Image	224
ImageMetadata	239
ImageStatistics	241
Internal	248
IPAddress	248
JPEGOption	249
JPG2Option	251
LUTData	251
MACAddress	253
MJPGOption	254
NodeMap	255
PGMOption	256
PGRGuid	257
PNGOption	258
PPMOption	259
Property	260
PropertyInfo	263
StrobeControl	266
StrobeInfo	267
SyncManager	269
SystemInfo	270
TIFFOption	272
TimeStamp	273
TopologyNode	275
TriggerMode	280
TriggerModelInfo	282
Utilities	284

Chapter 5

Class Index

5.1 Class List

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

AVIOption	Options for saving AVI files	49
BMPOption	Options for saving Bitmap image	50
BusManager	Functionality for the user to get an PGRGuid for a desired camera or device easily	51
Camera	The Camera object represents a physical camera that uses the IIDC register set	62
CameraBase	Abstract base class that defines a general interface to a camera . . .	92
CameraControlDlg	The CameraControlDlg object represents a dialog that provides a graphical interface to a specified camera	119
CameraInfo	Camera information	121
CameraSelectionDlg	The CameraSelectionDlg object represents a dialog that provides a graphical interface that lists the number of cameras available to the library	126
CameraStats	Camera diagnostic information	128
ConfigROM	Camera configuration ROM	130
EmbeddedImageInfo	Properties of the possible embedded image information	133
EmbeddedImageInfoProperty	Properties of a single embedded image info property	134

Error	
The Error object represents an error that is returned from the library	135
EventCallbackData	139
EventOptions	
Options for enabling device event registration	140
FC2Config	
Configuration for a camera	141
FC2Version	
The current version of the library	144
FlyCapture2Video	
Functionality for the user to record images to an AVI file	145
FlyCapture3ApiGuiWrapper	148
Format7ImageSettings	
Format 7 image settings	149
Format7Info	
Format 7 information for a single mode	151
Format7PacketInfo	
Format 7 packet information	154
GCCamera	155
GigECamera	
The GigECamera object represents a physical Gigabit Ethernet camera	180
GigEConfig	
Configuration for a GigE camera	215
GigEImageSettings	
Image settings for a GigE camera	216
GigEImageSettingsInfo	
Format 7 information for a single mode	217
GigEProperty	
A GigE property	219
GigEStreamChannel	
Information about a single GigE stream channel	220
H264Option	
Options for saving H264 files	222
Image	
Used to retrieve images from a camera, convert between multiple pixel formats and save images to disk	224
ImageMetadata	
Metadata related to an image	239
ImageStatistics	
The ImageStatistics object represents image statistics for an image	241
Internal	248
IPAddress	
IPv4 address	248
JPEGOption	
Options for saving JPEG image	249
JPG2Option	
Options for saving JPEG2000 image	251
LUTData	
Information about the camera's look up table	251

MACAddress	
MAC address	253
MJPGOption	
Options for saving MJPG files	254
NodeMap	255
PGMOption	
Options for saving PGM images	256
PGRGuid	
A GUID to the camera	257
PNGOption	
Options for saving PNG images	258
PPMOption	
Options for saving PPM images	259
Property	
A specific camera property	260
PropertyInfo	
Information about a specific camera property	263
StrobeControl	
A camera strobe	266
StrobeInfo	
A camera strobe property	267
SyncManager	269
SystemInfo	
Description of the system	270
TIFFOption	
Options for saving TIFF images	272
TimeStamp	
Timestamp information	273
TopologyNode	
Topology information that can be used to generate a tree structure of all cameras and devices connected to a computer	275
TriggerMode	
A camera trigger	280
TriggerModelInfo	
Information about a camera trigger property	282
Utilities	
The Utility class is generally used to query for general system infor- mation such as operating system, available memory etc	284

Chapter 6

File Index

6.1 File List

Here is a list of all files with brief descriptions:

BusManager.h	289
Camera.h	289
CameraBase.h	290
Error.h	290
FlyCapture2.h	290
FlyCapture2Defs.h	290
FlyCapture2GUI.h	295
FlyCapture2Platform.h	296
FlyCapture2Video.h	296
FlyCapture2VideoDefs.h	296
FlyCapture3ApiGuiWrapper.h	297
GCCamera.h	297
GigECamera.h	298
Image.h	298
ImageStatistics.h	298
Internal.h	298
MultiSyncLibrary.h	299
MultiSyncLibraryDefs.h	299
MultiSyncLibraryPlatform.h	299
NodeMap.h	300
TopologyNode.h	300
Utilities.h	300

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_JUJU firewire_core.

DRIVER_1394_VIDEO1394 video1394.

DRIVER_1394_RAW1394 raw1394.

DRIVER_USB_NONE No usb driver used just BSD stack. (Linux only)

DRIVER_USB_CAM 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`

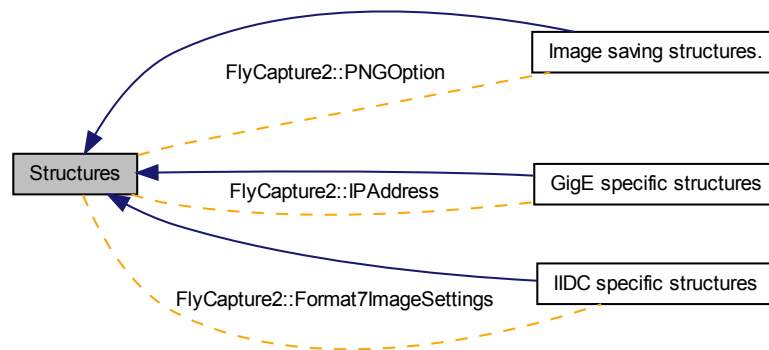
Possible properties that can be queried from the camera.

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 [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)`

7.8 Video saving structures.

These structures define various parameters used for saving videos.

Classes

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

7.8.1 Detailed Description

These structures define various parameters used for saving videos.

Chapter 8

Namespace Documentation

8.1 FlyCap3CameraControl Namespace Reference

Classes

- class [FlyCapture3ApiGuiWrapper](#)

8.2 FlyCapture2 Namespace Reference

Classes

- 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 [TriggerModelInfo](#)
Information about a camera trigger property.
- struct [TriggerMode](#)
A camera trigger.
- struct [StrobeInfo](#)
A camera strobe property.
- struct [StrobeControl](#)
A camera strobe.
- struct [TimeStamp](#)
Timestamp information.
- struct [ConfigROM](#)
Camera configuration ROM.
- struct [CameraInfo](#)
Camera information.
- struct [EmbeddedImageInfoProperty](#)
Properties of a single embedded image info property.
- struct [EmbeddedImageInfo](#)
Properties of the possible embedded image information.
- struct [ImageMetadata](#)
Metadata related to an image.
- struct [LUTData](#)

Information about the camera's look up table.

- struct [CameraStats](#)

Camera diagnostic information.

- struct [PNGOption](#)

Options for saving PNG images.

- struct [PPMOption](#)

Options for saving PPM images.

- struct [PGMOption](#)

Options for saving PGM images.

- struct [TIFFOption](#)

Options for saving TIFF images.

- struct [JPEGOption](#)

Options for saving JPEG image.

- struct [JPG2Option](#)

Options for saving JPEG2000 image.

- struct [BMPOption](#)

Options for saving Bitmap image.

- struct [EventOptions](#)

Options for enabling device event registration.

- struct [EventCallbackData](#)

- 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 [FlyCapture2Video](#)

The [FlyCapture2Video](#) class provides the functionality for the user to record images to an AVI file.

- struct [MJPGOption](#)

Options for saving MJPG files.

- struct [H264Option](#)

Options for saving H264 files.

- struct [AVIOption](#)

Options for saving AVI files.

- 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](#),

PGRROR_PROPERTY_FAILED, PGRROR_PROPERTY_NOT_PRESENT, PGRROR_REGISTER_FAILED, PGRROR_READ_REGISTER_FAILED, PGRROR_WRITE_REGISTER_FAILED, PGRROR_ISOCH_FAILED, PGRROR_ISOCH_ALREADY_STARTED, PGRROR_ISOCH_NOT_STARTED, PGRROR_ISOCH_START_FAILED, PGRROR_ISOCH_RETRIEVE_BUFFER_FAILED, PGRROR_ISOCH_STOP_FAILED, PGRROR_ISOCH_SYNC_FAILED, PGRROR_ISOCH_BANDWIDTH_EXCEEDED, PGRROR_IMAGE_CONVERSION_FAILED, PGRROR_IMAGE_LIBRARY_FAILURE, PGRROR_BUFFER_TOO_SMALL, PGRROR_IMAGE_CONSISTENCY_ERROR, PGRROR_INCOMPATIBLE_DRIVER, PGRROR_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`,

```

    VIDEO_MODE_1024x768Y16, VIDEO_MODE_1280x960YUV422, VIDEO_MODE_1280x960RGB, VIDEO_MODE_1280x960Y8, VIDEO_MODE_1280x960Y16, VIDEO_MODE_1600x1200YUV422, VIDEO_MODE_1600x1200RGB, VIDEO_MODE_1600x1200Y8, VIDEO_MODE_1600x1200Y16, VIDEO_MODE_FORMAT7, NUM_VIDEO_MODES, VIDEO_MODE_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:

- [FlyCapture2VideoDefs.h](#)

9.2 BMPOption Struct Reference

Options for saving Bitmap image.

Public Member Functions

- [BMPOption](#) ()

Public Attributes

- bool [indexedColor_8bit](#)
- unsigned int [reserved](#) [16]

Reserved for future use.

9.2.1 Detailed Description

Options for saving Bitmap image.

9.2.2 Constructor & Destructor Documentation

9.2.2.1 [BMPOption](#) () `[inline]`

9.2.3 Member Data Documentation

9.2.3.1 bool [indexedColor_8bit](#)

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

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

9.3 BusManager Class Reference

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

Public Member Functions

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

Gets the topology information for the PC.

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

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

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

Unregister a callback function.

- virtual [Error RescanBus](#) ()

Force a rescan of the buses.

- [Error IsCameraControlable](#) ([PGRGuid](#) *pGuid, bool *pControlable)

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

Static Public Member Functions

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

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

- static [Error ForceAllIPAddressesAutomatically](#) ()

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

- static [Error ForceAllIPAddressesAutomatically](#) (unsigned int serialNumber)

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

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

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

9.3.1 Detailed Description

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

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

9.3.2 Constructor & Destructor Documentation

9.3.2.1 [BusManager](#) ()

Default constructor.

9.3.2.2 virtual ~BusManager () [virtual]

Default destructor.

9.3.3 Member Function Documentation

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

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

This is useful in situations where 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 CameraInfo structures.
<i>arraySize</i>	Size of the array. Number of discovered cameras is returned in the same value.

Returns

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

9.3.3.2 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>	PGRGuid of the camera or the device to cause bus reset.
--------------	---

Returns

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

9.3.3.3 static Error ForceAllIPAddressesAutomatically () [static]

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

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

Returns

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

9.3.3.4 static Error ForceAllIPAddressesAutomatically (unsigned int *serialNumber*) [static]

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

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

Returns

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

9.3.3.5 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.

9.3.3.6 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 PGRGuid for the camera.

See also

[GetCameraFromSerialNumber\(\)](#)

Returns

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

9.3.3.7 `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 PGRGuid for the camera.

Returns

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

9.3.3.8 `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 PGRGuid for the camera.

See also

[GetCameraFromIndex\(\)](#)

Returns

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

9.3.3.9 **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.

9.3.3.10 **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 PGRGuid for the device.

See also

[GetNumOfDevices\(\)](#)

Returns

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

9.3.3.11 **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 PGRGuid to get the interface for.
<i>pInterface-Type</i>	The interface type of the PGRGuid .

Returns

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

9.3.3.12 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.

9.3.3.13 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.

9.3.3.14 virtual Error GetTopology ([TopologyNode](#) * *pNode*) [virtual]

Gets the topology information for the PC.

Parameters

<i>pNode</i>	TopologyNode object that will contain the topology information.
--------------	---

Returns

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

9.3.3.15 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. Bit 15:0 = Link Error Count. Default = 0. This field returns the number of link errors that occurred since the last reset. Bit 19:16 = Rx Lane Count. Default = 0. This field that identifies the number of Rx lanes that are active. Bit 23:20 = Tx Lane Count. Default = 0. This field that identifies the number of Tx lanes that are active. Bit 31:24 = Reserved.

Refer to XHCI 1.1 section 5.4.10 for Port Link Info:

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

Returns

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

9.3.3.16 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>	<p>Value read from the card register.</p> <p>Bit 0 = Current Connect Status. Default = 0. 1 = A device is connected to the port 0 = A device is not connected. This value reflects the current state of the port, and may not correspond to the physical state of the port.</p> <p>Bit 1 = Port Enabled/Disabled. Default = 0. 1 = Enabled. 0 = Disabled.</p> <p>Bit 2 = Reserved.</p> <p>Bit 3 = Over-current Active. Default = 0. 1 = This port currently has an over-current condition. 0 = This port does not have an over-current condition.</p> <p>Bit 4 = Port Reset. Default = 0. 1 = Port Reset signaling is asserted. 0 = Port is not in Reset.</p> <p>Bit 8:5 = Port Link State. Default = RxDetect(5). This field is used to power manage the port.</p> <p>Bit 9 = Port Power. Default = 1. This flag reflects a port's logical, power controlled state. 0 = This port is in the powered-off state. 1 = This port is not in the powered-off state.</p> <p>Bit 13:10 = Port Speed. Default = 0. This field identifies the speed of the connection. 0 : Undefined speed 1-15 : Protocol Speed ID (refer to other sections)</p> <p>Bit 15:14 = Port Indicator Control. Default = 0. 0 = Port indicators are off. 1 = Amber. 2 = Green. 3 = Undefined.</p> <p>Bit 16 = Port Link State Write Strobe. Default = 0. When this bit is set to 1 on a write reference to this register, this bit is set to 0.</p> <p>Bit 17 = Connect Status Change. Default = 0. 1 = Change in current connect status. 0 = No change.</p> <p>Bit 18 = Port Enabled/Disabled Change. Default = 0. 1 = change in PED. 0 = No change.</p> <p>Bit 19 = Warm Port Reset Change. Default = 0. This bit is set when Warm Reset procedure is complete. 0 = No change. 1 = Warm Reset complete.</p> <p>Bit 20 = Over current change. Default = 0. This bit shall be set to a 1 when there is a 0 to 1 or 1 to 0 transition.</p> <p>Bit 21 = Port Reset Change. Default = 0. This flag is set to 1 due to a 1 to 0 transition.</p> <p>Bit 22 = Port Link State Change. Default = 0. This flag is set to 1 due to PLS transition.</p> <p>Bit 23 = Port Config Error Change. Default = 0. This flag indicates that the port configuration has changed. 0 = No change. 1 = Port Config Error detected.</p> <p>Bit 24 = Cold Attach Status. Default = 0. 1 = Far-end receiver terminations were detected in the disconnected state. 0 - This flag is 0 if PP is 0 or for USB2 protocol parts.</p> <p>Bit 25 = Wake on Connect Enable. Default = 0. Writing this bit to a 1 enables the port to wake up.</p> <p>Bit 26 = Wake on Disconnect Enable. Default = 0. Writing this bit to a 1 enables the port to wake up.</p> <p>Bit 27 = Wake on Over-current Enable. Default = 0. Writing this bit to a 1 enables the port to wake up.</p> <p>Bit 29:28 = Reserved</p> <p>Bit 30 = Device Removable. This flag indicates if this port has a removable device. 1 = Device is non-removable. 0 = Device is removable.</p> <p>Bit 31 = Warm Port Reset. Default = 0. This flag shall always return 0 when read.</p>

Refer to XHCI 1.1 section 5.4.8 for Port Status:

eXtensible Host Controller interface for USB xHCI

Returns

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

9.3.3.17 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.

9.3.3.18 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>	PGRGuid of the device to read from.
<i>page</i>	Page to read from.
<i>port</i>	Port to read from.
<i>address</i>	Address to read from.
<i>pValue</i>	Value read from the phy register.

Returns

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

9.3.3.19 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.

9.3.3.20 virtual Error RescanBus () [virtual]

Force a rescan of the buses.

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

Returns

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

9.3.3.21 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.

9.3.3.22 `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>	PGRGuid of the device to write to.
<i>page</i>	Page to write to.
<i>port</i>	Port to write to.
<i>address</i>	Address to write to.
<i>value</i>	Value to write to phy register.

Returns

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

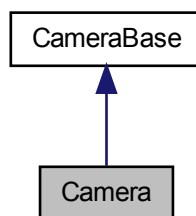
The documentation for this class was generated from the following file:

- [BusManager.h](#)

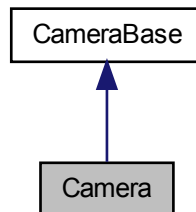
9.4 Camera Class Reference

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

Inheritance diagram for Camera:



Collaboration diagram for Camera:



Public Member Functions

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

They are only used for firewire and usb2 cameras.

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

Format7

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

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

9.4.1 Detailed Description

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

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

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

9.4.2 Constructor & Destructor Documentation

9.4.2.1 [Camera](#) ()

Default constructor.

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

Default destructor.

9.4.3 Member Function Documentation

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

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

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

Implements [CameraBase](#).

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

Implements [CameraBase](#).

9.4.3.3 [virtual Error DeregisterEvent](#) ([EventOptions](#) * *pOpts*) [virtual]

Implements [CameraBase](#).

9.4.3.4 [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](#).

9.4.3.5 `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](#).

9.4.3.6 `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](#).

9.4.3.7 `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](#).

9.4.3.8 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](#).

9.4.3.9 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](#).

9.4.3.10 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](#).

9.4.3.11 `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](#).

9.4.3.12 `virtual Error GetFormat7Configuration (Format7ImageSettings * pImageSettings, unsigned int * pPacketSize, float * pPercentage)` [virtual]

Get the current Format7 configuration from the camera.

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

Parameters

<i>pImage-Settings</i>	Current image settings.
<i>pPacketSize</i>	Current packet size.
<i>pPercentage</i>	Current packet size as a percentage.

See also

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

[GetVideoModeAndFrameRate\(\)](#)

Returns

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

9.4.3.13 `virtual Error GetFormat7Info (Format7Info * pInfo, bool * pSupported)`
[virtual]

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

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

Parameters

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

See also

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

Returns

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

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

Get the GPIO pin direction for the specified pin.

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

Parameters

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

See also

[SetGPIOPinDirection\(\)](#)

Returns

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

Implements [CameraBase](#).

9.4.3.15 `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](#).

9.4.3.16 `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](#).

9.4.3.17 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](#).

9.4.3.18 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](#).

9.4.3.19 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](#).

9.4.3.20 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 Property structure to be filled.
--------------	---

See also

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

Returns

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

Implements [CameraBase](#).

9.4.3.21 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 PropertyInfo structure to be filled.
------------------	---

See also

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

Returns

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

Implements [CameraBase](#).

9.4.3.22 `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](#).

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

Implements [CameraBase](#).

9.4.3.24 `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](#).

9.4.3.25 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](#).

9.4.3.26 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

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

Returns

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

Implements [CameraBase](#).

9.4.3.27 `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

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

Returns

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

Implements [CameraBase](#).

9.4.3.28 `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](#).

9.4.3.29 **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](#).

9.4.3.30 **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.

9.4.3.31 **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.

9.4.3.32 **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](#).

9.4.3.33 **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](#).

9.4.3.34 `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](#).

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

Implements [CameraBase](#).

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

Implements [CameraBase](#).

9.4.3.37 `virtual Error ResetStats ()` [virtual]

Implements [CameraBase](#).

9.4.3.38 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](#).

9.4.3.39 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 Image object to store image data.
---------------	--

See also

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

Returns

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

Implements [CameraBase](#).

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

Save the current settings to the specified current memory channel.

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

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

Set the LUT bank that will be used.

Parameters

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

Returns

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

Implements [CameraBase](#).

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

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

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

Parameters

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

See also

[StartCapture\(\)](#)

Returns

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

Implements [CameraBase](#).

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

Set the configuration associated with the camera object.

Parameters

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

See also

[GetConfiguration\(\)](#)

Returns

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

Implements [CameraBase](#).

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

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

Parameters

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

See also

[GetEmbeddedImageInfo\(\)](#)

Returns

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

Implements [CameraBase](#).

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

Set the current Format7 configuration to the camera.

Parameters

<i>pImage-Settings</i>	Image settings to be written to the camera.
<i>packetSize</i>	Packet size to be written to the camera.

See also

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

Returns

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

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

Set the current Format7 configuration to the camera.

Parameters

<i>pImage-Settings</i>	Image 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.

9.4.3.47 `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](#).

9.4.3.48 `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](#).

9.4.3.49 `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 Property structure to be used.
<i>broadcast</i>	Whether the action should be broadcast.

See also

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

Returns

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

Implements [CameraBase](#).

```
9.4.3.50 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](#).

```
9.4.3.51 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

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

Returns

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

Implements [CameraBase](#).

9.4.3.52 `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

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

Returns

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

Implements [CameraBase](#).

9.4.3.53 `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>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](#).

9.4.3.54 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.

9.4.3.55 `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](#).

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

9.4.3.57 `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](#).

```
9.4.3.58 virtual Error ValidateFormat7Settings ( const Format7ImageSettings *
        pImageSettings, bool * pSettingsAreValid, Format7PacketInfo * pPacketInfo )
        [virtual]
```

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

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

Parameters

<i>pImage-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.

```
9.4.3.59 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 Image 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](#).

9.4.3.60 `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](#).

9.4.3.61 `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](#).

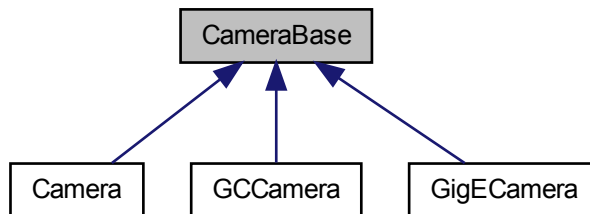
The documentation for this class was generated from the following file:

- [Camera.h](#)

9.5 CameraBase Class Reference

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

Inheritance diagram for CameraBase:



Public Member Functions

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

Protected Attributes

- CameraData * [m_pCameraData](#)

Connection and Image Retrieval

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

- virtual [Error Connect](#) ([PGRGuid](#) *pGuid=NULL)=0
Connects the camera object to the camera specified by the GUID.
- virtual [Error Disconnect](#) ()=0
Disconnects the camera object from the camera.
- virtual bool [IsConnected](#) ()=0
Checks if the camera object is connected to a physical camera specified by a GUID.
- virtual [Error SetCallback](#) ([ImageEventCallback](#) callbackFn, const void *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.

9.5.1 Detailed Description

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

9.5.2 Constructor & Destructor Documentation

9.5.2.1 [CameraBase](#) () [inline]

Default constructor.

9.5.2.2 virtual ~CameraBase () [inline, virtual]

Default destructor.

9.5.3 Member Function Documentation

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

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

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

Parameters

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

See also

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

Returns

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

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

9.5.3.2 virtual Error DeregisterAllEvents (void) [pure virtual]

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

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

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

9.5.3.4 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](#).

9.5.3.5 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](#).

9.5.3.6 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](#).

9.5.3.7 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](#).

9.5.3.8 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](#).

9.5.3.9 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](#).

9.5.3.10 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](#).

9.5.3.11 `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](#).

9.5.3.12 `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](#).

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

Returns

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

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

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

Get the LUT channel settings from the camera.

Parameters

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

See also

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

Returns

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

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

9.5.3.15 `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](#).

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

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

9.5.3.17 `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](#).

9.5.3.18 `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 Property structure to be filled.
--------------	---

See also

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

Returns

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

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

9.5.3.19 `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 PropertyInfo structure to be filled.
------------------	---

See also

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

Returns

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

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

9.5.3.20 `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](#).

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

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

9.5.3.22 `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](#).

9.5.3.23 `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](#).

9.5.3.24 `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](#).

9.5.3.25 `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](#).

9.5.3.26 `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](#).

9.5.3.27 `virtual Error GetTriggerModelInfo (TriggerModelInfo * pTriggerModelInfo)`
[pure 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.

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

9.5.3.28 `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](#).

9.5.3.29 `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](#).

9.5.3.30 `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](#).

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

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

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

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

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

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

9.5.3.34 `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](#).

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

Retrieves the the next image object containing the next image.

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

Parameters

<i>pImage</i>	Pointer to Image object to store image data.
---------------	--

See also

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

Returns

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

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

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

Save the current settings to the specified current memory channel.

Parameters

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

See also

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

Returns

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

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

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

Set the LUT bank that will be used.

Parameters

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

Returns

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

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

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

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

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

Parameters

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

See also

[StartCapture\(\)](#)

Returns

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

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

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

Set the configuration associated with the camera object.

Parameters

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

See also

[GetConfiguration\(\)](#)

Returns

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

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

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

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

Parameters

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

See also

[GetEmbeddedImageInfo\(\)](#)

Returns

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

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

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

Set the GPIO pin direction for the specified pin.

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

Parameters

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

See also

[GetGPIOPinDirection\(\)](#)

Returns

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

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

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

Set the LUT channel settings to the camera.

Parameters

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

See also

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

Returns

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

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

9.5.3.43 `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 Property structure to be used.
<i>broadcast</i>	Whether the action should be broadcast.

See also

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

Returns

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

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

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

Set current strobe settings to the camera.

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

Parameters

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

See also

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

Returns

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

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

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

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

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

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

9.5.3.47 `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](#).

9.5.3.48 `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](#).

```
9.5.3.49 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. This function is only used with firewire cameras.

Parameters

<i>num-Cameras</i>	Number of Camera objects in the ppCameras array.
<i>ppCameras</i>	Array of pointers to Camera 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.

9.5.3.50 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](#).

9.5.3.51 virtual Error WaitForBufferEvent (Image * *plImage*, unsigned int *eventNumber*) [pure virtual]

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

Parameters

<i>plImage</i>	Pointer to Image 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](#).

9.5.3.52 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](#).

```
9.5.3.53 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](#).

9.5.4 Member Data Documentation

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

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

- [CameraBase.h](#)

9.6 CameraControlDlg Class Reference

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

Public Member Functions

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

9.6.1 Detailed Description

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

9.6.2 Constructor & Destructor Documentation

9.6.2.1 CameraControlDlg ()

Default constructor.

9.6.2.2 ~CameraControlDlg ()

Default destructor.

9.6.3 Member Function Documentation

9.6.3.1 void Connect (CameraBase * *pCamera*)

Connect dialog to a camera.

Parameters

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

9.6.3.2 void Disconnect ()

Disconnect a connected camera from the dialog.

9.6.3.3 void Hide ()

Hide the dialog.

9.6.3.4 bool IsVisible ()

Get the visibility of the dialog.

Returns

Whether the dialog is visible.

9.6.3.5 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.
--------------	--

9.6.3.6 void Show ()

Show the dialog.

9.6.3.7 void Show (void * *pParent*)

Show the dialog.

9.6.3.8 void ShowModal ()

Show the modal dialog.

9.6.3.9 void ShowModal (void * *pParent*)

Show the modal dialog.

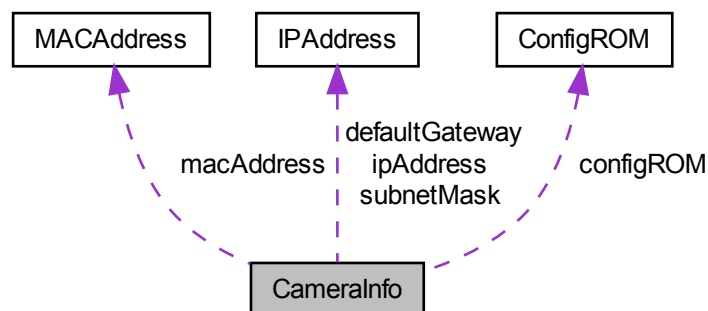
The documentation for this class was generated from the following file:

- [FlyCapture2GUI.h](#)

9.7 CameraInfo Struct Reference

[Camera](#) information.

Collaboration diagram for CameraInfo:



Public Member Functions

- [CameraInfo](#) ()

Public Attributes

- unsigned int [serialNumber](#)
Device serial number.
- [InterfaceType](#) [interfaceType](#)
Interface type.
- [DriverType](#) [driverType](#)
Driver type.
- bool [isColorCamera](#)
Flag indicating if this is a color camera.
- char [modelName](#) [[sk_maxStringLength](#)]
Device model name.
- char [vendorName](#) [[sk_maxStringLength](#)]
Device vendor name.
- char [sensorInfo](#) [[sk_maxStringLength](#)]
String detailing the sensor information.
- char [sensorResolution](#) [[sk_maxStringLength](#)]
String providing the sensor resolution.
- char [driverName](#) [[sk_maxStringLength](#)]
Driver name of driver being used.
- char [firmwareVersion](#) [[sk_maxStringLength](#)]
Firmware version of camera.
- char [firmwareBuildTime](#) [[sk_maxStringLength](#)]
Firmware build time.
- [BusSpeed](#) [maximumBusSpeed](#)
Maximum bus speed.
- [BayerTileFormat](#) [bayerTileFormat](#)
Bayer tile format.
- unsigned short [busNumber](#)
Bus number, set to 0 for GigE and USB cameras.
- unsigned short [nodeNumber](#)
ieee1394 Node number, set to 0 for GigE and USB cameras
- [PCleBusSpeed](#) [pcieBusSpeed](#)
PCIe Bus Speed, set to PCIE_BUSSPEED_UNKNOWN for unsupported drivers.
- unsigned int [reserved](#) [16]
Reserved for future use.

IIDC specific information

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

GigE specific information

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

9.7.1 Detailed Description

[Camera](#) information.

9.7.2 Constructor & Destructor Documentation

9.7.2.1 [CameraInfo](#) () `[inline]`

9.7.3 Member Data Documentation

9.7.3.1 unsigned int [applicationIPAddress](#)

Local Application IP Address.

9.7.3.2 unsigned int [applicationPort](#)

Local Application port.

9.7.3.3 [BayerTileFormat](#) [bayerTileFormat](#)

Bayer tile format.

9.7.3.4 unsigned short busNumber

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

9.7.3.5 unsigned int ccpStatus

Status/Content of CCP register.

9.7.3.6 ConfigROM configROM

Configuration ROM data.

9.7.3.7 IPAddress defaultGateway

Default gateway.

9.7.3.8 char driverName[sk_maxStringLength]

Driver name of driver being used.

9.7.3.9 DriverType driverType

Driver type.

9.7.3.10 char firmwareBuildTime[sk_maxStringLength]

Firmware build time.

9.7.3.11 char firmwareVersion[sk_maxStringLength]

Firmware version of camera.

9.7.3.12 unsigned int gigEMajorVersion

GigE Vision version.

9.7.3.13 unsigned int gigEMinorVersion

GigE Vision minor version.

9.7.3.14 unsigned int iidcVer

DCAM version.

9.7.3.15 InterfaceType interfaceType

Interface type.

9.7.3.16 IPAddress ipAddress

IP address.

9.7.3.17 bool isColorCamera

Flag indicating if this is a color camera.

9.7.3.18 MACAddress macAddress

MAC address.

9.7.3.19 BusSpeed maximumBusSpeed

Maximum bus speed.

9.7.3.20 char modelName[sk_maxStringLength]

Device model name.

9.7.3.21 unsigned short nodeNumber

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

9.7.3.22 PCIeBusSpeed pcieBusSpeed

PCIe Bus Speed, set to PCIE_BUSSPEED_UNKNOWN for unsupported drivers.

9.7.3.23 unsigned int reserved[16]

Reserved for future use.

9.7.3.24 char sensorInfo[sk_maxStringLength]

String detailing the sensor information.

9.7.3.25 char sensorResolution[sk_maxStringLength]

String providing the sensor resolution.

9.7.3.26 unsigned int serialNumber

Device serial number.

9.7.3.27 IPAddress subnetMask

Subnet mask.

9.7.3.28 char userDefinedName[sk_maxStringLength]

User defined name.

9.7.3.29 char vendorName[sk_maxStringLength]

Device vendor name.

9.7.3.30 char xmlURL1[sk_maxStringLength]

XML URL 1.

9.7.3.31 char xmlURL2[sk_maxStringLength]

XML URL 2.

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

- [FlyCapture2Defs.h](#)

9.8 CameraSelectionDlg Class Reference

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

Public Member Functions

- [CameraSelectionDlg](#) ()
Default constructor.
- [~CameraSelectionDlg](#) ()
Default destructor.
- void [ShowModal](#) (bool *pOk, [PGRGuid](#) *pGuid, unsigned int *pSize)
Show the [CameraSelectionDlg](#).
- void [SetTitle](#) (const char *title)
Set the window title.

9.8.1 Detailed Description

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

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

9.8.2 Constructor & Destructor Documentation

9.8.2.1 [CameraSelectionDlg](#) ()

Default constructor.

9.8.2.2 [~CameraSelectionDlg](#) ()

Default destructor.

9.8.3 Member Function Documentation

9.8.3.1 void [SetTitle](#) (const char * *title*)

Set the window title.

Parameters

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

9.8.3.2 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 PGRGuid array.

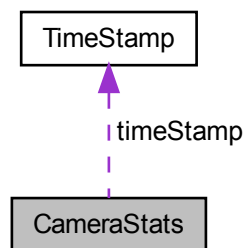
The documentation for this class was generated from the following file:

- [FlyCapture2GUI.h](#)

9.9 CameraStats Struct Reference

[Camera](#) diagnostic information.

Collaboration diagram for CameraStats:



Public Member Functions

- [CameraStats](#) ()

Public Attributes

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

- float [cameraVoltages](#) [8]
- 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.

9.9.1 Detailed Description

[Camera](#) diagnostic information.

9.9.2 Constructor & Destructor Documentation

9.9.2.1 [CameraStats](#) () [[inline](#)]

9.9.3 Member Data Documentation

9.9.3.1 float [cameraCurrents](#)[8]

9.9.3.2 bool [cameraPowerUp](#)

9.9.3.3 float [cameraVoltages](#)[8]

9.9.3.4 unsigned int [imageCorrupt](#)

9.9.3.5 unsigned int [imageDriverDropped](#)

9.9.3.6 unsigned int [imageDropped](#)

9.9.3.7 unsigned int [imageXmitFailed](#)

9.9.3.8 unsigned int [numCurrents](#)

The number of current registers available.

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

9.9.3.9 unsigned int **numResendPacketsReceived**

9.9.3.10 unsigned int **numResendPacketsRequested**

9.9.3.11 unsigned int **numVoltages**

The number of voltage registers available.

0: the values in cameraVoltages[] are invalid.

9.9.3.12 unsigned int **portErrors**

9.9.3.13 unsigned int **regReadFailed**

9.9.3.14 unsigned int **regWriteFailed**

9.9.3.15 unsigned int **reserved[16]**

Reserved for future use.

9.9.3.16 unsigned int **temperature**

9.9.3.17 unsigned int **timeSinceBusReset**

9.9.3.18 unsigned int **timeSinceInitialization**

9.9.3.19 **TimeStamp** **timeStamp**

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

- [FlyCapture2Defs.h](#)

9.10 ConfigROM Struct Reference

[Camera](#) configuration ROM.

Public Member Functions

- [ConfigROM](#) ()

Public Attributes

- unsigned int [nodeVendorId](#)
Vendor ID of a node.

- unsigned int [chipIdHi](#)
Chip ID (high part).
- unsigned int [chipIdLo](#)
Chip ID (low part).
- unsigned int [unitSpecId](#)
Unit Spec ID, usually 0xa02d.
- unsigned int [unitSWVer](#)
Unit software version.
- unsigned int [unitSubSWVer](#)
Unit sub software version.
- unsigned int [vendorUniqueInfo_0](#)
Vendor unique info 0.
- unsigned int [vendorUniqueInfo_1](#)
Vendor unique info 1.
- unsigned int [vendorUniqueInfo_2](#)
Vendor unique info 2.
- unsigned int [vendorUniqueInfo_3](#)
Vendor unique info 3.
- char [pszKeyword](#) [[sk_maxStringLength](#)]
Keyword.
- unsigned int [reserved](#) [16]
Reserved for future use.

9.10.1 Detailed Description

[Camera](#) configuration ROM.

9.10.2 Constructor & Destructor Documentation

9.10.2.1 [ConfigROM](#)() [[inline](#)]

9.10.3 Member Data Documentation

9.10.3.1 unsigned int [chipIdHi](#)

Chip ID (high part).

9.10.3.2 unsigned int [chipIdLo](#)

Chip ID (low part).

9.10.3.3 unsigned int nodeVendorId

Vendor ID of a node.

9.10.3.4 char pszKeyword[sk_maxStringLength]

Keyword.

9.10.3.5 unsigned int reserved[16]

Reserved for future use.

9.10.3.6 unsigned int unitSpecId

Unit Spec ID, usually 0xa02d.

9.10.3.7 unsigned int unitSubSWVer

Unit sub software version.

9.10.3.8 unsigned int unitSWVer

Unit software version.

9.10.3.9 unsigned int vendorUniqueInfo_0

Vendor unique info 0.

9.10.3.10 unsigned int vendorUniqueInfo_1

Vendor unique info 1.

9.10.3.11 unsigned int vendorUniqueInfo_2

Vendor unique info 2.

9.10.3.12 unsigned int vendorUniqueInfo_3

Vendor unique info 3.

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

- [FlyCapture2Defs.h](#)

9.11 EmbeddedImageInfo Struct Reference

Properties of the possible embedded image information.

Collaboration diagram for EmbeddedImageInfo:



Public Attributes

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

9.11.1 Detailed Description

Properties of the possible embedded image information.

9.11.2 Member Data Documentation

9.11.2.1 `EmbeddedImageInfoProperty` `brightness`

9.11.2.2 `EmbeddedImageInfoProperty` `exposure`

9.11.2.3 `EmbeddedImageInfoProperty` `frameCounter`

9.11.2.4 `EmbeddedImageInfoProperty` `gain`

9.11.2.5 `EmbeddedImageInfoProperty` `GPIOPinState`

9.11.2.6 `EmbeddedImageInfoProperty` `ROIPosition`

9.11.2.7 `EmbeddedImageInfoProperty` `shutter`

9.11.2.8 `EmbeddedImageInfoProperty` `strobePattern`

9.11.2.9 `EmbeddedImageInfoProperty` `timestamp`

9.11.2.10 `EmbeddedImageInfoProperty` `whiteBalance`

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

- [FlyCapture2Defs.h](#)

9.12 `EmbeddedImageInfoProperty` Struct Reference

Properties of a single embedded image info property.

Public Member Functions

- [EmbeddedImageInfoProperty](#) ()

Public Attributes

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

9.12.1 Detailed Description

Properties of a single embedded image info property.

9.12.2 Constructor & Destructor Documentation

9.12.2.1 EmbeddedImageInfoProperty () [inline]

9.12.3 Member Data Documentation

9.12.3.1 bool available

Whether this property is available.

9.12.3.2 bool onOff

Whether this property is on or off.

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

- [FlyCapture2Defs.h](#)

9.13 Error Class Reference

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

Public Member Functions

- [Error](#) ()
Default constructor.
- [Error](#) (const [Error](#) &error)
Copy constructor.
- virtual [~Error](#) ()
Default destructor.
- virtual [Error](#) & [operator=](#) (const [Error](#) &error)
Assignment operator.
- virtual bool [operator==](#) (const [Error](#) &error) const
Equality operator.
- virtual bool [operator==](#) (const [ErrorType](#) &errorType) const
Equality operator.
- virtual bool [operator!=](#) (const [Error](#) &error) const
Inequality operator.
- virtual bool [operator!=](#) (const [ErrorType](#) &errorType) const
Inequality operator.
- virtual [ErrorType](#) [GetType](#) () const
Retrieve the ErrorType of the error.
- virtual const char * [GetDescription](#) () const

Retrieve the top level description of the error that occurred.

- virtual unsigned int [GetLine](#) () const

Retrieve the line number where the error originated.

- virtual const char * [GetFilename](#) () const

Retrieve the source filename where the error originated.

- virtual [Error GetCause](#) () const

Get the error which caused this error.

- virtual const char * [GetBuildDate](#) () const

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

- virtual const char * [CollectSupportInformation](#) () const

Retrieve the support information.

- virtual void [PrintErrorTrace](#) () const

Print a formatted log trace to stderr.

Friends

- class [InternalError](#)

9.13.1 Detailed Description

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

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

9.13.2 Constructor & Destructor Documentation

9.13.2.1 [Error](#) ()

Default constructor.

9.13.2.2 [Error](#) (const [Error](#) & *error*)

Copy constructor.

9.13.2.3 virtual [~Error](#) () [virtual]

Default destructor.

9.13.3 Member Function Documentation

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

Retrieve the support information.

It is not implemented in this release.

Returns

A string containing support information.

9.13.3.2 `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.

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

Get the error which caused this error.

Returns

An error object representing the cause of this error.

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

Retrieve the top level description of the error that occurred.

Returns

A string with the error description.

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

Retrieve the source filename where the error originated.

Returns

A string with the file name.

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

Retrieve the line number where the error originated.

Returns

The line number.

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

Retrieve the ErrorType of the error.

Returns

The ErrorType of the error.

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

Inequality operator.

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

Inequality operator.

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

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

Assignment operator.

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

Equality operator.

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

Equality operator.

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

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

Print a formatted log trace to stderr.

9.13.4 Friends And Related Function Documentation

9.13.4.1 friend class InternalError [friend]

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

- [Error.h](#)

9.14 EventCallbackData Struct Reference

Public Attributes

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

9.14.1 Member Data Documentation

9.14.1.1 void* EventData

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

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

9.14.1.2 size_t EventDataSize

The size of the structure pointed to by EventData.

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

9.14.1.3 long long unsigned EventID

The device register which EventName maps to.

Provides an alternate means of indexing into different event types.

9.14.1.4 const char* EventName

The event name used to register the event.

Provided so the user knows which event triggered the callback.

9.14.1.5 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.

9.14.1.6 void* EventUserData

Pointer to the user-supplied data struct.

9.14.1.7 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](#)

9.15 EventOptions Struct Reference

Options for enabling device event registration.

Public Attributes

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

9.15.1 Detailed Description

Options for enabling device event registration.

9.15.2 Member Data Documentation

9.15.2.1 CameraEventCallback EventCallbackFcn

Callback function pointer.

9.15.2.2 const char* EventName

Event name to register.

9.15.2.3 const void* EventUserData

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

9.15.2.4 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](#)

9.16 FC2Config Struct Reference

Configuration for a camera.

Public Member Functions

- [FC2Config](#) ()

Public Attributes

- unsigned int [numBuffers](#)
Number of buffers used by the [FlyCapture2](#) library to grab images.
- unsigned int [numImageNotifications](#)
Number of notifications per image.
- unsigned int [minNumImageNotifications](#)
Minimum number of notifications needed for the current image settings on the camera.

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

9.16.1 Detailed Description

Configuration for a camera.

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

9.16.2 Constructor & Destructor Documentation

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

9.16.3 Member Data Documentation

9.16.3.1 [BusSpeed asyncBusSpeed](#)

Asynchronous bus speed.

9.16.3.2 [BandwidthAllocation bandwidthAllocation](#)

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

9.16.3.3 GrabMode grabMode

Grab mode for the camera.

The default is DROP_FRAMES.

9.16.3.4 int grabTimeout

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

9.16.3.5 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.

9.16.3.6 BusSpeed isochBusSpeed

Isochronous bus speed.

9.16.3.7 unsigned int minNumImageNotifications

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

Read-only value.

9.16.3.8 unsigned int numBuffers

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

9.16.3.9 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.

9.16.3.10 unsigned int registerTimeout

Register read/write timeout value, in microseconds.

The default value is dependent on the interface type.

9.16.3.11 unsigned int registerTimeoutRetries

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

The default value is 0.

9.16.3.12 unsigned int reserved[16]

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

9.17 FC2Version Struct Reference

The current version of the library.

Public Attributes

- unsigned int [major](#)
Major version number.
- unsigned int [minor](#)
Minor version number.
- unsigned int [type](#)
Type version number.
- unsigned int [build](#)
Build version number.

9.17.1 Detailed Description

The current version of the library.

9.17.2 Member Data Documentation

9.17.2.1 unsigned int build

Build version number.

9.17.2.2 unsigned int major

Major version number.

9.17.2.3 unsigned int minor

Minor version number.

9.17.2.4 unsigned int type

Type version number.

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

- [FlyCapture2Defs.h](#)

9.18 FlyCapture2Video Class Reference

The [FlyCapture2Video](#) class provides the functionality for the user to record images to an AVI file.

Public Member Functions

- [FlyCapture2Video](#) ()
Default constructor.
- virtual [~FlyCapture2Video](#) ()
Default destructor.
- virtual [Error Open](#) (const char *pFileName, [AVIOption](#) *pOption)
Open an AVI file in preparation for writing Images to disk.
- virtual [Error Open](#) (const char *pFileName, [MJPGOption](#) *pOption)
Open an MJPEG AVI file in preparation for writing Images to disk.
- virtual [Error Open](#) (const char *pFileName, [H264Option](#) *pOption)
Open an H.264 video file in preparation for writing Images to disk.

- virtual [Error Append](#) ([Image](#) *pImage)
Append an image to the AVI/MP4 file.
- virtual [Error Close](#) ()
Close the AVI/MP4 file.
- virtual void [SetMaximumFileSize](#) (unsigned int size)
Set the maximum file size (in megabytes) of a AVI/MP4 file.

9.18.1 Detailed Description

The [FlyCapture2Video](#) class provides the functionality for the user to record images to an AVI file.

9.18.2 Constructor & Destructor Documentation

9.18.2.1 FlyCapture2Video ()

Default constructor.

9.18.2.2 virtual ~FlyCapture2Video () [virtual]

Default destructor.

9.18.3 Member Function Documentation

9.18.3.1 virtual Error Append (Image * pImage) [virtual]

Append an image to the AVI/MP4 file.

Parameters

<i>pImage</i>	The image to append.
---------------	----------------------

Returns

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

9.18.3.2 virtual Error Close () [virtual]

Close the AVI/MP4 file.

See also

[Open\(\)](#)

Returns

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

9.18.3.3 virtual Error Open (const char * *pFileName*, AVIOption * *pOption*)
[virtual]

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

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

Parameters

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

See also

[SetMaximumFileSize\(\)](#)
[Close\(\)](#)

Returns

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

9.18.3.4 virtual Error Open (const char * *pFileName*, MJPGOption * *pOption*)
[virtual]

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

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

Parameters

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

See also

[SetMaximumFileSize\(\)](#)
[Close\(\)](#)
[MJPGOption](#)

Returns

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

9.18.3.5 virtual Error Open (const char * *pFileName*, H264Option * *pOption*) [virtual]

Open an H.264 video file in preparation for writing Images to disk.

If the file extension is not specified, MP4 will be used as the default container. Consult ffmpeg documentation for a list of supported containers.

Parameters

<i>pFileName</i>	The filename of the video file.
<i>pOption</i>	H.264 options to apply to the video file.

See also

[Close\(\)](#)
[H264Option](#)

Returns

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

9.18.3.6 virtual void SetMaximumFileSize (unsigned int *size*) [virtual]

Set the maximum file size (in megabytes) of a AVI/MP4 file.

A new AVI/MP4 file is created automatically when file size limit is reached. Setting a maximum size of 0 indicates no limit on file size.

Parameters

<i>size</i>	The maximum AVI file size in MB.
-------------	----------------------------------

See also

[Append\(\)](#)

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

- [FlyCapture2Video.h](#)

9.19 FlyCapture3ApiGuiWrapper Class Reference

Public Member Functions

- WRAPPER_API [FlyCapture3ApiGuiWrapper](#) (void)
- WRAPPER_API [~FlyCapture3ApiGuiWrapper](#) (void)

- WRAPPER_API void [ConnectGUILibrary](#) (FlyCapture2::GCCamera &camera)
- WRAPPER_API void [DisconnectGUILibrary](#) ()
- WRAPPER_API void [ShowPropertyGridDialog](#) ()
- WRAPPER_API void [ShowCameraSelectionDialog](#) ()
- WRAPPER_API int [GetNumDialogs](#) ()
- WRAPPER_API std::list < std::string > [GetDialogNameList](#) ()
- WRAPPER_API void [ShowDialogByName](#) (std::string dialogName)
- WRAPPER_API void [ShowDialogByIndex](#) (int index)
- WRAPPER_API int [GetNumOfControls](#) ()
- WRAPPER_API std::list < std::string > [GetControlNameList](#) ()

9.19.1 Constructor & Destructor Documentation

9.19.1.1 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

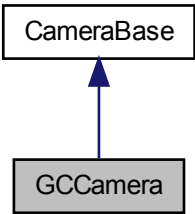
Minimum bytes per packet.

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

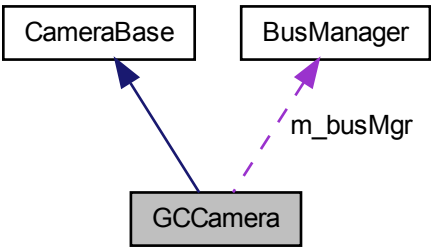
- [FlyCapture2Defs.h](#)

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>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](#).

9.23.2.15 `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 Property 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 PropertyInfo 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>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](#).

9.23.2.25 **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>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](#).

9.23.2.28 `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](#).

9.23.2.29 `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 * *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 Image 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 specified current memory channel.

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

9.23.2.40 **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

[SetGPIOPinDirection\(\)](#)

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 <code>numEntries</code> returned by <code>GetLutInfo()</code> .
<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 Property 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

[GetTriggerModelInfo\(\)](#)
[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

[GetTriggerModelInfo\(\)](#)
[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>pCallbackData</i>	A pointer to data that can be passed to the callback function.

See also

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

Returns

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

Implements [CameraBase](#).

```
9.23.2.54 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 Image 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]

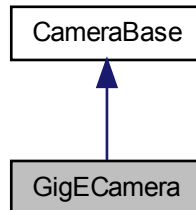
The documentation for this class was generated from the following file:

- [GCCamera.h](#)

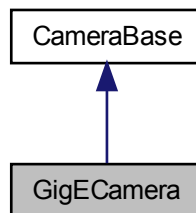
9.24 GigECamera Class Reference

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** GetGigElmageSettings (**GigElmageSettings** * *plmageSettings*)
[virtual]

Get the current image settings on the camera.

Parameters

<i>plmageSettings</i>	Current image settings on camera.
-----------------------	-----------------------------------

Returns

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

9.24.3.16 **virtual Error** GetGigElmageSettingsInfo (**GigElmageSettingsInfo** * *plInfo*)
[virtual]

Get information about the image settings possible on the camera.

Parameters

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

Returns

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

9.24.3.17 **virtual Error** GetGigElmagingMode (**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 Property 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 PropertyInfo 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>pStrobeControl</i>	Structure to receive strobe settings.
-----------------------	---------------------------------------

See also

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

Returns

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

Implements [CameraBase](#).

9.24.3.32 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>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](#).

9.24.3.34 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](#).

9.24.3.35 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](#).

9.24.3.36 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](#).

9.24.3.37 `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 * *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 Image 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 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](#).

9.24.3.50 **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>pCallbackData</i>	A pointer to data that can be passed to the callback function.

See also

[StartCapture\(\)](#)

Returns

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

Implements [CameraBase](#).

9.24.3.52 **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 * plImageSettings
) [virtual]`

Set the image settings specified to the camera.

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

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

Set the specified trigger delay settings to the camera.

Parameters

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

See also

[GetTriggerModeInfo\(\)](#)
[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

[GetTriggerModeInfo\(\)](#)
[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>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](#).

```
9.24.3.68 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 Image 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 GigEImageSettingsInfo Struct Reference

Format 7 information for a single mode.

Public Member Functions

- [GigEImageSettingsInfo](#) ()

Public Attributes

- unsigned int [maxWidth](#)
Maximum image width.
- unsigned int [maxHeight](#)
Maximum image height.

- unsigned int [offsetHStepSize](#)
Horizontal step size for the offset.
- unsigned int [offsetVStepSize](#)
Vertical step size for the offset.
- unsigned int [imageHStepSize](#)
Horizontal step size for the image.
- unsigned int [imageVStepSize](#)
Vertical step size for the image.
- unsigned int [pixelFormatBitField](#)
Supported pixel formats in a bit field.
- unsigned int [vendorPixelFormatBitField](#)
Vendor unique pixel formats in a bit field.
- unsigned int [reserved](#) [16]
Reserved for future use.

9.27.1 Detailed Description

Format 7 information for a single mode.

9.27.2 Constructor & Destructor Documentation

9.27.2.1 [GigEImageSettingsInfo](#) () `[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.

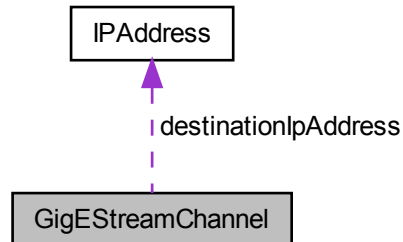
The documentation for this struct was generated from the following file:

- [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:

- [FlyCapture2VideoDefs.h](#)

9.31 Image Class Reference

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

Public Member Functions

- [Image](#) ()
Default constructor.
- [Image](#) (unsigned int rows, unsigned int cols, unsigned int stride, unsigned char *pData, unsigned int dataSize, [PixelFormat](#) format, [BayerTileFormat](#) bayerFormat=NONE)
Construct an [Image](#) object with the specified arguments.
- [Image](#) (unsigned int rows, unsigned int cols, unsigned int stride, unsigned char *pData, unsigned int dataSize, unsigned int receivedDataSize, [PixelFormat](#) format, [BayerTileFormat](#) bayerFormat=NONE)
Construct an [Image](#) object with the specified arguments.
- [Image](#) (unsigned char *pData, unsigned int dataSize)
Construct an [Image](#) object with the specified arguments.
- [Image](#) (unsigned int rows, unsigned int cols, [PixelFormat](#) format, [BayerTileFormat](#) bayerFormat=NONE)
Construct an [Image](#) object with the specified arguments.
- [Image](#) (const [Image](#) &image)
Copy constructor.
- virtual [~Image](#) ()
Default destructor.
- virtual [Image](#) & [operator=](#) (const [Image](#) &image)
Assignment operator.
- virtual unsigned char * [operator\[\]](#) (unsigned int index)
Indexing operator.
- virtual unsigned char * [operator\(\)](#) (unsigned int row, unsigned int col)
Indexing operator.
- virtual [Error DeepCopy](#) (const [Image](#) *pImage)
Perform a deep copy of the [Image](#).
- virtual [Error SetDimensions](#) (unsigned int rows, unsigned int cols, unsigned int stride, [PixelFormat](#) 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](#) default-Method)

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 ImageStatistics 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 * *pImage*) [virtual]

Perform a deep copy of the [Image](#).

After this operation, the image contents and member variables will be the same. The Images will not share a buffer. The Image's current buffer will not be released.

Parameters

<i>pImage</i>	The Image 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*, JPG2Option * *pOption*)**
[virtual]

Save the image to the specified file name with the options specified.

Parameters

<i>pFilename</i>	Filename to save image with.
<i>pOption</i>	Options to use while saving image.

Returns

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

9.31.3.34 `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>pPixelValue-Mean</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>pPixelValueMin</i>	The minimum pixel value.
<i>pPixelValueMax</i>	The maximum pixel value.
<i>pNumPixelValues</i>	The number of unique pixel values.
<i>pPixelValueMean</i>	The mean of the image.
<i>ppHistogram</i>	Pointer to an array containing the histogram.

Returns

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

9.33.4.13 ImageStatistics& operator= (const ImageStatistics & *other*)

Assignment operator.

Parameters

<i>other</i>	The ImageStatistics 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:

- [FlyCapture2VideoDefs.h](#)

9.41 NodeMap Class Reference

Public Member Functions

- [NodeMap](#) (GenApi::CNodeMapRef *ref)
- virtual [~NodeMap](#) (void)
- GenICam::gcstring [_GetDeviceName](#) ()
Get device name.
- void [_Poll](#) (int64_t ElapsedTime)
Fires nodes which have a polling time.

- void [_GetNodes](#) (NodeList_t &Nodes)
Retrieves all nodes in the node map.
- INode * [_GetNode](#) (const GenICam::gcstring &key)
Retrieves the node from the central map by name.
- void [_InvalidateNodes](#) () const
Invalidates all nodes.

9.41.1 Constructor & Destructor Documentation

9.41.1.1 **NodeMap** (GenApi::CNodeMapRef * *ref*)

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

9.41.2 Member Function Documentation

9.41.2.1 GenICam::gcstring **_GetDeviceName** ()

Get device name.

9.41.2.2 INode* **_GetNode** (const GenICam::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 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 StrobeControl Struct Reference

A camera strobe.

Public Member Functions

- [StrobeControl](#) ()

Public Attributes

- unsigned int [source](#)
Source value.
- bool [onOff](#)
Flag controlling on/off.
- unsigned int [polarity](#)
Signal polarity.
- float [delay](#)
Signal delay (in ms).
- float [duration](#)
Signal duration (in ms).
- unsigned int [reserved](#) [8]
Reserved for future use.

9.48.1 Detailed Description

A camera strobe.

9.48.2 Constructor & Destructor Documentation

9.48.2.1 [StrobeControl](#) () `[inline]`

9.48.3 Member Data Documentation

9.48.3.1 float [delay](#)

Signal delay (in ms).

9.48.3.2 float duration

Signal duration (in ms).

9.48.3.3 bool onOff

Flag controlling on/off.

9.48.3.4 unsigned int polarity

Signal polarity.

9.48.3.5 unsigned int reserved[8]

Reserved for future use.

9.48.3.6 unsigned int source

Source value.

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

- [FlyCapture2Defs.h](#)

9.49 StrobelInfo Struct Reference

A camera strobe property.

Public Member Functions

- [StrobelInfo](#) ()

Public Attributes

- unsigned int [source](#)
Source value.
- bool [present](#)
Presence of strobe.
- bool [readOutSupported](#)
Flag indicating if strobe value can be read out.
- bool [onOffSupported](#)
Flag indicating if on/off is supported.

- bool [polaritySupported](#)
Flag indicating if polarity is supported.
- float [minValue](#)
Minimum value.
- float [maxValue](#)
Maximum value.
- unsigned int [reserved](#) [8]
Reserved for future use.

9.49.1 Detailed Description

A camera strobe property.

9.49.2 Constructor & Destructor Documentation

9.49.2.1 `StrobeInfo ()` `[inline]`

9.49.3 Member Data Documentation

9.49.3.1 float `maxValue`

Maximum value.

9.49.3.2 float `minValue`

Minimum value.

9.49.3.3 bool `onOffSupported`

Flag indicating if on/off is supported.

9.49.3.4 bool `polaritySupported`

Flag indicating if polarity is supported.

9.49.3.5 bool `present`

Presence of strobe.

9.49.3.6 bool `readOutSupported`

Flag indicating if strobe value can be read out.

9.49.3.7 unsigned int reserved[8]

Reserved for future use.

9.49.3.8 unsigned int source

Source value.

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

- [FlyCapture2Defs.h](#)

9.50 SyncManager Class Reference

Public Member Functions

- MULTISYNCLIBRARY_API [SyncManager](#) ()
- MULTISYNCLIBRARY_API [~SyncManager](#) ()
- MULTISYNCLIBRARY_API [PGRSyncError Start](#) ()
- MULTISYNCLIBRARY_API [PGRSyncError Stop](#) ()
- MULTISYNCLIBRARY_API [PGRSyncError RescanMasterTimingBus](#) ()
- MULTISYNCLIBRARY_API [PGRSyncMessage GetSyncStatus](#) ()
- MULTISYNCLIBRARY_API double [GetTimeSinceSynced](#) ()
- MULTISYNCLIBRARY_API bool [IsTimingBusConnected](#) ()
- MULTISYNCLIBRARY_API bool [EnableCrossPCSynchronization](#) ()
- MULTISYNCLIBRARY_API bool [DisableCrossPCSynchronization](#) ()
- MULTISYNCLIBRARY_API bool [QueryCrossPCSynchronizationSetting](#) ()

9.50.1 Constructor & Destructor Documentation

9.50.1.1 MULTISYNCLIBRARY_API [SyncManager](#) ()9.50.1.2 MULTISYNCLIBRARY_API [~SyncManager](#) ()

9.50.2 Member Function Documentation

9.50.2.1 MULTISYNCLIBRARY_API bool [DisableCrossPCSynchronization](#) ()9.50.2.2 MULTISYNCLIBRARY_API bool [EnableCrossPCSynchronization](#) ()9.50.2.3 MULTISYNCLIBRARY_API [PGRSyncMessage GetSyncStatus](#) ()9.50.2.4 MULTISYNCLIBRARY_API double [GetTimeSinceSynced](#) ()9.50.2.5 MULTISYNCLIBRARY_API bool [IsTimingBusConnected](#) ()

9.50.2.6 MULTISYNCLIBRARY_API bool QueryCrossPCSynchronizationSetting ()

9.50.2.7 MULTISYNCLIBRARY_API PGRSyncError RescanMasterTimingBus ()

9.50.2.8 MULTISYNCLIBRARY_API PGRSyncError Start ()

9.50.2.9 MULTISYNCLIBRARY_API PGRSyncError Stop ()

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

- [MultiSyncLibraryDefs.h](#)

9.51 SystemInfo Struct Reference

Description of the system.

Public Attributes

- [OSType osType](#)
Operating system type as described by OSType.
- char [osDescription](#) [[sk_maxStringLength](#)]
Detailed description of the operating system.
- [ByteOrder byteOrder](#)
Byte order of the system.
- size_t [sysMemSize](#)
Amount of memory available on the system.
- char [cpuDescription](#) [[sk_maxStringLength](#)]
Detailed description of the CPU.
- size_t [numCpuCores](#)
Number of cores on all CPUs on the system.
- char [driverList](#) [[sk_maxStringLength](#)]
List of drivers used.
- char [libraryList](#) [[sk_maxStringLength](#)]
List of libraries used.
- char [gpuDescription](#) [[sk_maxStringLength](#)]
Detailed description of the GPU.
- size_t [screenWidth](#)
Screen resolution width in pixels.
- size_t [screenHeight](#)
Screen resolution height in pixels.
- unsigned int [reserved](#) [16]
Reserved for future use.

9.51.1 Detailed Description

Description of the system.

9.51.2 Member Data Documentation

9.51.2.1 ByteOrder byteOrder

Byte order of the system.

9.51.2.2 char cpuDescription[sk_maxStringLength]

Detailed description of the CPU.

9.51.2.3 char driverList[sk_maxStringLength]

List of drivers used.

9.51.2.4 char gpuDescription[sk_maxStringLength]

Detailed description of the GPU.

9.51.2.5 char libraryList[sk_maxStringLength]

List of libraries used.

9.51.2.6 size_t numCpuCores

Number of cores on all CPUs on the system.

9.51.2.7 char osDescription[sk_maxStringLength]

Detailed description of the operating system.

9.51.2.8 OSType osType

Operating system type as described by OSType.

9.51.2.9 unsigned int reserved[16]

Reserved for future use.

9.51.2.10 `size_t screenHeight`

Screen resolution height in pixels.

9.51.2.11 `size_t screenWidth`

Screen resolution width in pixels.

9.51.2.12 `size_t sysMemSize`

Amount of memory available on the system.

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

- [Utilities.h](#)

9.52 TIFFOption Struct Reference

Options for saving TIFF images.

Public Types

- enum `CompressionMethod` { `NONE` = 1, `PACKBITS`, `DEFLATE`, `ADOBE_DEFLATE`, `CCITTFAX3`, `CCITTFAX4`, `LZW`, `JPEG` }

Public Member Functions

- [TIFFOption](#) ()

Public Attributes

- [CompressionMethod](#) `compression`
Compression method to use for encoding TIFF images.
- unsigned int [reserved](#) [16]
Reserved for future use.

9.52.1 Detailed Description

Options for saving TIFF images.

9.52.2 Member Enumeration Documentation

9.52.2.1 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.

9.52.3 Constructor & Destructor Documentation

9.52.3.1 TIFFOption () [inline]

9.52.4 Member Data Documentation

9.52.4.1 CompressionMethod compression

Compression method to use for encoding TIFF images.

9.52.4.2 unsigned int reserved[16]

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

9.53 TimeStamp Struct Reference

Timestamp information.

Public Member Functions

- [TimeStamp \(\)](#)

Public Attributes

- long long [seconds](#)
Seconds.
- unsigned int [microSeconds](#)
Microseconds.
- unsigned int [cycleSeconds](#)
1394 cycle time seconds.
- unsigned int [cycleCount](#)
1394 cycle time count.
- unsigned int [cycleOffset](#)
1394 cycle time offset.
- unsigned int [reserved](#) [8]
Reserved for future use.

9.53.1 Detailed Description

Timestamp information.

9.53.2 Constructor & Destructor Documentation

9.53.2.1 `TimeStamp ()` [`inline`]

9.53.3 Member Data Documentation

9.53.3.1 unsigned int `cycleCount`

1394 cycle time count.

9.53.3.2 unsigned int `cycleOffset`

1394 cycle time offset.

9.53.3.3 unsigned int `cycleSeconds`

1394 cycle time seconds.

9.53.3.4 unsigned int `microSeconds`

Microseconds.

9.53.3.5 unsigned int reserved[8]

Reserved for future use.

9.53.3.6 long long seconds

Seconds.

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

- [FlyCapture2Defs.h](#)

9.54 TopologyNode Class Reference

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

Public Types

- enum [PortType](#) { [NOT_CONNECTED](#) = 1, [CONNECTED_TO_PARENT](#), [CONNECTED_TO_CHILD](#) }
Possible states of a port on a node.
- enum [NodeType](#) { [COMPUTER](#), [BUS](#), [CAMERA](#), [NODE](#) }
Type of node.

Public Member Functions

- [TopologyNode](#) ()
Default constructor.
- [TopologyNode](#) ([PGRGuid](#) guid, int deviceId, [NodeType](#) nodeType, [InterfaceType](#) interfaceType)
Constructor.
- virtual [~TopologyNode](#) ()
Default destructor.
- [TopologyNode](#) (const [TopologyNode](#) &other)
Copy constructor.
- virtual [TopologyNode](#) & operator= (const [TopologyNode](#) &other)
Assignment operator.
- virtual [PGRGuid](#) GetGuid ()
Get the [PGRGuid](#) associated with the node.
- virtual int GetDeviceId ()
Get the device ID associated with the node.
- virtual [NodeType](#) GetNodeType ()

Get the node type associated with the node.

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

Get the interface type associated with the node.

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

Get the number of child nodes.

- virtual [TopologyNode](#) [GetChild](#) (unsigned int position)

Get child node located at the specified position.

- virtual void [AddChild](#) ([TopologyNode](#) childNode)

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

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

Get the number of ports.

- virtual [PortType](#) [GetPortType](#) (unsigned int position)

Get type of port located at the specified position.

- virtual void [AddPortType](#) ([PortType](#) childPort)

Add the specified [PortType](#) as a port of the node.

- virtual bool [AssignGuidToNode](#) ([PGRGuid](#) guid, int deviceId)

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

- virtual bool [AssignGuidToNode](#) ([PGRGuid](#) guid, int deviceId, [NodeType](#) nodeType)

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

9.54.1 Detailed Description

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

9.54.2 Member Enumeration Documentation

9.54.2.1 enum [NodeType](#)

Type of node.

Enumerator:

COMPUTER

BUS

CAMERA

NODE

9.54.2.2 enum PortType

Possible states of a port on a node.

Enumerator:

NOT_CONNECTED

CONNECTED_TO_PARENT

CONNECTED_TO_CHILD

9.54.3 Constructor & Destructor Documentation

9.54.3.1 TopologyNode ()

Default constructor.

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

Constructor.

Parameters

<i>guid</i>	The PGRGuid 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.

9.54.3.3 virtual ~TopologyNode () [virtual]

Default destructor.

9.54.3.4 TopologyNode (const TopologyNode & *other*)

Copy constructor.

9.54.4 Member Function Documentation

9.54.4.1 virtual void AddChild (TopologyNode *childNode*) [virtual]

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

Parameters

<i>childNode</i>	The TopologyNode to add.
------------------	--

9.54.4.2 `virtual void AddPortType (PortType childPort) [virtual]`

Add the specified PortType as a port of the node.

Parameters

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

9.54.4.3 `virtual bool AssignGuidToNode (PGRGuid guid, int deviceld) [virtual]`

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

Parameters

<i>guid</i>	PGRGuid to be assigned.
<i>deviceld</i>	Device ID to be assigned.

Returns

Whether the data was successfully set to the node.

9.54.4.4 `virtual bool AssignGuidToNode (PGRGuid guid, int deviceld, NodeType nodeType) [virtual]`

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

Parameters

<i>guid</i>	PGRGuid 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.

9.54.4.5 `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.

9.54.4.6 `virtual int GetDeviceld () [virtual]`

Get the device ID associated with the node.

Returns

Device ID of the node.

9.54.4.7 `virtual PGRGuid GetGuid () [virtual]`

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

Returns

[PGRGuid](#) of the node.

9.54.4.8 `virtual InterfaceType GetInterfaceType () [virtual]`

Get the interface type associated with the node.

Returns

Interface type of the node.

9.54.4.9 `virtual NodeType GetNodeType () [virtual]`

Get the node type associated with the node.

Returns

Node type of the node.

9.54.4.10 `virtual unsigned int GetNumChildren () [virtual]`

Get the number of child nodes.

Returns

Number of child nodes.

9.54.4.11 virtual unsigned int GetNumPorts () [virtual]

Get the number of ports.

Returns

Number of ports.

9.54.4.12 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.

9.54.4.13 virtual TopologyNode& operator= (const TopologyNode & *other*) [virtual]

Assignment operator.

Parameters

<i>other</i>	The TopologyNode to copy from.
--------------	--

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

- [TopologyNode.h](#)

9.55 TriggerMode Struct Reference

A camera trigger.

Public Member Functions

- [TriggerMode](#) ()

Public Attributes

- bool [onOff](#)

Flag controlling on/off.

- unsigned int [polarity](#)

Polarity value.

- unsigned int [source](#)

Source value.

- unsigned int [mode](#)

Mode value.

- unsigned int [parameter](#)

Parameter value.

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

Reserved for future use.

9.55.1 Detailed Description

A camera trigger.

9.55.2 Constructor & Destructor Documentation

9.55.2.1 `TriggerMode ()` `[inline]`

9.55.3 Member Data Documentation

9.55.3.1 unsigned int `mode`

Mode value.

9.55.3.2 bool `onOff`

Flag controlling on/off.

9.55.3.3 unsigned int `parameter`

Parameter value.

9.55.3.4 unsigned int `polarity`

Polarity value.

9.55.3.5 unsigned int `reserved`[8]

Reserved for future use.

9.55.3.6 unsigned int source

Source value.

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

- [FlyCapture2Defs.h](#)

9.56 TriggerModelInfo Struct Reference

Information about a camera trigger property.

Public Member Functions

- [TriggerModelInfo](#) ()

Public Attributes

- bool [present](#)
Presence of trigger mode.
- bool [readOutSupported](#)
Flag indicating if trigger value can be read out.
- bool [onOffSupported](#)
Flag indicating if on/off is supported.
- bool [polaritySupported](#)
Flag indicating if polarity is supported.
- bool [valueReadable](#)
Flag indicating if the value is readable.
- unsigned int [sourceMask](#)
Source mask.
- bool [softwareTriggerSupported](#)
Flag indicating if software trigger is supported.
- unsigned int [modeMask](#)
Mode mask.
- unsigned int [reserved](#) [8]
Reserved for future use.

9.56.1 Detailed Description

Information about a camera trigger property.

9.56.2 Constructor & Destructor Documentation

9.56.2.1 TriggerModelInfo () [inline]

9.56.3 Member Data Documentation

9.56.3.1 unsigned int modeMask

Mode mask.

9.56.3.2 bool onOffSupported

Flag indicating if on/off is supported.

9.56.3.3 bool polaritySupported

Flag indicating if polarity is supported.

9.56.3.4 bool present

Presence of trigger mode.

9.56.3.5 bool readOutSupported

Flag indicating if trigger value can be read out.

9.56.3.6 unsigned int reserved[8]

Reserved for future use.

9.56.3.7 bool softwareTriggerSupported

Flag indicating if software trigger is supported.

9.56.3.8 unsigned int sourceMask

Source mask.

9.56.3.9 bool valueReadable

Flag indicating if the value is readable.

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

- [FlyCapture2Defs.h](#)

9.57 Utilities Class Reference

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

Static Public Member Functions

- static [Error CheckDriver](#) (const [PGRGuid](#) *guid)
Check for driver compatibility for the given camera guid.
- static [Error GetDriverDeviceName](#) (const [PGRGuid](#) *guid, std::string &device-Name)
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, [AsyncCommand-Callback](#) pCallback, void *pUserData)
Execute a command in the terminal.

9.57.1 Detailed Description

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

It can also be used to launch browsers, CHM viewers or terminal commands.

9.57.2 Member Function Documentation

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

Check for driver compatibility for the given camera guid.

Parameters

<i>guid</i>	Pointer to the guid of the device to check.
-------------	---

Returns

PGR_NO_ERROR if the library is compatible with the currently loaded driver, otherwise an error indicating the type of failure.

9.57.2.2 `static Error GetDriverDeviceName (const PGRGuid * guid, std::string & deviceName) [static]`

Get the driver's name for a device.

Parameters

<i>guid</i>	Pointer to the guid of the device to check.
<i>deviceName</i>	The device name will be returned in this string

Returns

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

9.57.2.3 `static Error GetLibraryVersion (FC2Version * pVersion) [static]`

Get library version.

Parameters

<i>pVersion</i>	Structure to receive the library version.
-----------------	---

Returns

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

9.57.2.4 `static Error GetSystemInfo (SystemInfo * pSystemInfo) [static]`

Get system information.

Parameters

<i>pSystemInfo</i>	Structure to receive system information.
--------------------	--

Returns

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

9.57.2.5 `static Error LaunchBrowser (const char * pAddress) [static]`

Launch a URL in the system default browser.

Parameters

<i>pAddress</i>	URL to open in browser.
-----------------	-------------------------

Returns

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

9.57.2.6 static Error LaunchCommand (const char * *pCommand*) [static]

Execute a command in the terminal.

This is a blocking call that will return when the command completes.

Parameters

<i>pCommand</i>	Command to execute.
-----------------	---------------------

See also

[LaunchCommandAsync\(\)](#)

Returns

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

**9.57.2.7 static Error LaunchCommandAsync (const char * *pCommand*,
AsyncCommandCallback *pCallback*, void * *pUserData*) [static]**

Execute a command in the terminal.

This is a non-blocking call that will return immediately. The return value of the command can be retrieved in the callback.

Parameters

<i>pCommand</i>	Command to execute.
<i>pCallback</i>	Callback to fire when command is complete.
<i>pUserData</i>	Data pointer to pass to callback.

See also

[LaunchCommand\(\)](#)

Returns

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

9.57.2.8 `static Error LaunchHelp (const char * pFileName)` `[static]`

Open a CHM file in the system default CHM viewer.

Parameters

<i>pFileName</i>	Filename of CHM file to open.
------------------	-------------------------------

Returns

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

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

- [Utilities.h](#)

Chapter 10

File Documentation

10.1 BusManager.h File Reference

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.

10.2 Camera.h File Reference

Classes

- class [Camera](#)

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

Namespaces

- namespace [FlyCapture2](#)

10.3 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.

10.4 Error.h File Reference

Classes

- class [Error](#)

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

Namespaces

- namespace [FlyCapture2](#)

10.5 FlyCapture2.h File Reference

10.6 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 [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 `PCleBusSpeed` { `PCIE_BUSSPEED_2_5`, `PCIE_BUSSPEED_5_0`, `PCIE_BUSSPEED_UNKNOWN` = -1, `PCIE_BUSSPEED_FORCE_32BITS` = `FULL_32BIT_VALUE` }
- enum `DriverType` { `DRIVER_1394_CAM`, `DRIVER_1394_PRO`, `DRIVER_1394_JUUJ`, `DRIVER_1394_VIDEO1394`, `DRIVER_1394_RAW1394`, `DRIVER_USB_NONE`, `DRIVER_USB_CAM`, `DRIVER_USB3_PRO`, `DRIVER_GIGE_NONE`, `DRIVER_GIGE_FILTER`, `DRIVER_GIGE_PRO`, `DRIVER_GIGE_LWF`, `DRIVER_UNKNOWN` = -1, `DRIVER_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Types of low level drivers that flycapture uses.

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

Color processing algorithms.

- enum `BayerTileFormat` { `NONE`, `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.

10.6.1 Define Documentation

10.6.1.1 `#define FULL_32BIT_VALUE 0x7FFFFFFF`

10.6.1.2 `#define NULL 0`

10.7 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](#)

10.8 FlyCapture2Platform.h File Reference

Defines

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

10.8.1 Define Documentation

10.8.1.1 #define [FLYCAPTURE2_API](#) __attribute__((visibility ("default")))

10.8.1.2 #define [FLYCAPTURE2_LOCAL](#) __attribute__((visibility ("hidden")))

10.9 FlyCapture2Video.h File Reference

Classes

- class [FlyCapture2Video](#)

The [FlyCapture2Video](#) class provides the functionality for the user to record images to an AVI file.

Namespaces

- namespace [FlyCapture2](#)

10.10 FlyCapture2VideoDefs.h File Reference

Classes

- struct [MJPGOption](#)
Options for saving MJPG files.
- struct [H264Option](#)

Options for saving H264 files.

- struct [AVIOption](#)

Options for saving AVI files.

Namespaces

- namespace [FlyCapture2](#)

10.11 FlyCapture3ApiGuiWrapper.h File Reference

Classes

- class [FlyCapture3ApiGuiWrapper](#)

Namespaces

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

Defines

- #define [WRAPPER_API](#) __declspec(dllimport)

10.11.1 Define Documentation

10.11.1.1 #define WRAPPER_API __declspec(dllimport)

10.12 GCCamera.h File Reference

Classes

- class [GCCamera](#)

Namespaces

- namespace [FlyCapture2](#)

10.13 GigECamera.h File Reference

Classes

- class [GigECamera](#)

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

Namespaces

- namespace [FlyCapture2](#)

10.14 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](#)

10.15 ImageStatistics.h File Reference

Classes

- class [ImageStatistics](#)

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

Namespaces

- namespace [FlyCapture2](#)

10.16 Internal.h File Reference

Classes

- class [Internal](#)

Namespaces

- namespace [FlyCapture2](#)

10.17 Licensing.dox File Reference

10.18 MultiSyncLibrary.h File Reference

10.19 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](#) }

10.20 MultiSyncLibraryPlatform.h File Reference

Defines

- #define [MULTISYNCLIBRARY_API](#) __attribute__((visibility ("default")))
- #define [MULTISYNCLIBRARY_LOCAL](#) __attribute__((visibility ("hidden")))

10.20.1 Define Documentation

10.20.1.1 #define MULTISYNCLIBRARY_API __attribute__((visibility ("default")))

10.20.1.2 `#define MULTISYNCLIBRARY_LOCAL __attribute__((visibility("hidden")))`

10.21 NodeMap.h File Reference

Classes

- class [NodeMap](#)

Namespaces

- namespace [FlyCapture2](#)

10.22 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](#)

10.23 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

- ~BusManager
 - FlyCapture2::BusManager, [52](#)
- ~Camera
 - FlyCapture2::Camera, [67](#)
- ~CameraBase
 - FlyCapture2::CameraBase, [96](#)
- ~CameraControlDlg
 - FlyCapture2::CameraControlDlg, [119](#)
- ~CameraSelectionDlg
 - FlyCapture2::CameraSelectionDlg, [127](#)
- ~Error
 - FlyCapture2::Error, [136](#)
- ~FlyCapture2Video
 - FlyCapture2::FlyCapture2Video, [146](#)
- ~FlyCapture3ApiGuiWrapper
 - FlyCap3CameraControl::FlyCapture3-ApiGuiWrapper, [149](#)
- ~GCCamera
 - FlyCapture2::GCCamera, [159](#)
- ~GigECamera
 - FlyCapture2::GigECamera, [186](#)
- ~Image
 - FlyCapture2::Image, [228](#)
- ~ImageStatistics
 - FlyCapture2::ImageStatistics, [243](#)
- ~NodeMap
 - FlyCapture2::NodeMap, [256](#)
- ~SyncManager
 - MultiSyncLibrary::SyncManager, [269](#)
- ~TopologyNode
 - FlyCapture2::TopologyNode, [277](#)
- ADOBE_DEFLATE
 - FlyCapture2::TIFFOption, [273](#)
- 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, [243](#)
- BMP
 - Enumerations, [24](#)
- BRIGHTNESS
 - Enumerations, [26](#)
- BT_FORCE_32BITS
 - Enumerations, [19](#)
- BUFFER_FRAMES
 - Enumerations, [23](#)
- BUS
 - FlyCapture2::TopologyNode, [276](#)
- 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, [46](#)
- BYTE_ORDER_FORCE_32BITS
 - FlyCapture2, [46](#)
- BYTE_ORDER_LITTLE_ENDIAN
 - FlyCapture2, [46](#)
- CALLBACK_TYPE_FORCE_32BITS
 - Enumerations, [19](#)
- CAMERA
 - FlyCapture2::TopologyNode, [276](#)
- CCITTFAX3
 - FlyCapture2::TIFFOption, [273](#)
- CCITTFAX4
 - FlyCapture2::TIFFOption, [273](#)
- COLOR_PROCESSING_ALGORITHM_FORCE_32BITS
 - Enumerations, [20](#)
- COMPUTER
 - FlyCapture2::TopologyNode, [276](#)
- CONNECTED_TO_CHILD
 - FlyCapture2::TopologyNode, [277](#)
- CONNECTED_TO_PARENT
 - FlyCapture2::TopologyNode, [277](#)
- DEFAULT
 - Enumerations, [20](#)
- DEFLATE
 - FlyCapture2::TIFFOption, [273](#)
- 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, [46](#)
 - BYTE_ORDER_FORCE_32BITS, [46](#)
 - BYTE_ORDER_LITTLE_ENDIAN, [46](#)
 - LINUX_X64, [46](#)
 - LINUX_X86, [46](#)
 - MAC, [46](#)
 - OSTYPE_FORCE_32BITS, [46](#)
 - UNKNOWN_OS, [46](#)
 - WINDOWS_X64, [46](#)
 - WINDOWS_X86, [46](#)
- FlyCapture2::ImageStatistics
 - BLUE, [243](#)
 - GREEN, [243](#)
 - GREY, [243](#)
 - HUE, [243](#)
 - LIGHTNESS, [243](#)
 - NUM_STATISTICS_CHANNELS, [243](#)
 - RED, [243](#)
 - SATURATION, [243](#)
- FlyCapture2::TIFFOption
 - ADOBE_DEFLATE, [273](#)
 - CCITTFAX3, [273](#)
 - CCITTFAX4, [273](#)
 - DEFLATE, [273](#)
 - JPEG, [273](#)
 - LZW, [273](#)
 - NONE, [273](#)
 - PACKBITS, [273](#)
- FlyCapture2::TopologyNode
 - BUS, [276](#)
 - CAMERA, [276](#)
 - COMPUTER, [276](#)
 - CONNECTED_TO_CHILD, [277](#)
 - CONNECTED_TO_PARENT, [277](#)
 - NODE, [276](#)
 - NOT_CONNECTED, [277](#)
- 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, [243](#)
- GREY
 - FlyCapture2::ImageStatistics, [243](#)
- 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, [243](#)
- 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, [273](#)
- JPEG2000
 - Enumerations, [24](#)
- LIGHTNESS
 - FlyCapture2::ImageStatistics, [243](#)
- LINUX_X64
 - FlyCapture2, [46](#)
- LINUX_X86
 - FlyCapture2, [46](#)

- LZW
 - FlyCapture2::TIFFOption, [273](#)
- MAC
 - FlyCapture2, [46](#)
- 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, [47](#)
 - PGRSyncError_ALREADY_STOPPED, [47](#)
 - PGRSyncError_CAMERA_NOT_FOUND, [47](#)
 - PGRSyncError_FAILED, [47](#)
 - PGRSyncError_OK, [47](#)
 - PGRSyncError_UNKNOWN_ERROR, [47](#)
 - PGRSyncMessage_BUS_RESET, [47](#)
 - PGRSyncMessage_DEVICE_ERROR, [47](#)
 - PGRSyncMessage_NOMASTER, [47](#)
 - PGRSyncMessage_NOT_ENOUGH_DEVICES, [47](#)
 - PGRSyncMessage_NOT_INITIALIZED, [47](#)
 - PGRSyncMessage_OK, [47](#)
 - PGRSyncMessage_STARTED, [47](#)
 - PGRSyncMessage_STOPPED, [47](#)
 - PGRSyncMessage_SYNCING, [47](#)
 - PGRSyncMessage_THREAD_ERROR, [47](#)
 - PGRSyncMessage_UNKNOWN_ERROR, [47](#)
- NEAREST_NEIGHBOR
 - Enumerations, [20](#)

- NODE
 - FlyCapture2::TopologyNode, [276](#)
- NONE
 - Enumerations, [19](#)
 - FlyCapture2::TIFFOption, [273](#)
- NOT_CONNECTED
 - FlyCapture2::TopologyNode, [277](#)
- NO_COLOR_PROCESSING
 - Enumerations, [20](#)
- NUM_FRAMERATES
 - Enumerations, [23](#)
- NUM_MODES
 - Enumerations, [25](#)
- NUM_PIXEL_FORMATS
 - Enumerations, [26](#)
- NUM_STATISTICS_CHANNELS
 - FlyCapture2::ImageStatistics, [243](#)
- NUM_VIDEOMODES
 - Enumerations, [28](#)
- OSTYPE_FORCE_32BITS
 - FlyCapture2, [46](#)
- PACKBITS
 - FlyCapture2::TIFFOption, [273](#)
- 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, [47](#)
- PGRSyncError_ALREADY_STOPPED
 - MultiSyncLibrary, [47](#)
- PGRSyncError_CAMERA_NOT_FOUND
 - MultiSyncLibrary, [47](#)
- PGRSyncError_FAILED
 - MultiSyncLibrary, [47](#)
- PGRSyncError_OK
 - MultiSyncLibrary, [47](#)
- PGRSyncError_UNKNOWN_ERROR
 - MultiSyncLibrary, [47](#)
- PGRSyncMessage_BUS_RESET
 - MultiSyncLibrary, [47](#)
- PGRSyncMessage_DEVICE_ERROR
 - MultiSyncLibrary, [47](#)
- PGRSyncMessage_NOMASTER
 - MultiSyncLibrary, [47](#)
- PGRSyncMessage_NOT_ENOUGH_DEVICES
 - MultiSyncLibrary, [47](#)
- PGRSyncMessage_NOT_INITIALIZED
 - MultiSyncLibrary, [47](#)
- PGRSyncMessage_OK
 - MultiSyncLibrary, [47](#)
- PGRSyncMessage_STARTED
 - MultiSyncLibrary, [47](#)
- PGRSyncMessage_STOPPED
 - MultiSyncLibrary, [47](#)
- PGRSyncMessage_SYNCING
 - MultiSyncLibrary, [47](#)
- PGRSyncMessage_THREAD_ERROR
 - MultiSyncLibrary, [47](#)
- PGRSyncMessage_UNKNOWN_ERROR
 - MultiSyncLibrary, [47](#)
- 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_RGBA
 - 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](#)
- RAW
 - Enumerations, [24](#)
- RED
 - FlyCapture2::ImageStatistics, [243](#)
- REMOVAL
 - Enumerations, [19](#)
- RGGB
 - Enumerations, [19](#)
- RIGOROUS
 - Enumerations, [20](#)
- SATURATION
 - Enumerations, [26](#)
 - FlyCapture2::ImageStatistics, [243](#)
- 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, [46](#)
- 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, [46](#)
- WINDOWS_X86
 - FlyCapture2, [46](#)
- ZOOM
 - Enumerations, [27](#)
- AVIOption, [49](#)
 - FlyCapture2::AVIOption, [49](#)
- AddChild
 - FlyCapture2::TopologyNode, [277](#)
- AddPortType
 - FlyCapture2::TopologyNode, [278](#)
- Append
 - FlyCapture2::FlyCapture2Video, [146](#)
- AssignGuidToNode
 - FlyCapture2::TopologyNode, [278](#)
- AsyncCommandCallback
 - FlyCapture2, [45](#)
- BMPOption, [50](#)
 - FlyCapture2::BMPOption, [50](#)
- BandwidthAllocation
 - Enumerations, [18](#)
- BayerTileFormat
 - Enumerations, [18](#)
- BusCallbackType
 - Enumerations, [19](#)
- BusEventCallback
 - FlyCapture2, [45](#)
- BusManager, [51](#)
 - FlyCapture2::BusManager, [52](#)
- BusManager.h, [289](#)
- BusSpeed
 - Enumerations, [19](#)
- ByteOrder
 - FlyCapture2, [46](#)
- CalculateStatistics
 - FlyCapture2::Image, [228](#)
- CallbackHandle
 - FlyCapture2, [45](#)
- Camera, [62](#)
 - FlyCapture2::Camera, [67](#)
- Camera.h, [289](#)
- CameraBase, [92](#)
 - FlyCapture2::CameraBase, [96](#)
- CameraBase.h, [290](#)
- CameraControlDlg, [119](#)
 - FlyCapture2::CameraControlDlg, [119](#)
- CameraEventCallback
 - Image saving structures., [36](#)
- CameraInfo, [121](#)
 - FlyCapture2::CameraInfo, [123](#)
- CameraSelectionDlg, [126](#)
 - FlyCapture2::CameraSelectionDlg, [127](#)
- CameraStats, [128](#)
 - FlyCapture2::CameraStats, [129](#)
- CheckDriver
 - FlyCapture2::Utilities, [284](#)
- Close
 - FlyCapture2::FlyCapture2Video, [146](#)
- CollectSupportInformation
 - FlyCapture2::Error, [137](#)
- ColorProcessingAlgorithm
 - Enumerations, [20](#)
- CompressionMethod
 - FlyCapture2::TIFFOption, [273](#)
- ConfigROM, [130](#)
 - FlyCapture2::ConfigROM, [131](#)
- Connect
 - FlyCapture2::Camera, [67](#)
 - FlyCapture2::CameraBase, [97](#)
 - FlyCapture2::CameraControlDlg, [120](#)
 - FlyCapture2::GCCamera, [159](#)
 - FlyCapture2::GigECamera, [186](#)
- ConnectGUILibrary
 - FlyCap3CameraControl::FlyCapture3-ApiGuiWrapper, [149](#)
- Convert
 - FlyCapture2::Image, [229](#)
- DeepCopy
 - FlyCapture2::Image, [229](#)
- DeregisterAllEvents

- FlyCapture2::Camera, [67](#)
- FlyCapture2::CameraBase, [97](#)
- FlyCapture2::GigECamera, [186](#)
- DeregisterEvent
 - FlyCapture2::Camera, [67](#)
 - FlyCapture2::CameraBase, [97](#)
 - FlyCapture2::GigECamera, [187](#)
- DetermineBitsPerPixel
 - FlyCapture2::Image, [230](#)
- DisableAll
 - FlyCapture2::ImageStatistics, [244](#)
- DisableCrossPCSSynchronization
 - MultiSyncLibrary::SyncManager, [269](#)
- Disconnect
 - FlyCapture2::Camera, [67](#)
 - FlyCapture2::CameraBase, [97](#)
 - FlyCapture2::CameraControlDlg, [120](#)
 - FlyCapture2::GCCamera, [159](#)
 - FlyCapture2::GigECamera, [187](#)
- DisconnectGUILibrary
 - FlyCap3CameraControl::FlyCapture3-ApiGuiWrapper, [149](#)
- DiscoverGigECameras
 - FlyCapture2::BusManager, [53](#)
- DiscoverGigEPacketSize
 - FlyCapture2::GigECamera, [187](#)
- DriverType
 - Enumerations, [20](#)
- EmbeddedImageInfo, [133](#)
- EmbeddedImageInfoProperty, [134](#)
 - FlyCapture2::EmbeddedImageInfo-Property, [135](#)
- EnableAll
 - FlyCapture2::ImageStatistics, [244](#)
- EnableCrossPCSSynchronization
 - MultiSyncLibrary::SyncManager, [269](#)
- EnableGreyOnly
 - FlyCapture2::ImageStatistics, [244](#)
- EnableHSLOnly
 - FlyCapture2::ImageStatistics, [244](#)
- EnableLUT
 - FlyCapture2::Camera, [68](#)
 - FlyCapture2::CameraBase, [98](#)
 - FlyCapture2::GCCamera, [160](#)
 - FlyCapture2::GigECamera, [187](#)
- EnableRGBOnly
 - FlyCapture2::ImageStatistics, [244](#)
- 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, [135](#)
 - FlyCapture2::Error, [136](#)
- Error.h, [290](#)
- ErrorType
 - Enumerations, [21](#)
- EventCallbackData, [139](#)
- EventCallbackFcn
 - FlyCapture2::EventOptions, [141](#)
- EventData
 - FlyCapture2::EventCallbackData, [139](#)
- EventDataSize
 - FlyCapture2::EventCallbackData, [139](#)
- EventID
 - FlyCapture2::EventCallbackData, [139](#)
- EventName
 - FlyCapture2::EventCallbackData, [140](#)
 - FlyCapture2::EventOptions, [141](#)
- EventOptions, [140](#)
- EventTimestamp
 - FlyCapture2::EventCallbackData, [140](#)
- EventUserData
 - FlyCapture2::EventCallbackData, [140](#)
 - FlyCapture2::EventOptions, [141](#)
- EventUserDataSize
 - FlyCapture2::EventCallbackData, [140](#)
 - FlyCapture2::EventOptions, [141](#)
- FC2Config, [141](#)

- FlyCapture2::FC2Config, 142
- FC2Version, 144
- FLYCAPTURE2_API
 - FlyCapture2Platform.h, 296
- FLYCAPTURE2_LOCAL
 - FlyCapture2Platform.h, 296
- FULL_32BIT_VALUE
 - FlyCapture2Defs.h, 295
- FireBusReset
 - FlyCapture2::BusManager, 53
- FireSoftwareTrigger
 - FlyCapture2::Camera, 68
 - FlyCapture2::CameraBase, 98
 - FlyCapture2::GCCamera, 160
 - FlyCapture2::GigECamera, 188
- FlyCap3CameraControl, 39
- FlyCap3CameraControl::FlyCapture3Api-
 - GuiWrapper
 - ~FlyCapture3ApiGuiWrapper, 149
 - ConnectGUILibrary, 149
 - DisconnectGUILibrary, 149
 - FlyCapture3ApiGuiWrapper, 149
 - GetControlNameList, 149
 - GetDialogNameList, 149
 - GetNumDialogs, 149
 - GetNumOfControls, 149
 - ShowCameraSelectionDialog, 149
 - ShowDialogByIndex, 149
 - ShowDialogByName, 149
 - ShowPropertyGridDialog, 149
- FlyCapture2, 39
 - AsyncCommandCallback, 45
 - BusEventCallback, 45
 - ByteOrder, 46
 - CallbackHandle, 45
 - ImageEventCallback, 45
 - OSType, 46
- FlyCapture2.h, 290
- FlyCapture2::AVIOption
 - AVIOption, 49
 - frameRate, 49
 - reserved, 49
- FlyCapture2::BMPOption
 - BMPOption, 50
 - indexedColor_8bit, 50
 - reserved, 50
- FlyCapture2::BusManager
 - ~BusManager, 52
 - BusManager, 52
 - DiscoverGigECameras, 53
 - FireBusReset, 53
 - ForceAllIPAddressesAutomatically, 53, 54
 - ForceIPAddressToCamera, 54
 - GetCameraFromIPAddress, 55
 - GetCameraFromIndex, 54
 - GetCameraFromSerialNumber, 55
 - GetCameraSerialNumberFromIndex, 56
 - GetDeviceFromIndex, 56
 - GetInterfaceTypeFromGuid, 56
 - GetNumOfCameras, 57
 - GetNumOfDevices, 57
 - GetTopology, 57
 - GetUsbLinkInfo, 58
 - GetUsbPortStatus, 58
 - IsCameraControlable, 60
 - ReadPhyRegister, 60
 - RegisterCallback, 60
 - RescanBus, 61
 - UnregisterCallback, 61
 - WritePhyRegister, 61
- FlyCapture2::Camera
 - ~Camera, 67
 - Camera, 67
 - Connect, 67
 - DeregisterAllEvents, 67
 - DeregisterEvent, 67
 - Disconnect, 67
 - EnableLUT, 68
 - FireSoftwareTrigger, 68
 - GetActiveLUTBank, 68
 - GetCameraInfo, 69
 - GetConfiguration, 69
 - GetCycleTime, 69
 - GetEmbeddedImageInfo, 70
 - GetFormat7Configuration, 70
 - GetFormat7Info, 71
 - GetGPIOPinDirection, 71
 - GetLUTBankInfo, 72
 - GetLUTChannel, 72
 - GetLUTInfo, 72
 - GetMemoryChannel, 73
 - GetMemoryChannelInfo, 73
 - GetProperty, 74
 - GetPropertyInfo, 74
 - GetRegisterString, 75
 - GetStats, 75
 - GetStrobe, 75
 - GetStrobeInfo, 76

- GetTriggerDelay, [76](#)
- GetTriggerDelayInfo, [76](#)
- GetTriggerMode, [77](#)
- GetTriggerModelInfo, [77](#)
- GetVideoModeAndFrameRate, [78](#)
- GetVideoModeAndFrameRateInfo, [78](#)
- IsConnected, [79](#)
- ReadRegister, [79](#)
- ReadRegisterBlock, [80](#)
- RegisterAllEvents, [80](#)
- RegisterEvent, [80](#)
- ResetStats, [80](#)
- RestoreFromMemoryChannel, [80](#)
- RetrieveBuffer, [81](#)
- SaveToMemoryChannel, [81](#)
- SetActiveLUTBank, [82](#)
- SetCallback, [82](#)
- SetConfiguration, [83](#)
- SetEmbeddedImageInfo, [83](#)
- SetFormat7Configuration, [83](#), [84](#)
- SetGPIOPinDirection, [84](#)
- SetLUTChannel, [85](#)
- SetProperty, [85](#)
- SetStrobe, [86](#)
- SetTriggerDelay, [86](#)
- SetTriggerMode, [87](#)
- SetUserBuffers, [87](#)
- SetVideoModeAndFrameRate, [88](#)
- StartCapture, [88](#)
- StartSyncCapture, [89](#)
- StopCapture, [89](#)
- ValidateFormat7Settings, [90](#)
- WaitForBufferEvent, [90](#)
- WriteRegister, [91](#)
- WriteRegisterBlock, [91](#)
- FlyCapture2::CameraBase
 - ~CameraBase, [96](#)
 - CameraBase, [96](#)
 - Connect, [97](#)
 - DeregisterAllEvents, [97](#)
 - DeregisterEvent, [97](#)
 - Disconnect, [97](#)
 - EnableLUT, [98](#)
 - FireSoftwareTrigger, [98](#)
 - GetActiveLUTBank, [98](#)
 - GetCameraInfo, [99](#)
 - GetConfiguration, [99](#)
 - GetCycleTime, [99](#)
 - GetEmbeddedImageInfo, [100](#)
 - GetGPIOPinDirection, [100](#)
 - GetLUTBankInfo, [101](#)
 - GetLUTChannel, [101](#)
 - GetLUTInfo, [101](#)
 - GetMemoryChannel, [102](#)
 - GetMemoryChannelInfo, [102](#)
 - GetProperty, [103](#)
 - GetPropertyInfo, [103](#)
 - GetRegisterString, [104](#)
 - GetStats, [104](#)
 - GetStrobe, [104](#)
 - GetStrobeInfo, [105](#)
 - GetTriggerDelay, [105](#)
 - GetTriggerDelayInfo, [106](#)
 - GetTriggerMode, [106](#)
 - GetTriggerModelInfo, [107](#)
 - IsConnected, [107](#)
 - ReadRegister, [107](#)
 - ReadRegisterBlock, [108](#)
 - RegisterAllEvents, [108](#)
 - RegisterEvent, [108](#)
 - ResetStats, [108](#)
 - RestoreFromMemoryChannel, [108](#)
 - RetrieveBuffer, [109](#)
 - SaveToMemoryChannel, [109](#)
 - SetActiveLUTBank, [110](#)
 - SetCallback, [110](#)
 - SetConfiguration, [111](#)
 - SetEmbeddedImageInfo, [111](#)
 - SetGPIOPinDirection, [111](#)
 - SetLUTChannel, [112](#)
 - SetProperty, [112](#)
 - SetStrobe, [113](#)
 - SetTriggerDelay, [113](#)
 - SetTriggerMode, [114](#)
 - SetUserBuffers, [114](#)
 - StartCapture, [115](#)
 - StartSyncCapture, [116](#)
 - StopCapture, [117](#)
 - WaitForBufferEvent, [117](#)
 - WriteRegister, [117](#)
 - WriteRegisterBlock, [118](#)
 - m_pCameraData, [118](#)
- FlyCapture2::CameraControlDlg
 - ~CameraControlDlg, [119](#)
 - CameraControlDlg, [119](#)
 - Connect, [120](#)
 - Disconnect, [120](#)
 - Hide, [120](#)
 - IsVisible, [120](#)

- SetTitle, [120](#)
- Show, [120](#)
- ShowModal, [121](#)
- FlyCapture2::CameraInfo
 - CameraInfo, [123](#)
 - applicationIPAddress, [123](#)
 - applicationPort, [123](#)
 - bayerTileFormat, [123](#)
 - busNumber, [123](#)
 - ccpStatus, [124](#)
 - configROM, [124](#)
 - defaultGateway, [124](#)
 - driverName, [124](#)
 - driverType, [124](#)
 - firmwareBuildTime, [124](#)
 - firmwareVersion, [124](#)
 - gigEMajorVersion, [124](#)
 - gigEMinorVersion, [124](#)
 - iidcVer, [124](#)
 - interfaceType, [125](#)
 - ipAddress, [125](#)
 - isColorCamera, [125](#)
 - macAddress, [125](#)
 - maximumBusSpeed, [125](#)
 - modelName, [125](#)
 - nodeNumber, [125](#)
 - pcieBusSpeed, [125](#)
 - reserved, [125](#)
 - sensorInfo, [125](#)
 - sensorResolution, [126](#)
 - serialNumber, [126](#)
 - subnetMask, [126](#)
 - userDefinedName, [126](#)
 - vendorName, [126](#)
 - xmlURL1, [126](#)
 - xmlURL2, [126](#)
- FlyCapture2::CameraSelectionDlg
 - ~CameraSelectionDlg, [127](#)
 - CameraSelectionDlg, [127](#)
 - SetTitle, [127](#)
 - ShowModal, [127](#)
- FlyCapture2::CameraStats
 - CameraStats, [129](#)
 - cameraCurrents, [129](#)
 - cameraPowerUp, [129](#)
 - cameraVoltages, [129](#)
 - imageCorrupt, [129](#)
 - imageDriverDropped, [129](#)
 - imageDropped, [129](#)
 - imageXmitFailed, [129](#)
 - numCurrents, [129](#)
 - numResendPacketsReceived, [129](#)
 - numResendPacketsRequested, [130](#)
 - numVoltages, [130](#)
 - portErrors, [130](#)
 - regReadFailed, [130](#)
 - regWriteFailed, [130](#)
 - reserved, [130](#)
 - temperature, [130](#)
 - timeSinceBusReset, [130](#)
 - timeSinceInitialization, [130](#)
 - timeStamp, [130](#)
- FlyCapture2::ConfigROM
 - ConfigROM, [131](#)
 - chipIdHi, [131](#)
 - chipIdLo, [131](#)
 - nodeVendorId, [131](#)
 - pszKeyword, [132](#)
 - reserved, [132](#)
 - unitSWVer, [132](#)
 - unitSpecId, [132](#)
 - unitSubSWVer, [132](#)
 - vendorUniqueInfo_0, [132](#)
 - vendorUniqueInfo_1, [132](#)
 - vendorUniqueInfo_2, [132](#)
 - vendorUniqueInfo_3, [132](#)
- FlyCapture2::EmbeddedImageInfo
 - GPIOPinState, [134](#)
 - ROIPosition, [134](#)
 - brightness, [134](#)
 - exposure, [134](#)
 - frameCounter, [134](#)
 - gain, [134](#)
 - shutter, [134](#)
 - strobePattern, [134](#)
 - timestamp, [134](#)
 - whiteBalance, [134](#)
- FlyCapture2::EmbeddedImageInfo-Property
 - EmbeddedImageInfoProperty, [135](#)
 - available, [135](#)
 - onOff, [135](#)
- FlyCapture2::Error
 - ~Error, [136](#)
 - CollectSupportInformation, [137](#)
 - Error, [136](#)
 - GetBuildDate, [137](#)
 - GetCause, [137](#)
 - GetDescription, [137](#)
 - GetFilename, [137](#)

- GetLine, [137](#)
- GetType, [138](#)
- InternalError, [139](#)
- PrintErrorTrace, [138](#)
- operator=, [138](#)
- operator==, [138](#)
- FlyCapture2::EventCallbackData
 - EventData, [139](#)
 - EventDataSize, [139](#)
 - EventID, [139](#)
 - EventName, [140](#)
 - EventTimestamp, [140](#)
 - EventUserData, [140](#)
 - EventUserDataSize, [140](#)
- FlyCapture2::EventOptions
 - EventCallbackFcn, [141](#)
 - EventName, [141](#)
 - EventUserData, [141](#)
 - EventUserDataSize, [141](#)
- FlyCapture2::FC2Config
 - FC2Config, [142](#)
 - asyncBusSpeed, [142](#)
 - bandwidthAllocation, [142](#)
 - grabMode, [142](#)
 - grabTimeout, [143](#)
 - highPerformanceRetrieveBuffer, [143](#)
 - isochBusSpeed, [143](#)
 - minNumImageNotifications, [143](#)
 - numBuffers, [143](#)
 - numImageNotifications, [143](#)
 - registerTimeout, [144](#)
 - registerTimeoutRetries, [144](#)
 - reserved, [144](#)
- FlyCapture2::FC2Version
 - build, [145](#)
 - major, [145](#)
 - minor, [145](#)
 - type, [145](#)
- FlyCapture2::FlyCapture2Video
 - ~FlyCapture2Video, [146](#)
 - Append, [146](#)
 - Close, [146](#)
 - FlyCapture2Video, [146](#)
 - Open, [147](#)
 - SetMaximumFileSize, [148](#)
- FlyCapture2::Format7ImageSettings
 - Format7ImageSettings, [150](#)
 - height, [150](#)
 - mode, [150](#)
 - offsetX, [150](#)
 - offsetY, [150](#)
 - pixelFormat, [151](#)
 - reserved, [151](#)
 - width, [151](#)
- FlyCapture2::Format7Info
 - Format7Info, [152](#)
 - imageHStepSize, [152](#)
 - imageVStepSize, [152](#)
 - maxHeight, [152](#)
 - maxPacketSize, [152](#)
 - maxWidth, [152](#)
 - minPacketSize, [153](#)
 - mode, [153](#)
 - offsetHStepSize, [153](#)
 - offsetVStepSize, [153](#)
 - packetSize, [153](#)
 - percentage, [153](#)
 - pixelFormatBitField, [153](#)
 - reserved, [153](#)
 - vendorPixelFormatBitField, [153](#)
- FlyCapture2::Format7PacketInfo
 - Format7PacketInfo, [154](#)
 - maxBytesPerPacket, [154](#)
 - recommendedBytesPerPacket, [154](#)
 - reserved, [154](#)
 - unitBytesPerPacket, [154](#)
- FlyCapture2::GCCamera
 - ~GCCamera, [159](#)
 - Connect, [159](#)
 - Disconnect, [159](#)
 - EnableLUT, [160](#)
 - FireSoftwareTrigger, [160](#)
 - GCCamera, [159](#)
 - GCCamera::GetXML, [160](#)
 - GetActiveLUTBank, [160](#)
 - GetCameraInfo, [161](#)
 - GetConfiguration, [161](#)
 - GetCycleTime, [161](#)
 - GetEmbeddedImageInfo, [162](#)
 - GetGPIOPinDirection, [162](#)
 - GetInterfaceType, [163](#)
 - GetLUTBankInfo, [163](#)
 - GetLUTChannel, [163](#)
 - GetLUTInfo, [164](#)
 - GetMemoryChannel, [164](#)
 - GetMemoryChannelInfo, [164](#)
 - GetNodeMap, [165](#)
 - GetProperty, [165](#)
 - GetPropertyInfo, [165](#)
 - GetRegisterString, [166](#)

- GetStats, [166](#)
- GetStrobe, [166](#)
- GetStrobeInfo, [167](#)
- GetTriggerDelay, [167](#)
- GetTriggerDelayInfo, [168](#)
- GetTriggerMode, [168](#)
- GetTriggerModeInfo, [169](#)
- IsConnected, [169](#)
- ReadGVCPMemory, [169](#)
- ReadGVCPRegister, [170](#)
- ReadGVCPRegisterBlock, [170](#)
- ReadRegister, [170](#)
- ReadRegisterBlock, [170](#)
- ResetStats, [171](#)
- RestoreFromMemoryChannel, [171](#)
- RetrieveBuffer, [171](#)
- SaveToMemoryChannel, [172](#)
- SetActiveLUTBank, [172](#)
- SetCallback, [172](#)
- SetCamera, [173](#)
- SetConfiguration, [173](#)
- SetEmbeddedImageInfo, [173](#)
- SetGPIOPinDirection, [174](#)
- SetLUTChannel, [174](#)
- SetProperty, [175](#)
- SetStrobe, [175](#)
- SetTriggerDelay, [176](#)
- SetTriggerMode, [176](#)
- SetUserBuffers, [177](#)
- StartCapture, [177](#)
- StartSyncCapture, [178](#)
- StopCapture, [178](#)
- TestGainNode, [178](#)
- WaitForBufferEvent, [179](#)
- WriteGVCPMemory, [179](#)
- WriteGVCPRegister, [179](#)
- WriteGVCPRegisterBlock, [179](#)
- WriteRegister, [179](#)
- WriteRegisterBlock, [180](#)
- m_busMgr, [180](#)
- FlyCapture2::GigECamera
 - ~GigECamera, [186](#)
 - Connect, [186](#)
 - DeregisterAllEvents, [186](#)
 - DeregisterEvent, [187](#)
 - Disconnect, [187](#)
 - DiscoverGigEPacketSize, [187](#)
 - EnableLUT, [187](#)
 - FireSoftwareTrigger, [188](#)
 - GetActiveLUTBank, [188](#)
 - GetCameraInfo, [188](#)
 - GetConfiguration, [189](#)
 - GetCycleTime, [189](#)
 - GetEmbeddedImageInfo, [189](#)
 - GetGPIOPinDirection, [192](#)
 - GetGigEConfig, [190](#)
 - GetGigEImageBinningSettings, [190](#)
 - GetGigEImageSettings, [191](#)
 - GetGigEImageSettingsInfo, [191](#)
 - GetGigEImagingMode, [191](#)
 - GetGigEProperty, [192](#)
 - GetGigEStreamChannelInfo, [192](#)
 - GetLUTBankInfo, [193](#)
 - GetLUTChannel, [193](#)
 - GetLUTInfo, [194](#)
 - GetMemoryChannel, [194](#)
 - GetMemoryChannelInfo, [194](#)
 - GetNumStreamChannels, [195](#)
 - GetProperty, [195](#)
 - GetPropertyInfo, [196](#)
 - GetRegisterString, [196](#)
 - GetStats, [197](#)
 - GetStrobe, [197](#)
 - GetStrobeInfo, [197](#)
 - GetTriggerDelay, [198](#)
 - GetTriggerDelayInfo, [198](#)
 - GetTriggerMode, [199](#)
 - GetTriggerModeInfo, [199](#)
 - GigECamera, [186](#)
 - IsConnected, [200](#)
 - QueryGigEImagingMode, [200](#)
 - ReadGVCPMemory, [200](#)
 - ReadGVCPRegister, [201](#)
 - ReadGVCPRegisterBlock, [201](#)
 - ReadRegister, [201](#)
 - ReadRegisterBlock, [202](#)
 - RegisterAllEvents, [202](#)
 - RegisterEvent, [202](#)
 - ResetStats, [202](#)
 - RestoreFromMemoryChannel, [202](#)
 - RetrieveBuffer, [203](#)
 - SaveToMemoryChannel, [203](#)
 - SetActiveLUTBank, [204](#)
 - SetCallback, [204](#)
 - SetConfiguration, [204](#)
 - SetEmbeddedImageInfo, [205](#)
 - SetGPIOPinDirection, [207](#)
 - SetGigEConfig, [205](#)
 - SetGigEImageBinningSettings, [206](#)
 - SetGigEImageSettings, [206](#)

- SetGigElmagingMode, [206](#)
- SetGigEProperty, [207](#)
- SetGigEStreamChannelInfo, [207](#)
- SetLUTChannel, [208](#)
- SetProperty, [208](#)
- SetStrobe, [209](#)
- SetTriggerDelay, [209](#)
- SetTriggerMode, [210](#)
- SetUserBuffers, [210](#)
- StartCapture, [211](#)
- StartSyncCapture, [212](#)
- StopCapture, [212](#)
- WaitForBufferEvent, [212](#)
- WriteGVCPMemory, [213](#)
- WriteGVCPRegister, [213](#)
- WriteGVCPRegisterBlock, [213](#)
- WriteRegister, [214](#)
- WriteRegisterBlock, [214](#)
- FlyCapture2::GigEConfig
 - GigEConfig, [215](#)
 - enablePacketResend, [215](#)
 - registerTimeout, [215](#)
 - registerTimeoutRetries, [215](#)
- FlyCapture2::GigElmImageSettings
 - GigElmImageSettings, [216](#)
 - height, [216](#)
 - offsetX, [217](#)
 - offsetY, [217](#)
 - pixelFormat, [217](#)
 - reserved, [217](#)
 - width, [217](#)
- FlyCapture2::GigElmImageSettingsInfo
 - GigElmImageSettingsInfo, [218](#)
 - imageHStepSize, [218](#)
 - imageVStepSize, [218](#)
 - maxHeight, [218](#)
 - maxWidth, [218](#)
 - offsetHStepSize, [218](#)
 - offsetVStepSize, [219](#)
 - pixelFormatBitField, [219](#)
 - reserved, [219](#)
 - vendorPixelFormatBitField, [219](#)
- FlyCapture2::GigEProperty
 - isReadable, [220](#)
 - isWritable, [220](#)
 - max, [220](#)
 - min, [220](#)
 - propType, [220](#)
 - value, [220](#)
- FlyCapture2::GigEStreamChannel
 - GigEStreamChannel, [222](#)
 - destinationIpAddress, [222](#)
 - doNotFragment, [222](#)
 - hostPort, [222](#)
 - interPacketDelay, [222](#)
 - networkInterfaceIndex, [222](#)
 - packetSize, [222](#)
 - sourcePort, [222](#)
- FlyCapture2::H264Option
 - H264Option, [223](#)
 - bitrate, [223](#)
 - frameRate, [223](#)
 - height, [223](#)
 - reserved, [223](#)
 - width, [223](#)
- FlyCapture2::IPAddress
 - IPAddress, [249](#)
 - octets, [249](#)
 - operator==, [249](#)
- FlyCapture2::Image
 - ~Image, [228](#)
 - CalculateStatistics, [228](#)
 - Convert, [229](#)
 - DeepCopy, [229](#)
 - DetermineBitsPerPixel, [230](#)
 - GetBayerTileFormat, [230](#)
 - GetBitsPerPixel, [230](#)
 - GetBlockId, [230](#)
 - GetColorProcessing, [231](#)
 - GetCols, [231](#)
 - GetData, [231](#)
 - GetDataSize, [231](#)
 - GetDefaultColorProcessing, [231](#)
 - GetDefaultOutputFormat, [232](#)
 - GetDimensions, [232](#)
 - GetMetadata, [232](#)
 - GetPixelFormat, [233](#)
 - GetReceivedDataSize, [233](#)
 - GetRows, [233](#)
 - GetStride, [233](#)
 - GetTimeStamp, [233](#)
 - Image, [227](#), [228](#)
 - Iso, [239](#)
 - ReleaseBuffer, [234](#)
 - Save, [234–237](#)
 - SetBlockId, [237](#)
 - SetColorProcessing, [237](#)
 - SetData, [237](#)
 - SetDefaultColorProcessing, [238](#)
 - SetDefaultOutputFormat, [238](#)

- SetDimensions, [239](#)
- operator(), [233](#)
- operator=, [234](#)
- FlyCapture2::ImageMetadata
 - ImageMetadata, [240](#)
 - embeddedBrightness, [240](#)
 - embeddedExposure, [240](#)
 - embeddedFrameCounter, [240](#)
 - embeddedGPIOPinState, [241](#)
 - embeddedGain, [241](#)
 - embeddedROIPosition, [241](#)
 - embeddedShutter, [241](#)
 - embeddedStrobePattern, [241](#)
 - embeddedTimeStamp, [241](#)
 - embeddedWhiteBalance, [241](#)
 - reserved, [241](#)
- FlyCapture2::ImageStatistics
 - ~ImageStatistics, [243](#)
 - DisableAll, [244](#)
 - EnableAll, [244](#)
 - EnableGreyOnly, [244](#)
 - EnableHSLOnly, [244](#)
 - EnableRGBOnly, [244](#)
 - GetChannelStatus, [244](#)
 - GetHistogram, [245](#)
 - GetMean, [245](#)
 - GetNumPixelValues, [245](#)
 - GetPixelValueRange, [246](#)
 - GetRange, [246](#)
 - GetStatistics, [247](#)
 - ImageStatistics, [243](#)
 - ImageStatsCalculator, [248](#)
 - SetChannelStatus, [247](#)
 - StatisticsChannel, [243](#)
 - operator=, [247](#)
- FlyCapture2::Internal
 - GetInternal, [248](#)
- FlyCapture2::JPEGOOption
 - JPEGOOption, [250](#)
 - progressive, [250](#)
 - quality, [250](#)
 - reserved, [250](#)
- FlyCapture2::JPG2Option
 - JPG2Option, [251](#)
 - quality, [251](#)
 - reserved, [251](#)
- FlyCapture2::LUTData
 - LUTData, [252](#)
 - enabled, [252](#)
 - inputBitDepth, [252](#)
 - numBanks, [252](#)
 - numChannels, [253](#)
 - numEntries, [253](#)
 - outputBitDepth, [253](#)
 - reserved, [253](#)
 - supported, [253](#)
- FlyCapture2::MACAddress
 - MACAddress, [254](#)
 - octets, [254](#)
 - operator==, [254](#)
- FlyCapture2::MJPGOption
 - MJPGOption, [255](#)
 - frameRate, [255](#)
 - quality, [255](#)
 - reserved, [255](#)
- FlyCapture2::NodeMap
 - ~NodeMap, [256](#)
 - NodeMap, [256](#)
 - _GetDeviceName, [256](#)
 - _GetNode, [256](#)
 - _GetNodes, [256](#)
 - _InvalidateNodes, [256](#)
 - _Poll, [256](#)
- FlyCapture2::PGMOption
 - PGMOption, [257](#)
 - binaryFile, [257](#)
 - reserved, [257](#)
- FlyCapture2::PGRGuid
 - PGRGuid, [258](#)
 - operator==, [258](#)
 - value, [258](#)
- FlyCapture2::PNGOption
 - PNGOption, [259](#)
 - compressionLevel, [259](#)
 - interlaced, [259](#)
 - reserved, [259](#)
- FlyCapture2::PPMOption
 - PPMOption, [260](#)
 - binaryFile, [260](#)
 - reserved, [260](#)
- FlyCapture2::Property
 - Property, [262](#)
 - absControl, [262](#)
 - absValue, [262](#)
 - autoManualMode, [262](#)
 - onOff, [262](#)
 - onePush, [262](#)
 - present, [262](#)
 - reserved, [262](#)
 - type, [262](#)

- valueA, [262](#)
- valueB, [263](#)
- FlyCapture2::PropertyInfo
 - PropertyInfo, [264](#)
 - absMax, [264](#)
 - absMin, [264](#)
 - absValSupported, [264](#)
 - autoSupported, [264](#)
 - manualSupported, [264](#)
 - max, [265](#)
 - min, [265](#)
 - onOffSupported, [265](#)
 - onePushSupported, [265](#)
 - pUnitAbbr, [265](#)
 - pUnits, [265](#)
 - present, [265](#)
 - readOutSupported, [265](#)
 - reserved, [265](#)
 - type, [265](#)
- FlyCapture2::StrobeControl
 - StrobeControl, [266](#)
 - delay, [266](#)
 - duration, [266](#)
 - onOff, [267](#)
 - polarity, [267](#)
 - reserved, [267](#)
 - source, [267](#)
- FlyCapture2::StrobeInfo
 - StrobeInfo, [268](#)
 - maxValue, [268](#)
 - minValue, [268](#)
 - onOffSupported, [268](#)
 - polaritySupported, [268](#)
 - present, [268](#)
 - readOutSupported, [268](#)
 - reserved, [268](#)
 - source, [269](#)
- FlyCapture2::SystemInfo
 - byteOrder, [271](#)
 - cpuDescription, [271](#)
 - driverList, [271](#)
 - gpuDescription, [271](#)
 - libraryList, [271](#)
 - numCpuCores, [271](#)
 - osDescription, [271](#)
 - osType, [271](#)
 - reserved, [271](#)
 - screenHeight, [271](#)
 - screenWidth, [272](#)
 - sysMemSize, [272](#)
- FlyCapture2::TIFFOption
 - CompressionMethod, [273](#)
 - TIFFOption, [273](#)
 - compression, [273](#)
 - reserved, [273](#)
- FlyCapture2::TimeStamp
 - TimeStamp, [274](#)
 - cycleCount, [274](#)
 - cycleOffset, [274](#)
 - cycleSeconds, [274](#)
 - microSeconds, [274](#)
 - reserved, [274](#)
 - seconds, [275](#)
- FlyCapture2::TopologyNode
 - ~TopologyNode, [277](#)
 - AddChild, [277](#)
 - AddPortType, [278](#)
 - AssignGuidToNode, [278](#)
 - GetChild, [278](#)
 - GetDeviceld, [279](#)
 - GetGuid, [279](#)
 - GetInterfaceType, [279](#)
 - GetNodeType, [279](#)
 - GetNumChildren, [279](#)
 - GetNumPorts, [279](#)
 - GetPortType, [280](#)
 - NodeType, [276](#)
 - PortType, [276](#)
 - TopologyNode, [277](#)
 - operator=, [280](#)
- FlyCapture2::TriggerMode
 - TriggerMode, [281](#)
 - mode, [281](#)
 - onOff, [281](#)
 - parameter, [281](#)
 - polarity, [281](#)
 - reserved, [281](#)
 - source, [281](#)
- FlyCapture2::TriggerModelInfo
 - TriggerModelInfo, [283](#)
 - modeMask, [283](#)
 - onOffSupported, [283](#)
 - polaritySupported, [283](#)
 - present, [283](#)
 - readOutSupported, [283](#)
 - reserved, [283](#)
 - softwareTriggerSupported, [283](#)
 - sourceMask, [283](#)
 - valueReadable, [283](#)
- FlyCapture2::Utilities

- CheckDriver, [284](#)
- GetDriverDeviceName, [285](#)
- GetLibraryVersion, [285](#)
- GetSystemInfo, [285](#)
- LaunchBrowser, [285](#)
- LaunchCommand, [286](#)
- LaunchCommandAsync, [286](#)
- LaunchHelp, [286](#)
- FlyCapture2Defs.h, [290](#)
- FULL_32BIT_VALUE, [295](#)
- NULL, [295](#)
- FlyCapture2GUI.h, [295](#)
- FlyCapture2Platform.h, [296](#)
- FLYCAPTURE2_API, [296](#)
- FLYCAPTURE2_LOCAL, [296](#)
- FlyCapture2Video, [145](#)
- FlyCapture2::FlyCapture2Video, [146](#)
- FlyCapture2Video.h, [296](#)
- FlyCapture2VideoDefs.h, [296](#)
- FlyCapture3ApiGuiWrapper, [148](#)
- FlyCap3CameraControl::FlyCapture3-
ApiGuiWrapper, [149](#)
- FlyCapture3ApiGuiWrapper.h, [297](#)
- WRAPPER_API, [297](#)
- ForceAllIPAddressesAutomatically
FlyCapture2::BusManager, [53](#), [54](#)
- ForceIPAddressToCamera
FlyCapture2::BusManager, [54](#)
- Format7ImageSettings, [149](#)
- FlyCapture2::Format7ImageSettings,
[150](#)
- Format7Info, [151](#)
- FlyCapture2::Format7Info, [152](#)
- Format7PacketInfo, [154](#)
- FlyCapture2::Format7PacketInfo,
[154](#)
- FrameRate
Enumerations, [22](#)
- GCCamera, [155](#)
- FlyCapture2::GCCamera, [159](#)
- GCCamera.h, [297](#)
- GCCamera::GetXML
FlyCapture2::GCCamera, [160](#)
- GPIOPinState
FlyCapture2::EmbeddedImageInfo,
[134](#)
- GetActiveLUTBank
FlyCapture2::Camera, [68](#)
- FlyCapture2::CameraBase, [98](#)
- FlyCapture2::GCCamera, [160](#)
- FlyCapture2::GigECamera, [188](#)
- GetBayerTileFormat
FlyCapture2::Image, [230](#)
- GetBitsPerPixel
FlyCapture2::Image, [230](#)
- GetBlockId
FlyCapture2::Image, [230](#)
- GetBuildDate
FlyCapture2::Error, [137](#)
- GetCameraFromIPAddress
FlyCapture2::BusManager, [55](#)
- GetCameraFromIndex
FlyCapture2::BusManager, [54](#)
- GetCameraFromSerialNumber
FlyCapture2::BusManager, [55](#)
- GetCameraInfo
FlyCapture2::Camera, [69](#)
- FlyCapture2::CameraBase, [99](#)
- FlyCapture2::GCCamera, [161](#)
- FlyCapture2::GigECamera, [188](#)
- GetCameraSerialNumberFromIndex
FlyCapture2::BusManager, [56](#)
- GetCause
FlyCapture2::Error, [137](#)
- GetChannelStatus
FlyCapture2::ImageStatistics, [244](#)
- GetChild
FlyCapture2::TopologyNode, [278](#)
- GetColorProcessing
FlyCapture2::Image, [231](#)
- GetCols
FlyCapture2::Image, [231](#)
- GetConfiguration
FlyCapture2::Camera, [69](#)
- FlyCapture2::CameraBase, [99](#)
- FlyCapture2::GCCamera, [161](#)
- FlyCapture2::GigECamera, [189](#)
- GetControlNameList
FlyCap3CameraControl::FlyCapture3-
ApiGuiWrapper, [149](#)
- GetCycleTime
FlyCapture2::Camera, [69](#)
- FlyCapture2::CameraBase, [99](#)
- FlyCapture2::GCCamera, [161](#)
- FlyCapture2::GigECamera, [189](#)
- GetData
FlyCapture2::Image, [231](#)
- GetDataSize
FlyCapture2::Image, [231](#)
- GetDefaultColorProcessing

- FlyCapture2::Image, [231](#)
- GetDefaultOutputFormat
 - FlyCapture2::Image, [232](#)
- GetDescription
 - FlyCapture2::Error, [137](#)
- GetDeviceFromIndex
 - FlyCapture2::BusManager, [56](#)
- GetDeviceId
 - FlyCapture2::TopologyNode, [279](#)
- GetDialogNameList
 - FlyCap3CameraControl::FlyCapture3-
ApiGuiWrapper, [149](#)
- GetDimensions
 - FlyCapture2::Image, [232](#)
- GetDriverDeviceName
 - FlyCapture2::Utilities, [285](#)
- GetEmbeddedImageInfo
 - FlyCapture2::Camera, [70](#)
 - FlyCapture2::CameraBase, [100](#)
 - FlyCapture2::GCCamera, [162](#)
 - FlyCapture2::GigECamera, [189](#)
- GetFilename
 - FlyCapture2::Error, [137](#)
- GetFormat7Configuration
 - FlyCapture2::Camera, [70](#)
- GetFormat7Info
 - FlyCapture2::Camera, [71](#)
- GetGPIOPinDirection
 - FlyCapture2::Camera, [71](#)
 - FlyCapture2::CameraBase, [100](#)
 - FlyCapture2::GCCamera, [162](#)
 - FlyCapture2::GigECamera, [192](#)
- GetGigEConfig
 - FlyCapture2::GigECamera, [190](#)
- GetGigEImageBinningSettings
 - FlyCapture2::GigECamera, [190](#)
- GetGigEImageSettings
 - FlyCapture2::GigECamera, [191](#)
- GetGigEImageSettingsInfo
 - FlyCapture2::GigECamera, [191](#)
- GetGigEImagingMode
 - FlyCapture2::GigECamera, [191](#)
- GetGigEProperty
 - FlyCapture2::GigECamera, [192](#)
- GetGigEStreamChannelInfo
 - FlyCapture2::GigECamera, [192](#)
- GetGuid
 - FlyCapture2::TopologyNode, [279](#)
- GetHistogram
 - FlyCapture2::ImageStatistics, [245](#)
- GetInterfaceType
 - FlyCapture2::GCCamera, [163](#)
 - FlyCapture2::TopologyNode, [279](#)
- GetInterfaceTypeFromGuid
 - FlyCapture2::BusManager, [56](#)
- GetInternal
 - FlyCapture2::Internal, [248](#)
- GetLUTBankInfo
 - FlyCapture2::Camera, [72](#)
 - FlyCapture2::CameraBase, [101](#)
 - FlyCapture2::GCCamera, [163](#)
 - FlyCapture2::GigECamera, [193](#)
- GetLUTChannel
 - FlyCapture2::Camera, [72](#)
 - FlyCapture2::CameraBase, [101](#)
 - FlyCapture2::GCCamera, [163](#)
 - FlyCapture2::GigECamera, [193](#)
- GetLUTInfo
 - FlyCapture2::Camera, [72](#)
 - FlyCapture2::CameraBase, [101](#)
 - FlyCapture2::GCCamera, [164](#)
 - FlyCapture2::GigECamera, [194](#)
- GetLibraryVersion
 - FlyCapture2::Utilities, [285](#)
- GetLine
 - FlyCapture2::Error, [137](#)
- GetMean
 - FlyCapture2::ImageStatistics, [245](#)
- GetMemoryChannel
 - FlyCapture2::Camera, [73](#)
 - FlyCapture2::CameraBase, [102](#)
 - FlyCapture2::GCCamera, [164](#)
 - FlyCapture2::GigECamera, [194](#)
- GetMemoryChannelInfo
 - FlyCapture2::Camera, [73](#)
 - FlyCapture2::CameraBase, [102](#)
 - FlyCapture2::GCCamera, [164](#)
 - FlyCapture2::GigECamera, [194](#)
- GetMetadata
 - FlyCapture2::Image, [232](#)
- GetNodeMap
 - FlyCapture2::GCCamera, [165](#)
- GetNodeType
 - FlyCapture2::TopologyNode, [279](#)
- GetNumChildren
 - FlyCapture2::TopologyNode, [279](#)
- GetNumDialogs
 - FlyCap3CameraControl::FlyCapture3-
ApiGuiWrapper, [149](#)
- GetNumOfCameras

- FlyCapture2::BusManager, 57
- GetNumOfControls
 - FlyCap3CameraControl::FlyCapture3-
ApiGuiWrapper, 149
- GetNumOfDevices
 - FlyCapture2::BusManager, 57
- GetNumPixelValues
 - FlyCapture2::ImageStatistics, 245
- GetNumPorts
 - FlyCapture2::TopologyNode, 279
- GetNumStreamChannels
 - FlyCapture2::GigECamera, 195
- GetPixelFormat
 - FlyCapture2::Image, 233
- GetPixelValueRange
 - FlyCapture2::ImageStatistics, 246
- GetPortType
 - FlyCapture2::TopologyNode, 280
- GetProperty
 - FlyCapture2::Camera, 74
 - FlyCapture2::CameraBase, 103
 - FlyCapture2::GCCamera, 165
 - FlyCapture2::GigECamera, 195
- GetPropertyInfo
 - FlyCapture2::Camera, 74
 - FlyCapture2::CameraBase, 103
 - FlyCapture2::GCCamera, 165
 - FlyCapture2::GigECamera, 196
- GetRange
 - FlyCapture2::ImageStatistics, 246
- GetReceivedDataSize
 - FlyCapture2::Image, 233
- GetRegisterString
 - FlyCapture2::Camera, 75
 - FlyCapture2::CameraBase, 104
 - FlyCapture2::GCCamera, 166
 - FlyCapture2::GigECamera, 196
- GetRows
 - FlyCapture2::Image, 233
- GetStatistics
 - FlyCapture2::ImageStatistics, 247
- GetStats
 - FlyCapture2::Camera, 75
 - FlyCapture2::CameraBase, 104
 - FlyCapture2::GCCamera, 166
 - FlyCapture2::GigECamera, 197
- GetStride
 - FlyCapture2::Image, 233
- GetStrobe
 - FlyCapture2::Camera, 75
- FlyCapture2::CameraBase, 104
- FlyCapture2::GCCamera, 166
- FlyCapture2::GigECamera, 197
- GetStrobeInfo
 - FlyCapture2::Camera, 76
 - FlyCapture2::CameraBase, 105
 - FlyCapture2::GCCamera, 167
 - FlyCapture2::GigECamera, 197
- GetSyncStatus
 - MultiSyncLibrary::SyncManager, 269
- GetSystemInfo
 - FlyCapture2::Utilities, 285
- GetTimeSinceSynced
 - MultiSyncLibrary::SyncManager, 269
- GetTimeStamp
 - FlyCapture2::Image, 233
- GetTopology
 - FlyCapture2::BusManager, 57
- GetTriggerDelay
 - FlyCapture2::Camera, 76
 - FlyCapture2::CameraBase, 105
 - FlyCapture2::GCCamera, 167
 - FlyCapture2::GigECamera, 198
- GetTriggerDelayInfo
 - FlyCapture2::Camera, 76
 - FlyCapture2::CameraBase, 106
 - FlyCapture2::GCCamera, 168
 - FlyCapture2::GigECamera, 198
- GetTriggerMode
 - FlyCapture2::Camera, 77
 - FlyCapture2::CameraBase, 106
 - FlyCapture2::GCCamera, 168
 - FlyCapture2::GigECamera, 199
- GetTriggerModeInfo
 - FlyCapture2::Camera, 77
 - FlyCapture2::CameraBase, 107
 - FlyCapture2::GCCamera, 169
 - FlyCapture2::GigECamera, 199
- GetType
 - FlyCapture2::Error, 138
- GetUsbLinkInfo
 - FlyCapture2::BusManager, 58
- GetUsbPortStatus
 - FlyCapture2::BusManager, 58
- GetVideoModeAndFrameRate
 - FlyCapture2::Camera, 78
- GetVideoModeAndFrameRateInfo
 - FlyCapture2::Camera, 78
- GigE specific enumerations, 29
- GigEPropertyType, 29

- GigE specific structures, [33](#)
- GigECamera, [180](#)
 - FlyCapture2::GigECamera, [186](#)
- GigECamera.h, [298](#)
- GigEConfig, [215](#)
 - FlyCapture2::GigEConfig, [215](#)
- GigEImageSettings, [216](#)
 - FlyCapture2::GigEImageSettings, [216](#)
- GigEImageSettingsInfo, [217](#)
 - FlyCapture2::GigEImageSettings-Info, [218](#)
- GigEProperty, [219](#)
- GigEPropertyType
 - GigE specific enumerations, [29](#)
- GigEStreamChannel, [220](#)
 - FlyCapture2::GigEStreamChannel, [222](#)
- Global constants, [15](#)
 - sk_maxNumPorts, [15](#)
 - sk_maxStringLength, [15](#)
- GrabMode
 - Enumerations, [23](#)
- GrabTimeout
 - Enumerations, [23](#)
- H264Option, [222](#)
 - FlyCapture2::H264Option, [223](#)
- Hide
 - FlyCapture2::CameraControlDlg, [120](#)
- IIDC specific structures, [34](#)
- IPAddress, [248](#)
 - FlyCapture2::IPAddress, [249](#)
- Image, [224](#)
 - FlyCapture2::Image, [227](#), [228](#)
- Image saving structures., [35](#)
 - CameraEventCallback, [36](#)
- Image.h, [298](#)
- ImageEventCallback
 - FlyCapture2, [45](#)
- ImageFileFormat
 - Enumerations, [23](#)
- ImageMetadata, [239](#)
 - FlyCapture2::ImageMetadata, [240](#)
- ImageStatistics, [241](#)
 - FlyCapture2::ImageStatistics, [243](#)
- ImageStatistics.h, [298](#)
- ImageStatsCalculator
 - FlyCapture2::ImageStatistics, [248](#)
- InterfaceType
 - Enumerations, [24](#)
- Internal, [248](#)
- Internal.h, [298](#)
- InternalError
 - FlyCapture2::Error, [139](#)
- IsCameraControlable
 - FlyCapture2::BusManager, [60](#)
- IsConnected
 - FlyCapture2::Camera, [79](#)
 - FlyCapture2::CameraBase, [107](#)
 - FlyCapture2::GCCamera, [169](#)
 - FlyCapture2::GigECamera, [200](#)
- IsTimingBusConnected
 - MultiSyncLibrary::SyncManager, [269](#)
- IsVisible
 - FlyCapture2::CameraControlDlg, [120](#)
- Iso
 - FlyCapture2::Image, [239](#)
- JPEGOOption, [249](#)
 - FlyCapture2::JPEGOOption, [250](#)
- JPG2Option, [251](#)
 - FlyCapture2::JPG2Option, [251](#)
- LUTData, [251](#)
 - FlyCapture2::LUTData, [252](#)
- LaunchBrowser
 - FlyCapture2::Utilities, [285](#)
- LaunchCommand
 - FlyCapture2::Utilities, [286](#)
- LaunchCommandAsync
 - FlyCapture2::Utilities, [286](#)
- LaunchHelp
 - FlyCapture2::Utilities, [286](#)
- Licensing.dox, [299](#)
- MACAddress, [253](#)
 - FlyCapture2::MACAddress, [254](#)
- MJPGOption, [254](#)
 - FlyCapture2::MJPGOption, [255](#)
- Mode
 - Enumerations, [24](#)
- MultiSyncLibrary, [46](#)
 - PGRSyncError, [47](#)
 - PGRSyncMessage, [47](#)
- MultiSyncLibrary.h, [299](#)
- MultiSyncLibrary::SyncManager
 - ~SyncManager, [269](#)
 - DisableCrossPCSynchronization, [269](#)
 - EnableCrossPCSynchronization, [269](#)
 - GetSyncStatus, [269](#)

- GetTimeSinceSynced, [269](#)
- IsTimingBusConnected, [269](#)
- QueryCrossPCSSynchronization-
Setting, [269](#)
- RescanMasterTimingBus, [270](#)
- Start, [270](#)
- Stop, [270](#)
- SyncManager, [269](#)
- MultiSyncLibraryDefs.h, [299](#)
- MultiSyncLibraryPlatform.h, [299](#)
- NULL
 - FlyCapture2Defs.h, [295](#)
- NodeMap, [255](#)
 - FlyCapture2::NodeMap, [256](#)
- NodeMap.h, [300](#)
- NodeType
 - FlyCapture2::TopologyNode, [276](#)
- OSType
 - FlyCapture2, [46](#)
- Open
 - FlyCapture2::FlyCapture2Video, [147](#)
- PCleBusSpeed
 - Enumerations, [25](#)
- PGMOption, [256](#)
 - FlyCapture2::PGMOption, [257](#)
- PGRGuid, [257](#)
 - FlyCapture2::PGRGuid, [258](#)
- PGRSyncError
 - MultiSyncLibrary, [47](#)
- PGRSyncMessage
 - MultiSyncLibrary, [47](#)
- PNGOption, [258](#)
 - FlyCapture2::PNGOption, [259](#)
- PPMOption, [259](#)
 - FlyCapture2::PPMOption, [260](#)
- PixelFormat
 - Enumerations, [25](#)
- PortType
 - FlyCapture2::TopologyNode, [276](#)
- PrintErrorTrace
 - FlyCapture2::Error, [138](#)
- Property, [260](#)
 - FlyCapture2::Property, [262](#)
- PropertyInfo, [263](#)
 - FlyCapture2::PropertyInfo, [264](#)
- PropertyType
 - Enumerations, [26](#)
- QueryCrossPCSSynchronizationSetting
 - MultiSyncLibrary::SyncManager, [269](#)
- QueryGigEImagingMode
 - FlyCapture2::GigECamera, [200](#)
- ROIPosition
 - FlyCapture2::EmbeddedImageInfo, [134](#)
- ReadGVCPMemory
 - FlyCapture2::GCCamera, [169](#)
 - FlyCapture2::GigECamera, [200](#)
- ReadGVCPRegister
 - FlyCapture2::GCCamera, [170](#)
 - FlyCapture2::GigECamera, [201](#)
- ReadGVCPRegisterBlock
 - FlyCapture2::GCCamera, [170](#)
 - FlyCapture2::GigECamera, [201](#)
- ReadPhyRegister
 - FlyCapture2::BusManager, [60](#)
- ReadRegister
 - FlyCapture2::Camera, [79](#)
 - FlyCapture2::CameraBase, [107](#)
 - FlyCapture2::GCCamera, [170](#)
 - FlyCapture2::GigECamera, [201](#)
- ReadRegisterBlock
 - FlyCapture2::Camera, [80](#)
 - FlyCapture2::CameraBase, [108](#)
 - FlyCapture2::GCCamera, [170](#)
 - FlyCapture2::GigECamera, [202](#)
- RegisterAllEvents
 - FlyCapture2::Camera, [80](#)
 - FlyCapture2::CameraBase, [108](#)
 - FlyCapture2::GigECamera, [202](#)
- RegisterCallback
 - FlyCapture2::BusManager, [60](#)
- RegisterEvent
 - FlyCapture2::Camera, [80](#)
 - FlyCapture2::CameraBase, [108](#)
 - FlyCapture2::GigECamera, [202](#)
- ReleaseBuffer
 - FlyCapture2::Image, [234](#)
- RescanBus
 - FlyCapture2::BusManager, [61](#)
- RescanMasterTimingBus
 - MultiSyncLibrary::SyncManager, [270](#)
- ResetStats
 - FlyCapture2::Camera, [80](#)
 - FlyCapture2::CameraBase, [108](#)
 - FlyCapture2::GCCamera, [171](#)
 - FlyCapture2::GigECamera, [202](#)
- RestoreFromMemoryChannel
 - FlyCapture2::Camera, [80](#)
 - FlyCapture2::CameraBase, [108](#)
 - FlyCapture2::GCCamera, [171](#)

- FlyCapture2::GigECamera, [202](#)
- RetrieveBuffer
 - FlyCapture2::Camera, [81](#)
 - FlyCapture2::CameraBase, [109](#)
 - FlyCapture2::GCCamera, [171](#)
 - FlyCapture2::GigECamera, [203](#)
- Save
 - FlyCapture2::Image, [234–237](#)
- SaveToMemoryChannel
 - FlyCapture2::Camera, [81](#)
 - FlyCapture2::CameraBase, [109](#)
 - FlyCapture2::GCCamera, [172](#)
 - FlyCapture2::GigECamera, [203](#)
- SetActiveLUTBank
 - FlyCapture2::Camera, [82](#)
 - FlyCapture2::CameraBase, [110](#)
 - FlyCapture2::GCCamera, [172](#)
 - FlyCapture2::GigECamera, [204](#)
- SetBlockId
 - FlyCapture2::Image, [237](#)
- SetCallback
 - FlyCapture2::Camera, [82](#)
 - FlyCapture2::CameraBase, [110](#)
 - FlyCapture2::GCCamera, [172](#)
 - FlyCapture2::GigECamera, [204](#)
- SetCamera
 - FlyCapture2::GCCamera, [173](#)
- SetChannelStatus
 - FlyCapture2::ImageStatistics, [247](#)
- SetColorProcessing
 - FlyCapture2::Image, [237](#)
- SetConfiguration
 - FlyCapture2::Camera, [83](#)
 - FlyCapture2::CameraBase, [111](#)
 - FlyCapture2::GCCamera, [173](#)
 - FlyCapture2::GigECamera, [204](#)
- SetData
 - FlyCapture2::Image, [237](#)
- SetDefaultColorProcessing
 - FlyCapture2::Image, [238](#)
- SetDefaultOutputFormat
 - FlyCapture2::Image, [238](#)
- SetDimensions
 - FlyCapture2::Image, [239](#)
- SetEmbeddedImageInfo
 - FlyCapture2::Camera, [83](#)
 - FlyCapture2::CameraBase, [111](#)
 - FlyCapture2::GCCamera, [173](#)
 - FlyCapture2::GigECamera, [205](#)
- SetFormat7Configuration
 - FlyCapture2::Camera, [83, 84](#)
- SetGPIOPinDirection
 - FlyCapture2::Camera, [84](#)
 - FlyCapture2::CameraBase, [111](#)
 - FlyCapture2::GCCamera, [174](#)
 - FlyCapture2::GigECamera, [207](#)
- SetGigEConfig
 - FlyCapture2::GigECamera, [205](#)
- SetGigEImageBinningSettings
 - FlyCapture2::GigECamera, [206](#)
- SetGigEImageSettings
 - FlyCapture2::GigECamera, [206](#)
- SetGigEImagingMode
 - FlyCapture2::GigECamera, [206](#)
- SetGigEProperty
 - FlyCapture2::GigECamera, [207](#)
- SetGigEStreamChannelInfo
 - FlyCapture2::GigECamera, [207](#)
- SetLUTChannel
 - FlyCapture2::Camera, [85](#)
 - FlyCapture2::CameraBase, [112](#)
 - FlyCapture2::GCCamera, [174](#)
 - FlyCapture2::GigECamera, [208](#)
- SetMaximumFileSize
 - FlyCapture2::FlyCapture2Video, [148](#)
- SetProperty
 - FlyCapture2::Camera, [85](#)
 - FlyCapture2::CameraBase, [112](#)
 - FlyCapture2::GCCamera, [175](#)
 - FlyCapture2::GigECamera, [208](#)
- SetStrobe
 - FlyCapture2::Camera, [86](#)
 - FlyCapture2::CameraBase, [113](#)
 - FlyCapture2::GCCamera, [175](#)
 - FlyCapture2::GigECamera, [209](#)
- SetTitle
 - FlyCapture2::CameraControlDlg, [120](#)
 - FlyCapture2::CameraSelectionDlg, [127](#)
- SetTriggerDelay
 - FlyCapture2::Camera, [86](#)
 - FlyCapture2::CameraBase, [113](#)
 - FlyCapture2::GCCamera, [176](#)
 - FlyCapture2::GigECamera, [209](#)
- SetTriggerMode
 - FlyCapture2::Camera, [87](#)
 - FlyCapture2::CameraBase, [114](#)
 - FlyCapture2::GCCamera, [176](#)
 - FlyCapture2::GigECamera, [210](#)

- SetUserBuffers
 - FlyCapture2::Camera, [87](#)
 - FlyCapture2::CameraBase, [114](#)
 - FlyCapture2::GCCamera, [177](#)
 - FlyCapture2::GigECamera, [210](#)
- SetVideoModeAndFrameRate
 - FlyCapture2::Camera, [88](#)
- Show
 - FlyCapture2::CameraControlDlg, [120](#)
- ShowCameraSelectionDialog
 - FlyCap3CameraControl::FlyCapture3-ApiGuiWrapper, [149](#)
- ShowDialogByIndex
 - FlyCap3CameraControl::FlyCapture3-ApiGuiWrapper, [149](#)
- ShowDialogByName
 - FlyCap3CameraControl::FlyCapture3-ApiGuiWrapper, [149](#)
- ShowModal
 - FlyCapture2::CameraControlDlg, [121](#)
 - FlyCapture2::CameraSelectionDlg, [127](#)
- ShowPropertyGridDialog
 - FlyCap3CameraControl::FlyCapture3-ApiGuiWrapper, [149](#)
- Start
 - MultiSyncLibrary::SyncManager, [270](#)
- StartCapture
 - FlyCapture2::Camera, [88](#)
 - FlyCapture2::CameraBase, [115](#)
 - FlyCapture2::GCCamera, [177](#)
 - FlyCapture2::GigECamera, [211](#)
- StartSyncCapture
 - FlyCapture2::Camera, [89](#)
 - FlyCapture2::CameraBase, [116](#)
 - FlyCapture2::GCCamera, [178](#)
 - FlyCapture2::GigECamera, [212](#)
- StatisticsChannel
 - FlyCapture2::ImageStatistics, [243](#)
- Stop
 - MultiSyncLibrary::SyncManager, [270](#)
- StopCapture
 - FlyCapture2::Camera, [89](#)
 - FlyCapture2::CameraBase, [117](#)
 - FlyCapture2::GCCamera, [178](#)
 - FlyCapture2::GigECamera, [212](#)
- StrobeControl, [266](#)
 - FlyCapture2::StrobeControl, [266](#)
- StrobeInfo, [267](#)
 - FlyCapture2::StrobeInfo, [268](#)
- Structures, [30](#)
 - TriggerDelay, [31](#)
 - TriggerDelayInfo, [31](#)
- SyncManager, [269](#)
 - MultiSyncLibrary::SyncManager, [269](#)
- SystemInfo, [270](#)
- TIFFOption, [272](#)
 - FlyCapture2::TIFFOption, [273](#)
- TestGainNode
 - FlyCapture2::GCCamera, [178](#)
- TimeStamp, [273](#)
 - FlyCapture2::TimeStamp, [274](#)
- TopologyNode, [275](#)
 - FlyCapture2::TopologyNode, [277](#)
- TopologyNode.h, [300](#)
- TriggerDelay
 - Structures, [31](#)
- TriggerDelayInfo
 - Structures, [31](#)
- TriggerMode, [280](#)
 - FlyCapture2::TriggerMode, [281](#)
- TriggerModelInfo, [282](#)
 - FlyCapture2::TriggerModelInfo, [283](#)
- UnregisterCallback
 - FlyCapture2::BusManager, [61](#)
- Utilities, [284](#)
- Utilities.h, [300](#)
- ValidateFormat7Settings
 - FlyCapture2::Camera, [90](#)
- Video saving structures., [37](#)
- VideoMode
 - Enumerations, [27](#)
- WRAPPER_API
 - FlyCapture3ApiGuiWrapper.h, [297](#)
- WaitForBufferEvent
 - FlyCapture2::Camera, [90](#)
 - FlyCapture2::CameraBase, [117](#)
 - FlyCapture2::GCCamera, [179](#)
 - FlyCapture2::GigECamera, [212](#)
- WriteGVCPMemory
 - FlyCapture2::GCCamera, [179](#)
 - FlyCapture2::GigECamera, [213](#)
- WriteGVCPRegister
 - FlyCapture2::GCCamera, [179](#)
 - FlyCapture2::GigECamera, [213](#)
- WriteGVCPRegisterBlock
 - FlyCapture2::GCCamera, [179](#)
 - FlyCapture2::GigECamera, [213](#)

- WritePhyRegister
 - FlyCapture2::BusManager, [61](#)
- WriteRegister
 - FlyCapture2::Camera, [91](#)
 - FlyCapture2::CameraBase, [117](#)
 - FlyCapture2::GCCamera, [179](#)
 - FlyCapture2::GigECamera, [214](#)
- WriteRegisterBlock
 - FlyCapture2::Camera, [91](#)
 - FlyCapture2::CameraBase, [118](#)
 - FlyCapture2::GCCamera, [180](#)
 - FlyCapture2::GigECamera, [214](#)
- _GetDeviceName
 - FlyCapture2::NodeMap, [256](#)
- _GetNode
 - FlyCapture2::NodeMap, [256](#)
- _GetNodes
 - FlyCapture2::NodeMap, [256](#)
- _InvalidateNodes
 - FlyCapture2::NodeMap, [256](#)
- _Poll
 - FlyCapture2::NodeMap, [256](#)
- absControl
 - FlyCapture2::Property, [262](#)
- absMax
 - FlyCapture2::PropertyInfo, [264](#)
- absMin
 - FlyCapture2::PropertyInfo, [264](#)
- absValSupported
 - FlyCapture2::PropertyInfo, [264](#)
- absValue
 - FlyCapture2::Property, [262](#)
- applicationIPAddress
 - FlyCapture2::CameraInfo, [123](#)
- applicationPort
 - FlyCapture2::CameraInfo, [123](#)
- asyncBusSpeed
 - FlyCapture2::FC2Config, [142](#)
- autoManualMode
 - FlyCapture2::Property, [262](#)
- autoSupported
 - FlyCapture2::PropertyInfo, [264](#)
- available
 - FlyCapture2::EmbeddedImageInfo-Property, [135](#)
- bandwidthAllocation
 - FlyCapture2::FC2Config, [142](#)
- bayerTileFormat
 - FlyCapture2::CameraInfo, [123](#)
- binaryFile
 - FlyCapture2::PGMOption, [257](#)
 - FlyCapture2::PPMOption, [260](#)
- bitrate
 - FlyCapture2::H264Option, [223](#)
- brightness
 - FlyCapture2::EmbeddedImageInfo, [134](#)
- build
 - FlyCapture2::FC2Version, [145](#)
- busNumber
 - FlyCapture2::CameraInfo, [123](#)
- byteOrder
 - FlyCapture2::SystemInfo, [271](#)
- cameraCurrents
 - FlyCapture2::CameraStats, [129](#)
- cameraPowerUp
 - FlyCapture2::CameraStats, [129](#)
- cameraVoltages
 - FlyCapture2::CameraStats, [129](#)
- ccpStatus
 - FlyCapture2::CameraInfo, [124](#)
- chipIdHi
 - FlyCapture2::ConfigROM, [131](#)
- chipIdLo
 - FlyCapture2::ConfigROM, [131](#)
- compression
 - FlyCapture2::TIFFOption, [273](#)
- compressionLevel
 - FlyCapture2::PNGOption, [259](#)
- configROM
 - FlyCapture2::CameraInfo, [124](#)
- cpuDescription
 - FlyCapture2::SystemInfo, [271](#)
- cycleCount
 - FlyCapture2::TimeStamp, [274](#)
- cycleOffset
 - FlyCapture2::TimeStamp, [274](#)
- cycleSeconds
 - FlyCapture2::TimeStamp, [274](#)
- defaultGateway
 - FlyCapture2::CameraInfo, [124](#)
- delay
 - FlyCapture2::StrobeControl, [266](#)
- destinationIpAddress
 - FlyCapture2::GigEStreamChannel, [222](#)

- doNotFragment
 - FlyCapture2::GigEStreamChannel, [222](#)
- driverList
 - FlyCapture2::SystemInfo, [271](#)
- driverName
 - FlyCapture2::CameraInfo, [124](#)
- driverType
 - FlyCapture2::CameraInfo, [124](#)
- duration
 - FlyCapture2::StrobeControl, [266](#)
- embeddedBrightness
 - FlyCapture2::ImageMetadata, [240](#)
- embeddedExposure
 - FlyCapture2::ImageMetadata, [240](#)
- embeddedFrameCounter
 - FlyCapture2::ImageMetadata, [240](#)
- embeddedGPIOPinState
 - FlyCapture2::ImageMetadata, [241](#)
- embeddedGain
 - FlyCapture2::ImageMetadata, [241](#)
- embeddedROIPosition
 - FlyCapture2::ImageMetadata, [241](#)
- embeddedShutter
 - FlyCapture2::ImageMetadata, [241](#)
- embeddedStrobePattern
 - FlyCapture2::ImageMetadata, [241](#)
- embeddedTimeStamp
 - FlyCapture2::ImageMetadata, [241](#)
- embeddedWhiteBalance
 - FlyCapture2::ImageMetadata, [241](#)
- enablePacketResend
 - FlyCapture2::GigEConfig, [215](#)
- enabled
 - FlyCapture2::LUTData, [252](#)
- exposure
 - FlyCapture2::EmbeddedImageInfo, [134](#)
- firmwareBuildTime
 - FlyCapture2::CameraInfo, [124](#)
- firmwareVersion
 - FlyCapture2::CameraInfo, [124](#)
- frameCounter
 - FlyCapture2::EmbeddedImageInfo, [134](#)
- frameRate
 - FlyCapture2::AVIOption, [49](#)
 - FlyCapture2::H264Option, [223](#)
 - FlyCapture2::MJPGOption, [255](#)
- gain
 - FlyCapture2::EmbeddedImageInfo, [134](#)
- gigEMajorVersion
 - FlyCapture2::CameraInfo, [124](#)
- gigEMinorVersion
 - FlyCapture2::CameraInfo, [124](#)
- gpuDescription
 - FlyCapture2::SystemInfo, [271](#)
- grabMode
 - FlyCapture2::FC2Config, [142](#)
- grabTimeout
 - FlyCapture2::FC2Config, [143](#)
- height
 - FlyCapture2::Format7ImageSettings, [150](#)
 - FlyCapture2::GigEImageSettings, [216](#)
 - FlyCapture2::H264Option, [223](#)
- highPerformanceRetrieveBuffer
 - FlyCapture2::FC2Config, [143](#)
- hostPort
 - FlyCapture2::GigEStreamChannel, [222](#)
- iidcVer
 - FlyCapture2::CameraInfo, [124](#)
- imageCorrupt
 - FlyCapture2::CameraStats, [129](#)
- imageDriverDropped
 - FlyCapture2::CameraStats, [129](#)
- imageDropped
 - FlyCapture2::CameraStats, [129](#)
- imageHStepSize
 - FlyCapture2::Format7Info, [152](#)
 - FlyCapture2::GigEImageSettings-Info, [218](#)
- imageVStepSize
 - FlyCapture2::Format7Info, [152](#)
 - FlyCapture2::GigEImageSettings-Info, [218](#)
- imageXmitFailed
 - FlyCapture2::CameraStats, [129](#)
- indexedColor_8bit
 - FlyCapture2::BMPOption, [50](#)
- inputBitDepth
 - FlyCapture2::LUTData, [252](#)

- interPacketDelay
 - FlyCapture2::GigEStreamChannel, 222
- interfaceType
 - FlyCapture2::CameraInfo, 125
- interlaced
 - FlyCapture2::PNGOption, 259
- ipAddress
 - FlyCapture2::CameraInfo, 125
- isColorCamera
 - FlyCapture2::CameraInfo, 125
- isReadable
 - FlyCapture2::GigEProperty, 220
- isWritable
 - FlyCapture2::GigEProperty, 220
- isochBusSpeed
 - FlyCapture2::FC2Config, 143
- libraryList
 - FlyCapture2::SystemInfo, 271
- m_busMgr
 - FlyCapture2::GCCamera, 180
- m_pCameraData
 - FlyCapture2::CameraBase, 118
- macAddress
 - FlyCapture2::CameraInfo, 125
- major
 - FlyCapture2::FC2Version, 145
- manualSupported
 - FlyCapture2::PropertyInfo, 264
- max
 - FlyCapture2::GigEProperty, 220
 - FlyCapture2::PropertyInfo, 265
- maxBytesPerPacket
 - FlyCapture2::Format7PacketInfo, 154
- maxHeight
 - FlyCapture2::Format7Info, 152
 - FlyCapture2::GigEImageSettings-Info, 218
- maxPacketSize
 - FlyCapture2::Format7Info, 152
- maxValue
 - FlyCapture2::StrobeInfo, 268
- maxWidth
 - FlyCapture2::Format7Info, 152
 - FlyCapture2::GigEImageSettings-Info, 218
- maximumBusSpeed
 - FlyCapture2::CameraInfo, 125
- microSeconds
 - FlyCapture2::TimeStamp, 274
- min
 - FlyCapture2::GigEProperty, 220
 - FlyCapture2::PropertyInfo, 265
- minNumImageNotifications
 - FlyCapture2::FC2Config, 143
- minPacketSize
 - FlyCapture2::Format7Info, 153
- minValue
 - FlyCapture2::StrobeInfo, 268
- minor
 - FlyCapture2::FC2Version, 145
- mode
 - FlyCapture2::Format7ImageSettings, 150
 - FlyCapture2::Format7Info, 153
 - FlyCapture2::TriggerMode, 281
- modeMask
 - FlyCapture2::TriggerModelInfo, 283
- modelName
 - FlyCapture2::CameraInfo, 125
- networkInterfaceIndex
 - FlyCapture2::GigEStreamChannel, 222
- nodeNumber
 - FlyCapture2::CameraInfo, 125
- nodeVendorId
 - FlyCapture2::ConfigROM, 131
- numBanks
 - FlyCapture2::LUTData, 252
- numBuffers
 - FlyCapture2::FC2Config, 143
- numChannels
 - FlyCapture2::LUTData, 253
- numCpuCores
 - FlyCapture2::SystemInfo, 271
- numCurrents
 - FlyCapture2::CameraStats, 129
- numEntries
 - FlyCapture2::LUTData, 253
- numImageNotifications
 - FlyCapture2::FC2Config, 143
- numResendPacketsReceived
 - FlyCapture2::CameraStats, 129
- numResendPacketsRequested
 - FlyCapture2::CameraStats, 130
- numVoltages

- FlyCapture2::CameraStats, 130
- octets
 - FlyCapture2::IPAddress, 249
 - FlyCapture2::MACAddress, 254
- offsetHStepSize
 - FlyCapture2::Format7Info, 153
 - FlyCapture2::GigEImageSettings-Info, 218
- offsetVStepSize
 - FlyCapture2::Format7Info, 153
 - FlyCapture2::GigEImageSettings-Info, 219
- offsetX
 - FlyCapture2::Format7ImageSettings, 150
 - FlyCapture2::GigEImageSettings, 217
- offsetY
 - FlyCapture2::Format7ImageSettings, 150
 - FlyCapture2::GigEImageSettings, 217
- onOff
 - FlyCapture2::EmbeddedImageInfo-Property, 135
 - FlyCapture2::Property, 262
 - FlyCapture2::StrobeControl, 267
 - FlyCapture2::TriggerMode, 281
- onOffSupported
 - FlyCapture2::PropertyInfo, 265
 - FlyCapture2::StrobeInfo, 268
 - FlyCapture2::TriggerModelInfo, 283
- onePush
 - FlyCapture2::Property, 262
- onePushSupported
 - FlyCapture2::PropertyInfo, 265
- operator()
 - FlyCapture2::Image, 233
- operator=
 - FlyCapture2::Error, 138
 - FlyCapture2::Image, 234
 - FlyCapture2::ImageStatistics, 247
 - FlyCapture2::TopologyNode, 280
- operator==
 - FlyCapture2::Error, 138
 - FlyCapture2::IPAddress, 249
 - FlyCapture2::MACAddress, 254
 - FlyCapture2::PGRGuid, 258
- osDescription
 - FlyCapture2::SystemInfo, 271
- osType
 - FlyCapture2::SystemInfo, 271
- outputBitDepth
 - FlyCapture2::LUTData, 253
- pUnitAbbr
 - FlyCapture2::PropertyInfo, 265
- pUnits
 - FlyCapture2::PropertyInfo, 265
- packetSize
 - FlyCapture2::Format7Info, 153
 - FlyCapture2::GigEStreamChannel, 222
- parameter
 - FlyCapture2::TriggerMode, 281
- pcieBusSpeed
 - FlyCapture2::CameraInfo, 125
- percentage
 - FlyCapture2::Format7Info, 153
- pixelFormat
 - FlyCapture2::Format7ImageSettings, 151
 - FlyCapture2::GigEImageSettings, 217
- pixelFormatBitField
 - FlyCapture2::Format7Info, 153
 - FlyCapture2::GigEImageSettings-Info, 219
- polarity
 - FlyCapture2::StrobeControl, 267
 - FlyCapture2::TriggerMode, 281
- polaritySupported
 - FlyCapture2::StrobeInfo, 268
 - FlyCapture2::TriggerModelInfo, 283
- portErrors
 - FlyCapture2::CameraStats, 130
- present
 - FlyCapture2::Property, 262
 - FlyCapture2::PropertyInfo, 265
 - FlyCapture2::StrobeInfo, 268
 - FlyCapture2::TriggerModelInfo, 283
- progressive
 - FlyCapture2::JPEGOption, 250
- propType
 - FlyCapture2::GigEProperty, 220
- pszKeyword
 - FlyCapture2::ConfigROM, 132
- quality

- FlyCapture2::JPEGOption, [250](#)
- FlyCapture2::JPG2Option, [251](#)
- FlyCapture2::MJPGOption, [255](#)
- readOutSupported
 - FlyCapture2::PropertyInfo, [265](#)
 - FlyCapture2::StrobeInfo, [268](#)
 - FlyCapture2::TriggerModelInfo, [283](#)
- recommendedBytesPerPacket
 - FlyCapture2::Format7PacketInfo, [154](#)
- regReadFailed
 - FlyCapture2::CameraStats, [130](#)
- regWriteFailed
 - FlyCapture2::CameraStats, [130](#)
- registerTimeout
 - FlyCapture2::FC2Config, [144](#)
 - FlyCapture2::GigEConfig, [215](#)
- registerTimeoutRetries
 - FlyCapture2::FC2Config, [144](#)
 - FlyCapture2::GigEConfig, [215](#)
- reserved
 - FlyCapture2::AVIOption, [49](#)
 - FlyCapture2::BMPOption, [50](#)
 - FlyCapture2::CameraInfo, [125](#)
 - FlyCapture2::CameraStats, [130](#)
 - FlyCapture2::ConfigROM, [132](#)
 - FlyCapture2::FC2Config, [144](#)
 - FlyCapture2::Format7ImageSettings, [151](#)
 - FlyCapture2::Format7Info, [153](#)
 - FlyCapture2::Format7PacketInfo, [154](#)
 - FlyCapture2::GigEImageSettings, [217](#)
 - FlyCapture2::GigEImageSettings-Info, [219](#)
 - FlyCapture2::H264Option, [223](#)
 - FlyCapture2::ImageMetadata, [241](#)
 - FlyCapture2::JPEGOption, [250](#)
 - FlyCapture2::JPG2Option, [251](#)
 - FlyCapture2::LUTData, [253](#)
 - FlyCapture2::MJPGOption, [255](#)
 - FlyCapture2::PGMOption, [257](#)
 - FlyCapture2::PNGOption, [259](#)
 - FlyCapture2::PPMOption, [260](#)
 - FlyCapture2::Property, [262](#)
 - FlyCapture2::PropertyInfo, [265](#)
 - FlyCapture2::StrobeControl, [267](#)
 - FlyCapture2::StrobeInfo, [268](#)
 - FlyCapture2::SystemInfo, [271](#)
 - FlyCapture2::TIFFOption, [273](#)
 - FlyCapture2::TimeStamp, [274](#)
 - FlyCapture2::TriggerMode, [281](#)
 - FlyCapture2::TriggerModelInfo, [283](#)
- screenHeight
 - FlyCapture2::SystemInfo, [271](#)
- screenWidth
 - FlyCapture2::SystemInfo, [272](#)
- seconds
 - FlyCapture2::TimeStamp, [275](#)
- sensorInfo
 - FlyCapture2::CameraInfo, [125](#)
- sensorResolution
 - FlyCapture2::CameraInfo, [126](#)
- serialNumber
 - FlyCapture2::CameraInfo, [126](#)
- shutter
 - FlyCapture2::EmbeddedImageInfo, [134](#)
- sk_maxNumPorts
 - Global constants, [15](#)
- sk_maxStringLength
 - Global constants, [15](#)
- softwareTriggerSupported
 - FlyCapture2::TriggerModelInfo, [283](#)
- source
 - FlyCapture2::StrobeControl, [267](#)
 - FlyCapture2::StrobeInfo, [269](#)
 - FlyCapture2::TriggerMode, [281](#)
- sourceMask
 - FlyCapture2::TriggerModelInfo, [283](#)
- sourcePort
 - FlyCapture2::GigEStreamChannel, [222](#)
- strobePattern
 - FlyCapture2::EmbeddedImageInfo, [134](#)
- subnetMask
 - FlyCapture2::CameraInfo, [126](#)
- supported
 - FlyCapture2::LUTData, [253](#)
- sysMemSize
 - FlyCapture2::SystemInfo, [272](#)
- temperature
 - FlyCapture2::CameraStats, [130](#)
- timeSinceBusReset
 - FlyCapture2::CameraStats, [130](#)

timeSinceInitialization
 FlyCapture2::CameraStats, [130](#)
 timeStamp
 FlyCapture2::CameraStats, [130](#)
 timestamp
 FlyCapture2::EmbeddedImageInfo, [134](#)
 type
 FlyCapture2::FC2Version, [145](#)
 FlyCapture2::Property, [262](#)
 FlyCapture2::PropertyInfo, [265](#)
 unitBytesPerPacket
 FlyCapture2::Format7PacketInfo, [154](#)
 unitSWVer
 FlyCapture2::ConfigROM, [132](#)
 unitSpecId
 FlyCapture2::ConfigROM, [132](#)
 unitSubSWVer
 FlyCapture2::ConfigROM, [132](#)
 userDefinedName
 FlyCapture2::CameraInfo, [126](#)

 value
 FlyCapture2::GigEProperty, [220](#)
 FlyCapture2::PGRGuid, [258](#)
 valueA
 FlyCapture2::Property, [262](#)
 valueB
 FlyCapture2::Property, [263](#)
 valueReadable
 FlyCapture2::TriggerModelInfo, [283](#)
 vendorName
 FlyCapture2::CameraInfo, [126](#)
 vendorPixelFormatBitField
 FlyCapture2::Format7Info, [153](#)
 FlyCapture2::GigEImageSettings-Info, [219](#)
 vendorUniqueInfo_0
 FlyCapture2::ConfigROM, [132](#)
 vendorUniqueInfo_1
 FlyCapture2::ConfigROM, [132](#)
 vendorUniqueInfo_2
 FlyCapture2::ConfigROM, [132](#)
 vendorUniqueInfo_3
 FlyCapture2::ConfigROM, [132](#)

 whiteBalance
 FlyCapture2::EmbeddedImageInfo, [134](#)
 width
 FlyCapture2::Format7ImageSettings, [151](#)
 FlyCapture2::GigEImageSettings, [217](#)
 FlyCapture2::H264Option, [223](#)
 xmlURL1
 FlyCapture2::CameraInfo, [126](#)
 xmlURL2
 FlyCapture2::CameraInfo, [126](#)