

FlyCapture2

2.10.3.0

Generated by Doxygen 1.8.11

Contents

1	Module Index	1
1.1	Modules	1
2	Namespace Index	3
2.1	Namespace List	3
3	Hierarchical Index	5
3.1	Class Hierarchy	5
4	Class Index	7
4.1	Class List	7
5	File Index	11
5.1	File List	11
6	Module Documentation	13
6.1	Global constants	13
6.1.1	Detailed Description	13
6.1.2	Variable Documentation	13
6.1.2.1	sk_maxNumPorts	13
6.1.2.2	sk_maxStringLength	13
6.2	Enumerations	14
6.2.1	Detailed Description	19
6.2.2	Enumeration Type Documentation	19
6.2.2.1	BandwidthAllocation	19
6.2.2.2	BayerTileFormat	19

6.2.2.3	BusCallbackType	20
6.2.2.4	BusSpeed	20
6.2.2.5	ColorProcessingAlgorithm	20
6.2.2.6	DriverType	21
6.2.2.7	ErrorType	21
6.2.2.8	FrameRate	22
6.2.2.9	GrabMode	23
6.2.2.10	GrabTimeout	23
6.2.2.11	ImageFileFormat	23
6.2.2.12	InterfaceType	24
6.2.2.13	Mode	24
6.2.2.14	PCleBusSpeed	25
6.2.2.15	PixelFormat	25
6.2.2.16	PropertyType	26
6.2.2.17	VideoMode	26
6.3	GigE specific enumerations	28
6.3.1	Detailed Description	28
6.3.2	Enumeration Type Documentation	28
6.3.2.1	GigEPropertyType	28
6.4	Structures	29
6.4.1	Detailed Description	30
6.4.2	Typedef Documentation	30
6.4.2.1	TriggerDelay	30
6.4.2.2	TriggerDelayInfo	30
6.5	GigE specific structures	31
6.5.1	Detailed Description	31
6.6	IIDC specific structures	32
6.6.1	Detailed Description	32
6.7	Image saving structures.	33
6.7.1	Detailed Description	33
6.7.2	Typedef Documentation	33
6.7.2.1	CameraEventCallback	33

7 Namespace Documentation	35
7.1 FlyCap3CameraControl Namespace Reference	35
7.2 FlyCapture2 Namespace Reference	35
7.2.1 Typedef Documentation	43
7.2.1.1 AsyncCommandCallback	43
7.2.1.2 BusEventCallback	44
7.2.1.3 CallbackHandle	44
7.2.1.4 ImageEventCallback	44
7.2.2 Enumeration Type Documentation	44
7.2.2.1 ByteOrder	44
7.2.2.2 OSType	44
7.3 MultiSyncLibrary Namespace Reference	45
7.3.1 Enumeration Type Documentation	45
7.3.1.1 PGRSyncError	45
7.3.1.2 PGRSyncMessage	45
8 Class Documentation	47
8.1 AVIOption Struct Reference	47
8.1.1 Detailed Description	47
8.1.2 Constructor & Destructor Documentation	47
8.1.2.1 AVIOption()	47
8.1.3 Member Data Documentation	47
8.1.3.1 frameRate	47
8.1.3.2 reserved	48
8.2 AVIRecorder Class Reference	48
8.2.1 Detailed Description	48
8.2.2 Constructor & Destructor Documentation	48
8.2.2.1 AVIRecorder()	48
8.2.2.2 ~AVIRecorder()	48
8.2.3 Member Function Documentation	48
8.2.3.1 AVIAppend(Image *pImage)	48

8.2.3.2	AVIClose()	49
8.2.3.3	AVIOpen(const char *pFileName, AVIOption *pOption)	49
8.2.3.4	AVIOpen(const char *pFileName, MJPGOption *pOption)	49
8.2.3.5	AVIOpen(const char *pFileName, H264Option *pOption)	50
8.3	BMPOption Struct Reference	50
8.3.1	Detailed Description	51
8.3.2	Constructor & Destructor Documentation	51
8.3.2.1	BMPOption()	51
8.3.3	Member Data Documentation	51
8.3.3.1	indexedColor_8bit	51
8.3.3.2	reserved	51
8.4	BusManager Class Reference	51
8.4.1	Detailed Description	52
8.4.2	Constructor & Destructor Documentation	53
8.4.2.1	BusManager()	53
8.4.2.2	~BusManager()	53
8.4.3	Member Function Documentation	53
8.4.3.1	DiscoverGigECameras(CameraInfo *gigECameras, unsigned int *arraySize)	53
8.4.3.2	FireBusReset(PGRGuid *pGuid)	53
8.4.3.3	ForceAllIPAddressesAutomatically()	54
8.4.3.4	ForceAllIPAddressesAutomatically(unsigned int serialNumber)	54
8.4.3.5	ForceIPAddressToCamera(MACAddress macAddress, IPAddress ipAddress, I↔PAddress subnetMask, IPAddress defaultGateway)	54
8.4.3.6	GetCameraFromIndex(unsigned int index, PGRGuid *pGuid)	54
8.4.3.7	GetCameraFromIPAddress(IPAddress ipAddress, PGRGuid *pGuid)	55
8.4.3.8	GetCameraFromSerialNumber(unsigned int serialNumber, PGRGuid *pGuid)	55
8.4.3.9	GetCameraSerialNumberFromIndex(unsigned int index, unsigned int *pSerial↔Number)	56
8.4.3.10	GetDeviceFromIndex(unsigned int index, PGRGuid *pGuid)	56
8.4.3.11	GetInterfaceTypeFromGuid(PGRGuid *pGuid, InterfaceType *pInterfaceType)	56
8.4.3.12	GetNumOfCameras(unsigned int *pNumCameras)	57

8.4.3.13	GetNumOfDevices(unsigned int *pNumDevices)	57
8.4.3.14	GetTopology(TopologyNode *pNode)	57
8.4.3.15	GetUsbLinkInfo(PGRGuid guid, unsigned int *pValue)	57
8.4.3.16	GetUsbPortStatus(PGRGuid guid, unsigned int *pValue)	58
8.4.3.17	IsCameraControlable(PGRGuid *pGuid, bool *pControlable)	58
8.4.3.18	ReadPhyRegister(PGRGuid guid, unsigned int page, unsigned int port, unsigned int address, unsigned int *pValue)	58
8.4.3.19	RegisterCallback(BusEventCallback busEventCallback, BusCallbackType callbackType, void *pParameter, CallbackHandle *pCallbackHandle)	59
8.4.3.20	RescanBus()	59
8.4.3.21	UnregisterCallback(CallbackHandle callbackHandle)	59
8.4.3.22	WritePhyRegister(PGRGuid guid, unsigned int page, unsigned int port, unsigned int address, unsigned int value)	60
8.5	Camera Class Reference	60
8.5.1	Detailed Description	63
8.5.2	Constructor & Destructor Documentation	64
8.5.2.1	Camera()	64
8.5.2.2	~Camera()	64
8.5.3	Member Function Documentation	64
8.5.3.1	Connect(PGRGuid *pGuid=NULL)	64
8.5.3.2	DeregisterAllEvents(void)	64
8.5.3.3	DeregisterEvent(EventOptions *pOpts)	64
8.5.3.4	Disconnect()	64
8.5.3.5	EnableLUT(bool on)	64
8.5.3.6	FireSoftwareTrigger(bool broadcast=false)	65
8.5.3.7	GetActiveLUTBank(unsigned int *pActiveBank)	65
8.5.3.8	GetCameraInfo(CameraInfo *pCameraInfo)	66
8.5.3.9	GetConfiguration(FC2Config *pConfig)	66
8.5.3.10	GetCycleTime(TimeStamp *timeStamp)	66
8.5.3.11	GetEmbeddedImageInfo(EmbeddedImageInfo *pInfo)	67
8.5.3.12	GetFormat7Configuration(Format7ImageSettings *pImageSettings, unsigned int *pPacketSize, float *pPercentage)	67

8.5.3.13	GetFormat7Info(Format7Info *pInfo, bool *pSupported)	67
8.5.3.14	GetGPIOPinDirection(unsigned int pin, unsigned int *pDirection)	68
8.5.3.15	GetLUTBankInfo(unsigned int bank, bool *pReadSupported, bool *pWriteSupported)	68
8.5.3.16	GetLUTChannel(unsigned int bank, unsigned int channel, unsigned int size, unsigned int *pEntries)	69
8.5.3.17	GetLUTInfo(LUTData *pData)	69
8.5.3.18	GetMemoryChannel(unsigned int *pCurrentChannel)	70
8.5.3.19	GetMemoryChannelInfo(unsigned int *pNumChannels)	70
8.5.3.20	GetProperty(Property *pProp)	71
8.5.3.21	GetPropertyInfo(PropertyInfo *pPropInfo)	71
8.5.3.22	GetRegisterString(unsigned int registerVal)	71
8.5.3.23	GetStats(CameraStats *pStats)	71
8.5.3.24	GetStrobe(StrobeControl *pStrobeControl)	72
8.5.3.25	GetStrobeInfo(StrobeInfo *pStrobeInfo)	72
8.5.3.26	GetTriggerDelay(TriggerDelay *pTriggerDelay)	72
8.5.3.27	GetTriggerDelayInfo(TriggerDelayInfo *pTriggerDelayInfo)	73
8.5.3.28	GetTriggerMode(TriggerMode *pTriggerMode)	73
8.5.3.29	GetTriggerModelInfo(TriggerModelInfo *pTriggerModelInfo)	74
8.5.3.30	GetVideoModeAndFrameRate(VideoMode *pVideoMode, FrameRate *pFrameRate)	74
8.5.3.31	GetVideoModeAndFrameRateInfo(VideoMode videoMode, FrameRate frameRate, bool *pSupported)	75
8.5.3.32	IsConnected()	75
8.5.3.33	ReadRegister(unsigned int address, unsigned int *pValue)	75
8.5.3.34	ReadRegisterBlock(unsigned short addressHigh, unsigned int addressLow, unsigned int *pBuffer, unsigned int length)	76
8.5.3.35	RegisterAllEvents(EventOptions *pOpts)	76
8.5.3.36	RegisterEvent(EventOptions *pOpts)	76
8.5.3.37	ResetStats()	76
8.5.3.38	RestoreFromMemoryChannel(unsigned int channel)	76
8.5.3.39	RetrieveBuffer(Image *pImage)	77
8.5.3.40	SaveToMemoryChannel(unsigned int channel)	77

8.5.3.41	SetActiveLUTBank(unsigned int activeBank)	78
8.5.3.42	SetCallback(ImageEventCallback callbackFn, const void *pCallbackData=NULL)	78
8.5.3.43	SetConfiguration(const FC2Config *pConfig)	79
8.5.3.44	SetEmbeddedImageInfo(EmbeddedImageInfo *pInfo)	80
8.5.3.45	SetFormat7Configuration(const Format7ImageSettings *pImageSettings, unsigned int packetSize)	80
8.5.3.46	SetFormat7Configuration(const Format7ImageSettings *pImageSettings, float percentSpeed)	81
8.5.3.47	SetGPIOPinDirection(unsigned int pin, unsigned int direction, bool broadcast=false)	81
8.5.3.48	SetLUTChannel(unsigned int bank, unsigned int channel, unsigned int size← Entries, const unsigned int *pEntries)	82
8.5.3.49	SetProperty(const Property *pProp, bool broadcast=false)	82
8.5.3.50	SetStrobe(const StrobeControl *pStrobeControl, bool broadcast=false)	83
8.5.3.51	SetTriggerDelay(const TriggerDelay *pTriggerDelay, bool broadcast=false)	83
8.5.3.52	SetTriggerMode(const TriggerMode *pTriggerMode, bool broadcast=false)	83
8.5.3.53	SetUserBuffers(unsigned char *const pMemBuffers, int size, int numBuffers)	84
8.5.3.54	SetVideoModeAndFrameRate(VideoMode videoMode, FrameRate frameRate)	84
8.5.3.55	StartCapture(ImageEventCallback callbackFn=NULL, const void *pCallback← Data=NULL)	85
8.5.3.56	StartSyncCapture(unsigned int numCameras, const Camera **ppCameras, const ImageEventCallback *pCallbackFns=NULL, const void **pCallbackData← Array=NULL)	85
8.5.3.57	StopCapture()	86
8.5.3.58	ValidateFormat7Settings(const Format7ImageSettings *pImageSettings, bool *pSettingsAreValid, Format7PacketInfo *pPacketInfo)	86
8.5.3.59	WaitForBufferEvent(Image *pImage, unsigned int eventNumber)	86
8.5.3.60	WriteRegister(unsigned int address, unsigned int value, bool broadcast=false)	87
8.5.3.61	WriteRegisterBlock(unsigned short addressHigh, unsigned int addressLow, const unsigned int *pBuffer, unsigned int length)	87
8.6	CameraBase Class Reference	88
8.6.1	Detailed Description	91
8.6.2	Constructor & Destructor Documentation	91
8.6.2.1	CameraBase()	91
8.6.2.2	~CameraBase()	91

8.6.3	Member Function Documentation	91
8.6.3.1	Connect(PGRGuid *pGuid=NULL)=0	91
8.6.3.2	DeregisterAllEvents(void)=0	92
8.6.3.3	DeregisterEvent(EventOptions *pOpts)=0	92
8.6.3.4	Disconnect()=0	92
8.6.3.5	EnableLUT(bool on)=0	92
8.6.3.6	FireSoftwareTrigger(bool broadcast=false)=0	93
8.6.3.7	GetActiveLUTBank(unsigned int *pActiveBank)=0	93
8.6.3.8	GetCameraInfo(CameraInfo *pCameraInfo)=0	93
8.6.3.9	GetConfiguration(FC2Config *pConfig)=0	94
8.6.3.10	GetCycleTime(TimeStamp *timeStamp)=0	94
8.6.3.11	GetEmbeddedImageInfo(EmbeddedImageInfo *pInfo)=0	94
8.6.3.12	GetGPIOPinDirection(unsigned int pin, unsigned int *pDirection)=0	95
8.6.3.13	GetLUTBankInfo(unsigned int bank, bool *pReadSupported, bool *pWriteSupported)=0	95
8.6.3.14	GetLUTChannel(unsigned int bank, unsigned int channel, unsigned int size, unsigned int *pEntries)=0	96
8.6.3.15	GetLUTInfo(LUTData *pData)=0	96
8.6.3.16	GetMemoryChannel(unsigned int *pCurrentChannel)=0	97
8.6.3.17	GetMemoryChannelInfo(unsigned int *pNumChannels)=0	97
8.6.3.18	GetProperty(Property *pProp)=0	98
8.6.3.19	GetPropertyInfo(PropertyInfo *pPropInfo)=0	98
8.6.3.20	GetRegisterString(unsigned int registerVal)	98
8.6.3.21	GetStats(CameraStats *pStats)=0	99
8.6.3.22	GetStrobe(StrobeControl *pStrobeControl)=0	99
8.6.3.23	GetStrobeInfo(StrobeInfo *pStrobeInfo)=0	99
8.6.3.24	GetTriggerDelay(TriggerDelay *pTriggerDelay)=0	100
8.6.3.25	GetTriggerDelayInfo(TriggerDelayInfo *pTriggerDelayInfo)=0	100
8.6.3.26	GetTriggerMode(TriggerMode *pTriggerMode)=0	101
8.6.3.27	GetTriggerModeInfo(TriggerModeInfo *pTriggerModeInfo)=0	101
8.6.3.28	IsConnected()=0	101

8.6.3.29	<code>ReadRegister(unsigned int address, unsigned int *pValue)=0</code>	102
8.6.3.30	<code>ReadRegisterBlock(unsigned short addressHigh, unsigned int addressLow, unsigned int *pBuffer, unsigned int length)=0</code>	102
8.6.3.31	<code>RegisterAllEvents(EventOptions *pOpts)=0</code>	103
8.6.3.32	<code>RegisterEvent(EventOptions *pOpts)=0</code>	103
8.6.3.33	<code>ResetStats()=0</code>	103
8.6.3.34	<code>RestoreFromMemoryChannel(unsigned int channel)=0</code>	103
8.6.3.35	<code>RetrieveBuffer(Image *pImage)=0</code>	103
8.6.3.36	<code>SaveToMemoryChannel(unsigned int channel)=0</code>	104
8.6.3.37	<code>SetActiveLUTBank(unsigned int activeBank)=0</code>	104
8.6.3.38	<code>SetCallback(ImageEventCallback callbackFn, const void *pCallbackData=NULL)=0</code>	105
8.6.3.39	<code>SetConfiguration(const FC2Config *pConfig)=0</code>	105
8.6.3.40	<code>SetEmbeddedImageInfo(EmbeddedImageInfo *pInfo)=0</code>	105
8.6.3.41	<code>SetGPIOPinDirection(unsigned int pin, unsigned int direction, bool broadcast=false)=0</code>	106
8.6.3.42	<code>SetLUTChannel(unsigned int bank, unsigned int channel, unsigned int size←Entries, const unsigned int *pEntries)=0</code>	106
8.6.3.43	<code>SetProperty(const Property *pProp, bool broadcast=false)=0</code>	107
8.6.3.44	<code>SetStrobe(const StrobeControl *pStrobeControl, bool broadcast=false)=0</code>	107
8.6.3.45	<code>SetTriggerDelay(const TriggerDelay *pTriggerDelay, bool broadcast=false)=0</code>	108
8.6.3.46	<code>SetTriggerMode(const TriggerMode *pTriggerMode, bool broadcast=false)=0</code>	108
8.6.3.47	<code>SetUserBuffers(unsigned char *const pMemBuffers, int size, int numBuffers)=0</code>	109
8.6.3.48	<code>StartCapture(ImageEventCallback callbackFn=NULL, const void *pCallback←Data=NULL)=0</code>	109
8.6.3.49	<code>StartSyncCapture(unsigned int numCameras, const CameraBase **ppCameras, const ImageEventCallback *pCallbackFns=NULL, const void **pCallbackData←Array=NULL)</code>	110
8.6.3.50	<code>StopCapture()=0</code>	110
8.6.3.51	<code>WaitForBufferEvent(Image *pImage, unsigned int eventNumber)=0</code>	111
8.6.3.52	<code>WriteRegister(unsigned int address, unsigned int value, bool broadcast=false)=0</code>	111
8.6.3.53	<code>WriteRegisterBlock(unsigned short addressHigh, unsigned int addressLow, const unsigned int *pBuffer, unsigned int length)=0</code>	112
8.6.4	Member Data Documentation	112

8.6.4.1	m_pCameraData	112
8.7	CameraControlDlg Class Reference	112
8.7.1	Detailed Description	113
8.7.2	Constructor & Destructor Documentation	113
8.7.2.1	CameraControlDlg()	113
8.7.2.2	~CameraControlDlg()	113
8.7.3	Member Function Documentation	113
8.7.3.1	Connect(CameraBase *pCamera)	113
8.7.3.2	Disconnect()	114
8.7.3.3	Hide()	114
8.7.3.4	IsVisible()	114
8.7.3.5	SetTitle(const char *title)	114
8.7.3.6	Show()	114
8.7.3.7	Show(void *pParent)	114
8.7.3.8	ShowModal()	114
8.7.3.9	ShowModal(void *pParent)	115
8.8	CameraInfo Struct Reference	115
8.8.1	Detailed Description	116
8.8.2	Constructor & Destructor Documentation	116
8.8.2.1	CameraInfo()	116
8.8.3	Member Data Documentation	116
8.8.3.1	applicationIPAddress	116
8.8.3.2	applicationPort	117
8.8.3.3	bayerTileFormat	117
8.8.3.4	busNumber	117
8.8.3.5	ccpStatus	117
8.8.3.6	configROM	117
8.8.3.7	defaultGateway	117
8.8.3.8	driverName	117
8.8.3.9	driverType	117

8.8.3.10	firmwareBuildTime	117
8.8.3.11	firmwareVersion	117
8.8.3.12	gigEMajorVersion	118
8.8.3.13	gigEMinorVersion	118
8.8.3.14	iidcVer	118
8.8.3.15	interfaceType	118
8.8.3.16	ipAddress	118
8.8.3.17	isColorCamera	118
8.8.3.18	macAddress	118
8.8.3.19	maximumBusSpeed	118
8.8.3.20	modelName	118
8.8.3.21	nodeNumber	118
8.8.3.22	pcieBusSpeed	119
8.8.3.23	reserved	119
8.8.3.24	sensorInfo	119
8.8.3.25	sensorResolution	119
8.8.3.26	serialNumber	119
8.8.3.27	subnetMask	119
8.8.3.28	userDefinedName	119
8.8.3.29	vendorName	119
8.8.3.30	xmlURL1	119
8.8.3.31	xmlURL2	119
8.9	CameraSelectionDlg Class Reference	120
8.9.1	Detailed Description	120
8.9.2	Constructor & Destructor Documentation	120
8.9.2.1	CameraSelectionDlg()	120
8.9.2.2	~CameraSelectionDlg()	120
8.9.3	Member Function Documentation	120
8.9.3.1	SetTitle(const char *title)	120
8.9.3.2	ShowModal(bool *pOk, PGRGuid *pGuid, unsigned int *pSize)	121

8.10 CameraStats Struct Reference	121
8.10.1 Detailed Description	122
8.10.2 Constructor & Destructor Documentation	122
8.10.2.1 CameraStats()	122
8.10.3 Member Data Documentation	122
8.10.3.1 cameraCurrents	122
8.10.3.2 cameraPowerUp	122
8.10.3.3 cameraVoltages	122
8.10.3.4 imageCorrupt	122
8.10.3.5 imageDriverDropped	122
8.10.3.6 imageDropped	122
8.10.3.7 imageXmitFailed	122
8.10.3.8 numCurrents	122
8.10.3.9 numResendPacketsReceived	122
8.10.3.10 numResendPacketsRequested	122
8.10.3.11 numVoltages	122
8.10.3.12 portErrors	123
8.10.3.13 regReadFailed	123
8.10.3.14 regWriteFailed	123
8.10.3.15 reserved	123
8.10.3.16 temperature	123
8.10.3.17 timeSinceBusReset	123
8.10.3.18 timeSinceInitialization	123
8.10.3.19 timeStamp	123
8.11 ConfigROM Struct Reference	123
8.11.1 Detailed Description	124
8.11.2 Constructor & Destructor Documentation	124
8.11.2.1 ConfigROM()	124
8.11.3 Member Data Documentation	124
8.11.3.1 chipIdHi	124

8.11.3.2	chipIdLo	124
8.11.3.3	nodeVendorId	124
8.11.3.4	pszKeyword	124
8.11.3.5	reserved	124
8.11.3.6	unitSpecId	125
8.11.3.7	unitSubSWVer	125
8.11.3.8	unitSWVer	125
8.11.3.9	vendorUniqueInfo_0	125
8.11.3.10	vendorUniqueInfo_1	125
8.11.3.11	vendorUniqueInfo_2	125
8.11.3.12	vendorUniqueInfo_3	125
8.12	EmbeddedImageInfo Struct Reference	125
8.12.1	Detailed Description	126
8.12.2	Member Data Documentation	126
8.12.2.1	brightness	126
8.12.2.2	exposure	126
8.12.2.3	frameCounter	126
8.12.2.4	gain	126
8.12.2.5	GPIOPinState	126
8.12.2.6	ROIPosition	126
8.12.2.7	shutter	126
8.12.2.8	strobePattern	126
8.12.2.9	timestamp	126
8.12.2.10	whiteBalance	126
8.13	EmbeddedImageInfoProperty Struct Reference	126
8.13.1	Detailed Description	127
8.13.2	Constructor & Destructor Documentation	127
8.13.2.1	EmbeddedImageInfoProperty()	127
8.13.3	Member Data Documentation	127
8.13.3.1	available	127

8.13.3.2	onOff	127
8.14	Error Class Reference	127
8.14.1	Detailed Description	128
8.14.2	Constructor & Destructor Documentation	128
8.14.2.1	Error()	128
8.14.2.2	Error(const Error &error)	129
8.14.2.3	~Error()	129
8.14.3	Member Function Documentation	129
8.14.3.1	CollectSupportInformation() const	129
8.14.3.2	GetBuildDate() const	129
8.14.3.3	GetCause() const	129
8.14.3.4	GetDescription() const	129
8.14.3.5	GetFilename() const	130
8.14.3.6	GetLine() const	130
8.14.3.7	GetType() const	130
8.14.3.8	operator!=(const Error &error) const	130
8.14.3.9	operator!=(const ErrorType &errorType) const	130
8.14.3.10	operator=(const Error &error)	130
8.14.3.11	operator==(const Error &error) const	130
8.14.3.12	operator==(const ErrorType &errorType) const	131
8.14.3.13	PrintErrorTrace() const	131
8.14.4	Friends And Related Function Documentation	131
8.14.4.1	InternalError	131
8.15	EventCallbackData Struct Reference	131
8.15.1	Member Data Documentation	131
8.15.1.1	EventData	131
8.15.1.2	EventDataSize	132
8.15.1.3	EventID	132
8.15.1.4	EventName	132
8.15.1.5	EventTimestamp	132

8.15.1.6	EventUserData	132
8.15.1.7	EventUserDataSize	132
8.16	EventOptions Struct Reference	132
8.16.1	Detailed Description	133
8.16.2	Member Data Documentation	133
8.16.2.1	EventCallbackFcn	133
8.16.2.2	EventName	133
8.16.2.3	EventUserData	133
8.16.2.4	EventUserDataSize	133
8.17	FC2Config Struct Reference	133
8.17.1	Detailed Description	134
8.17.2	Constructor & Destructor Documentation	134
8.17.2.1	FC2Config()	134
8.17.3	Member Data Documentation	134
8.17.3.1	asyncBusSpeed	134
8.17.3.2	bandwidthAllocation	135
8.17.3.3	grabMode	135
8.17.3.4	grabTimeout	135
8.17.3.5	highPerformanceRetrieveBuffer	135
8.17.3.6	isochBusSpeed	135
8.17.3.7	minNumImageNotifications	135
8.17.3.8	numBuffers	135
8.17.3.9	numImageNotifications	136
8.17.3.10	registerTimeout	136
8.17.3.11	registerTimeoutRetries	136
8.17.3.12	reserved	136
8.18	FC2Version Struct Reference	136
8.18.1	Detailed Description	137
8.18.2	Member Data Documentation	137
8.18.2.1	build	137

8.18.2.2	major	137
8.18.2.3	minor	137
8.18.2.4	type	137
8.19	FlyCapture3ApiGuiWrapper Class Reference	138
8.19.1	Constructor & Destructor Documentation	138
8.19.1.1	FlyCapture3ApiGuiWrapper(void)	138
8.19.1.2	~FlyCapture3ApiGuiWrapper(void)	138
8.19.2	Member Function Documentation	138
8.19.2.1	ConnectGUILibrary(FlyCapture2::GCCamera &camera)	138
8.19.2.2	DisconnectGUILibrary()	138
8.19.2.3	GetControlNameList()	138
8.19.2.4	GetDialogNameList()	138
8.19.2.5	GetNumDialogs()	138
8.19.2.6	GetNumOfControls()	138
8.19.2.7	ShowCameraSelectionDialog()	138
8.19.2.8	ShowDialogByIndex(int index)	138
8.19.2.9	ShowDialogByName(std::string dialogName)	138
8.19.2.10	ShowPropertyGridDialog()	138
8.20	Format7ImageSettings Struct Reference	139
8.20.1	Detailed Description	139
8.20.2	Constructor & Destructor Documentation	139
8.20.2.1	Format7ImageSettings()	139
8.20.3	Member Data Documentation	139
8.20.3.1	height	139
8.20.3.2	mode	139
8.20.3.3	offsetX	140
8.20.3.4	offsetY	140
8.20.3.5	pixelFormat	140
8.20.3.6	reserved	140
8.20.3.7	width	140

8.21	Format7Info Struct Reference	140
8.21.1	Detailed Description	141
8.21.2	Constructor & Destructor Documentation	141
8.21.2.1	Format7Info()	141
8.21.3	Member Data Documentation	141
8.21.3.1	imageHStepSize	141
8.21.3.2	imageVStepSize	141
8.21.3.3	maxHeight	142
8.21.3.4	maxPacketSize	142
8.21.3.5	maxWidth	142
8.21.3.6	minPacketSize	142
8.21.3.7	mode	142
8.21.3.8	offsetHStepSize	142
8.21.3.9	offsetVStepSize	142
8.21.3.10	packetSize	142
8.21.3.11	percentage	142
8.21.3.12	pixelFormatBitField	142
8.21.3.13	reserved	143
8.21.3.14	vendorPixelFormatBitField	143
8.22	Format7PacketInfo Struct Reference	143
8.22.1	Detailed Description	143
8.22.2	Constructor & Destructor Documentation	143
8.22.2.1	Format7PacketInfo()	143
8.22.3	Member Data Documentation	143
8.22.3.1	maxBytesPerPacket	143
8.22.3.2	recommendedBytesPerPacket	144
8.22.3.3	reserved	144
8.22.3.4	unitBytesPerPacket	144
8.23	GCCamera Class Reference	144
8.23.1	Constructor & Destructor Documentation	147

8.23.1.1	GCCamera(void)	147
8.23.1.2	~GCCamera(void)	147
8.23.2	Member Function Documentation	147
8.23.2.1	Connect(PGRGuid *pGuid=NULL)	147
8.23.2.2	Connect(PGRGuid *pGuid=NULL, const char *filepath=NULL)	147
8.23.2.3	Disconnect()	147
8.23.2.4	EnableLUT(bool on)	147
8.23.2.5	FireSoftwareTrigger(bool broadcast=false)	148
8.23.2.6	GCCamera::GetXML()	148
8.23.2.7	GetActiveLUTBank(unsigned int *pActiveBank)	148
8.23.2.8	GetCameraInfo(CameraInfo *pCameraInfo)	148
8.23.2.9	GetConfiguration(FC2Config *pConfig)	149
8.23.2.10	GetCycleTime(TimeStamp *timeStamp)	149
8.23.2.11	GetEmbeddedImageInfo(EmbeddedImageInfo *pInfo)	149
8.23.2.12	GetGPIOPinDirection(unsigned int pin, unsigned int *pDirection)	150
8.23.2.13	GetInterfaceType()	150
8.23.2.14	GetLUTBankInfo(unsigned int bank, bool *pReadSupported, bool *pWriteSupported)	150
8.23.2.15	GetLUTChannel(unsigned int bank, unsigned int channel, unsigned int size, unsigned int *pEntries, unsigned int *pEntries)	151
8.23.2.16	GetLUTInfo(LUTData *pData)	151
8.23.2.17	GetMemoryChannel(unsigned int *pCurrentChannel)	151
8.23.2.18	GetMemoryChannelInfo(unsigned int *pNumChannels)	152
8.23.2.19	GetNodeMap()	152
8.23.2.20	GetProperty(Property *pProp)	152
8.23.2.21	GetPropertyInfo(PropertyInfo *pPropInfo)	153
8.23.2.22	GetRegisterString(unsigned int registerVal)	153
8.23.2.23	GetStats(CameraStats *pStats)	153
8.23.2.24	GetStrobe(StrobeControl *pStrobeControl)	153
8.23.2.25	GetStrobeInfo(StrobeInfo *pStrobeInfo)	154
8.23.2.26	GetTriggerDelay(TriggerDelay *pTriggerDelay)	154

8.23.2.27 GetTriggerDelayInfo(TriggerDelayInfo *pTriggerDelayInfo)	155
8.23.2.28 GetTriggerMode(TriggerMode *pTriggerMode)	155
8.23.2.29 GetTriggerModelInfo(TriggerModelInfo *pTriggerModelInfo)	156
8.23.2.30 IsConnected()	156
8.23.2.31 ReadGVCPMemory(unsigned int address, unsigned char *pBuffer, unsigned int length)	156
8.23.2.32 ReadGVCPRegister(unsigned int address, unsigned int *pValue)	156
8.23.2.33 ReadGVCPRegisterBlock(unsigned int address, unsigned int *pBuffer, unsigned int length)	156
8.23.2.34 ReadRegister(unsigned int address, unsigned int *pValue)	156
8.23.2.35 ReadRegisterBlock(unsigned short addressHigh, unsigned int addressLow, un- signed int *pBuffer, unsigned int length)	157
8.23.2.36 ResetStats()	157
8.23.2.37 RestoreFromMemoryChannel(unsigned int channel)	157
8.23.2.38 RetrieveBuffer(Image *pImage)	158
8.23.2.39 SaveToMemoryChannel(unsigned int channel)	158
8.23.2.40 SetActiveLUTBank(unsigned int activeBank)	159
8.23.2.41 SetCallback(ImageEventCallback callbackFn, const void *pCallbackData=NULL)	159
8.23.2.42 SetCamera(CameraBase *camera)	160
8.23.2.43 SetCamera(CameraBase *camera, const char *filepath=NULL)	160
8.23.2.44 SetConfiguration(const FC2Config *pConfig)	160
8.23.2.45 SetEmbeddedImageInfo(EmbeddedImageInfo *pInfo)	160
8.23.2.46 SetGPIOPinDirection(unsigned int pin, unsigned int direction, bool broadcast=false)	160
8.23.2.47 SetLUTChannel(unsigned int bank, unsigned int channel, unsigned int size← Entries, const unsigned int *pEntries)	161
8.23.2.48 SetProperty(const Property *pProp, bool broadcast=false)	161
8.23.2.49 SetStrobe(const StrobeControl *pStrobeControl, bool broadcast=false)	162
8.23.2.50 SetTriggerDelay(const TriggerDelay *pTriggerDelay, bool broadcast=false)	162
8.23.2.51 SetTriggerMode(const TriggerMode *pTriggerMode, bool broadcast=false)	163
8.23.2.52 SetUserBuffers(unsigned char *const pMemBuffers, int size, int numBuffers)	163
8.23.2.53 StartCapture(ImageEventCallback callbackFn=NULL, const void *pCallback← Data=NULL)	164

8.23.2.54 StartSyncCapture(unsigned int numCameras, const GigECamera **ppCameras, const ImageEventCallback *pCallbackFns=NULL, const void **pCallbackData↔ Array=NULL)	164
8.23.2.55 StopCapture()	164
8.23.2.56 TestGainNode()	165
8.23.2.57 WaitForBufferEvent(Image *pImage, unsigned int eventNumber)	165
8.23.2.58 WriteGVCPMemory(unsigned int address, const unsigned char *pBuffer, unsigned int length)	165
8.23.2.59 WriteGVCPRegister(unsigned int address, unsigned int value, bool broadcast=false)	165
8.23.2.60 WriteGVCPRegisterBlock(unsigned int address, const unsigned int *pBuffer, unsigned int length)	165
8.23.2.61 WriteRegister(unsigned int address, unsigned int value, bool broadcast=false)	165
8.23.2.62 WriteRegisterBlock(unsigned short addressHigh, unsigned int addressLow, const unsigned int *pBuffer, unsigned int length)	166
8.23.3 Member Data Documentation	166
8.23.3.1 m_busMgr	166
8.24 GigECamera Class Reference	167
8.24.1 Detailed Description	170
8.24.2 Constructor & Destructor Documentation	171
8.24.2.1 GigECamera()	171
8.24.2.2 ~GigECamera()	171
8.24.3 Member Function Documentation	171
8.24.3.1 Connect(PGRGuid *pGuid=NULL)	171
8.24.3.2 DeregisterAllEvents(void)	171
8.24.3.3 DeregisterEvent(EventOptions *pOpts)	171
8.24.3.4 Disconnect()	171
8.24.3.5 DiscoverGigEPacketSize(unsigned int *packetSize)	171
8.24.3.6 EnableLUT(bool on)	172
8.24.3.7 FireSoftwareTrigger(bool broadcast=false)	172
8.24.3.8 GetActiveLUTBank(unsigned int *pActiveBank)	172
8.24.3.9 GetCameraInfo(CameraInfo *pCameraInfo)	173
8.24.3.10 GetConfiguration(FC2Config *pConfig)	173

8.24.3.11 GetCycleTime(TimeStamp *timeStamp)	173
8.24.3.12 GetEmbeddedImageInfo(EmbeddedImageInfo *pInfo)	174
8.24.3.13 GetGigEConfig(GigEConfig *pGigEConfig)	174
8.24.3.14 GetGigEImageBinningSettings(unsigned int *horzBinningValue, unsigned int *vertBinningValue)	174
8.24.3.15 GetGigEImageSettings(GigEImageSettings *pImageSettings)	175
8.24.3.16 GetGigEImageSettingsInfo(GigEImageSettingsInfo *pInfo)	175
8.24.3.17 GetGigEImagingMode(Mode *mode)	175
8.24.3.18 GetGigEProperty(GigEProperty *pGigEProp)	176
8.24.3.19 GetGigEStreamChannelInfo(unsigned int channel, GigEStreamChannel *p↵ Channel)	176
8.24.3.20 GetGPIOPinDirection(unsigned int pin, unsigned int *pDirection)	176
8.24.3.21 GetLUTBankInfo(unsigned int bank, bool *pReadSupported, bool *pWrite↵ Supported)	177
8.24.3.22 GetLUTChannel(unsigned int bank, unsigned int channel, unsigned int size↵ Entries, unsigned int *pEntries)	177
8.24.3.23 GetLUTInfo(LUTData *pData)	177
8.24.3.24 GetMemoryChannel(unsigned int *pCurrentChannel)	178
8.24.3.25 GetMemoryChannelInfo(unsigned int *pNumChannels)	178
8.24.3.26 GetNumStreamChannels(unsigned int *numChannels)	179
8.24.3.27 GetProperty(Property *pProp)	179
8.24.3.28 GetPropertyInfo(PropertyInfo *pPropInfo)	180
8.24.3.29 GetRegisterString(unsigned int registerVal)	181
8.24.3.30 GetStats(CameraStats *pStats)	181
8.24.3.31 GetStrobe(StrobeControl *pStrobeControl)	181
8.24.3.32 GetStrobeInfo(StrobeInfo *pStrobeInfo)	181
8.24.3.33 GetTriggerDelay(TriggerDelay *pTriggerDelay)	182
8.24.3.34 GetTriggerDelayInfo(TriggerDelayInfo *pTriggerDelayInfo)	182
8.24.3.35 GetTriggerMode(TriggerMode *pTriggerMode)	183
8.24.3.36 GetTriggerModeInfo(TriggerModeInfo *pTriggerModeInfo)	183
8.24.3.37 IsConnected()	184
8.24.3.38 QueryGigEImagingMode(Mode mode, bool *isSupported)	184

8.24.3.39 ReadGVCPMemory(unsigned int address, unsigned char *pBuffer, unsigned int length)	184
8.24.3.40 ReadGVCPRegister(unsigned int address, unsigned int *pValue)	185
8.24.3.41 ReadGVCPRegisterBlock(unsigned int address, unsigned int *pBuffer, unsigned int length)	185
8.24.3.42 ReadRegister(unsigned int address, unsigned int *pValue)	185
8.24.3.43 ReadRegisterBlock(unsigned short addressHigh, unsigned int addressLow, unsigned int *pBuffer, unsigned int length)	186
8.24.3.44 RegisterAllEvents(EventOptions *pOpts)	186
8.24.3.45 RegisterEvent(EventOptions *pOpts)	186
8.24.3.46 ResetStats()	186
8.24.3.47 RestoreFromMemoryChannel(unsigned int channel)	186
8.24.3.48 RetrieveBuffer(Image *pImage)	187
8.24.3.49 SaveToMemoryChannel(unsigned int channel)	187
8.24.3.50 SetActiveLUTBank(unsigned int activeBank)	188
8.24.3.51 SetCallback(ImageEventCallback callbackFn, const void *pCallbackData=NULL)	188
8.24.3.52 SetConfiguration(const FC2Config *pConfig)	189
8.24.3.53 SetEmbeddedImageInfo(EmbeddedImageInfo *pInfo)	190
8.24.3.54 SetGigEConfig(const GigEConfig *pGigEConfig)	190
8.24.3.55 SetGigEImageBinningSettings(unsigned int horzBinningValue, unsigned int vertBinningValue)	191
8.24.3.56 SetGigEImageSettings(const GigEImageSettings *pImageSettings)	191
8.24.3.57 SetGigEImagingMode(Mode mode)	191
8.24.3.58 SetGigEProperty(const GigEProperty *pGigEProp)	191
8.24.3.59 SetGigEStreamChannelInfo(unsigned int channel, GigEStreamChannel *p← Channel)	192
8.24.3.60 SetGPIOPinDirection(unsigned int pin, unsigned int direction, bool broadcast=false)	192
8.24.3.61 SetLUTChannel(unsigned int bank, unsigned int channel, unsigned int size← Entries, const unsigned int *pEntries)	193
8.24.3.62 SetProperty(const Property *pProp, bool broadcast=false)	193
8.24.3.63 SetStrobe(const StrobeControl *pStrobeControl, bool broadcast=false)	194
8.24.3.64 SetTriggerDelay(const TriggerDelay *pTriggerDelay, bool broadcast=false)	194
8.24.3.65 SetTriggerMode(const TriggerMode *pTriggerMode, bool broadcast=false)	194

8.24.3.66 SetUserBuffers(unsigned char *const pMemBuffers, int size, int numBuffers) . .	195
8.24.3.67 StartCapture(ImageEventCallback callbackFn=NULL, const void *pCallback↔ Data=NULL)	196
8.24.3.68 StartSyncCapture(unsigned int numCameras, const GigECamera **ppCameras, const ImageEventCallback *pCallbackFns=NULL, const void **pCallbackData↔ Array=NULL)	196
8.24.3.69 StopCapture()	196
8.24.3.70 WaitForBufferEvent(Image *pImage, unsigned int eventNumber)	196
8.24.3.71 WriteGVCPMemory(unsigned int address, const unsigned char *pBuffer, un- signed int length)	197
8.24.3.72 WriteGVCPRegister(unsigned int address, unsigned int value, bool broad- cast=false)	197
8.24.3.73 WriteGVCPRegisterBlock(unsigned int address, const unsigned int *pBuffer, un- signed int length)	198
8.24.3.74 WriteRegister(unsigned int address, unsigned int value, bool broadcast=false) . .	198
8.24.3.75 WriteRegisterBlock(unsigned short addressHigh, unsigned int addressLow, const unsigned int *pBuffer, unsigned int length)	198
8.25 GigEConfig Struct Reference	199
8.25.1 Detailed Description	199
8.25.2 Constructor & Destructor Documentation	199
8.25.2.1 GigEConfig()	199
8.25.3 Member Data Documentation	199
8.25.3.1 enablePacketResend	199
8.25.3.2 registerTimeout	200
8.25.3.3 registerTimeoutRetries	200
8.26 GigEImageSettings Struct Reference	200
8.26.1 Detailed Description	200
8.26.2 Constructor & Destructor Documentation	201
8.26.2.1 GigEImageSettings()	201
8.26.3 Member Data Documentation	201
8.26.3.1 height	201
8.26.3.2 offsetX	201
8.26.3.3 offsetY	201

8.26.3.4	pixelFormat	201
8.26.3.5	reserved	201
8.26.3.6	width	201
8.27	GigEImageSettingsInfo Struct Reference	201
8.27.1	Detailed Description	202
8.27.2	Constructor & Destructor Documentation	202
8.27.2.1	GigEImageSettingsInfo()	202
8.27.3	Member Data Documentation	202
8.27.3.1	imageHStepSize	202
8.27.3.2	imageVStepSize	202
8.27.3.3	maxHeight	202
8.27.3.4	maxWidth	202
8.27.3.5	offsetHStepSize	203
8.27.3.6	offsetVStepSize	203
8.27.3.7	pixelFormatBitField	203
8.27.3.8	reserved	203
8.27.3.9	vendorPixelFormatBitField	203
8.28	GigEProperty Struct Reference	203
8.28.1	Detailed Description	204
8.28.2	Member Data Documentation	204
8.28.2.1	isReadable	204
8.28.2.2	isWritable	204
8.28.2.3	max	204
8.28.2.4	min	204
8.28.2.5	propType	204
8.28.2.6	value	204
8.29	GigEStreamChannel Struct Reference	204
8.29.1	Detailed Description	205
8.29.2	Constructor & Destructor Documentation	205
8.29.2.1	GigEStreamChannel()	205

8.29.3	Member Data Documentation	205
8.29.3.1	destinationIpAddress	205
8.29.3.2	doNotFragment	205
8.29.3.3	hostPort	205
8.29.3.4	interPacketDelay	206
8.29.3.5	networkInterfaceIndex	206
8.29.3.6	packetSize	206
8.29.3.7	sourcePort	206
8.30	H264Option Struct Reference	206
8.30.1	Detailed Description	207
8.30.2	Constructor & Destructor Documentation	207
8.30.2.1	H264Option()	207
8.30.3	Member Data Documentation	207
8.30.3.1	bitrate	207
8.30.3.2	frameRate	207
8.30.3.3	height	207
8.30.3.4	reserved	207
8.30.3.5	width	207
8.31	Image Class Reference	207
8.31.1	Detailed Description	210
8.31.2	Constructor & Destructor Documentation	210
8.31.2.1	Image()	210
8.31.2.2	Image(unsigned int rows, unsigned int cols, unsigned int stride, unsigned char *pData, unsigned int dataSize, PixelFormat format, BayerTileFormat bayer← Format=NONE)	210
8.31.2.3	Image(unsigned int rows, unsigned int cols, unsigned int stride, unsigned char *pData, unsigned int dataSize, unsigned int receivedDataSize, PixelFormat for- mat, BayerTileFormat bayerFormat=NONE)	210
8.31.2.4	Image(unsigned char *pData, unsigned int dataSize)	211
8.31.2.5	Image(unsigned int rows, unsigned int cols, PixelFormat format, BayerTileFormat bayerFormat=NONE)	211
8.31.2.6	Image(const Image &image)	211
8.31.2.7	~Image()	211

8.31.3 Member Function Documentation	212
8.31.3.1 CalculateStatistics(ImageStatistics *pStatistics)	212
8.31.3.2 Convert(PixelFormat format, Image *pDestImage) const	212
8.31.3.3 Convert(Image *pDestImage) const	212
8.31.3.4 DeepCopy(const Image *pImage)	213
8.31.3.5 DetermineBitsPerPixel(PixelFormat format)	213
8.31.3.6 GetBayerTileFormat() const	213
8.31.3.7 GetBitsPerPixel() const	213
8.31.3.8 GetBlockId()	214
8.31.3.9 GetColorProcessing() const	214
8.31.3.10 GetCols() const	214
8.31.3.11 GetData()	214
8.31.3.12 GetData() const	214
8.31.3.13 GetDataSize() const	214
8.31.3.14 GetDefaultColorProcessing()	215
8.31.3.15 GetDefaultOutputFormat()	215
8.31.3.16 GetDimensions(unsigned int *pRows, unsigned int *pCols=NULL, unsigned int *pStride=NULL, PixelFormat *pPixelFormat=NULL, BayerTileFormat *pBayer↵ Format=NULL) const	215
8.31.3.17 GetMetadata() const	215
8.31.3.18 GetPixelFormat() const	216
8.31.3.19 GetReceivedDataSize() const	216
8.31.3.20 GetRows() const	216
8.31.3.21 GetStride() const	216
8.31.3.22 GetTimeStamp() const	216
8.31.3.23 operator()(unsigned int row, unsigned int col)	216
8.31.3.24 operator=(const Image &image)	217
8.31.3.25 operator[](unsigned int index)	217
8.31.3.26 ReleaseBuffer()	217
8.31.3.27 Save(const char *pFilename, ImageFileFormat format=FROM_FILE_EXT)	217
8.31.3.28 Save(const char *pFilename, PNGOption *pOption)	218

8.31.3.29 Save(const char *pFilename, PPMOption *pOption)	218
8.31.3.30 Save(const char *pFilename, PGMOption *pOption)	218
8.31.3.31 Save(const char *pFilename, TIFFOption *pOption)	219
8.31.3.32 Save(const char *pFilename, JPEGOption *pOption)	219
8.31.3.33 Save(const char *pFilename, JPG2Option *pOption)	219
8.31.3.34 Save(const char *pFilename, BMPOption *pOption)	219
8.31.3.35 SetBlockId(const unsigned int blockId)	220
8.31.3.36 SetColorProcessing(ColorProcessingAlgorithm colorProc)	220
8.31.3.37 SetData(const unsigned char *pData, unsigned int dataSize)	220
8.31.3.38 SetDefaultColorProcessing(ColorProcessingAlgorithm defaultMethod)	221
8.31.3.39 SetDefaultOutputFormat(PixelFormat format)	221
8.31.3.40 SetDimensions(unsigned int rows, unsigned int cols, unsigned int stride, Pixel↔ Format pixelFormat, BayerTileFormat bayerFormat)	221
8.31.4 Friends And Related Function Documentation	222
8.31.4.1 Iso	222
8.32 ImageMetadata Struct Reference	222
8.32.1 Detailed Description	223
8.32.2 Constructor & Destructor Documentation	223
8.32.2.1 ImageMetadata()	223
8.32.3 Member Data Documentation	223
8.32.3.1 embeddedBrightness	223
8.32.3.2 embeddedExposure	223
8.32.3.3 embeddedFrameCounter	223
8.32.3.4 embeddedGain	223
8.32.3.5 embeddedGPIOPinState	223
8.32.3.6 embeddedROIPosition	223
8.32.3.7 embeddedShutter	223
8.32.3.8 embeddedStrobePattern	223
8.32.3.9 embeddedTimeStamp	224
8.32.3.10 embeddedWhiteBalance	224
8.32.3.11 reserved	224

8.33 ImageStatistics Class Reference	224
8.33.1 Detailed Description	225
8.33.2 Member Enumeration Documentation	225
8.33.2.1 StatisticsChannel	225
8.33.3 Constructor & Destructor Documentation	226
8.33.3.1 ImageStatistics()	226
8.33.3.2 ~ImageStatistics()	226
8.33.3.3 ImageStatistics(const ImageStatistics &other)	226
8.33.4 Member Function Documentation	226
8.33.4.1 DisableAll()	226
8.33.4.2 EnableAll()	226
8.33.4.3 EnableGreyOnly()	226
8.33.4.4 EnableHSLOnly()	227
8.33.4.5 EnableRGBOnly()	227
8.33.4.6 GetChannelStatus(StatisticsChannel channel, bool *pEnabled) const	227
8.33.4.7 GetHistogram(StatisticsChannel channel, int **ppHistogram) const	227
8.33.4.8 GetMean(StatisticsChannel channel, float *pPixelValueMean) const	228
8.33.4.9 GetNumPixelValues(StatisticsChannel channel, unsigned int *pNumPixelValues) const	228
8.33.4.10 GetPixelValueRange(StatisticsChannel channel, unsigned int *pPixelValueMin, unsigned int *pPixelValueMax) const	228
8.33.4.11 GetRange(StatisticsChannel channel, unsigned int *pMin, unsigned int *pMax) const	228
8.33.4.12 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	229
8.33.4.13 operator=(const ImageStatistics &other)	229
8.33.4.14 SetChannelStatus(StatisticsChannel channel, bool enabled)	229
8.33.5 Friends And Related Function Documentation	230
8.33.5.1 ImageStatsCalculator	230
8.34 Internal Class Reference	230
8.34.1 Member Function Documentation	230

8.34.1.1	GetInternal(unsigned int index)	230
8.35	IPAddress Struct Reference	230
8.35.1	Detailed Description	231
8.35.2	Constructor & Destructor Documentation	231
8.35.2.1	IPAddress()	231
8.35.2.2	IPAddress(unsigned int ipAddressVal)	231
8.35.3	Member Function Documentation	231
8.35.3.1	operator!=(const IPAddress &address)	231
8.35.3.2	operator==(const IPAddress &address) const	231
8.35.4	Member Data Documentation	231
8.35.4.1	octets	231
8.36	JPEGOption Struct Reference	231
8.36.1	Detailed Description	232
8.36.2	Constructor & Destructor Documentation	232
8.36.2.1	JPEGOption()	232
8.36.3	Member Data Documentation	232
8.36.3.1	progressive	232
8.36.3.2	quality	232
8.36.3.3	reserved	232
8.37	JPG2Option Struct Reference	233
8.37.1	Detailed Description	233
8.37.2	Constructor & Destructor Documentation	233
8.37.2.1	JPG2Option()	233
8.37.3	Member Data Documentation	233
8.37.3.1	quality	233
8.37.3.2	reserved	233
8.38	LUTData Struct Reference	233
8.38.1	Detailed Description	234
8.38.2	Constructor & Destructor Documentation	234
8.38.2.1	LUTData()	234

8.38.3	Member Data Documentation	234
8.38.3.1	enabled	234
8.38.3.2	inputBitDepth	234
8.38.3.3	numBanks	234
8.38.3.4	numChannels	235
8.38.3.5	numEntries	235
8.38.3.6	outputBitDepth	235
8.38.3.7	reserved	235
8.38.3.8	supported	235
8.39	MACAddress Struct Reference	235
8.39.1	Detailed Description	236
8.39.2	Constructor & Destructor Documentation	236
8.39.2.1	MACAddress()	236
8.39.2.2	MACAddress(unsigned int macAddressValHigh, unsigned int macAddressValLow)	236
8.39.3	Member Function Documentation	236
8.39.3.1	operator!=(const MACAddress &address)	236
8.39.3.2	operator==(const MACAddress &address) const	236
8.39.4	Member Data Documentation	236
8.39.4.1	octets	236
8.40	MJPGOption Struct Reference	236
8.40.1	Detailed Description	237
8.40.2	Constructor & Destructor Documentation	237
8.40.2.1	MJPGOption()	237
8.40.3	Member Data Documentation	237
8.40.3.1	frameRate	237
8.40.3.2	quality	237
8.40.3.3	reserved	237
8.41	NodeMap Class Reference	237
8.41.1	Constructor & Destructor Documentation	238
8.41.1.1	NodeMap(GenApi::CNodeMapRef *ref)	238

8.41.1.2	<code>~NodeMap(void)</code>	238
8.41.2	Member Function Documentation	238
8.41.2.1	<code>_GetDeviceName()</code>	238
8.41.2.2	<code>_GetNode(const GenICam::gcstring &key)</code>	238
8.41.2.3	<code>_GetNodes(NodeList_t &Nodes)</code>	238
8.41.2.4	<code>_InvalidateNodes() const</code>	238
8.41.2.5	<code>_Poll(int64_t ElapsedTime)</code>	238
8.42	PGMOption Struct Reference	238
8.42.1	Detailed Description	239
8.42.2	Constructor & Destructor Documentation	239
8.42.2.1	<code>PGMOption()</code>	239
8.42.3	Member Data Documentation	239
8.42.3.1	<code>binaryFile</code>	239
8.42.3.2	<code>reserved</code>	239
8.43	PGRGuid Class Reference	239
8.43.1	Detailed Description	240
8.43.2	Constructor & Destructor Documentation	240
8.43.2.1	<code>PGRGuid()</code>	240
8.43.3	Member Function Documentation	240
8.43.3.1	<code>operator!=(const PGRGuid &guid)</code>	240
8.43.3.2	<code>operator==(const PGRGuid &guid) const</code>	240
8.43.4	Member Data Documentation	240
8.43.4.1	<code>value</code>	240
8.44	PNGOption Struct Reference	240
8.44.1	Detailed Description	241
8.44.2	Constructor & Destructor Documentation	241
8.44.2.1	<code>PNGOption()</code>	241
8.44.3	Member Data Documentation	241
8.44.3.1	<code>compressionLevel</code>	241
8.44.3.2	<code>interlaced</code>	241

8.44.3.3 reserved	241
8.45 PPMOption Struct Reference	241
8.45.1 Detailed Description	242
8.45.2 Constructor & Destructor Documentation	242
8.45.2.1 PPMOption()	242
8.45.3 Member Data Documentation	242
8.45.3.1 binaryFile	242
8.45.3.2 reserved	242
8.46 Property Struct Reference	242
8.46.1 Detailed Description	243
8.46.2 Constructor & Destructor Documentation	243
8.46.2.1 Property()	243
8.46.2.2 Property(PropertyType propType)	243
8.46.3 Member Data Documentation	243
8.46.3.1 absControl	243
8.46.3.2 absValue	244
8.46.3.3 autoManualMode	244
8.46.3.4 onePush	244
8.46.3.5 onOff	244
8.46.3.6 present	244
8.46.3.7 reserved	244
8.46.3.8 type	244
8.46.3.9 valueA	244
8.46.3.10 valueB	244
8.47 PropertyInfo Struct Reference	245
8.47.1 Detailed Description	245
8.47.2 Constructor & Destructor Documentation	246
8.47.2.1 PropertyInfo()	246
8.47.2.2 PropertyInfo(PropertyType propType)	246
8.47.3 Member Data Documentation	246

8.47.3.1	absMax	246
8.47.3.2	absMin	246
8.47.3.3	absValSupported	246
8.47.3.4	autoSupported	246
8.47.3.5	manualSupported	246
8.47.3.6	max	246
8.47.3.7	min	246
8.47.3.8	onePushSupported	246
8.47.3.9	onOffSupported	247
8.47.3.10	present	247
8.47.3.11	pUnitAbbr	247
8.47.3.12	pUnits	247
8.47.3.13	readOutSupported	247
8.47.3.14	reserved	247
8.47.3.15	type	247
8.48	StrobeControl Struct Reference	247
8.48.1	Detailed Description	248
8.48.2	Constructor & Destructor Documentation	248
8.48.2.1	StrobeControl()	248
8.48.3	Member Data Documentation	248
8.48.3.1	delay	248
8.48.3.2	duration	248
8.48.3.3	onOff	248
8.48.3.4	polarity	248
8.48.3.5	reserved	248
8.48.3.6	source	249
8.49	StrobeInfo Struct Reference	249
8.49.1	Detailed Description	249
8.49.2	Constructor & Destructor Documentation	249
8.49.2.1	StrobeInfo()	249

8.49.3	Member Data Documentation	249
8.49.3.1	maxValue	249
8.49.3.2	minValue	250
8.49.3.3	onOffSupported	250
8.49.3.4	polaritySupported	250
8.49.3.5	present	250
8.49.3.6	readOutSupported	250
8.49.3.7	reserved	250
8.49.3.8	source	250
8.50	SyncManager Class Reference	250
8.50.1	Constructor & Destructor Documentation	251
8.50.1.1	SyncManager()	251
8.50.1.2	~SyncManager()	251
8.50.2	Member Function Documentation	251
8.50.2.1	DisableCrossPCsynchronization()	251
8.50.2.2	EnableCrossPCsynchronization()	251
8.50.2.3	GetSyncStatus()	251
8.50.2.4	GetTimeSinceSynced()	251
8.50.2.5	IsTimingBusConnected()	251
8.50.2.6	QueryCrossPCsynchronizationSetting()	251
8.50.2.7	RescanMasterTimingBus()	251
8.50.2.8	Start()	251
8.50.2.9	Stop()	251
8.51	SystemInfo Struct Reference	251
8.51.1	Detailed Description	252
8.51.2	Member Data Documentation	252
8.51.2.1	byteOrder	252
8.51.2.2	cpuDescription	252
8.51.2.3	driverList	252
8.51.2.4	gpuDescription	253

8.51.2.5	libraryList	253
8.51.2.6	numCpuCores	253
8.51.2.7	osDescription	253
8.51.2.8	osType	253
8.51.2.9	reserved	253
8.51.2.10	screenHeight	253
8.51.2.11	screenWidth	253
8.51.2.12	sysMemSize	253
8.52	TIFFOption Struct Reference	254
8.52.1	Detailed Description	254
8.52.2	Member Enumeration Documentation	254
8.52.2.1	CompressionMethod	254
8.52.3	Constructor & Destructor Documentation	255
8.52.3.1	TIFFOption()	255
8.52.4	Member Data Documentation	255
8.52.4.1	compression	255
8.52.4.2	reserved	255
8.53	TimeStamp Struct Reference	255
8.53.1	Detailed Description	255
8.53.2	Constructor & Destructor Documentation	256
8.53.2.1	TimeStamp()	256
8.53.3	Member Data Documentation	256
8.53.3.1	cycleCount	256
8.53.3.2	cycleOffset	256
8.53.3.3	cycleSeconds	256
8.53.3.4	microSeconds	256
8.53.3.5	reserved	256
8.53.3.6	seconds	256
8.54	TopologyNode Class Reference	256
8.54.1	Detailed Description	258

8.54.2	Member Enumeration Documentation	258
8.54.2.1	NodeType	258
8.54.2.2	PortType	258
8.54.3	Constructor & Destructor Documentation	258
8.54.3.1	TopologyNode()	258
8.54.3.2	TopologyNode(PGRGuid guid, int deviceId, NodeType nodeType, InterfaceType interfaceType)	258
8.54.3.3	~TopologyNode()	259
8.54.3.4	TopologyNode(const TopologyNode &other)	259
8.54.4	Member Function Documentation	259
8.54.4.1	AddChild(TopologyNode childNode)	259
8.54.4.2	AddPortType(PortType childPort)	259
8.54.4.3	AssignGuidToNode(PGRGuid guid, int deviceId)	259
8.54.4.4	AssignGuidToNode(PGRGuid guid, int deviceId, NodeType nodeType)	259
8.54.4.5	GetChild(unsigned int position)	260
8.54.4.6	GetDeviceld()	260
8.54.4.7	GetGuid()	260
8.54.4.8	GetInterfaceType()	260
8.54.4.9	GetNodeType()	261
8.54.4.10	GetNumChildren()	261
8.54.4.11	GetNumPorts()	261
8.54.4.12	GetPortType(unsigned int position)	261
8.54.4.13	operator=(const TopologyNode &other)	261
8.55	TriggerMode Struct Reference	262
8.55.1	Detailed Description	262
8.55.2	Constructor & Destructor Documentation	262
8.55.2.1	TriggerMode()	262
8.55.3	Member Data Documentation	262
8.55.3.1	mode	262
8.55.3.2	onOff	262
8.55.3.3	parameter	263

8.55.3.4	polarity	263
8.55.3.5	reserved	263
8.55.3.6	source	263
8.56	TriggerModelInfo Struct Reference	263
8.56.1	Detailed Description	264
8.56.2	Constructor & Destructor Documentation	264
8.56.2.1	TriggerModelInfo()	264
8.56.3	Member Data Documentation	264
8.56.3.1	modeMask	264
8.56.3.2	onOffSupported	264
8.56.3.3	polaritySupported	264
8.56.3.4	present	264
8.56.3.5	readOutSupported	264
8.56.3.6	reserved	264
8.56.3.7	softwareTriggerSupported	264
8.56.3.8	sourceMask	264
8.56.3.9	valueReadable	265
8.57	Utilities Class Reference	265
8.57.1	Detailed Description	265
8.57.2	Member Function Documentation	265
8.57.2.1	CheckDriver(const PGRGuid *guid)	265
8.57.2.2	GetDriverDeviceName(const PGRGuid *guid, std::string &deviceName)	266
8.57.2.3	GetLibraryVersion(FC2Version *pVersion)	266
8.57.2.4	GetSystemInfo(SystemInfo *pSystemInfo)	266
8.57.2.5	LaunchBrowser(const char *pAddress)	267
8.57.2.6	LaunchCommand(const char *pCommand)	267
8.57.2.7	LaunchCommandAsync(const char *pCommand, AsyncCommandCallback p← Callback, void *pUserData)	267
8.57.2.8	LaunchHelp(const char *pFileName)	268

9 File Documentation	269
9.1 AVIRecorder.h File Reference	269
9.2 BusManager.h File Reference	269
9.3 Camera.h File Reference	269
9.4 CameraBase.h File Reference	270
9.5 Error.h File Reference	270
9.6 FlyCapture2.h File Reference	271
9.7 FlyCapture2Defs.h File Reference	271
9.7.1 Macro Definition Documentation	278
9.7.1.1 FULL_32BIT_VALUE	278
9.7.1.2 NULL	278
9.8 FlyCapture2GUI.h File Reference	278
9.9 FlyCapture2Platform.h File Reference	279
9.9.1 Macro Definition Documentation	279
9.9.1.1 FLYCAPTURE2_API	279
9.9.1.2 FLYCAPTURE2_LOCAL	279
9.10 FlyCapture3ApiGuiWrapper.h File Reference	279
9.10.1 Macro Definition Documentation	279
9.10.1.1 WRAPPER_API	279
9.11 GCCamera.h File Reference	279
9.12 GigECamera.h File Reference	280
9.13 Image.h File Reference	280
9.14 ImageStatistics.h File Reference	280
9.15 Internal.h File Reference	281
9.16 MultiSyncLibrary.h File Reference	281
9.17 MultiSyncLibraryDefs.h File Reference	281
9.18 MultiSyncLibraryPlatform.h File Reference	282
9.18.1 Macro Definition Documentation	282
9.18.1.1 MULTISYNCLIBRARY_API	282
9.18.1.2 MULTISYNCLIBRARY_LOCAL	282
9.19 NodeMap.h File Reference	282
9.20 TopologyNode.h File Reference	282
9.21 Utilities.h File Reference	283
Index	285

Chapter 1

Module Index

1.1 Modules

Here is a list of all modules:

Global constants	13
Enumerations	14
GigE specific enumerations	28
Structures	29
GigE specific structures	31
IIDC specific structures	32
Image saving structures.	33

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all namespaces with brief descriptions:

FlyCap3CameraControl	35
FlyCapture2	35
MultiSyncLibrary	45

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

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

AVIOption	47
AVIRecorder	48
BMPOption	50
BusManager	51
CameraBase	88
Camera	60
GCCamera	144
GigECamera	167
CameraControlDlg	112
CameraInfo	115
CameraSelectionDlg	120
CameraStats	121
ConfigROM	123
EmbeddedImageInfo	125
EmbeddedImageInfoProperty	126
Error	127
EventCallbackData	131
EventOptions	132
FC2Config	133
FC2Version	136
FlyCapture3ApiGuiWrapper	138
Format7ImageSettings	139
Format7Info	140
Format7PacketInfo	143
GigEConfig	199
GigEImageSettings	200
GigEImageSettingsInfo	201
GigEProperty	203
GigEStreamChannel	204
H264Option	206
Image	207
ImageMetadata	222
ImageStatistics	224
Internal	230
IPAddress	230

JPEGOption	231
JPG2Option	233
LUTData	233
MACAddress	235
MJPGOption	236
NodeMap	237
PGMOption	238
PGRGuid	239
PNGOption	240
PPMOption	241
Property	242
PropertyInfo	245
StrobeControl	247
StrobeInfo	249
SyncManager	250
SystemInfo	251
TIFFOption	254
TimeStamp	255
TopologyNode	256
TriggerMode	262
TriggerModelInfo	263
Utilities	265

Chapter 4

Class Index

4.1 Class List

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

AVIOption	Options for saving AVI files	47
AVIRecorder	Functionality for the user to record images to an AVI file	48
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	60
CameraBase	Abstract base class that defines a general interface to a camera	88
CameraControlDlg	The CameraControlDlg object represents a dialog that provides a graphical interface to a specified camera	112
CameraInfo	Camera information	115
CameraSelectionDlg	The CameraSelectionDlg object represents a dialog that provides a graphical interface that lists the number of cameras available to the library	120
CameraStats	Camera diagnostic information	121
ConfigROM	Camera configuration ROM	123
EmbeddedImageInfo	Properties of the possible embedded image information	125
EmbeddedImageInfoProperty	Properties of a single embedded image info property	126
Error	The Error object represents an error that is returned from the library	127
EventCallbackData	131
EventOptions	Options for enabling device event registration	132
FC2Config	Configuration for a camera	133

FC2Version	
The current version of the library	136
FlyCapture3ApiGuiWrapper	138
Format7ImageSettings	
Format 7 image settings	139
Format7Info	
Format 7 information for a single mode	140
Format7PacketInfo	
Format 7 packet information	143
GCCamera	144
GigECamera	
The GigECamera object represents a physical Gigabit Ethernet camera	167
GigEConfig	
Configuration for a GigE camera	199
GigEImageSettings	
Image settings for a GigE camera	200
GigEImageSettingsInfo	
Format 7 information for a single mode	201
GigEProperty	
A GigE property	203
GigEStreamChannel	
Information about a single GigE stream channel	204
H264Option	
Options for saving H264 files	206
Image	
Used to retrieve images from a camera, convert between multiple pixel formats and save images to disk	207
ImageMetadata	
Metadata related to an image	222
ImageStatistics	
The ImageStatistics object represents image statistics for an image	224
Internal	230
IPAddress	
IPv4 address	230
JPEGOption	
Options for saving JPEG image	231
JPG2Option	
Options for saving JPEG2000 image	233
LUTData	
Information about the camera's look up table	233
MACAddress	
MAC address	235
MJPGOption	
Options for saving MJPG files	236
NodeMap	237
PGMOption	
Options for saving PGM images	238
PGRGuid	
A GUID to the camera	239
PNGOption	
Options for saving PNG images	240
PPMOption	
Options for saving PPM images	241
Property	
A specific camera property	242
PropertyInfo	
Information about a specific camera property	245

StrobeControl	
A camera strobe	247
StrobeInfo	
A camera strobe property	249
SyncManager	250
SystemInfo	
Description of the system	251
TIFFOption	
Options for saving TIFF images	254
TimeStamp	
Timestamp information	255
TopologyNode	
Topology information that can be used to generate a tree structure of all cameras and devices connected to a computer	256
TriggerMode	
A camera trigger	262
TriggerModelInfo	
Information about a camera trigger property	263
Utilities	
The Utility class is generally used to query for general system information such as operating system, available memory etc	265

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

AVIRecorder.h	269
BusManager.h	269
Camera.h	269
CameraBase.h	270
Error.h	270
FlyCapture2.h	271
FlyCapture2Defs.h	271
FlyCapture2GUI.h	278
FlyCapture2Platform.h	279
FlyCapture3ApiGuiWrapper.h	279
GCCamera.h	279
GigECamera.h	280
Image.h	280
ImageStatistics.h	280
Internal.h	281
MultiSyncLibrary.h	281
MultiSyncLibraryDefs.h	281
MultiSyncLibraryPlatform.h	282
NodeMap.h	282
TopologyNode.h	282
Utilities.h	283

Chapter 6

Module Documentation

6.1 Global constants

Variables

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

6.1.1 Detailed Description

6.1.2 Variable Documentation

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

The maximum number of ports one device can have.

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

The maximum length that is allocated for a string.

6.2 Enumerations

Enumerations

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

The error types returned by functions.

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

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

- enum `GrabMode` {
 `DROP_FRAMES`,
 `BUFFER_FRAMES`,
 `UNSPECIFIED_GRAB_MODE`,
 `GRAB_MODE_FORCE_32BITS` = `FULL_32BIT_VALUE` }

The grab strategy employed during image transfer.

- enum `GrabTimeout` {
 `TIMEOUT_NONE` = 0,
 `TIMEOUT_INFINITE` = -1,
 `TIMEOUT_UNSPECIFIED` = -2,
 `GRAB_TIMEOUT_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Timeout options for grabbing images.

- enum `BandwidthAllocation` {
 `BANDWIDTH_ALLOCATION_OFF` = 0,
 `BANDWIDTH_ALLOCATION_ON` = 1,
 `BANDWIDTH_ALLOCATION_UNSUPPORTED` = 2,
 `BANDWIDTH_ALLOCATION_UNSPECIFIED` = 3,
 `BANDWIDTH_ALLOCATION_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Bandwidth allocation options for 1394 devices.

- enum `InterfaceType` {
 `INTERFACE_IEEE1394`,
 `INTERFACE_USB2`,
 `INTERFACE_USB3`,
 `INTERFACE_GIGE`,
 `INTERFACE_UNKNOWN`,
 `INTERFACE_TYPE_FORCE_32BITS` = `FULL_32BIT_VALUE` }

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

- enum `PropertyType` {
 `BRIGHTNESS`,
 `AUTO_EXPOSURE`,
 `SHARPNESS`,
 `WHITE_BALANCE`,
 `HUE`,
 `SATURATION`,
 `GAMMA`,
 `IRIS`,
 `FOCUS`,
 `ZOOM`,
 `PAN`,
 `TILT`,
 `SHUTTER`,
 `GAIN`,
 `TRIGGER_MODE`,
 `TRIGGER_DELAY`,
 `FRAME_RATE`,
 `TEMPERATURE`,
 `UNSPECIFIED_PROPERTY_TYPE`,
 `PROPERTY_TYPE_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Camera properties.

- enum `FrameRate` {

```

FRAMERATE_1_875,
FRAMERATE_3_75,
FRAMERATE_7_5,
FRAMERATE_15,
FRAMERATE_30,
FRAMERATE_60,
FRAMERATE_120,
FRAMERATE_240,
FRAMERATE_FORMAT7,
NUM_FRAMERATES,
FRAMERATE_FORCE_32BITS = FULL_32BIT_VALUE }

```

Frame rates in frames per second.

- enum `VideoMode` {


```

VIDEOMODE_160x120YUV444,
VIDEOMODE_320x240YUV422,
VIDEOMODE_640x480YUV411,
VIDEOMODE_640x480YUV422,
VIDEOMODE_640x480RGB,
VIDEOMODE_640x480Y8,
VIDEOMODE_640x480Y16,
VIDEOMODE_800x600YUV422,
VIDEOMODE_800x600RGB,
VIDEOMODE_800x600Y8,
VIDEOMODE_800x600Y16,
VIDEOMODE_1024x768YUV422,
VIDEOMODE_1024x768RGB,
VIDEOMODE_1024x768Y8,
VIDEOMODE_1024x768Y16,
VIDEOMODE_1280x960YUV422,
VIDEOMODE_1280x960RGB,
VIDEOMODE_1280x960Y8,
VIDEOMODE_1280x960Y16,
VIDEOMODE_1600x1200YUV422,
VIDEOMODE_1600x1200RGB,
VIDEOMODE_1600x1200Y8,
VIDEOMODE_1600x1200Y16,
VIDEOMODE_FORMAT7,
NUM_VIDEOMODES,
VIDEOMODE_FORCE_32BITS = FULL_32BIT_VALUE }

```

DCAM video modes.

- enum `Mode` {


```

MODE_0 = 0,
MODE_1,
MODE_2,
MODE_3,
MODE_4,
MODE_5,
MODE_6,
MODE_7,
MODE_8,
MODE_9,
MODE_10,
MODE_11,
MODE_12,
MODE_13,
MODE_14,
MODE_15,
MODE_16,
MODE_17,
MODE_18,
MODE_19,
MODE_20,
MODE_21,
MODE_22,
MODE_23,
MODE_24,
MODE_25,
MODE_26,
MODE_27,
MODE_28,
MODE_29,
MODE_30,
MODE_31,
NUM_MODES,
MODE_FORCE_32BITS = FULL_32BIT_VALUE }

```

Camera modes for DCAM formats as well as Format7.

- enum PixelFormat {


```

PIXEL_FORMAT_MONO8 = 0x80000000,
PIXEL_FORMAT_411YUV8 = 0x40000000,
PIXEL_FORMAT_422YUV8 = 0x20000000,
PIXEL_FORMAT_444YUV8 = 0x10000000,
PIXEL_FORMAT_RGB8 = 0x08000000,
PIXEL_FORMAT_MONO16 = 0x04000000,
PIXEL_FORMAT_RGB16 = 0x02000000,
PIXEL_FORMAT_S_MONO16 = 0x01000000,
PIXEL_FORMAT_S_RGB16 = 0x00800000,
PIXEL_FORMAT_RAW8 = 0x00400000,
PIXEL_FORMAT_RAW16 = 0x00200000,
PIXEL_FORMAT_MONO12 = 0x00100000,
PIXEL_FORMAT_RAW12 = 0x00080000,
PIXEL_FORMAT_BGR = 0x80000008,
PIXEL_FORMAT_BGRU = 0x40000008,
PIXEL_FORMAT_RGB = PIXEL_FORMAT_RGB8,
PIXEL_FORMAT_RBGU = 0x40000002,
PIXEL_FORMAT_BGR16 = 0x02000001,
PIXEL_FORMAT_BGRU16 = 0x02000002,
PIXEL_FORMAT_422YUV8_JPEG = 0x40000001,
NUM_PIXEL_FORMATS = 20,
UNSPECIFIED_PIXEL_FORMAT = 0 }

```

Pixel formats available for Format7 modes.

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

Bus speeds.

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

Types of low level drivers that flycapture uses.

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

Color processing algorithms.

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

Bayer tile formats.

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

File formats to be used for saving images to disk.

6.2.1 Detailed Description

6.2.2 Enumeration Type Documentation

6.2.2.1 enum BandwidthAllocation

Bandwidth allocation options for 1394 devices.

Enumerator

BANDWIDTH_ALLOCATION_OFF Do not allocate bandwidth.

BANDWIDTH_ALLOCATION_ON Allocate bandwidth. This is the default setting.

BANDWIDTH_ALLOCATION_UNSUPPORTED Bandwidth allocation is not supported by either the camera or operating system.

BANDWIDTH_ALLOCATION_UNSPECIFIED Not specified. This leaves the current setting unchanged.

BANDWIDTH_ALLOCATION_FORCE_32BITS

6.2.2.2 enum BayerTileFormat

Bayer tile formats.

Enumerator

NONE No bayer tile format.

RGGB Red-Green-Green-Blue.

GRBG Green-Red-Blue-Green.

GBRG Green-Blue-Red-Green.

BGGR Blue-Green-Green-Red.

BT_FORCE_32BITS

6.2.2.3 enum BusCallbackType

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

Enumerator

BUS_RESET Register for all bus events.
ARRIVAL Register for arrivals only.
REMOVAL Register for removals only.
CALLBACK_TYPE_FORCE_32BITS

6.2.2.4 enum BusSpeed

Bus speeds.

Enumerator

BUSSPEED_S100 100Mbps/sec.
BUSSPEED_S200 200Mbps/sec.
BUSSPEED_S400 400Mbps/sec.
BUSSPEED_S480 480Mbps/sec. Only for 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

6.2.2.5 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.
COLOR_PROCESSING_ALGORITHM_FORCE_32BITS

6.2.2.6 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 PGRUsbCam.sys.
DRIVER_USB3_PRO PGRXHCl.sys.
DRIVER_GIGE_NONE no gige drivers used,MS/BSD stack.
DRIVER_GIGE_FILTER PGRGigE.sys.
DRIVER_GIGE_PRO PGRGigEPro.sys.
DRIVER_GIGE_LWF PgrLwf.sys.
DRIVER_UNKNOWN Unknown driver type.
DRIVER_FORCE_32BITS

6.2.2.7 enum ErrorType

The error types returned by functions.

Enumerator

PGRERROR_UNDEFINED Undefined.
PGRERROR_OK Function returned with no errors.
PGRERROR_FAILED General failure.
PGRERROR_NOT_IMPLEMENTED Function has not been implemented.
PGRERROR_FAILED_BUS_MASTER_CONNECTION Could not connect to Bus Master.
PGRERROR_NOT_CONNECTED [Camera](#) has not been connected.
PGRERROR_INIT_FAILED Initialization failed.
PGRERROR_NOT_INITIALIZED [Camera](#) has not been initialized.
PGRERROR_INVALID_PARAMETER Invalid parameter passed to function.
PGRERROR_INVALID_SETTINGS Setting set to camera is invalid.
PGRERROR_INVALID_BUS_MANAGER Invalid Bus Manager object.
PGRERROR_MEMORY_ALLOCATION_FAILED Could not allocate memory.
PGRERROR_LOW_LEVEL_FAILURE Low level error.
PGRERROR_NOT_FOUND Device not found.
PGRERROR_FAILED_GUID GUID failure.
PGRERROR_INVALID_PACKET_SIZE Packet size set to camera is invalid.
PGRERROR_INVALID_MODE Invalid mode has been passed to function.
PGRERROR_NOT_IN_FORMAT7 [Error](#) due to not being in Format7.
PGRERROR_NOT_SUPPORTED This feature is unsupported.

PGRERROR_TIMEOUT Timeout error.

PGRERROR_BUS_MASTER_FAILED Bus Master Failure.

PGRERROR_INVALID_GENERATION Generation Count Mismatch.

PGRERROR_LUT_FAILED Look Up Table failure.

PGRERROR_IIDC_FAILED IIDC failure.

PGRERROR_STROBE_FAILED Strobe failure.

PGRERROR_TRIGGER_FAILED Trigger failure.

PGRERROR_PROPERTY_FAILED [Property](#) failure.

PGRERROR_PROPERTY_NOT_PRESENT [Property](#) is not present.

PGRERROR_REGISTER_FAILED Register access failed.

PGRERROR_READ_REGISTER_FAILED Register read failed.

PGRERROR_WRITE_REGISTER_FAILED Register write failed.

PGRERROR_ISOCH_FAILED Isochronous failure.

PGRERROR_ISOCH_ALREADY_STARTED Isochronous transfer has already been started.

PGRERROR_ISOCH_NOT_STARTED Isochronous transfer has not been started.

PGRERROR_ISOCH_START_FAILED Isochronous start failed.

PGRERROR_ISOCH_RETRIEVE_BUFFER_FAILED Isochronous retrieve buffer failed.

PGRERROR_ISOCH_STOP_FAILED Isochronous stop failed.

PGRERROR_ISOCH_SYNC_FAILED Isochronous image synchronization failed.

PGRERROR_ISOCH_BANDWIDTH_EXCEEDED Isochronous bandwidth exceeded.

PGRERROR_IMAGE_CONVERSION_FAILED [Image](#) conversion failed.

PGRERROR_IMAGE_LIBRARY_FAILURE [Image](#) library failure.

PGRERROR_BUFFER_TOO_SMALL Buffer is too small.

PGRERROR_IMAGE_CONSISTENCY_ERROR There is an image consistency error.

PGRERROR_INCOMPATIBLE_DRIVER The installed driver is not compatible with the library.

PGRERROR_FORCE_32BITS

6.2.2.8 enum FrameRate

Frame rates in frames per second.

Enumerator

FRAMERATE_1_875 1.875 fps.

FRAMERATE_3_75 3.75 fps.

FRAMERATE_7_5 7.5 fps.

FRAMERATE_15 15 fps.

FRAMERATE_30 30 fps.

FRAMERATE_60 60 fps.

FRAMERATE_120 120 fps.

FRAMERATE_240 240 fps.

FRAMERATE_FORMAT7 Custom frame rate for Format7 functionality.

NUM_FRAMERATES Number of possible camera frame rates.

FRAMERATE_FORCE_32BITS

6.2.2.9 enum GrabMode

The grab strategy employed during image transfer.

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

Enumerator

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

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

UNSPECIFIED_GRAB_MODE Unspecified grab mode.

GRAB_MODE_FORCE_32BITS

6.2.2.10 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

6.2.2.11 enum ImageFileFormat

File formats to be used for saving images to disk.

Enumerator

FROM_FILE_EXT Determine file format from file extension.

PGM Portable gray map.

PPM Portable pixmap.

BMP Bitmap.

JPEG JPEG.

JPEG2000 JPEG 2000.

TIFF Tagged image file format.

PNG Portable network graphics.

RAW Raw data.

IMAGE_FILE_FORMAT_FORCE_32BITS

6.2.2.12 enum InterfaceType

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

Enumerator

INTERFACE_IEEE1394 IEEE-1394 (Includes 1394a and 1394b).
INTERFACE_USB2 USB 2.0.
INTERFACE_USB3 USB 3.0.
INTERFACE_GIGE GigE.
INTERFACE_UNKNOWN Unknown interface.
INTERFACE_TYPE_FORCE_32BITS

6.2.2.13 enum Mode

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

Enumerator

MODE_0
MODE_1
MODE_2
MODE_3
MODE_4
MODE_5
MODE_6
MODE_7
MODE_8
MODE_9
MODE_10
MODE_11
MODE_12
MODE_13
MODE_14
MODE_15
MODE_16
MODE_17
MODE_18
MODE_19
MODE_20
MODE_21
MODE_22
MODE_23
MODE_24
MODE_25
MODE_26
MODE_27
MODE_28
MODE_29
MODE_30
MODE_31
NUM_MODES Number of modes.
MODE_FORCE_32BITS

6.2.2.14 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.

6.2.2.15 enum PixelFormat

Pixel formats available for Format7 modes.

Enumerator

PIXEL_FORMAT_MONO8 8 bits of mono information.
PIXEL_FORMAT_411YUV8 YUV 4:1:1.
PIXEL_FORMAT_422YUV8 YUV 4:2:2.
PIXEL_FORMAT_444YUV8 YUV 4:4:4.
PIXEL_FORMAT_RGB8 R = G = B = 8 bits.
PIXEL_FORMAT_MONO16 16 bits of mono information.
PIXEL_FORMAT_RGB16 R = G = B = 16 bits.
PIXEL_FORMAT_S_MONO16 16 bits of signed mono information.
PIXEL_FORMAT_S_RGB16 R = G = B = 16 bits signed.
PIXEL_FORMAT_RAW8 8 bit raw data output of sensor.
PIXEL_FORMAT_RAW16 16 bit raw data output of sensor.
PIXEL_FORMAT_MONO12 12 bits of mono information.
PIXEL_FORMAT_RAW12 12 bit raw data output of sensor.
PIXEL_FORMAT_BGR 24 bit BGR.
PIXEL_FORMAT_BGRU 32 bit BGRU.
PIXEL_FORMAT_RGB 24 bit RGB.
PIXEL_FORMAT_RGBU 32 bit RGBU.
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.

6.2.2.16 enum PropertyType

Camera properties.

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

Enumerator

BRIGHTNESS Brightness.
AUTO_EXPOSURE Auto exposure.
SHARPNESS Sharpness.
WHITE_BALANCE White balance.
HUE Hue.
SATURATION Saturation.
GAMMA Gamma.
IRIS Iris.
FOCUS Focus.
ZOOM Zoom.
PAN Pan.
TILT Tilt.
SHUTTER Shutter.
GAIN Gain.
TRIGGER_MODE Trigger mode.
TRIGGER_DELAY Trigger delay.
FRAME_RATE Frame rate.
TEMPERATURE Temperature.
UNSPECIFIED_PROPERTY_TYPE Unspecified property type.
PROPERTY_TYPE_FORCE_32BITS

6.2.2.17 enum VideoMode

DCAM video modes.

Enumerator

VIDEOMODE_160x120YUV444 160x120 YUV444.
VIDEOMODE_320x240YUV422 320x240 YUV422.
VIDEOMODE_640x480YUV411 640x480 YUV411.
VIDEOMODE_640x480YUV422 640x480 YUV422.
VIDEOMODE_640x480RGB 640x480 24-bit RGB.
VIDEOMODE_640x480Y8 640x480 8-bit.
VIDEOMODE_640x480Y16 640x480 16-bit.
VIDEOMODE_800x600YUV422 800x600 YUV422.
VIDEOMODE_800x600RGB 800x600 RGB.
VIDEOMODE_800x600Y8 800x600 8-bit.
VIDEOMODE_800x600Y16 800x600 16-bit.
VIDEOMODE_1024x768YUV422 1024x768 YUV422.

VIDEOMODE_1024x768RGB 1024x768 RGB.
VIDEOMODE_1024x768Y8 1024x768 8-bit.
VIDEOMODE_1024x768Y16 1024x768 16-bit.
VIDEOMODE_1280x960YUV422 1280x960 YUV422.
VIDEOMODE_1280x960RGB 1280x960 RGB.
VIDEOMODE_1280x960Y8 1280x960 8-bit.
VIDEOMODE_1280x960Y16 1280x960 16-bit.
VIDEOMODE_1600x1200YUV422 1600x1200 YUV422.
VIDEOMODE_1600x1200RGB 1600x1200 RGB.
VIDEOMODE_1600x1200Y8 1600x1200 8-bit.
VIDEOMODE_1600x1200Y16 1600x1200 16-bit.
VIDEOMODE_FORMAT7 Custom video mode for Format7 functionality.
NUM_VIDEOMODES Number of possible video modes.
VIDEOMODE_FORCE_32BITS

6.3 GigE specific enumerations

These enumerations are specific to GigE camera operation only.

Enumerations

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

Possible properties that can be queried from the camera.

6.3.1 Detailed Description

These enumerations are specific to GigE camera operation only.

6.3.2 Enumeration Type Documentation

6.3.2.1 enum `GigEPropertyType`

Possible properties that can be queried from the camera.

Enumerator

`HEARTBEAT`

`HEARTBEAT_TIMEOUT`

`PACKET_SIZE`

`PACKET_DELAY`

6.4 Structures

Collaboration diagram for Structures:

Modules

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

Classes

- struct [FC2Version](#)
The current version of the library.
- class [PGRGuid](#)
A GUID to the camera.
- struct [IPAddress](#)
IPv4 address.
- struct [Format7ImageSettings](#)
Format 7 image settings.
- struct [FC2Config](#)
Configuration for a camera.
- struct [PropertyInfo](#)
Information about a specific camera property.
- struct [Property](#)
A specific camera property.
- struct [TriggerModelInfo](#)
Information about a camera trigger property.
- struct [TriggerMode](#)
A camera trigger.
- struct [StrobeInfo](#)
A camera strobe property.
- struct [StrobeControl](#)
A camera strobe.
- struct [TimeStamp](#)
Timestamp information.
- struct [ConfigROM](#)
Camera configuration ROM.
- struct [CameraInfo](#)
Camera information.
- struct [EmbeddedImageInfoProperty](#)
Properties of a single embedded image info property.
- struct [EmbeddedImageInfo](#)
Properties of the possible embedded image information.
- struct [ImageMetadata](#)
Metadata related to an image.

- struct [LUTData](#)
Information about the camera's look up table.
- struct [CameraStats](#)
Camera diagnostic information.
- struct [PNGOption](#)
Options for saving PNG images.

Typedefs

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

6.4.1 Detailed Description

6.4.2 Typedef Documentation

6.4.2.1 typedef Property TriggerDelay

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

6.4.2.2 typedef PropertyInfo TriggerDelayInfo

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

6.5 GigE specific structures

These structures are specific to GigE camera operation only.

Collaboration diagram for GigE specific structures:

Classes

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

6.5.1 Detailed Description

These structures are specific to GigE camera operation only.

6.6 IIDC specific structures

These structures are specific to IIDC camera operation only.

Collaboration diagram for IIDC specific structures:

Classes

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

6.6.1 Detailed Description

These structures are specific to IIDC camera operation only.

6.7 Image saving structures.

These structures define various parameters used for saving images.

Collaboration diagram for Image saving structures.:

Classes

- struct [PNGOption](#)
Options for saving PNG images.
- struct [PPMOption](#)
Options for saving PPM images.
- struct [PGMOption](#)
Options for saving PGM images.
- struct [TIFFOption](#)
Options for saving TIFF images.
- struct [JPEGOption](#)
Options for saving JPEG image.
- struct [JPG2Option](#)
Options for saving JPEG2000 image.
- struct [BMPOption](#)
Options for saving Bitmap image.
- struct [MJPGOption](#)
Options for saving MJPG files.
- struct [H264Option](#)
Options for saving H264 files.
- struct [AVIOption](#)
Options for saving AVI files.
- struct [EventOptions](#)
Options for enabling device event registration.
- struct [EventCallbackData](#)

Typedefs

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

6.7.1 Detailed Description

These structures define various parameters used for saving images.

6.7.2 Typedef Documentation

6.7.2.1 typedef void(* CameraEventCallback) (void *data)

Chapter 7

Namespace Documentation

7.1 FlyCap3CameraControl Namespace Reference

Classes

- class [FlyCapture3ApiGuiWrapper](#)

7.2 FlyCapture2 Namespace Reference

Classes

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

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

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

Typedefs

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

Enumerations

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

The error types returned by functions.

- enum [BusCallbackType](#) {
[BUS_RESET](#),
[ARRIVAL](#),
[REMOVAL](#),
[CALLBACK_TYPE_FORCE_32BITS](#) = FULL_32BIT_VALUE }

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

- enum [GrabMode](#) {
[DROP_FRAMES](#),
[BUFFER_FRAMES](#),
[UNSPECIFIED_GRAB_MODE](#),

GRAB_MODE_FORCE_32BITS = FULL_32BIT_VALUE }

The grab strategy employed during image transfer.

- enum GrabTimeout {
TIMEOUT_NONE = 0,
TIMEOUT_INFINITE = -1,
TIMEOUT_UNSPECIFIED = -2,
GRAB_TIMEOUT_FORCE_32BITS = FULL_32BIT_VALUE }

Timeout options for grabbing images.

- enum BandwidthAllocation {
BANDWIDTH_ALLOCATION_OFF = 0,
BANDWIDTH_ALLOCATION_ON = 1,
BANDWIDTH_ALLOCATION_UNSUPPORTED = 2,
BANDWIDTH_ALLOCATION_UNSPECIFIED = 3,
BANDWIDTH_ALLOCATION_FORCE_32BITS = FULL_32BIT_VALUE }

Bandwidth allocation options for 1394 devices.

- enum InterfaceType {
INTERFACE_IEEE1394,
INTERFACE_USB2,
INTERFACE_USB3,
INTERFACE_GIGE,
INTERFACE_UNKNOWN,
INTERFACE_TYPE_FORCE_32BITS = FULL_32BIT_VALUE }

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

- enum PropertyType {
BRIGHTNESS,
AUTO_EXPOSURE,
SHARPNESS,
WHITE_BALANCE,
HUE,
SATURATION,
GAMMA,
IRIS,
FOCUS,
ZOOM,
PAN,
TILT,
SHUTTER,
GAIN,
TRIGGER_MODE,
TRIGGER_DELAY,
FRAME_RATE,
TEMPERATURE,
UNSPECIFIED_PROPERTY_TYPE,
PROPERTY_TYPE_FORCE_32BITS = FULL_32BIT_VALUE }

Camera properties.

- enum FrameRate {
FRAMERATE_1_875,
FRAMERATE_3_75,
FRAMERATE_7_5,
FRAMERATE_15,
FRAMERATE_30,
FRAMERATE_60,
FRAMERATE_120,
FRAMERATE_240,
FRAMERATE_FORMAT7,
NUM_FRAMERATES,
FRAMERATE_FORCE_32BITS = FULL_32BIT_VALUE }

Frame rates in frames per second.

- enum [VideoMode](#) {
 VIDEOMODE_160x120YUV444,
 VIDEOMODE_320x240YUV422,
 VIDEOMODE_640x480YUV411,
 VIDEOMODE_640x480YUV422,
 VIDEOMODE_640x480RGB,
 VIDEOMODE_640x480Y8,
 VIDEOMODE_640x480Y16,
 VIDEOMODE_800x600YUV422,
 VIDEOMODE_800x600RGB,
 VIDEOMODE_800x600Y8,
 VIDEOMODE_800x600Y16,
 VIDEOMODE_1024x768YUV422,
 VIDEOMODE_1024x768RGB,
 VIDEOMODE_1024x768Y8,
 VIDEOMODE_1024x768Y16,
 VIDEOMODE_1280x960YUV422,
 VIDEOMODE_1280x960RGB,
 VIDEOMODE_1280x960Y8,
 VIDEOMODE_1280x960Y16,
 VIDEOMODE_1600x1200YUV422,
 VIDEOMODE_1600x1200RGB,
 VIDEOMODE_1600x1200Y8,
 VIDEOMODE_1600x1200Y16,
 VIDEOMODE_FORMAT7,
 NUM_VIDEOMODES,
 VIDEOMODE_FORCE_32BITS = FULL_32BIT_VALUE }

DCAM video modes.

- enum [Mode](#) {


```

MODE_0 = 0,
MODE_1,
MODE_2,
MODE_3,
MODE_4,
MODE_5,
MODE_6,
MODE_7,
MODE_8,
MODE_9,
MODE_10,
MODE_11,
MODE_12,
MODE_13,
MODE_14,
MODE_15,
MODE_16,
MODE_17,
MODE_18,
MODE_19,
MODE_20,
MODE_21,
MODE_22,
MODE_23,
MODE_24,
MODE_25,
MODE_26,
MODE_27,
MODE_28,
MODE_29,
MODE_30,
MODE_31,
NUM_MODES,
MODE_FORCE_32BITS = FULL_32BIT_VALUE }

```

Camera modes for DCAM formats as well as Format7.

- enum PixelFormat {


```

PIXEL_FORMAT_MONO8 = 0x80000000,
PIXEL_FORMAT_411YUV8 = 0x40000000,
PIXEL_FORMAT_422YUV8 = 0x20000000,
PIXEL_FORMAT_444YUV8 = 0x10000000,
PIXEL_FORMAT_RGB8 = 0x08000000,
PIXEL_FORMAT_MONO16 = 0x04000000,
PIXEL_FORMAT_RGB16 = 0x02000000,
PIXEL_FORMAT_S_MONO16 = 0x01000000,
PIXEL_FORMAT_S_RGB16 = 0x00800000,
PIXEL_FORMAT_RAW8 = 0x00400000,
PIXEL_FORMAT_RAW16 = 0x00200000,
PIXEL_FORMAT_MONO12 = 0x00100000,
PIXEL_FORMAT_RAW12 = 0x00080000,
PIXEL_FORMAT_BGR = 0x80000008,
PIXEL_FORMAT_BGRU = 0x40000008,
PIXEL_FORMAT_RGB = PIXEL_FORMAT_RGB8,
PIXEL_FORMAT_RBGU = 0x40000002,
PIXEL_FORMAT_BGR16 = 0x02000001,
PIXEL_FORMAT_BGRU16 = 0x02000002,
PIXEL_FORMAT_422YUV8_JPEG = 0x40000001,
NUM_PIXEL_FORMATS = 20,
UNSPECIFIED_PIXEL_FORMAT = 0 }

```

Pixel formats available for Format7 modes.

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

Bus speeds.

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

Types of low level drivers that flycapture uses.

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

Color processing algorithms.

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

Bayer tile formats.

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

File formats to be used for saving images to disk.

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

Possible properties that can be queried from the camera.

- enum `OSType` {
`WINDOWS_X86`,
`WINDOWS_X64`,
`LINUX_X86`,
`LINUX_X64`,
`MAC`,
`UNKNOWN_OS`,
`OSTYPE_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Possible operating systems.

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

Possible byte orders.

Variables

- static const unsigned int `sk_maxStringLength` = 512

The maximum length that is allocated for a string.

- static const unsigned int `sk_maxNumPorts` = 32

The maximum number of ports one device can have.

7.2.1 Typedef Documentation

7.2.1.1 `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()`.

7.2.1.2 `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.

7.2.1.3 `typedef void* CallbackHandle`

Handle that is returned when registering a callback.

It is required when unregistering the callback.

7.2.1.4 `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.

7.2.2 Enumeration Type Documentation

7.2.2.1 `enum ByteOrder`

Possible byte orders.

Enumerator

`BYTE_ORDER_LITTLE_ENDIAN`

`BYTE_ORDER_BIG_ENDIAN`

`BYTE_ORDER_FORCE_32BITS`

7.2.2.2 `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`

7.3 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](#) }

7.3.1 Enumeration Type Documentation

7.3.1.1 enum [PGRSyncError](#)

Enumerator

[PGRSyncError_OK](#)
[PGRSyncError_FAILED](#)
[PGRSyncError_ALREADY_STARTED](#)
[PGRSyncError_ALREADY_STOPPED](#)
[PGRSyncError_CAMERA_NOT_FOUND](#)
[PGRSyncError_UNKNOWN_ERROR](#)

7.3.1.2 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 8

Class Documentation

8.1 AVIOption Struct Reference

Options for saving AVI files.

Public Member Functions

- [AVIOption](#) ()

Public Attributes

- float [frameRate](#)
Frame rate of the stream.
- unsigned int [reserved](#) [256]
Reserved for future use.

8.1.1 Detailed Description

Options for saving AVI files.

8.1.2 Constructor & Destructor Documentation

8.1.2.1 [AVIOption](#) () [`inline`]

8.1.3 Member Data Documentation

8.1.3.1 float [frameRate](#)

Frame rate of the stream.

8.1.3.2 unsigned int reserved[256]

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

8.2 AVIRecorder Class Reference

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

Public Member Functions

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

8.2.1 Detailed Description

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

8.2.2 Constructor & Destructor Documentation

8.2.2.1 AVIRecorder ()

Default constructor.

8.2.2.2 virtual ~AVIRecorder () [virtual]

Default destructor.

8.2.3 Member Function Documentation

8.2.3.1 virtual Error AVIAppend (Image * pImage) [virtual]

Append an image to the AVI/MP4 file.

Parameters

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

Returns

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

8.2.3.2 virtual Error AVIClose () [virtual]

Close the AVI/MP4 file.

See also

[AVIOpen\(\)](#)

Returns

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

8.2.3.3 virtual Error AVIOpen (const char * *pFileName*, AVIOption * *pOption*) [virtual]

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

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

Parameters

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

See also

[AVIClose\(\)](#)

Returns

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

8.2.3.4 virtual Error AVIOpen (const char * *pFileName*, MJPGOption * *pOption*) [virtual]

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

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

Parameters

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

See also

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

Returns

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

8.2.3.5 virtual **Error** AVIOpen (const char * *pFileName*, **H264Option** * *pOption*) [virtual]

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

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

Parameters

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

See also

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

Returns

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

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

- [AVIRecorder.h](#)

8.3 BMPOption Struct Reference

Options for saving Bitmap image.

Public Member Functions

- [BMPOption](#) ()

Public Attributes

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

Reserved for future use.

8.3.1 Detailed Description

Options for saving Bitmap image.

8.3.2 Constructor & Destructor Documentation

8.3.2.1 `BMPOption()` `[inline]`

8.3.3 Member Data Documentation

8.3.3.1 bool `indexedColor_8bit`

8.3.3.2 unsigned int `reserved[16]`

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

8.4 BusManager Class Reference

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

Public Member Functions

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

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

Static Public Member Functions

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

8.4.1 Detailed Description

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

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

8.4.2 Constructor & Destructor Documentation

8.4.2.1 BusManager ()

Default constructor.

8.4.2.2 virtual ~BusManager () [virtual]

Default destructor.

8.4.3 Member Function Documentation

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

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

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

Parameters

<i>gigECameras</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.

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

Fire a bus reset.

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

Parameters

<i>pGuid</i>	PGRGuid of the camera or the device to cause bus reset.
--------------	---

Returns

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

8.4.3.3 static Error ForceAllIPAddressesAutomatically () [static]

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

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

Returns

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

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

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

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

Returns

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

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

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

This is useful in situations where a GigE Vision camera is using Persistent IP and the application's subnet is different from the device subnet.

Parameters

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

Returns

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

8.4.3.6 virtual Error GetCameraFromIndex (unsigned int *index*, **PGRGuid** * *pGuid*) [virtual]

Gets the [PGRGuid](#) for a camera on the PC.

It uniquely identifies the camera specified by the index and is used to identify the camera during a [Camera::Connect\(\)](#) call.

Parameters

<i>index</i>	Zero based index of camera.
<i>pGuid</i>	Unique PGRGuid for the camera.

See also

[GetCameraFromSerialNumber\(\)](#)

Returns

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

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

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

Parameters

<i>ipAddress</i>	IP address to get GUID for.
<i>pGuid</i>	Unique PGRGuid for the camera.

Returns

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

8.4.3.8 `virtual Error GetCameraFromSerialNumber (unsigned int serialNumber, PGRGuid * pGuid)` [virtual]

Gets the [PGRGuid](#) for a camera on the PC.

It uniquely identifies the camera specified by the serial number and is used to identify the camera during a [Camera↔::Connect\(\)](#) call.

Parameters

<i>serialNumber</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.

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

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

Parameters

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

Returns

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

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

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

It uniquely identifies the device specified by the index.

Parameters

<i>index</i>	Zero based index of device.
<i>pGuid</i>	Unique PGRGuid for the device.

See also

[GetNumOfDevices\(\)](#)

Returns

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

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

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

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

Parameters

<i>pGuid</i>	The PGRGuid to get the interface for.
<i>pInterfaceType</i>	The interface type of the PGRGuid .

Returns

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

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

Gets the number of cameras attached to the PC.

Parameters

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

Returns

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

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

Gets the number of devices.

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

Parameters

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

Returns

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

8.4.3.14 `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.

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

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

Parameters

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

Returns

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

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

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

Parameters

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

Returns

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

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

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>pControlable</i>	Indicates whether camera is controllable

Returns

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

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

Read a phy register on the specified device.

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

Parameters

<i>guid</i>	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.

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

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

Parameters

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

See also

[UnregisterCallback\(\)](#)

Returns

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

8.4.3.20 `virtual Error RescanBus () [virtual]`

Force a rescan of the buses.

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

Returns

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

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

Unregister a callback function.

Parameters

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

See also

[RegisterCallback\(\)](#)

Returns

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

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

Write a phy register on the specified device.

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

Parameters

<i>guid</i>	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](#)

8.5 Camera Class Reference

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

Inheritance diagram for Camera:

Collaboration diagram for Camera:

Public Member Functions

- [Camera](#) ()
Default constructor.
- virtual [~Camera](#) ()
Default destructor.
- virtual [Error Connect](#) ([PGRGuid](#) *pGuid=NULL)
The following functions are inherited from [CameraBase](#).
- virtual [Error Disconnect](#) ()
Disconnects the camera object from the camera.
- virtual bool [IsConnected](#) ()
Checks if the camera object is currently connected to a physical camera.
- virtual [Error SetCallback](#) ([ImageEventCallback](#) callbackFn, const void *pCallbackData=NULL)
Sets the callback data to be used on completion of image transfer.
- virtual [Error StartCapture](#) ([ImageEventCallback](#) callbackFn=NULL, const void *pCallbackData=NULL)
Starts isochronous image capture.
- virtual [Error RetrieveBuffer](#) ([Image](#) *pImage)
Retrieves the the next image object containing the next image.
- virtual [Error StopCapture](#) ()
Stops isochronous image transfer and cleans up all associated resources.
- virtual [Error WaitForBufferEvent](#) ([Image](#) *pImage, unsigned int eventNumber)
Retrieves the next image event containing the next part of the image.
- virtual [Error SetUserBuffers](#) (unsigned char *const pMemBuffers, int size, int numBuffers)
Specify user allocated buffers to use as image data buffers.
- virtual [Error GetConfiguration](#) ([FC2Config](#) *pConfig)
Get the configuration associated with the camera object.
- virtual [Error SetConfiguration](#) (const [FC2Config](#) *pConfig)
Set the configuration associated with the camera object.
- virtual [Error GetCameraInfo](#) ([CameraInfo](#) *pCameraInfo)
Retrieves information from the camera such as serial number, model name and other camera information.
- virtual [Error GetPropertyInfo](#) ([PropertyInfo](#) *pPropInfo)
Retrieves information about the specified camera property.
- virtual [Error GetProperty](#) ([Property](#) *pProp)
Reads the settings for the specified property from the camera.
- virtual [Error SetProperty](#) (const [Property](#) *pProp, bool broadcast=false)
Writes the settings for the specified property to the camera.
- virtual [Error GetGPIOPinDirection](#) (unsigned int pin, unsigned int *pDirection)
Get the GPIO pin direction for the specified pin.
- virtual [Error SetGPIOPinDirection](#) (unsigned int pin, unsigned int direction, bool broadcast=false)
Set the GPIO pin direction for the specified pin.
- virtual [Error 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 [ImageEvent](#)↵
[Callback](#) *pCallbackFns=NULL, const void **pCallbackDataArray=NULL)
- static const char * [GetRegisterString](#) (unsigned int registerVal)

DCAM Formats

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

- virtual [Error GetVideoModeAndFrameRateInfo](#) ([VideoMode](#) videoMode, [FrameRate](#) frameRate, bool *p↵
Supported)
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 *pSettingsAre↵
Valid, [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 packet↵
Size)
Set the current Format7 configuration to the camera.
- virtual [Error SetFormat7Configuration](#) (const [Format7ImageSettings](#) *pImageSettings, float percentSpeed)
Set the current Format7 configuration to the camera.

Additional Inherited Members

8.5.1 Detailed Description

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

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

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

8.5.2 Constructor & Destructor Documentation

8.5.2.1 Camera ()

Default constructor.

8.5.2.2 virtual ~Camera () [virtual]

Default destructor.

8.5.3 Member Function Documentation

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

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

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

Implements [CameraBase](#).

8.5.3.2 virtual Error DeregisterAllEvents (void) [virtual]

Implements [CameraBase](#).

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

Implements [CameraBase](#).

8.5.3.4 virtual Error Disconnect () [virtual]

Disconnects the camera object from the camera.

This allows another physical camera to be connected to the camera object.

See also

[Connect\(\)](#)

Returns

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

Implements [CameraBase](#).

8.5.3.5 virtual Error EnableLUT (bool *on*) [virtual]

Enable or disable LUT functionality on the camera.

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

8.5.3.6 virtual Error FireSoftwareTrigger (bool *broadcast* = false) [virtual]

Fire the software trigger according to the DCAM specifications.

Parameters

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

Returns

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

Implements [CameraBase](#).

8.5.3.7 virtual Error GetActiveLUTBank (unsigned int * *pActiveBank*) [virtual]

Get the LUT bank that is currently being used.

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

Parameters

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

Returns

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

Implements [CameraBase](#).

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

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

Parameters

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

Returns

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

Implements [CameraBase](#).

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

Get the configuration associated with the camera object.

Parameters

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

See also

[SetConfiguration\(\)](#)

Returns

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

Implements [CameraBase](#).

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

Returns a Timestamp struct containing 1394 CYCLE_TIME information.

Parameters

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

Returns

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

Implements [CameraBase](#).

8.5.3.11 virtual **Error** GetEmbeddedImageInfo (**EmbeddedImageInfo** * *pInfo*) [virtual]

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

Parameters

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

See also

[SetEmbeddedImageInfo\(\)](#)

Returns

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

Implements [CameraBase](#).

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

See also

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

Returns

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

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

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

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

Parameters

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

See also

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

Returns

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

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

Get the GPIO pin direction for the specified pin.

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

Parameters

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

See also

[SetGPIOPinDirection\(\)](#)

Returns

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

Implements [CameraBase](#).

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

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

Parameters

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

Returns

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

Implements [CameraBase](#).

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

Get the LUT channel settings from the camera.

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

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

Query if LUT support is available on the camera.

Note that some cameras may report support for the LUT and return an inputBitDepth of 0. In these cases use log2(numEntries) for the inputBitDepth.

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

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

Retrieve the current memory channel from the camera.

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

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

Query the camera for memory channel support.

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

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

8.5.3.20 virtual **Error** **GetProperty** (**Property** * *pProp*) [virtual]

Reads the settings for the specified property from the camera.

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

Parameters

<i>pProp</i>	Pointer to the Property structure to be filled.
--------------	---

See also

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

Returns

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

Implements [CameraBase](#).

8.5.3.21 virtual **Error** **GetPropertyInfo** (**PropertyInfo** * *pPropInfo*) [virtual]

Retrieves information about the specified camera property.

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

Parameters

<i>pPropInfo</i>	Pointer to the PropertyInfo structure to be filled.
------------------	---

See also

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

Returns

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

Implements [CameraBase](#).

8.5.3.22 static const char* **GetRegisterString** (unsigned int *registerVal*) [static]**8.5.3.23** virtual **Error** **GetStats** (**CameraStats** * *pStats*) [virtual]

Implements [CameraBase](#).

8.5.3.24 virtual Error GetStrobe (StrobeControl * *pStrobeControl*) [virtual]

Retrieve current strobe settings from the camera.

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

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

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

Retrieve strobe information from the camera.

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

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

Retrieve current trigger delay settings from the camera.

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

8.5.3.27 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

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

Returns

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

Implements [CameraBase](#).

8.5.3.28 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

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

[GetTriggerDelay\(\)](#)
[SetTriggerDelay\(\)](#)

Returns

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

Implements [CameraBase](#).

8.5.3.29 `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](#).

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

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

If the camera is in Format7, the video mode will be VIDEOMODE_FORMAT7 and the frame rate will be FRAME↵
RATE_FORMAT7.

Parameters

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

See also

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

Returns

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

8.5.3.31 `virtual Error GetVideoModeAndFrameRateInfo (VideoMode videoMode, FrameRate frameRate, bool * pSupported)` `[virtual]`

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

Parameters

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

See also

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

Returns

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

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

Checks if the camera object is currently connected to a physical camera.

See also

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

Returns

Whether the camera object is connected to a physical camera.

Implements [CameraBase](#).

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

Read the specified register from the camera.

Parameters

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

See also

[WriteRegister\(\)](#)

Returns

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

Implements [CameraBase](#).

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

Read from the specified register block on the camera.

Parameters

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

See also

[WriteRegisterBlock\(\)](#)

Returns

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

Implements [CameraBase](#).

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

Implements [CameraBase](#).

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

Implements [CameraBase](#).

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

Implements [CameraBase](#).

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

Restore the specified current memory channel.

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

8.5.3.39 virtual Error RetrieveBuffer (Image * *plimage*) [virtual]

Retrieves the the next image object containing the next image.

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

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

8.5.3.40 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](#).

8.5.3.41 virtual Error SetActiveLUTBank (unsigned int *activeBank*) [virtual]

Set the LUT bank that will be used.

Parameters

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

Returns

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

Implements [CameraBase](#).

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

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

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

Parameters

<i>callbackFn</i>	A function to be called when a new image is received.
<i>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](#).

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

Set the configuration associated with the camera object.

Parameters

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

See also

[GetConfiguration\(\)](#)

Returns

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

Implements [CameraBase](#).

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

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

Parameters

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

See also

[GetEmbeddedImageInfo\(\)](#)

Returns

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

Implements [CameraBase](#).

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

Set the current Format7 configuration to the camera.

Parameters

<i>pImageSettings</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.

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

Set the current Format7 configuration to the camera.

Parameters

<i>plImageSettings</i>	Image settings to be written to the camera.
<i>percentSpeed</i>	Percentage of packet size to be written to the camera.

See also

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

Returns

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

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

Set the GPIO pin direction for the specified pin.

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

Parameters

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

See also

[SetGPIOPinDirection\(\)](#)

Returns

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

Implements [CameraBase](#).

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

Set the LUT channel settings to the camera.

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

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

Writes the settings for the specified property to the camera.

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

Parameters

<i>pProp</i>	Pointer to the 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](#).

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

Set current strobe settings to the camera.

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

Parameters

<i>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](#).

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

Set the specified trigger delay settings to the camera.

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

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

Set the specified trigger settings to the camera.

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

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

Specify user allocated buffers to use as image data buffers.

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

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

8.5.3.54 `virtual Error SetVideoModeAndFrameRate (VideoMode videoMode, FrameRate frameRate)` `[virtual]`

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

It is not possible to set the camera to VIDEOMODE_FORMAT7 or FRAMERATE_FORMAT7. Use the Format7 functions to set the camera into Format7.

Parameters

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

See also

[GetVideoModeAndFrameRateInfo\(\)](#)
[GetVideoModeAndFrameRate\(\)](#)

Returns

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

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

Starts isochronous image capture.

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

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

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

8.5.3.57 virtual Error StopCapture () [virtual]

Stops isochronous image transfer and cleans up all associated resources.

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

See also

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

Returns

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

Implements [CameraBase](#).

8.5.3.58 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>pImageSettings</i>	Structure containing the image settings.
<i>pSettingsAreValid</i>	Whether the settings are valid.
<i>pPacketInfo</i>	Packet size information that can be used to determine a valid packet size.

See also

[GetFormat7Info\(\)](#)
[GetFormat7Configuration\(\)](#)
[SetFormat7Configuration\(\)](#)

Returns

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

8.5.3.59 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>eventNumber</i>	The event number to wait for.

See also

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

Returns

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

Implements [CameraBase](#).

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

Write to the specified register on the camera.

Parameters

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

See also

[ReadRegister\(\)](#)

Returns

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

Implements [CameraBase](#).

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

Write to the specified register block on the camera.

Parameters

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

See also

[ReadRegisterBlock\(\)](#)

Returns

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

Implements [CameraBase](#).

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

- [Camera.h](#)

8.6 CameraBase Class Reference

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

Inheritance diagram for CameraBase:

Public Member Functions

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

Protected Attributes

- CameraData * [m_pCameraData](#)

Connection and Image Retrieval

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

- virtual [Error Connect](#) ([PGRGuid](#) *pGuid=NULL)=0
Connects the camera object to the camera specified by the GUID.
- virtual [Error Disconnect](#) ()=0
Disconnects the camera object from the camera.
- virtual bool [IsConnected](#) ()=0
Checks if the camera object is currently connected to a physical camera.
- virtual [Error SetCallback](#) ([ImageEventCallback](#) callbackFn, const void *pCallbackData=NULL)=0
Sets the callback data to be used on completion of image transfer.
- virtual [Error StartCapture](#) ([ImageEventCallback](#) callbackFn=NULL, const void *pCallbackData=NULL)=0
Starts isochronous image capture.
- virtual [Error RetrieveBuffer](#) ([Image](#) *pImage)=0
Retrieves the the next image object containing the next image.
- virtual [Error StopCapture](#) ()=0
Stops isochronous image transfer and cleans up all associated resources.
- virtual [Error WaitForBufferEvent](#) ([Image](#) *pImage, unsigned int eventNumber)=0
Retrieves the next image event containing the next part of the image.
- virtual [Error SetUserBuffers](#) (unsigned char *const pMemBuffers, int size, int numBuffers)=0
Specify user allocated buffers to use as image data buffers.
- virtual [Error GetConfiguration](#) ([FC2Config](#) *pConfig)=0
Get the configuration associated with the camera object.
- virtual [Error SetConfiguration](#) (const [FC2Config](#) *pConfig)=0
Set the configuration associated with the camera object.
- static [Error StartSyncCapture](#) (unsigned int numCameras, const [CameraBase](#) **ppCameras, const [ImageEventCallback](#) *pCallbackFns=NULL, const void **pCallbackDataArray=NULL)
Starts isochronous image capture on multiple cameras.

Information and Properties

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

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

General Purpose Input / Output

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

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

Trigger

These functions deal with trigger control on the camera.

- virtual [Error 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.

8.6.1 Detailed Description

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

8.6.2 Constructor & Destructor Documentation

8.6.2.1 [CameraBase](#) () [inline]

Default constructor.

8.6.2.2 virtual ~[CameraBase](#) () [inline], [virtual]

Default destructor.

8.6.3 Member Function Documentation

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

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

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

Parameters

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

See also

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

Returns

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

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

8.6.3.2 virtual Error DeregisterAllEvents (void) [pure virtual]

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

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

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

8.6.3.4 virtual Error Disconnect () [pure virtual]

Disconnects the camera object from the camera.

This allows another physical camera to be connected to the camera object.

See also

[Connect\(\)](#)

Returns

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

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

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

Enable or disable LUT functionality on the camera.

Parameters

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

See also

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

Returns

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

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

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

Fire the software trigger according to the DCAM specifications.

Parameters

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

Returns

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

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

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

Get the LUT bank that is currently being used.

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

Parameters

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

Returns

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

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

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

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

Parameters

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

Returns

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

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

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

Get the configuration associated with the camera object.

Parameters

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

See also

[SetConfiguration\(\)](#)

Returns

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

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

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

Returns a Timestamp struct containing 1394 CYCLE_TIME information.

Parameters

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

Returns

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

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

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

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

Parameters

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

See also

[SetEmbeddedImageInfo\(\)](#)

Returns

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

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

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

Get the GPIO pin direction for the specified pin.

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

Parameters

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

See also

[SetGPIOPinDirection\(\)](#)

Returns

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

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

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

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

Parameters

<i>bank</i>	The bank to query.
<i>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](#).

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

Get the LUT channel settings from the camera.

Parameters

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

See also

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

Returns

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

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

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

Query if LUT support is available on the camera.

Note that some cameras may report support for the LUT and return an inputBitDepth of 0. In these cases use log2(numEntries) for the inputBitDepth.

Parameters

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

See also

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

Returns

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

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

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

Retrieve the current memory channel from the camera.

Parameters

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

See also

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

Returns

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

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

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

Query the camera for memory channel support.

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

Parameters

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

See also

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

Returns

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

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

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

Reads the settings for the specified property from the camera.

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

Parameters

<i>pProp</i>	Pointer to the 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](#).

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

Retrieves information about the specified camera property.

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

Parameters

<i>pPropInfo</i>	Pointer to the 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](#).

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

Returns a text representation of the register value.

Parameters

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

Returns

The text representation of the register.

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

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

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

Retrieve current strobe settings from the camera.

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

Parameters

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

See also

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

Returns

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

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

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

Retrieve strobe information from the camera.

Parameters

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

See also

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

Returns

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

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

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

Retrieve current trigger delay settings from the camera.

Parameters

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

See also

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

Returns

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

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

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

Retrieve trigger delay information from the camera.

Parameters

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

See also

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

Returns

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

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

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

Retrieve current trigger settings from the camera.

Parameters

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

See also

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

Returns

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

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

8.6.3.27 virtual **Error** GetTriggerModelInfo (**TriggerModelInfo** * *pTriggerModelInfo*) [pure virtual]

Retrieve trigger information from the camera.

Parameters

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

See also

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

Returns

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

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

8.6.3.28 virtual **bool** IsConnected () [pure virtual]

Checks if the camera object is currently connected to a physical camera.

See also

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

Returns

Whether the camera object is connected to a physical camera.

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

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

Read the specified register from the camera.

Parameters

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

See also

[WriteRegister\(\)](#)

Returns

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

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

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

Read from the specified register block on the camera.

Parameters

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

See also

[WriteRegisterBlock\(\)](#)

Returns

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

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

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

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

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

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

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

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

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

Restore the specified current memory channel.

Parameters

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

See also

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

Returns

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

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

8.6.3.35 `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](#).

8.6.3.36 virtual Error SaveToMemoryChannel (unsigned int *channel*) [pure virtual]

Save the current settings to the specified current memory channel.

Parameters

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

See also

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

Returns

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

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

8.6.3.37 virtual Error SetActiveLUTBank (unsigned int *activeBank*) [pure virtual]

Set the LUT bank that will be used.

Parameters

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

Returns

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

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

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

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

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

Parameters

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

See also

[StartCapture\(\)](#)

Returns

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

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

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

Set the configuration associated with the camera object.

Parameters

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

See also

[GetConfiguration\(\)](#)

Returns

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

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

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

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

Parameters

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

See also

[GetEmbeddedImageInfo\(\)](#)

Returns

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

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

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

Set the GPIO pin direction for the specified pin.

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

Parameters

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

See also

[GetGPIOPinDirection\(\)](#)

Returns

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

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

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

Set the LUT channel settings to the camera.

Parameters

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

See also

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

Returns

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

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

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

Writes the settings for the specified property to the camera.

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

Parameters

<i>pProp</i>	Pointer to the 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](#).

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

Set current strobe settings to the camera.

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

Parameters

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

See also

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

Returns

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

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

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

Set the specified trigger delay settings to the camera.

Parameters

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

See also

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

Returns

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

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

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

Set the specified trigger settings to the camera.

Parameters

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

See also

[GetTriggerModelInfo\(\)](#)
[GetTriggerMode\(\)](#)

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

Returns

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

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

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

Specify user allocated buffers to use as image data buffers.

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

Parameters

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

See also

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

Returns

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

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

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

Starts isochronous image capture.

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

Parameters

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

See also

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

Returns

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

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

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

Starts isochronous image capture on multiple cameras.

On each frame, the time stamps across the cameras are aligned which means the frames are synchronized. Note that the cameras must be synchronized by external means in order for this function to work. This means that the cameras should either be on the same bus, hardware synchronized (e.g. through triggering) or Multisync is running. Note: The use of this function with GigE Cameras is not supported.

Parameters

<i>numCameras</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>pCallbackFns</i>	Array of callback functions for each camera.
<i>pCallbackDataArray</i>	Array of callback data pointers.

See also

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

Returns

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

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

Stops isochronous image transfer and cleans up all associated resources.

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

See also

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

Returns

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

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

8.6.3.51 `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>eventNumber</i>	The event number to wait for.

See also

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

Returns

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

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

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

Write to the specified register on the camera.

Parameters

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

See also

[ReadRegister\(\)](#)

Returns

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

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

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

Write to the specified register block on the camera.

Parameters

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

See also

[ReadRegisterBlock\(\)](#)

Returns

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

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

8.6.4 Member Data Documentation

8.6.4.1 `CameraData* m_pCameraData [protected]`

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

- [CameraBase.h](#)

8.7 CameraControlDlg Class Reference

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

Public Member Functions

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

8.7.1 Detailed Description

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

8.7.2 Constructor & Destructor Documentation

8.7.2.1 [CameraControlDlg](#) ()

Default constructor.

8.7.2.2 [~CameraControlDlg](#) ()

Default destructor.

8.7.3 Member Function Documentation

8.7.3.1 void [Connect](#) ([CameraBase](#) * pCamera)

Connect dialog to a camera.

Parameters

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

8.7.3.2 void Disconnect ()

Disconnect a connected camera from the dialog.

8.7.3.3 void Hide ()

Hide the dialog.

8.7.3.4 bool IsVisible ()

Get the visibility of the dialog.

Returns

Whether the dialog is visible.

8.7.3.5 void SetTitle (const char * *title*)

Change the title of the window.

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

Parameters

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

8.7.3.6 void Show ()

Show the dialog.

8.7.3.7 void Show (void * *pParent*)

Show the dialog.

8.7.3.8 void ShowModal ()

Show the modal dialog.

8.7.3.9 void ShowModal (void * *pParent*)

Show the modal dialog.

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

- [FlyCapture2GUI.h](#)

8.8 CameraInfo Struct Reference

[Camera](#) information.

Collaboration diagram for CameraInfo:

Public Member Functions

- [CameraInfo](#) ()

Public Attributes

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

- unsigned short [nodeNumber](#)
ieee1394 Node number, set to 0 for GigE and USB cameras
- [PCleBusSpeed](#) [pcieBusSpeed](#)
PCle Bus Speed, set to PCIE_BUSSPEED_UNKNOWN for unsupported drivers.
- unsigned int [reserved](#) [16]
Reserved for future use.

IIDC specific information

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

GigE specific information

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

8.8.1 Detailed Description

[Camera](#) information.

8.8.2 Constructor & Destructor Documentation

8.8.2.1 [CameraInfo](#)() [[inline](#)]

8.8.3 Member Data Documentation

8.8.3.1 unsigned int [applicationIPAddress](#)

Local Application IP Address.

8.8.3.2 unsigned int applicationPort

Local Application port.

8.8.3.3 BayerTileFormat bayerTileFormat

Bayer tile format.

8.8.3.4 unsigned short busNumber

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

8.8.3.5 unsigned int ccpStatus

Status/Content of CCP register.

8.8.3.6 ConfigROM configROM

Configuration ROM data.

8.8.3.7 IPAddress defaultGateway

Default gateway.

8.8.3.8 char driverName[sk_maxStringLength]

Driver name of driver being used.

8.8.3.9 DriverType driverType

Driver type.

8.8.3.10 char firmwareBuildTime[sk_maxStringLength]

Firmware build time.

8.8.3.11 char firmwareVersion[sk_maxStringLength]

Firmware version of camera.

8.8.3.12 unsigned int gigEMajorVersion

GigE Vision version.

8.8.3.13 unsigned int gigEMinorVersion

GigE Vision minor version.

8.8.3.14 unsigned int iidcVer

DCAM version.

8.8.3.15 InterfaceType interfaceType

Interface type.

8.8.3.16 IPAddress ipAddress

IP address.

8.8.3.17 bool isColorCamera

Flag indicating if this is a color camera.

8.8.3.18 MACAddress macAddress

MAC address.

8.8.3.19 BusSpeed maximumBusSpeed

Maximum bus speed.

8.8.3.20 char modelName[sk_maxStringLength]

Device model name.

8.8.3.21 unsigned short nodeNumber

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

8.8.3.22 PCIeBusSpeed pcieBusSpeed

PCIe Bus Speed, set to PCIE_BUSSPEED_UNKNOWN for unsupported drivers.

8.8.3.23 unsigned int reserved[16]

Reserved for future use.

8.8.3.24 char sensorInfo[sk_maxStringLength]

String detailing the sensor information.

8.8.3.25 char sensorResolution[sk_maxStringLength]

String providing the sensor resolution.

8.8.3.26 unsigned int serialNumber

Device serial number.

8.8.3.27 IPAddress subnetMask

Subnet mask.

8.8.3.28 char userDefinedName[sk_maxStringLength]

User defined name.

8.8.3.29 char vendorName[sk_maxStringLength]

Device vendor name.

8.8.3.30 char xmlURL1[sk_maxStringLength]

XML URL 1.

8.8.3.31 char xmlURL2[sk_maxStringLength]

XML URL 2.

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

- [FlyCapture2Defs.h](#)

8.9 CameraSelectionDlg Class Reference

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

Public Member Functions

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

8.9.1 Detailed Description

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

8.9.2 Constructor & Destructor Documentation

8.9.2.1 CameraSelectionDlg ()

Default constructor.

8.9.2.2 ~CameraSelectionDlg ()

Default destructor.

8.9.3 Member Function Documentation

8.9.3.1 void SetTitle (const char * title)

Set the window title.

Parameters

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

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

Show the [CameraSelectionDlg](#).

Parameters

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

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

- [FlyCapture2GUI.h](#)

8.10 CameraStats Struct Reference

[Camera](#) diagnostic information.

Collaboration diagram for CameraStats:

Public Member Functions

- [CameraStats](#) ()

Public Attributes

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

8.10.1 Detailed Description

[Camera](#) diagnostic information.

8.10.2 Constructor & Destructor Documentation

8.10.2.1 **CameraStats**() `[inline]`

8.10.3 Member Data Documentation

8.10.3.1 **float** cameraCurrents[8]

8.10.3.2 **bool** cameraPowerUp

8.10.3.3 **float** cameraVoltages[8]

8.10.3.4 **unsigned int** imageCorrupt

8.10.3.5 **unsigned int** imageDriverDropped

8.10.3.6 **unsigned int** imageDropped

8.10.3.7 **unsigned int** imageXmitFailed

8.10.3.8 **unsigned int** numCurrents

The number of current registers available.

0: the values in cameraCurrents[] are invalid.

8.10.3.9 **unsigned int** numResendPacketsReceived

8.10.3.10 **unsigned int** numResendPacketsRequested

8.10.3.11 **unsigned int** numVoltages

The number of voltage registers available.

0: the values in cameraVoltages[] are invalid.

8.10.3.12 unsigned int portErrors

8.10.3.13 unsigned int regReadFailed

8.10.3.14 unsigned int regWriteFailed

8.10.3.15 unsigned int reserved[16]

Reserved for future use.

8.10.3.16 unsigned int temperature

8.10.3.17 unsigned int timeSinceBusReset

8.10.3.18 unsigned int timeSinceInitialization

8.10.3.19 TimeStamp timeStamp

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

- [FlyCapture2Defs.h](#)

8.11 ConfigROM Struct Reference

[Camera](#) configuration ROM.

Public Member Functions

- [ConfigROM](#) ()

Public Attributes

- unsigned int [nodeVendorId](#)
Vendor ID of a node.
- unsigned int [chipIdHi](#)
Chip ID (high part).
- unsigned int [chipIdLo](#)
Chip ID (low part).
- unsigned int [unitSpecId](#)
Unit Spec ID, usually 0xa02d.
- unsigned int [unitSWVer](#)
Unit software version.
- unsigned int [unitSubSWVer](#)
Unit sub software version.
- unsigned int [vendorUniqueInfo_0](#)

Vendor unique info 0.

- unsigned int [vendorUniqueInfo_1](#)

Vendor unique info 1.

- unsigned int [vendorUniqueInfo_2](#)

Vendor unique info 2.

- unsigned int [vendorUniqueInfo_3](#)

Vendor unique info 3.

- char [pszKeyword](#) [[sk_maxStringLength](#)]

Keyword.

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

Reserved for future use.

8.11.1 Detailed Description

[Camera](#) configuration ROM.

8.11.2 Constructor & Destructor Documentation

8.11.2.1 `ConfigROM()` [`inline`]

8.11.3 Member Data Documentation

8.11.3.1 unsigned int `chipIdHi`

Chip ID (high part).

8.11.3.2 unsigned int `chipIdLo`

Chip ID (low part).

8.11.3.3 unsigned int `nodeVendorId`

Vendor ID of a node.

8.11.3.4 char `pszKeyword[sk_maxStringLength]`

Keyword.

8.11.3.5 unsigned int `reserved[16]`

Reserved for future use.

8.11.3.6 unsigned int unitSpecId

Unit Spec ID, usually 0xa02d.

8.11.3.7 unsigned int unitSubSWVer

Unit sub software version.

8.11.3.8 unsigned int unitSWVer

Unit software version.

8.11.3.9 unsigned int vendorUniqueInfo_0

Vendor unique info 0.

8.11.3.10 unsigned int vendorUniqueInfo_1

Vendor unique info 1.

8.11.3.11 unsigned int vendorUniqueInfo_2

Vendor unique info 2.

8.11.3.12 unsigned int vendorUniqueInfo_3

Vendor unique info 3.

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

- [FlyCapture2Defs.h](#)

8.12 EmbeddedImageInfo Struct Reference

Properties of the possible embedded image information.

Collaboration diagram for EmbeddedImageInfo:

Public Attributes

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

8.12.1 Detailed Description

Properties of the possible embedded image information.

8.12.2 Member Data Documentation

8.12.2.1 [EmbeddedImageInfoProperty brightness](#)

8.12.2.2 [EmbeddedImageInfoProperty exposure](#)

8.12.2.3 [EmbeddedImageInfoProperty frameCounter](#)

8.12.2.4 [EmbeddedImageInfoProperty gain](#)

8.12.2.5 [EmbeddedImageInfoProperty GPIOPinState](#)

8.12.2.6 [EmbeddedImageInfoProperty ROIPosition](#)

8.12.2.7 [EmbeddedImageInfoProperty shutter](#)

8.12.2.8 [EmbeddedImageInfoProperty strobePattern](#)

8.12.2.9 [EmbeddedImageInfoProperty timestamp](#)

8.12.2.10 [EmbeddedImageInfoProperty whiteBalance](#)

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

- [FlyCapture2Defs.h](#)

8.13 EmbeddedImageInfoProperty Struct Reference

Properties of a single embedded image info property.

Public Member Functions

- [EmbeddedImageInfoProperty](#) ()

Public Attributes

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

8.13.1 Detailed Description

Properties of a single embedded image info property.

8.13.2 Constructor & Destructor Documentation

8.13.2.1 [EmbeddedImageInfoProperty](#) () `[inline]`

8.13.3 Member Data Documentation

8.13.3.1 bool [available](#)

Whether this property is available.

8.13.3.2 bool [onOff](#)

Whether this property is on or off.

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

- [FlyCapture2Defs.h](#)

8.14 Error Class Reference

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

Public Member Functions

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

8.14.1 Detailed Description

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

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

8.14.2 Constructor & Destructor Documentation

8.14.2.1 [Error](#) ()

Default constructor.

8.14.2.2 `Error (const Error & error)`

Copy constructor.

8.14.2.3 `virtual ~Error () [virtual]`

Default destructor.

8.14.3 Member Function Documentation

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

Retrieve the support information.

It is not implemented in this release.

Returns

A string containing support information.

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

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

Returns

A string with the build date and time.

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

Get the error which caused this error.

Returns

An error object representing the cause of this error.

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

Retrieve the top level description of the error that occurred.

Returns

A string with the error description.

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

Retrieve the source filename where the error originated.

Returns

A string with the file name.

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

Retrieve the line number where the error originated.

Returns

The line number.

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

Retrieve the ErrorType of the error.

Returns

The ErrorType of the error.

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

Inequality operator.

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

Inequality operator.

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

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

Assignment operator.

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

Equality operator.

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

Equality operator.

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

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

Print a formatted log trace to stderr.

8.14.4 Friends And Related Function Documentation

8.14.4.1 `friend class InternalError` [friend]

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

- [Error.h](#)

8.15 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.

8.15.1 Member Data Documentation

8.15.1.1 `void* EventData`

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

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

8.15.1.2 `size_t EventDataSize`

The size of the structure pointed to by EventData.

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

8.15.1.3 `long long unsigned EventID`

The device register which EventName maps to.

Provides an alternate means of indexing into different event types.

8.15.1.4 `const char* EventName`

The event name used to register the event.

Provided so the user knows which event triggered the callback.

8.15.1.5 `long long unsigned EventTimestamp`

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

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

8.15.1.6 `void* EventUserData`

Pointer to the user-supplied data struct.

8.15.1.7 `size_t EventUserDataSize`

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

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

- [FlyCapture2Defs.h](#)

8.16 EventOptions Struct Reference

Options for enabling device event registration.

Public Attributes

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

8.16.1 Detailed Description

Options for enabling device event registration.

8.16.2 Member Data Documentation

8.16.2.1 [CameraEventCallback EventCallbackFcn](#)

Callback function pointer.

8.16.2.2 `const char*` [EventName](#)

Event name to register.

8.16.2.3 `const void*` [EventUserData](#)

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

8.16.2.4 `size_t` [EventUserDataSize](#)

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

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

- [FlyCapture2Defs.h](#)

8.17 FC2Config Struct Reference

Configuration for a camera.

Public Member Functions

- [FC2Config](#) ()

Public Attributes

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

8.17.1 Detailed Description

Configuration for a camera.

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

8.17.2 Constructor & Destructor Documentation

8.17.2.1 [FC2Config](#) () `[inline]`

8.17.3 Member Data Documentation

8.17.3.1 [BusSpeed](#) [asyncBusSpeed](#)

Asynchronous bus speed.

8.17.3.2 **BandwidthAllocation** bandwidthAllocation

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

8.17.3.3 **GrabMode** grabMode

Grab mode for the camera.

The default is DROP_FRAMES.

8.17.3.4 **int** grabTimeout

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

8.17.3.5 **bool** highPerformanceRetrieveBuffer

This parameter enables RetrieveBuffer to run in high performance mode.

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

8.17.3.6 **BusSpeed** isochBusSpeed

Isochronous bus speed.

8.17.3.7 **unsigned int** minNumImageNotifications

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

Read-only value.

8.17.3.8 **unsigned int** numBuffers

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

8.17.3.9 unsigned int numImageNotifications

Number of notifications per image.

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

There are 4 general scenarios:

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

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

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

8.17.3.10 unsigned int registerTimeout

Register read/write timeout value, in microseconds.

The default value is dependent on the interface type.

8.17.3.11 unsigned int registerTimeoutRetries

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

The default value is 0.

8.17.3.12 unsigned int reserved[16]

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

8.18 FC2Version Struct Reference

The current version of the library.

Public Attributes

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

8.18.1 Detailed Description

The current version of the library.

8.18.2 Member Data Documentation

8.18.2.1 unsigned int build

Build version number.

8.18.2.2 unsigned int major

Major version number.

8.18.2.3 unsigned int minor

Minor version number.

8.18.2.4 unsigned int type

Type version number.

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

- [FlyCapture2Defs.h](#)

8.19 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](#) ()

8.19.1 Constructor & Destructor Documentation

8.19.1.1 [WRAPPER_API FlyCapture3ApiGuiWrapper](#) (void)

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

8.19.2 Member Function Documentation

8.19.2.1 [WRAPPER_API](#) void [ConnectGUILibrary](#) ([FlyCapture2::GCCamera](#) & *camera*)

8.19.2.2 [WRAPPER_API](#) void [DisconnectGUILibrary](#) ()

8.19.2.3 [WRAPPER_API](#) std::list<std::string> [GetControlNameList](#) ()

8.19.2.4 [WRAPPER_API](#) std::list<std::string> [GetDialogNameList](#) ()

8.19.2.5 [WRAPPER_API](#) int [GetNumDialogs](#) ()

8.19.2.6 [WRAPPER_API](#) int [GetNumOfControls](#) ()

8.19.2.7 [WRAPPER_API](#) void [ShowCameraSelectionDialog](#) ()

8.19.2.8 [WRAPPER_API](#) void [ShowDialogByIndex](#) (int *index*)

8.19.2.9 [WRAPPER_API](#) void [ShowDialogByName](#) (std::string *dialogName*)

8.19.2.10 [WRAPPER_API](#) void [ShowPropertyGridDialog](#) ()

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

- [FlyCapture3ApiGuiWrapper.h](#)

8.20 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.

8.20.1 Detailed Description

Format 7 image settings.

8.20.2 Constructor & Destructor Documentation

8.20.2.1 [Format7ImageSettings](#) () `[inline]`

8.20.3 Member Data Documentation

8.20.3.1 unsigned int `height`

Height of image.

8.20.3.2 **Mode** `mode`

Format 7 mode.

8.20.3.3 unsigned int offsetX

Horizontal image offset.

8.20.3.4 unsigned int offsetY

Vertical image offset.

8.20.3.5 PixelFormat pixelFormat

Pixel format of image.

8.20.3.6 unsigned int reserved[8]

Reserved for future use.

8.20.3.7 unsigned int width

Width of image.

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

- [FlyCapture2Defs.h](#)

8.21 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.

8.21.1 Detailed Description

Format 7 information for a single mode.

8.21.2 Constructor & Destructor Documentation

8.21.2.1 `Format7Info ()` `[inline]`

8.21.3 Member Data Documentation

8.21.3.1 unsigned int imageHStepSize

Horizontal step size for the image.

8.21.3.2 unsigned int imageVStepSize

Vertical step size for the image.

8.21.3.3 unsigned int maxHeight

Maximum image height.

8.21.3.4 unsigned int maxPacketSize

Maximum packet size in bytes for current mode.

8.21.3.5 unsigned int maxWidth

Maximum image width.

8.21.3.6 unsigned int minPacketSize

Minimum packet size in bytes for current mode.

8.21.3.7 Mode mode

Format 7 mode.

8.21.3.8 unsigned int offsetHStepSize

Horizontal step size for the offset.

8.21.3.9 unsigned int offsetVStepSize

Vertical step size for the offset.

8.21.3.10 unsigned int packetSize

Current packet size in bytes.

8.21.3.11 float percentage

Current packet size as a percentage of maximum packet size.

8.21.3.12 unsigned int pixelFormatBitField

Supported pixel formats in a bit field.

8.21.3.13 unsigned int reserved[16]

Reserved for future use.

8.21.3.14 unsigned int vendorPixelFormatBitField

Vendor unique pixel formats in a bit field.

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

- [FlyCapture2Defs.h](#)

8.22 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.

8.22.1 Detailed Description

Format 7 packet information.

8.22.2 Constructor & Destructor Documentation

8.22.2.1 [Format7PacketInfo](#) () `[inline]`

8.22.3 Member Data Documentation

8.22.3.1 unsigned int [maxBytesPerPacket](#)

Maximum bytes per packet.

8.22.3.2 unsigned int recommendedBytesPerPacket

Recommended bytes per packet.

8.22.3.3 unsigned int reserved[8]

Reserved for future use.

8.22.3.4 unsigned int unitBytesPerPacket

Minimum bytes per packet.

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

- [FlyCapture2Defs.h](#)

8.23 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 currently connected to a physical camera.

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

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

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

Starts isochronous image capture.

- virtual [Error RetrieveBuffer](#) ([Image](#) *pImage)
Retrieves the the next image object containing the next image.
- virtual [Error StopCapture](#) ()
Stops isochronous image transfer and cleans up all associated resources.
- virtual [Error WaitForBufferEvent](#) ([Image](#) *pImage, unsigned int eventNumber)
Retrieves the next image event containing the next part of the image.
- virtual [Error SetUserBuffers](#) (unsigned char *const pMemBuffers, int size, int numBuffers)
Specify user allocated buffers to use as image data buffers.
- virtual [Error GetConfiguration](#) ([FC2Config](#) *pConfig)
Get the configuration associated with the camera object.
- virtual [Error SetConfiguration](#) (const [FC2Config](#) *pConfig)
Set the configuration associated with the camera object.
- virtual [Error GetCameraInfo](#) ([CameraInfo](#) *pCameraInfo)
Retrieves information from the camera such as serial number, model name and other camera information.
- virtual [Error GetPropertyInfo](#) ([PropertyInfo](#) *pPropInfo)
Retrieves information about the specified camera property.
- virtual [Error GetProperty](#) ([Property](#) *pProp)
Reads the settings for the specified property from the camera.
- virtual [Error SetProperty](#) (const [Property](#) *pProp, bool broadcast=false)
Writes the settings for the specified property to the camera.
- virtual [Error GetGPIOPinDirection](#) (unsigned int pin, unsigned int *pDirection)
Get the GPIO pin direction for the specified pin.
- virtual [Error SetGPIOPinDirection](#) (unsigned int pin, unsigned int direction, bool broadcast=false)
Set the GPIO pin direction for the specified pin.
- virtual [Error 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 [Image↔EventCallback](#) *pCallbackFns=NULL, const void **pCallbackDataArray=NULL)
- static const char * [GetRegisterString](#) (unsigned int registerVal)

Protected Member Functions

- void [TestGainNode](#) ()

Protected Attributes

- [BusManager](#) m_busMgr

8.23.1 Constructor & Destructor Documentation

8.23.1.1 `GCCamera (void)`

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

8.23.2 Member Function Documentation

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

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

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

Implements [CameraBase](#).

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

8.23.2.3 `virtual Error Disconnect ()` [virtual]

Disconnects the camera object from the camera.

This allows another physical camera to be connected to the camera object.

See also

[Connect\(\)](#)

Returns

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

Implements [CameraBase](#).

8.23.2.4 `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](#).

8.23.2.5 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](#).

8.23.2.6 std::string GCCamera::GetXML ()**8.23.2.7 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](#).

8.23.2.8 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](#).

8.23.2.9 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](#).

8.23.2.10 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](#).

8.23.2.11 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](#).

8.23.2.12 `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](#).

8.23.2.13 `InterfaceType GetInterfaceType ()`

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

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

Parameters

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

Returns

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

Implements [CameraBase](#).

8.23.2.15 `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](#).

8.23.2.16 `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 $\log_2(\text{numEntries})$ for the inputBitDepth.

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

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

Retrieve the current memory channel from the camera.

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

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

See also

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

Returns

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

Implements [CameraBase](#).

8.23.2.19 `::GenApi::INodeMap* GetNodeMap ()`

8.23.2.20 `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](#).

8.23.2.21 `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](#).

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

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

Implements [CameraBase](#).

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

Retrieve current strobe settings from the camera.

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

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

8.23.2.25 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](#).

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

Retrieve current trigger delay settings from the camera.

Parameters

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

See also

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

[GetTriggerDelayInfo\(\)](#)
[SetTriggerDelay\(\)](#)

Returns

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

Implements [CameraBase](#).

8.23.2.27 `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

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

Returns

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

Implements [CameraBase](#).

8.23.2.28 `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

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

Returns

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

Implements [CameraBase](#).

8.23.2.29 `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](#).

8.23.2.30 `virtual bool IsConnected ()` [virtual]

Checks if the camera object is currently connected to a physical camera.

See also

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

Returns

Whether the camera object is connected to a physical camera.

Implements [CameraBase](#).

8.23.2.31 `virtual Error ReadGVCPMemory (unsigned int address, unsigned char * pBuffer, unsigned int length)` [virtual]

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

8.23.2.33 `virtual Error ReadGVCPRegisterBlock (unsigned int address, unsigned int * pBuffer, unsigned int length)` [virtual]

8.23.2.34 `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](#).

8.23.2.35 `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](#).

8.23.2.36 `virtual Error ResetStats () [virtual]`

Implements [CameraBase](#).

8.23.2.37 `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](#).

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

Retrieves the the next image object containing the next image.

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

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

8.23.2.39 `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](#).

8.23.2.40 `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](#).

8.23.2.41 `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](#).

8.23.2.42 **Error** SetCamera (CameraBase * camera)

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

8.23.2.44 **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](#).

8.23.2.45 **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](#).

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

Set the GPIO pin direction for the specified pin.

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

Parameters

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

See also

[GetGPIOPinDirection\(\)](#)

Returns

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

Implements [CameraBase](#).

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

Set the LUT channel settings to the camera.

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

8.23.2.48 `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](#).

8.23.2.49 `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>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](#).

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

Set the specified trigger delay settings to the camera.

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

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

Set the specified trigger settings to the camera.

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

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

See also

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

Returns

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

Implements [CameraBase](#).

8.23.2.53 `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](#).

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

8.23.2.55 `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](#).

8.23.2.56 `void TestGainNode ()` `[protected]`

8.23.2.57 `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>eventNumber</i>	The event number to wait for.

See also

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

Returns

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

Implements [CameraBase](#).

8.23.2.58 `virtual Error WriteGVCPMemory (unsigned int address, const unsigned char * pBuffer, unsigned int length)` `[virtual]`

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

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

8.23.2.61 `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](#).

8.23.2.62 **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](#).

8.23.3 Member Data Documentation

8.23.3.1 **BusManager m_busMgr** [protected]

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

- [GCCamera.h](#)

8.24 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 currently connected to a physical camera.
- virtual [Error SetCallback](#) ([ImageEventCallback](#) callbackFn, const void *pCallbackData=NULL)
Sets the callback data to be used on completion of image transfer.
- virtual [Error StartCapture](#) ([ImageEventCallback](#) callbackFn=NULL, const void *pCallbackData=NULL)
Starts isochronous image capture.
- virtual [Error RetrieveBuffer](#) ([Image](#) *pImage)
Retrieves the the next image object containing the next image.
- virtual [Error StopCapture](#) ()
Stops isochronous image transfer and cleans up all associated resources.
- virtual [Error WaitForBufferEvent](#) ([Image](#) *pImage, unsigned int eventNumber)
Retrieves the next image event containing the next part of the image.
- virtual [Error SetUserBuffers](#) (unsigned char *const pMemBuffers, int size, int numBuffers)
Specify user allocated buffers to use as image data buffers.
- virtual [Error GetConfiguration](#) ([FC2Config](#) *pConfig)
Get the configuration associated with the camera object.
- virtual [Error SetConfiguration](#) (const [FC2Config](#) *pConfig)
Set the configuration associated with the camera object.
- virtual [Error GetCameraInfo](#) ([CameraInfo](#) *pCameraInfo)
Retrieves information from the camera such as serial number, model name and other camera information.
- virtual [Error GetPropertyInfo](#) ([PropertyInfo](#) *pPropInfo)
Retrieves information about the specified camera property.
- virtual [Error GetProperty](#) ([Property](#) *pProp)
Reads the settings for the specified property from the camera.
- virtual [Error SetProperty](#) (const [Property](#) *pProp, bool broadcast=false)
Writes the settings for the specified property to the camera.
- virtual [Error GetGPIOPinDirection](#) (unsigned int pin, unsigned int *pDirection)
Get the GPIO pin direction for the specified pin.
- virtual [Error SetGPIOPinDirection](#) (unsigned int pin, unsigned int direction, bool broadcast=false)
Set the GPIO pin direction for the specified pin.
- virtual [Error 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 **pCallbackDataArray=NULL)
StartSyncCapture() with GigE Cameras is not supported.
- static const char * [GetRegisterString](#) (unsigned int registerVal)

GVCP Register Operation

These functions deal with GVCP register operation on the camera.

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

GigE property manipulation

These functions deal with GigE properties.

- virtual [Error GetGigEProperty](#) (GigEProperty *pGigEProp)
Get the specified GigEProperty.
- virtual [Error SetGigEProperty](#) (const GigEProperty *pGigEProp)
Set the specified GigEProperty.
- virtual [Error DiscoverGigEPacketSize](#) (unsigned int *packetSize)
Discover the largest packet size that works for the network link between the PC and the camera.

GigE image settings

These functions deal with GigE image setting.

- virtual [Error QueryGigEImagingMode](#) ([Mode](#) mode, bool *isSupported)
Check if the particular imaging mode is supported by the camera.
- virtual [Error GetGigEImagingMode](#) ([Mode](#) *mode)
Get the current imaging mode on the camera.
- virtual [Error SetGigEImagingMode](#) ([Mode](#) mode)
Set the current imaging mode to the camera.
- virtual [Error GetGigEImageSettingsInfo](#) ([GigEImageSettingsInfo](#) *pInfo)
Get information about the image settings possible on the camera.
- virtual [Error GetGigEImageSettings](#) ([GigEImageSettings](#) *pImageSettings)
Get the current image settings on the camera.
- virtual [Error SetGigEImageSettings](#) (const [GigEImageSettings](#) *pImageSettings)
Set the image settings specified to the camera.

GigE image binning settings

These functions deal with GigE image binning settings.

- virtual [Error GetGigEImageBinningSettings](#) (unsigned int *horzBinningValue, unsigned int *vertBinningValue)
Get the current binning settings on the camera.
- virtual [Error SetGigEImageBinningSettings](#) (unsigned int horzBinningValue, unsigned int vertBinningValue)
Set the specified binning values to the camera.

GigE image stream configuration

These functions deal with GigE image stream configuration.

- virtual [Error GetNumStreamChannels](#) (unsigned int *numChannels)
Get the number of stream channels present on the camera.
- virtual [Error GetGigEStreamChannelInfo](#) (unsigned int channel, [GigEStreamChannel](#) *pChannel)
Get the stream channel information for the specified channel.
- virtual [Error SetGigEStreamChannelInfo](#) (unsigned int channel, [GigEStreamChannel](#) *pChannel)
Set the stream channel information for the specified channel.
- virtual [Error GetGigEConfig](#) ([GigEConfig](#) *pGigEConfig)
Get the current gige config on the camera.
- virtual [Error SetGigEConfig](#) (const [GigEConfig](#) *pGigEConfig)
Set the gige config specified to the camera.

Additional Inherited Members

8.24.1 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.

8.24.2 Constructor & Destructor Documentation

8.24.2.1 GigE Camera ()

Default constructor.

8.24.2.2 virtual ~GigE Camera () [virtual]

Default destructor.

8.24.3 Member Function Documentation

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

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

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

Implements [CameraBase](#).

8.24.3.2 virtual Error DeregisterAllEvents (void) [virtual]

Implements [CameraBase](#).

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

Implements [CameraBase](#).

8.24.3.4 virtual Error Disconnect () [virtual]

Disconnects the camera object from the camera.

This allows another physical camera to be connected to the camera object.

See also

[Connect\(\)](#)

Returns

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

Implements [CameraBase](#).

8.24.3.5 virtual Error DiscoverGigE PacketSize (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.

8.24.3.6 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](#).

8.24.3.7 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](#).

8.24.3.8 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](#).

8.24.3.9 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](#).

8.24.3.10 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](#).

8.24.3.11 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](#).

8.24.3.12 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](#).

8.24.3.13 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.

8.24.3.14 virtual Error GetGigEImageBinningSettings (unsigned int * *horzBinningValue*, unsigned int * *vertBinningValue*) [virtual]

Get the current binning settings on the camera.

Parameters

<i>horzBinningValue</i>	Current horizontal binning value.
<i>vertBinningValue</i>	Current vertical binning value.

Returns

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

8.24.3.15 virtual Error GetGigEImageSettings (GigEImageSettings * pImageSettings) [virtual]

Get the current image settings on the camera.

Parameters

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

Returns

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

8.24.3.16 virtual Error GetGigEImageSettingsInfo (GigEImageSettingsInfo * pInfo) [virtual]

Get information about the image settings possible on the camera.

Parameters

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

Returns

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

8.24.3.17 virtual Error GetGigEImagingMode (Mode * mode) [virtual]

Get the current imaging mode on the camera.

Parameters

<i>mode</i>	Current imaging mode on the camera.
-------------	-------------------------------------

Returns

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

8.24.3.18 **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.

8.24.3.19 **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.

8.24.3.20 **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](#).

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

Returns

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

Implements [CameraBase](#).

8.24.3.22 `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](#).

8.24.3.23 `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 $\log_2(\text{numEntries})$ for the inputBitDepth.

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

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

Retrieve the current memory channel from the camera.

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

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

See also

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

Returns

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

Implements [CameraBase](#).

8.24.3.26 `virtual Error GetNumStreamChannels (unsigned int * numChannels)` `[virtual]`

Get the number of stream channels present on the camera.

Parameters

<i>numChannels</i>	Number of stream channels present.
--------------------	------------------------------------

Returns

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

8.24.3.27 `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](#).

8.24.3.28 `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](#).

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

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

Implements [CameraBase](#).

8.24.3.31 `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](#).

8.24.3.32 `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](#).

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

Retrieve current trigger delay settings from the camera.

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

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

Retrieve trigger delay information from the camera.

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

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

Retrieve current trigger settings from the camera.

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

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

Retrieve trigger information from the camera.

Parameters

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

See also

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

[GetTriggerDelay\(\)](#)
[SetTriggerDelay\(\)](#)

Returns

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

Implements [CameraBase](#).

8.24.3.37 `virtual bool IsConnected () [virtual]`

Checks if the camera object is currently connected to a physical camera.

See also

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

Returns

Whether the camera object is connected to a physical camera.

Implements [CameraBase](#).

8.24.3.38 `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.

8.24.3.39 `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.

8.24.3.40 `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.

8.24.3.41 `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.

8.24.3.42 `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](#).

8.24.3.43 `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](#).

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

Implements [CameraBase](#).

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

Implements [CameraBase](#).

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

Implements [CameraBase](#).

8.24.3.47 `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](#).

8.24.3.48 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](#).

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

Save the current settings to the specified current memory channel.

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

8.24.3.50 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](#).

8.24.3.51 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](#).

8.24.3.52 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](#).

8.24.3.53 `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](#).

8.24.3.54 `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.

8.24.3.55 `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>horzBinningValue</i>	Horizontal binning value.
<i>vertBinningValue</i>	Vertical binning value.

Returns

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

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

Set the image settings specified to the camera.

Parameters

<i>pImageSettings</i>	Image settings to set to camera.
-----------------------	--

Returns

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

8.24.3.57 `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.

8.24.3.58 `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.

8.24.3.59 `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.

8.24.3.60 `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](#).

8.24.3.61 `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](#).

8.24.3.62 `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](#).

8.24.3.63 `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](#).

8.24.3.64 `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

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

Returns

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

Implements [CameraBase](#).

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

Set the specified trigger settings to the camera.

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

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

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

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

Starts isochronous image capture.

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

Parameters

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

See also

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

Returns

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

Implements [CameraBase](#).

8.24.3.68 `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.

8.24.3.69 `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](#).

8.24.3.70 `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>eventNumber</i>	The event number to wait for.

See also

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

Returns

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

Implements [CameraBase](#).

8.24.3.71 `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.

8.24.3.72 `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.

8.24.3.73 `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.

8.24.3.74 `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](#).

8.24.3.75 `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](#)

8.25 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.

8.25.1 Detailed Description

Configuration for a GigE camera.

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

8.25.2 Constructor & Destructor Documentation

8.25.2.1 [GigEConfig](#) () `[inline]`

8.25.3 Member Data Documentation

8.25.3.1 bool [enablePacketResend](#)

Turn on/off packet resend functionality.

8.25.3.2 unsigned int registerTimeout

Register read/write timeout value, in microseconds.

The default value is dependent on the interface type.

8.25.3.3 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](#)

8.26 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.

8.26.1 Detailed Description

[Image](#) settings for a GigE camera.

8.26.2 Constructor & Destructor Documentation

8.26.2.1 GigElImageSettings () `[inline]`

8.26.3 Member Data Documentation

8.26.3.1 unsigned int height

Height of image.

8.26.3.2 unsigned int offsetX

Horizontal image offset.

8.26.3.3 unsigned int offsetY

Vertical image offset.

8.26.3.4 PixelFormat pixelFormat

Pixel format of image.

8.26.3.5 unsigned int reserved[8]

Reserved for future use.

8.26.3.6 unsigned int width

Width of image.

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

- [FlyCapture2Defs.h](#)

8.27 GigElImageSettingsInfo Struct Reference

Format 7 information for a single mode.

Public Member Functions

- [GigElImageSettingsInfo \(\)](#)

Public Attributes

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

8.27.1 Detailed Description

Format 7 information for a single mode.

8.27.2 Constructor & Destructor Documentation

8.27.2.1 `GigEImageSettingsInfo ()` `[inline]`

8.27.3 Member Data Documentation

8.27.3.1 unsigned int `imageHStepSize`

Horizontal step size for the image.

8.27.3.2 unsigned int `imageVStepSize`

Vertical step size for the image.

8.27.3.3 unsigned int `maxHeight`

Maximum image height.

8.27.3.4 unsigned int `maxWidth`

Maximum image width.

8.27.3.5 unsigned int offsetHStepSize

Horizontal step size for the offset.

8.27.3.6 unsigned int offsetVStepSize

Vertical step size for the offset.

8.27.3.7 unsigned int pixelFormatBitField

Supported pixel formats in a bit field.

8.27.3.8 unsigned int reserved[16]

Reserved for future use.

8.27.3.9 unsigned int vendorPixelFormatBitField

Vendor unique pixel formats in a bit field.

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

- [FlyCapture2Defs.h](#)

8.28 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.

8.28.1 Detailed Description

A GigE property.

8.28.2 Member Data Documentation

8.28.2.1 bool isReadable

Whether the property is readable.

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

8.28.2.2 bool isWritable

Whether the property is writable.

8.28.2.3 unsigned int max

Maximum value.

8.28.2.4 unsigned int min

Minimum value.

8.28.2.5 GigEPropertyType propType

The type of property.

8.28.2.6 unsigned int value

Current value.

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

- [FlyCapture2Defs.h](#)

8.29 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.

8.29.1 Detailed Description

Information about a single GigE stream channel.

8.29.2 Constructor & Destructor Documentation

8.29.2.1 [GigEStreamChannel](#) () `[inline]`

8.29.3 Member Data Documentation

8.29.3.1 [IPAddress](#) [destinationIpAddress](#)

Destination IP address.

It can be a multicast or unicast address.

8.29.3.2 [bool](#) [doNotFragment](#)

Disable IP fragmentation of packets.

8.29.3.3 [unsigned int](#) [hostPort](#)

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

8.29.3.4 unsigned int interPacketDelay

Inter packet delay, in timestamp counter units.

8.29.3.5 unsigned int networkInterfaceIndex

Network interface index used (or to use).

8.29.3.6 unsigned int packetSize

Packet size, in bytes.

8.29.3.7 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](#)

8.30 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.

8.30.1 Detailed Description

Options for saving H264 files.

8.30.2 Constructor & Destructor Documentation

8.30.2.1 `H264Option()` `[inline]`

8.30.3 Member Data Documentation

8.30.3.1 `unsigned int bitrate`

Bitrate to encode at.

8.30.3.2 `float frameRate`

Frame rate of the stream.

8.30.3.3 `unsigned int height`

Height of source image.

8.30.3.4 `unsigned int reserved[256]`

Reserved for future use.

8.30.3.5 `unsigned int width`

Width of source image.

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

- [FlyCapture2Defs.h](#)

8.31 Image Class Reference

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

Public Member Functions

- [Image](#) ()
Default constructor.
- [Image](#) (unsigned int rows, unsigned int cols, unsigned int stride, unsigned char *pData, unsigned int dataSize, [PixelFormat](#) format, [BayerTileFormat](#) bayerFormat=[NONE](#))
Construct an [Image](#) object with the specified arguments.
- [Image](#) (unsigned int rows, unsigned int cols, unsigned int stride, unsigned char *pData, unsigned int dataSize, unsigned int receivedDataSize, [PixelFormat](#) format, [BayerTileFormat](#) bayerFormat=[NONE](#))
Construct an [Image](#) object with the specified arguments.
- [Image](#) (unsigned char *pData, unsigned int dataSize)
Construct an [Image](#) object with the specified arguments.
- [Image](#) (unsigned int rows, unsigned int cols, [PixelFormat](#) format, [BayerTileFormat](#) bayerFormat=[NONE](#))
Construct an [Image](#) object with the specified arguments.
- [Image](#) (const [Image](#) &image)
Copy constructor.
- virtual [~Image](#) ()
Default destructor.
- virtual [Image](#) & [operator=](#) (const [Image](#) &image)
Assignment operator.
- virtual unsigned char * [operator\[\]](#) (unsigned int index)
Indexing operator.
- virtual unsigned char * [operator\(\)](#) (unsigned int row, unsigned int col)
Indexing operator.
- virtual [Error DeepCopy](#) (const [Image](#) *pImage)
Perform a deep copy of the [Image](#).
- virtual [Error SetDimensions](#) (unsigned int rows, unsigned int cols, unsigned int stride, [PixelFormat](#) pixelFormat, [BayerTileFormat](#) bayerFormat)
Sets the dimensions of the image object.
- virtual [Error SetData](#) (const unsigned char *pData, unsigned int dataSize)
Set the data of the [Image](#) object.
- virtual [Error SetBlockId](#) (const unsigned int blockId)
Set the block id of the [Image](#) object.
- virtual unsigned int [GetBlockId](#) ()
get the block id of the [Image](#) object.
- virtual [PixelFormat](#) [GetPixelFormat](#) () const
Get the current pixel format.
- virtual [ColorProcessingAlgorithm](#) [GetColorProcessing](#) () const
Get the current color processing algorithm.
- virtual [Error SetColorProcessing](#) ([ColorProcessingAlgorithm](#) colorProc)
Set the color processing algorithm.
- virtual unsigned int [GetCols](#) () const
Get the number of columns in the image.
- virtual unsigned int [GetRows](#) () const
Get the number of rows in the image.
- virtual unsigned int [GetStride](#) () const
Get the stride in the image.
- virtual unsigned int [GetBitsPerPixel](#) () const
Get the bits per pixel of the image.
- virtual [BayerTileFormat](#) [GetBayerTileFormat](#) () const
Get the Bayer tile format of the image.

- virtual unsigned int [GetDataSize](#) () const
Get the size of the buffer associated with the image, in bytes.
- virtual unsigned int [GetReceivedDataSize](#) () const
Get the size of the compressed data, in bytes.
- virtual void [GetDimensions](#) (unsigned int *pRows, unsigned int *pCols=NULL, unsigned int *pStride=NULL, [PixelFormat](#) *pPixelFormat=NULL, [BayerTileFormat](#) *pBayerFormat=NULL) const
Get the image dimensions associated with the image.
- virtual unsigned char * [GetData](#) ()
Get a pointer to the data associated with the image.
- virtual unsigned char *const [GetData](#) () const
- virtual [ImageMetadata](#) [GetMetadata](#) () const
Get the metadata associated with the image.
- virtual [Error](#) [CalculateStatistics](#) ([ImageStatistics](#) *pStatistics)
Calculate statistics associated with the image.
- virtual [TimeStamp](#) [GetTimeStamp](#) () const
Get the timestamp data associated with the image.
- virtual [Error](#) [Save](#) (const char *pFilename, [ImageFileFormat](#) format=FROM_FILE_EXT)
Save the image to the specified file name with the file format specified.
- virtual [Error](#) [Save](#) (const char *pFilename, [PNGOption](#) *pOption)
Save the image to the specified file name with the options specified.
- virtual [Error](#) [Save](#) (const char *pFilename, [PPMOption](#) *pOption)
Save the image to the specified file name with the options specified.
- virtual [Error](#) [Save](#) (const char *pFilename, [PGMOption](#) *pOption)
Save the image to the specified file name with the options specified.
- virtual [Error](#) [Save](#) (const char *pFilename, [TIFFOption](#) *pOption)
Save the image to the specified file name with the options specified.
- virtual [Error](#) [Save](#) (const char *pFilename, [JPEGOption](#) *pOption)
Save the image to the specified file name with the options specified.
- virtual [Error](#) [Save](#) (const char *pFilename, [JPG2Option](#) *pOption)
Save the image to the specified file name with the options specified.
- virtual [Error](#) [Save](#) (const char *pFilename, [BMPOption](#) *pOption)
Save the image to the specified file name with the options specified.
- virtual [Error](#) [Convert](#) ([PixelFormat](#) format, [Image](#) *pDestImage) const
Converts the current image buffer to the specified output format and stores the result in the specified image.
- virtual [Error](#) [Convert](#) ([Image](#) *pDestImage) const
Converts the current image buffer to the specified output format and stores the result in the specified image.
- virtual [Error](#) [ReleaseBuffer](#) ()
Release the buffer associated with the [Image](#).

Static Public Member Functions

- static [Error](#) [SetDefaultColorProcessing](#) ([ColorProcessingAlgorithm](#) defaultMethod)
Set the default color processing algorithm.
- static [ColorProcessingAlgorithm](#) [GetDefaultColorProcessing](#) ()
Get the default color processing algorithm.
- static [Error](#) [SetDefaultOutputFormat](#) ([PixelFormat](#) format)
Set the default output pixel format.
- static [PixelFormat](#) [GetDefaultOutputFormat](#) ()
Get the default output pixel format.
- static unsigned int [DetermineBitsPerPixel](#) ([PixelFormat](#) format)
Calculate the bits per pixel for the specified pixel format.

Friends

- class [Iso](#)

8.31.1 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.

8.31.2 Constructor & Destructor Documentation

8.31.2.1 [Image](#) ()

Default constructor.

8.31.2.2 [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.

8.31.2.3 [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.
-------------	--------------------

Parameters

<i>cols</i>	Columns in the image.
<i>stride</i>	Stride of the image buffer.
<i>pData</i>	Pointer to the image buffer.
<i>dataSize</i>	Size of the image buffer.
<i>receivedDataSize</i>	Actual size of data.
<i>format</i>	Pixel format.
<i>bayerFormat</i>	Format of the Bayer tiled raw image.

8.31.2.4 `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.

8.31.2.5 `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.

8.31.2.6 `Image (const Image & image)`

Copy constructor.

Both images will point to the same image buffer internally.

8.31.2.7 `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 reused internally.

8.31.3 Member Function Documentation

8.31.3.1 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.

8.31.3.2 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.

8.31.3.3 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.

8.31.3.4 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.

8.31.3.5 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.

8.31.3.6 virtual BayerTileFormat GetBayerTileFormat () const [virtual]

Get the Bayer tile format of the image.

Returns

The Bayer tile format.

8.31.3.7 virtual unsigned int GetBitsPerPixel () const [virtual]

Get the bits per pixel of the image.

Returns

The bits per pixel.

8.31.3.8 virtual unsigned int GetBlockId () [virtual]

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

Returns

The blockId assigned to the image.

8.31.3.9 virtual ColorProcessingAlgorithm GetColorProcessing () const [virtual]

Get the current color processing algorithm.

See also

[SetColorProcessing\(\)](#)

Returns

The current color processing algorithm.

8.31.3.10 virtual unsigned int GetCols () const [virtual]

Get the number of columns in the image.

Returns

The number of columns.

8.31.3.11 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.

8.31.3.12 virtual unsigned char* const GetData () const [virtual]

8.31.3.13 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.

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

Get the default color processing algorithm.

See also

[SetDefaultColorProcessing\(\)](#)

Returns

The default color processing algorithm.

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

Get the default output pixel format.

See also

[SetDefaultOutputFormat\(\)](#)

Returns

The default pixel format.

8.31.3.16 **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.

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

Get the metadata associated with the image.

This includes embedded image information.

Returns

Metadata associated with the image.

8.31.3.18 virtual PixelFormat GetPixelFormat () const [virtual]

Get the current pixel format.

Returns

The current pixel format.

8.31.3.19 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.

8.31.3.20 virtual unsigned int GetRows () const [virtual]

Get the number of rows in the image.

Returns

The number of rows.

8.31.3.21 virtual unsigned int GetStride () const [virtual]

Get the stride in the image.

Returns

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

8.31.3.22 virtual TimeStamp GetTimeStamp () const [virtual]

Get the timestamp data associated with the image.

Returns

Timestamp data associated with the image.

8.31.3.23 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.

8.31.3.24 `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.
--------------	-------------------------

8.31.3.25 `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.

8.31.3.26 `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.

8.31.3.27 `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.

8.31.3.28 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.

8.31.3.29 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.

8.31.3.30 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.

8.31.3.31 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.

8.31.3.32 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.

8.31.3.33 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.

8.31.3.34 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.

8.31.3.35 `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.
----------------	-------------------------------------

8.31.3.36 `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.

8.31.3.37 `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.

8.31.3.38 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>defaultMethod</i>	The color processing algorithm to set.
----------------------	--

See also

[GetDefaultColorProcessing\(\)](#)

Returns

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

8.31.3.39 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.

8.31.3.40 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.

8.31.4 Friends And Related Function Documentation

8.31.4.1 friend class Iso `[friend]`

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

- [Image.h](#)

8.32 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.

8.32.1 Detailed Description

Metadata related to an image.

8.32.2 Constructor & Destructor Documentation

8.32.2.1 ImageMetadata () [inline]

8.32.3 Member Data Documentation

8.32.3.1 unsigned int embeddedBrightness

Embedded brightness.

8.32.3.2 unsigned int embeddedExposure

Embedded exposure.

8.32.3.3 unsigned int embeddedFrameCounter

Embedded frame counter.

8.32.3.4 unsigned int embeddedGain

Embedded gain.

8.32.3.5 unsigned int embeddedGPIOPinState

Embedded GPIO pin state.

8.32.3.6 unsigned int embeddedROIPosition

Embedded ROI position.

8.32.3.7 unsigned int embeddedShutter

Embedded shutter.

8.32.3.8 unsigned int embeddedStrobePattern

Embedded strobe pattern.

8.32.3.9 unsigned int embeddedTimeStamp

Embedded timestamp.

8.32.3.10 unsigned int embeddedWhiteBalance

Embedded white balance.

8.32.3.11 unsigned int reserved[31]

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

8.33 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](#)

8.33.1 Detailed Description

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

8.33.2 Member Enumeration Documentation

8.33.2.1 enum [StatisticsChannel](#)

Channels that allow statistics to be calculated.

Enumerator

GREY
RED
GREEN
BLUE
HUE
SATURATION
LIGHTNESS
NUM_STATISTICS_CHANNELS

8.33.3 Constructor & Destructor Documentation

8.33.3.1 ImageStatistics ()

Default constructor.

8.33.3.2 virtual ~ImageStatistics () [virtual]

Default destructor.

8.33.3.3 ImageStatistics (const ImageStatistics & other)

Copy constructor.

8.33.4 Member Function Documentation

8.33.4.1 Error DisableAll ()

Disable all channels.

Returns

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

8.33.4.2 Error EnableAll ()

Enable all channels.

Returns

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

8.33.4.3 Error EnableGreyOnly ()

Enable only the grey channel.

Returns

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

8.33.4.4 Error EnableHSLOnly ()

Enable only the HSL channels.

Returns

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

8.33.4.5 Error EnableRGBOnly ()

Enable only the RGB channels.

Returns

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

8.33.4.6 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.

8.33.4.7 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.

8.33.4.8 Error GetMean (**StatisticsChannel** *channel*, float * *pPixelValueMean*) const

Get the mean of the image.

Parameters

<i>channel</i>	The statistics channel.
<i>pPixelValueMean</i>	The mean of the image.

Returns

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

8.33.4.9 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.

8.33.4.10 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.

8.33.4.11 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.

8.33.4.12 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.

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

Assignment operator.

Parameters

<i>other</i>	The ImageStatistics object to copy from.
--------------	--

8.33.4.14 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.

8.33.5 Friends And Related Function Documentation

8.33.5.1 friend class ImageStatsCalculator [friend]

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

- [ImageStatistics.h](#)

8.34 Internal Class Reference

Static Public Member Functions

- static void * [GetInternal](#) (unsigned int index)

8.34.1 Member Function Documentation

8.34.1.1 static void* GetInternal (unsigned int *index*) [static]

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

- [Internal.h](#)

8.35 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]

8.35.1 Detailed Description

IPv4 address.

8.35.2 Constructor & Destructor Documentation

8.35.2.1 [IPAddress](#) () [inline]

8.35.2.2 [IPAddress](#) (unsigned int *ipAddressVal*) [inline]

8.35.3 Member Function Documentation

8.35.3.1 bool [operator!=](#) (const [IPAddress](#) & *address*) [inline]

Inequality operator.

8.35.3.2 bool [operator==](#) (const [IPAddress](#) & *address*) const [inline]

Equality operator.

8.35.4 Member Data Documentation

8.35.4.1 unsigned char [octets](#)[4]

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

- [FlyCapture2Defs.h](#)

8.36 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.

8.36.1 Detailed Description

Options for saving JPEG image.

8.36.2 Constructor & Destructor Documentation

8.36.2.1 [JPEGOption](#) () [inline]

8.36.3 Member Data Documentation

8.36.3.1 bool [progressive](#)

Whether to save as a progressive JPEG file.

8.36.3.2 unsigned int [quality](#)

JPEG image quality in range (0-100).

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

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

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

8.37 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.

8.37.1 Detailed Description

Options for saving JPEG2000 image.

8.37.2 Constructor & Destructor Documentation

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

8.37.3 Member Data Documentation

8.37.3.1 unsigned int [quality](#)

JPEG saving quality in range (1-512).

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

Reserved for future use.

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

- [FlyCapture2Defs.h](#)

8.38 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.

8.38.1 Detailed Description

Information about the camera's look up table.

8.38.2 Constructor & Destructor Documentation

8.38.2.1 [LUTData](#) () [inline]

8.38.3 Member Data Documentation

8.38.3.1 bool [enabled](#)

Flag indicating if LUT is enabled.

8.38.3.2 unsigned int [inputBitDepth](#)

The input bit depth of the LUT.

8.38.3.3 unsigned int [numBanks](#)

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

8.38.3.4 unsigned int numChannels

The number of LUT channels per bank available.

8.38.3.5 unsigned int numEntries

The number of entries in the LUT.

8.38.3.6 unsigned int outputBitDepth

The output bit depth of the LUT.

8.38.3.7 unsigned int reserved[8]

Reserved for future use.

8.38.3.8 bool supported

Flag indicating if LUT is supported.

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

- [FlyCapture2Defs.h](#)

8.39 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]

8.39.1 Detailed Description

MAC address.

8.39.2 Constructor & Destructor Documentation

8.39.2.1 **MACAddress** () `[inline]`

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

8.39.3 Member Function Documentation

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

Inequality operator.

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

Equality operator.

8.39.4 Member Data Documentation

8.39.4.1 unsigned char *octets*[6]

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

- [FlyCapture2Defs.h](#)

8.40 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]

8.40.1 Detailed Description

Options for saving MJPG files.

8.40.2 Constructor & Destructor Documentation

8.40.2.1 MJPGOption () `[inline]`

8.40.3 Member Data Documentation

8.40.3.1 float frameRate

Frame rate of the stream.

8.40.3.2 unsigned int quality

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

8.40.3.3 unsigned int reserved[256]

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

- [FlyCapture2Defs.h](#)

8.41 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.

8.41.1 Constructor & Destructor Documentation

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

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

8.41.2 Member Function Documentation

8.41.2.1 **GenICam::gcstring _GetDeviceName** ()

Get device name.

8.41.2.2 **INode* _GetNode** (**const** **GenICam::gcstring** & *key*)

Retrieves the node from the central map by name.

8.41.2.3 **void _GetNodes** (**NodeList_t** & *Nodes*)

Retrieves all nodes in the node map.

8.41.2.4 **void _InvalidateNodes** () **const**

Invalidates all nodes.

8.41.2.5 **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](#)

8.42 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.

8.42.1 Detailed Description

Options for saving PGM images.

8.42.2 Constructor & Destructor Documentation

8.42.2.1 PGMOption () [inline]

8.42.3 Member Data Documentation

8.42.3.1 bool binaryFile

Whether to save the PPM as a binary file.

8.42.3.2 unsigned int reserved[16]

Reserved for future use.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

8.43 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]

8.43.1 Detailed Description

A GUID to the camera.

It is used to uniquely identify a camera.

8.43.2 Constructor & Destructor Documentation

8.43.2.1 `PGRGuid ()` `[inline]`

Constructor.

8.43.3 Member Function Documentation

8.43.3.1 `bool operator!=(const PGRGuid & guid)` `[inline]`

Inequality operator.

8.43.3.2 `bool operator==(const PGRGuid & guid) const` `[inline]`

Equality operator.

8.43.4 Member Data Documentation

8.43.4.1 `unsigned int value[4]`

The documentation for this class was generated from the following file:

- [FlyCapture2Defs.h](#)

8.44 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.

8.44.1 Detailed Description

Options for saving PNG images.

8.44.2 Constructor & Destructor Documentation

8.44.2.1 PNGOption () [inline]

8.44.3 Member Data Documentation

8.44.3.1 unsigned int compressionLevel

Compression level (0-9).

0 is no compression, 9 is best compression.

8.44.3.2 bool interlaced

Whether to save the PNG as interlaced.

8.44.3.3 unsigned int reserved[16]

Reserved for future use.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

8.45 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.

8.45.1 Detailed Description

Options for saving PPM images.

8.45.2 Constructor & Destructor Documentation

8.45.2.1 `PPMOption()` `[inline]`

8.45.3 Member Data Documentation

8.45.3.1 bool `binaryFile`

Whether to save the PPM as a binary file.

8.45.3.2 unsigned int `reserved[16]`

Reserved for future use.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

8.46 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.

8.46.1 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

8.46.2 Constructor & Destructor Documentation

8.46.2.1 `Property ()` `[inline]`

8.46.2.2 `Property (PropertyType propType)` `[inline]`

8.46.3 Member Data Documentation

8.46.3.1 bool `absControl`

Flag controlling absolute mode (real world units) or non-absolute mode (camera internal units).

8.46.3.2 float absValue

Floating point value.

Used to configure properties in absolute mode.

8.46.3.3 bool autoManualMode

Flag controlling auto.

8.46.3.4 bool onePush

Flag controlling one push.

8.46.3.5 bool onOff

Flag controlling on/off.

8.46.3.6 bool present

Flag indicating if the property is present.

8.46.3.7 unsigned int reserved[8]

Reserved for future use.

8.46.3.8 PropertyType type

[Property](#) info type.

8.46.3.9 unsigned int valueA

Value A (integer).

Used to configure properties in non-absolute mode.

8.46.3.10 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](#)

8.47 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.

8.47.1 Detailed Description

Information about a specific camera property.

This structure is also also used as the [TriggerDelayInfo](#) structure.

8.47.2 Constructor & Destructor Documentation

8.47.2.1 **PropertyInfo** () `[inline]`

8.47.2.2 **PropertyInfo** (*PropertyType propType*) `[inline]`

8.47.3 Member Data Documentation

8.47.3.1 **float absMax**

Maximum value (as a floating point value).

8.47.3.2 **float absMin**

Minimum value (as a floating point value).

8.47.3.3 **bool absValSupported**

Flag indicating if absolute mode is supported.

8.47.3.4 **bool autoSupported**

Flag indicating if auto is supported.

8.47.3.5 **bool manualSupported**

Flag indicating if manual is supported.

8.47.3.6 **unsigned int max**

Maximum value (as an integer).

8.47.3.7 **unsigned int min**

Minimum value (as an integer).

8.47.3.8 **bool onePushSupported**

Flag indicating if one push is supported.

8.47.3.9 bool onOffSupported

Flag indicating if on/off is supported.

8.47.3.10 bool present

Flag indicating if the property is present.

8.47.3.11 char pUnitAbbr[sk_maxStringLength]

Abbreviated textual description of units.

8.47.3.12 char pUnits[sk_maxStringLength]

Textual description of units.

8.47.3.13 bool readOutSupported

Flag indicating if property value can be read out.

8.47.3.14 unsigned int reserved[8]

Reserved for future use.

8.47.3.15 PropertyType type

[Property](#) info type.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

8.48 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.

8.48.1 Detailed Description

A camera strobe.

8.48.2 Constructor & Destructor Documentation

8.48.2.1 `StrobeControl()` `[inline]`

8.48.3 Member Data Documentation

8.48.3.1 float `delay`

Signal delay (in ms).

8.48.3.2 float `duration`

Signal duration (in ms).

8.48.3.3 bool `onOff`

Flag controlling on/off.

8.48.3.4 unsigned int `polarity`

Signal polarity.

8.48.3.5 unsigned int `reserved`[8]

Reserved for future use.

8.48.3.6 unsigned int source

Source value.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

8.49 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.

8.49.1 Detailed Description

A camera strobe property.

8.49.2 Constructor & Destructor Documentation

8.49.2.1 [StrobelInfo](#) () [inline]

8.49.3 Member Data Documentation

8.49.3.1 float [maxValue](#)

Maximum value.

8.49.3.2 float minValue

Minimum value.

8.49.3.3 bool onOffSupported

Flag indicating if on/off is supported.

8.49.3.4 bool polaritySupported

Flag indicating if polarity is supported.

8.49.3.5 bool present

Presence of strobe.

8.49.3.6 bool readOutSupported

Flag indicating if strobe value can be read out.

8.49.3.7 unsigned int reserved[8]

Reserved for future use.

8.49.3.8 unsigned int source

Source value.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

8.50 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 \(\)](#)

8.50.1 Constructor & Destructor Documentation

8.50.1.1 **MULTISYNCLIBRARY_API** SyncManager ()

8.50.1.2 **MULTISYNCLIBRARY_API** ~SyncManager ()

8.50.2 Member Function Documentation

8.50.2.1 **MULTISYNCLIBRARY_API** bool DisableCrossPCSynchronization ()

8.50.2.2 **MULTISYNCLIBRARY_API** bool EnableCrossPCSynchronization ()

8.50.2.3 **MULTISYNCLIBRARY_API** PGRSyncMessage GetSyncStatus ()

8.50.2.4 **MULTISYNCLIBRARY_API** double GetTimeSinceSynced ()

8.50.2.5 **MULTISYNCLIBRARY_API** bool IsTimingBusConnected ()

8.50.2.6 **MULTISYNCLIBRARY_API** bool QueryCrossPCSynchronizationSetting ()

8.50.2.7 **MULTISYNCLIBRARY_API** PGRSyncError RescanMasterTimingBus ()

8.50.2.8 **MULTISYNCLIBRARY_API** PGRSyncError Start ()

8.50.2.9 **MULTISYNCLIBRARY_API** PGRSyncError Stop ()

The documentation for this class was generated from the following file:

- [MultiSyncLibraryDefs.h](#)

8.51 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.

8.51.1 Detailed Description

Description of the system.

8.51.2 Member Data Documentation

8.51.2.1 ByteOrder byteOrder

Byte order of the system.

8.51.2.2 char cpuDescription[sk_maxStringLength]

Detailed description of the CPU.

8.51.2.3 char driverList[sk_maxStringLength]

List of drivers used.

8.51.2.4 char gpuDescription[sk_maxStringLength]

Detailed description of the GPU.

8.51.2.5 char libraryList[sk_maxStringLength]

List of libraries used.

8.51.2.6 size_t numCpuCores

Number of cores on all CPUs on the system.

8.51.2.7 char osDescription[sk_maxStringLength]

Detailed description of the operating system.

8.51.2.8 OSType osType

Operating system type as described by OSType.

8.51.2.9 unsigned int reserved[16]

Reserved for future use.

8.51.2.10 size_t screenHeight

Screen resolution height in pixels.

8.51.2.11 size_t screenWidth

Screen resolution width in pixels.

8.51.2.12 size_t sysMemSize

Amount of memory available on the system.

The documentation for this struct was generated from the following file:

- [Utilities.h](#)

8.52 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.

8.52.1 Detailed Description

Options for saving TIFF images.

8.52.2 Member Enumeration Documentation

8.52.2.1 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.

8.52.3 Constructor & Destructor Documentation

8.52.3.1 TIFFOption () [inline]

8.52.4 Member Data Documentation

8.52.4.1 CompressionMethod compression

Compression method to use for encoding TIFF images.

8.52.4.2 unsigned int reserved[16]

Reserved for future use.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

8.53 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.

8.53.1 Detailed Description

Timestamp information.

8.53.2 Constructor & Destructor Documentation

8.53.2.1 `TimeStamp () [inline]`

8.53.3 Member Data Documentation

8.53.3.1 `unsigned int cycleCount`

1394 cycle time count.

8.53.3.2 `unsigned int cycleOffset`

1394 cycle time offset.

8.53.3.3 `unsigned int cycleSeconds`

1394 cycle time seconds.

8.53.3.4 `unsigned int microSeconds`

Microseconds.

8.53.3.5 `unsigned int reserved[8]`

Reserved for future use.

8.53.3.6 `long long seconds`

Seconds.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

8.54 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.

8.54.1 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.

8.54.2 Member Enumeration Documentation

8.54.2.1 enum NodeType

Type of node.

Enumerator

COMPUTER

BUS

CAMERA

NODE

8.54.2.2 enum PortType

Possible states of a port on a node.

Enumerator

NOT_CONNECTED

CONNECTED_TO_PARENT

CONNECTED_TO_CHILD

8.54.3 Constructor & Destructor Documentation

8.54.3.1 TopologyNode ()

Default constructor.

8.54.3.2 TopologyNode (PGRGuid *guid*, int *deviceId*, NodeType *nodeType*, InterfaceType *interfaceType*)

Constructor.

Parameters

<i>guid</i>	The PGRGuid of the node (if applicable).
<i>deviceId</i>	Device ID of the node.
<i>nodeType</i>	Type of the node.
<i>interfaceType</i>	Interface type of the node.

8.54.3.3 virtual ~TopologyNode () [virtual]

Default destructor.

8.54.3.4 TopologyNode (const TopologyNode & other)

Copy constructor.

8.54.4 Member Function Documentation

8.54.4.1 virtual void AddChild (TopologyNode *childNode*) [virtual]

Add the specified [TopologyNode](#) as a child of the node.

Parameters

<i>childNode</i>	The TopologyNode to add.
------------------	--

8.54.4.2 virtual void AddPortType (PortType *childPort*) [virtual]

Add the specified PortType as a port of the node.

Parameters

<i>childPort</i>	The port to add.
------------------	------------------

8.54.4.3 virtual bool AssignGuidToNode (PGRGuid *guid*, int *deviceId*) [virtual]

Assign a [PGRGuid](#) and device ID to the node.

Parameters

<i>guid</i>	PGRGuid to be assigned.
<i>deviceId</i>	Device ID to be assigned.

Returns

Whether the data was successfully set to the node.

8.54.4.4 virtual bool AssignGuidToNode (PGRGuid *guid*, int *deviceId*, NodeType *nodeType*) [virtual]

Assign a [PGRGuid](#), device ID and nodeType to the node.

Parameters

<i>guid</i>	PGRGuid to be assigned.
<i>deviceId</i>	Device ID to be assigned.
<i>nodeType</i>	NodeType to be assigned

Returns

Whether the data was successfully set to the node.

8.54.4.5 `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.

8.54.4.6 `virtual int GetDeviceId () [virtual]`

Get the device ID associated with the node.

Returns

Device ID of the node.

8.54.4.7 `virtual PGRGuid GetGuid () [virtual]`

Get the [PGRGuid](#) associated with the node.

Returns

[PGRGuid](#) of the node.

8.54.4.8 `virtual InterfaceType GetInterfaceType () [virtual]`

Get the interface type associated with the node.

Returns

Interface type of the node.

8.54.4.9 virtual NodeType GetNodeType () [virtual]

Get the node type associated with the node.

Returns

Node type of the node.

8.54.4.10 virtual unsigned int GetNumChildren () [virtual]

Get the number of child nodes.

Returns

Number of child nodes.

8.54.4.11 virtual unsigned int GetNumPorts () [virtual]

Get the number of ports.

Returns

Number of ports.

8.54.4.12 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.

8.54.4.13 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](#)

8.55 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.

8.55.1 Detailed Description

A camera trigger.

8.55.2 Constructor & Destructor Documentation

8.55.2.1 [TriggerMode](#) () [inline]

8.55.3 Member Data Documentation

8.55.3.1 unsigned int [mode](#)

Mode value.

8.55.3.2 bool [onOff](#)

Flag controlling on/off.

8.55.3.3 unsigned int parameter

Parameter value.

8.55.3.4 unsigned int polarity

Polarity value.

8.55.3.5 unsigned int reserved[8]

Reserved for future use.

8.55.3.6 unsigned int source

Source value.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

8.56 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.

8.56.1 Detailed Description

Information about a camera trigger property.

8.56.2 Constructor & Destructor Documentation

8.56.2.1 `TriggerModelInfo ()` `[inline]`

8.56.3 Member Data Documentation

8.56.3.1 `unsigned int modeMask`

Mode mask.

8.56.3.2 `bool onOffSupported`

Flag indicating if on/off is supported.

8.56.3.3 `bool polaritySupported`

Flag indicating if polarity is supported.

8.56.3.4 `bool present`

Presence of trigger mode.

8.56.3.5 `bool readOutSupported`

Flag indicating if trigger value can be read out.

8.56.3.6 `unsigned int reserved[8]`

Reserved for future use.

8.56.3.7 `bool softwareTriggerSupported`

Flag indicating if software trigger is supported.

8.56.3.8 `unsigned int sourceMask`

Source mask.

8.56.3.9 bool valueReadable

Flag indicating if the value is readable.

The documentation for this struct was generated from the following file:

- [FlyCapture2Defs.h](#)

8.57 Utilities Class Reference

The Utility class is generally used to query for general system information such as operating system, available memory etc.

Static Public Member Functions

- static [Error CheckDriver](#) (const [PGRGuid](#) *guid)
Check for driver compatibility for the given camera guid.
- static [Error GetDriverDeviceName](#) (const [PGRGuid](#) *guid, std::string &deviceName)
Get the driver's name for a device.
- static [Error GetSystemInfo](#) ([SystemInfo](#) *pSystemInfo)
Get system information.
- static [Error GetLibraryVersion](#) ([FC2Version](#) *pVersion)
Get library version.
- static [Error LaunchBrowser](#) (const char *pAddress)
Launch a URL in the system default browser.
- static [Error LaunchHelp](#) (const char *pFileName)
Open a CHM file in the system default CHM viewer.
- static [Error LaunchCommand](#) (const char *pCommand)
Execute a command in the terminal.
- static [Error LaunchCommandAsync](#) (const char *pCommand, [AsyncCommandCallback](#) pCallback, void *pUserData)
Execute a command in the terminal.

8.57.1 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.

8.57.2 Member Function Documentation

8.57.2.1 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.

8.57.2.2 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.

8.57.2.3 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.

8.57.2.4 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.

8.57.2.5 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.

8.57.2.6 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.

8.57.2.7 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.

8.57.2.8 `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 9

File Documentation

9.1 AVIRecorder.h File Reference

Include dependency graph for AVIRecorder.h:

9.2 BusManager.h File Reference

Include dependency graph for BusManager.h: This graph shows which files directly or indirectly include this file:

Classes

- class [BusManager](#)

The [BusManager](#) class provides the functionality for the user to get an [PGRGuid](#) for a desired camera or device easily.

Namespaces

- [FlyCapture2](#)

Typedefs

- typedef void(* [BusEventCallback](#)) (void *pParameter, unsigned int serialNumber)
Bus event callback function prototype.
- typedef void * [CallbackHandle](#)
Handle that is returned when registering a callback.

9.3 Camera.h File Reference

Include dependency graph for Camera.h: This graph shows which files directly or indirectly include this file:

Classes

- class [Camera](#)

The [Camera](#) object represents a physical camera that uses the IIDC register set.

Namespaces

- [FlyCapture2](#)

9.4 CameraBase.h File Reference

Include dependency graph for CameraBase.h: This graph shows which files directly or indirectly include this file:

Classes

- class [CameraBase](#)

The [CameraBase](#) class is an abstract base class that defines a general interface to a camera.

Namespaces

- [FlyCapture2](#)

Typedefs

- typedef void(* [ImageEventCallback](#)) (class Image *pImage, const void *pCallbackData)
[Image](#) event callback function prototype.

9.5 Error.h File Reference

Include dependency graph for Error.h: This graph shows which files directly or indirectly include this file:

Classes

- class [Error](#)

The [Error](#) object represents an error that is returned from the library.

Namespaces

- [FlyCapture2](#)

9.6 FlyCapture2.h File Reference

Include dependency graph for FlyCapture2.h: This graph shows which files directly or indirectly include this file:

9.7 FlyCapture2Defs.h File Reference

Include dependency graph for FlyCapture2Defs.h: This graph shows which files directly or indirectly include this file:

Classes

- struct [FC2Version](#)
The current version of the library.
- class [PGRGuid](#)
A GUID to the camera.
- struct [IPAddress](#)
IPv4 address.
- struct [MACAddress](#)
MAC address.
- struct [GigEProperty](#)
A GigE property.
- struct [GigEStreamChannel](#)
Information about a single GigE stream channel.
- struct [GigEConfig](#)
Configuration for a GigE camera.
- struct [GigEImageSettingsInfo](#)
Format 7 information for a single mode.
- struct [GigEImageSettings](#)
Image settings for a GigE camera.
- struct [Format7ImageSettings](#)
Format 7 image settings.
- struct [Format7Info](#)
Format 7 information for a single mode.
- struct [Format7PacketInfo](#)
Format 7 packet information.
- struct [FC2Config](#)
Configuration for a camera.
- struct [PropertyInfo](#)
Information about a specific camera property.
- struct [Property](#)
A specific camera property.
- struct [TriggerModelInfo](#)
Information about a camera trigger property.
- struct [TriggerMode](#)
A camera trigger.
- struct [StrobeInfo](#)
A camera strobe property.
- struct [StrobeControl](#)

- A camera strobe.*
- struct [TimeStamp](#)
 - Timestamp information.*
- struct [ConfigROM](#)
 - Camera configuration ROM.*
- struct [CameraInfo](#)
 - Camera information.*
- struct [EmbeddedImageInfoProperty](#)
 - Properties of a single embedded image info property.*
- struct [EmbeddedImageInfo](#)
 - Properties of the possible embedded image information.*
- struct [ImageMetadata](#)
 - Metadata related to an image.*
- struct [LUTData](#)
 - Information about the camera's look up table.*
- struct [CameraStats](#)
 - Camera diagnostic information.*
- struct [PNGOption](#)
 - Options for saving PNG images.*
- struct [PPMOption](#)
 - Options for saving PPM images.*
- struct [PGMOption](#)
 - Options for saving PGM images.*
- struct [TIFFOption](#)
 - Options for saving TIFF images.*
- struct [JPEGOption](#)
 - Options for saving JPEG image.*
- struct [JPG2Option](#)
 - Options for saving JPEG2000 image.*
- struct [BMPOption](#)
 - Options for saving Bitmap image.*
- struct [MJPGOption](#)
 - Options for saving MJPG files.*
- struct [H264Option](#)
 - Options for saving H264 files.*
- struct [AVIOption](#)
 - Options for saving AVI files.*
- struct [EventOptions](#)
 - Options for enabling device event registration.*
- struct [EventCallbackData](#)

Namespaces

- [FlyCapture2](#)

Macros

- #define [NULL](#) 0
- #define [FULL_32BIT_VALUE](#) 0x7FFFFFFF

Typedefs

- typedef PropertyInfo [TriggerDelayInfo](#)
The TriggerDelayInfo structure is identical to [PropertyInfo](#).
- typedef Property [TriggerDelay](#)
The TriggerDelay structure is identical to [Property](#).
- typedef void(* [CameraEventCallback](#)) (void *data)

Enumerations

- enum [ErrorType](#) {
[PGRERROR_UNDEFINED](#) = -1,
[PGRERROR_OK](#),
[PGRERROR_FAILED](#),
[PGRERROR_NOT_IMPLEMENTED](#),
[PGRERROR_FAILED_BUS_MASTER_CONNECTION](#),
[PGRERROR_NOT_CONNECTED](#),
[PGRERROR_INIT_FAILED](#),
[PGRERROR_NOT_INITIALIZED](#),
[PGRERROR_INVALID_PARAMETER](#),
[PGRERROR_INVALID_SETTINGS](#),
[PGRERROR_INVALID_BUS_MANAGER](#),
[PGRERROR_MEMORY_ALLOCATION_FAILED](#),
[PGRERROR_LOW_LEVEL_FAILURE](#),
[PGRERROR_NOT_FOUND](#),
[PGRERROR_FAILED_GUID](#),
[PGRERROR_INVALID_PACKET_SIZE](#),
[PGRERROR_INVALID_MODE](#),
[PGRERROR_NOT_IN_FORMAT7](#),
[PGRERROR_NOT_SUPPORTED](#),
[PGRERROR_TIMEOUT](#),
[PGRERROR_BUS_MASTER_FAILED](#),
[PGRERROR_INVALID_GENERATION](#),
[PGRERROR_LUT_FAILED](#),
[PGRERROR_IIDC_FAILED](#),
[PGRERROR_STROBE_FAILED](#),
[PGRERROR_TRIGGER_FAILED](#),
[PGRERROR_PROPERTY_FAILED](#),
[PGRERROR_PROPERTY_NOT_PRESENT](#),
[PGRERROR_REGISTER_FAILED](#),
[PGRERROR_READ_REGISTER_FAILED](#),
[PGRERROR_WRITE_REGISTER_FAILED](#),
[PGRERROR_ISOCH_FAILED](#),
[PGRERROR_ISOCH_ALREADY_STARTED](#),
[PGRERROR_ISOCH_NOT_STARTED](#),
[PGRERROR_ISOCH_START_FAILED](#),
[PGRERROR_ISOCH_RETRIEVE_BUFFER_FAILED](#),
[PGRERROR_ISOCH_STOP_FAILED](#),
[PGRERROR_ISOCH_SYNC_FAILED](#),
[PGRERROR_ISOCH_BANDWIDTH_EXCEEDED](#),
[PGRERROR_IMAGE_CONVERSION_FAILED](#),
[PGRERROR_IMAGE_LIBRARY_FAILURE](#),
[PGRERROR_BUFFER_TOO_SMALL](#),
[PGRERROR_IMAGE_CONSISTENCY_ERROR](#),
[PGRERROR_INCOMPATIBLE_DRIVER](#),
[PGRERROR_FORCE_32BITS](#) = FULL_32BIT_VALUE }

The error types returned by functions.

- enum `BusCallbackType` {
`BUS_RESET`,
`ARRIVAL`,
`REMOVAL`,
`CALLBACK_TYPE_FORCE_32BITS` = `FULL_32BIT_VALUE` }

The type of bus callback to register a callback function for.

- enum `GrabMode` {
`DROP_FRAMES`,
`BUFFER_FRAMES`,
`UNSPECIFIED_GRAB_MODE`,
`GRAB_MODE_FORCE_32BITS` = `FULL_32BIT_VALUE` }

The grab strategy employed during image transfer.

- enum `GrabTimeout` {
`TIMEOUT_NONE` = 0,
`TIMEOUT_INFINITE` = -1,
`TIMEOUT_UNSPECIFIED` = -2,
`GRAB_TIMEOUT_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Timeout options for grabbing images.

- enum `BandwidthAllocation` {
`BANDWIDTH_ALLOCATION_OFF` = 0,
`BANDWIDTH_ALLOCATION_ON` = 1,
`BANDWIDTH_ALLOCATION_UNSUPPORTED` = 2,
`BANDWIDTH_ALLOCATION_UNSPECIFIED` = 3,
`BANDWIDTH_ALLOCATION_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Bandwidth allocation options for 1394 devices.

- enum `InterfaceType` {
`INTERFACE_IEEE1394`,
`INTERFACE_USB2`,
`INTERFACE_USB3`,
`INTERFACE_GIGE`,
`INTERFACE_UNKNOWN`,
`INTERFACE_TYPE_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Interfaces that a camera may use to communicate with a host.

- enum `PropertyType` {
`BRIGHTNESS`,
`AUTO_EXPOSURE`,
`SHARPNESS`,
`WHITE_BALANCE`,
`HUE`,
`SATURATION`,
`GAMMA`,
`IRIS`,
`FOCUS`,
`ZOOM`,
`PAN`,
`TILT`,
`SHUTTER`,
`GAIN`,
`TRIGGER_MODE`,
`TRIGGER_DELAY`,
`FRAME_RATE`,
`TEMPERATURE`,
`UNSPECIFIED_PROPERTY_TYPE`,
`PROPERTY_TYPE_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Camera properties.

- enum [FrameRate](#) {
 FRAMERATE_1_875,
 FRAMERATE_3_75,
 FRAMERATE_7_5,
 FRAMERATE_15,
 FRAMERATE_30,
 FRAMERATE_60,
 FRAMERATE_120,
 FRAMERATE_240,
 FRAMERATE_FORMAT7,
 NUM_FRAMERATES,
 FRAMERATE_FORCE_32BITS = FULL_32BIT_VALUE }

Frame rates in frames per second.

- enum [VideoMode](#) {
 VIDEOMODE_160x120YUV444,
 VIDEOMODE_320x240YUV422,
 VIDEOMODE_640x480YUV411,
 VIDEOMODE_640x480YUV422,
 VIDEOMODE_640x480RGB,
 VIDEOMODE_640x480Y8,
 VIDEOMODE_640x480Y16,
 VIDEOMODE_800x600YUV422,
 VIDEOMODE_800x600RGB,
 VIDEOMODE_800x600Y8,
 VIDEOMODE_800x600Y16,
 VIDEOMODE_1024x768YUV422,
 VIDEOMODE_1024x768RGB,
 VIDEOMODE_1024x768Y8,
 VIDEOMODE_1024x768Y16,
 VIDEOMODE_1280x960YUV422,
 VIDEOMODE_1280x960RGB,
 VIDEOMODE_1280x960Y8,
 VIDEOMODE_1280x960Y16,
 VIDEOMODE_1600x1200YUV422,
 VIDEOMODE_1600x1200RGB,
 VIDEOMODE_1600x1200Y8,
 VIDEOMODE_1600x1200Y16,
 VIDEOMODE_FORMAT7,
 NUM_VIDEOMODES,
 VIDEOMODE_FORCE_32BITS = FULL_32BIT_VALUE }

DCAM video modes.

- enum [Mode](#) {

```

MODE_0 = 0,
MODE_1,
MODE_2,
MODE_3,
MODE_4,
MODE_5,
MODE_6,
MODE_7,
MODE_8,
MODE_9,
MODE_10,
MODE_11,
MODE_12,
MODE_13,
MODE_14,
MODE_15,
MODE_16,
MODE_17,
MODE_18,
MODE_19,
MODE_20,
MODE_21,
MODE_22,
MODE_23,
MODE_24,
MODE_25,
MODE_26,
MODE_27,
MODE_28,
MODE_29,
MODE_30,
MODE_31,
NUM_MODES,
MODE_FORCE_32BITS = FULL_32BIT_VALUE }

```

Camera modes for DCAM formats as well as Format7.

- enum PixelFormat {


```

PIXEL_FORMAT_MONO8 = 0x80000000,
PIXEL_FORMAT_411YUV8 = 0x40000000,
PIXEL_FORMAT_422YUV8 = 0x20000000,
PIXEL_FORMAT_444YUV8 = 0x10000000,
PIXEL_FORMAT_RGB8 = 0x08000000,
PIXEL_FORMAT_MONO16 = 0x04000000,
PIXEL_FORMAT_RGB16 = 0x02000000,
PIXEL_FORMAT_S_MONO16 = 0x01000000,
PIXEL_FORMAT_S_RGB16 = 0x00800000,
PIXEL_FORMAT_RAW8 = 0x00400000,
PIXEL_FORMAT_RAW16 = 0x00200000,
PIXEL_FORMAT_MONO12 = 0x00100000,
PIXEL_FORMAT_RAW12 = 0x00080000,
PIXEL_FORMAT_BGR = 0x80000008,
PIXEL_FORMAT_BGRU = 0x40000008,
PIXEL_FORMAT_RGB = PIXEL_FORMAT_RGB8,
PIXEL_FORMAT_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_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`,
`COLOR_PROCESSING_ALGORITHM_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Color processing algorithms.

- enum `BayerTileFormat` {
`NONE`,
`RGGB`,
`GRBG`,
`GBRG`,
`BGGR`,
`BT_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Bayer tile formats.

- enum `ImageFileFormat` {
`FROM_FILE_EXT` = -1,
`PGM`,
`PPM`,
`BMP`,
`JPEG`,
`JPEG2000`,
`TIFF`,
`PNG`,
`RAW`,
`IMAGE_FILE_FORMAT_FORCE_32BITS` = `FULL_32BIT_VALUE` }

File formats to be used for saving images to disk.

- enum `GigEPropertyType` {
`HEARTBEAT`,
`HEARTBEAT_TIMEOUT`,
`PACKET_SIZE`,
`PACKET_DELAY` }

Possible properties that can be queried from the camera.

Variables

- static const unsigned int `sk_maxStringLength` = 512
The maximum length that is allocated for a string.
- static const unsigned int `sk_maxNumPorts` = 32
The maximum number of ports one device can have.

9.7.1 Macro Definition Documentation

9.7.1.1 `#define FULL_32BIT_VALUE 0x7FFFFFFF`

9.7.1.2 `#define NULL 0`

9.8 FlyCapture2GUI.h File Reference

Include dependency graph for FlyCapture2GUI.h:

Classes

- class `CameraControlDlg`
The `CameraControlDlg` object represents a dialog that provides a graphical interface to a specified camera.
- class `CameraSelectionDlg`
The `CameraSelectionDlg` object represents a dialog that provides a graphical interface that lists the number of cameras available to the library.

Namespaces

- `FlyCapture2`

9.9 FlyCapture2Platform.h File Reference

This graph shows which files directly or indirectly include this file:

Macros

- `#define FLYCAPTURE2_API __attribute__((visibility("default")))`
- `#define FLYCAPTURE2_LOCAL __attribute__((visibility("hidden")))`

9.9.1 Macro Definition Documentation

9.9.1.1 `#define FLYCAPTURE2_API __attribute__((visibility("default")))`

9.9.1.2 `#define FLYCAPTURE2_LOCAL __attribute__((visibility("hidden")))`

9.10 FlyCapture3ApiGuiWrapper.h File Reference

Include dependency graph for FlyCapture3ApiGuiWrapper.h:

Classes

- class [FlyCapture3ApiGuiWrapper](#)

Namespaces

- [FlyCapture2](#)
- [FlyCap3CameraControl](#)

Macros

- `#define WRAPPER_API __declspec(dllimport)`

9.10.1 Macro Definition Documentation

9.10.1.1 `#define WRAPPER_API __declspec(dllimport)`

9.11 GCCamera.h File Reference

Include dependency graph for GCCamera.h: This graph shows which files directly or indirectly include this file:

Classes

- class [GCCamera](#)

Namespaces

- [FlyCapture2](#)

9.12 GigECamera.h File Reference

Include dependency graph for GigECamera.h: This graph shows which files directly or indirectly include this file:

Classes

- class [GigECamera](#)

The [GigECamera](#) object represents a physical Gigabit Ethernet camera.

Namespaces

- [FlyCapture2](#)

9.13 Image.h File Reference

Include dependency graph for Image.h: This graph shows which files directly or indirectly include this file:

Classes

- class [Image](#)

The [Image](#) class is used to retrieve images from a camera, convert between multiple pixel formats and save images to disk.

Namespaces

- [FlyCapture2](#)

9.14 ImageStatistics.h File Reference

Include dependency graph for ImageStatistics.h: This graph shows which files directly or indirectly include this file:

Classes

- class [ImageStatistics](#)

The [ImageStatistics](#) object represents image statistics for an image.

Namespaces

- [FlyCapture2](#)

9.15 Internal.h File Reference

Classes

- class [Internal](#)

Namespaces

- [FlyCapture2](#)

9.16 MultiSyncLibrary.h File Reference

Include dependency graph for MultiSyncLibrary.h:

9.17 MultiSyncLibraryDefs.h File Reference

Include dependency graph for MultiSyncLibraryDefs.h: This graph shows which files directly or indirectly include this file:

Classes

- class [SyncManager](#)

Namespaces

- [MultiSyncLibrary](#)

Enumerations

- enum [PGRSyncError](#) {
 [PGRSyncError_OK](#) = 0,
 [PGRSyncError_FAILED](#),
 [PGRSyncError_ALREADY_STARTED](#),
 [PGRSyncError_ALREADY_STOPPED](#),
 [PGRSyncError_CAMERA_NOT_FOUND](#),
 [PGRSyncError_UNKNOWN_ERROR](#) }
- enum [PGRSyncMessage](#) {
 [PGRSyncMessage_OK](#) = 0,
 [PGRSyncMessage_STARTED](#),
 [PGRSyncMessage_STOPPED](#),
 [PGRSyncMessage_SYNCING](#),
 [PGRSyncMessage_NOMASTER](#),
 [PGRSyncMessage_THREAD_ERROR](#),
 [PGRSyncMessage_DEVICE_ERROR](#),
 [PGRSyncMessage_NOT_ENOUGH_DEVICES](#),
 [PGRSyncMessage_BUS_RESET](#),
 [PGRSyncMessage_NOT_INITIALIZED](#),
 [PGRSyncMessage_UNKNOWN_ERROR](#) }

9.18 MultiSyncLibraryPlatform.h File Reference

This graph shows which files directly or indirectly include this file:

Macros

- `#define MULTISYNCLIBRARY_API __attribute__((visibility ("default")))`
- `#define MULTISYNCLIBRARY_LOCAL __attribute__((visibility ("hidden")))`

9.18.1 Macro Definition Documentation

9.18.1.1 `#define MULTISYNCLIBRARY_API __attribute__((visibility ("default")))`

9.18.1.2 `#define MULTISYNCLIBRARY_LOCAL __attribute__((visibility ("hidden")))`

9.19 NodeMap.h File Reference

Include dependency graph for NodeMap.h:

Classes

- class [NodeMap](#)

Namespaces

- [FlyCapture2](#)

9.20 TopologyNode.h File Reference

Include dependency graph for TopologyNode.h: This graph shows which files directly or indirectly include this file:

Classes

- class [TopologyNode](#)
The [TopologyNode](#) class contains topology information that can be used to generate a tree structure of all cameras and devices connected to a computer.

Namespaces

- [FlyCapture2](#)

9.21 Utilities.h File Reference

Include dependency graph for Utilities.h: This graph shows which files directly or indirectly include this file:

Classes

- struct [SystemInfo](#)
Description of the system.
- class [Utilities](#)
The Utility class is generally used to query for general system information such as operating system, available memory etc.

Namespaces

- [FlyCapture2](#)

Typedefs

- typedef void(* [AsyncCommandCallback](#)) (class Error retError, void *pUserData)
Async command callback function prototype.

Enumerations

- enum [OSType](#) {
 [WINDOWS_X86](#),
 [WINDOWS_X64](#),
 [LINUX_X86](#),
 [LINUX_X64](#),
 [MAC](#),
 [UNKNOWN_OS](#),
 [OSTYPE_FORCE_32BITS](#) = [FULL_32BIT_VALUE](#) }
Possible operating systems.
- enum [ByteOrder](#) {
 [BYTE_ORDER_LITTLE_ENDIAN](#),
 [BYTE_ORDER_BIG_ENDIAN](#),
 [BYTE_ORDER_FORCE_32BITS](#) = [FULL_32BIT_VALUE](#) }
Possible byte orders.

Index

- _GetDeviceName
 - FlyCapture2::NodeMap, [238](#)
 - _GetNode
 - FlyCapture2::NodeMap, [238](#)
 - _GetNodes
 - FlyCapture2::NodeMap, [238](#)
 - _InvalidateNodes
 - FlyCapture2::NodeMap, [238](#)
 - _Poll
 - FlyCapture2::NodeMap, [238](#)
- ~AVIRecorder
 - FlyCapture2::AVIRecorder, [48](#)
- ~BusManager
 - FlyCapture2::BusManager, [53](#)
- ~Camera
 - FlyCapture2::Camera, [64](#)
- ~CameraBase
 - FlyCapture2::CameraBase, [91](#)
- ~CameraControlDlg
 - FlyCapture2::CameraControlDlg, [113](#)
- ~CameraSelectionDlg
 - FlyCapture2::CameraSelectionDlg, [120](#)
- ~Error
 - FlyCapture2::Error, [129](#)
- ~FlyCapture3ApiGuiWrapper
 - FlyCap3CameraControl::FlyCapture3ApiGui←Wrapper, [138](#)
- ~GCCamera
 - FlyCapture2::GCCamera, [147](#)
- ~GigECamera
 - FlyCapture2::GigECamera, [171](#)
- ~Image
 - FlyCapture2::Image, [211](#)
- ~ImageStatistics
 - FlyCapture2::ImageStatistics, [226](#)
- ~NodeMap
 - FlyCapture2::NodeMap, [238](#)
- ~SyncManager
 - MultiSyncLibrary::SyncManager, [251](#)
- ~TopologyNode
 - FlyCapture2::TopologyNode, [258](#)
- ADOBE_DEFLATE
 - FlyCapture2::TIFFOption, [254](#)
- ARRIVAL
 - Enumerations, [20](#)
- AUTO_EXPOSURE
 - Enumerations, [26](#)
- AVIAppend
 - FlyCapture2::AVIRecorder, [48](#)
- AVIClose
 - FlyCapture2::AVIRecorder, [49](#)
- AVIOpen
 - FlyCapture2::AVIRecorder, [49](#), [50](#)
- AVIOption, [47](#)
 - FlyCapture2::AVIOption, [47](#)
- AVIRecorder, [48](#)
 - FlyCapture2::AVIRecorder, [48](#)
- AVIRecorder.h, [269](#)
- absControl
 - FlyCapture2::Property, [243](#)
- absMax
 - FlyCapture2::PropertyInfo, [246](#)
- absMin
 - FlyCapture2::PropertyInfo, [246](#)
- absValSupported
 - FlyCapture2::PropertyInfo, [246](#)
- absValue
 - FlyCapture2::Property, [243](#)
- AddChild
 - FlyCapture2::TopologyNode, [259](#)
- AddPortType
 - FlyCapture2::TopologyNode, [259](#)
- applicationIPAddress
 - FlyCapture2::CameraInfo, [116](#)
- applicationPort
 - FlyCapture2::CameraInfo, [116](#)
- AssignGuidToNode
 - FlyCapture2::TopologyNode, [259](#)
- asyncBusSpeed
 - FlyCapture2::FC2Config, [134](#)
- AsyncCommandCallback
 - FlyCapture2, [43](#)
- autoManualMode
 - FlyCapture2::Property, [244](#)
- autoSupported
 - FlyCapture2::PropertyInfo, [246](#)
- available
 - FlyCapture2::EmbeddedImageInfoProperty, [127](#)
- BANDWIDTH_ALLOCATION_FORCE_32BITS
 - Enumerations, [19](#)
- BANDWIDTH_ALLOCATION_OFF
 - Enumerations, [19](#)
- BANDWIDTH_ALLOCATION_ON
 - Enumerations, [19](#)
- BANDWIDTH_ALLOCATION_UNSPECIFIED
 - Enumerations, [19](#)
- BANDWIDTH_ALLOCATION_UNSUPPORTED
 - Enumerations, [19](#)

- BGGR
 - Enumerations, [19](#)
- BLUE
 - FlyCapture2::ImageStatistics, [225](#)
- BMPOption, [50](#)
 - FlyCapture2::BMPOption, [51](#)
- BMP
 - Enumerations, [23](#)
- BRIGHTNESS
 - Enumerations, [26](#)
- BT_FORCE_32BITS
 - Enumerations, [19](#)
- BUFFER_FRAMES
 - Enumerations, [23](#)
- BUS_RESET
 - Enumerations, [20](#)
- BUSSPEED_1000BASE_T
 - Enumerations, [20](#)
- BUSSPEED_1000BASE_T
 - Enumerations, [20](#)
- BUSSPEED_100BASE_T
 - Enumerations, [20](#)
- BUSSPEED_10BASE_T
 - Enumerations, [20](#)
- BUSSPEED_ANY
 - Enumerations, [20](#)
- BUSSPEED_FORCE_32BITS
 - Enumerations, [20](#)
- BUSSPEED_S100
 - Enumerations, [20](#)
- BUSSPEED_S1600
 - Enumerations, [20](#)
- BUSSPEED_S200
 - Enumerations, [20](#)
- BUSSPEED_S3200
 - Enumerations, [20](#)
- BUSSPEED_S400
 - Enumerations, [20](#)
- BUSSPEED_S480
 - Enumerations, [20](#)
- BUSSPEED_S5000
 - Enumerations, [20](#)
- BUSSPEED_S800
 - Enumerations, [20](#)
- BUSSPEED_S_FASTEST
 - Enumerations, [20](#)
- BUSSPEED_SPEED_UNKNOWN
 - Enumerations, [20](#)
- BUS
 - FlyCapture2::TopologyNode, [258](#)
- BYTE_ORDER_BIG_ENDIAN
 - FlyCapture2, [44](#)
- BYTE_ORDER_FORCE_32BITS
 - FlyCapture2, [44](#)
- BYTE_ORDER_LITTLE_ENDIAN
 - FlyCapture2, [44](#)
- BandwidthAllocation
 - Enumerations, [19](#)
- bandwidthAllocation
 - FlyCapture2::FC2Config, [134](#)
- BayerTileFormat
 - Enumerations, [19](#)
- bayerTileFormat
 - FlyCapture2::CameraInfo, [117](#)
- binaryFile
 - FlyCapture2::PGMOption, [239](#)
 - FlyCapture2::PPMOption, [242](#)
- bitrate
 - FlyCapture2::H264Option, [207](#)
- brightness
 - FlyCapture2::EmbeddedImageInfo, [126](#)
- build
 - FlyCapture2::FC2Version, [137](#)
- BusCallbackType
 - Enumerations, [19](#)
- BusEventCallback
 - FlyCapture2, [43](#)
- BusManager, [51](#)
 - FlyCapture2::BusManager, [53](#)
- BusManager.h, [269](#)
- busNumber
 - FlyCapture2::CameraInfo, [117](#)
- BusSpeed
 - Enumerations, [20](#)
- ByteOrder
 - FlyCapture2, [44](#)
- byteOrder
 - FlyCapture2::SystemInfo, [252](#)
- CALLBACK_TYPE_FORCE_32BITS
 - Enumerations, [20](#)
- CAMERA
 - FlyCapture2::TopologyNode, [258](#)
- CCITTFAX3
 - FlyCapture2::TIFFOption, [254](#)
- CCITTFAX4
 - FlyCapture2::TIFFOption, [254](#)
- COLOR_PROCESSING_ALGORITHM_FORCE_32BITS
 - Enumerations, [20](#)
- COMPUTER
 - FlyCapture2::TopologyNode, [258](#)
- CONNECTED_TO_CHILD
 - FlyCapture2::TopologyNode, [258](#)
- CONNECTED_TO_PARENT
 - FlyCapture2::TopologyNode, [258](#)
- CalculateStatistics
 - FlyCapture2::Image, [212](#)
- CallbackHandle
 - FlyCapture2, [44](#)
- Camera, [60](#)
 - FlyCapture2::Camera, [64](#)
- Camera.h, [269](#)
- CameraBase, [88](#)
 - FlyCapture2::CameraBase, [91](#)
- CameraBase.h, [270](#)
- CameraControlDlg, [112](#)

- FlyCapture2::CameraControlDlg, 113
- cameraCurrents
 - FlyCapture2::CameraStats, 122
- CameraEventCallback
 - Image saving structures., 33
- CameraInfo, 115
 - FlyCapture2::CameraInfo, 116
- cameraPowerUp
 - FlyCapture2::CameraStats, 122
- CameraSelectionDlg, 120
 - FlyCapture2::CameraSelectionDlg, 120
- CameraStats, 121
 - FlyCapture2::CameraStats, 122
- cameraVoltages
 - FlyCapture2::CameraStats, 122
- ccpStatus
 - FlyCapture2::CameraInfo, 117
- CheckDriver
 - FlyCapture2::Utilities, 265
- chipIdHi
 - FlyCapture2::ConfigROM, 124
- chipIdLo
 - FlyCapture2::ConfigROM, 124
- CollectSupportInformation
 - FlyCapture2::Error, 129
- ColorProcessingAlgorithm
 - Enumerations, 20
- compression
 - FlyCapture2::TIFFOption, 255
- compressionLevel
 - FlyCapture2::PNGOption, 241
- CompressionMethod
 - FlyCapture2::TIFFOption, 254
- ConfigROM, 123
 - FlyCapture2::ConfigROM, 124
- configROM
 - FlyCapture2::CameraInfo, 117
- Connect
 - FlyCapture2::Camera, 64
 - FlyCapture2::CameraBase, 91
 - FlyCapture2::CameraControlDlg, 113
 - FlyCapture2::GCCamera, 147
 - FlyCapture2::GigECamera, 171
- ConnectGUILibrary
 - FlyCap3CameraControl::FlyCapture3ApiGui←
Wrapper, 138
- Convert
 - FlyCapture2::Image, 212
- cpuDescription
 - FlyCapture2::SystemInfo, 252
- cycleCount
 - FlyCapture2::TimeStamp, 256
- cycleOffset
 - FlyCapture2::TimeStamp, 256
- cycleSeconds
 - FlyCapture2::TimeStamp, 256
- DEFAULT
 - Enumerations, 20
- DEFLATE
 - FlyCapture2::TIFFOption, 254
- DIRECTIONAL_FILTER
 - Enumerations, 20
- DRIVER_1394_CAM
 - Enumerations, 21
- DRIVER_1394_JUJU
 - Enumerations, 21
- DRIVER_1394_PRO
 - Enumerations, 21
- DRIVER_1394_RAW1394
 - Enumerations, 21
- DRIVER_1394_VIDEO1394
 - Enumerations, 21
- 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, 21
- DRIVER_USB_NONE
 - Enumerations, 21
- DROP_FRAMES
 - Enumerations, 23
- DeepCopy
 - FlyCapture2::Image, 212
- defaultGateway
 - FlyCapture2::CameraInfo, 117
- delay
 - FlyCapture2::StrobeControl, 248
- DeregisterAllEvents
 - FlyCapture2::Camera, 64
 - FlyCapture2::CameraBase, 92
 - FlyCapture2::GigECamera, 171
- DeregisterEvent
 - FlyCapture2::Camera, 64
 - FlyCapture2::CameraBase, 92
 - FlyCapture2::GigECamera, 171
- destinationIpAddress
 - FlyCapture2::GigEStreamChannel, 205
- DetermineBitsPerPixel
 - FlyCapture2::Image, 213
- DisableAll
 - FlyCapture2::ImageStatistics, 226
- DisableCrossPCSSynchronization
 - MultiSyncLibrary::SyncManager, 251
- Disconnect
 - FlyCapture2::Camera, 64

- FlyCapture2::CameraBase, 92
- FlyCapture2::CameraControlDlg, 114
- FlyCapture2::GCCamera, 147
- FlyCapture2::GigECamera, 171
- DisconnectGUILibrary
 - FlyCap3CameraControl::FlyCapture3ApiGui↔
Wrapper, 138
- DiscoverGigECameras
 - FlyCapture2::BusManager, 53
- DiscoverGigEPacketSize
 - FlyCapture2::GigECamera, 171
- doNotFragment
 - FlyCapture2::GigEStreamChannel, 205
- driverList
 - FlyCapture2::SystemInfo, 252
- driverName
 - FlyCapture2::CameraInfo, 117
- DriverType
 - Enumerations, 20
- driverType
 - FlyCapture2::CameraInfo, 117
- duration
 - FlyCapture2::StrobeControl, 248
- EDGE_SENSING
 - Enumerations, 20
- embeddedBrightness
 - FlyCapture2::ImageMetadata, 223
- embeddedExposure
 - FlyCapture2::ImageMetadata, 223
- embeddedFrameCounter
 - FlyCapture2::ImageMetadata, 223
- embeddedGPIOPinState
 - FlyCapture2::ImageMetadata, 223
- embeddedGain
 - FlyCapture2::ImageMetadata, 223
- EmbeddedImageInfo, 125
- EmbeddedImageInfoProperty, 126
 - FlyCapture2::EmbeddedImageInfoProperty, 127
- embeddedROIPosition
 - FlyCapture2::ImageMetadata, 223
- embeddedShutter
 - FlyCapture2::ImageMetadata, 223
- embeddedStrobePattern
 - FlyCapture2::ImageMetadata, 223
- embeddedTimeStamp
 - FlyCapture2::ImageMetadata, 223
- embeddedWhiteBalance
 - FlyCapture2::ImageMetadata, 224
- EnableAll
 - FlyCapture2::ImageStatistics, 226
- EnableCrossPCSSynchronization
 - MultiSyncLibrary::SyncManager, 251
- EnableGreyOnly
 - FlyCapture2::ImageStatistics, 226
- EnableHSLOnly
 - FlyCapture2::ImageStatistics, 226
- EnableLUT
 - FlyCapture2::Camera, 64
- FlyCapture2::CameraBase, 92
- FlyCapture2::GCCamera, 147
- FlyCapture2::GigECamera, 172
- enablePacketResend
 - FlyCapture2::GigEConfig, 199
- EnableRGBOnly
 - FlyCapture2::ImageStatistics, 227
- enabled
 - FlyCapture2::LUTData, 234
- Enumerations, 14
 - ARRIVAL, 20
 - AUTO_EXPOSURE, 26
 - BANDWIDTH_ALLOCATION_FORCE_32BITS, 19
 - BANDWIDTH_ALLOCATION_OFF, 19
 - BANDWIDTH_ALLOCATION_ON, 19
 - BANDWIDTH_ALLOCATION_UNSPECIFIED, 19
 - BANDWIDTH_ALLOCATION_UNSUPPORTED, 19
 - BGGR, 19
 - BMP, 23
 - BRIGHTNESS, 26
 - BT_FORCE_32BITS, 19
 - BUFFER_FRAMES, 23
 - BUS_RESET, 20
 - BUSSPEED_10000BASE_T, 20
 - BUSSPEED_1000BASE_T, 20
 - BUSSPEED_100BASE_T, 20
 - BUSSPEED_10BASE_T, 20
 - BUSSPEED_ANY, 20
 - BUSSPEED_FORCE_32BITS, 20
 - BUSSPEED_S100, 20
 - BUSSPEED_S1600, 20
 - BUSSPEED_S200, 20
 - BUSSPEED_S3200, 20
 - BUSSPEED_S400, 20
 - BUSSPEED_S480, 20
 - BUSSPEED_S5000, 20
 - BUSSPEED_S800, 20
 - BUSSPEED_S_FASTEST, 20
 - BUSSPEED_SPEED_UNKNOWN, 20
 - BandwidthAllocation, 19
 - BayerTileFormat, 19
 - BusCallbackType, 19
 - BusSpeed, 20
 - CALLBACK_TYPE_FORCE_32BITS, 20
 - COLOR_PROCESSING_ALGORITHM_FORCE↔
_32BITS, 20
 - ColorProcessingAlgorithm, 20
 - DEFAULT, 20
 - DIRECTIONAL_FILTER, 20
 - DRIVER_1394_CAM, 21
 - DRIVER_1394_JUJU, 21
 - DRIVER_1394_PRO, 21
 - DRIVER_1394_RAW1394, 21
 - DRIVER_1394_VIDEO1394, 21
 - 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, 21
- DRIVER_USB_NONE, 21
- DROP_FRAMES, 23
- DriverType, 20
- EDGE_SENSING, 20
- ErrorType, 21
- FOCUS, 26
- FRAME_RATE, 26
- FRAMERATE_120, 22
- FRAMERATE_15, 22
- FRAMERATE_1_875, 22
- FRAMERATE_240, 22
- FRAMERATE_30, 22
- FRAMERATE_3_75, 22
- FRAMERATE_60, 22
- FRAMERATE_7_5, 22
- FRAMERATE_FORCE_32BITS, 22
- FRAMERATE_FORMAT7, 22
- FROM_FILE_EXT, 23
- FrameRate, 22
- GAIN, 26
- GAMMA, 26
- GBRG, 19
- GRAB_MODE_FORCE_32BITS, 23
- GRAB_TIMEOUT_FORCE_32BITS, 23
- GRBG, 19
- GrabMode, 22
- GrabTimeout, 23
- HQ_LINEAR, 20
- HUE, 26
- IMAGE_FILE_FORMAT_FORCE_32BITS, 23
- INTERFACE_GIGE, 24
- INTERFACE_IEEE1394, 24
- INTERFACE_TYPE_FORCE_32BITS, 24
- INTERFACE_UNKNOWN, 24
- INTERFACE_USB2, 24
- INTERFACE_USB3, 24
- IPP, 20
- IRIS, 26
- ImageFileFormat, 23
- InterfaceType, 23
- JPEG2000, 23
- JPEG, 23
- MODE_0, 24
- MODE_1, 24
- MODE_10, 24
- MODE_11, 24
- MODE_12, 24
- MODE_13, 24
- MODE_14, 24
- MODE_15, 24
- MODE_16, 24
- MODE_17, 24
- MODE_18, 24
- MODE_19, 24
- MODE_2, 24
- MODE_20, 24
- MODE_21, 24
- MODE_22, 24
- MODE_23, 24
- MODE_24, 24
- MODE_25, 24
- MODE_26, 24
- MODE_27, 24
- MODE_28, 24
- MODE_29, 24
- MODE_3, 24
- MODE_30, 24
- MODE_31, 24
- MODE_4, 24
- MODE_5, 24
- MODE_6, 24
- MODE_7, 24
- MODE_8, 24
- MODE_9, 24
- MODE_FORCE_32BITS, 24
- Mode, 24
- NEAREST_NEIGHBOR, 20
- NO_COLOR_PROCESSING, 20
- NONE, 19
- NUM_FRAMERATES, 22
- NUM_MODES, 24
- NUM_PIXEL_FORMATS, 25
- NUM_VIDEOMODES, 27
- PAN, 26
- PCIE_BUSSPEED_2_5, 25
- PCIE_BUSSPEED_5_0, 25
- PCIE_BUSSPEED_FORCE_32BITS, 25
- PCIE_BUSSPEED_UNKNOWN, 25
- PCleBusSpeed, 24
- PGRERROR_BUFFER_TOO_SMALL, 22
- PGRERROR_BUS_MASTER_FAILED, 22
- PGRERROR_FAILED_BUS_MASTER_CONNECTION, 21
- PGRERROR_FAILED_GUID, 21
- PGRERROR_FAILED, 21
- PGRERROR_FORCE_32BITS, 22
- PGRERROR_IIDC_FAILED, 22
- 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, 22
- 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, [22](#)
- PGRERROR_MEMORY_ALLOCATION_FAILED, [21](#)
- PGRERROR_NOT_CONNECTED, [21](#)
- PGRERROR_NOT_FOUND, [21](#)
- PGRERROR_NOT_IMPLEMENTED, [21](#)
- PGRERROR_NOT_IN_FORMAT7, [21](#)
- PGRERROR_NOT_INITIALIZED, [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](#)
- PGM, [23](#)
- PIXEL_FORMAT_411YUV8, [25](#)
- PIXEL_FORMAT_422YUV8, [25](#)
- PIXEL_FORMAT_422YUV8_JPEG, [25](#)
- PIXEL_FORMAT_444YUV8, [25](#)
- PIXEL_FORMAT_BGR16, [25](#)
- PIXEL_FORMAT_BGRU16, [25](#)
- PIXEL_FORMAT_BGRU, [25](#)
- PIXEL_FORMAT_BGR, [25](#)
- PIXEL_FORMAT_MONO12, [25](#)
- PIXEL_FORMAT_MONO16, [25](#)
- PIXEL_FORMAT_MONO8, [25](#)
- PIXEL_FORMAT_RAW12, [25](#)
- PIXEL_FORMAT_RAW16, [25](#)
- PIXEL_FORMAT_RAW8, [25](#)
- PIXEL_FORMAT_RGB16, [25](#)
- PIXEL_FORMAT_RGB8, [25](#)
- PIXEL_FORMAT_RGBU, [25](#)
- PIXEL_FORMAT_RGB, [25](#)
- PIXEL_FORMAT_S_MONO16, [25](#)
- PIXEL_FORMAT_S_RGB16, [25](#)
- PNG, [23](#)
- PPM, [23](#)
- PROPERTY_TYPE_FORCE_32BITS, [26](#)
- PixelFormat, [25](#)
- PropertyType, [25](#)
- RAW, [23](#)
- REMOVAL, [20](#)
- RGB, [19](#)
- RIGOROUS, [20](#)
- SATURATION, [26](#)
- SHARPNESS, [26](#)
- SHUTTER, [26](#)
- TEMPERATURE, [26](#)
- TIFF, [23](#)
- TILT, [26](#)
- TIMEOUT_INFINITE, [23](#)
- TIMEOUT_NONE, [23](#)
- TIMEOUT_UNSPECIFIED, [23](#)
- TRIGGER_DELAY, [26](#)
- TRIGGER_MODE, [26](#)
- UNSPECIFIED_GRAB_MODE, [23](#)
- UNSPECIFIED_PIXEL_FORMAT, [25](#)
- UNSPECIFIED_PROPERTY_TYPE, [26](#)
- VIDEOMODE_1024x768RGB, [26](#)
- VIDEOMODE_1024x768Y16, [27](#)
- VIDEOMODE_1024x768Y8, [27](#)
- VIDEOMODE_1024x768YUV422, [26](#)
- VIDEOMODE_1280x960RGB, [27](#)
- VIDEOMODE_1280x960Y16, [27](#)
- VIDEOMODE_1280x960Y8, [27](#)
- VIDEOMODE_1280x960YUV422, [27](#)
- VIDEOMODE_1600x1200RGB, [27](#)
- VIDEOMODE_1600x1200Y16, [27](#)
- VIDEOMODE_1600x1200Y8, [27](#)
- VIDEOMODE_1600x1200YUV422, [27](#)
- VIDEOMODE_160x120YUV444, [26](#)
- VIDEOMODE_320x240YUV422, [26](#)
- VIDEOMODE_640x480RGB, [26](#)
- VIDEOMODE_640x480Y16, [26](#)
- VIDEOMODE_640x480Y8, [26](#)
- VIDEOMODE_640x480YUV411, [26](#)
- VIDEOMODE_640x480YUV422, [26](#)
- VIDEOMODE_800x600RGB, [26](#)
- VIDEOMODE_800x600Y16, [26](#)
- VIDEOMODE_800x600Y8, [26](#)
- VIDEOMODE_800x600YUV422, [26](#)
- VIDEOMODE_FORCE_32BITS, [27](#)
- VIDEOMODE_FORMAT7, [27](#)
- VideoMode, [26](#)
- WHITE_BALANCE, [26](#)
- ZOOM, [26](#)
- Error, [127](#)
 - FlyCapture2::Error, [128](#)
- Error.h, [270](#)
- ErrorType
 - Enumerations, [21](#)
- EventCallbackData, [131](#)
- EventCallbackFcn
 - FlyCapture2::EventOptions, [133](#)
- EventData
 - FlyCapture2::EventCallbackData, [131](#)
- EventDataSize
 - FlyCapture2::EventCallbackData, [131](#)
- EventID
 - FlyCapture2::EventCallbackData, [132](#)
- EventName

- FlyCapture2::EventCallbackData, [132](#)
 - FlyCapture2::EventOptions, [133](#)
- EventOptions, [132](#)
- EventTimestamp
 - FlyCapture2::EventCallbackData, [132](#)
- EventUserData
 - FlyCapture2::EventCallbackData, [132](#)
 - FlyCapture2::EventOptions, [133](#)
- EventUserDataSize
 - FlyCapture2::EventCallbackData, [132](#)
 - FlyCapture2::EventOptions, [133](#)
- exposure
 - FlyCapture2::EmbeddedImageInfo, [126](#)
- FC2Config, [133](#)
 - FlyCapture2::FC2Config, [134](#)
- FC2Version, [136](#)
- FLYCAPTURE2_API
 - FlyCapture2Platform.h, [279](#)
- FLYCAPTURE2_LOCAL
 - FlyCapture2Platform.h, [279](#)
- FOCUS
 - Enumerations, [26](#)
- FRAME_RATE
 - Enumerations, [26](#)
- FRAMERATE_120
 - Enumerations, [22](#)
- FRAMERATE_15
 - Enumerations, [22](#)
- FRAMERATE_1_875
 - Enumerations, [22](#)
- FRAMERATE_240
 - Enumerations, [22](#)
- FRAMERATE_30
 - Enumerations, [22](#)
- FRAMERATE_3_75
 - Enumerations, [22](#)
- FRAMERATE_60
 - Enumerations, [22](#)
- FRAMERATE_7_5
 - Enumerations, [22](#)
- FRAMERATE_FORCE_32BITS
 - Enumerations, [22](#)
- FRAMERATE_FORMAT7
 - Enumerations, [22](#)
- FROM_FILE_EXT
 - Enumerations, [23](#)
- FULL_32BIT_VALUE
 - FlyCapture2Defs.h, [278](#)
- FireBusReset
 - FlyCapture2::BusManager, [53](#)
- FireSoftwareTrigger
 - FlyCapture2::Camera, [65](#)
 - FlyCapture2::CameraBase, [93](#)
 - FlyCapture2::GCCamera, [148](#)
 - FlyCapture2::GigECamera, [172](#)
- firmwareBuildTime
 - FlyCapture2::CameraInfo, [117](#)
- firmwareVersion
 - FlyCapture2::CameraInfo, [117](#)
- FlyCap3CameraControl, [35](#)
- FlyCap3CameraControl::FlyCapture3ApiGuiWrapper
 - ~FlyCapture3ApiGuiWrapper, [138](#)
 - ConnectGUILibrary, [138](#)
 - DisconnectGUILibrary, [138](#)
 - FlyCapture3ApiGuiWrapper, [138](#)
 - GetControlNameList, [138](#)
 - GetDialogNameList, [138](#)
 - GetNumDialogs, [138](#)
 - GetNumOfControls, [138](#)
 - ShowCameraSelectionDialog, [138](#)
 - ShowDialogByIndex, [138](#)
 - ShowDialogByName, [138](#)
 - ShowPropertyGridDialog, [138](#)
- FlyCapture2, [35](#)
 - AsyncCommandCallback, [43](#)
 - BYTE_ORDER_BIG_ENDIAN, [44](#)
 - BYTE_ORDER_FORCE_32BITS, [44](#)
 - BYTE_ORDER_LITTLE_ENDIAN, [44](#)
 - BusEventCallback, [43](#)
 - ByteOrder, [44](#)
 - CallbackHandle, [44](#)
 - ImageEventCallback, [44](#)
 - LINUX_X64, [44](#)
 - LINUX_X86, [44](#)
 - MAC, [44](#)
 - OSTYPE_FORCE_32BITS, [44](#)
 - OSType, [44](#)
 - UNKNOWN_OS, [44](#)
 - WINDOWS_X64, [44](#)
 - WINDOWS_X86, [44](#)
- FlyCapture2.h, [271](#)
- FlyCapture2::AVIOption
 - AVIOption, [47](#)
 - frameRate, [47](#)
 - reserved, [47](#)
- FlyCapture2::AVIRecorder
 - ~AVIRecorder, [48](#)
 - AVIAppend, [48](#)
 - AVIClose, [49](#)
 - AVIOpen, [49](#), [50](#)
 - AVIRecorder, [48](#)
- FlyCapture2::BMPOption
 - BMPOption, [51](#)
 - indexedColor_8bit, [51](#)
 - reserved, [51](#)
- FlyCapture2::BusManager
 - ~BusManager, [53](#)
 - BusManager, [53](#)
 - DiscoverGigECameras, [53](#)
 - FireBusReset, [53](#)
 - ForceAllIPAddressesAutomatically, [53](#), [54](#)
 - ForceIPAddressToCamera, [54](#)
 - GetCameraFromIPAddress, [55](#)
 - GetCameraFromIndex, [54](#)
 - GetCameraFromSerialNumber, [55](#)
 - GetCameraSerialNumberFromIndex, [55](#)

- GetDeviceFromIndex, 56
- GetInterfaceTypeFromGuid, 56
- GetNumOfCameras, 57
- GetNumOfDevices, 57
- GetTopology, 57
- GetUsbLinkInfo, 57
- GetUsbPortStatus, 58
- IsCameraControlable, 58
- ReadPhyRegister, 58
- RegisterCallback, 59
- RescanBus, 59
- UnregisterCallback, 59
- WritePhyRegister, 60
- FlyCapture2::Camera
 - ~Camera, 64
 - Camera, 64
 - Connect, 64
 - DeregisterAllEvents, 64
 - DeregisterEvent, 64
 - Disconnect, 64
 - EnableLUT, 64
 - FireSoftwareTrigger, 65
 - GetActiveLUTBank, 65
 - GetCameraInfo, 65
 - GetConfiguration, 66
 - GetCycleTime, 66
 - GetEmbeddedImageInfo, 66
 - GetFormat7Configuration, 67
 - GetFormat7Info, 67
 - GetGPIOPinDirection, 68
 - GetLUTBankInfo, 68
 - GetLUTChannel, 69
 - GetLUTInfo, 69
 - GetMemoryChannel, 70
 - GetMemoryChannelInfo, 70
 - GetProperty, 70
 - GetPropertyInfo, 71
 - GetRegisterString, 71
 - GetStats, 71
 - GetStrobe, 71
 - GetStrobeInfo, 72
 - GetTriggerDelay, 72
 - GetTriggerDelayInfo, 73
 - GetTriggerMode, 73
 - GetTriggerModelInfo, 74
 - GetVideoModeAndFrameRate, 74
 - GetVideoModeAndFrameRateInfo, 75
 - IsConnected, 75
 - ReadRegister, 75
 - ReadRegisterBlock, 76
 - RegisterAllEvents, 76
 - RegisterEvent, 76
 - ResetStats, 76
 - RestoreFromMemoryChannel, 76
 - RetrieveBuffer, 77
 - SaveToMemoryChannel, 77
 - SetActiveLUTBank, 78
 - SetCallback, 78
 - SetConfiguration, 78
 - SetEmbeddedImageInfo, 80
 - SetFormat7Configuration, 80, 81
 - SetGPIOPinDirection, 81
 - SetLUTChannel, 81
 - SetProperty, 82
 - SetStrobe, 82
 - SetTriggerDelay, 83
 - SetTriggerMode, 83
 - SetUserBuffers, 84
 - SetVideoModeAndFrameRate, 84
 - StartCapture, 85
 - StartSyncCapture, 85
 - StopCapture, 85
 - ValidateFormat7Settings, 86
 - WaitForBufferEvent, 86
 - WriteRegister, 87
 - WriteRegisterBlock, 87
- FlyCapture2::CameraBase
 - ~CameraBase, 91
 - CameraBase, 91
 - Connect, 91
 - DeregisterAllEvents, 92
 - DeregisterEvent, 92
 - Disconnect, 92
 - EnableLUT, 92
 - FireSoftwareTrigger, 93
 - GetActiveLUTBank, 93
 - GetCameraInfo, 93
 - GetConfiguration, 94
 - GetCycleTime, 94
 - GetEmbeddedImageInfo, 94
 - GetGPIOPinDirection, 95
 - GetLUTBankInfo, 95
 - GetLUTChannel, 96
 - GetLUTInfo, 96
 - GetMemoryChannel, 97
 - GetMemoryChannelInfo, 97
 - GetProperty, 97
 - GetPropertyInfo, 98
 - GetRegisterString, 98
 - GetStats, 99
 - GetStrobe, 99
 - GetStrobeInfo, 99
 - GetTriggerDelay, 100
 - GetTriggerDelayInfo, 100
 - GetTriggerMode, 100
 - GetTriggerModelInfo, 101
 - IsConnected, 101
 - m_pCameraData, 112
 - ReadRegister, 102
 - ReadRegisterBlock, 102
 - RegisterAllEvents, 103
 - RegisterEvent, 103
 - ResetStats, 103
 - RestoreFromMemoryChannel, 103
 - RetrieveBuffer, 103
 - SaveToMemoryChannel, 104

- SetActiveLUTBank, [104](#)
- SetCallback, [104](#)
- SetConfiguration, [105](#)
- SetEmbeddedImageInfo, [105](#)
- SetGPIOPinDirection, [106](#)
- SetLUTChannel, [106](#)
- SetProperty, [107](#)
- SetStrobe, [107](#)
- SetTriggerDelay, [108](#)
- SetTriggerMode, [108](#)
- SetUserBuffers, [109](#)
- StartCapture, [109](#)
- StartSyncCapture, [110](#)
- StopCapture, [110](#)
- WaitForBufferEvent, [111](#)
- WriteRegister, [111](#)
- WriteRegisterBlock, [112](#)
- FlyCapture2::CameraControlDlg
 - ~CameraControlDlg, [113](#)
 - CameraControlDlg, [113](#)
 - Connect, [113](#)
 - Disconnect, [114](#)
 - Hide, [114](#)
 - IsVisible, [114](#)
 - SetTitle, [114](#)
 - Show, [114](#)
 - ShowModal, [114](#)
- FlyCapture2::CameraInfo
 - applicationIPAddress, [116](#)
 - applicationPort, [116](#)
 - bayerTileFormat, [117](#)
 - busNumber, [117](#)
 - CameraInfo, [116](#)
 - ccpStatus, [117](#)
 - configROM, [117](#)
 - defaultGateway, [117](#)
 - driverName, [117](#)
 - driverType, [117](#)
 - firmwareBuildTime, [117](#)
 - firmwareVersion, [117](#)
 - gigEMajorVersion, [117](#)
 - gigEMinorVersion, [118](#)
 - iidcVer, [118](#)
 - interfaceType, [118](#)
 - ipAddress, [118](#)
 - isColorCamera, [118](#)
 - macAddress, [118](#)
 - maximumBusSpeed, [118](#)
 - modelName, [118](#)
 - nodeNumber, [118](#)
 - pcieBusSpeed, [118](#)
 - reserved, [119](#)
 - sensorInfo, [119](#)
 - sensorResolution, [119](#)
 - serialNumber, [119](#)
 - subnetMask, [119](#)
 - userDefinedName, [119](#)
 - vendorName, [119](#)
 - xmlURL1, [119](#)
 - xmlURL2, [119](#)
- FlyCapture2::CameraSelectionDlg
 - ~CameraSelectionDlg, [120](#)
 - CameraSelectionDlg, [120](#)
 - SetTitle, [120](#)
 - ShowModal, [120](#)
- FlyCapture2::CameraStats
 - cameraCurrents, [122](#)
 - cameraPowerUp, [122](#)
 - CameraStats, [122](#)
 - cameraVoltages, [122](#)
 - imageCorrupt, [122](#)
 - imageDriverDropped, [122](#)
 - imageDropped, [122](#)
 - imageXmitFailed, [122](#)
 - numCurrents, [122](#)
 - numResendPacketsReceived, [122](#)
 - numResendPacketsRequested, [122](#)
 - numVoltages, [122](#)
 - portErrors, [122](#)
 - regReadFailed, [123](#)
 - regWriteFailed, [123](#)
 - reserved, [123](#)
 - temperature, [123](#)
 - timeSinceBusReset, [123](#)
 - timeSinceInitialization, [123](#)
 - timeStamp, [123](#)
- FlyCapture2::ConfigROM
 - chipIdHi, [124](#)
 - chipIdLo, [124](#)
 - ConfigROM, [124](#)
 - nodeVendorId, [124](#)
 - pszKeyword, [124](#)
 - reserved, [124](#)
 - unitSWVer, [125](#)
 - unitSpecId, [124](#)
 - unitSubSWVer, [125](#)
 - vendorUniqueInfo_0, [125](#)
 - vendorUniqueInfo_1, [125](#)
 - vendorUniqueInfo_2, [125](#)
 - vendorUniqueInfo_3, [125](#)
- FlyCapture2::EmbeddedImageInfo
 - brightness, [126](#)
 - exposure, [126](#)
 - frameCounter, [126](#)
 - GPIOPinState, [126](#)
 - gain, [126](#)
 - ROIPosition, [126](#)
 - shutter, [126](#)
 - strobePattern, [126](#)
 - timestamp, [126](#)
 - whiteBalance, [126](#)
- FlyCapture2::EmbeddedImageInfoProperty
 - available, [127](#)
 - EmbeddedImageInfoProperty, [127](#)
 - onOff, [127](#)
- FlyCapture2::Error

- ~Error, [129](#)
- CollectSupportInformation, [129](#)
- Error, [128](#)
- GetBuildDate, [129](#)
- GetCause, [129](#)
- GetDescription, [129](#)
- GetFilename, [129](#)
- GetLine, [130](#)
- GetType, [130](#)
- InternalError, [131](#)
- operator!=, [130](#)
- operator=, [130](#)
- operator==, [130](#)
- PrintErrorTrace, [131](#)
- FlyCapture2::EventCallbackData
 - EventData, [131](#)
 - EventDataSize, [131](#)
 - EventID, [132](#)
 - EventName, [132](#)
 - EventTimestamp, [132](#)
 - EventUserData, [132](#)
 - EventUserDataSize, [132](#)
- FlyCapture2::EventOptions
 - EventCallbackFcn, [133](#)
 - EventName, [133](#)
 - EventUserData, [133](#)
 - EventUserDataSize, [133](#)
- FlyCapture2::FC2Config
 - asyncBusSpeed, [134](#)
 - bandwidthAllocation, [134](#)
 - FC2Config, [134](#)
 - grabMode, [135](#)
 - grabTimeout, [135](#)
 - highPerformanceRetrieveBuffer, [135](#)
 - isochBusSpeed, [135](#)
 - minNumImageNotifications, [135](#)
 - numBuffers, [135](#)
 - numImageNotifications, [135](#)
 - registerTimeout, [136](#)
 - registerTimeoutRetries, [136](#)
 - reserved, [136](#)
- FlyCapture2::FC2Version
 - build, [137](#)
 - major, [137](#)
 - minor, [137](#)
 - type, [137](#)
- FlyCapture2::Format7ImageSettings
 - Format7ImageSettings, [139](#)
 - height, [139](#)
 - mode, [139](#)
 - offsetX, [139](#)
 - offsetY, [140](#)
 - pixelFormat, [140](#)
 - reserved, [140](#)
 - width, [140](#)
- FlyCapture2::Format7Info
 - Format7Info, [141](#)
 - imageHStepSize, [141](#)
 - imageVStepSize, [141](#)
 - maxHeight, [141](#)
 - maxPacketSize, [142](#)
 - maxWidth, [142](#)
 - minPacketSize, [142](#)
 - mode, [142](#)
 - offsetHStepSize, [142](#)
 - offsetVStepSize, [142](#)
 - packetSize, [142](#)
 - percentage, [142](#)
 - pixelFormatBitField, [142](#)
 - reserved, [142](#)
 - vendorPixelFormatBitField, [143](#)
- FlyCapture2::Format7PacketInfo
 - Format7PacketInfo, [143](#)
 - maxBytesPerPacket, [143](#)
 - recommendedBytesPerPacket, [143](#)
 - reserved, [144](#)
 - unitBytesPerPacket, [144](#)
- FlyCapture2::GCCamera
 - ~GCCamera, [147](#)
 - Connect, [147](#)
 - Disconnect, [147](#)
 - EnableLUT, [147](#)
 - FireSoftwareTrigger, [148](#)
 - GCCamera, [147](#)
 - GCCamera::GetXML, [148](#)
 - GetActiveLUTBank, [148](#)
 - GetCameraInfo, [148](#)
 - GetConfiguration, [149](#)
 - GetCycleTime, [149](#)
 - GetEmbeddedImageInfo, [149](#)
 - GetGPIOPinDirection, [150](#)
 - GetInterfaceType, [150](#)
 - GetLUTBankInfo, [150](#)
 - GetLUTChannel, [150](#)
 - GetLUTInfo, [151](#)
 - GetMemoryChannel, [151](#)
 - GetMemoryChannelInfo, [152](#)
 - GetNodeMap, [152](#)
 - GetProperty, [152](#)
 - GetPropertyInfo, [153](#)
 - GetRegisterString, [153](#)
 - GetStats, [153](#)
 - GetStrobe, [153](#)
 - GetStrobeInfo, [154](#)
 - GetTriggerDelay, [154](#)
 - GetTriggerDelayInfo, [155](#)
 - GetTriggerMode, [155](#)
 - GetTriggerModeInfo, [156](#)
 - IsConnected, [156](#)
 - m_busMgr, [166](#)
 - ReadGVCPMemory, [156](#)
 - ReadGVCPRegister, [156](#)
 - ReadGVCPRegisterBlock, [156](#)
 - ReadRegister, [156](#)
 - ReadRegisterBlock, [157](#)
 - ResetStats, [157](#)

- RestoreFromMemoryChannel, 157
- RetrieveBuffer, 158
- SaveToMemoryChannel, 158
- SetActiveLUTBank, 159
- SetCallback, 159
- SetCamera, 159, 160
- SetConfiguration, 160
- SetEmbeddedImageInfo, 160
- SetGPIOPinDirection, 160
- SetLUTChannel, 161
- SetProperty, 161
- SetStrobe, 162
- SetTriggerDelay, 162
- SetTriggerMode, 163
- SetUserBuffers, 163
- StartCapture, 164
- StartSyncCapture, 164
- StopCapture, 164
- TestGainNode, 165
- WaitForBufferEvent, 165
- WriteGVCPMemory, 165
- WriteGVCPRegister, 165
- WriteGVCPRegisterBlock, 165
- WriteRegister, 165
- WriteRegisterBlock, 166
- FlyCapture2::GigECamera
 - ~GigECamera, 171
 - Connect, 171
 - DeregisterAllEvents, 171
 - DeregisterEvent, 171
 - Disconnect, 171
 - DiscoverGigEPacketSize, 171
 - EnableLUT, 172
 - FireSoftwareTrigger, 172
 - GetActiveLUTBank, 172
 - GetCameraInfo, 173
 - GetConfiguration, 173
 - GetCycleTime, 173
 - GetEmbeddedImageInfo, 174
 - GetGPIOPinDirection, 176
 - GetGigEConfig, 174
 - GetGigEImageBinningSettings, 174
 - GetGigEImageSettings, 175
 - GetGigEImageSettingsInfo, 175
 - GetGigEImagingMode, 175
 - GetGigEProperty, 175
 - GetGigEStreamChannelInfo, 176
 - GetLUTBankInfo, 176
 - GetLUTChannel, 177
 - GetLUTInfo, 177
 - GetMemoryChannel, 178
 - GetMemoryChannelInfo, 178
 - GetNumStreamChannels, 179
 - GetProperty, 179
 - GetPropertyInfo, 179
 - GetRegisterString, 181
 - GetStats, 181
 - GetStrobe, 181
 - GetStrobeInfo, 181
 - GetTriggerDelay, 182
 - GetTriggerDelayInfo, 182
 - GetTriggerMode, 183
 - GetTriggerModeInfo, 183
 - GigECamera, 171
 - IsConnected, 184
 - QueryGigEImagingMode, 184
 - ReadGVCPMemory, 184
 - ReadGVCPRegister, 185
 - ReadGVCPRegisterBlock, 185
 - ReadRegister, 185
 - ReadRegisterBlock, 186
 - RegisterAllEvents, 186
 - RegisterEvent, 186
 - ResetStats, 186
 - RestoreFromMemoryChannel, 186
 - RetrieveBuffer, 187
 - SaveToMemoryChannel, 187
 - SetActiveLUTBank, 188
 - SetCallback, 188
 - SetConfiguration, 188
 - SetEmbeddedImageInfo, 190
 - SetGPIOPinDirection, 192
 - SetGigEConfig, 190
 - SetGigEImageBinningSettings, 190
 - SetGigEImageSettings, 191
 - SetGigEImagingMode, 191
 - SetGigEProperty, 191
 - SetGigEStreamChannelInfo, 192
 - SetLUTChannel, 192
 - SetProperty, 193
 - SetStrobe, 193
 - SetTriggerDelay, 194
 - SetTriggerMode, 194
 - SetUserBuffers, 195
 - StartCapture, 195
 - StartSyncCapture, 196
 - StopCapture, 196
 - WaitForBufferEvent, 196
 - WriteGVCPMemory, 197
 - WriteGVCPRegister, 197
 - WriteGVCPRegisterBlock, 197
 - WriteRegister, 198
 - WriteRegisterBlock, 198
- FlyCapture2::GigEConfig
 - enablePacketResend, 199
 - GigEConfig, 199
 - registerTimeout, 199
 - registerTimeoutRetries, 200
- FlyCapture2::GigEImageSettings
 - GigEImageSettings, 201
 - height, 201
 - offsetX, 201
 - offsetY, 201
 - pixelFormat, 201
 - reserved, 201
 - width, 201

- FlyCapture2::GigEImageSettingsInfo
 - GigEImageSettingsInfo, 202
 - imageHStepSize, 202
 - imageVStepSize, 202
 - maxHeight, 202
 - maxWidth, 202
 - offsetHStepSize, 202
 - offsetVStepSize, 203
 - pixelFormatBitField, 203
 - reserved, 203
 - vendorPixelFormatBitField, 203
- FlyCapture2::GigEProperty
 - isReadable, 204
 - isWritable, 204
 - max, 204
 - min, 204
 - propType, 204
 - value, 204
- FlyCapture2::GigEStreamChannel
 - destinationIpAddress, 205
 - doNotFragment, 205
 - GigEStreamChannel, 205
 - hostPort, 205
 - interPacketDelay, 205
 - networkInterfaceIndex, 206
 - packetSize, 206
 - sourcePort, 206
- FlyCapture2::H264Option
 - bitrate, 207
 - frameRate, 207
 - H264Option, 207
 - height, 207
 - reserved, 207
 - width, 207
- FlyCapture2::IPAddress
 - IPAddress, 231
 - octets, 231
 - operator!=, 231
 - operator==, 231
- FlyCapture2::Image
 - ~Image, 211
 - CalculateStatistics, 212
 - Convert, 212
 - DeepCopy, 212
 - DetermineBitsPerPixel, 213
 - GetBayerTileFormat, 213
 - GetBitsPerPixel, 213
 - GetBlockId, 213
 - GetColorProcessing, 214
 - GetCols, 214
 - GetData, 214
 - GetDataSize, 214
 - GetDefaultColorProcessing, 214
 - GetDefaultOutputFormat, 215
 - GetDimensions, 215
 - GetMetadata, 215
 - GetPixelFormat, 215
 - GetReceivedDataSize, 216
 - GetRows, 216
 - GetStride, 216
 - GetTimeStamp, 216
 - Image, 210, 211
 - Iso, 222
 - operator(), 216
 - operator=, 217
 - operator[], 217
 - ReleaseBuffer, 217
 - Save, 217–219
 - SetBlockId, 220
 - SetColorProcessing, 220
 - SetData, 220
 - SetDefaultColorProcessing, 221
 - SetDefaultOutputFormat, 221
 - SetDimensions, 221
- FlyCapture2::ImageMetadata
 - embeddedBrightness, 223
 - embeddedExposure, 223
 - embeddedFrameCounter, 223
 - embeddedGPIOPinState, 223
 - embeddedGain, 223
 - embeddedROIPosition, 223
 - embeddedShutter, 223
 - embeddedStrobePattern, 223
 - embeddedTimeStamp, 223
 - embeddedWhiteBalance, 224
 - ImageMetadata, 223
 - reserved, 224
- FlyCapture2::ImageStatistics
 - ~ImageStatistics, 226
 - BLUE, 225
 - DisableAll, 226
 - EnableAll, 226
 - EnableGreyOnly, 226
 - EnableHSLOnly, 226
 - EnableRGBOnly, 227
 - GREEN, 225
 - GREY, 225
 - GetChannelStatus, 227
 - GetHistogram, 227
 - GetMean, 227
 - GetNumPixelValues, 228
 - GetPixelValueRange, 228
 - GetRange, 228
 - GetStatistics, 229
 - HUE, 225
 - ImageStatistics, 226
 - ImageStatsCalculator, 230
 - LIGHTNESS, 225
 - NUM_STATISTICS_CHANNELS, 225
 - operator=, 229
 - RED, 225
 - SATURATION, 225
 - SetChannelStatus, 229
 - StatisticsChannel, 225
- FlyCapture2::Internal
 - GetInternal, 230

- FlyCapture2::JPEGOption
 - JPEGOption, [232](#)
 - progressive, [232](#)
 - quality, [232](#)
 - reserved, [232](#)
- FlyCapture2::JPG2Option
 - JPG2Option, [233](#)
 - quality, [233](#)
 - reserved, [233](#)
- FlyCapture2::LUTData
 - enabled, [234](#)
 - inputBitDepth, [234](#)
 - LUTData, [234](#)
 - numBanks, [234](#)
 - numChannels, [234](#)
 - numEntries, [235](#)
 - outputBitDepth, [235](#)
 - reserved, [235](#)
 - supported, [235](#)
- FlyCapture2::MACAddress
 - MACAddress, [236](#)
 - octets, [236](#)
 - operator!=, [236](#)
 - operator==, [236](#)
- FlyCapture2::MJPGOption
 - frameRate, [237](#)
 - MJPGOption, [237](#)
 - quality, [237](#)
 - reserved, [237](#)
- FlyCapture2::NodeMap
 - _GetDeviceName, [238](#)
 - _GetNode, [238](#)
 - _GetNodes, [238](#)
 - _InvalidateNodes, [238](#)
 - _Poll, [238](#)
 - ~NodeMap, [238](#)
 - NodeMap, [238](#)
- FlyCapture2::PGMOption
 - binaryFile, [239](#)
 - PGMOption, [239](#)
 - reserved, [239](#)
- FlyCapture2::PGRGuid
 - operator!=, [240](#)
 - operator==, [240](#)
 - PGRGuid, [240](#)
 - value, [240](#)
- FlyCapture2::PNGOption
 - compressionLevel, [241](#)
 - interlaced, [241](#)
 - PNGOption, [241](#)
 - reserved, [241](#)
- FlyCapture2::PPMOption
 - binaryFile, [242](#)
 - PPMOption, [242](#)
 - reserved, [242](#)
- FlyCapture2::Property
 - absControl, [243](#)
 - absValue, [243](#)
 - autoManualMode, [244](#)
 - onOff, [244](#)
 - onePush, [244](#)
 - present, [244](#)
 - Property, [243](#)
 - reserved, [244](#)
 - type, [244](#)
 - valueA, [244](#)
 - valueB, [244](#)
- FlyCapture2::PropertyInfo
 - absMax, [246](#)
 - absMin, [246](#)
 - absValSupported, [246](#)
 - autoSupported, [246](#)
 - manualSupported, [246](#)
 - max, [246](#)
 - min, [246](#)
 - onOffSupported, [246](#)
 - onePushSupported, [246](#)
 - pUnitAbbr, [247](#)
 - pUnits, [247](#)
 - present, [247](#)
 - PropertyInfo, [246](#)
 - readOutSupported, [247](#)
 - reserved, [247](#)
 - type, [247](#)
- FlyCapture2::StrobeControl
 - delay, [248](#)
 - duration, [248](#)
 - onOff, [248](#)
 - polarity, [248](#)
 - reserved, [248](#)
 - source, [248](#)
 - StrobeControl, [248](#)
- FlyCapture2::StrobeInfo
 - maxValue, [249](#)
 - minValue, [249](#)
 - onOffSupported, [250](#)
 - polaritySupported, [250](#)
 - present, [250](#)
 - readOutSupported, [250](#)
 - reserved, [250](#)
 - source, [250](#)
 - StrobeInfo, [249](#)
- FlyCapture2::SystemInfo
 - byteOrder, [252](#)
 - cpuDescription, [252](#)
 - driverList, [252](#)
 - gpuDescription, [252](#)
 - libraryList, [253](#)
 - numCpuCores, [253](#)
 - osDescription, [253](#)
 - osType, [253](#)
 - reserved, [253](#)
 - screenHeight, [253](#)
 - screenWidth, [253](#)
 - sysMemSize, [253](#)
- FlyCapture2::TIFFOption

- ADOBE_DEFLATE, 254
- CCITTFAX3, 254
- CCITTFAX4, 254
- compression, 255
- CompressionMethod, 254
- DEFLATE, 254
- JPEG, 254
- LZW, 254
- NONE, 254
- PACKBITS, 254
- reserved, 255
- TIFFOption, 255
- FlyCapture2::TimeStamp
 - cycleCount, 256
 - cycleOffset, 256
 - cycleSeconds, 256
 - microSeconds, 256
 - reserved, 256
 - seconds, 256
 - TimeStamp, 256
- FlyCapture2::TopologyNode
 - ~TopologyNode, 258
 - AddChild, 259
 - AddPortType, 259
 - AssignGuidToNode, 259
 - BUS, 258
 - CAMERA, 258
 - COMPUTER, 258
 - CONNECTED_TO_CHILD, 258
 - CONNECTED_TO_PARENT, 258
 - GetChild, 260
 - GetDeviceId, 260
 - GetGuid, 260
 - GetInterfaceType, 260
 - GetNodeType, 260
 - GetNumChildren, 261
 - GetNumPorts, 261
 - GetPortType, 261
 - NODE, 258
 - NOT_CONNECTED, 258
 - NodeType, 258
 - operator=, 261
 - PortType, 258
 - TopologyNode, 258, 259
- FlyCapture2::TriggerMode
 - mode, 262
 - onOff, 262
 - parameter, 262
 - polarity, 263
 - reserved, 263
 - source, 263
 - TriggerMode, 262
- FlyCapture2::TriggerModelInfo
 - modeMask, 264
 - onOffSupported, 264
 - polaritySupported, 264
 - present, 264
 - readOutSupported, 264
 - reserved, 264
 - softwareTriggerSupported, 264
 - sourceMask, 264
 - TriggerModelInfo, 264
 - valueReadable, 264
- FlyCapture2::Utilities
 - CheckDriver, 265
 - GetDriverDeviceName, 266
 - GetLibraryVersion, 266
 - GetSystemInfo, 266
 - LaunchBrowser, 267
 - LaunchCommand, 267
 - LaunchCommandAsync, 267
 - LaunchHelp, 268
- FlyCapture2Defs.h, 271
 - FULL_32BIT_VALUE, 278
 - NULL, 278
- FlyCapture2GUI.h, 278
- FlyCapture2Platform.h, 279
 - FLYCAPTURE2_API, 279
 - FLYCAPTURE2_LOCAL, 279
- FlyCapture3ApiGuiWrapper, 138
 - FlyCap3CameraControl::FlyCapture3ApiGui↔Wrapper, 138
- FlyCapture3ApiGuiWrapper.h, 279
 - WRAPPER_API, 279
- ForceAllIPAddressesAutomatically
 - FlyCapture2::BusManager, 53, 54
- ForceIPAddressToCamera
 - FlyCapture2::BusManager, 54
- Format7ImageSettings, 139
 - FlyCapture2::Format7ImageSettings, 139
- Format7Info, 140
 - FlyCapture2::Format7Info, 141
- Format7PacketInfo, 143
 - FlyCapture2::Format7PacketInfo, 143
- frameCounter
 - FlyCapture2::EmbeddedImageInfo, 126
- FrameRate
 - Enumerations, 22
- frameRate
 - FlyCapture2::AVIOption, 47
 - FlyCapture2::H264Option, 207
 - FlyCapture2::MJPGOption, 237
- GAIN
 - Enumerations, 26
- GAMMA
 - Enumerations, 26
- GBRG
 - Enumerations, 19
- GCCamera, 144
 - FlyCapture2::GCCamera, 147
- GCCamera.h, 279
- GCCamera::GetXML
 - FlyCapture2::GCCamera, 148
- GPIOPinState
 - FlyCapture2::EmbeddedImageInfo, 126
- GRAB_MODE_FORCE_32BITS

- Enumerations, [23](#)
- GRAB_TIMEOUT_FORCE_32BITS
 - Enumerations, [23](#)
- GRBG
 - Enumerations, [19](#)
- GREEN
 - FlyCapture2::ImageStatistics, [225](#)
- GREY
 - FlyCapture2::ImageStatistics, [225](#)
- gain
 - FlyCapture2::EmbeddedImageInfo, [126](#)
- GetActiveLUTBank
 - FlyCapture2::Camera, [65](#)
 - FlyCapture2::CameraBase, [93](#)
 - FlyCapture2::GCCamera, [148](#)
 - FlyCapture2::GigECamera, [172](#)
- GetBayerTileFormat
 - FlyCapture2::Image, [213](#)
- GetBitsPerPixel
 - FlyCapture2::Image, [213](#)
- GetBlockId
 - FlyCapture2::Image, [213](#)
- GetBuildDate
 - FlyCapture2::Error, [129](#)
- GetCameraFromIPAddress
 - FlyCapture2::BusManager, [55](#)
- GetCameraFromIndex
 - FlyCapture2::BusManager, [54](#)
- GetCameraFromSerialNumber
 - FlyCapture2::BusManager, [55](#)
- GetCameraInfo
 - FlyCapture2::Camera, [65](#)
 - FlyCapture2::CameraBase, [93](#)
 - FlyCapture2::GCCamera, [148](#)
 - FlyCapture2::GigECamera, [173](#)
- GetCameraSerialNumberFromIndex
 - FlyCapture2::BusManager, [55](#)
- GetCause
 - FlyCapture2::Error, [129](#)
- GetChannelStatus
 - FlyCapture2::ImageStatistics, [227](#)
- GetChild
 - FlyCapture2::TopologyNode, [260](#)
- GetColorProcessing
 - FlyCapture2::Image, [214](#)
- GetCols
 - FlyCapture2::Image, [214](#)
- GetConfiguration
 - FlyCapture2::Camera, [66](#)
 - FlyCapture2::CameraBase, [94](#)
 - FlyCapture2::GCCamera, [149](#)
 - FlyCapture2::GigECamera, [173](#)
- GetControlNameList
 - FlyCap3CameraControl::FlyCapture3ApiGui↔
Wrapper, [138](#)
- GetCycleTime
 - FlyCapture2::Camera, [66](#)
 - FlyCapture2::CameraBase, [94](#)
- FlyCapture2::GCCamera, [149](#)
- FlyCapture2::GigECamera, [173](#)
- GetData
 - FlyCapture2::Image, [214](#)
- GetDataSize
 - FlyCapture2::Image, [214](#)
- GetDefaultColorProcessing
 - FlyCapture2::Image, [214](#)
- GetDefaultOutputFormat
 - FlyCapture2::Image, [215](#)
- GetDescription
 - FlyCapture2::Error, [129](#)
- GetDeviceFromIndex
 - FlyCapture2::BusManager, [56](#)
- GetDeviceId
 - FlyCapture2::TopologyNode, [260](#)
- GetDialogNameList
 - FlyCap3CameraControl::FlyCapture3ApiGui↔
Wrapper, [138](#)
- GetDimensions
 - FlyCapture2::Image, [215](#)
- GetDriverDeviceName
 - FlyCapture2::Utilities, [266](#)
- GetEmbeddedImageInfo
 - FlyCapture2::Camera, [66](#)
 - FlyCapture2::CameraBase, [94](#)
 - FlyCapture2::GCCamera, [149](#)
 - FlyCapture2::GigECamera, [174](#)
- GetFilename
 - FlyCapture2::Error, [129](#)
- GetFormat7Configuration
 - FlyCapture2::Camera, [67](#)
- GetFormat7Info
 - FlyCapture2::Camera, [67](#)
- GetGPIOPinDirection
 - FlyCapture2::Camera, [68](#)
 - FlyCapture2::CameraBase, [95](#)
 - FlyCapture2::GCCamera, [150](#)
 - FlyCapture2::GigECamera, [176](#)
- GetGigEConfig
 - FlyCapture2::GigECamera, [174](#)
- GetGigEImageBinningSettings
 - FlyCapture2::GigECamera, [174](#)
- GetGigEImageSettings
 - FlyCapture2::GigECamera, [175](#)
- GetGigEImageSettingsInfo
 - FlyCapture2::GigECamera, [175](#)
- GetGigEImagingMode
 - FlyCapture2::GigECamera, [175](#)
- GetGigEProperty
 - FlyCapture2::GigECamera, [175](#)
- GetGigEStreamChannelInfo
 - FlyCapture2::GigECamera, [176](#)
- GetGuid
 - FlyCapture2::TopologyNode, [260](#)
- GetHistogram
 - FlyCapture2::ImageStatistics, [227](#)
- GetInterfaceType

- FlyCapture2::GCCamera, 150
- FlyCapture2::TopologyNode, 260
- GetInterfaceTypeFromGuid
 - FlyCapture2::BusManager, 56
- GetInternal
 - FlyCapture2::Internal, 230
- GetLUTBankInfo
 - FlyCapture2::Camera, 68
 - FlyCapture2::CameraBase, 95
 - FlyCapture2::GCCamera, 150
 - FlyCapture2::GigECamera, 176
- GetLUTChannel
 - FlyCapture2::Camera, 69
 - FlyCapture2::CameraBase, 96
 - FlyCapture2::GCCamera, 150
 - FlyCapture2::GigECamera, 177
- GetLUTInfo
 - FlyCapture2::Camera, 69
 - FlyCapture2::CameraBase, 96
 - FlyCapture2::GCCamera, 151
 - FlyCapture2::GigECamera, 177
- GetLibraryVersion
 - FlyCapture2::Utilities, 266
- GetLine
 - FlyCapture2::Error, 130
- GetMean
 - FlyCapture2::ImageStatistics, 227
- GetMemoryChannel
 - FlyCapture2::Camera, 70
 - FlyCapture2::CameraBase, 97
 - FlyCapture2::GCCamera, 151
 - FlyCapture2::GigECamera, 178
- GetMemoryChannelInfo
 - FlyCapture2::Camera, 70
 - FlyCapture2::CameraBase, 97
 - FlyCapture2::GCCamera, 152
 - FlyCapture2::GigECamera, 178
- GetMetadata
 - FlyCapture2::Image, 215
- GetNodeMap
 - FlyCapture2::GCCamera, 152
- GetNodeType
 - FlyCapture2::TopologyNode, 260
- GetNumChildren
 - FlyCapture2::TopologyNode, 261
- GetNumDialogs
 - FlyCap3CameraControl::FlyCapture3ApiGui↔
Wrapper, 138
- GetNumOfCameras
 - FlyCapture2::BusManager, 57
- GetNumOfControls
 - FlyCap3CameraControl::FlyCapture3ApiGui↔
Wrapper, 138
- GetNumOfDevices
 - FlyCapture2::BusManager, 57
- GetNumPixelValues
 - FlyCapture2::ImageStatistics, 228
- GetNumPorts
 - FlyCapture2::TopologyNode, 261
- GetNumStreamChannels
 - FlyCapture2::GigECamera, 179
- GetPixelFormat
 - FlyCapture2::Image, 215
- GetPixelValueRange
 - FlyCapture2::ImageStatistics, 228
- GetPortType
 - FlyCapture2::TopologyNode, 261
- GetProperty
 - FlyCapture2::Camera, 70
 - FlyCapture2::CameraBase, 97
 - FlyCapture2::GCCamera, 152
 - FlyCapture2::GigECamera, 179
- GetPropertyInfo
 - FlyCapture2::Camera, 71
 - FlyCapture2::CameraBase, 98
 - FlyCapture2::GCCamera, 153
 - FlyCapture2::GigECamera, 179
- GetRange
 - FlyCapture2::ImageStatistics, 228
- GetReceivedDataSize
 - FlyCapture2::Image, 216
- GetRegisterString
 - FlyCapture2::Camera, 71
 - FlyCapture2::CameraBase, 98
 - FlyCapture2::GCCamera, 153
 - FlyCapture2::GigECamera, 181
- GetRows
 - FlyCapture2::Image, 216
- GetStatistics
 - FlyCapture2::ImageStatistics, 229
- GetStats
 - FlyCapture2::Camera, 71
 - FlyCapture2::CameraBase, 99
 - FlyCapture2::GCCamera, 153
 - FlyCapture2::GigECamera, 181
- GetStride
 - FlyCapture2::Image, 216
- GetStrobe
 - FlyCapture2::Camera, 71
 - FlyCapture2::CameraBase, 99
 - FlyCapture2::GCCamera, 153
 - FlyCapture2::GigECamera, 181
- GetStrobeInfo
 - FlyCapture2::Camera, 72
 - FlyCapture2::CameraBase, 99
 - FlyCapture2::GCCamera, 154
 - FlyCapture2::GigECamera, 181
- GetSyncStatus
 - MultiSyncLibrary::SyncManager, 251
- GetSystemInfo
 - FlyCapture2::Utilities, 266
- GetTimeSinceSynced
 - MultiSyncLibrary::SyncManager, 251
- GetTimeStamp
 - FlyCapture2::Image, 216
- GetTopology

- FlyCapture2::BusManager, 57
- GetTriggerDelay
 - FlyCapture2::Camera, 72
 - FlyCapture2::CameraBase, 100
 - FlyCapture2::GCCamera, 154
 - FlyCapture2::GigECamera, 182
- GetTriggerDelayInfo
 - FlyCapture2::Camera, 73
 - FlyCapture2::CameraBase, 100
 - FlyCapture2::GCCamera, 155
 - FlyCapture2::GigECamera, 182
- GetTriggerMode
 - FlyCapture2::Camera, 73
 - FlyCapture2::CameraBase, 100
 - FlyCapture2::GCCamera, 155
 - FlyCapture2::GigECamera, 183
- GetTriggerModelInfo
 - FlyCapture2::Camera, 74
 - FlyCapture2::CameraBase, 101
 - FlyCapture2::GCCamera, 156
 - FlyCapture2::GigECamera, 183
- GetType
 - FlyCapture2::Error, 130
- GetUsbLinkInfo
 - FlyCapture2::BusManager, 57
- GetUsbPortStatus
 - FlyCapture2::BusManager, 58
- GetVideoModeAndFrameRate
 - FlyCapture2::Camera, 74
- GetVideoModeAndFrameRateInfo
 - FlyCapture2::Camera, 75
- GigE specific enumerations, 28
 - GigEPropertyType, 28
 - HEARTBEAT_TIMEOUT, 28
 - HEARTBEAT, 28
 - PACKET_DELAY, 28
 - PACKET_SIZE, 28
- GigE specific structures, 31
- GigECamera, 167
 - FlyCapture2::GigECamera, 171
- GigECamera.h, 280
- GigEConfig, 199
 - FlyCapture2::GigEConfig, 199
- GigEImageSettings, 200
 - FlyCapture2::GigEImageSettings, 201
- GigEImageSettingsInfo, 201
 - FlyCapture2::GigEImageSettingsInfo, 202
- gigEMajorVersion
 - FlyCapture2::CameraInfo, 117
- gigEMinorVersion
 - FlyCapture2::CameraInfo, 118
- GigEProperty, 203
- GigEPropertyType
 - GigE specific enumerations, 28
- GigEStreamChannel, 204
 - FlyCapture2::GigEStreamChannel, 205
- Global constants, 13
 - sk_maxNumPorts, 13
 - sk_maxStringLength, 13
- gpuDescription
 - FlyCapture2::SystemInfo, 252
- GrabMode
 - Enumerations, 22
- grabMode
 - FlyCapture2::FC2Config, 135
- GrabTimeout
 - Enumerations, 23
- grabTimeout
 - FlyCapture2::FC2Config, 135
- H264Option, 206
 - FlyCapture2::H264Option, 207
- HEARTBEAT_TIMEOUT
 - GigE specific enumerations, 28
- HEARTBEAT
 - GigE specific enumerations, 28
- HQ_LINEAR
 - Enumerations, 20
- HUE
 - Enumerations, 26
 - FlyCapture2::ImageStatistics, 225
- height
 - FlyCapture2::Format7ImageSettings, 139
 - FlyCapture2::GigEImageSettings, 201
 - FlyCapture2::H264Option, 207
- Hide
 - FlyCapture2::CameraControlDlg, 114
- highPerformanceRetrieveBuffer
 - FlyCapture2::FC2Config, 135
- hostPort
 - FlyCapture2::GigEStreamChannel, 205
- IIDC specific structures, 32
- IMAGE_FILE_FORMAT_FORCE_32BITS
 - Enumerations, 23
- 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
- IPAddress, 230
 - FlyCapture2::IPAddress, 231
- IPP
 - Enumerations, 20
- IRIS
 - Enumerations, 26
- iidcVer
 - FlyCapture2::CameraInfo, 118
- Image, 207
 - FlyCapture2::Image, 210, 211

- Image saving structures., 33
 - CameraEventCallback, 33
- Image.h, 280
- imageCorrupt
 - FlyCapture2::CameraStats, 122
- imageDriverDropped
 - FlyCapture2::CameraStats, 122
- imageDropped
 - FlyCapture2::CameraStats, 122
- ImageEventCallback
 - FlyCapture2, 44
- ImageFileFormat
 - Enumerations, 23
- imageHStepSize
 - FlyCapture2::Format7Info, 141
 - FlyCapture2::GigEImageSettingsInfo, 202
- ImageMetadata, 222
 - FlyCapture2::ImageMetadata, 223
- ImageStatistics, 224
 - FlyCapture2::ImageStatistics, 226
- ImageStatistics.h, 280
- ImageStatsCalculator
 - FlyCapture2::ImageStatistics, 230
- imageVStepSize
 - FlyCapture2::Format7Info, 141
 - FlyCapture2::GigEImageSettingsInfo, 202
- imageXmitFailed
 - FlyCapture2::CameraStats, 122
- indexedColor_8bit
 - FlyCapture2::BMPOption, 51
- inputBitDepth
 - FlyCapture2::LUTData, 234
- interPacketDelay
 - FlyCapture2::GigEStreamChannel, 205
- InterfaceType
 - Enumerations, 23
- interfaceType
 - FlyCapture2::CameraInfo, 118
- interlaced
 - FlyCapture2::PNGOption, 241
- Internal, 230
- Internal.h, 281
- InternalError
 - FlyCapture2::Error, 131
- ipAddress
 - FlyCapture2::CameraInfo, 118
- IsCameraControllable
 - FlyCapture2::BusManager, 58
- isColorCamera
 - FlyCapture2::CameraInfo, 118
- IsConnected
 - FlyCapture2::Camera, 75
 - FlyCapture2::CameraBase, 101
 - FlyCapture2::GCCamera, 156
 - FlyCapture2::GigECamera, 184
- isReadable
 - FlyCapture2::GigEProperty, 204
- IsTimingBusConnected
 - MultiSyncLibrary::SyncManager, 251
- IsVisible
 - FlyCapture2::CameraControlDlg, 114
- isWritable
 - FlyCapture2::GigEProperty, 204
- Iso
 - FlyCapture2::Image, 222
- isochBusSpeed
 - FlyCapture2::FC2Config, 135
- JPEG2000
 - Enumerations, 23
- JPEGOOption, 231
 - FlyCapture2::JPEGOOption, 232
- JPEG
 - Enumerations, 23
 - FlyCapture2::TIFFOption, 254
- JPG2Option, 233
 - FlyCapture2::JPG2Option, 233
- LIGHTNESS
 - FlyCapture2::ImageStatistics, 225
- LINUX_X64
 - FlyCapture2, 44
- LINUX_X86
 - FlyCapture2, 44
- LUTData, 233
 - FlyCapture2::LUTData, 234
- LZW
 - FlyCapture2::TIFFOption, 254
- LaunchBrowser
 - FlyCapture2::Utilities, 267
- LaunchCommand
 - FlyCapture2::Utilities, 267
- LaunchCommandAsync
 - FlyCapture2::Utilities, 267
- LaunchHelp
 - FlyCapture2::Utilities, 268
- libraryList
 - FlyCapture2::SystemInfo, 253
- m_busMgr
 - FlyCapture2::GCCamera, 166
- m_pCameraData
 - FlyCapture2::CameraBase, 112
- MACAddress, 235
 - FlyCapture2::MACAddress, 236
- MAC
 - FlyCapture2, 44
- MJPGOption, 236
 - FlyCapture2::MJPGOption, 237
- MODE_0
 - Enumerations, 24
- MODE_1
 - Enumerations, 24
- MODE_10
 - Enumerations, 24
- MODE_11
 - Enumerations, 24

- MODE_12
 - Enumerations, [24](#)
- MODE_13
 - Enumerations, [24](#)
- MODE_14
 - Enumerations, [24](#)
- MODE_15
 - Enumerations, [24](#)
- MODE_16
 - Enumerations, [24](#)
- MODE_17
 - Enumerations, [24](#)
- MODE_18
 - Enumerations, [24](#)
- MODE_19
 - Enumerations, [24](#)
- MODE_2
 - Enumerations, [24](#)
- MODE_20
 - Enumerations, [24](#)
- MODE_21
 - Enumerations, [24](#)
- MODE_22
 - Enumerations, [24](#)
- MODE_23
 - Enumerations, [24](#)
- MODE_24
 - Enumerations, [24](#)
- MODE_25
 - Enumerations, [24](#)
- MODE_26
 - Enumerations, [24](#)
- MODE_27
 - Enumerations, [24](#)
- MODE_28
 - Enumerations, [24](#)
- MODE_29
 - Enumerations, [24](#)
- MODE_3
 - Enumerations, [24](#)
- MODE_30
 - Enumerations, [24](#)
- MODE_31
 - Enumerations, [24](#)
- MODE_4
 - Enumerations, [24](#)
- MODE_5
 - Enumerations, [24](#)
- MODE_6
 - Enumerations, [24](#)
- MODE_7
 - Enumerations, [24](#)
- MODE_8
 - Enumerations, [24](#)
- MODE_9
 - Enumerations, [24](#)
- MODE_FORCE_32BITS
 - Enumerations, [24](#)
- MULTISYNCLIBRARY_API
 - MultiSyncLibraryPlatform.h, [282](#)
- MULTISYNCLIBRARY_LOCAL
 - MultiSyncLibraryPlatform.h, [282](#)
- macAddress
 - FlyCapture2::CameraInfo, [118](#)
- major
 - FlyCapture2::FC2Version, [137](#)
- manualSupported
 - FlyCapture2::PropertyInfo, [246](#)
- max
 - FlyCapture2::GigEProperty, [204](#)
 - FlyCapture2::PropertyInfo, [246](#)
- maxBytesPerPacket
 - FlyCapture2::Format7PacketInfo, [143](#)
- maxHeight
 - FlyCapture2::Format7Info, [141](#)
 - FlyCapture2::GigEImageSettingsInfo, [202](#)
- maxPacketSize
 - FlyCapture2::Format7Info, [142](#)
- maxValue
 - FlyCapture2::StrobeInfo, [249](#)
- maxWidth
 - FlyCapture2::Format7Info, [142](#)
 - FlyCapture2::GigEImageSettingsInfo, [202](#)
- maximumBusSpeed
 - FlyCapture2::CameraInfo, [118](#)
- microSeconds
 - FlyCapture2::TimeStamp, [256](#)
- min
 - FlyCapture2::GigEProperty, [204](#)
 - FlyCapture2::PropertyInfo, [246](#)
- minNumImageNotifications
 - FlyCapture2::FC2Config, [135](#)
- minPacketSize
 - FlyCapture2::Format7Info, [142](#)
- minValue
 - FlyCapture2::StrobeInfo, [249](#)
- minor
 - FlyCapture2::FC2Version, [137](#)
- Mode
 - Enumerations, [24](#)
- mode
 - FlyCapture2::Format7ImageSettings, [139](#)
 - FlyCapture2::Format7Info, [142](#)
 - FlyCapture2::TriggerMode, [262](#)
- modeMask
 - FlyCapture2::TriggerModelInfo, [264](#)
- modelName
 - FlyCapture2::CameraInfo, [118](#)
- MultiSyncLibrary, [45](#)
 - PGRSyncError, [45](#)
 - PGRSyncError_ALREADY_STARTED, [45](#)
 - PGRSyncError_ALREADY_STOPPED, [45](#)
 - PGRSyncError_CAMERA_NOT_FOUND, [45](#)
 - PGRSyncError_FAILED, [45](#)
 - PGRSyncError_OK, [45](#)
 - PGRSyncError_UNKNOWN_ERROR, [45](#)

- PGRSyncMessage, [45](#)
- PGRSyncMessage_BUS_RESET, [46](#)
- PGRSyncMessage_DEVICE_ERROR, [46](#)
- PGRSyncMessage_NOMASTER, [45](#)
- PGRSyncMessage_NOT_ENOUGH_DEVICES, [46](#)
- PGRSyncMessage_NOT_INITIALIZED, [46](#)
- PGRSyncMessage_OK, [45](#)
- PGRSyncMessage_STARTED, [45](#)
- PGRSyncMessage_STOPPED, [45](#)
- PGRSyncMessage_SYNCING, [45](#)
- PGRSyncMessage_THREAD_ERROR, [45](#)
- PGRSyncMessage_UNKNOWN_ERROR, [46](#)
- MultiSyncLibrary.h, [281](#)
- MultiSyncLibrary::SyncManager
 - ~SyncManager, [251](#)
 - DisableCrossPCSynchronization, [251](#)
 - EnableCrossPCSynchronization, [251](#)
 - GetSyncStatus, [251](#)
 - GetTimeSinceSynced, [251](#)
 - IsTimingBusConnected, [251](#)
 - QueryCrossPCSynchronizationSetting, [251](#)
 - RescanMasterTimingBus, [251](#)
 - Start, [251](#)
 - Stop, [251](#)
 - SyncManager, [251](#)
- MultiSyncLibraryDefs.h, [281](#)
- MultiSyncLibraryPlatform.h, [282](#)
 - MULTISYNCLIBRARY_API, [282](#)
 - MULTISYNCLIBRARY_LOCAL, [282](#)
- NEAREST_NEIGHBOR
 - Enumerations, [20](#)
- NO_COLOR_PROCESSING
 - Enumerations, [20](#)
- NODE
 - FlyCapture2::TopologyNode, [258](#)
- NONE
 - Enumerations, [19](#)
 - FlyCapture2::TIFFOption, [254](#)
- NOT_CONNECTED
 - FlyCapture2::TopologyNode, [258](#)
- NULL
 - FlyCapture2Defs.h, [278](#)
- NUM_FRAMERATES
 - Enumerations, [22](#)
- NUM_MODES
 - Enumerations, [24](#)
- NUM_PIXEL_FORMATS
 - Enumerations, [25](#)
- NUM_STATISTICS_CHANNELS
 - FlyCapture2::ImageStatistics, [225](#)
- NUM_VIDEOMODES
 - Enumerations, [27](#)
- networkInterfaceIndex
 - FlyCapture2::GigEStreamChannel, [206](#)
- NodeMap, [237](#)
 - FlyCapture2::NodeMap, [238](#)
- NodeMap.h, [282](#)
- nodeNumber
 - FlyCapture2::CameraInfo, [118](#)
- NodeType
 - FlyCapture2::TopologyNode, [258](#)
- nodeVendorId
 - FlyCapture2::ConfigROM, [124](#)
- numBanks
 - FlyCapture2::LUTData, [234](#)
- numBuffers
 - FlyCapture2::FC2Config, [135](#)
- numChannels
 - FlyCapture2::LUTData, [234](#)
- numCpuCores
 - FlyCapture2::SystemInfo, [253](#)
- numCurrents
 - FlyCapture2::CameraStats, [122](#)
- numEntries
 - FlyCapture2::LUTData, [235](#)
- numImageNotifications
 - FlyCapture2::FC2Config, [135](#)
- numResendPacketsReceived
 - FlyCapture2::CameraStats, [122](#)
- numResendPacketsRequested
 - FlyCapture2::CameraStats, [122](#)
- numVoltages
 - FlyCapture2::CameraStats, [122](#)
- OSTYPE_FORCE_32BITS
 - FlyCapture2, [44](#)
- OSType
 - FlyCapture2, [44](#)
- octets
 - FlyCapture2::IPAddress, [231](#)
 - FlyCapture2::MACAddress, [236](#)
- offsetHStepSize
 - FlyCapture2::Format7Info, [142](#)
 - FlyCapture2::GigEImageSettingsInfo, [202](#)
- offsetVStepSize
 - FlyCapture2::Format7Info, [142](#)
 - FlyCapture2::GigEImageSettingsInfo, [203](#)
- offsetX
 - FlyCapture2::Format7ImageSettings, [139](#)
 - FlyCapture2::GigEImageSettings, [201](#)
- offsetY
 - FlyCapture2::Format7ImageSettings, [140](#)
 - FlyCapture2::GigEImageSettings, [201](#)
- onOff
 - FlyCapture2::EmbeddedImageInfoProperty, [127](#)
 - FlyCapture2::Property, [244](#)
 - FlyCapture2::StrobeControl, [248](#)
 - FlyCapture2::TriggerMode, [262](#)
- onOffSupported
 - FlyCapture2::PropertyInfo, [246](#)
 - FlyCapture2::StrobeInfo, [250](#)
 - FlyCapture2::TriggerModeInfo, [264](#)
- onePush
 - FlyCapture2::Property, [244](#)
- onePushSupported
 - FlyCapture2::PropertyInfo, [246](#)

- operator!=
 - FlyCapture2::Error, [130](#)
 - FlyCapture2::IPAddress, [231](#)
 - FlyCapture2::MACAddress, [236](#)
 - FlyCapture2::PGRGuid, [240](#)
- operator()
 - FlyCapture2::Image, [216](#)
- operator=
 - FlyCapture2::Error, [130](#)
 - FlyCapture2::Image, [217](#)
 - FlyCapture2::ImageStatistics, [229](#)
 - FlyCapture2::TopologyNode, [261](#)
- operator==
 - FlyCapture2::Error, [130](#)
 - FlyCapture2::IPAddress, [231](#)
 - FlyCapture2::MACAddress, [236](#)
 - FlyCapture2::PGRGuid, [240](#)
- operator[]
 - FlyCapture2::Image, [217](#)
- osDescription
 - FlyCapture2::SystemInfo, [253](#)
- osType
 - FlyCapture2::SystemInfo, [253](#)
- outputBitDepth
 - FlyCapture2::LUTData, [235](#)
- PACKBITS
 - FlyCapture2::TIFFOption, [254](#)
- PACKET_DELAY
 - GigE specific enumerations, [28](#)
- PACKET_SIZE
 - GigE specific enumerations, [28](#)
- PAN
 - Enumerations, [26](#)
- PCIE_BUSSPEED_2_5
 - Enumerations, [25](#)
- PCIE_BUSSPEED_5_0
 - Enumerations, [25](#)
- PCIE_BUSSPEED_FORCE_32BITS
 - Enumerations, [25](#)
- PCIE_BUSSPEED_UNKNOWN
 - Enumerations, [25](#)
- PCleBusSpeed
 - Enumerations, [24](#)
- PGMOption, [238](#)
 - FlyCapture2::PGMOption, [239](#)
- PGRERROR_BUFFER_TOO_SMALL
 - Enumerations, [22](#)
- PGRERROR_BUS_MASTER_FAILED
 - Enumerations, [22](#)
- PGRERROR_FAILED_BUS_MASTER_CONNECTION
 - Enumerations, [21](#)
- PGRERROR_FAILED_GUID
 - Enumerations, [21](#)
- PGRERROR_FAILED
 - Enumerations, [21](#)
- PGRERROR_FORCE_32BITS
 - Enumerations, [22](#)
- PGRERROR_IIDC_FAILED
 - Enumerations, [22](#)
- 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, [22](#)
- 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, [22](#)
- PGRERROR_MEMORY_ALLOCATION_FAILED
 - Enumerations, [21](#)
- PGRERROR_NOT_CONNECTED
 - Enumerations, [21](#)
- PGRERROR_NOT_FOUND
 - Enumerations, [21](#)
- PGRERROR_NOT_IMPLEMENTED
 - Enumerations, [21](#)
- PGRERROR_NOT_IN_FORMAT7
 - Enumerations, [21](#)
- PGRERROR_NOT_INITIALIZED
 - 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](#)
- PGRGuid, [239](#)
 - FlyCapture2::PGRGuid, [240](#)
- PGRSyncError
 - MultiSyncLibrary, [45](#)
- PGRSyncError_ALREADY_STARTED
 - MultiSyncLibrary, [45](#)
- PGRSyncError_ALREADY_STOPPED
 - MultiSyncLibrary, [45](#)
- PGRSyncError_CAMERA_NOT_FOUND
 - MultiSyncLibrary, [45](#)
- PGRSyncError_FAILED
 - MultiSyncLibrary, [45](#)
- PGRSyncError_OK
 - MultiSyncLibrary, [45](#)
- PGRSyncError_UNKNOWN_ERROR
 - MultiSyncLibrary, [45](#)
- PGRSyncMessage
 - MultiSyncLibrary, [45](#)
- PGRSyncMessage_BUS_RESET
 - MultiSyncLibrary, [46](#)
- PGRSyncMessage_DEVICE_ERROR
 - MultiSyncLibrary, [46](#)
- PGRSyncMessage_NOMASTER
 - MultiSyncLibrary, [45](#)
- PGRSyncMessage_NOT_ENOUGH_DEVICES
 - MultiSyncLibrary, [46](#)
- PGRSyncMessage_NOT_INITIALIZED
 - MultiSyncLibrary, [46](#)
- PGRSyncMessage_OK
 - MultiSyncLibrary, [45](#)
- PGRSyncMessage_STARTED
 - MultiSyncLibrary, [45](#)
- PGRSyncMessage_STOPPED
 - MultiSyncLibrary, [45](#)
- PGRSyncMessage_SYNCING
 - MultiSyncLibrary, [45](#)
- PGRSyncMessage_THREAD_ERROR
 - MultiSyncLibrary, [45](#)
- PGRSyncMessage_UNKNOWN_ERROR
 - MultiSyncLibrary, [46](#)
- MultiSyncLibrary, [46](#)
- PGM
 - Enumerations, [23](#)
- PIXEL_FORMAT_411YUV8
 - Enumerations, [25](#)
- PIXEL_FORMAT_422YUV8
 - Enumerations, [25](#)
- PIXEL_FORMAT_422YUV8_JPEG
 - Enumerations, [25](#)
- PIXEL_FORMAT_444YUV8
 - Enumerations, [25](#)
- PIXEL_FORMAT_BGR16
 - Enumerations, [25](#)
- PIXEL_FORMAT_BGRU16
 - Enumerations, [25](#)
- PIXEL_FORMAT_BGRU
 - Enumerations, [25](#)
- PIXEL_FORMAT_BGR
 - Enumerations, [25](#)
- PIXEL_FORMAT_MONO12
 - Enumerations, [25](#)
- PIXEL_FORMAT_MONO16
 - Enumerations, [25](#)
- PIXEL_FORMAT_MONO8
 - Enumerations, [25](#)
- PIXEL_FORMAT_RAW12
 - Enumerations, [25](#)
- PIXEL_FORMAT_RAW16
 - Enumerations, [25](#)
- PIXEL_FORMAT_RAW8
 - Enumerations, [25](#)
- PIXEL_FORMAT_RGB16
 - Enumerations, [25](#)
- PIXEL_FORMAT_RGB8
 - Enumerations, [25](#)
- PIXEL_FORMAT_RGBU
 - Enumerations, [25](#)
- PIXEL_FORMAT_RGB
 - Enumerations, [25](#)
- PIXEL_FORMAT_S_MONO16
 - Enumerations, [25](#)
- PIXEL_FORMAT_S_RGB16
 - Enumerations, [25](#)
- PNGOption, [240](#)
 - FlyCapture2::PNGOption, [241](#)
- PNG
 - Enumerations, [23](#)
- PPMOption, [241](#)
 - FlyCapture2::PPMOption, [242](#)
- PPM
 - Enumerations, [23](#)
- PROPERTY_TYPE_FORCE_32BITS
 - Enumerations, [26](#)
- pUnitAbbr
 - FlyCapture2::PropertyInfo, [247](#)
- pUnits
 - FlyCapture2::PropertyInfo, [247](#)
- packetSize

- FlyCapture2::Format7Info, 142
 - FlyCapture2::GigEStreamChannel, 206
- parameter
 - FlyCapture2::TriggerMode, 262
- pcieBusSpeed
 - FlyCapture2::CameraInfo, 118
- percentage
 - FlyCapture2::Format7Info, 142
- PixelFormat
 - Enumerations, 25
- pixelFormat
 - FlyCapture2::Format7ImageSettings, 140
 - FlyCapture2::GigEImageSettings, 201
- pixelFormatBitField
 - FlyCapture2::Format7Info, 142
 - FlyCapture2::GigEImageSettingsInfo, 203
- polarity
 - FlyCapture2::StrobeControl, 248
 - FlyCapture2::TriggerMode, 263
- polaritySupported
 - FlyCapture2::StrobeInfo, 250
 - FlyCapture2::TriggerModelInfo, 264
- portErrors
 - FlyCapture2::CameraStats, 122
- PortType
 - FlyCapture2::TopologyNode, 258
- present
 - FlyCapture2::Property, 244
 - FlyCapture2::PropertyInfo, 247
 - FlyCapture2::StrobeInfo, 250
 - FlyCapture2::TriggerModelInfo, 264
- PrintErrorTrace
 - FlyCapture2::Error, 131
- progressive
 - FlyCapture2::JPEGOption, 232
- propType
 - FlyCapture2::GigEProperty, 204
- Property, 242
 - FlyCapture2::Property, 243
- PropertyInfo, 245
 - FlyCapture2::PropertyInfo, 246
- PropertyType
 - Enumerations, 25
- pszKeyword
 - FlyCapture2::ConfigROM, 124
- quality
 - FlyCapture2::JPEGOption, 232
 - FlyCapture2::JPG2Option, 233
 - FlyCapture2::MJPGOption, 237
- QueryCrossPCSSynchronizationSetting
 - MultiSyncLibrary::SyncManager, 251
- QueryGigEImagingMode
 - FlyCapture2::GigECamera, 184
- RAW
 - Enumerations, 23
- REMOVAL
 - Enumerations, 20
- RED
 - FlyCapture2::ImageStatistics, 225
- RGGB
 - Enumerations, 19
- RIGOROUS
 - Enumerations, 20
- ROIPosition
 - FlyCapture2::EmbeddedImageInfo, 126
- ReadGVCPMemory
 - FlyCapture2::GCCamera, 156
 - FlyCapture2::GigECamera, 184
- ReadGVCPRegister
 - FlyCapture2::GCCamera, 156
 - FlyCapture2::GigECamera, 185
- ReadGVCPRegisterBlock
 - FlyCapture2::GCCamera, 156
 - FlyCapture2::GigECamera, 185
- readOutSupported
 - FlyCapture2::PropertyInfo, 247
 - FlyCapture2::StrobeInfo, 250
 - FlyCapture2::TriggerModelInfo, 264
- ReadPhyRegister
 - FlyCapture2::BusManager, 58
- ReadRegister
 - FlyCapture2::Camera, 75
 - FlyCapture2::CameraBase, 102
 - FlyCapture2::GCCamera, 156
 - FlyCapture2::GigECamera, 185
- ReadRegisterBlock
 - FlyCapture2::Camera, 76
 - FlyCapture2::CameraBase, 102
 - FlyCapture2::GCCamera, 157
 - FlyCapture2::GigECamera, 186
- recommendedBytesPerPacket
 - FlyCapture2::Format7PacketInfo, 143
- regReadFailed
 - FlyCapture2::CameraStats, 123
- regWriteFailed
 - FlyCapture2::CameraStats, 123
- RegisterAllEvents
 - FlyCapture2::Camera, 76
 - FlyCapture2::CameraBase, 103
 - FlyCapture2::GigECamera, 186
- RegisterCallback
 - FlyCapture2::BusManager, 59
- RegisterEvent
 - FlyCapture2::Camera, 76
 - FlyCapture2::CameraBase, 103
 - FlyCapture2::GigECamera, 186
- registerTimeout
 - FlyCapture2::FC2Config, 136
 - FlyCapture2::GigEConfig, 199
- registerTimeoutRetries
 - FlyCapture2::FC2Config, 136
 - FlyCapture2::GigEConfig, 200
- ReleaseBuffer
 - FlyCapture2::Image, 217
- RescanBus

- FlyCapture2::BusManager, 59
- RescanMasterTimingBus
 - MultiSyncLibrary::SyncManager, 251
- reserved
 - FlyCapture2::AVIOption, 47
 - FlyCapture2::BMPOption, 51
 - FlyCapture2::CameraInfo, 119
 - FlyCapture2::CameraStats, 123
 - FlyCapture2::ConfigROM, 124
 - FlyCapture2::FC2Config, 136
 - FlyCapture2::Format7ImageSettings, 140
 - FlyCapture2::Format7Info, 142
 - FlyCapture2::Format7PacketInfo, 144
 - FlyCapture2::GigEImageSettings, 201
 - FlyCapture2::GigEImageSettingsInfo, 203
 - FlyCapture2::H264Option, 207
 - FlyCapture2::ImageMetadata, 224
 - FlyCapture2::JPEGOption, 232
 - FlyCapture2::JPG2Option, 233
 - FlyCapture2::LUTData, 235
 - FlyCapture2::MJPGOption, 237
 - FlyCapture2::PGMOption, 239
 - FlyCapture2::PNGOption, 241
 - FlyCapture2::PPMOption, 242
 - FlyCapture2::Property, 244
 - FlyCapture2::PropertyInfo, 247
 - FlyCapture2::StrobeControl, 248
 - FlyCapture2::StrobeInfo, 250
 - FlyCapture2::SystemInfo, 253
 - FlyCapture2::TIFFOption, 255
 - FlyCapture2::TimeStamp, 256
 - FlyCapture2::TriggerMode, 263
 - FlyCapture2::TriggerModelInfo, 264
- ResetStats
 - FlyCapture2::Camera, 76
 - FlyCapture2::CameraBase, 103
 - FlyCapture2::GCCamera, 157
 - FlyCapture2::GigECamera, 186
- RestoreFromMemoryChannel
 - FlyCapture2::Camera, 76
 - FlyCapture2::CameraBase, 103
 - FlyCapture2::GCCamera, 157
 - FlyCapture2::GigECamera, 186
- RetrieveBuffer
 - FlyCapture2::Camera, 77
 - FlyCapture2::CameraBase, 103
 - FlyCapture2::GCCamera, 158
 - FlyCapture2::GigECamera, 187
- SATURATION
 - Enumerations, 26
 - FlyCapture2::ImageStatistics, 225
- SHARPNESS
 - Enumerations, 26
- SHUTTER
 - Enumerations, 26
- Save
 - FlyCapture2::Image, 217–219
- SaveToMemoryChannel
 - FlyCapture2::Camera, 77
 - FlyCapture2::CameraBase, 104
 - FlyCapture2::GCCamera, 158
 - FlyCapture2::GigECamera, 187
- screenHeight
 - FlyCapture2::SystemInfo, 253
- screenWidth
 - FlyCapture2::SystemInfo, 253
- seconds
 - FlyCapture2::TimeStamp, 256
- sensorInfo
 - FlyCapture2::CameraInfo, 119
- sensorResolution
 - FlyCapture2::CameraInfo, 119
- serialNumber
 - FlyCapture2::CameraInfo, 119
- SetActiveLUTBank
 - FlyCapture2::Camera, 78
 - FlyCapture2::CameraBase, 104
 - FlyCapture2::GCCamera, 159
 - FlyCapture2::GigECamera, 188
- SetBlockId
 - FlyCapture2::Image, 220
- SetCallback
 - FlyCapture2::Camera, 78
 - FlyCapture2::CameraBase, 104
 - FlyCapture2::GCCamera, 159
 - FlyCapture2::GigECamera, 188
- SetCamera
 - FlyCapture2::GCCamera, 159, 160
- SetChannelStatus
 - FlyCapture2::ImageStatistics, 229
- SetColorProcessing
 - FlyCapture2::Image, 220
- SetConfiguration
 - FlyCapture2::Camera, 78
 - FlyCapture2::CameraBase, 105
 - FlyCapture2::GCCamera, 160
 - FlyCapture2::GigECamera, 188
- SetData
 - FlyCapture2::Image, 220
- SetDefaultColorProcessing
 - FlyCapture2::Image, 221
- SetDefaultOutputFormat
 - FlyCapture2::Image, 221
- SetDimensions
 - FlyCapture2::Image, 221
- SetEmbeddedImageInfo
 - FlyCapture2::Camera, 80
 - FlyCapture2::CameraBase, 105
 - FlyCapture2::GCCamera, 160
 - FlyCapture2::GigECamera, 190
- SetFormat7Configuration
 - FlyCapture2::Camera, 80, 81
- SetGPIOPinDirection
 - FlyCapture2::Camera, 81
 - FlyCapture2::CameraBase, 106
 - FlyCapture2::GCCamera, 160

- FlyCapture2::GigECamera, [192](#)
- SetGigEConfig
 - FlyCapture2::GigECamera, [190](#)
- SetGigEImageBinningSettings
 - FlyCapture2::GigECamera, [190](#)
- SetGigEImageSettings
 - FlyCapture2::GigECamera, [191](#)
- SetGigEImagingMode
 - FlyCapture2::GigECamera, [191](#)
- SetGigEProperty
 - FlyCapture2::GigECamera, [191](#)
- SetGigEStreamChannelInfo
 - FlyCapture2::GigECamera, [192](#)
- SetLUTChannel
 - FlyCapture2::Camera, [81](#)
 - FlyCapture2::CameraBase, [106](#)
 - FlyCapture2::GCCamera, [161](#)
 - FlyCapture2::GigECamera, [192](#)
- SetProperty
 - FlyCapture2::Camera, [82](#)
 - FlyCapture2::CameraBase, [107](#)
 - FlyCapture2::GCCamera, [161](#)
 - FlyCapture2::GigECamera, [193](#)
- SetStrobe
 - FlyCapture2::Camera, [82](#)
 - FlyCapture2::CameraBase, [107](#)
 - FlyCapture2::GCCamera, [162](#)
 - FlyCapture2::GigECamera, [193](#)
- SetTitle
 - FlyCapture2::CameraControlDlg, [114](#)
 - FlyCapture2::CameraSelectionDlg, [120](#)
- SetTriggerDelay
 - FlyCapture2::Camera, [83](#)
 - FlyCapture2::CameraBase, [108](#)
 - FlyCapture2::GCCamera, [162](#)
 - FlyCapture2::GigECamera, [194](#)
- SetTriggerMode
 - FlyCapture2::Camera, [83](#)
 - FlyCapture2::CameraBase, [108](#)
 - FlyCapture2::GCCamera, [163](#)
 - FlyCapture2::GigECamera, [194](#)
- SetUserBuffers
 - FlyCapture2::Camera, [84](#)
 - FlyCapture2::CameraBase, [109](#)
 - FlyCapture2::GCCamera, [163](#)
 - FlyCapture2::GigECamera, [195](#)
- SetVideoModeAndFrameRate
 - FlyCapture2::Camera, [84](#)
- Show
 - FlyCapture2::CameraControlDlg, [114](#)
- ShowCameraSelectionDialog
 - FlyCap3CameraControl::FlyCapture3ApiGui↔
Wrapper, [138](#)
- ShowDialogByIndex
 - FlyCap3CameraControl::FlyCapture3ApiGui↔
Wrapper, [138](#)
- ShowDialogByName
 - FlyCap3CameraControl::FlyCapture3ApiGui↔
Wrapper, [138](#)
- ShowModal
 - FlyCapture2::CameraControlDlg, [114](#)
 - FlyCapture2::CameraSelectionDlg, [120](#)
- ShowPropertyGridDialog
 - FlyCap3CameraControl::FlyCapture3ApiGui↔
Wrapper, [138](#)
- shutter
 - FlyCapture2::EmbeddedImageInfo, [126](#)
- sk_maxNumPorts
 - Global constants, [13](#)
- sk_maxStringLength
 - Global constants, [13](#)
- softwareTriggerSupported
 - FlyCapture2::TriggerModelInfo, [264](#)
- source
 - FlyCapture2::StrobeControl, [248](#)
 - FlyCapture2::StrobeInfo, [250](#)
 - FlyCapture2::TriggerMode, [263](#)
- sourceMask
 - FlyCapture2::TriggerModelInfo, [264](#)
- sourcePort
 - FlyCapture2::GigEStreamChannel, [206](#)
- Start
 - MultiSyncLibrary::SyncManager, [251](#)
- StartCapture
 - FlyCapture2::Camera, [85](#)
 - FlyCapture2::CameraBase, [109](#)
 - FlyCapture2::GCCamera, [164](#)
 - FlyCapture2::GigECamera, [195](#)
- StartSyncCapture
 - FlyCapture2::Camera, [85](#)
 - FlyCapture2::CameraBase, [110](#)
 - FlyCapture2::GCCamera, [164](#)
 - FlyCapture2::GigECamera, [196](#)
- StatisticsChannel
 - FlyCapture2::ImageStatistics, [225](#)
- Stop
 - MultiSyncLibrary::SyncManager, [251](#)
- StopCapture
 - FlyCapture2::Camera, [85](#)
 - FlyCapture2::CameraBase, [110](#)
 - FlyCapture2::GCCamera, [164](#)
 - FlyCapture2::GigECamera, [196](#)
- StrobeControl, [247](#)
 - FlyCapture2::StrobeControl, [248](#)
- StrobeInfo, [249](#)
 - FlyCapture2::StrobeInfo, [249](#)
- strobePattern
 - FlyCapture2::EmbeddedImageInfo, [126](#)
- Structures, [29](#)
 - TriggerDelay, [30](#)
 - TriggerDelayInfo, [30](#)
- subnetMask
 - FlyCapture2::CameraInfo, [119](#)
- supported
 - FlyCapture2::LUTData, [235](#)

- SyncManager, [250](#)
 - MultiSyncLibrary::SyncManager, [251](#)
- sysMemSize
 - FlyCapture2::SystemInfo, [253](#)
- SystemInfo, [251](#)
- TEMPERATURE
 - Enumerations, [26](#)
- TIFFOption, [254](#)
 - FlyCapture2::TIFFOption, [255](#)
- TIFF
 - Enumerations, [23](#)
- TILT
 - Enumerations, [26](#)
- TIMEOUT_INFINITE
 - Enumerations, [23](#)
- TIMEOUT_NONE
 - Enumerations, [23](#)
- TIMEOUT_UNSPECIFIED
 - Enumerations, [23](#)
- TRIGGER_DELAY
 - Enumerations, [26](#)
- TRIGGER_MODE
 - Enumerations, [26](#)
- temperature
 - FlyCapture2::CameraStats, [123](#)
- TestGainNode
 - FlyCapture2::GCCamera, [165](#)
- timeSinceBusReset
 - FlyCapture2::CameraStats, [123](#)
- timeSinceInitialization
 - FlyCapture2::CameraStats, [123](#)
- TimeStamp, [255](#)
 - FlyCapture2::TimeStamp, [256](#)
- timeStamp
 - FlyCapture2::CameraStats, [123](#)
- timestamp
 - FlyCapture2::EmbeddedImageInfo, [126](#)
- TopologyNode, [256](#)
 - FlyCapture2::TopologyNode, [258](#), [259](#)
- TopologyNode.h, [282](#)
- TriggerDelay
 - Structures, [30](#)
- TriggerDelayInfo
 - Structures, [30](#)
- TriggerMode, [262](#)
 - FlyCapture2::TriggerMode, [262](#)
- TriggerModelInfo, [263](#)
 - FlyCapture2::TriggerModelInfo, [264](#)
- type
 - FlyCapture2::FC2Version, [137](#)
 - FlyCapture2::Property, [244](#)
 - FlyCapture2::PropertyInfo, [247](#)
- UNKNOWN_OS
 - FlyCapture2, [44](#)
- UNSPECIFIED_GRAB_MODE
 - Enumerations, [23](#)
- UNSPECIFIED_PIXEL_FORMAT
 - Enumerations, [25](#)
- UNSPECIFIED_PROPERTY_TYPE
 - Enumerations, [26](#)
- unitBytesPerPacket
 - FlyCapture2::Format7PacketInfo, [144](#)
- unitSWVer
 - FlyCapture2::ConfigROM, [125](#)
- unitSpecId
 - FlyCapture2::ConfigROM, [124](#)
- unitSubSWVer
 - FlyCapture2::ConfigROM, [125](#)
- UnregisterCallback
 - FlyCapture2::BusManager, [59](#)
- userDefinedName
 - FlyCapture2::CameraInfo, [119](#)
- Utilities, [265](#)
- Utilities.h, [283](#)
- VIDEOMODE_1024x768RGB
 - Enumerations, [26](#)
- VIDEOMODE_1024x768Y16
 - Enumerations, [27](#)
- VIDEOMODE_1024x768Y8
 - Enumerations, [27](#)
- VIDEOMODE_1024x768YUV422
 - Enumerations, [26](#)
- VIDEOMODE_1280x960RGB
 - Enumerations, [27](#)
- VIDEOMODE_1280x960Y16
 - Enumerations, [27](#)
- VIDEOMODE_1280x960Y8
 - Enumerations, [27](#)
- VIDEOMODE_1280x960YUV422
 - Enumerations, [27](#)
- VIDEOMODE_1600x1200RGB
 - Enumerations, [27](#)
- VIDEOMODE_1600x1200Y16
 - Enumerations, [27](#)
- VIDEOMODE_1600x1200Y8
 - Enumerations, [27](#)
- VIDEOMODE_1600x1200YUV422
 - Enumerations, [27](#)
- VIDEOMODE_160x120YUV444
 - Enumerations, [26](#)
- VIDEOMODE_320x240YUV422
 - Enumerations, [26](#)
- VIDEOMODE_640x480RGB
 - Enumerations, [26](#)
- VIDEOMODE_640x480Y16
 - Enumerations, [26](#)
- VIDEOMODE_640x480Y8
 - Enumerations, [26](#)
- VIDEOMODE_640x480YUV411
 - Enumerations, [26](#)
- VIDEOMODE_640x480YUV422
 - Enumerations, [26](#)
- VIDEOMODE_800x600RGB
 - Enumerations, [26](#)
- VIDEOMODE_800x600Y16
 - Enumerations, [26](#)

- Enumerations, [26](#)
- VIDEOMODE_800x600Y8
 - Enumerations, [26](#)
- VIDEOMODE_800x600YUV422
 - Enumerations, [26](#)
- VIDEOMODE_FORCE_32BITS
 - Enumerations, [27](#)
- VIDEOMODE_FORMAT7
 - Enumerations, [27](#)
- ValidateFormat7Settings
 - FlyCapture2::Camera, [86](#)
- value
 - FlyCapture2::GigEProperty, [204](#)
 - FlyCapture2::PGRGuid, [240](#)
- valueReadable
 - FlyCapture2::TriggerModelInfo, [264](#)
- valueA
 - FlyCapture2::Property, [244](#)
- valueB
 - FlyCapture2::Property, [244](#)
- vendorName
 - FlyCapture2::CameraInfo, [119](#)
- vendorPixelFormatBitField
 - FlyCapture2::Format7Info, [143](#)
 - FlyCapture2::GigEImageSettingsInfo, [203](#)
- vendorUniqueInfo_0
 - FlyCapture2::ConfigROM, [125](#)
- vendorUniqueInfo_1
 - FlyCapture2::ConfigROM, [125](#)
- vendorUniqueInfo_2
 - FlyCapture2::ConfigROM, [125](#)
- vendorUniqueInfo_3
 - FlyCapture2::ConfigROM, [125](#)
- VideoMode
 - Enumerations, [26](#)
- WHITE_BALANCE
 - Enumerations, [26](#)
- WINDOWS_X64
 - FlyCapture2, [44](#)
- WINDOWS_X86
 - FlyCapture2, [44](#)
- WRAPPER_API
 - FlyCapture3ApiGuiWrapper.h, [279](#)
- WaitForBufferEvent
 - FlyCapture2::Camera, [86](#)
 - FlyCapture2::CameraBase, [111](#)
 - FlyCapture2::GCCamera, [165](#)
 - FlyCapture2::GigECamera, [196](#)
- whiteBalance
 - FlyCapture2::EmbeddedImageInfo, [126](#)
- width
 - FlyCapture2::Format7ImageSettings, [140](#)
 - FlyCapture2::GigEImageSettings, [201](#)
 - FlyCapture2::H264Option, [207](#)
- WriteGVCPMemory
 - FlyCapture2::GCCamera, [165](#)
 - FlyCapture2::GigECamera, [197](#)
- WriteGVCPRegister
 - FlyCapture2::GCCamera, [165](#)
 - FlyCapture2::GigECamera, [197](#)
- WriteGVCPRegisterBlock
 - FlyCapture2::GCCamera, [165](#)
 - FlyCapture2::GigECamera, [197](#)
- WritePhyRegister
 - FlyCapture2::BusManager, [60](#)
- WriteRegister
 - FlyCapture2::Camera, [87](#)
 - FlyCapture2::CameraBase, [111](#)
 - FlyCapture2::GCCamera, [165](#)
 - FlyCapture2::GigECamera, [198](#)
- WriteRegisterBlock
 - FlyCapture2::Camera, [87](#)
 - FlyCapture2::CameraBase, [112](#)
 - FlyCapture2::GCCamera, [166](#)
 - FlyCapture2::GigECamera, [198](#)
- xmlURL1
 - FlyCapture2::CameraInfo, [119](#)
- xmlURL2
 - FlyCapture2::CameraInfo, [119](#)
- ZOOM
 - Enumerations, [26](#)