

FlyCapture2 C

2.10.3.0

Generated by Doxygen 1.8.11

Contents

1	Deprecated List	1
2	Module Index	3
2.1	Modules	3
3	Data Structure Index	5
3.1	Data Structures	5
4	File Index	7
4.1	File List	7
5	Module Documentation	9
5.1	Bus Manager Operation	9
5.1.1	Detailed Description	10
5.1.2	Function Documentation	10
5.1.2.1	fc2DiscoverGigECameras(fc2Context context, fc2CameraInfo *gigECameras, unsigned int *arraySize)	10
5.1.2.2	fc2FireBusReset(fc2Context context, fc2PGRGuid *pGuid)	11
5.1.2.3	fc2ForceAllIPAddressesAutomatically()	11
5.1.2.4	fc2ForceIPAddressAutomatically(unsigned int serialNumber)	11
5.1.2.5	fc2ForceIPAddressToCamera(fc2Context context, fc2MACAddress macAddress, fc2IPAddress ipAddress, fc2IPAddress subnetMask, fc2IPAddress defaultGateway)	12
5.1.2.6	fc2GetCameraFromIndex(fc2Context context, unsigned int index, fc2PGRGuid *pGuid)	12
5.1.2.7	fc2GetCameraFromIPAddress(fc2Context context, fc2IPAddress ipAddress, fc2PGRGuid *pGuid)	12
5.1.2.8	fc2GetCameraFromSerialNumber(fc2Context context, unsigned int serialNumber, fc2PGRGuid *pGuid)	13

5.1.2.9	fc2GetCameraSerialNumberFromIndex(fc2Context context, unsigned int index, unsigned int *pSerialNumber)	13
5.1.2.10	fc2GetDeviceFromIndex(fc2Context context, unsigned int index, fc2PGRGuid *pGuid)	13
5.1.2.11	fc2GetInterfaceTypeFromGuid(fc2Context context, fc2PGRGuid *pGuid, fc2InterfaceType *pInterfaceType)	14
5.1.2.12	fc2GetNumOfCameras(fc2Context context, unsigned int *pNumCameras)	14
5.1.2.13	fc2GetNumOfDevices(fc2Context context, unsigned int *pNumDevices)	14
5.1.2.14	fc2GetTopology(fc2Context context, fc2TopologyNodeContext *pTopologyNodeContext)	15
5.1.2.15	fc2GetUsbLinkInfo(fc2Context context, fc2PGRGuid guid, unsigned int *pValue)	15
5.1.2.16	fc2GetUsbPortStatus(fc2Context context, fc2PGRGuid guid, unsigned int *pValue)	15
5.1.2.17	fc2IsCameraControlable(fc2Context context, fc2PGRGuid *pGuid, BOOL *pControlable)	16
5.1.2.18	fc2ReadPhyRegister(fc2Context context, fc2PGRGuid guid, unsigned int page, unsigned int port, unsigned int address, unsigned int *pValue)	16
5.1.2.19	fc2RegisterCallback(fc2Context context, fc2BusEventCallback enumCallback, fc2BusCallbackType callbackType, void *pParameter, fc2CallbackHandle *pCallbackHandle)	16
5.1.2.20	fc2RescanBus(fc2Context context)	17
5.1.2.21	fc2UnregisterCallback(fc2Context context, fc2CallbackHandle callbackHandle)	17
5.1.2.22	fc2WritePhyRegister(fc2Context context, fc2PGRGuid guid, unsigned int page, unsigned int port, unsigned int address, unsigned int value)	17
5.2	Connection and Image Retrieval	19
5.2.1	Detailed Description	19
5.2.2	Function Documentation	19
5.2.2.1	fc2Connect(fc2Context context, fc2PGRGuid *guid)	19
5.2.2.2	fc2Disconnect(fc2Context context)	20
5.2.2.3	fc2GetConfiguration(fc2Context context, fc2Config *config)	20
5.2.2.4	fc2RetrieveBuffer(fc2Context context, fc2Image *pImage)	20
5.2.2.5	fc2SetCallback(fc2Context context, fc2ImageEventCallback pCallbackFn, void *pCallbackData)	21
5.2.2.6	fc2SetConfiguration(fc2Context context, fc2Config *config)	21
5.2.2.7	fc2SetUserBuffers(fc2Context context, unsigned char *const ppMemBuffers, int size, int nNumBuffers)	22

5.2.2.8	<code>fc2StartCapture(fc2Context context)</code>	22
5.2.2.9	<code>fc2StartCaptureCallback(fc2Context context, fc2ImageEventCallback p↔ CallbackFn, void *pCallbackData)</code>	23
5.2.2.10	<code>fc2StartSyncCapture(unsigned int numCameras, fc2Context *pContexts)</code>	23
5.2.2.11	<code>fc2StartSyncCaptureCallback(unsigned int numCameras, fc2Context *pContexts, fc2ImageEventCallback *pCallbackFns, void **pCallbackDataArray)</code>	23
5.2.2.12	<code>fc2StopCapture(fc2Context context)</code>	24
5.2.2.13	<code>fc2WaitForBufferEvent(fc2Context context, fc2Image *pImage, unsigned int eventNumber)</code>	24
5.3	Information and Properties	26
5.3.1	Detailed Description	26
5.3.2	Function Documentation	26
5.3.2.1	<code>fc2GetCameraInfo(fc2Context context, fc2CameraInfo *pCameraInfo)</code>	26
5.3.2.2	<code>fc2GetProperty(fc2Context context, fc2Property *prop)</code>	26
5.3.2.3	<code>fc2GetPropertyInfo(fc2Context context, fc2PropertyInfo *propInfo)</code>	27
5.3.2.4	<code>fc2SetProperty(fc2Context context, fc2Property *prop)</code>	27
5.3.2.5	<code>fc2SetPropertyBroadcast(fc2Context context, fc2Property *prop)</code>	28
5.4	General Purpose Input / Output	29
5.4.1	Detailed Description	29
5.4.2	Function Documentation	29
5.4.2.1	<code>fc2GetGPIOPinDirection(fc2Context context, unsigned int pin, unsigned int *p↔ Direction)</code>	29
5.4.2.2	<code>fc2SetGPIOPinDirection(fc2Context context, unsigned int pin, unsigned int direc- tion)</code>	30
5.4.2.3	<code>fc2SetGPIOPinDirectionBroadcast(fc2Context context, unsigned int pin, unsigned int direction)</code>	30
5.5	Trigger	31
5.5.1	Detailed Description	31
5.5.2	Function Documentation	31
5.5.2.1	<code>fc2FireSoftwareTrigger(fc2Context context)</code>	31
5.5.2.2	<code>fc2FireSoftwareTriggerBroadcast(fc2Context context)</code>	32
5.5.2.3	<code>fc2GetTriggerDelay(fc2Context context, fc2TriggerDelay *triggerDelay)</code>	32
5.5.2.4	<code>fc2GetTriggerDelayInfo(fc2Context context, fc2TriggerDelayInfo *triggerDelayInfo)</code>	32

5.5.2.5	fc2GetTriggerMode(fc2Context context, fc2TriggerMode *triggerMode)	33
5.5.2.6	fc2GetTriggerModelInfo(fc2Context context, fc2TriggerModelInfo *triggerModelInfo)	33
5.5.2.7	fc2SetTriggerDelay(fc2Context context, fc2TriggerDelay *triggerDelay)	34
5.5.2.8	fc2SetTriggerDelayBroadcast(fc2Context context, fc2TriggerDelay *triggerDelay)	34
5.5.2.9	fc2SetTriggerMode(fc2Context context, fc2TriggerMode *triggerMode)	35
5.5.2.10	fc2SetTriggerModeBroadcast(fc2Context context, fc2TriggerMode *triggerMode)	35
5.6	Strobe	37
5.6.1	Detailed Description	37
5.6.2	Function Documentation	37
5.6.2.1	fc2GetStrobe(fc2Context context, fc2StrobeControl *strobeControl)	37
5.6.2.2	fc2GetStrobeInfo(fc2Context context, fc2StrobeInfo *strobeInfo)	37
5.6.2.3	fc2SetStrobe(fc2Context context, fc2StrobeControl *strobeControl)	38
5.6.2.4	fc2SetStrobeBroadcast(fc2Context context, fc2StrobeControl *strobeControl)	38
5.7	Look Up Table	40
5.7.1	Detailed Description	40
5.7.2	Function Documentation	40
5.7.2.1	fc2EnableLUT(fc2Context context, BOOL on)	40
5.7.2.2	fc2GetActiveLUTBank(fc2Context context, unsigned int *pActiveBank)	41
5.7.2.3	fc2GetLUTBankInfo(fc2Context context, unsigned int bank, BOOL *pReadSupported, BOOL *pWriteSupported)	41
5.7.2.4	fc2GetLUTChannel(fc2Context context, unsigned int bank, unsigned int channel, unsigned int sizeEntries, unsigned int *pEntries)	41
5.7.2.5	fc2GetLUTInfo(fc2Context context, fc2LUTData *pData)	42
5.7.2.6	fc2SetActiveLUTBank(fc2Context context, unsigned int activeBank)	42
5.7.2.7	fc2SetLUTChannel(fc2Context context, unsigned int bank, unsigned int channel, unsigned int sizeEntries, unsigned int *pEntries)	43
5.8	Memory Channels	44
5.8.1	Detailed Description	44
5.8.2	Function Documentation	44
5.8.2.1	fc2GetEmbeddedImageInfo(fc2Context context, fc2EmbeddedImageInfo *pInfo)	44
5.8.2.2	fc2GetMemoryChannel(fc2Context context, unsigned int *pCurrentChannel)	45
5.8.2.3	fc2GetMemoryChannelInfo(fc2Context context, unsigned int *pNumChannels)	45

5.8.2.4	fc2RestoreFromMemoryChannel(fc2Context context, unsigned int channel) . . .	45
5.8.2.5	fc2SaveToMemoryChannel(fc2Context context, unsigned int channel)	46
5.8.2.6	fc2SetEmbeddedImageInfo(fc2Context context, fc2EmbeddedImageInfo *pInfo) .	46
5.9	Register Operation	48
5.9.1	Detailed Description	48
5.9.2	Function Documentation	48
5.9.2.1	fc2GetRegisterString(unsigned int registerVal)	48
5.9.2.2	fc2ReadRegister(fc2Context context, unsigned int address, unsigned int *pValue)	48
5.9.2.3	fc2ReadRegisterBlock(fc2Context context, unsigned short addressHigh, unsigned int addressLow, unsigned int *pBuffer, unsigned int length)	49
5.9.2.4	fc2WriteRegister(fc2Context context, unsigned int address, unsigned int value) .	49
5.9.2.5	fc2WriteRegisterBlock(fc2Context context, unsigned short addressHigh, unsigned int addressLow, const unsigned int *pBuffer, unsigned int length)	50
5.9.2.6	fc2WriteRegisterBroadcast(fc2Context context, unsigned int address, unsigned int value)	50
5.10	DCAM Formats	51
5.10.1	Detailed Description	51
5.10.2	Function Documentation	51
5.10.2.1	fc2GetVideoModeAndFrameRate(fc2Context context, fc2VideoMode *videoMode, fc2FrameRate *frameRate)	51
5.10.2.2	fc2GetVideoModeAndFrameRateInfo(fc2Context context, fc2VideoMode videoMode, fc2FrameRate frameRate, BOOL *pSupported)	51
5.10.2.3	fc2SetVideoModeAndFrameRate(fc2Context context, fc2VideoMode videoMode, fc2FrameRate frameRate)	52
5.11	Format7	53
5.11.1	Detailed Description	53
5.11.2	Function Documentation	53
5.11.2.1	fc2GetFormat7Configuration(fc2Context context, fc2Format7ImageSettings *imageSettings, unsigned int *packetSize, float *percentage)	53
5.11.2.2	fc2GetFormat7Info(fc2Context context, fc2Format7Info *info, BOOL *pSupported)	54
5.11.2.3	fc2SetFormat7Configuration(fc2Context context, fc2Format7ImageSettings *imageSettings, float percentSpeed)	54
5.11.2.4	fc2SetFormat7ConfigurationPacket(fc2Context context, fc2Format7ImageSettings *imageSettings, unsigned int packetSize)	54

5.11.2.5	<code>fc2ValidateFormat7Settings(fc2Context context, fc2Format7ImageSettings *imageSettings, BOOL *settingsAreValid, fc2Format7PacketInfo *packetInfo)</code>	55
5.12	GVCP Register Operation	56
5.12.1	Detailed Description	56
5.12.2	Function Documentation	56
5.12.2.1	<code>fc2ReadGVCPMemory(fc2Context context, unsigned int address, unsigned char *pBuffer, unsigned int length)</code>	56
5.12.2.2	<code>fc2ReadGVCPRegister(fc2Context context, unsigned int address, unsigned int *pValue)</code>	57
5.12.2.3	<code>fc2ReadGVCPRegisterBlock(fc2Context context, unsigned int address, unsigned int *pBuffer, unsigned int length)</code>	57
5.12.2.4	<code>fc2WriteGVCPMemory(fc2Context context, unsigned int address, const unsigned char *pBuffer, unsigned int length)</code>	57
5.12.2.5	<code>fc2WriteGVCPRegister(fc2Context context, unsigned int address, unsigned int value)</code>	58
5.12.2.6	<code>fc2WriteGVCPRegisterBlock(fc2Context context, unsigned int address, const unsigned int *pBuffer, unsigned int length)</code>	58
5.12.2.7	<code>fc2WriteGVCPRegisterBroadcast(fc2Context context, unsigned int address, unsigned int value)</code>	58
5.13	GigE property manipulation	60
5.13.1	Detailed Description	60
5.13.2	Function Documentation	60
5.13.2.1	<code>fc2DiscoverGigEPacketSize(fc2Context context, unsigned int *packetSize)</code>	60
5.13.2.2	<code>fc2GetGigEProperty(fc2Context context, fc2GigEProperty *pGigEProp)</code>	60
5.13.2.3	<code>fc2SetGigEProperty(fc2Context context, const fc2GigEProperty *pGigEProp)</code>	61
5.14	GigE image settings	62
5.14.1	Detailed Description	62
5.14.2	Function Documentation	62
5.14.2.1	<code>fc2GetGigEImageSettings(fc2Context context, fc2GigEImageSettings *pImageSettings)</code>	62
5.14.2.2	<code>fc2GetGigEImageSettingsInfo(fc2Context context, fc2GigEImageSettingsInfo *pInfo)</code>	62
5.14.2.3	<code>fc2GetGigEImagingMode(fc2Context context, fc2Mode *mode)</code>	63
5.14.2.4	<code>fc2QueryGigEImagingMode(fc2Context context, fc2Mode mode, BOOL *isSupported)</code>	63

5.14.2.5	<code>fc2SetGigEImageSettings(fc2Context context, const fc2GigEImageSettings *p← ImageSettings)</code>	63
5.14.2.6	<code>fc2SetGigEImagingMode(fc2Context context, fc2Mode mode)</code>	64
5.15	GigE image binning settings	65
5.15.1	Detailed Description	65
5.15.2	Function Documentation	65
5.15.2.1	<code>fc2GetGigEImageBinningSettings(fc2Context context, unsigned int *horz← BinningValue, unsigned int *vertBinningValue)</code>	65
5.15.2.2	<code>fc2SetGigEImageBinningSettings(fc2Context context, unsigned int horz← BinningValue, unsigned int vertBinningValue)</code>	65
5.16	GigE image stream configuration	67
5.16.1	Detailed Description	67
5.16.2	Function Documentation	67
5.16.2.1	<code>fc2GetGigEConfig(fc2Context context, fc2GigEConfig *pConfig)</code>	67
5.16.2.2	<code>fc2GetGigEStreamChannelInfo(fc2Context context, unsigned int channel, fc2← GigEStreamChannel *pChannel)</code>	67
5.16.2.3	<code>fc2GetNumStreamChannels(fc2Context context, unsigned int *numChannels)</code>	68
5.16.2.4	<code>fc2SetGigEConfig(fc2Context context, const fc2GigEConfig *pConfig)</code>	68
5.16.2.5	<code>fc2SetGigEStreamChannelInfo(fc2Context context, unsigned int channel, fc2← GigEStreamChannel *pChannel)</code>	68
5.17	Image Operation	70
5.17.1	Detailed Description	71
5.17.2	Function Documentation	71
5.17.2.1	<code>fc2CalculateImageStatistics(fc2Image *pImage, fc2ImageStatisticsContext *p← ImageStatisticsContext)</code>	71
5.17.2.2	<code>fc2ConvertImage(fc2Image *pImageIn, fc2Image *pImageOut)</code>	71
5.17.2.3	<code>fc2ConvertImageTo(fc2PixelFormat format, fc2Image *pImageIn, fc2Image *p← ImageOut)</code>	71
5.17.2.4	<code>fc2CreateImage(fc2Image *pImage)</code>	72
5.17.2.5	<code>fc2DestroyImage(fc2Image *image)</code>	72
5.17.2.6	<code>fc2DetermineBitsPerPixel(fc2PixelFormat format, unsigned int *pBitsPerPixel)</code>	72
5.17.2.7	<code>fc2GetDefaultColorProcessing(fc2ColorProcessingAlgorithm *pDefaultMethod)</code>	73
5.17.2.8	<code>fc2GetDefaultOutputFormat(fc2PixelFormat *pFormat)</code>	73

5.17.2.9	<code>fc2GetImageData(fc2Image *pImage, unsigned char **ppData)</code>	73
5.17.2.10	<code>fc2GetImageTimeStamp(fc2Image *pImage)</code>	74
5.17.2.11	<code>fc2SaveImage(fc2Image *pImage, const char *pFilename, fc2ImageFileFormat format)</code>	74
5.17.2.12	<code>fc2SaveImageWithOptions(fc2Image *pImage, const char *pFilename, fc2ImageFileFormat format, void *pOption)</code>	74
5.17.2.13	<code>fc2SetDefaultColorProcessing(fc2ColorProcessingAlgorithm defaultMethod)</code>	75
5.17.2.14	<code>fc2SetDefaultOutputFormat(fc2PixelFormat format)</code>	75
5.17.2.15	<code>fc2SetImageData(fc2Image *pImage, const unsigned char *pData, unsigned int dataSize)</code>	75
5.17.2.16	<code>fc2SetImageDimensions(fc2Image *pImage, unsigned int rows, unsigned int cols, unsigned int stride, fc2PixelFormat pixelFormat, fc2BayerTileFormat bayerFormat)</code>	76
5.18	Image Statistics Operation	77
5.18.1	Detailed Description	78
5.18.2	Function Documentation	78
5.18.2.1	<code>fc2CreateImageStatistics(fc2ImageStatisticsContext *pImageStatisticsContext)</code>	78
5.18.2.2	<code>fc2DestroyImageStatistics(fc2ImageStatisticsContext imageStatisticsContext)</code>	78
5.18.2.3	<code>fc2GetChannelHistogram(fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, int **ppHistogram)</code>	78
5.18.2.4	<code>fc2GetChannelMean(fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, float *pPixelValueMean)</code>	79
5.18.2.5	<code>fc2GetChannelNumPixelValues(fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, unsigned int *pNumPixelValues)</code>	79
5.18.2.6	<code>fc2GetChannelPixelValueRange(fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, unsigned int *pPixelValueMin, unsigned int *pPixelValueMax)</code>	79
5.18.2.7	<code>fc2GetChannelRange(fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, unsigned int *pMin, unsigned int *pMax)</code>	80
5.18.2.8	<code>fc2GetChannelStatus(fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, BOOL *pEnabled)</code>	80
5.18.2.9	<code>fc2GetImageStatistics(fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, unsigned int *pRangeMin, unsigned int *pRangeMax, unsigned int *pPixelValueMin, unsigned int *pPixelValueMax, unsigned int *pNumPixelValues, float *pPixelValueMean, int **ppHistogram)</code>	80
5.18.2.10	<code>fc2ImageStatisticsDisableAll(fc2ImageStatisticsContext imageStatisticsContext)</code>	81
5.18.2.11	<code>fc2ImageStatisticsEnableAll(fc2ImageStatisticsContext imageStatisticsContext)</code>	81

5.18.2.12	fc2ImageStatisticsEnableGreyOnly(fc2ImageStatisticsContext imageStatistics↔ Context)	81
5.18.2.13	fc2ImageStatisticsEnableHSLOnly(fc2ImageStatisticsContext imageStatistics↔ Context)	82
5.18.2.14	fc2ImageStatisticsEnableRGBOnly(fc2ImageStatisticsContext imageStatistics↔ Context)	82
5.18.2.15	fc2SetChannelStatus(fc2ImageStatisticsContext imageStatisticsContext, fc2↔ StatisticsChannel channel, BOOL enabled)	82
5.19	AVI Recording Operation	84
5.19.1	Detailed Description	84
5.19.2	Function Documentation	84
5.19.2.1	fc2AVIAppend(fc2AVIContext AVIContext, fc2Image *pImage)	84
5.19.2.2	fc2AVIClose(fc2AVIContext AVIContext)	84
5.19.2.3	fc2AVIOpen(fc2AVIContext AVIContext, const char *pFileName, fc2AVIOption *pOption)	85
5.19.2.4	fc2CreateAVI(fc2AVIContext *pAVIContext)	85
5.19.2.5	fc2DestroyAVI(fc2AVIContext AVIContext)	85
5.19.2.6	fc2H264Open(fc2AVIContext AVIContext, const char *pFileName, fc2H264↔ Option *pOption)	86
5.19.2.7	fc2MJPGOpen(fc2AVIContext AVIContext, const char *pFileName, fc2MJPG↔ Option *pOption)	86
5.20	TopologyNode Operation	87
5.20.1	Detailed Description	88
5.20.2	Function Documentation	88
5.20.2.1	fc2CreateTopologyNode(fc2TopologyNodeContext *pTopologyNodeContext)	88
5.20.2.2	fc2DestroyTopologyNode(fc2TopologyNodeContext TopologyNodeContext)	88
5.20.2.3	fc2TopologyNodeAddChild(fc2TopologyNodeContext TopologyNodeContext, fc2TopologyNodeContext TopologyNodeChildContext)	88
5.20.2.4	fc2TopologyNodeAddPortType(fc2TopologyNodeContext TopologyNodeContext, fc2PortType portType)	89
5.20.2.5	fc2TopologyNodeAssignGuidToNode(fc2TopologyNodeContext TopologyNode↔ Context, fc2PGRGuid guid, int deviceId)	89
5.20.2.6	fc2TopologyNodeAssignGuidToNodeEx(fc2TopologyNodeContext Topology↔ NodeContext, fc2PGRGuid guid, int deviceId, fc2NodeType nodeType)	89
5.20.2.7	fc2TopologyNodeGetChild(fc2TopologyNodeContext TopologyNodeContext, un↔ signed int position, fc2TopologyNodeContext *pChildTopologyNodeContext)	90

5.20.2.8	fc2TopologyNodeGetDeviceId(fc2TopologyNodeContext TopologyNodeContext, int *pID)	90
5.20.2.9	fc2TopologyNodeGetGuid(fc2TopologyNodeContext TopologyNodeContext, fc2PGRGuid *pGuid)	90
5.20.2.10	fc2TopologyNodeGetInterfaceType(fc2TopologyNodeContext TopologyNodeContext, fc2InterfaceType *pInterfaceType)	91
5.20.2.11	fc2TopologyNodeGetNodeType(fc2TopologyNodeContext TopologyNodeContext, fc2NodeType *pNodeType)	91
5.20.2.12	fc2TopologyNodeGetNumChildren(fc2TopologyNodeContext TopologyNodeContext, unsigned int *pNumChildNodes)	91
5.20.2.13	fc2TopologyNodeGetNumPorts(fc2TopologyNodeContext TopologyNodeContext, unsigned int *pNumPorts)	91
5.20.2.14	fc2TopologyNodeGetPortType(fc2TopologyNodeContext TopologyNodeContext, unsigned int position, fc2PortType *pPortType)	92
5.21	Utilities	93
5.21.1	Detailed Description	93
5.21.2	Function Documentation	93
5.21.2.1	fc2CheckDriver(const fc2PGRGuid *pGuid)	93
5.21.2.2	fc2ErrorToDescription(fc2Error error)	94
5.21.2.3	fc2GetDriverDeviceName(const fc2PGRGuid *pGuid, char *pDeviceName, size_t *deviceNameLength)	94
5.21.2.4	fc2GetLibraryVersion(fc2Version *pVersion)	94
5.21.2.5	fc2GetSystemInfo(fc2SystemInfo *pSystemInfo)	95
5.21.2.6	fc2LaunchBrowser(const char *pAddress)	96
5.21.2.7	fc2LaunchCommand(const char *pCommand)	96
5.21.2.8	fc2LaunchCommandAsync(const char *pCommand, fc2AsyncCommandCallback pCallback, void *pUserData)	96
5.21.2.9	fc2LaunchHelp(const char *pFileName)	97
5.22	TypeDefs	98
5.22.1	Detailed Description	98
5.22.2	Macro Definition Documentation	98
5.22.2.1	FALSE	98
5.22.2.2	FULL_32BIT_VALUE	98
5.22.2.3	MAX_STRING_LENGTH	98

5.22.2.4	TRUE	98
5.22.3	Typedef Documentation	98
5.22.3.1	BOOL	98
5.22.3.2	fc2AVIContext	98
5.22.3.3	fc2Context	99
5.22.3.4	fc2GuiContext	99
5.22.3.5	fc2ImageImpl	99
5.22.3.6	fc2ImageStatisticsContext	99
5.22.3.7	fc2TopologyNodeContext	99
5.23	Enumerations	100
5.23.1	Detailed Description	105
5.23.2	Enumeration Type Documentation	105
5.23.2.1	fc2BandwidthAllocation	105
5.23.2.2	fc2BayerTileFormat	105
5.23.2.3	fc2BusCallbackType	106
5.23.2.4	fc2BusSpeed	106
5.23.2.5	fc2ColorProcessingAlgorithm	106
5.23.2.6	fc2DriverType	107
5.23.2.7	fc2Error	107
5.23.2.8	fc2FrameRate	108
5.23.2.9	fc2GrabMode	109
5.23.2.10	fc2GrabTimeout	109
5.23.2.11	fc2ImageFileFormat	109
5.23.2.12	fc2InterfaceType	110
5.23.2.13	fc2Mode	110
5.23.2.14	fc2PCleBusSpeed	111
5.23.2.15	fc2PixelFormat	111
5.23.2.16	fc2PropertyType	112
5.23.2.17	fc2VideoMode	112
5.24	GigE specific enumerations	114

5.24.1 Detailed Description	114
5.24.2 Enumeration Type Documentation	114
5.24.2.1 fc2GigEPropertyType	114
5.25 Structures	115
5.25.1 Detailed Description	116
5.26 GigE specific structures	117
5.26.1 Detailed Description	117
5.27 IIDC specific structures	118
5.27.1 Detailed Description	118
5.28 Image saving structures.	119
5.28.1 Detailed Description	120
5.28.2 Typedef Documentation	120
5.28.2.1 fc2AsyncCommandCallback	120
5.28.2.2 fc2BusEventCallback	120
5.28.2.3 fc2CallbackHandle	120
5.28.2.4 fc2CameraEventCallback	120
5.28.2.5 fc2ImageEventCallback	120
5.28.3 Enumeration Type Documentation	120
5.28.3.1 fc2TIFFCompressionMethod	120
6 Data Structure Documentation	121
6.1 fc2AVIOption Struct Reference	121
6.1.1 Detailed Description	121
6.1.2 Field Documentation	121
6.1.2.1 frameRate	121
6.1.2.2 reserved	121
6.2 fc2BMPOption Struct Reference	122
6.2.1 Detailed Description	122
6.2.2 Field Documentation	122
6.2.2.1 indexedColor_8bit	122
6.2.2.2 reserved	122

6.3	fc2CameraInfo Struct Reference	122
6.3.1	Detailed Description	124
6.3.2	Field Documentation	124
6.3.2.1	applicationIPAddress	124
6.3.2.2	applicationPort	124
6.3.2.3	bayerTileFormat	124
6.3.2.4	busNumber	124
6.3.2.5	ccpStatus	124
6.3.2.6	configROM	125
6.3.2.7	defaultGateway	125
6.3.2.8	driverName	125
6.3.2.9	driverType	125
6.3.2.10	firmwareBuildTime	125
6.3.2.11	firmwareVersion	125
6.3.2.12	gigEMajorVersion	125
6.3.2.13	gigEMinorVersion	125
6.3.2.14	iidcVer	125
6.3.2.15	interfaceType	125
6.3.2.16	ipAddress	126
6.3.2.17	isColorCamera	126
6.3.2.18	macAddress	126
6.3.2.19	maximumBusSpeed	126
6.3.2.20	modelName	126
6.3.2.21	nodeNumber	126
6.3.2.22	pcieBusSpeed	126
6.3.2.23	reserved	126
6.3.2.24	sensorInfo	126
6.3.2.25	sensorResolution	126
6.3.2.26	serialNumber	127
6.3.2.27	subnetMask	127

6.3.2.28	userDefinedName	127
6.3.2.29	vendorName	127
6.3.2.30	xmlURL1	127
6.3.2.31	xmlURL2	127
6.4	fc2CameraStats Struct Reference	127
6.4.1	Detailed Description	128
6.4.2	Field Documentation	128
6.4.2.1	cameraCurrents	128
6.4.2.2	cameraPowerUp	128
6.4.2.3	cameraVoltages	128
6.4.2.4	imageCorrupt	128
6.4.2.5	imageDriverDropped	128
6.4.2.6	imageDropped	128
6.4.2.7	imageXmitFailed	128
6.4.2.8	numCurrents	128
6.4.2.9	numResendPacketsReceived	129
6.4.2.10	numResendPacketsRequested	129
6.4.2.11	numVoltages	129
6.4.2.12	portErrors	129
6.4.2.13	regReadFailed	129
6.4.2.14	regWriteFailed	129
6.4.2.15	reserved	129
6.4.2.16	temperature	129
6.4.2.17	timeSinceBusReset	129
6.4.2.18	timeSinceInitialization	129
6.4.2.19	timeStamp	129
6.5	fc2Config Struct Reference	129
6.5.1	Detailed Description	130
6.5.2	Field Documentation	130
6.5.2.1	asyncBusSpeed	130

6.5.2.2	bandwidthAllocation	130
6.5.2.3	grabMode	130
6.5.2.4	grabTimeout	131
6.5.2.5	highPerformanceRetrieveBuffer	131
6.5.2.6	isochBusSpeed	131
6.5.2.7	minNumImageNotifications	131
6.5.2.8	numBuffers	131
6.5.2.9	numImageNotifications	131
6.5.2.10	registerTimeout	132
6.5.2.11	registerTimeoutRetries	132
6.5.2.12	reserved	132
6.6	fc2ConfigROM Struct Reference	132
6.6.1	Detailed Description	133
6.6.2	Field Documentation	133
6.6.2.1	chipIdHi	133
6.6.2.2	chipIdLo	133
6.6.2.3	nodeVendorId	133
6.6.2.4	pszKeyword	133
6.6.2.5	reserved	133
6.6.2.6	unitSpecId	133
6.6.2.7	unitSubSWVer	133
6.6.2.8	unitSWVer	133
6.6.2.9	vendorUniqueInfo_0	133
6.6.2.10	vendorUniqueInfo_1	134
6.6.2.11	vendorUniqueInfo_2	134
6.6.2.12	vendorUniqueInfo_3	134
6.7	fc2EmbeddedImageInfo Struct Reference	134
6.7.1	Detailed Description	134
6.7.2	Field Documentation	135
6.7.2.1	brightness	135

6.7.2.2	exposure	135
6.7.2.3	frameCounter	135
6.7.2.4	gain	135
6.7.2.5	GPIOPinState	135
6.7.2.6	ROIPosition	135
6.7.2.7	shutter	135
6.7.2.8	strobePattern	135
6.7.2.9	timestamp	135
6.7.2.10	whiteBalance	135
6.8	fc2EmbeddedImageInfoProperty Struct Reference	135
6.8.1	Detailed Description	135
6.8.2	Field Documentation	136
6.8.2.1	available	136
6.8.2.2	onOff	136
6.9	fc2EventCallbackData Struct Reference	136
6.9.1	Field Documentation	136
6.9.1.1	EventData	136
6.9.1.2	EventDataSize	136
6.9.1.3	EventID	137
6.9.1.4	EventName	137
6.9.1.5	EventTimestamp	137
6.9.1.6	EventUserData	137
6.9.1.7	EventUserDataSize	137
6.10	fc2EventOptions Struct Reference	137
6.10.1	Detailed Description	138
6.10.2	Field Documentation	138
6.10.2.1	EventCallbackFcn	138
6.10.2.2	EventName	138
6.10.2.3	EventUserData	138
6.10.2.4	EventUserDataSize	138

6.11	fc2Format7ImageSettings Struct Reference	138
6.11.1	Detailed Description	139
6.11.2	Field Documentation	139
6.11.2.1	height	139
6.11.2.2	mode	139
6.11.2.3	offsetX	139
6.11.2.4	offsetY	139
6.11.2.5	pixelFormat	139
6.11.2.6	reserved	139
6.11.2.7	width	139
6.12	fc2Format7Info Struct Reference	139
6.12.1	Detailed Description	140
6.12.2	Field Documentation	140
6.12.2.1	imageHStepSize	140
6.12.2.2	imageVStepSize	140
6.12.2.3	maxHeight	141
6.12.2.4	maxPacketSize	141
6.12.2.5	maxWidth	141
6.12.2.6	minPacketSize	141
6.12.2.7	mode	141
6.12.2.8	offsetHStepSize	141
6.12.2.9	offsetVStepSize	141
6.12.2.10	packetSize	141
6.12.2.11	percentage	141
6.12.2.12	pixelFormatBitField	141
6.12.2.13	reserved	142
6.12.2.14	vendorPixelFormatBitField	142
6.13	fc2Format7PacketInfo Struct Reference	142
6.13.1	Detailed Description	142
6.13.2	Field Documentation	142

6.13.2.1	maxBytesPerPacket	142
6.13.2.2	recommendedBytesPerPacket	142
6.13.2.3	reserved	143
6.13.2.4	unitBytesPerPacket	143
6.14	fc2GigEConfig Struct Reference	143
6.14.1	Detailed Description	143
6.14.2	Field Documentation	143
6.14.2.1	enablePacketResend	143
6.14.2.2	registerTimeout	143
6.14.2.3	registerTimeoutRetries	144
6.14.2.4	reserved	144
6.15	fc2GigEImageSettings Struct Reference	144
6.15.1	Detailed Description	144
6.15.2	Field Documentation	144
6.15.2.1	height	144
6.15.2.2	offsetX	144
6.15.2.3	offsetY	145
6.15.2.4	pixelFormat	145
6.15.2.5	reserved	145
6.15.2.6	width	145
6.16	fc2GigEImageSettingsInfo Struct Reference	145
6.16.1	Detailed Description	146
6.16.2	Field Documentation	146
6.16.2.1	imageHStepSize	146
6.16.2.2	imageVStepSize	146
6.16.2.3	maxHeight	146
6.16.2.4	maxWidth	146
6.16.2.5	offsetHStepSize	146
6.16.2.6	offsetVStepSize	146
6.16.2.7	pixelFormatBitField	146

6.16.2.8	reserved	146
6.16.2.9	vendorPixelFormatBitField	147
6.17	fc2GigEProperty Struct Reference	147
6.17.1	Detailed Description	147
6.17.2	Field Documentation	147
6.17.2.1	isReadable	147
6.17.2.2	isWritable	147
6.17.2.3	max	148
6.17.2.4	min	148
6.17.2.5	propType	148
6.17.2.6	reserved	148
6.17.2.7	value	148
6.18	fc2GigEStreamChannel Struct Reference	148
6.18.1	Detailed Description	149
6.18.2	Field Documentation	149
6.18.2.1	destinationIpAddress	149
6.18.2.2	doNotFragment	149
6.18.2.3	hostPort	149
6.18.2.4	interPacketDelay	149
6.18.2.5	networkInterfaceIndex	149
6.18.2.6	packetSize	149
6.18.2.7	reserved	149
6.18.2.8	sourcePort	149
6.19	fc2H264Option Struct Reference	150
6.19.1	Detailed Description	150
6.19.2	Field Documentation	150
6.19.2.1	bitrate	150
6.19.2.2	frameRate	150
6.19.2.3	height	150
6.19.2.4	reserved	150

6.19.2.5	width	150
6.20	fc2Image Struct Reference	151
6.20.1	Field Documentation	151
6.20.1.1	bayerFormat	151
6.20.1.2	cols	151
6.20.1.3	dataSize	151
6.20.1.4	format	151
6.20.1.5	imageImpl	151
6.20.1.6	pData	151
6.20.1.7	receivedDataSize	151
6.20.1.8	rows	151
6.20.1.9	stride	151
6.21	fc2ImageMetadata Struct Reference	151
6.21.1	Detailed Description	152
6.21.2	Field Documentation	152
6.21.2.1	embeddedBrightness	152
6.21.2.2	embeddedExposure	152
6.21.2.3	embeddedFrameCounter	152
6.21.2.4	embeddedGain	152
6.21.2.5	embeddedGPIOPinState	153
6.21.2.6	embeddedROIPosition	153
6.21.2.7	embeddedShutter	153
6.21.2.8	embeddedStrobePattern	153
6.21.2.9	embeddedTimeStamp	153
6.21.2.10	embeddedWhiteBalance	153
6.21.2.11	reserved	153
6.22	fc2ImageContext Struct Reference	153
6.22.1	Field Documentation	154
6.22.1.1	pBusMgr	154
6.22.1.2	pCamera	154

6.23	fc2InternalGuiContext Struct Reference	154
6.23.1	Field Documentation	154
6.23.1.1	pCameraControlDlg	154
6.23.1.2	pCameraSelectionDlg	154
6.24	fc2InternalImageCallback Struct Reference	154
6.24.1	Field Documentation	154
6.24.1.1	pCallback	154
6.24.1.2	pCallbackData	154
6.25	fc2IPAddress Struct Reference	155
6.25.1	Detailed Description	155
6.25.2	Field Documentation	155
6.25.2.1	octets	155
6.26	fc2JPEGOption Struct Reference	155
6.26.1	Detailed Description	155
6.26.2	Field Documentation	155
6.26.2.1	progressive	155
6.26.2.2	quality	156
6.26.2.3	reserved	156
6.27	fc2JPG2Option Struct Reference	156
6.27.1	Detailed Description	156
6.27.2	Field Documentation	156
6.27.2.1	quality	156
6.27.2.2	reserved	157
6.28	fc2LUTData Struct Reference	157
6.28.1	Detailed Description	157
6.28.2	Field Documentation	157
6.28.2.1	enabled	157
6.28.2.2	inputBitDepth	157
6.28.2.3	numBanks	158
6.28.2.4	numChannels	158

6.28.2.5	numEntries	158
6.28.2.6	outputBitDepth	158
6.28.2.7	reserved	158
6.28.2.8	supported	158
6.29	fc2MACAddress Struct Reference	158
6.29.1	Detailed Description	158
6.29.2	Field Documentation	159
6.29.2.1	octets	159
6.30	fc2MJPGOption Struct Reference	159
6.30.1	Detailed Description	159
6.30.2	Field Documentation	159
6.30.2.1	frameRate	159
6.30.2.2	quality	159
6.30.2.3	reserved	159
6.31	fc2PGMOption Struct Reference	160
6.31.1	Detailed Description	160
6.31.2	Field Documentation	160
6.31.2.1	binaryFile	160
6.31.2.2	reserved	160
6.32	fc2PGRGuid Struct Reference	160
6.32.1	Detailed Description	160
6.32.2	Field Documentation	161
6.32.2.1	value	161
6.33	fc2PNGOption Struct Reference	161
6.33.1	Detailed Description	161
6.33.2	Field Documentation	161
6.33.2.1	compressionLevel	161
6.33.2.2	interlaced	161
6.33.2.3	reserved	161
6.34	fc2PPMOption Struct Reference	162

6.34.1 Detailed Description	162
6.34.2 Field Documentation	162
6.34.2.1 binaryFile	162
6.34.2.2 reserved	162
6.35 fc2StrobeControl Struct Reference	162
6.35.1 Detailed Description	163
6.35.2 Field Documentation	163
6.35.2.1 delay	163
6.35.2.2 duration	163
6.35.2.3 onOff	163
6.35.2.4 polarity	163
6.35.2.5 reserved	163
6.35.2.6 source	163
6.36 fc2StrobeInfo Struct Reference	163
6.36.1 Detailed Description	164
6.36.2 Field Documentation	164
6.36.2.1 maxValue	164
6.36.2.2 minValue	164
6.36.2.3 onOffSupported	164
6.36.2.4 polaritySupported	164
6.36.2.5 present	164
6.36.2.6 readOutSupported	165
6.36.2.7 reserved	165
6.36.2.8 source	165
6.37 fc2SystemInfo Struct Reference	165
6.37.1 Detailed Description	166
6.37.2 Field Documentation	166
6.37.2.1 byteOrder	166
6.37.2.2 cpuDescription	166
6.37.2.3 driverList	166

6.37.2.4	gpuDescription	166
6.37.2.5	libraryList	166
6.37.2.6	numCpuCores	166
6.37.2.7	osDescription	166
6.37.2.8	osType	166
6.37.2.9	reserved	166
6.37.2.10	screenHeight	167
6.37.2.11	screenWidth	167
6.37.2.12	sysMemSize	167
6.38	fc2TIFFOption Struct Reference	167
6.38.1	Detailed Description	167
6.38.2	Field Documentation	167
6.38.2.1	compression	167
6.38.2.2	reserved	167
6.39	fc2TimeStamp Struct Reference	168
6.39.1	Detailed Description	168
6.39.2	Field Documentation	168
6.39.2.1	cycleCount	168
6.39.2.2	cycleOffset	168
6.39.2.3	cycleSeconds	168
6.39.2.4	microSeconds	168
6.39.2.5	reserved	168
6.39.2.6	seconds	169
6.40	fc2TriggerDelay Struct Reference	169
6.40.1	Detailed Description	169
6.40.2	Field Documentation	170
6.40.2.1	absControl	170
6.40.2.2	absValue	170
6.40.2.3	autoManualMode	170
6.40.2.4	onePush	170

6.40.2.5	onOff	170
6.40.2.6	present	170
6.40.2.7	reserved	170
6.40.2.8	type	170
6.40.2.9	valueA	170
6.40.2.10	valueB	171
6.41	fc2TriggerDelayInfo Struct Reference	171
6.41.1	Detailed Description	172
6.41.2	Field Documentation	172
6.41.2.1	absMax	172
6.41.2.2	absMin	172
6.41.2.3	absValSupported	172
6.41.2.4	autoSupported	172
6.41.2.5	manualSupported	172
6.41.2.6	max	172
6.41.2.7	min	172
6.41.2.8	onePushSupported	172
6.41.2.9	onOffSupported	173
6.41.2.10	present	173
6.41.2.11	pUnitAbbr	173
6.41.2.12	pUnits	173
6.41.2.13	readOutSupported	173
6.41.2.14	reserved	173
6.41.2.15	type	173
6.42	fc2TriggerMode Struct Reference	173
6.42.1	Detailed Description	174
6.42.2	Field Documentation	174
6.42.2.1	mode	174
6.42.2.2	onOff	174
6.42.2.3	parameter	174

6.42.2.4	polarity	174
6.42.2.5	reserved	174
6.42.2.6	source	175
6.43	fc2TriggerModelInfo Struct Reference	175
6.43.1	Detailed Description	175
6.43.2	Field Documentation	175
6.43.2.1	modeMask	175
6.43.2.2	onOffSupported	176
6.43.2.3	polaritySupported	176
6.43.2.4	present	176
6.43.2.5	readOutSupported	176
6.43.2.6	reserved	176
6.43.2.7	softwareTriggerSupported	176
6.43.2.8	sourceMask	176
6.43.2.9	valueReadable	176
6.44	fc2Version Struct Reference	176
6.44.1	Detailed Description	177
6.44.2	Field Documentation	177
6.44.2.1	build	177
6.44.2.2	major	177
6.44.2.3	minor	177
6.44.2.4	type	177

7 File Documentation	179
7.1 FlyCapture2_C.h File Reference	179
7.2 FlyCapture2Defs_C.h File Reference	179
7.2.1 Enumeration Type Documentation	188
7.2.1.1 fc2ByteOrder	188
7.2.1.2 fc2NodeType	188
7.2.1.3 fc2OSType	188
7.2.1.4 fc2PortType	189
7.2.1.5 fc2StatisticsChannel	189
7.3 FlyCapture2GUI_C.h File Reference	189
7.3.1 Function Documentation	190
7.3.1.1 fc2CreateGUIContext(fc2GuiContext *pContext)	190
7.3.1.2 fc2DestroyGUIContext(fc2GuiContext context)	190
7.3.1.3 fc2Disconnect(fc2GuiContext context) __attribute__((deprecated))	190
7.3.1.4 fc2GUIConnect(fc2GuiContext context, fc2Context cameraContext)	190
7.3.1.5 fc2GUIDisconnect(fc2GuiContext context)	191
7.3.1.6 fc2Hide(fc2GuiContext context)	191
7.3.1.7 fc2IsVisible(fc2GuiContext context)	191
7.3.1.8 fc2Show(fc2GuiContext context)	192
7.3.1.9 fc2ShowModal(fc2GuiContext context, BOOL *pOkSelected, fc2PGRGuid *guidArray, unsigned int *size)	192
7.4 FlyCapture2Internal_C.h File Reference	192
7.4.1 Function Documentation	193
7.4.1.1 IsContextValid(fc2Context context)	193
7.4.1.2 IsGuiContextValid(fc2GuiContext context)	193
7.4.1.3 SyncCpplImageToStruct(fc2Image *pImage)	193
7.5 FlyCapture2Platform_C.h File Reference	193
7.5.1 Macro Definition Documentation	193
7.5.1.1 FLYCAPTURE2_C_API	193
7.5.1.2 FLYCAPTURE2_C_CALL_CONVEN	193
7.6 FlyCapture2Private_C.h File Reference	193

7.6.1	Function Documentation	193
7.6.1.1	GetInternal(unsigned int index)	193
7.7	MultiSyncLibrary_C.h File Reference	193
7.7.1	Function Documentation	194
7.7.1.1	syncCreateContext(syncContext *pContext)	194
7.7.1.2	syncDestroyContext(syncContext context)	194
7.7.1.3	syncDisableCrossPCSSynchronization(syncContext context)	195
7.7.1.4	syncEnableCrossPCSSynchronization(syncContext context)	195
7.7.1.5	syncGetStatus(syncContext context)	195
7.7.1.6	syncGetTimeSinceSynced(syncContext context)	196
7.7.1.7	syncIsTimingBusConnected(syncContext context)	196
7.7.1.8	syncQueryCrossPCSSynchronizationSetting(syncContext context)	196
7.7.1.9	syncRescanMasterTimingBus(syncContext context)	196
7.7.1.10	syncStart(syncContext context)	197
7.7.1.11	syncStop(syncContext context)	197
7.8	MultiSyncLibraryDefs_C.h File Reference	197
7.8.1	Macro Definition Documentation	198
7.8.1.1	FALSE	198
7.8.1.2	FULL_32BIT_VALUE	198
7.8.1.3	MAX_STRING_LENGTH	198
7.8.1.4	TRUE	198
7.8.2	Typedef Documentation	198
7.8.2.1	BOOL	198
7.8.2.2	syncContext	198
7.8.3	Enumeration Type Documentation	198
7.8.3.1	syncError	198
7.8.3.2	syncMessage	199
7.9	MultiSyncLibraryPlatform_C.h File Reference	199
7.9.1	Macro Definition Documentation	199
7.9.1.1	MULTISYNCLIBRARY_C_API	199
7.9.1.2	MULTISYNCLIBRARY_C_CALL_CONVEN	199

Chapter 1

Deprecated List

Global **fc2Disonnect** (fc2GuiContext context) __attribute__((deprecated))

This method is deprecated and will be removed in a future FlyCapture2 release. Please use fc2GUIDisconnect instead.

Chapter 2

Module Index

2.1 Modules

Here is a list of all modules:

Bus Manager Operation	9
Connection and Image Retrieval	19
Information and Properties	26
General Purpose Input / Output	29
Trigger	31
Strobe	37
Look Up Table	40
Memory Channels	44
Register Operation	48
DCAM Formats	51
Format7	53
GVCP Register Operation	56
GigE property manipulation	60
GigE image settings	62
GigE image binning settings	65
GigE image stream configuration	67
Image Operation	70
Image Statistics Operation	77
AVI Recording Operation	84
TopologyNode Operation	87
Utilities	93
TypeDefs	98
Enumerations	100
GigE specific enumerations	114
Structures	115
GigE specific structures	117
IIDC specific structures	118
Image saving structures.	119

Chapter 3

Data Structure Index

3.1 Data Structures

Here are the data structures with brief descriptions:

fc2AVIOption	Options for saving AVI files	121
fc2BMPOption	Options for saving Bitmap image	122
fc2CameraInfo	Camera information	122
fc2CameraStats	Camera diagnostic information	127
fc2Config	Configuration for a camera	129
fc2ConfigROM	Camera configuration ROM	132
fc2EmbeddedImageInfo	Properties of the possible embedded image information	134
fc2EmbeddedImageInfoProperty	Properties of a single embedded image info property	135
fc2EventCallbackData	136
fc2EventOptions	Options for enabling device event registration	137
fc2Format7ImageSettings	Format 7 image settings	138
fc2Format7Info	Format 7 information for a single mode	139
fc2Format7PacketInfo	Format 7 packet information	142
fc2GigEConfig	Configuration for a GigE camera	143
fc2GigEImageSettings	Image settings for a GigE camera	144
fc2GigEImageSettingsInfo	Format 7 information for a single mode	145
fc2GigEProperty	A GigE property	147
fc2GigEStreamChannel	Information about a single GigE stream channel	148

fc2H264Option	
Options for saving H264 files	150
fc2Image	151
fc2ImageMetadata	
Metadata related to an image	151
fc2InternalContext	153
fc2InternalGuiContext	154
fc2InternalImageCallback	154
fc2IPAddress	
IPv4 address	155
fc2JPEGOption	
Options for saving JPEG image	155
fc2JPG2Option	
Options for saving JPEG2000 image	156
fc2LUTData	
Information about the camera's look up table	157
fc2MACAddress	
MAC address	158
fc2MJPGOption	
Options for saving MJPG files	159
fc2PGMOption	
Options for saving PGM images	160
fc2PGRGuid	
A GUID to the camera	160
fc2PNGOption	
Options for saving PNG images	161
fc2PPMOption	
Options for saving PPM images	162
fc2StrobeControl	
A camera strobe	162
fc2StrobeInfo	
A camera strobe property	163
fc2SystemInfo	
Description of the system	165
fc2TIFFOption	
Options for saving TIFF images	167
fc2TimeStamp	
Timestamp information	168
fc2TriggerDelay	
A specific camera property	169
fc2TriggerDelayInfo	
Information about a specific camera property	171
fc2TriggerMode	
A camera trigger	173
fc2TriggerModelInfo	
Information about a camera trigger property	175
fc2Version	
The current version of the library	176

Chapter 4

File Index

4.1 File List

Here is a list of all files with brief descriptions:

FlyCapture2_C.h	179
FlyCapture2Defs_C.h	179
FlyCapture2GUI_C.h	189
FlyCapture2Internal_C.h	192
FlyCapture2Platform_C.h	193
FlyCapture2Private_C.h	193
MultiSyncLibrary_C.h	193
MultiSyncLibraryDefs_C.h	197
MultiSyncLibraryPlatform_C.h	199

Chapter 5

Module Documentation

5.1 Bus Manager Operation

The functions in this section provide access to BusManager operations.

Functions

- [FLYCAPTURE2_C_API fc2Error fc2FireBusReset](#) ([fc2Context](#) context, [fc2PGRGuid](#) *pGuid)
Fire a bus reset.
- [FLYCAPTURE2_C_API fc2Error fc2GetNumOfCameras](#) ([fc2Context](#) context, unsigned int *pNumCameras)
Gets the number of cameras attached to the PC.
- [FLYCAPTURE2_C_API fc2Error fc2GetCameraFromIPAddress](#) ([fc2Context](#) context, [fc2IPAddress](#) ip↔Address, [fc2PGRGuid](#) *pGuid)
Gets the PGRGuid for a camera with the specified IPv4 address.
- [FLYCAPTURE2_C_API fc2Error fc2GetCameraFromIndex](#) ([fc2Context](#) context, unsigned int index, [fc2PGRGuid](#) *pGuid)
Gets the PGRGuid for a camera on the PC.
- [FLYCAPTURE2_C_API fc2Error fc2GetCameraFromSerialNumber](#) ([fc2Context](#) context, unsigned int serial↔Number, [fc2PGRGuid](#) *pGuid)
Gets the PGRGuid for a camera on the PC.
- [FLYCAPTURE2_C_API fc2Error fc2GetCameraSerialNumberFromIndex](#) ([fc2Context](#) context, unsigned int index, unsigned int *pSerialNumber)
Gets the serial number of the camera with the specified index.
- [FLYCAPTURE2_C_API fc2Error fc2GetInterfaceTypeFromGuid](#) ([fc2Context](#) context, [fc2PGRGuid](#) *pGuid, [fc2InterfaceType](#) *pInterfaceType)
Gets the interface type associated with a PGRGuid.
- [FLYCAPTURE2_C_API fc2Error fc2GetNumOfDevices](#) ([fc2Context](#) context, unsigned int *pNumDevices)
Gets the number of devices.
- [FLYCAPTURE2_C_API fc2Error fc2GetDeviceFromIndex](#) ([fc2Context](#) context, unsigned int index, [fc2PGRGuid](#) *pGuid)
Gets the PGRGuid for a device.
- [FLYCAPTURE2_C_API fc2Error fc2ReadPhyRegister](#) ([fc2Context](#) context, [fc2PGRGuid](#) guid, unsigned int page, unsigned int port, unsigned int address, unsigned int *pValue)
Read a phy register on the specified device.
- [FLYCAPTURE2_C_API fc2Error fc2WritePhyRegister](#) ([fc2Context](#) context, [fc2PGRGuid](#) guid, unsigned int page, unsigned int port, unsigned int address, unsigned int value)

Write a phy register on the specified device.

- `FLYCAPTURE2_C_API fc2Error fc2GetUsbLinkInfo (fc2Context context, fc2PGRGuid guid, unsigned int *pValue)`

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

- `FLYCAPTURE2_C_API fc2Error fc2GetUsbPortStatus (fc2Context context, fc2PGRGuid guid, unsigned int *pValue)`

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

- `FLYCAPTURE2_C_API fc2Error fc2GetTopology (fc2Context context, fc2TopologyNodeContext *pTopologyNodeContext)`

Gets the topology information for the PC.

- `FLYCAPTURE2_C_API fc2Error fc2RegisterCallback (fc2Context context, fc2BusEventCallback enumCallback, fc2BusCallbackType callbackType, void *pParameter, fc2CallbackHandle *pCallbackHandle)`

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

- `FLYCAPTURE2_C_API fc2Error fc2UnregisterCallback (fc2Context context, fc2CallbackHandle callbackHandle)`

Unregister a callback function.

- `FLYCAPTURE2_C_API fc2Error fc2RescanBus (fc2Context context)`

Force a rescan of the buses.

- `FLYCAPTURE2_C_API fc2Error fc2ForceIPAddressToCamera (fc2Context context, fc2MACAddress macAddress, fc2IPAddress ipAddress, fc2IPAddress subnetMask, fc2IPAddress defaultGateway)`

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

- `FLYCAPTURE2_C_API fc2Error fc2ForceAllIPAddressesAutomatically ()`

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.

- `FLYCAPTURE2_C_API fc2Error fc2ForceIPAddressAutomatically (unsigned int serialNumber)`

Force cameras on the network to be assigned sequential IP addresses on the same subnet as the network adapters that it is connected to.

- `FLYCAPTURE2_C_API fc2Error fc2DiscoverGigECameras (fc2Context context, fc2CameraInfo *gigECameras, unsigned int *arraySize)`

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

- `FLYCAPTURE2_C_API fc2Error fc2IsCameraControlable (fc2Context context, fc2PGRGuid *pGuid, BOOL *pControlable)`

Query whether a GigE camera is controllable.

5.1.1 Detailed Description

The functions in this section provide access to BusManager operations.

5.1.2 Function Documentation

5.1.2.1 `FLYCAPTURE2_C_API fc2Error fc2DiscoverGigECameras (fc2Context context, fc2CameraInfo *gigECameras, unsigned int * arraySize)`

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

This is useful in situations where a GigE camera is using Persistent IP and the application's subnet is different from the device subnet. After discovering the camera, it is easy to use `ForceIPAddressToCamera()` to set a different IP configuration.

Parameters

<i>context</i>	The fc2Context to be used.
<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.

5.1.2.2 FLYCAPTURE2_C_API fc2Error fc2FireBusReset (fc2Context *context*, fc2PGRGuid * *pGuid*)

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>context</i>	The fc2Context to be used.
<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.

5.1.2.3 FLYCAPTURE2_C_API fc2Error fc2ForceAllIPAddressesAutomatically ()

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.

5.1.2.4 FLYCAPTURE2_C_API fc2Error fc2ForceIPAddressAutomatically (unsigned int *serialNumber*)

Force cameras on the network to be assigned sequential IP addresses on the same subnet as the network adapters that it is connected to.

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

Returns

An Error indicating the success or failure of the function.

5.1.2.5 FLYCAPTURE2_C_API `fc2Error fc2ForceIPAddressToCamera (fc2Context context, fc2MACAddress macAddress, fc2IPAddress ipAddress, fc2IPAddress subnetMask, fc2IPAddress defaultGateway)`

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>context</i>	The fc2Context to be used.
<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.

5.1.2.6 FLYCAPTURE2_C_API `fc2Error fc2GetCameraFromIndex (fc2Context context, unsigned int index, fc2PGRGuid * pGuid)`

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 [fc2Connect\(\)](#) call.

Parameters

<i>context</i>	The fc2Context to be used.
<i>index</i>	Zero based index of camera.
<i>pGuid</i>	Unique PGRGuid for the camera.

Returns

A fc2Error indicating the success or failure of the function.

5.1.2.7 FLYCAPTURE2_C_API `fc2Error fc2GetCameraFromIPAddress (fc2Context context, fc2IPAddress ipAddress, fc2PGRGuid * pGuid)`

Gets the PGRGuid for a camera with the specified IPv4 address.

Parameters

<i>context</i>	The fc2Context to be used.
<i>ipAddress</i>	IP address to get GUID for.
<i>pGuid</i>	Unique PGRGuid for the camera.

Returns

A `fc2Error` indicating the success or failure of the function.

5.1.2.8 FLYCAPTURE2_C_API `fc2Error` `fc2GetCameraFromSerialNumber` (`fc2Context` *context*, unsigned int *serialNumber*, `fc2PGRGuid` * *pGuid*)

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 [fc2↔Connect\(\)](#) call.

Parameters

<i>context</i>	The <code>fc2Context</code> to be used.
<i>serialNumber</i>	Serial number of camera.
<i>pGuid</i>	Unique PGRGuid for the camera.

Returns

A `fc2Error` indicating the success or failure of the function.

5.1.2.9 FLYCAPTURE2_C_API `fc2Error` `fc2GetCameraSerialNumberFromIndex` (`fc2Context` *context*, unsigned int *index*, unsigned int * *pSerialNumber*)

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

Parameters

<i>context</i>	The <code>fc2Context</code> to be used.
<i>index</i>	Zero based index of desired camera.
<i>pSerialNumber</i>	Serial number of camera.

Returns

A `fc2Error` indicating the success or failure of the function.

5.1.2.10 FLYCAPTURE2_C_API `fc2Error` `fc2GetDeviceFromIndex` (`fc2Context` *context*, unsigned int *index*, `fc2PGRGuid` * *pGuid*)

Gets the PGRGuid for a device.

It uniquely identifies the device specified by the index.

Parameters

<i>context</i>	The <code>fc2Context</code> to be used.
<i>index</i>	Zero based index of device.
<i>pGuid</i>	Unique PGRGuid for the device.

See also

[fc2GetNumOfDevices\(\)](#)

Returns

An Error indicating the success or failure of the function.

5.1.2.11 FLYCAPTURE2_C_API fc2Error fc2GetInterfaceTypeFromGuid (fc2Context *context*, fc2PGRGuid * *pGuid*, fc2InterfaceType * *pInterfaceType*)

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>context</i>	The fc2Context to be used.
<i>pGuid</i>	The PGRGuid to get the interface for.
<i>pInterfaceType</i>	The interface type of the PGRGuid.

Returns

5.1.2.12 FLYCAPTURE2_C_API fc2Error fc2GetNumOfCameras (fc2Context *context*, unsigned int * *pNumCameras*)

Gets the number of cameras attached to the PC.

Parameters

<i>context</i>	The fc2Context to be used.
<i>pNumCameras</i>	Number of cameras detected.

Returns

A fc2Error indicating the success or failure of the function.

5.1.2.13 FLYCAPTURE2_C_API fc2Error fc2GetNumOfDevices (fc2Context *context*, unsigned int * *pNumDevices*)

Gets the number of devices.

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

Parameters

<i>context</i>	The fc2Context to be used.
<i>pNumDevices</i>	The number of devices found.

Returns

An Error indicating the success or failure of the function.

5.1.2.14 FLYCAPTURE2_C_API fc2Error fc2GetTopology (fc2Context context, fc2TopologyNodeContext * pTopologyNodeContext)

Gets the topology information for the PC.

Parameters

<i>context</i>	The fc2Context to be used.
<i>pTopologyNodeContext</i>	A Topology Node context that will contain the topology information

Returns

An Error indicating the success or failure of the function.

5.1.2.15 FLYCAPTURE2_C_API fc2Error fc2GetUsbLinkInfo (fc2Context context, fc2PGRGuid guid, unsigned int * pValue)

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

Parameters

<i>context</i>	The fc2Context to be used.
<i>guid</i>	Unique PGRGuid for the device.
<i>pValue</i>	Value read from the card register.

Returns

An Error indicating the success or failure of the function.

5.1.2.16 FLYCAPTURE2_C_API fc2Error fc2GetUsbPortStatus (fc2Context context, fc2PGRGuid guid, unsigned int * pValue)

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

Parameters

<i>context</i>	The fc2Context to be used.
<i>guid</i>	Unique PGRGuid for the device.
<i>pValue</i>	Value read from the card register.

Returns

An Error indicating the success or failure of the function.

5.1.2.17 FLYCAPTURE2_C_API **fc2Error** **fc2IsCameraControllable** (**fc2Context** *context*, **fc2PGRGuid** * *pGuid*, **BOOL** * *pControllable*)

Query whether a GigE camera is controllable.

Parameters

<i>context</i>	The fc2Context to be used.
<i>pGuid</i>	Unique PGRGuid for the camera.
<i>pControllable</i>	True indicates camera is controllable

Returns

A fc2Error indicating the success or failure of the function.

5.1.2.18 FLYCAPTURE2_C_API **fc2Error** **fc2ReadPhyRegister** (**fc2Context** *context*, **fc2PGRGuid** *guid*, **unsigned int** *page*, **unsigned int** *port*, **unsigned int** *address*, **unsigned int** * *pValue*)

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>context</i>	The fc2Context to be used.
<i>guid</i>	Unique PGRGuid for the device.
<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.

5.1.2.19 FLYCAPTURE2_C_API **fc2Error** **fc2RegisterCallback** (**fc2Context** *context*, **fc2BusEventCallback** *enumCallback*, **fc2BusCallbackType** *callbackType*, **void** * *pParameter*, **fc2CallbackHandle** * *pCallbackHandle*)

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

Parameters

<i>context</i>	The fc2Context to be used.
<i>enumCallback</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.

Returns

A fc2Error indicating the success or failure of the function.

5.1.2.20 FLYCAPTURE2_C_API fc2Error fc2RescanBus (fc2Context context)

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.

5.1.2.21 FLYCAPTURE2_C_API fc2Error fc2UnregisterCallback (fc2Context context, fc2CallbackHandle callbackHandle)

Unregister a callback function.

Parameters

<i>context</i>	The fc2Context to be used.
<i>callbackHandle</i>	Unique callback handle.

Returns

A fc2Error indicating the success or failure of the function.

5.1.2.22 FLYCAPTURE2_C_API fc2Error fc2WritePhyRegister (fc2Context context, fc2PGRGuid guid, unsigned int page, unsigned int port, unsigned int address, unsigned int value)

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>context</i>	The fc2Context to be used.
----------------	----------------------------

Parameters

<i>guid</i>	Unique PGRGuid for the device.
<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.

5.2 Connection and Image Retrieval

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

Functions

- **FLYCAPTURE2_C_API fc2Error fc2Connect** (fc2Context context, fc2PGRGuid *guid)
Connects the camera object to the camera specified by the GUID.
- **FLYCAPTURE2_C_API fc2Error fc2Disconnect** (fc2Context context)
Disconnects the fc2Context from the camera.
- **FLYCAPTURE2_C_API fc2Error fc2SetCallback** (fc2Context context, fc2ImageEventCallback pCallbackFn, void *pCallbackData)
Sets the callback data to be used on completion of image transfer.
- **FLYCAPTURE2_C_API fc2Error fc2StartCapture** (fc2Context context)
Starts isochronous image capture.
- **FLYCAPTURE2_C_API fc2Error fc2StartCaptureCallback** (fc2Context context, fc2ImageEventCallback pCallbackFn, void *pCallbackData)
Starts isochronous image capture.
- **FLYCAPTURE2_C_API fc2Error fc2StartSyncCapture** (unsigned int numCameras, fc2Context *pContexts)
Starts synchronized isochronous image capture on multiple cameras.
- **FLYCAPTURE2_C_API fc2Error fc2StartSyncCaptureCallback** (unsigned int numCameras, fc2Context *pContexts, fc2ImageEventCallback *pCallbackFns, void **pCallbackDataArray)
Starts synchronized isochronous image capture on multiple cameras.
- **FLYCAPTURE2_C_API fc2Error fc2RetrieveBuffer** (fc2Context context, fc2Image *pImage)
Retrieves the next image object containing the next image.
- **FLYCAPTURE2_C_API fc2Error fc2StopCapture** (fc2Context context)
Stops isochronous image transfer and cleans up all associated resources.
- **FLYCAPTURE2_C_API fc2Error fc2WaitForBufferEvent** (fc2Context context, fc2Image *pImage, unsigned int eventNumber)
Retrieves the next image event containing the next part of the image.
- **FLYCAPTURE2_C_API fc2Error fc2SetUserBuffers** (fc2Context context, unsigned char *const ppMemBuffers, int size, int nNumBuffers)
Specify user allocated buffers to use as image data buffers.
- **FLYCAPTURE2_C_API fc2Error fc2GetConfiguration** (fc2Context context, fc2Config *config)
Get the configuration associated with the camera.
- **FLYCAPTURE2_C_API fc2Error fc2SetConfiguration** (fc2Context context, fc2Config *config)
Set the configuration associated with the camera.

5.2.1 Detailed Description

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

5.2.2 Function Documentation

5.2.2.1 FLYCAPTURE2_C_API fc2Error fc2Connect (fc2Context context, fc2PGRGuid * guid)

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

Parameters

<i>context</i>	The fc2Context to be used.
<i>guid</i>	The unique identifier for a specific camera on the PC.

Returns

A fc2Error indicating the success or failure of the function.

5.2.2.2 FLYCAPTURE2_C_API fc2Error fc2Disconnect (fc2Context *context*)

Disconnects the fc2Context from the camera.

Parameters

<i>context</i>	The fc2Context to be used.
----------------	----------------------------

Returns

A fc2Error indicating the success or failure of the function.

5.2.2.3 FLYCAPTURE2_C_API fc2Error fc2GetConfiguration (fc2Context *context*, fc2Config * *config*)

Get the configuration associated with the camera.

See also

[fc2SetConfiguration\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>config</i>	Pointer to the configuration structure to be filled.

Returns

A fc2Error indicating the success or failure of the function.

5.2.2.4 FLYCAPTURE2_C_API fc2Error fc2RetrieveBuffer (fc2Context *context*, fc2Image * *plmage*)

Retrieves the next image object containing the next image.

See also

[fc2StartCapture\(\)](#)

[fc2StopCapture\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>pImage</i>	Pointer to fc2Image to store image data.

Returns

A fc2Error indicating the success or failure of the function.

5.2.2.5 FLYCAPTURE2_C_API fc2Error fc2SetCallback (fc2Context *context*, fc2ImageEventCallback *pCallbackFn*, void * *pCallbackData*)

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

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

See also

[fc2StartCapture\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>pCallbackFn</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.

Returns

A fc2Error indicating the success or failure of the function.

5.2.2.6 FLYCAPTURE2_C_API fc2Error fc2SetConfiguration (fc2Context *context*, fc2Config * *config*)

Set the configuration associated with the camera.

See also

[fc2GetConfiguration\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>config</i>	Pointer to the configuration structure to be used.

Returns

A fc2Error indicating the success or failure of the function.

5.2.2.7 FLYCAPTURE2_C_API `fc2Error fc2SetUserBuffers (fc2Context context, unsigned char *const ppMemBuffers, int size, int nNumBuffers)`

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

See also

[fc2StartCapture\(\)](#)
[fc2RetrieveBuffer\(\)](#)
[fc2StopCapture\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>ppMemBuffers</i>	Pointer to memory buffers to be written to. The size of the data should be equal to $(\text{size} * \text{numBuffers})$ or larger.
<i>size</i>	The size of each buffer (in bytes).
<i>nNumBuffers</i>	Number of buffers in the array.

Returns

A fc2Error indicating the success or failure of the function.

5.2.2.8 FLYCAPTURE2_C_API `fc2Error fc2StartCapture (fc2Context context)`

Starts isochronous image capture.

It will use either the current video mode or the most recently set video mode of the camera.

See also

[fc2RetrieveBuffer\(\)](#)
[fc2StartSyncCapture\(\)](#)
[fc2StopCapture\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
----------------	----------------------------

Returns

A fc2Error indicating the success or failure of the function.

5.2.2.9 FLYCAPTURE2_C_API `fc2Error fc2StartCaptureCallback (fc2Context context, fc2ImageEventCallback pCallbackFn, void * pCallbackData)`

Starts isochronous image capture.

It will use either the current video mode or the most recently set video mode of the camera. The callback function is called when a new image is received from the camera.

See also

[fc2RetrieveBuffer\(\)](#)
[fc2StartSyncCapture\(\)](#)
[fc2StopCapture\(\)](#)

Parameters

<i>context</i>	The <code>fc2Context</code> to be used.
<i>pCallbackFn</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. A NULL pointer is acceptable.

Returns

A `fc2Error` indicating the success or failure of the function.

5.2.2.10 FLYCAPTURE2_C_API `fc2Error fc2StartSyncCapture (unsigned int numCameras, fc2Context * pContexts)`

Starts synchronized isochronous image capture on multiple cameras.

See also

[fc2RetrieveBuffer\(\)](#)
[fc2StartCapture\(\)](#)
[fc2StopCapture\(\)](#)

Parameters

<i>numCameras</i>	Number of <code>fc2Contexts</code> in the <code>ppCameras</code> array.
<i>pContexts</i>	Array of <code>fc2Contexts</code> .

Returns

A `fc2Error` indicating the success or failure of the function.

5.2.2.11 FLYCAPTURE2_C_API `fc2Error fc2StartSyncCaptureCallback (unsigned int numCameras, fc2Context * pContexts, fc2ImageEventCallback * pCallbackFns, void ** pCallbackDataArray)`

Starts synchronized isochronous image capture on multiple cameras.

See also

[fc2RetrieveBuffer\(\)](#)
[fc2StartCapture\(\)](#)
[fc2StopCapture\(\)](#)

Parameters

<i>numCameras</i>	Number of fc2Contexts in the ppCameras array.
<i>pContexts</i>	Array of fc2Contexts.
<i>pCallbackFns</i>	Array of callback functions for each camera.
<i>pCallbackDataArray</i>	Array of callback data pointers.

Returns

A fc2Error indicating the success or failure of the function.

5.2.2.12 FLYCAPTURE2_C_API fc2Error fc2StopCapture (fc2Context context)

Stops isochronous image transfer and cleans up all associated resources.

See also

[fc2StartCapture\(\)](#)
[fc2RetrieveBuffer\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
----------------	----------------------------

Returns

A fc2Error indicating the success or failure of the function.

5.2.2.13 FLYCAPTURE2_C_API fc2Error fc2WaitForBufferEvent (fc2Context context, fc2Image * plmage, unsigned int eventNumber)

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

See also

[fc2StartCapture\(\)](#)
[fc2RetrieveBuffer\(\)](#)
[fc2StopCapture\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>plmage</i>	Pointer to fc2Image to store image data.
<i>eventNumber</i>	The event number to wait for.

Returns

An Error indicating the success or failure of the function.

5.3 Information and Properties

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

Functions

- [FLYCAPTURE2_C_API fc2Error fc2GetCameraInfo](#) ([fc2Context](#) context, [fc2CameraInfo](#) *pCameraInfo)
Retrieves information from the camera such as serial number, model name and other camera information.
- [FLYCAPTURE2_C_API fc2Error fc2GetPropertyInfo](#) ([fc2Context](#) context, [fc2PropertyInfo](#) *propInfo)
Retrieves information about the specified camera property.
- [FLYCAPTURE2_C_API fc2Error fc2GetProperty](#) ([fc2Context](#) context, [fc2Property](#) *prop)
Reads the settings for the specified property from the camera.
- [FLYCAPTURE2_C_API fc2Error fc2SetProperty](#) ([fc2Context](#) context, [fc2Property](#) *prop)
Writes the settings for the specified property to the camera.
- [FLYCAPTURE2_C_API fc2Error fc2SetPropertyBroadcast](#) ([fc2Context](#) context, [fc2Property](#) *prop)
Writes the settings for the specified property to the camera.

5.3.1 Detailed Description

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

5.3.2 Function Documentation

5.3.2.1 [FLYCAPTURE2_C_API fc2Error fc2GetCameraInfo](#) ([fc2Context](#) context, [fc2CameraInfo](#) * pCameraInfo)

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

Parameters

<i>context</i>	The fc2Context to be used.
<i>pCameraInfo</i>	Pointer to the camera information structure to be filled.

Returns

A [fc2Error](#) indicating the success or failure of the function.

5.3.2.2 [FLYCAPTURE2_C_API fc2Error fc2GetProperty](#) ([fc2Context](#) context, [fc2Property](#) * prop)

Reads the settings for the specified property from the camera.

The property type must be specified in the [fc2Property](#) 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.

See also

[fc2GetPropertyInfo\(\)](#)
[fc2SetProperty\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>prop</i>	Pointer to the Property structure to be filled.

Returns

A fc2Error indicating the success or failure of the function.

5.3.2.3 FLYCAPTURE2_C_API fc2Error fc2GetPropertyInfo (fc2Context *context*, fc2PropertyInfo * *propInfo*)

Retrieves information about the specified camera property.

The property type must be specified in the fc2PropertyInfo structure passed into the function in order for the function to succeed.

See also

[fc2GetProperty\(\)](#)
[fc2SetProperty\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>propInfo</i>	Pointer to the PropertyInfo structure to be filled.

Returns

A fc2Error indicating the success or failure of the function.

5.3.2.4 FLYCAPTURE2_C_API fc2Error fc2SetProperty (fc2Context *context*, fc2Property * *prop*)

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 [fc2GetPropertyInfo\(\)](#) to query which options are available for a specific property.

See also

[fc2GetPropertyInfo\(\)](#)
[fc2GetProperty\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>prop</i>	Pointer to the Property structure to be used.

Returns

A `fc2Error` indicating the success or failure of the function.

5.3.2.5 FLYCAPTURE2_C_API `fc2Error fc2SetPropertyBroadcast (fc2Context context, fc2Property * prop)`

Writes the settings for the specified property to the camera.

The property type must be specified in the Property structure passed into the function in order for the function to succeed. The `absControl` flag controls whether the absolute or integer value is written to the camera.

Parameters

<i>context</i>	The <code>fc2Context</code> to be used.
<i>prop</i>	Pointer to the Property structure to be used.

Returns

A `fc2Error` indicating the success or failure of the function.

5.4 General Purpose Input / Output

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

Functions

- [FLYCAPTURE2_C_API fc2Error fc2GetGPIOPinDirection](#) ([fc2Context](#) context, unsigned int pin, unsigned int *pDirection)
Get the GPIO pin direction for the specified pin.
- [FLYCAPTURE2_C_API fc2Error fc2SetGPIOPinDirection](#) ([fc2Context](#) context, unsigned int pin, unsigned int direction)
Set the GPIO pin direction for the specified pin.
- [FLYCAPTURE2_C_API fc2Error fc2SetGPIOPinDirectionBroadcast](#) ([fc2Context](#) context, unsigned int pin, unsigned int direction)
Set the GPIO pin direction for the specified pin.

5.4.1 Detailed Description

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

5.4.2 Function Documentation

5.4.2.1 [FLYCAPTURE2_C_API fc2Error fc2GetGPIOPinDirection](#) ([fc2Context](#) context, unsigned int pin, unsigned int *pDirection)

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.

See also

[fc2SetGPIOPinDirection\(\)](#)
[fc2SetGPIOPinDirectionBroadcast\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>pin</i>	Pin to get the direction for.
<i>pDirection</i>	Direction of the pin. 0 for input, 1 for output.

Returns

A [fc2Error](#) indicating the success or failure of the function.

5.4.2.2 FLYCAPTURE2_C_API `fc2Error fc2SetGPIOPinDirection (fc2Context context, unsigned int pin, unsigned int direction)`

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.

See also

[fc2GetGPIOPinDirection\(\)](#)
[fc2SetGPIOPinDirectionBroadcast\(\)](#)

Parameters

<i>context</i>	The <code>fc2Context</code> to be used.
<i>pin</i>	Pin to get the direction for.
<i>direction</i>	Direction of the pin. 0 for input, 1 for output.

Returns

A `fc2Error` indicating the success or failure of the function.

5.4.2.3 FLYCAPTURE2_C_API `fc2Error fc2SetGPIOPinDirectionBroadcast (fc2Context context, unsigned int pin, unsigned int direction)`

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.

See also

[fc2GetGPIOPinDirection\(\)](#)

Parameters

<i>context</i>	The <code>fc2Context</code> to be used.
<i>pin</i>	Pin to get the direction for.
<i>direction</i>	Direction of the pin. 0 for input, 1 for output.

Returns

A `fc2Error` indicating the success or failure of the function.

5.5 Trigger

These functions deal with trigger control on the camera.

Functions

- **FLYCAPTURE2_C_API fc2Error fc2GetTriggerModelInfo** (**fc2Context** context, **fc2TriggerModelInfo** *triggerModelInfo)
Retrieve trigger information from the camera.
- **FLYCAPTURE2_C_API fc2Error fc2GetTriggerMode** (**fc2Context** context, **fc2TriggerMode** *triggerMode)
Retrieve current trigger settings from the camera.
- **FLYCAPTURE2_C_API fc2Error fc2SetTriggerMode** (**fc2Context** context, **fc2TriggerMode** *triggerMode)
Set the specified trigger settings to the camera.
- **FLYCAPTURE2_C_API fc2Error fc2SetTriggerModeBroadcast** (**fc2Context** context, **fc2TriggerMode** *triggerMode)
Set the specified trigger settings to the camera.
- **FLYCAPTURE2_C_API fc2Error fc2FireSoftwareTrigger** (**fc2Context** context)
Fire the software trigger according to the DCAM specifications.
- **FLYCAPTURE2_C_API fc2Error fc2FireSoftwareTriggerBroadcast** (**fc2Context** context)
Fire the software trigger according to the DCAM specifications.
- **FLYCAPTURE2_C_API fc2Error fc2GetTriggerDelayInfo** (**fc2Context** context, **fc2TriggerDelayInfo** *triggerDelayInfo)
Retrieve trigger delay information from the camera.
- **FLYCAPTURE2_C_API fc2Error fc2GetTriggerDelay** (**fc2Context** context, **fc2TriggerDelay** *triggerDelay)
Retrieve current trigger delay settings from the camera.
- **FLYCAPTURE2_C_API fc2Error fc2SetTriggerDelay** (**fc2Context** context, **fc2TriggerDelay** *triggerDelay)
Set the specified trigger delay settings to the camera.
- **FLYCAPTURE2_C_API fc2Error fc2SetTriggerDelayBroadcast** (**fc2Context** context, **fc2TriggerDelay** *triggerDelay)
Set the specified trigger delay settings to the camera.

5.5.1 Detailed Description

These functions deal with trigger control on the camera.

5.5.2 Function Documentation

5.5.2.1 FLYCAPTURE2_C_API fc2Error fc2FireSoftwareTrigger (fc2Context context)

Fire the software trigger according to the DCAM specifications.

Parameters

<i>context</i>	The fc2Context to be used.
----------------	----------------------------

Returns

A `fc2Error` indicating the success or failure of the function.

5.5.2.2 FLYCAPTURE2_C_API `fc2Error` `fc2FireSoftwareTriggerBroadcast` (`fc2Context` *context*)

Fire the software trigger according to the DCAM specifications.

Parameters

<i>context</i>	The <code>fc2Context</code> to be used.
----------------	---

Returns

A `fc2Error` indicating the success or failure of the function.

5.5.2.3 FLYCAPTURE2_C_API `fc2Error` `fc2GetTriggerDelay` (`fc2Context` *context*, `fc2TriggerDelay` * *triggerDelay*)

Retrieve current trigger delay settings from the camera.

See also

[fc2GetTriggerModelInfo\(\)](#)
[fc2GetTriggerMode\(\)](#)
[fc2SetTriggerMode\(\)](#)
[fc2GetTriggerDelayInfo\(\)](#)
[fc2SetTriggerDelay\(\)](#)
[fc2SetTriggerDelayBroadcast\(\)](#)

Parameters

<i>context</i>	The <code>fc2Context</code> to be used.
<i>triggerDelay</i>	Structure to receive trigger delay settings.

Returns

A `fc2Error` indicating the success or failure of the function.

5.5.2.4 FLYCAPTURE2_C_API `fc2Error` `fc2GetTriggerDelayInfo` (`fc2Context` *context*, `fc2TriggerDelayInfo` * *triggerDelayInfo*)

Retrieve trigger delay information from the camera.

See also

[fc2GetTriggerModelInfo\(\)](#)
[fc2GetTriggerMode\(\)](#)
[fc2SetTriggerMode\(\)](#)
[fc2GetTriggerDelay\(\)](#)
[fc2SetTriggerDelay\(\)](#)
[fc2SetTriggerDelayBroadcast\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>triggerDelayInfo</i>	Structure to receive trigger delay information.

Returns

A fc2Error indicating the success or failure of the function.

5.5.2.5 FLYCAPTURE2_C_API fc2Error fc2GetTriggerMode (fc2Context *context*, fc2TriggerMode * *triggerMode*)

Retrieve current trigger settings from the camera.

See also

[fc2GetTriggerModelInfo\(\)](#)
[fc2SetTriggerMode\(\)](#)
[fc2SetTriggerModeBroadcast\(\)](#)
[fc2GetTriggerDelayInfo\(\)](#)
[fc2GetTriggerDelay\(\)](#)
[fc2SetTriggerDelay\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>triggerMode</i>	Structure to receive trigger mode settings.

Returns

A fc2Error indicating the success or failure of the function.

5.5.2.6 FLYCAPTURE2_C_API fc2Error fc2GetTriggerModelInfo (fc2Context *context*, fc2TriggerModelInfo * *triggerModelInfo*)

Retrieve trigger information from the camera.

See also

[fc2GetTriggerMode\(\)](#)
[fc2SetTriggerMode\(\)](#)
[fc2SetTriggerModeBroadcast\(\)](#)
[fc2GetTriggerDelayInfo\(\)](#)
[fc2GetTriggerDelay\(\)](#)
[fc2SetTriggerDelay\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>triggerModeInfo</i>	Structure to receive trigger information.

Returns

A fc2Error indicating the success or failure of the function.

5.5.2.7 FLYCAPTURE2_C_API fc2Error fc2SetTriggerDelay (fc2Context context, fc2TriggerDelay * triggerDelay)

Set the specified trigger delay settings to the camera.

See also

[fc2GetTriggerModeInfo\(\)](#)
[fc2GetTriggerMode\(\)](#)
[fc2SetTriggerMode\(\)](#)
[fc2GetTriggerDelayInfo\(\)](#)
[fc2GetTriggerDelay\(\)](#)
[fc2SetTriggerDelayBroadcast\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>triggerDelay</i>	Structure providing trigger delay settings.

Returns

A fc2Error indicating the success or failure of the function.

5.5.2.8 FLYCAPTURE2_C_API fc2Error fc2SetTriggerDelayBroadcast (fc2Context context, fc2TriggerDelay * triggerDelay)

Set the specified trigger delay settings to the camera.

See also

[fc2GetTriggerModelInfo\(\)](#)
[fc2GetTriggerMode\(\)](#)
[fc2SetTriggerMode\(\)](#)
[fc2GetTriggerDelayInfo\(\)](#)
[fc2GetTriggerDelay\(\)](#)
[fc2SetTriggerDelay\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>triggerDelay</i>	Structure providing trigger delay settings.

Returns

A fc2Error indicating the success or failure of the function.

5.5.2.9 FLYCAPTURE2_C_API fc2Error fc2SetTriggerMode (fc2Context *context*, fc2TriggerMode * *triggerMode*)

Set the specified trigger settings to the camera.

See also

[fc2GetTriggerModelInfo\(\)](#)
[fc2GetTriggerMode\(\)](#)
[fc2GetTriggerDelayInfo\(\)](#)
[fc2GetTriggerDelay\(\)](#)
[fc2SetTriggerDelay\(\)](#)
[fc2SetTriggerModeBroadcast\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>triggerMode</i>	Structure providing trigger mode settings.

Returns

A fc2Error indicating the success or failure of the function.

5.5.2.10 FLYCAPTURE2_C_API fc2Error fc2SetTriggerModeBroadcast (fc2Context *context*, fc2TriggerMode * *triggerMode*)

Set the specified trigger settings to the camera.

See also

[fc2GetTriggerModelInfo\(\)](#)
[fc2GetTriggerMode\(\)](#)
[fc2GetTriggerDelayInfo\(\)](#)
[fc2GetTriggerDelay\(\)](#)
[fc2SetTriggerDelay\(\)](#)
[fc2SetTriggerMode\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>triggerMode</i>	Structure providing trigger mode settings.

Returns

A fc2Error indicating the success or failure of the function.

5.6 Strobe

These functions deal with strobe control on the camera.

Functions

- [FLYCAPTURE2_C_API fc2Error fc2GetStrobeInfo](#) ([fc2Context](#) context, [fc2StrobeInfo](#) *strobeInfo)
Retrieve strobe information from the camera.
- [FLYCAPTURE2_C_API fc2Error fc2GetStrobe](#) ([fc2Context](#) context, [fc2StrobeControl](#) *strobeControl)
Retrieve current strobe settings from the camera.
- [FLYCAPTURE2_C_API fc2Error fc2SetStrobe](#) ([fc2Context](#) context, [fc2StrobeControl](#) *strobeControl)
Set current strobe settings to the camera.
- [FLYCAPTURE2_C_API fc2Error fc2SetStrobeBroadcast](#) ([fc2Context](#) context, [fc2StrobeControl](#) *strobeControl)
Set current strobe settings to the camera.

5.6.1 Detailed Description

These functions deal with strobe control on the camera.

5.6.2 Function Documentation

5.6.2.1 [FLYCAPTURE2_C_API fc2Error fc2GetStrobe](#) ([fc2Context](#) context, [fc2StrobeControl](#) * strobeControl)

Retrieve current strobe settings from the camera.

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

See also

[fc2GetStrobeInfo\(\)](#)
[fc2SetStrobe\(\)](#)
[fc2SetStrobeBroadcast\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>strobeControl</i>	Structure to receive strobe settings.

Returns

A [fc2Error](#) indicating the success or failure of the function.

5.6.2.2 [FLYCAPTURE2_C_API fc2Error fc2GetStrobeInfo](#) ([fc2Context](#) context, [fc2StrobeInfo](#) * strobeInfo)

Retrieve strobe information from the camera.

See also

[fc2GetStrobe\(\)](#)
[fc2SetStrobe\(\)](#)
[fc2SetStrobeBroadcast\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>strobeInfo</i>	Structure to receive strobe information.

Returns

A fc2Error indicating the success or failure of the function.

5.6.2.3 FLYCAPTURE2_C_API fc2Error fc2SetStrobe (fc2Context *context*, fc2StrobeControl * *strobeControl*)

Set current strobe settings to the camera.

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

See also

[fc2GetStrobeInfo\(\)](#)
[fc2GetStrobe\(\)](#)
[fc2SetStrobeBroadcast\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>strobeControl</i>	Structure providing strobe settings.

Returns

A fc2Error indicating the success or failure of the function.

5.6.2.4 FLYCAPTURE2_C_API fc2Error fc2SetStrobeBroadcast (fc2Context *context*, fc2StrobeControl * *strobeControl*)

Set current strobe settings to the camera.

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

See also

[fc2GetStrobeInfo\(\)](#)
[fc2GetStrobe\(\)](#)
[fc2SetStrobe\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>strobeControl</i>	Structure providing strobe settings.

Returns

A fc2Error indicating the success or failure of the function.

5.7 Look Up Table

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

Functions

- [FLYCAPTURE2_C_API fc2Error fc2GetLUTInfo](#) ([fc2Context](#) context, [fc2LUTData](#) *pData)
Query if LUT support is available on the camera.
- [FLYCAPTURE2_C_API fc2Error fc2GetLUTBankInfo](#) ([fc2Context](#) context, unsigned int bank, [BOOL](#) *pReadSupported, [BOOL](#) *pWriteSupported)
Query the read/write status of a single LUT bank.
- [FLYCAPTURE2_C_API fc2Error fc2GetActiveLUTBank](#) ([fc2Context](#) context, unsigned int *pActiveBank)
Get the LUT bank that is currently being used.
- [FLYCAPTURE2_C_API fc2Error fc2SetActiveLUTBank](#) ([fc2Context](#) context, unsigned int activeBank)
Set the LUT bank that will be used.
- [FLYCAPTURE2_C_API fc2Error fc2EnableLUT](#) ([fc2Context](#) context, [BOOL](#) on)
Enable or disable LUT functionality on the camera.
- [FLYCAPTURE2_C_API fc2Error fc2GetLUTChannel](#) ([fc2Context](#) context, unsigned int bank, unsigned int channel, unsigned int sizeEntries, unsigned int *pEntries)
Get the LUT channel settings from the camera.
- [FLYCAPTURE2_C_API fc2Error fc2SetLUTChannel](#) ([fc2Context](#) context, unsigned int bank, unsigned int channel, unsigned int sizeEntries, unsigned int *pEntries)
Set the LUT channel settings to the camera.

5.7.1 Detailed Description

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

5.7.2 Function Documentation

5.7.2.1 [FLYCAPTURE2_C_API fc2Error fc2EnableLUT](#) ([fc2Context](#) context, [BOOL](#) on)

Enable or disable LUT functionality on the camera.

See also

[fc2GetLUTInfo\(\)](#)
[fc2GetLUTChannel\(\)](#)
[fc2SetLUTChannel\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>on</i>	Whether to enable or disable LUT.

Returns

A `fc2Error` indicating the success or failure of the function.

5.7.2.2 FLYCAPTURE2_C_API `fc2Error fc2GetActiveLUTBank (fc2Context context, unsigned int * pActiveBank)`

Get the LUT bank that is currently being used.

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

Parameters

<i>context</i>	The <code>fc2Context</code> to be used.
<i>pActiveBank</i>	The currently active bank.

Returns

A `fc2Error` indicating the success or failure of the function.

5.7.2.3 FLYCAPTURE2_C_API `fc2Error fc2GetLUTBankInfo (fc2Context context, unsigned int bank, BOOL * pReadSupported, BOOL * pWriteSupported)`

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

Parameters

<i>context</i>	The <code>fc2Context</code> to be used.
<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

A `fc2Error` indicating the success or failure of the function.

5.7.2.4 FLYCAPTURE2_C_API `fc2Error fc2GetLUTChannel (fc2Context context, unsigned int bank, unsigned int channel, unsigned int sizeEntries, unsigned int * pEntries)`

Get the LUT channel settings from the camera.

See also

[fc2GetLUTInfo\(\)](#)
[fc2EnableLUT\(\)](#)
[fc2SetLUTChannel\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<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.

Returns

A fc2Error indicating the success or failure of the function.

5.7.2.5 FLYCAPTURE2_C_API fc2Error fc2GetLUTInfo (fc2Context *context*, fc2LUTData * *pData*)

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.

See also

[fc2EnableLUT\(\)](#)
[fc2GetLUTChannel\(\)](#)
[fc2SetLUTChannel\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>pData</i>	The LUT structure to be filled.

Returns

A fc2Error indicating the success or failure of the function.

5.7.2.6 FLYCAPTURE2_C_API fc2Error fc2SetActiveLUTBank (fc2Context *context*, unsigned int *activeBank*)

Set the LUT bank that will be used.

Parameters

<i>context</i>	The fc2Context to be used.
<i>activeBank</i>	The bank to be set as active.

Returns

A fc2Error indicating the success or failure of the function.

5.7.2.7 FLYCAPTURE2_C_API `fc2Error fc2SetLUTChannel (fc2Context context, unsigned int bank, unsigned int channel, unsigned int sizeEntries, unsigned int * pEntries)`

Set the LUT channel settings to the camera.

See also

[fc2GetLUTInfo\(\)](#)
[fc2EnableLUT\(\)](#)
[fc2GetLUTChannel\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<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.

Returns

A fc2Error indicating the success or failure of the function.

5.8 Memory Channels

These functions deal with memory channel control on the camera.

Functions

- [FLYCAPTURE2_C_API](#) [fc2Error](#) [fc2GetMemoryChannel](#) ([fc2Context](#) context, unsigned int *pCurrent↔Channel)
Retrieve the current memory channel from the camera.
- [FLYCAPTURE2_C_API](#) [fc2Error](#) [fc2SaveToMemoryChannel](#) ([fc2Context](#) context, unsigned int channel)
Save the current settings to the specified current memory channel.
- [FLYCAPTURE2_C_API](#) [fc2Error](#) [fc2RestoreFromMemoryChannel](#) ([fc2Context](#) context, unsigned int channel)
Restore the specified current memory channel.
- [FLYCAPTURE2_C_API](#) [fc2Error](#) [fc2GetMemoryChannelInfo](#) ([fc2Context](#) context, unsigned int *pNum↔Channels)
Query the camera for memory channel support.
- [FLYCAPTURE2_C_API](#) [fc2Error](#) [fc2GetEmbeddedImageInfo](#) ([fc2Context](#) context, [fc2EmbeddedImageInfo](#) *pInfo)
Get the current status of the embedded image information register, as well as the availability of each embedded property.
- [FLYCAPTURE2_C_API](#) [fc2Error](#) [fc2SetEmbeddedImageInfo](#) ([fc2Context](#) context, [fc2EmbeddedImageInfo](#) *pInfo)
Sets the on/off values of the embedded image information structure to the camera.

5.8.1 Detailed Description

These functions deal with memory channel control on the camera.

5.8.2 Function Documentation

5.8.2.1 [FLYCAPTURE2_C_API](#) [fc2Error](#) [fc2GetEmbeddedImageInfo](#) ([fc2Context](#) context, [fc2EmbeddedImageInfo](#) * pInfo)

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

See also

[fc2SetEmbeddedImageInfo\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>pInfo</i>	Structure to be filled.

Returns

A `fc2Error` indicating the success or failure of the function.

5.8.2.2 FLYCAPTURE2_C_API `fc2Error fc2GetMemoryChannel (fc2Context context, unsigned int * pCurrentChannel)`

Retrieve the current memory channel from the camera.

See also

[fc2SaveToMemoryChannel\(\)](#)
[fc2RestoreFromMemoryChannel\(\)](#)
[fc2GetMemoryChannelInfo\(\)](#)

Parameters

<i>context</i>	The <code>fc2Context</code> to be used.
<i>pCurrentChannel</i>	Current memory channel.

Returns

A `fc2Error` indicating the success or failure of the function.

5.8.2.3 FLYCAPTURE2_C_API `fc2Error fc2GetMemoryChannelInfo (fc2Context context, unsigned int * pNumChannels)`

Query the camera for memory channel support.

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

See also

[fc2GetMemoryChannel\(\)](#)
[fc2SaveToMemoryChannel\(\)](#)
[fc2RestoreFromMemoryChannel\(\)](#)

Parameters

<i>context</i>	The <code>fc2Context</code> to be used.
<i>pNumChannels</i>	Number of memory channels supported.

Returns

A `fc2Error` indicating the success or failure of the function.

5.8.2.4 FLYCAPTURE2_C_API `fc2Error fc2RestoreFromMemoryChannel (fc2Context context, unsigned int channel)`

Restore the specified current memory channel.

See also

[fc2GetMemoryChannel\(\)](#)
[fc2SaveToMemoryChannel\(\)](#)
[fc2GetMemoryChannelInfo\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>channel</i>	Memory channel to restore from.

Returns

A fc2Error indicating the success or failure of the function.

5.8.2.5 FLYCAPTURE2_C_API fc2Error fc2SaveToMemoryChannel (fc2Context *context*, unsigned int *channel*)

Save the current settings to the specified current memory channel.

See also

[fc2GetMemoryChannel\(\)](#)
[fc2RestoreFromMemoryChannel\(\)](#)
[fc2GetMemoryChannelInfo\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>channel</i>	Memory channel to save to.

Returns

A fc2Error indicating the success or failure of the function.

5.8.2.6 FLYCAPTURE2_C_API fc2Error fc2SetEmbeddedImageInfo (fc2Context *context*, fc2EmbeddedImageInfo * *pInfo*)

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

See also

[fc2GetEmbeddedImageInfo\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>pInfo</i>	Structure to be used.

Returns

A `fc2Error` indicating the success or failure of the function.

5.9 Register Operation

These functions deal with register operation on the camera.

Functions

- `FLYCAPTURE2_C_API fc2Error fc2WriteRegister` (`fc2Context` context, unsigned int address, unsigned int value)
Write to the specified register on the camera.
- `FLYCAPTURE2_C_API fc2Error fc2ReadRegister` (`fc2Context` context, unsigned int address, unsigned int *pValue)
Read the specified register from the camera.
- `FLYCAPTURE2_C_API fc2Error fc2WriteRegisterBroadcast` (`fc2Context` context, unsigned int address, unsigned int value)
Write to the specified register on the camera with broadcast.
- `FLYCAPTURE2_C_API fc2Error fc2WriteRegisterBlock` (`fc2Context` context, unsigned short addressHigh, unsigned int addressLow, const unsigned int *pBuffer, unsigned int length)
Write to the specified register block on the camera.
- `FLYCAPTURE2_C_API fc2Error fc2ReadRegisterBlock` (`fc2Context` context, unsigned short addressHigh, unsigned int addressLow, unsigned int *pBuffer, unsigned int length)
Read the specified register block on the camera.
- `FLYCAPTURE2_C_API const char * fc2GetRegisterString` (unsigned int registerVal)
Returns a text representation of the register value.

5.9.1 Detailed Description

These functions deal with register operation on the camera.

5.9.2 Function Documentation

5.9.2.1 `FLYCAPTURE2_C_API const char* fc2GetRegisterString (unsigned int registerVal)`

Returns a text representation of the register value.

Parameters

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

Returns

A `fc2Error` indicating the success or failure of the function.

5.9.2.2 `FLYCAPTURE2_C_API fc2Error fc2ReadRegister (fc2Context context, unsigned int address, unsigned int * pValue)`

Read the specified register from the camera.

See also

[fc2WriteRegister\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>address</i>	DCAM address to be read from.
<i>pValue</i>	The value that is read.

Returns

A fc2Error indicating the success or failure of the function.

5.9.2.3 FLYCAPTURE2_C_API fc2Error fc2ReadRegisterBlock (fc2Context context, unsigned short addressHigh, unsigned int addressLow, unsigned int * pBuffer, unsigned int length)

Write to the specified register block on the camera.

See also

[fc2WriteRegisterBlock\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<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.

Returns

A fc2Error indicating the success or failure of the function.

5.9.2.4 FLYCAPTURE2_C_API fc2Error fc2WriteRegister (fc2Context context, unsigned int address, unsigned int value)

Write to the specified register on the camera.

See also

[fc2ReadRegister\(\)](#)

Parameters

<i>context</i>	The fc2Context to be used.
<i>address</i>	DCAM address to be written to.
<i>value</i>	The value to be written.

Returns

A `fc2Error` indicating the success or failure of the function.

5.9.2.5 FLYCAPTURE2_C_API `fc2Error fc2WriteRegisterBlock (fc2Context context, unsigned short addressHigh, unsigned int addressLow, const unsigned int * pBuffer, unsigned int length)`

Write to the specified register block on the camera.

See also

[fc2ReadRegisterBlock\(\)](#)

Parameters

<i>context</i>	The <code>fc2Context</code> to be used.
<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.

Returns

A `fc2Error` indicating the success or failure of the function.

5.9.2.6 FLYCAPTURE2_C_API `fc2Error fc2WriteRegisterBroadcast (fc2Context context, unsigned int address, unsigned int value)`

Write to the specified register on the camera with broadcast.

See also

[fc2ReadRegisterBlock\(\)](#)

Parameters

<i>context</i>	The <code>fc2Context</code> to be used.
<i>address</i>	DCAM address to be written to.
<i>value</i>	The value to be written.

Returns

A `fc2Error` indicating the success or failure of the function.

5.10 DCAM Formats

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

Functions

- **FLYCAPTURE2_C_API fc2Error fc2GetVideoModeAndFrameRateInfo** (fc2Context context, fc2VideoMode videoMode, fc2FrameRate frameRate, BOOL *pSupported)
Query the camera to determine if the specified video mode and frame rate is supported.
- **FLYCAPTURE2_C_API fc2Error fc2GetVideoModeAndFrameRate** (fc2Context context, fc2VideoMode *videoMode, fc2FrameRate *frameRate)
Get the current video mode and frame rate from the camera.
- **FLYCAPTURE2_C_API fc2Error fc2SetVideoModeAndFrameRate** (fc2Context context, fc2VideoMode videoMode, fc2FrameRate frameRate)
Set the specified video mode and frame rate to the camera.

5.10.1 Detailed Description

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

5.10.2 Function Documentation

5.10.2.1 FLYCAPTURE2_C_API fc2Error fc2GetVideoModeAndFrameRate (fc2Context context, fc2VideoMode * videoMode, fc2FrameRate * frameRate)

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>context</i>	The fc2Context to be used.
<i>videoMode</i>	Current video mode.
<i>frameRate</i>	Current frame rate.

Returns

A fc2Error indicating the success or failure of the function.

5.10.2.2 FLYCAPTURE2_C_API fc2Error fc2GetVideoModeAndFrameRateInfo (fc2Context context, fc2VideoMode videoMode, fc2FrameRate frameRate, BOOL * pSupported)

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

Parameters

<i>context</i>	The fc2Context to be used.
<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.

Returns

A fc2Error indicating the success or failure of the function.

5.10.2.3 FLYCAPTURE2_C_API fc2Error fc2SetVideoModeAndFrameRate (fc2Context *context*, fc2VideoMode *videoMode*, fc2FrameRate *frameRate*)

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>context</i>	The fc2Context to be used.
<i>videoMode</i>	Video mode to set to camera.
<i>frameRate</i>	Frame rate to set to camera.

Returns

A fc2Error indicating the success or failure of the function.

5.11 Format7

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

Functions

- **FLYCAPTURE2_C_API** **fc2Error** **fc2GetFormat7Info** (**fc2Context** context, **fc2Format7Info** *info, **BOOL** *pSupported)
Retrieve the availability of Format7 custom image mode and the camera capabilities for the specified Format7 mode.
- **FLYCAPTURE2_C_API** **fc2Error** **fc2ValidateFormat7Settings** (**fc2Context** context, **fc2Format7ImageSettings** *imageSettings, **BOOL** *settingsAreValid, **fc2Format7PacketInfo** *packetInfo)
Validates Format7ImageSettings structure and returns valid packet size information if the image settings are valid.
- **FLYCAPTURE2_C_API** **fc2Error** **fc2GetFormat7Configuration** (**fc2Context** context, **fc2Format7ImageSettings** *imageSettings, unsigned int *packetSize, float *percentage)
Get the current Format7 configuration from the camera.
- **FLYCAPTURE2_C_API** **fc2Error** **fc2SetFormat7ConfigurationPacket** (**fc2Context** context, **fc2Format7ImageSettings** *imageSettings, unsigned int packetSize)
Set the current Format7 configuration to the camera.
- **FLYCAPTURE2_C_API** **fc2Error** **fc2SetFormat7Configuration** (**fc2Context** context, **fc2Format7ImageSettings** *imageSettings, float percentSpeed)
Set the current Format7 configuration to the camera.

5.11.1 Detailed Description

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

5.11.2 Function Documentation

5.11.2.1 **FLYCAPTURE2_C_API** **fc2Error** **fc2GetFormat7Configuration** (**fc2Context** context, **fc2Format7ImageSettings** * imageSettings, unsigned int * packetSize, float * percentage)

Get the current Format7 configuration from the camera.

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

Parameters

<i>context</i>	The fc2Context to be used.
<i>imageSettings</i>	Current image settings.
<i>packetSize</i>	Current packet size.
<i>percentage</i>	Current packet size as a percentage.

Returns

A fc2Error indicating the success or failure of the function.

5.11.2.2 FLYCAPTURE2_C_API `fc2Error fc2GetFormat7Info (fc2Context context, fc2Format7Info * info, BOOL * pSupported)`

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>context</i>	The fc2Context to be used.
<i>info</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.

Returns

A fc2Error indicating the success or failure of the function.

5.11.2.3 FLYCAPTURE2_C_API `fc2Error fc2SetFormat7Configuration (fc2Context context, fc2Format7ImageSettings * imageSettings, float percentSpeed)`

Set the current Format7 configuration to the camera.

Parameters

<i>context</i>	The fc2Context to be used.
<i>imageSettings</i>	Image settings to be written to the camera.
<i>percentSpeed</i>	Packet size as a percentage to be written to the camera.

Returns

A fc2Error indicating the success or failure of the function.

5.11.2.4 FLYCAPTURE2_C_API `fc2Error fc2SetFormat7ConfigurationPacket (fc2Context context, fc2Format7ImageSettings * imageSettings, unsigned int packetSize)`

Set the current Format7 configuration to the camera.

Parameters

<i>context</i>	The fc2Context to be used.
<i>imageSettings</i>	Image settings to be written to the camera.
<i>packetSize</i>	Packet size to be written to the camera.

Returns

A fc2Error indicating the success or failure of the function.

5.11.2.5 **FLYCAPTURE2_C_API** **fc2Error** **fc2ValidateFormat7Settings** (**fc2Context** *context*,
fc2Format7ImageSettings * *imageSettings*, **BOOL** * *settingsAreValid*, **fc2Format7PacketInfo** * *packetInfo*)

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>context</i>	The fc2Context to be used.
<i>imageSettings</i>	Structure containing the image settings.
<i>settingsAreValid</i>	Whether the settings are valid.
<i>packetInfo</i>	Packet size information that can be used to determine a valid packet size.

Returns

A fc2Error indicating the success or failure of the function.

5.12 GVCP Register Operation

These functions deal with GVCP register operation on the camera.

Functions

- [FLYCAPTURE2_C_API fc2Error fc2WriteGVCPRegister](#) ([fc2Context](#) context, unsigned int address, unsigned int value)
Write a GVCP register.
- [FLYCAPTURE2_C_API fc2Error fc2WriteGVCPRegisterBroadcast](#) ([fc2Context](#) context, unsigned int address, unsigned int value)
Write a GVCP register with broadcast.
- [FLYCAPTURE2_C_API fc2Error fc2ReadGVCPRegister](#) ([fc2Context](#) context, unsigned int address, unsigned int *pValue)
Read a GVCP register.
- [FLYCAPTURE2_C_API fc2Error fc2WriteGVCPRegisterBlock](#) ([fc2Context](#) context, unsigned int address, const unsigned int *pBuffer, unsigned int length)
Write a GVCP register block.
- [FLYCAPTURE2_C_API fc2Error fc2ReadGVCPRegisterBlock](#) ([fc2Context](#) context, unsigned int address, unsigned int *pBuffer, unsigned int length)
Read a GVCP register block.
- [FLYCAPTURE2_C_API fc2Error fc2WriteGVCPMemory](#) ([fc2Context](#) context, unsigned int address, const unsigned char *pBuffer, unsigned int length)
Write a GVCP memory block.
- [FLYCAPTURE2_C_API fc2Error fc2ReadGVCPMemory](#) ([fc2Context](#) context, unsigned int address, unsigned char *pBuffer, unsigned int length)
Read a GVCP memory block.

5.12.1 Detailed Description

These functions deal with GVCP register operation on the camera.

5.12.2 Function Documentation

5.12.2.1 [FLYCAPTURE2_C_API fc2Error fc2ReadGVCPMemory](#) ([fc2Context](#) context, unsigned int address, unsigned char * pBuffer, unsigned int length)

Read a GVCP memory block.

Parameters

<i>context</i>	The fc2Context to be used.
<i>address</i>	GVCP address to be read from.
<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.

5.12.2.2 FLYCAPTURE2_C_API **fc2Error** **fc2ReadGVCPRegister** (**fc2Context** *context*, unsigned int *address*, unsigned int * *pValue*)

Read a GVCP register.

Parameters

<i>context</i>	The fc2Context to be used.
<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.

5.12.2.3 FLYCAPTURE2_C_API **fc2Error** **fc2ReadGVCPRegisterBlock** (**fc2Context** *context*, unsigned int *address*, unsigned int * *pBuffer*, unsigned int *length*)

Read a GVCP register block.

Parameters

<i>context</i>	The fc2Context to be used.
<i>address</i>	GVCP address to be read from.
<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.

5.12.2.4 FLYCAPTURE2_C_API **fc2Error** **fc2WriteGVCPMemory** (**fc2Context** *context*, unsigned int *address*, const unsigned char * *pBuffer*, unsigned int *length*)

Write a GVCP memory block.

Parameters

<i>context</i>	The fc2Context to be used.
<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.

5.12.2.5 FLYCAPTURE2_C_API fc2Error fc2WriteGVCPRegister (fc2Context context, unsigned int address, unsigned int value)

Write a GVCP register.

Parameters

<i>context</i>	The fc2Context to be used.
<i>address</i>	GVCP address to be written to.
<i>value</i>	The value to be written.

Returns

An Error indicating the success or failure of the function.

5.12.2.6 FLYCAPTURE2_C_API fc2Error fc2WriteGVCPRegisterBlock (fc2Context context, unsigned int address, const unsigned int * pBuffer, unsigned int length)

Write a GVCP register block.

Parameters

<i>context</i>	The fc2Context to be used.
<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.

5.12.2.7 FLYCAPTURE2_C_API fc2Error fc2WriteGVCPRegisterBroadcast (fc2Context context, unsigned int address, unsigned int value)

Write a GVCP register with broadcast.

Parameters

<i>context</i>	The fc2Context to be used.
<i>address</i>	GVCP address to be written to.
<i>value</i>	The value to be written.

Returns

An Error indicating the success or failure of the function.

5.13 GigE property manipulation

These functions deal with GigE properties.

Functions

- **FLYCAPTURE2_C_API fc2Error fc2GetGigEProperty** (**fc2Context** context, **fc2GigEProperty** *pGigEProp)
Get the specified GigEProperty.
- **FLYCAPTURE2_C_API fc2Error fc2SetGigEProperty** (**fc2Context** context, const **fc2GigEProperty** *pGigEProp)
Set the specified GigEProperty.
- **FLYCAPTURE2_C_API fc2Error fc2DiscoverGigEPacketSize** (**fc2Context** context, unsigned int *packetSize)
Discover the largest packet size that works for the network link between the PC and the camera.

5.13.1 Detailed Description

These functions deal with GigE properties.

5.13.2 Function Documentation

5.13.2.1 FLYCAPTURE2_C_API fc2Error fc2DiscoverGigEPacketSize (fc2Context context, unsigned int * packetSize)

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>context</i>	The fc2Context to be used.
<i>packetSize</i>	The maximum packet size supported by the link.

Returns

An Error indicating the success or failure of the function.

5.13.2.2 FLYCAPTURE2_C_API fc2Error fc2GetGigEProperty (fc2Context context, fc2GigEProperty * pGigEProp)

Get the specified GigEProperty.

The GigEPropertyType field must be set in order for this function to succeed.

Parameters

<i>context</i>	The fc2Context to be used.
<i>pGigEProp</i>	The GigE property to get.

Returns

An Error indicating the success or failure of the function.

5.13.2.3 FLYCAPTURE2_C_API fc2Error fc2SetGigEProperty (fc2Context *context*, const fc2GigEProperty * *pGigEProp*)

Set the specified GigEProperty.

The GigEPropertyType field must be set in order for this function to succeed.

Parameters

<i>context</i>	The fc2Context to be used.
<i>pGigEProp</i>	The GigE property to set.

Returns

An Error indicating the success or failure of the function.

5.14 GigE image settings

These functions deal with GigE image setting.

Functions

- **FLYCAPTURE2_C_API fc2Error fc2QueryGigElmagingMode** (**fc2Context** context, **fc2Mode** mode, **BOOL** *isSupported)
Check if the particular imaging mode is supported by the camera.
- **FLYCAPTURE2_C_API fc2Error fc2GetGigElmagingMode** (**fc2Context** context, **fc2Mode** *mode)
Get the current imaging mode on the camera.
- **FLYCAPTURE2_C_API fc2Error fc2SetGigElmagingMode** (**fc2Context** context, **fc2Mode** mode)
Set the current imaging mode to the camera.
- **FLYCAPTURE2_C_API fc2Error fc2GetGigElmageSettingsInfo** (**fc2Context** context, **fc2GigElmageSettingsInfo** *pInfo)
Get information about the image settings possible on the camera.
- **FLYCAPTURE2_C_API fc2Error fc2GetGigElmageSettings** (**fc2Context** context, **fc2GigElmageSettings** *pImageSettings)
Get the current image settings on the camera.
- **FLYCAPTURE2_C_API fc2Error fc2SetGigElmageSettings** (**fc2Context** context, const **fc2GigElmageSettings** *pImageSettings)
Set the image settings specified to the camera.

5.14.1 Detailed Description

These functions deal with GigE image setting.

5.14.2 Function Documentation

5.14.2.1 FLYCAPTURE2_C_API fc2Error fc2GetGigElmageSettings (**fc2Context** context, **fc2GigElmageSettings** *pImageSettings)

Get the current image settings on the camera.

Parameters

<i>context</i>	The fc2Context to be used.
<i>pImageSettings</i>	Current image settings on camera.

Returns

An Error indicating the success or failure of the function.

5.14.2.2 FLYCAPTURE2_C_API fc2Error fc2GetGigElmageSettingsInfo (**fc2Context** context, **fc2GigElmageSettingsInfo** * pInfo)

Get information about the image settings possible on the camera.

Parameters

<i>context</i>	The fc2Context to be used.
<i>plInfo</i>	Image settings information.

Returns

An Error indicating the success or failure of the function.

5.14.2.3 FLYCAPTURE2_C_API fc2Error fc2GetGigElmagingMode (fc2Context *context*, fc2Mode * *mode*)

Get the current imaging mode on the camera.

Parameters

<i>context</i>	The fc2Context to be used.
<i>mode</i>	Current imaging mode on the camera.

Returns

An Error indicating the success or failure of the function.

5.14.2.4 FLYCAPTURE2_C_API fc2Error fc2QueryGigElmagingMode (fc2Context *context*, fc2Mode *mode*, BOOL * *isSupported*)

Check if the particular imaging mode is supported by the camera.

Parameters

<i>context</i>	The fc2Context to be used.
<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.

5.14.2.5 FLYCAPTURE2_C_API fc2Error fc2SetGigElmageSettings (fc2Context *context*, const fc2GigElmageSettings * *plImageSettings*)

Set the image settings specified to the camera.

Parameters

<i>context</i>	The fc2Context to be used.
<i>plImageSettings</i>	Image settings to set to camera.

Returns

An Error indicating the success or failure of the function.

5.14.2.6 FLYCAPTURE2_C_API fc2Error fc2SetGigElmagingMode (fc2Context *context*, fc2Mode *mode*)

Set the current imaging mode to the camera.

This should only be done when the camera is not streaming images.

Parameters

<i>context</i>	The fc2Context to be used.
<i>mode</i>	Imaging mode to set to the camera.

Returns

An Error indicating the success or failure of the function.

5.15 GigE image binning settings

These functions deal with GigE image binning settings.

Functions

- **FLYCAPTURE2_C_API** **fc2Error** **fc2GetGigEImageBinningSettings** (**fc2Context** context, unsigned int *horz↔BinningValue, unsigned int *vertBinningValue)
Get the current binning settings on the camera.
- **FLYCAPTURE2_C_API** **fc2Error** **fc2SetGigEImageBinningSettings** (**fc2Context** context, unsigned int horz↔BinningValue, unsigned int vertBinningValue)
Set the specified binning values to the camera.

5.15.1 Detailed Description

These functions deal with GigE image binning settings.

5.15.2 Function Documentation

5.15.2.1 **FLYCAPTURE2_C_API** **fc2Error** **fc2GetGigEImageBinningSettings** (**fc2Context** context, unsigned int * horzBinningValue, unsigned int * vertBinningValue)

Get the current binning settings on the camera.

Parameters

<i>context</i>	The fc2Context to be used.
<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.

5.15.2.2 **FLYCAPTURE2_C_API** **fc2Error** **fc2SetGigEImageBinningSettings** (**fc2Context** context, unsigned int horzBinningValue, unsigned int vertBinningValue)

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>context</i>	The fc2Context to be used.
<i>horzBinningValue</i>	Horizontal binning value.
<i>vertBinningValue</i>	Vertical binning value.

Returns

An Error indicating the success or failure of the function.

5.16 GigE image stream configuration

These functions deal with GigE image stream configuration.

Functions

- **FLYCAPTURE2_C_API** **fc2Error** **fc2GetNumStreamChannels** (**fc2Context** context, unsigned int *numChannels)
Get the number of stream channels present on the camera.
- **FLYCAPTURE2_C_API** **fc2Error** **fc2GetGigEStreamChannelInfo** (**fc2Context** context, unsigned int channel, **fc2GigEStreamChannel** *pChannel)
Get the stream channel information for the specified channel.
- **FLYCAPTURE2_C_API** **fc2Error** **fc2SetGigEStreamChannelInfo** (**fc2Context** context, unsigned int channel, **fc2GigEStreamChannel** *pChannel)
Set the stream channel information for the specified channel.
- **FLYCAPTURE2_C_API** **fc2Error** **fc2GetGigEConfig** (**fc2Context** context, **fc2GigEConfig** *pConfig)
Get the current gige config on the camera.
- **FLYCAPTURE2_C_API** **fc2Error** **fc2SetGigEConfig** (**fc2Context** context, const **fc2GigEConfig** *pConfig)
Set the gige config specified to the camera.

5.16.1 Detailed Description

These functions deal with GigE image stream configuration.

5.16.2 Function Documentation

5.16.2.1 **FLYCAPTURE2_C_API** **fc2Error** **fc2GetGigEConfig** (**fc2Context** context, **fc2GigEConfig** * pConfig)

Get the current gige config on the camera.

Parameters

<i>context</i>	The fc2Context to be used.
<i>pGigEConfig</i>	Current configuration on camera.

Returns

An Error indicating the success or failure of the function.

5.16.2.2 **FLYCAPTURE2_C_API** **fc2Error** **fc2GetGigEStreamChannelInfo** (**fc2Context** context, unsigned int channel, **fc2GigEStreamChannel** * pChannel)

Get the stream channel information for the specified channel.

Parameters

<i>context</i>	The fc2Context to be used.
<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.

5.16.2.3 FLYCAPTURE2_C_API fc2Error fc2GetNumStreamChannels (fc2Context *context*, unsigned int * *numChannels*)

Get the number of stream channels present on the camera.

Parameters

<i>context</i>	The fc2Context to be used.
<i>numChannels</i>	Number of stream channels present.

Returns

An Error indicating the success or failure of the function.

5.16.2.4 FLYCAPTURE2_C_API fc2Error fc2SetGigEConfig (fc2Context *context*, const fc2GigEConfig * *pConfig*)

Set the gige config specified to the camera.

Parameters

<i>context</i>	The fc2Context to be used.
<i>pGigEConfig</i>	configuration to set to camera.

Returns

An Error indicating the success or failure of the function.

5.16.2.5 FLYCAPTURE2_C_API fc2Error fc2SetGigEStreamChannelInfo (fc2Context *context*, unsigned int *channel*, fc2GigEStreamChannel * *pChannel*)

Set the stream channel information for the specified channel.

Note that the source UDP port of the stream channel is read-only.

Parameters

<i>context</i>	The fc2Context to be used.
<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.

5.17 Image Operation

The Image operations are used to retrieve images from a camera, convert between multiple pixel formats and save images to disk.

Functions

- [FLYCAPTURE2_C_API](#) [fc2Error](#) [fc2SetDefaultColorProcessing](#) ([fc2ColorProcessingAlgorithm](#) default↵
Method)
Set the default color processing algorithm.
- [FLYCAPTURE2_C_API](#) [fc2Error](#) [fc2GetDefaultColorProcessing](#) ([fc2ColorProcessingAlgorithm](#) *pDefault↵
Method)
Get the default color processing algorithm.
- [FLYCAPTURE2_C_API](#) [fc2Error](#) [fc2SetDefaultOutputFormat](#) ([fc2PixelFormat](#) format)
Set the default output pixel format.
- [FLYCAPTURE2_C_API](#) [fc2Error](#) [fc2GetDefaultOutputFormat](#) ([fc2PixelFormat](#) *pFormat)
Get the default output pixel format.
- [FLYCAPTURE2_C_API](#) [fc2Error](#) [fc2DetermineBitsPerPixel](#) ([fc2PixelFormat](#) format, unsigned int *pBitsPer↵
Pixel)
Calculate the bits per pixel for the specified pixel format.
- [FLYCAPTURE2_C_API](#) [fc2Error](#) [fc2CreateImage](#) ([fc2Image](#) *pImage)
Create a [fc2Image](#).
- [FLYCAPTURE2_C_API](#) [fc2Error](#) [fc2DestroyImage](#) ([fc2Image](#) *image)
Destroy the [fc2Image](#).
- [FLYCAPTURE2_C_API](#) [fc2Error](#) [fc2SetImageDimensions](#) ([fc2Image](#) *pImage, unsigned int rows, unsigned
int cols, unsigned int stride, [fc2PixelFormat](#) pixelFormat, [fc2BayerTileFormat](#) bayerFormat)
Sets the dimensions of the image object.
- [FLYCAPTURE2_C_API](#) [fc2Error](#) [fc2SetImageData](#) ([fc2Image](#) *pImage, const unsigned char *pData, un-
signed int dataSize)
Set the data of the Image object.
- [FLYCAPTURE2_C_API](#) [fc2Error](#) [fc2GetImageData](#) ([fc2Image](#) *pImage, unsigned char **ppData)
Get a pointer to the data associated with the image.
- [FLYCAPTURE2_C_API](#) [fc2TimeStamp](#) [fc2GetImageTimeStamp](#) ([fc2Image](#) *pImage)
Get the timestamp data associated with the image.
- [FLYCAPTURE2_C_API](#) [fc2Error](#) [fc2SaveImage](#) ([fc2Image](#) *pImage, const char *pFilename, [fc2ImageFile↵
Format](#) format)
Save the image to the specified file name with the file format specified.
- [FLYCAPTURE2_C_API](#) [fc2Error](#) [fc2SaveImageWithOptions](#) ([fc2Image](#) *pImage, const char *pFilename,
[fc2ImageFileFormat](#) format, void *pOption)
Save the image to the specified file name with the file format specified.
- [FLYCAPTURE2_C_API](#) [fc2Error](#) [fc2ConvertImage](#) ([fc2Image](#) *pImageIn, [fc2Image](#) *pImageOut)
- [FLYCAPTURE2_C_API](#) [fc2Error](#) [fc2ConvertImageTo](#) ([fc2PixelFormat](#) format, [fc2Image](#) *pImageIn, [fc2Image](#)
*pImageOut)
Converts the current image buffer to the specified output format and stores the result in the specified image.
- [FLYCAPTURE2_C_API](#) [fc2Error](#) [fc2CalculateImageStatistics](#) ([fc2Image](#) *pImage, [fc2ImageStatisticsContext](#)
*pImageStatisticsContext)
Calculate statistics associated with the image.

5.17.1 Detailed Description

The Image operations are used to retrieve images from a camera, convert between multiple pixel formats and save images to disk.

Operations on images are not guaranteed to be thread safe. It is recommended that operations on images be protected by thread synchronization constructs such as mutexes.

5.17.2 Function Documentation

5.17.2.1 FLYCAPTURE2_C_API `fc2Error fc2CalculateImageStatistics (fc2Image * plmage, fc2ImageStatisticsContext * plmageStatisticsContext)`

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>plmage</i>	The fc2Image to be used.
<i>plmageStatisticsContext</i>	The <code>fc2ImageStatisticsContext</code> to hold the statistics.

Returns

A `fc2Error` indicating the success or failure of the function.

5.17.2.2 FLYCAPTURE2_C_API `fc2Error fc2ConvertImage (fc2Image * plmageIn, fc2Image * plmageOut)`

Parameters

<i>plmageIn</i>	
<i>plmageOut</i>	

Returns

A `fc2Error` indicating the success or failure of the function.

5.17.2.3 FLYCAPTURE2_C_API `fc2Error fc2ConvertImageTo (fc2PixelFormat format, fc2Image * plmageIn, fc2Image * plmageOut)`

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>pImageIn</i>	Input image.
<i>pImageOut</i>	Output image.

Returns

A `fc2Error` indicating the success or failure of the function.

5.17.2.4 FLYCAPTURE2_C_API `fc2Error fc2CreateImage (fc2Image * pImage)`

Create a [fc2Image](#).

If externally allocated memory is to be used for the converted image, simply assigning the `pData` member of the [fc2Image](#) structure is insufficient. [fc2SetImageData\(\)](#) should be called in order to populate the [fc2Image](#) structure correctly.

See also

[fc2SetImageData\(\)](#)

Parameters

<i>pImage</i>	Pointer to image to be created.
---------------	---------------------------------

Returns

A `fc2Error` indicating the success or failure of the function.

5.17.2.5 FLYCAPTURE2_C_API `fc2Error fc2DestroyImage (fc2Image * image)`

Destroy the [fc2Image](#).

Parameters

<i>image</i>	Pointer to image to be destroyed.
--------------	-----------------------------------

Returns

A `fc2Error` indicating the success or failure of the function.

5.17.2.6 FLYCAPTURE2_C_API `fc2Error fc2DetermineBitsPerPixel (fc2PixelFormat format, unsigned int * pBitsPerPixel)`

Calculate the bits per pixel for the specified pixel format.

Parameters

<i>format</i>	The pixel format.
<i>pBitsPerPixel</i>	The bits per pixel.

Returns

A `fc2Error` indicating the success or failure of the function.

5.17.2.7 FLYCAPTURE2_C_API `fc2Error fc2GetDefaultColorProcessing (fc2ColorProcessingAlgorithm * pDefaultMethod)`

Get the default color processing algorithm.

Parameters

<i>pDefaultMethod</i>	The default color processing algorithm.
-----------------------	---

Returns

A `fc2Error` indicating the success or failure of the function.

5.17.2.8 FLYCAPTURE2_C_API `fc2Error fc2GetDefaultOutputFormat (fc2PixelFormat * pFormat)`

Get the default output pixel format.

Parameters

<i>pFormat</i>	The default pixel format.
----------------	---------------------------

Returns

A `fc2Error` indicating the success or failure of the function.

5.17.2.9 FLYCAPTURE2_C_API `fc2Error fc2GetImageData (fc2Image * pImage, unsigned char ** ppData)`

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 [fc2RetrieveBuffer\(\)](#).

Parameters

<i>pImage</i>	The fc2Image to be used.
<i>ppData</i>	A pointer to the image data.

Returns

A `fc2Error` indicating the success or failure of the function.

5.17.2.10 FLYCAPTURE2_C_API `fc2TimeStamp fc2GetImageTimeStamp (fc2Image * pImage)`

Get the timestamp data associated with the image.

Parameters

<i>pImage</i>	The fc2Image to be used.
---------------	--

Returns

Timestamp data associated with the image.

5.17.2.11 FLYCAPTURE2_C_API `fc2Error fc2SaveImage (fc2Image * pImage, const char * pFilename, fc2ImageFileFormat format)`

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

Parameters

<i>pImage</i>	The fc2Image to be used.
<i>pFilename</i>	Filename to save image with.
<i>format</i>	File format to save in.

Returns

A `fc2Error` indicating the success or failure of the function.

5.17.2.12 FLYCAPTURE2_C_API `fc2Error fc2SaveImageWithOptions (fc2Image * pImage, const char * pFilename, fc2ImageFileFormat format, void * pOption)`

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

Parameters

<i>pImage</i>	The fc2Image to be used.
<i>pFilename</i>	Filename to save image with.
<i>format</i>	File format to save in.
<i>pOption</i>	Options for saving image.

Returns

A `fc2Error` indicating the success or failure of the function.

5.17.2.13 FLYCAPTURE2_C_API `fc2Error fc2SetDefaultColorProcessing (fc2ColorProcessingAlgorithm defaultMethod)`

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.
----------------------	--

Returns

A `fc2Error` indicating the success or failure of the function.

5.17.2.14 FLYCAPTURE2_C_API `fc2Error fc2SetDefaultOutputFormat (fc2PixelFormat format)`

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.
---------------	---------------------------------

Returns

A `fc2Error` indicating the success or failure of the function.

5.17.2.15 FLYCAPTURE2_C_API `fc2Error fc2SetImageData (fc2Image * pImage, const unsigned char * pData, unsigned int dataSize)`

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>pImage</i>	The fc2Image to be used.
<i>pData</i>	Pointer to the image buffer.
<i>dataSize</i>	Size of the image buffer.

Returns

A `fc2Error` indicating the success or failure of the function.

5.17.2.16 FLYCAPTURE2_C_API `fc2Error fc2SetImageDimensions (fc2Image * pImage, unsigned int rows, unsigned int cols, unsigned int stride, fc2PixelFormat pixelFormat, fc2BayerTileFormat bayerFormat)`

Sets the dimensions of the image object.

Parameters

<i>pImage</i>	The fc2Image to be used.
<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.

Returns

A `fc2Error` indicating the success or failure of the function.

5.18 Image Statistics Operation

The Image Statistics operation provides the functionality for the user to collect image channel statistics.

Functions

- `FLYCAPTURE2_C_API fc2Error fc2CreateImageStatistics (fc2ImageStatisticsContext *pImageStatisticsContext)`
Create a statistics context.
- `FLYCAPTURE2_C_API fc2Error fc2DestroyImageStatistics (fc2ImageStatisticsContext imageStatisticsContext)`
Destroy a statistics context.
- `FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableAll (fc2ImageStatisticsContext imageStatisticsContext)`
Enable all channels.
- `FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsDisableAll (fc2ImageStatisticsContext imageStatisticsContext)`
Disable all channels.
- `FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableGreyOnly (fc2ImageStatisticsContext imageStatisticsContext)`
Enable only the grey channel.
- `FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableRGBOnly (fc2ImageStatisticsContext imageStatisticsContext)`
Enable only the RGB channels.
- `FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableHSLOnly (fc2ImageStatisticsContext imageStatisticsContext)`
Enable only the HSL channels.
- `FLYCAPTURE2_C_API fc2Error fc2GetChannelStatus (fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, BOOL *pEnabled)`
Get the status of a statistics channel.
- `FLYCAPTURE2_C_API fc2Error fc2SetChannelStatus (fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, BOOL enabled)`
Set the status of a statistics channel.
- `FLYCAPTURE2_C_API fc2Error fc2GetChannelRange (fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, unsigned int *pMin, unsigned int *pMax)`
Get the range of a statistics channel.
- `FLYCAPTURE2_C_API fc2Error fc2GetChannelPixelValueRange (fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, unsigned int *pPixelValueMin, unsigned int *pPixelValueMax)`
Get the range of a statistics channel.
- `FLYCAPTURE2_C_API fc2Error fc2GetChannelNumPixelValues (fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, unsigned int *pNumPixelValues)`
Get the number of unique pixel values in the image.
- `FLYCAPTURE2_C_API fc2Error fc2GetChannelMean (fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, float *pPixelValueMean)`
Get the mean of the image.
- `FLYCAPTURE2_C_API fc2Error fc2GetChannelHistogram (fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, int **ppHistogram)`
Get the histogram for the image.
- `FLYCAPTURE2_C_API fc2Error fc2GetImageStatistics (fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, unsigned int *pRangeMin, unsigned int *pRangeMax, unsigned int *pPixelValueMin, unsigned int *pPixelValueMax, unsigned int *pNumPixelValues, float *pPixelValueMean, int **ppHistogram)`
Get all statistics for the image.

5.18.1 Detailed Description

The Image Statistics operation provides the functionality for the user to collect image channel statistics.

5.18.2 Function Documentation

5.18.2.1 FLYCAPTURE2_C_API `fc2Error fc2CreateImageStatistics (fc2ImageStatisticsContext * pImageStatisticsContext)`

Create a statistics context.

Parameters

<i>pImageStatisticsContext</i>	A statistics context.
--------------------------------	-----------------------

Returns

A `fc2Error` indicating the success or failure of the function.

5.18.2.2 FLYCAPTURE2_C_API `fc2Error fc2DestroyImageStatistics (fc2ImageStatisticsContext imageStatisticsContext)`

Destroy a statistics context.

Parameters

<i>imageStatisticsContext</i>	A statistics context.
-------------------------------	-----------------------

Returns

A `fc2Error` indicating the success or failure of the function.

5.18.2.3 FLYCAPTURE2_C_API `fc2Error fc2GetChannelHistogram (fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, int ** ppHistogram)`

Get the histogram for the image.

Parameters

<i>imageStatisticsContext</i>	A statistics context.
<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.

5.18.2.4 FLYCAPTURE2_C_API fc2Error fc2GetChannelMean (fc2ImageStatisticsContext *imageStatisticsContext*, fc2StatisticsChannel *channel*, float * *pPixelValueMean*)

Get the mean of the image.

Parameters

<i>imageStatisticsContext</i>	A statistics context.
<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.

5.18.2.5 FLYCAPTURE2_C_API fc2Error fc2GetChannelNumPixelValues (fc2ImageStatisticsContext *imageStatisticsContext*, fc2StatisticsChannel *channel*, unsigned int * *pNumPixelValues*)

Get the number of unique pixel values in the image.

Parameters

<i>imageStatisticsContext</i>	A statistics context.
<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.

5.18.2.6 FLYCAPTURE2_C_API fc2Error fc2GetChannelPixelValueRange (fc2ImageStatisticsContext *imageStatisticsContext*, fc2StatisticsChannel *channel*, unsigned int * *pPixelValueMin*, unsigned int * *pPixelValueMax*)

Get the range of a statistics channel.

The values returned are the maximum values recorded for all pixels in the image.

Parameters

<i>imageStatisticsContext</i>	A statistics context.
<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.

5.18.2.7 FLYCAPTURE2_C_API **fc2Error** **fc2GetChannelRange** (**fc2ImageStatisticsContext** *imageStatisticsContext*, **fc2StatisticsChannel** *channel*, unsigned int * *pMin*, unsigned int * *pMax*)

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>imageStatisticsContext</i>	A statistics context.
<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.

5.18.2.8 FLYCAPTURE2_C_API **fc2Error** **fc2GetChannelStatus** (**fc2ImageStatisticsContext** *imageStatisticsContext*, **fc2StatisticsChannel** *channel*, **BOOL** * *pEnabled*)

Get the status of a statistics channel.

See also

[fc2SetChannelStatus\(\)](#)

Parameters

<i>imageStatisticsContext</i>	A statistics context.
<i>channel</i>	The statistics channel.
<i>pEnabled</i>	Whether the channel is enabled.

Returns

An Error indicating the success or failure of the function.

5.18.2.9 FLYCAPTURE2_C_API **fc2Error** **fc2GetImageStatistics** (**fc2ImageStatisticsContext** *imageStatisticsContext*, **fc2StatisticsChannel** *channel*, unsigned int * *pRangeMin*, unsigned int * *pRangeMax*, unsigned int * *pPixelValueMin*, unsigned int * *pPixelValueMax*, unsigned int * *pNumPixelValues*, float * *pPixelValueMean*, int ** *ppHistogram*)

Get all statistics for the image.

Parameters

<i>imageStatisticsContext</i>	The statistics context.
<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

A `fc2Error` indicating the success or failure of the function.

5.18.2.10 FLYCAPTURE2_C_API `fc2Error fc2ImageStatisticsDisableAll (fc2ImageStatisticsContext imageStatisticsContext)`

Disable all channels.

Parameters

<i>imageStatisticsContext</i>	A statistics context.
-------------------------------	-----------------------

Returns

An Error indicating the success or failure of the function.

5.18.2.11 FLYCAPTURE2_C_API `fc2Error fc2ImageStatisticsEnableAll (fc2ImageStatisticsContext imageStatisticsContext)`

Enable all channels.

Parameters

<i>imageStatisticsContext</i>	A statistics context.
-------------------------------	-----------------------

Returns

An Error indicating the success or failure of the function.

5.18.2.12 FLYCAPTURE2_C_API `fc2Error fc2ImageStatisticsEnableGreyOnly (fc2ImageStatisticsContext imageStatisticsContext)`

Enable only the grey channel.

Parameters

<i>imageStatisticsContext</i>	A statistics context.
-------------------------------	-----------------------

Returns

An Error indicating the success or failure of the function.

5.18.2.13 FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableHSLOnly (fc2ImageStatisticsContext *imageStatisticsContext*)

Enable only the HSL channels.

Parameters

<i>imageStatisticsContext</i>	A statistics context.
-------------------------------	-----------------------

Returns

An Error indicating the success or failure of the function.

5.18.2.14 FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableRGBOnly (fc2ImageStatisticsContext *imageStatisticsContext*)

Enable only the RGB channels.

Parameters

<i>imageStatisticsContext</i>	A statistics context.
-------------------------------	-----------------------

Returns

An Error indicating the success or failure of the function.

5.18.2.15 FLYCAPTURE2_C_API fc2Error fc2SetChannelStatus (fc2ImageStatisticsContext *imageStatisticsContext*, fc2StatisticsChannel *channel*, BOOL *enabled*)

Set the status of a statistics channel.

See also

[fc2GetChannelStatus\(\)](#)

Parameters

<i>imageStatisticsContext</i>	A statistics context.
<i>channel</i>	The statistics channel.
<i>enabled</i>	Whether the channel should be enabled.

Returns

An Error indicating the success or failure of the function.

5.19 AVI Recording Operation

The AVI recording operation provides the functionality for the user to record images to an AVI file.

Functions

- `FLYCAPTURE2_C_API fc2Error fc2CreateAVI (fc2AVIContext *pAVIContext)`
Create a AVI context.
- `FLYCAPTURE2_C_API fc2Error fc2AVIOpen (fc2AVIContext AVIContext, const char *pFileName, fc2AVI↵
Option *pOption)`
Open an AVI file in preparation for writing Images to disk.
- `FLYCAPTURE2_C_API fc2Error fc2MJPGOpen (fc2AVIContext AVIContext, const char *pFileName, fc2↵
MJPGOption *pOption)`
Open an MJPEG file in preparation for writing Images to disk.
- `FLYCAPTURE2_C_API fc2Error fc2H264Open (fc2AVIContext AVIContext, const char *pFileName, fc2↵
H264Option *pOption)`
Open an H.264 file in preparation for writing Images to disk.
- `FLYCAPTURE2_C_API fc2Error fc2AVIAppend (fc2AVIContext AVIContext, fc2Image *pImage)`
Append an image to the AVI file.
- `FLYCAPTURE2_C_API fc2Error fc2AVIClose (fc2AVIContext AVIContext)`
Close the AVI file.
- `FLYCAPTURE2_C_API fc2Error fc2DestroyAVI (fc2AVIContext AVIContext)`
Destroy a AVI context.

5.19.1 Detailed Description

The AVI recording operation provides the functionality for the user to record images to an AVI file.

5.19.2 Function Documentation

5.19.2.1 FLYCAPTURE2_C_API fc2Error fc2AVIAppend (fc2AVIContext AVIContext, fc2Image * pImage)

Append an image to the AVI file.

Parameters

<i>AVIContext</i>	The AVI context to use.
<i>pImage</i>	The image to append.

Returns

A `fc2Error` indicating the success or failure of the function.

5.19.2.2 FLYCAPTURE2_C_API fc2Error fc2AVIClose (fc2AVIContext AVIContext)

Close the AVI file.

Parameters

<i>AVIContext</i>	The AVI context to use.
-------------------	-------------------------

Returns

A `fc2Error` indicating the success or failure of the function.

5.19.2.3 FLYCAPTURE2_C_API `fc2Error fc2AVIOpen (fc2AVIContext AVIContext, const char * pFileName, fc2AVIOption * pOption)`

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

Returns

A `fc2Error` indicating the success or failure of the function.

5.19.2.4 FLYCAPTURE2_C_API `fc2Error fc2CreateAVI (fc2AVIContext * pAVIContext)`

Create a AVI context.

Parameters

<i>pAVIContext</i>	A AVI context.
--------------------	----------------

Returns

A `fc2Error` indicating the success or failure of the function.

5.19.2.5 FLYCAPTURE2_C_API `fc2Error fc2DestroyAVI (fc2AVIContext AVIContext)`

Destroy a AVI context.

Parameters

<i>AVIContext</i>	A AVI context.
-------------------	----------------

Returns

A `fc2Error` indicating the success or failure of the function.

5.19.2.6 FLYCAPTURE2_C_API `fc2Error fc2H264Open (fc2AVIContext AVIContext, const char * pFileName, fc2H264Option * pOption)`

Open an H.264 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>AVIContext</i>	The AVI context to use.
<i>pFileName</i>	The filename of the AVI file.
<i>pOption</i>	Options to apply to the AVI file.

Returns

A `fc2Error` indicating the success or failure of the function.

5.19.2.7 FLYCAPTURE2_C_API `fc2Error fc2MJPGOpen (fc2AVIContext AVIContext, const char * pFileName, fc2MJPGOption * pOption)`

Open an MJPEG 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>AVIContext</i>	The AVI context to use.
<i>pFileName</i>	The filename of the AVI file.
<i>pOption</i>	Options to apply to the AVI file.

Returns

A `fc2Error` indicating the success or failure of the function.

5.20 TopologyNode Operation

The TopologyNode operation provides the functionality for the user to generate a tree structure of all cameras and devices connected to a computer.

Functions

- **FLYCAPTURE2_C_API** **fc2Error** **fc2CreateTopologyNode** (**fc2TopologyNodeContext** *pTopologyNodeContext)
Create a TopologyNode context.
- **FLYCAPTURE2_C_API** **fc2Error** **fc2TopologyNodeGetGuid** (**fc2TopologyNodeContext** TopologyNodeContext, **fc2PGRGuid** *pGuid)
Get the PGRGuid associated with the node.
- **FLYCAPTURE2_C_API** **fc2Error** **fc2TopologyNodeGetDeviceId** (**fc2TopologyNodeContext** TopologyNodeContext, **int** *pID)
Get the device ID associated with the node.
- **FLYCAPTURE2_C_API** **fc2Error** **fc2TopologyNodeGetNodeType** (**fc2TopologyNodeContext** TopologyNodeContext, **fc2NodeType** *pNodeType)
Get the node type associated with the node.
- **FLYCAPTURE2_C_API** **fc2Error** **fc2TopologyNodeGetInterfaceType** (**fc2TopologyNodeContext** TopologyNodeContext, **fc2InterfaceType** *pInterfaceType)
Get the interface type associated with the node.
- **FLYCAPTURE2_C_API** **fc2Error** **fc2TopologyNodeGetNumChildren** (**fc2TopologyNodeContext** TopologyNodeContext, **unsigned int** *pNumChildNodes)
Get the number of child nodes.
- **FLYCAPTURE2_C_API** **fc2Error** **fc2TopologyNodeGetChild** (**fc2TopologyNodeContext** TopologyNodeContext, **unsigned int** position, **fc2TopologyNodeContext** *pChildTopologyNodeContext)
Get child node located at the specified position.
- **FLYCAPTURE2_C_API** **fc2Error** **fc2TopologyNodeAddChild** (**fc2TopologyNodeContext** TopologyNodeContext, **fc2TopologyNodeContext** TopologyNodeChildContext)
Add the specified TopologyNode as a child of the node.
- **FLYCAPTURE2_C_API** **fc2Error** **fc2TopologyNodeGetNumPorts** (**fc2TopologyNodeContext** TopologyNodeContext, **unsigned int** *pNumPorts)
Get the number of ports.
- **FLYCAPTURE2_C_API** **fc2Error** **fc2TopologyNodeGetPortType** (**fc2TopologyNodeContext** TopologyNodeContext, **unsigned int** position, **fc2PortType** *pPortType)
Get type of port located at the specified position.
- **FLYCAPTURE2_C_API** **fc2Error** **fc2TopologyNodeAddPortType** (**fc2TopologyNodeContext** TopologyNodeContext, **fc2PortType** portType)
Add the specified PortType as a port of the node.
- **FLYCAPTURE2_C_API** **BOOL** **fc2TopologyNodeAssignGuidToNode** (**fc2TopologyNodeContext** TopologyNodeContext, **fc2PGRGuid** guid, **int** deviceId)
Assign a PGRGuid and device ID to the node.
- **FLYCAPTURE2_C_API** **BOOL** **fc2TopologyNodeAssignGuidToNodeEx** (**fc2TopologyNodeContext** TopologyNodeContext, **fc2PGRGuid** guid, **int** deviceId, **fc2NodeType** nodeType)
Assign a PGRGuid, device ID and nodeType to the node.
- **FLYCAPTURE2_C_API** **fc2Error** **fc2DestroyTopologyNode** (**fc2TopologyNodeContext** TopologyNodeContext)
Destroy a TopologyNode context.

5.20.1 Detailed Description

The TopologyNode operation provides the functionality for the user to generate a tree structure of all cameras and devices connected to a computer.

5.20.2 Function Documentation

5.20.2.1 FLYCAPTURE2_C_API fc2Error fc2CreateTopologyNode (fc2TopologyNodeContext * *pTopologyNodeContext*)

Create a TopologyNode context.

Parameters

<i>pTopologyNodeContext</i>	A Topology Node context.
-----------------------------	--------------------------

Returns

A fc2Error indicating the success or failure of the function.

5.20.2.2 FLYCAPTURE2_C_API fc2Error fc2DestroyTopologyNode (fc2TopologyNodeContext *TopologyNodeContext*)

Destroy a TopologyNode context.

Parameters

<i>TopologyNodeContext</i>	A Topology Node context.
----------------------------	--------------------------

Returns

A fc2Error indicating the success or failure of the function.

5.20.2.3 FLYCAPTURE2_C_API fc2Error fc2TopologyNodeAddChild (fc2TopologyNodeContext *TopologyNodeContext*, fc2TopologyNodeContext *TopologyNodeChildContext*)

Add the specified TopologyNode as a child of the node.

Parameters

<i>TopologyNodeContext</i>	The Topology Node context to use.
<i>TopologyNodeChildContext</i>	The TopologyNode child context to add.

Returns

A `fc2Error` indicating the success or failure of the function.

5.20.2.4 FLYCAPTURE2_C_API `fc2Error` `fc2TopologyNodeAddPortType` (`fc2TopologyNodeContext` `TopologyNodeContext`, `fc2PortType` `portType`)

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

Parameters

<i>TopologyNodeContext</i>	The Topology Node context to use.
<i>portType</i>	childPort The port to add.

Returns

A `fc2Error` indicating the success or failure of the function.

5.20.2.5 FLYCAPTURE2_C_API `BOOL` `fc2TopologyNodeAssignGuidToNode` (`fc2TopologyNodeContext` `TopologyNodeContext`, `fc2PGRGuid` `guid`, `int` `deviceId`)

Assign a `PGRGuid` and device ID to the node.

Parameters

<i>TopologyNodeContext</i>	The Topology Node context to use.
<i>guid</i>	<code>PGRGuid</code> to be assigned.
<i>deviceId</i>	Device ID to be assigned.

Returns

A `fc2Error` indicating the success or failure of the function.

5.20.2.6 FLYCAPTURE2_C_API `BOOL` `fc2TopologyNodeAssignGuidToNodeEx` (`fc2TopologyNodeContext` `TopologyNodeContext`, `fc2PGRGuid` `guid`, `int` `deviceId`, `fc2NodeType` `nodeType`)

Assign a `PGRGuid`, device ID and `nodeType` to the node.

Parameters

<i>TopologyNodeContext</i>	The Topology Node context to use.
<i>guid</i>	<code>PGRGuid</code> to be assigned.
<i>deviceId</i>	Device ID to be assigned.
<i>nodeType</i>	<code>NodeType</code> to be assigned

Returns

A fc2Error indicating the success or failure of the function.

**5.20.2.7 FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetChild (fc2TopologyNodeContext
TopologyNodeContext, unsigned int position, fc2TopologyNodeContext * pChildTopologyNodeContext)**

Get child node located at the specified position.

Parameters

<i>TopologyNodeContext</i>	The Topology Node context to use.
<i>position</i>	Position of the child node.
<i>pChildTopologyNodeContext</i>	The Topology Node context the contains information on the child topology

Returns

A fc2Error indicating the success or failure of the function.

**5.20.2.8 FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetDeviceld (fc2TopologyNodeContext
TopologyNodeContext, int * pID)**

Get the device ID associated with the node.

Parameters

<i>TopologyNodeContext</i>	The Topology Node context to use.
<i>pID</i>	Device ID of the node.

Returns

A fc2Error indicating the success or failure of the function.

**5.20.2.9 FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetGuid (fc2TopologyNodeContext
TopologyNodeContext, fc2PGRGuid * pGuid)**

Get the PGRGuid associated with the node.

Parameters

<i>TopologyNodeContext</i>	The Topology Node context to use.
<i>pGuid</i>	The unique identifier associated with the node.

Returns

A fc2Error indicating the success or failure of the function.

5.20.2.10 FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetInterfaceType (fc2TopologyNodeContext TopologyNodeContext, fc2InterfaceType * pInterfaceType)

Get the interface type associated with the node.

Parameters

<i>TopologyNodeContext</i>	The Topology Node context to use.
<i>pInterfaceType</i>	Interface type of the node.

Returns

A fc2Error indicating the success or failure of the function.

5.20.2.11 FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetNodeType (fc2TopologyNodeContext TopologyNodeContext, fc2NodeType * pNodeType)

Get the node type associated with the node.

Parameters

<i>TopologyNodeContext</i>	The Topology Node context to use.
<i>pNodeType</i>	Node type of the node.

Returns

A fc2Error indicating the success or failure of the function.

5.20.2.12 FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetNumChildren (fc2TopologyNodeContext TopologyNodeContext, unsigned int * pNumChildNodes)

Get the number of child nodes.

Parameters

<i>TopologyNodeContext</i>	The Topology Node context to use.
<i>pNumChildNodes</i>	Number of child nodes.

Returns

A fc2Error indicating the success or failure of the function.

5.20.2.13 FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetNumPorts (fc2TopologyNodeContext TopologyNodeContext, unsigned int * pNumPorts)

Get the number of ports.

Parameters

<i>TopologyNodeContext</i>	The Topology Node context to use.
<i>pNumPorts</i>	Number of ports.

Returns

A `fc2Error` indicating the success or failure of the function.

5.20.2.14 FLYCAPTURE2_C_API `fc2Error fc2TopologyNodeGetPortType (fc2TopologyNodeContext TopologyNodeContext, unsigned int position, fc2PortType * pPortType)`

Get type of port located at the specified position.

Parameters

<i>TopologyNodeContext</i>	The Topology Node context to use.
<i>position</i>	Position of the port.
<i>pPortType</i>	PortType at the specified position.

Returns

A `fc2Error` indicating the success or failure of the function.

5.21 Utilities

The utility operations are used to query for general system information such as operating system, available memory etc.

Functions

- `FLYCAPTURE2_C_API fc2Error fc2CheckDriver (const fc2PGRGuid *pGuid)`
Check for driver compatibility for the given camera guid.
- `FLYCAPTURE2_C_API fc2Error fc2GetDriverDeviceName (const fc2PGRGuid *pGuid, char *pDeviceName, size_t *deviceNameLength)`
Get the driver's name for a device.
- `FLYCAPTURE2_C_API fc2Error fc2GetSystemInfo (fc2SystemInfo *pSystemInfo)`
Get system information.
- `FLYCAPTURE2_C_API fc2Error fc2GetLibraryVersion (fc2Version *pVersion)`
Get library version.
- `FLYCAPTURE2_C_API fc2Error fc2LaunchBrowser (const char *pAddress)`
Launch a URL in the system default browser.
- `FLYCAPTURE2_C_API fc2Error fc2LaunchHelp (const char *pFileName)`
Open a CHM file in the system default CHM viewer.
- `FLYCAPTURE2_C_API fc2Error fc2LaunchCommand (const char *pCommand)`
Execute a command in the terminal.
- `FLYCAPTURE2_C_API fc2Error fc2LaunchCommandAsync (const char *pCommand, fc2AsyncCommandCallback pCallback, void *pUserData)`
Execute a command in the terminal.
- `FLYCAPTURE2_C_API const char * fc2ErrorToDescription (fc2Error error)`
Get a string representation of an error.

5.21.1 Detailed Description

The utility operations are used to query for general system information such as operating system, available memory etc.

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

5.21.2 Function Documentation

5.21.2.1 FLYCAPTURE2_C_API fc2Error fc2CheckDriver (const fc2PGRGuid * pGuid)

Check for driver compatibility for the given camera guid.

Parameters

<code>pGuid</code>	The PGRGuid of the device to check.
--------------------	-------------------------------------

Returns

FC2_ERROR_OK if the library is compatible with the currently loaded driver, otherwise an error indicating the type of failure.

5.21.2.2 FLYCAPTURE2_C_API `const char* fc2ErrorToDescription (fc2Error error)`

Get a string representation of an error.

Parameters

<i>error</i>	Error to be parsed.
--------------	---------------------

Returns

A fc2Error indicating the success or failure of the function.

5.21.2.3 FLYCAPTURE2_C_API `fc2Error fc2GetDriverDeviceName (const fc2PGRGuid * pGuid, char * pDeviceName, size_t * deviceNameLength)`

Get the driver's name for a device.

Parameters

<i>pGuid</i>	The PGRGuid of the device to check.
<i>pDeviceName</i>	The device name will be returned in this string
<i>pDeviceNameLength</i>	The length of the device name string returned

Returns

An Error indicating the success or failure of the function.

5.21.2.4 FLYCAPTURE2_C_API `fc2Error fc2GetLibraryVersion (fc2Version * pVersion)`

Get library version.

Parameters

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

Returns

A fc2Error indicating the success or failure of the function.

5.21.2.5 FLYCAPTURE2_C_API `fc2Error fc2GetSystemInfo (fc2SystemInfo * pSystemInfo)`

Get system information.

Parameters

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

Returns

A `fc2Error` indicating the success or failure of the function.

5.21.2.6 FLYCAPTURE2_C_API `fc2Error fc2LaunchBrowser (const char * pAddress)`

Launch a URL in the system default browser.

Parameters

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

Returns

A `fc2Error` indicating the success or failure of the function.

5.21.2.7 FLYCAPTURE2_C_API `fc2Error fc2LaunchCommand (const char * pCommand)`

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.
-----------------	---------------------

Returns

A `fc2Error` indicating the success or failure of the function.

5.21.2.8 FLYCAPTURE2_C_API `fc2Error fc2LaunchCommandAsync (const char * pCommand, fc2AsyncCommandCallback pCallback, void * pUserData)`

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.

Returns

A fc2Error indicating the success or failure of the function.

5.21.2.9 FLYCAPTURE2_C_API fc2Error fc2LaunchHelp (const char * *pFileName*)

Open a CHM file in the system default CHM viewer.

Parameters

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

Returns

A fc2Error indicating the success or failure of the function.

5.22 TypeDefs

Data Structures

- struct [fc2PGRGuid](#)
A GUID to the camera.

Macros

- #define [FALSE](#) 0
- #define [TRUE](#) 1
- #define [FULL_32BIT_VALUE](#) 0x7FFFFFFF
- #define [MAX_STRING_LENGTH](#) 512

Typedefs

- typedef int [BOOL](#)
- typedef void * [fc2Context](#)
A context to the FlyCapture2 C library.
- typedef void * [fc2GuiContext](#)
A context to the FlyCapture2 C GUI library.
- typedef void * [fc2ImageImpl](#)
An internal pointer used in the [fc2Image](#) structure.
- typedef void * [fc2AVIContext](#)
A context referring to the AVI recorder object.
- typedef void * [fc2ImageStatisticsContext](#)
A context referring to the ImageStatistics object.
- typedef void * [fc2TopologyNodeContext](#)
A context referring to the TopologyNode object.

5.22.1 Detailed Description

5.22.2 Macro Definition Documentation

5.22.2.1 #define [FALSE](#) 0

5.22.2.2 #define [FULL_32BIT_VALUE](#) 0x7FFFFFFF

5.22.2.3 #define [MAX_STRING_LENGTH](#) 512

5.22.2.4 #define [TRUE](#) 1

5.22.3 Typedef Documentation

5.22.3.1 typedef int [BOOL](#)

5.22.3.2 typedef void* [fc2AVIContext](#)

A context referring to the AVI recorder object.

5.22.3.3 `typedef void* fc2Context`

A context to the FlyCapture2 C library.

It must be created before performing any calls to the library.

5.22.3.4 `typedef void* fc2GuiContext`

A context to the FlyCapture2 C GUI library.

It must be created before performing any calls to the library.

5.22.3.5 `typedef void* fc2ImageImpl`

An internal pointer used in the [fc2Image](#) structure.

5.22.3.6 `typedef void* fc2ImageStatisticsContext`

A context referring to the ImageStatistics object.

5.22.3.7 `typedef void* fc2TopologyNodeContext`

A context referring to the TopologyNode object.

5.23 Enumerations

Enumerations

- `enum fc2Error {`
`FC2_ERROR_UNDEFINED = -1,`
`FC2_ERROR_OK,`
`FC2_ERROR_FAILED,`
`FC2_ERROR_NOT_IMPLEMENTED,`
`FC2_ERROR_FAILED_BUS_MASTER_CONNECTION,`
`FC2_ERROR_NOT_CONNECTED,`
`FC2_ERROR_INIT_FAILED,`
`FC2_ERROR_NOT_INITIALIZED,`
`FC2_ERROR_INVALID_PARAMETER,`
`FC2_ERROR_INVALID_SETTINGS,`
`FC2_ERROR_INVALID_BUS_MANAGER,`
`FC2_ERROR_MEMORY_ALLOCATION_FAILED,`
`FC2_ERROR_LOW_LEVEL_FAILURE,`
`FC2_ERROR_NOT_FOUND,`
`FC2_ERROR_FAILED_GUID,`
`FC2_ERROR_INVALID_PACKET_SIZE,`
`FC2_ERROR_INVALID_MODE,`
`FC2_ERROR_NOT_IN_FORMAT7,`
`FC2_ERROR_NOT_SUPPORTED,`
`FC2_ERROR_TIMEOUT,`
`FC2_ERROR_BUS_MASTER_FAILED,`
`FC2_ERROR_INVALID_GENERATION,`
`FC2_ERROR_LUT_FAILED,`
`FC2_ERROR_IIDC_FAILED,`
`FC2_ERROR_STROBE_FAILED,`
`FC2_ERROR_TRIGGER_FAILED,`
`FC2_ERROR_PROPERTY_FAILED,`
`FC2_ERROR_PROPERTY_NOT_PRESENT,`
`FC2_ERROR_REGISTER_FAILED,`
`FC2_ERROR_READ_REGISTER_FAILED,`
`FC2_ERROR_WRITE_REGISTER_FAILED,`
`FC2_ERROR_ISOCH_FAILED,`
`FC2_ERROR_ISOCH_ALREADY_STARTED,`
`FC2_ERROR_ISOCH_NOT_STARTED,`
`FC2_ERROR_ISOCH_START_FAILED,`
`FC2_ERROR_ISOCH_RETRIEVE_BUFFER_FAILED,`
`FC2_ERROR_ISOCH_STOP_FAILED,`
`FC2_ERROR_ISOCH_SYNC_FAILED,`
`FC2_ERROR_ISOCH_BANDWIDTH_EXCEEDED,`
`FC2_ERROR_IMAGE_CONVERSION_FAILED,`
`FC2_ERROR_IMAGE_LIBRARY_FAILURE,`
`FC2_ERROR_BUFFER_TOO_SMALL,`
`FC2_ERROR_IMAGE_CONSISTENCY_ERROR,`
`FC2_ERROR_INCOMPATIBLE_DRIVER,`
`FC2_ERROR_FORCE_32BITS = FULL_32BIT_VALUE }`

The error types returned by functions.
- `enum fc2BusCallbackType {`
`FC2_BUS_RESET,`
`FC2_ARRIVAL,`
`FC2_REMOVAL,`
`FC2_CALLBACK_TYPE_FORCE_32BITS = FULL_32BIT_VALUE }`

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

- enum `fc2GrabMode` {
`FC2_DROP_FRAMES`,
`FC2_BUFFER_FRAMES`,
`FC2_UNSPECIFIED_GRAB_MODE`,
`FC2_GRAB_MODE_FORCE_32BITS` = `FULL_32BIT_VALUE` }

The grab strategy employed during image transfer.

- enum `fc2GrabTimeout` {
`FC2_TIMEOUT_NONE` = 0,
`FC2_TIMEOUT_INFINITE` = -1,
`FC2_TIMEOUT_UNSPECIFIED` = -2,
`FC2_GRAB_TIMEOUT_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Timeout options for grabbing images.

- enum `fc2BandwidthAllocation` {
`FC2_BANDWIDTH_ALLOCATION_OFF` = 0,
`FC2_BANDWIDTH_ALLOCATION_ON` = 1,
`FC2_BANDWIDTH_ALLOCATION_UNSUPPORTED` = 2,
`FC2_BANDWIDTH_ALLOCATION_UNSPECIFIED` = 3,
`FC2_BANDWIDTH_ALLOCATION_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Bandwidth allocation options for 1394 devices.

- enum `fc2InterfaceType` {
`FC2_INTERFACE_IEEE1394`,
`FC2_INTERFACE_USB_2`,
`FC2_INTERFACE_USB_3`,
`FC2_INTERFACE_GIGE`,
`FC2_INTERFACE_UNKNOWN`,
`FC2_INTERFACE_TYPE_FORCE_32BITS` = `FULL_32BIT_VALUE` }

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

- enum `fc2PropertyType` {
`FC2_BRIGHTNESS`,
`FC2_AUTO_EXPOSURE`,
`FC2_SHARPNESS`,
`FC2_WHITE_BALANCE`,
`FC2_HUE`,
`FC2_SATURATION`,
`FC2_GAMMA`,
`FC2_IRIS`,
`FC2_FOCUS`,
`FC2_ZOOM`,
`FC2_PAN`,
`FC2_TILT`,
`FC2_SHUTTER`,
`FC2_GAIN`,
`FC2_TRIGGER_MODE`,
`FC2_TRIGGER_DELAY`,
`FC2_FRAME_RATE`,
`FC2_TEMPERATURE`,
`FC2_UNSPECIFIED_PROPERTY_TYPE`,
`FC2_PROPERTY_TYPE_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Camera properties.

- enum `fc2FrameRate` {

```

FC2_FRAMERATE_1_875,
FC2_FRAMERATE_3_75,
FC2_FRAMERATE_7_5,
FC2_FRAMERATE_15,
FC2_FRAMERATE_30,
FC2_FRAMERATE_60,
FC2_FRAMERATE_120,
FC2_FRAMERATE_240,
FC2_FRAMERATE_FORMAT7,
FC2_NUM_FRAMERATES,
FC2_FRAMERATE_FORCE_32BITS = FULL_32BIT_VALUE }

```

Frame rates in frames per second.

- enum `fc2VideoMode` {


```

FC2_VIDEOMODE_160x120YUV444,
FC2_VIDEOMODE_320x240YUV422,
FC2_VIDEOMODE_640x480YUV411,
FC2_VIDEOMODE_640x480YUV422,
FC2_VIDEOMODE_640x480RGB,
FC2_VIDEOMODE_640x480Y8,
FC2_VIDEOMODE_640x480Y16,
FC2_VIDEOMODE_800x600YUV422,
FC2_VIDEOMODE_800x600RGB,
FC2_VIDEOMODE_800x600Y8,
FC2_VIDEOMODE_800x600Y16,
FC2_VIDEOMODE_1024x768YUV422,
FC2_VIDEOMODE_1024x768RGB,
FC2_VIDEOMODE_1024x768Y8,
FC2_VIDEOMODE_1024x768Y16,
FC2_VIDEOMODE_1280x960YUV422,
FC2_VIDEOMODE_1280x960RGB,
FC2_VIDEOMODE_1280x960Y8,
FC2_VIDEOMODE_1280x960Y16,
FC2_VIDEOMODE_1600x1200YUV422,
FC2_VIDEOMODE_1600x1200RGB,
FC2_VIDEOMODE_1600x1200Y8,
FC2_VIDEOMODE_1600x1200Y16,
FC2_VIDEOMODE_FORMAT7,
FC2_NUM_VIDEOMODES,
FC2_VIDEOMODE_FORCE_32BITS = FULL_32BIT_VALUE }

```

DCAM video modes.

- enum `fc2Mode` {

```

FC2_MODE_0 = 0,
FC2_MODE_1,
FC2_MODE_2,
FC2_MODE_3,
FC2_MODE_4,
FC2_MODE_5,
FC2_MODE_6,
FC2_MODE_7,
FC2_MODE_8,
FC2_MODE_9,
FC2_MODE_10,
FC2_MODE_11,
FC2_MODE_12,
FC2_MODE_13,
FC2_MODE_14,
FC2_MODE_15,
FC2_MODE_16,
FC2_MODE_17,
FC2_MODE_18,
FC2_MODE_19,
FC2_MODE_20,
FC2_MODE_21,
FC2_MODE_22,
FC2_MODE_23,
FC2_MODE_24,
FC2_MODE_25,
FC2_MODE_26,
FC2_MODE_27,
FC2_MODE_28,
FC2_MODE_29,
FC2_MODE_30,
FC2_MODE_31,
FC2_NUM_MODES,
FC2_MODE_FORCE_32BITS = FULL_32BIT_VALUE }

```

Camera modes for DCAM formats as well as Format7.

```

• enum fc2PixelFormat {
    FC2_PIXEL_FORMAT_MONO8 = 0x80000000,
    FC2_PIXEL_FORMAT_411YUV8 = 0x40000000,
    FC2_PIXEL_FORMAT_422YUV8 = 0x20000000,
    FC2_PIXEL_FORMAT_444YUV8 = 0x10000000,
    FC2_PIXEL_FORMAT_RGB8 = 0x08000000,
    FC2_PIXEL_FORMAT_MONO16 = 0x04000000,
    FC2_PIXEL_FORMAT_RGB16 = 0x02000000,
    FC2_PIXEL_FORMAT_S_MONO16 = 0x01000000,
    FC2_PIXEL_FORMAT_S_RGB16 = 0x00800000,
    FC2_PIXEL_FORMAT_RAW8 = 0x00400000,
    FC2_PIXEL_FORMAT_RAW16 = 0x00200000,
    FC2_PIXEL_FORMAT_MONO12 = 0x00100000,
    FC2_PIXEL_FORMAT_RAW12 = 0x00080000,
    FC2_PIXEL_FORMAT_BGR = 0x80000008,
    FC2_PIXEL_FORMAT_BGRU = 0x40000008,
    FC2_PIXEL_FORMAT_RGB = FC2_PIXEL_FORMAT_RGB8,
    FC2_PIXEL_FORMAT_RGBU = 0x40000002,
    FC2_PIXEL_FORMAT_BGR16 = 0x02000001,
    FC2_PIXEL_FORMAT_BGRU16 = 0x02000002,
    FC2_PIXEL_FORMAT_422YUV8_JPEG = 0x40000001,
    FC2_NUM_PIXEL_FORMATS = 20,
    FC2_UNSPECIFIED_PIXEL_FORMAT = 0 }

```

Pixel formats available for Format7 modes.

- enum `fc2BusSpeed` {
`FC2_BUSSPEED_S100`,
`FC2_BUSSPEED_S200`,
`FC2_BUSSPEED_S400`,
`FC2_BUSSPEED_S480`,
`FC2_BUSSPEED_S800`,
`FC2_BUSSPEED_S1600`,
`FC2_BUSSPEED_S3200`,
`FC2_BUSSPEED_S5000`,
`FC2_BUSSPEED_10BASE_T`,
`FC2_BUSSPEED_100BASE_T`,
`FC2_BUSSPEED_1000BASE_T`,
`FC2_BUSSPEED_10000BASE_T`,
`FC2_BUSSPEED_S_FASTEST`,
`FC2_BUSSPEED_ANY`,
`FC2_BUSSPEED_SPEED_UNKNOWN` = -1,
`FC2_BUSSPEED_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Bus speeds.

- enum `fc2PCleBusSpeed` {
`FC2_PCIE_BUSSPEED_2_5`,
`FC2_PCIE_BUSSPEED_5_0`,
`FC2_PCIE_BUSSPEED_UNKNOWN` = -1,
`FC2_PCIE_BUSSPEED_FORCE_32BITS` = `FULL_32BIT_VALUE` }
- enum `fc2DriverType` {
`FC2_DRIVER_1394_CAM`,
`FC2_DRIVER_1394_PRO`,
`FC2_DRIVER_1394_JUUU`,
`FC2_DRIVER_1394_VIDEO1394`,
`FC2_DRIVER_1394_RAW1394`,
`FC2_DRIVER_USB_NONE`,
`FC2_DRIVER_USB_CAM`,
`FC2_DRIVER_USB3_PRO`,
`FC2_DRIVER_GIGE_NONE`,
`FC2_DRIVER_GIGE_FILTER`,
`FC2_DRIVER_GIGE_PRO`,
`FC2_DRIVER_GIGE_LWF`,
`FC2_DRIVER_UNKNOWN` = -1,
`FC2_DRIVER_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Types of low level drivers that FlyCapture uses.

- enum `fc2ColorProcessingAlgorithm` {
`FC2_DEFAULT`,
`FC2_NO_COLOR_PROCESSING`,
`FC2_NEAREST_NEIGHBOR_FAST`,
`FC2_EDGE_SENSING`,
`FC2_HQ_LINEAR`,
`FC2_RIGOROUS`,
`FC2_IPP`,
`FC2_DIRECTIONAL`,
`FC2_COLOR_PROCESSING_ALGORITHM_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Color processing algorithms.

- enum `fc2BayerTileFormat` {
`FC2_BT_NONE`,
`FC2_BT_RGGB`,
`FC2_BT_GRGB`,
`FC2_BT_GBRG`,
`FC2_BT_BGGR`,
`FC2_BT_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Bayer tile formats.

```
enum fc2ImageFileFormat {
    FC2_FROM_FILE_EXT = -1,
    FC2_PGM,
    FC2_PPM,
    FC2_BMP,
    FC2_JPEG,
    FC2_JPEG2000,
    FC2_TIFF,
    FC2_PNG,
    FC2_RAW,
    FC2_IMAGE_FILE_FORMAT_FORCE_32BITS = FULL_32BIT_VALUE }

```

File formats to be used for saving images to disk.

5.23.1 Detailed Description

5.23.2 Enumeration Type Documentation

5.23.2.1 enum fc2BandwidthAllocation

Bandwidth allocation options for 1394 devices.

Enumerator

FC2_BANDWIDTH_ALLOCATION_OFF Do not allocate bandwidth.

FC2_BANDWIDTH_ALLOCATION_ON Allocate bandwidth. This is the default setting.

FC2_BANDWIDTH_ALLOCATION_UNSUPPORTED Bandwidth allocation is not supported by either the camera or operating system.

FC2_BANDWIDTH_ALLOCATION_UNSPECIFIED Not specified. This leaves the current setting unchanged.

FC2_BANDWIDTH_ALLOCATION_FORCE_32BITS

5.23.2.2 enum fc2BayerTileFormat

Bayer tile formats.

Enumerator

FC2_BT_NONE No bayer tile format.

FC2_BT_RGGB Red-Green-Green-Blue.

FC2_BT_GRBG Green-Red-Blue-Green.

FC2_BT_GBRG Green-Blue-Red-Green.

FC2_BT_BGGR Blue-Green-Green-Red.

FC2_BT_FORCE_32BITS

5.23.2.3 enum fc2BusCallbackType

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

Enumerator

FC2_BUS_RESET Register for all bus events.
FC2_ARRIVAL Register for arrivals only.
FC2_REMOVAL Register for removals only.
FC2_CALLBACK_TYPE_FORCE_32BITS

5.23.2.4 enum fc2BusSpeed

Bus speeds.

Enumerator

FC2_BUSSPEED_S100 100Mbps/sec.
FC2_BUSSPEED_S200 200Mbps/sec.
FC2_BUSSPEED_S400 400Mbps/sec.
FC2_BUSSPEED_S480 480Mbps/sec. Only for USB2 cameras.
FC2_BUSSPEED_S800 800Mbps/sec.
FC2_BUSSPEED_S1600 1600Mbps/sec.
FC2_BUSSPEED_S3200 3200Mbps/sec.
FC2_BUSSPEED_S5000 5000Mbps/sec. Only for USB3 cameras.
FC2_BUSSPEED_10BASE_T 10Base-T. Only for GigE cameras.
FC2_BUSSPEED_100BASE_T 100Base-T. Only for GigE cameras.
FC2_BUSSPEED_1000BASE_T 1000Base-T (Gigabit Ethernet). Only for GigE cameras.
FC2_BUSSPEED_10000BASE_T 10000Base-T. Only for GigE cameras.
FC2_BUSSPEED_S_FASTEST The fastest speed available.
FC2_BUSSPEED_ANY Any speed that is available.
FC2_BUSSPEED_SPEED_UNKNOWN Unknown bus speed.
FC2_BUSSPEED_FORCE_32BITS

5.23.2.5 enum fc2ColorProcessingAlgorithm

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

FC2_DEFAULT Default method.
FC2_NO_COLOR_PROCESSING No color processing.
FC2_NEAREST_NEIGHBOR_FAST Fastest but lowest quality. Equivalent to FLYCAPTURE_NEAREST_NEIGHBOR_FAST in FlyCapture.
FC2_EDGE_SENSING Weights surrounding pixels based on localized edge orientation.
FC2_HQ_LINEAR Well-balanced speed and quality.
FC2_RIGOROUS Slowest but produces good results.
FC2_IPP Multithreaded with similar results to edge sensing.
FC2_DIRECTIONAL Best quality but much faster than rigorous.
FC2_COLOR_PROCESSING_ALGORITHM_FORCE_32BITS

5.23.2.6 enum fc2DriverType

Types of low level drivers that FlyCapture uses.

Enumerator

FC2_DRIVER_1394_CAM PGRCam.sys.
FC2_DRIVER_1394_PRO PGR1394.sys.
FC2_DRIVER_1394_JUJU firewire_core.
FC2_DRIVER_1394_VIDEO1394 video1394.
FC2_DRIVER_1394_RAW1394 raw1394.
FC2_DRIVER_USB_NONE No usb driver used just BSD stack. (Linux only)
FC2_DRIVER_USB_CAM PGRUsbCam.sys.
FC2_DRIVER_USB3_PRO PGRXHCl.sys.
FC2_DRIVER_GIGE_NONE no GigE drivers used, MS/BSD stack.
FC2_DRIVER_GIGE_FILTER PGRGigE.sys.
FC2_DRIVER_GIGE_PRO PGRGigEPro.sys.
FC2_DRIVER_GIGE_LWF PgrLwf.sys.
FC2_DRIVER_UNKNOWN Unknown driver type.
FC2_DRIVER_FORCE_32BITS

5.23.2.7 enum fc2Error

The error types returned by functions.

Enumerator

FC2_ERROR_UNDEFINED Undefined.
FC2_ERROR_OK Function returned with no errors.
FC2_ERROR_FAILED General failure.
FC2_ERROR_NOT_IMPLEMENTED Function has not been implemented.
FC2_ERROR_FAILED_BUS_MASTER_CONNECTION Could not connect to Bus Master.
FC2_ERROR_NOT_CONNECTED Camera has not been connected.
FC2_ERROR_INIT_FAILED Initialization failed.
FC2_ERROR_NOT_INITIALIZED Camera has not been initialized.
FC2_ERROR_INVALID_PARAMETER Invalid parameter passed to function.
FC2_ERROR_INVALID_SETTINGS Setting set to camera is invalid.
FC2_ERROR_INVALID_BUS_MANAGER Invalid Bus Manager object.
FC2_ERROR_MEMORY_ALLOCATION_FAILED Could not allocate memory.
FC2_ERROR_LOW_LEVEL_FAILURE Low level error.
FC2_ERROR_NOT_FOUND Device not found.
FC2_ERROR_FAILED_GUID GUID failure.
FC2_ERROR_INVALID_PACKET_SIZE Packet size set to camera is invalid.
FC2_ERROR_INVALID_MODE Invalid mode has been passed to function.
FC2_ERROR_NOT_IN_FORMAT7 Error due to not being in Format7.
FC2_ERROR_NOT_SUPPORTED This feature is unsupported.

FC2_ERROR_TIMEOUT Timeout error.

FC2_ERROR_BUS_MASTER_FAILED Bus Master Failure.

FC2_ERROR_INVALID_GENERATION Generation Count Mismatch.

FC2_ERROR_LUT_FAILED Look Up Table failure.

FC2_ERROR_IIDC_FAILED IIDC failure.

FC2_ERROR_STROBE_FAILED Strobe failure.

FC2_ERROR_TRIGGER_FAILED Trigger failure.

FC2_ERROR_PROPERTY_FAILED Property failure.

FC2_ERROR_PROPERTY_NOT_PRESENT Property is not present.

FC2_ERROR_REGISTER_FAILED Register access failed.

FC2_ERROR_READ_REGISTER_FAILED Register read failed.

FC2_ERROR_WRITE_REGISTER_FAILED Register write failed.

FC2_ERROR_ISOCH_FAILED Isochronous failure.

FC2_ERROR_ISOCH_ALREADY_STARTED Isochronous transfer has already been started.

FC2_ERROR_ISOCH_NOT_STARTED Isochronous transfer has not been started.

FC2_ERROR_ISOCH_START_FAILED Isochronous start failed.

FC2_ERROR_ISOCH_RETRIEVE_BUFFER_FAILED Isochronous retrieve buffer failed.

FC2_ERROR_ISOCH_STOP_FAILED Isochronous stop failed.

FC2_ERROR_ISOCH_SYNC_FAILED Isochronous image synchronization failed.

FC2_ERROR_ISOCH_BANDWIDTH_EXCEEDED Isochronous bandwidth exceeded.

FC2_ERROR_IMAGE_CONVERSION_FAILED Image conversion failed.

FC2_ERROR_IMAGE_LIBRARY_FAILURE Image library failure.

FC2_ERROR_BUFFER_TOO_SMALL Buffer is too small.

FC2_ERROR_IMAGE_CONSISTENCY_ERROR There is an image consistency error.

FC2_ERROR_INCOMPATIBLE_DRIVER The installed driver is not compatible with the library.

FC2_ERROR_FORCE_32BITS

5.23.2.8 enum fc2FrameRate

Frame rates in frames per second.

Enumerator

FC2_FRAMERATE_1_875 1.875 fps.

FC2_FRAMERATE_3_75 3.75 fps.

FC2_FRAMERATE_7_5 7.5 fps.

FC2_FRAMERATE_15 15 fps.

FC2_FRAMERATE_30 30 fps.

FC2_FRAMERATE_60 60 fps.

FC2_FRAMERATE_120 120 fps.

FC2_FRAMERATE_240 240 fps.

FC2_FRAMERATE_FORMAT7 Custom frame rate for Format7 functionality.

FC2_NUM_FRAMERATES Number of possible camera frame rates.

FC2_FRAMERATE_FORCE_32BITS

5.23.2.9 enum fc2GrabMode

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

FC2_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.

FC2_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 flycaptureLock↔Next in earlier versions of the FlyCapture SDK.

FC2_UNSPECIFIED_GRAB_MODE Unspecified grab mode.

FC2_GRAB_MODE_FORCE_32BITS

5.23.2.10 enum fc2GrabTimeout

Timeout options for grabbing images.

Enumerator

FC2_TIMEOUT_NONE Non-blocking wait.

FC2_TIMEOUT_INFINITE Wait indefinitely.

FC2_TIMEOUT_UNSPECIFIED Unspecified timeout setting.

FC2_GRAB_TIMEOUT_FORCE_32BITS

5.23.2.11 enum fc2ImageFileFormat

File formats to be used for saving images to disk.

Enumerator

FC2_FROM_FILE_EXT Determine file format from file extension.

FC2_PGM Portable gray map.

FC2_PPM Portable pixmap.

FC2_BMP Bitmap.

FC2_JPEG JPEG.

FC2_JPEG2000 JPEG 2000.

FC2_TIFF Tagged image file format.

FC2_PNG Portable network graphics.

FC2_RAW Raw data.

FC2_IMAGE_FILE_FORMAT_FORCE_32BITS

5.23.2.12 enum fc2InterfaceType

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

Enumerator

FC2_INTERFACE_IEEE1394 IEEE-1394 (Includes 1394a and 1394b).
FC2_INTERFACE_USB_2 USB 2.0.
FC2_INTERFACE_USB_3 USB 3.0.
FC2_INTERFACE_GIGE GigE.
FC2_INTERFACE_UNKNOWN Unknown interface.
FC2_INTERFACE_TYPE_FORCE_32BITS

5.23.2.13 enum fc2Mode

Camera modes for DCAM formats as well as Format7.

Enumerator

FC2_MODE_0
FC2_MODE_1
FC2_MODE_2
FC2_MODE_3
FC2_MODE_4
FC2_MODE_5
FC2_MODE_6
FC2_MODE_7
FC2_MODE_8
FC2_MODE_9
FC2_MODE_10
FC2_MODE_11
FC2_MODE_12
FC2_MODE_13
FC2_MODE_14
FC2_MODE_15
FC2_MODE_16
FC2_MODE_17
FC2_MODE_18
FC2_MODE_19
FC2_MODE_20
FC2_MODE_21
FC2_MODE_22
FC2_MODE_23
FC2_MODE_24
FC2_MODE_25
FC2_MODE_26
FC2_MODE_27
FC2_MODE_28
FC2_MODE_29
FC2_MODE_30
FC2_MODE_31
FC2_NUM_MODES Number of modes.
FC2_MODE_FORCE_32BITS

5.23.2.14 enum fc2PCleBusSpeed

Enumerator

FC2_PCIE_BUSSPEED_2_5
FC2_PCIE_BUSSPEED_5_0 2.5 Gb/s
FC2_PCIE_BUSSPEED_UNKNOWN 5.0 Gb/s
FC2_PCIE_BUSSPEED_FORCE_32BITS Speed is unknown.

5.23.2.15 enum fc2PixelFormat

Pixel formats available for Format7 modes.

Enumerator

FC2_PIXEL_FORMAT_MONO8 8 bits of mono information.
FC2_PIXEL_FORMAT_411YUV8 YUV 4:1:1.
FC2_PIXEL_FORMAT_422YUV8 YUV 4:2:2.
FC2_PIXEL_FORMAT_444YUV8 YUV 4:4:4.
FC2_PIXEL_FORMAT_RGB8 R = G = B = 8 bits.
FC2_PIXEL_FORMAT_MONO16 16 bits of mono information.
FC2_PIXEL_FORMAT_RGB16 R = G = B = 16 bits.
FC2_PIXEL_FORMAT_S_MONO16 16 bits of signed mono information.
FC2_PIXEL_FORMAT_S_RGB16 R = G = B = 16 bits signed.
FC2_PIXEL_FORMAT_RAW8 8 bit raw data output of sensor.
FC2_PIXEL_FORMAT_RAW16 16 bit raw data output of sensor.
FC2_PIXEL_FORMAT_MONO12 12 bits of mono information.
FC2_PIXEL_FORMAT_RAW12 12 bit raw data output of sensor.
FC2_PIXEL_FORMAT_BGR 24 bit BGR.
FC2_PIXEL_FORMAT_BGRU 32 bit BGRU.
FC2_PIXEL_FORMAT_RGB 24 bit RGB.
FC2_PIXEL_FORMAT_RGBU 32 bit RGBU.
FC2_PIXEL_FORMAT_BGR16 R = G = B = 16 bits.
FC2_PIXEL_FORMAT_BGRU16 64 bit BGRU.
FC2_PIXEL_FORMAT_422YUV8_JPEG JPEG compressed stream.
FC2_NUM_PIXEL_FORMATS Number of pixel formats.
FC2_UNSPECIFIED_PIXEL_FORMAT Unspecified pixel format.

5.23.2.16 enum fc2PropertyType

Camera properties.

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

Enumerator

FC2_BRIGHTNESS
FC2_AUTO_EXPOSURE
FC2_SHARPNESS
FC2_WHITE_BALANCE
FC2_HUE
FC2_SATURATION
FC2_GAMMA
FC2_IRIS
FC2_FOCUS
FC2_ZOOM
FC2_PAN
FC2_TILT
FC2_SHUTTER
FC2_GAIN
FC2_TRIGGER_MODE
FC2_TRIGGER_DELAY
FC2_FRAME_RATE
FC2_TEMPERATURE
FC2_UNSPECIFIED_PROPERTY_TYPE
FC2_PROPERTY_TYPE_FORCE_32BITS

5.23.2.17 enum fc2VideoMode

DCAM video modes.

Enumerator

FC2_VIDEOMODE_160x120YUV444 160x120 YUV444.
FC2_VIDEOMODE_320x240YUV422 320x240 YUV422.
FC2_VIDEOMODE_640x480YUV411 640x480 YUV411.
FC2_VIDEOMODE_640x480YUV422 640x480 YUV422.
FC2_VIDEOMODE_640x480RGB 640x480 24-bit RGB.
FC2_VIDEOMODE_640x480Y8 640x480 8-bit.
FC2_VIDEOMODE_640x480Y16 640x480 16-bit.
FC2_VIDEOMODE_800x600YUV422 800x600 YUV422.
FC2_VIDEOMODE_800x600RGB 800x600 RGB.
FC2_VIDEOMODE_800x600Y8 800x600 8-bit.
FC2_VIDEOMODE_800x600Y16 800x600 16-bit.
FC2_VIDEOMODE_1024x768YUV422 1024x768 YUV422.

FC2_VIDEOMODE_1024x768RGB 1024x768 RGB.
FC2_VIDEOMODE_1024x768Y8 1024x768 8-bit.
FC2_VIDEOMODE_1024x768Y16 1024x768 16-bit.
FC2_VIDEOMODE_1280x960YUV422 1280x960 YUV422.
FC2_VIDEOMODE_1280x960RGB 1280x960 RGB.
FC2_VIDEOMODE_1280x960Y8 1280x960 8-bit.
FC2_VIDEOMODE_1280x960Y16 1280x960 16-bit.
FC2_VIDEOMODE_1600x1200YUV422 1600x1200 YUV422.
FC2_VIDEOMODE_1600x1200RGB 1600x1200 RGB.
FC2_VIDEOMODE_1600x1200Y8 1600x1200 8-bit.
FC2_VIDEOMODE_1600x1200Y16 1600x1200 16-bit.
FC2_VIDEOMODE_FORMAT7 Custom video mode for Format7 functionality.
FC2_NUM_VIDEOMODES Number of possible video modes.
FC2_VIDEOMODE_FORCE_32BITS

5.24 GigE specific enumerations

These enumerations are specific to GigE camera operation only.

Enumerations

- enum `fc2GigEPropertyType` {
 `FC2_HEARTBEAT`,
 `FC2_HEARTBEAT_TIMEOUT`,
 `PACKET_SIZE`,
 `PACKET_DELAY` }

Possible properties that can be queried from the camera.

5.24.1 Detailed Description

These enumerations are specific to GigE camera operation only.

5.24.2 Enumeration Type Documentation

5.24.2.1 enum `fc2GigEPropertyType`

Possible properties that can be queried from the camera.

Enumerator

`FC2_HEARTBEAT`

`FC2_HEARTBEAT_TIMEOUT`

`PACKET_SIZE`

`PACKET_DELAY`

5.25 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.

Data Structures

- struct [fc2Image](#)
- struct [fc2SystemInfo](#)
Description of the system.
- struct [fc2Version](#)
The current version of the library.
- struct [fc2IPAddress](#)
IPv4 address.
- struct [fc2Format7ImageSettings](#)
Format 7 image settings.
- struct [fc2Config](#)
Configuration for a camera.
- struct [fc2TriggerDelayInfo](#)
Information about a specific camera property.
- struct [fc2TriggerDelay](#)
A specific camera property.
- struct [fc2TriggerModelInfo](#)
Information about a camera trigger property.
- struct [fc2TriggerMode](#)
A camera trigger.
- struct [fc2StrobeInfo](#)
A camera strobe property.
- struct [fc2StrobeControl](#)
A camera strobe.
- struct [fc2TimeStamp](#)
Timestamp information.
- struct [fc2ConfigROM](#)
Camera configuration ROM.
- struct [fc2CameraInfo](#)
Camera information.
- struct [fc2EmbeddedImageInfoProperty](#)
Properties of a single embedded image info property.
- struct [fc2EmbeddedImageInfo](#)
Properties of the possible embedded image information.
- struct [fc2ImageMetadata](#)

Metadata related to an image.

- struct [fc2LUTData](#)

Information about the camera's look up table.

- struct [fc2CameraStats](#)

Camera diagnostic information.

- struct [fc2PNGOption](#)

Options for saving PNG images.

5.25.1 Detailed Description

5.26 GigE specific structures

These structures are specific to GigE camera operation only.

Collaboration diagram for GigE specific structures:

Data Structures

- struct [fc2IPAddress](#)
IPv4 address.
- struct [fc2MACAddress](#)
MAC address.
- struct [fc2GigEProperty](#)
A GigE property.
- struct [fc2GigEStreamChannel](#)
Information about a single GigE stream channel.
- struct [fc2GigEConfig](#)
Configuration for a GigE camera.
- struct [fc2GigEImageSettingsInfo](#)
Format 7 information for a single mode.
- struct [fc2GigEImageSettings](#)
Image settings for a GigE camera.

5.26.1 Detailed Description

These structures are specific to GigE camera operation only.

5.27 IIDC specific structures

These structures are specific to IIDC camera operation only.

Collaboration diagram for IIDC specific structures:

Data Structures

- struct [fc2Format7ImageSettings](#)
Format 7 image settings.
- struct [fc2Format7Info](#)
Format 7 information for a single mode.
- struct [fc2Format7PacketInfo](#)
Format 7 packet information.

5.27.1 Detailed Description

These structures are specific to IIDC camera operation only.

5.28 Image saving structures.

These structures define various parameters used for saving images.

Collaboration diagram for Image saving structures.:

Data Structures

- struct [fc2PNGOption](#)
Options for saving PNG images.
- struct [fc2PPMOption](#)
Options for saving PPM images.
- struct [fc2PGMOption](#)
Options for saving PGM images.
- struct [fc2TIFFOption](#)
Options for saving TIFF images.
- struct [fc2JPEGOption](#)
Options for saving JPEG image.
- struct [fc2JPG2Option](#)
Options for saving JPEG2000 image.
- struct [fc2BMPOption](#)
Options for saving Bitmap image.
- struct [fc2MJPGOption](#)
Options for saving MJPG files.
- struct [fc2H264Option](#)
Options for saving H264 files.
- struct [fc2AVIOption](#)
Options for saving AVI files.
- struct [fc2EventOptions](#)
Options for enabling device event registration.
- struct [fc2EventCallbackData](#)

Typedefs

- typedef void * [fc2CallbackHandle](#)
- typedef void(* [fc2BusEventCallback](#)) (void *pParameter, unsigned int serialNumber)
- typedef void(* [fc2ImageEventCallback](#)) ([fc2Image](#) *image, void *pCallbackData)
- typedef void(* [fc2AsyncCommandCallback](#)) ([fc2Error](#) retError, void *pUserData)
- typedef void(* [fc2CameraEventCallback](#)) (void *pCallbackData)

Enumerations

- enum [fc2TIFFCompressionMethod](#) {
[FC2_TIFF_NONE](#) = 1,
[FC2_TIFF_PACKBITS](#),
[FC2_TIFF_DEFLATE](#),
[FC2_TIFF_ADOBE_DEFLATE](#),
[FC2_TIFF_CCITTFAX3](#),
[FC2_TIFF_CCITTFAX4](#),
[FC2_TIFF_LZW](#),
[FC2_TIFF_JPEG](#) }

5.28.1 Detailed Description

These structures define various parameters used for saving images.

5.28.2 Typedef Documentation

5.28.2.1 `typedef void(* fc2AsyncCommandCallback) (fc2Error retError, void *pUserData)`

5.28.2.2 `typedef void(* fc2BusEventCallback) (void *pParameter, unsigned int serialNumber)`

5.28.2.3 `typedef void* fc2CallbackHandle`

5.28.2.4 `typedef void(* fc2CameraEventCallback) (void *pCallbackData)`

5.28.2.5 `typedef void(* fc2ImageEventCallback) (fc2Image *image, void *pCallbackData)`

5.28.3 Enumeration Type Documentation

5.28.3.1 `enum fc2TIFFCompressionMethod`

Enumerator

FC2_TIFF_NONE Save without any compression.

FC2_TIFF_PACKBITS Save using PACKBITS compression.

FC2_TIFF_DEFLATE Save using DEFLATE compression (ZLIB compression).

FC2_TIFF_ADOBE_DEFLATE Save using ADOBE DEFLATE compression.

FC2_TIFF_CCITTFAX3 Save using CCITT Group 3 fax encoding. This is only valid for 1-bit images only. Default to LZW for other bit depths.

FC2_TIFF_CCITTFAX4 Save using CCITT Group 4 fax encoding. This is only valid for 1-bit images only. Default to LZW for other bit depths.

FC2_TIFF_LZW Save using LZW compression.

FC2_TIFF_JPEG Save using JPEG compression. This is only valid for 8-bit greyscale and 24-bit only. Default to LZW for other bit depths.

Chapter 6

Data Structure Documentation

6.1 fc2AVIOption Struct Reference

Options for saving AVI files.

Data Fields

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

6.1.1 Detailed Description

Options for saving AVI files.

6.1.2 Field Documentation

6.1.2.1 float frameRate

Frame rate of the stream.

6.1.2.2 unsigned int reserved[256]

Reserved for future use.

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

- [FlyCapture2Defs_C.h](#)

6.2 fc2BMPOption Struct Reference

Options for saving Bitmap image.

Data Fields

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

Reserved for future use.

6.2.1 Detailed Description

Options for saving Bitmap image.

6.2.2 Field Documentation

6.2.2.1 **BOOL** indexedColor_8bit

6.2.2.2 unsigned int reserved[16]

Reserved for future use.

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

- [FlyCapture2Defs_C.h](#)

6.3 fc2CameraInfo Struct Reference

Camera information.

Collaboration diagram for fc2CameraInfo:

Data Fields

- unsigned int [serialNumber](#)
Device serial number.
- [fc2InterfaceType](#) [interfaceType](#)
Interface type.
- [fc2DriverType](#) [driverType](#)
Driver type.
- [BOOL](#) [isColorCamera](#)
Flag indicating if this is a color camera.
- char [modelName](#) [MAX_STRING_LENGTH]
Device model name.
- char [vendorName](#) [MAX_STRING_LENGTH]
Device vendor name.
- char [sensorInfo](#) [MAX_STRING_LENGTH]
String detailing the sensor information.
- char [sensorResolution](#) [MAX_STRING_LENGTH]
String providing the sensor resolution.
- char [driverName](#) [MAX_STRING_LENGTH]
Driver name of driver being used.
- char [firmwareVersion](#) [MAX_STRING_LENGTH]
Firmware version of camera.
- char [firmwareBuildTime](#) [MAX_STRING_LENGTH]
Firmware build time.
- [fc2BusSpeed](#) [maximumBusSpeed](#)
Maximum bus speed.
- [fc2BayerTileFormat](#) [bayerTileFormat](#)
Bayer tile format.
- [fc2PCleBusSpeed](#) [pcieBusSpeed](#)
Bus number, set to 0 for GigE and USB cameras.
- unsigned short [nodeNumber](#)
ieee1394 Node number, set to 0 for GigE and USB cameras
- unsigned short [busNumber](#)
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.
- [fc2ConfigROM](#) [configROM](#)
Configuration ROM data.

GigE specific information

- unsigned int [gigEMajorVersion](#)
GigE Vision version.
- unsigned int [gigEMinorVersion](#)
GigE Vision minor version.
- char [userDefinedName](#) [MAX_STRING_LENGTH]
User defined name.

- char `xmlURL1` [`MAX_STRING_LENGTH`]
XML URL 1.
- char `xmlURL2` [`MAX_STRING_LENGTH`]
XML URL 2.
- `fc2MACAddress` `macAddress`
MAC address.
- `fc2IPAddress` `ipAddress`
IP address.
- `fc2IPAddress` `subnetMask`
Subnet mask.
- `fc2IPAddress` `defaultGateway`
Default gateway.
- unsigned int `ccpStatus`
Status/Content of CCP register.
- unsigned int `applicationIPAddress`
Local Application IP Address.
- unsigned int `applicationPort`
Local Application port.

6.3.1 Detailed Description

Camera information.

6.3.2 Field Documentation

6.3.2.1 unsigned int `applicationIPAddress`

Local Application IP Address.

6.3.2.2 unsigned int `applicationPort`

Local Application port.

6.3.2.3 `fc2BayerTileFormat` `bayerTileFormat`

Bayer tile format.

6.3.2.4 unsigned short `busNumber`

PCIe Bus Speed, set to `PCIE_BUSSPEED_UNKNOWN` for unsupported drivers.

6.3.2.5 unsigned int `ccpStatus`

Status/Content of CCP register.

6.3.2.6 fc2ConfigROM configROM

Configuration ROM data.

6.3.2.7 fc2IPAddress defaultGateway

Default gateway.

6.3.2.8 char driverName[MAX_STRING_LENGTH]

Driver name of driver being used.

6.3.2.9 fc2DriverType driverType

Driver type.

6.3.2.10 char firmwareBuildTime[MAX_STRING_LENGTH]

Firmware build time.

6.3.2.11 char firmwareVersion[MAX_STRING_LENGTH]

Firmware version of camera.

6.3.2.12 unsigned int gigEMajorVersion

GigE Vision version.

6.3.2.13 unsigned int gigEMinorVersion

GigE Vision minor version.

6.3.2.14 unsigned int iidcVer

DCAM version.

6.3.2.15 fc2InterfaceType interfaceType

Interface type.

6.3.2.16 fc2IPAddress ipAddress

IP address.

6.3.2.17 BOOL isColorCamera

Flag indicating if this is a color camera.

6.3.2.18 fc2MACAddress macAddress

MAC address.

6.3.2.19 fc2BusSpeed maximumBusSpeed

Maximum bus speed.

6.3.2.20 char modelName[MAX_STRING_LENGTH]

Device model name.

6.3.2.21 unsigned short nodeNumber

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

6.3.2.22 fc2PCleBusSpeed pcieBusSpeed

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

6.3.2.23 unsigned int reserved[16]

Reserved for future use.

6.3.2.24 char sensorInfo[MAX_STRING_LENGTH]

String detailing the sensor information.

6.3.2.25 char sensorResolution[MAX_STRING_LENGTH]

String providing the sensor resolution.

6.3.2.26 unsigned int serialNumber

Device serial number.

6.3.2.27 fc2IPAddress subnetMask

Subnet mask.

6.3.2.28 char userDefinedName[MAX_STRING_LENGTH]

User defined name.

6.3.2.29 char vendorName[MAX_STRING_LENGTH]

Device vendor name.

6.3.2.30 char xmlURL1[MAX_STRING_LENGTH]

XML URL 1.

6.3.2.31 char xmlURL2[MAX_STRING_LENGTH]

XML URL 2.

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

- [FlyCapture2Defs_C.h](#)

6.4 fc2CameraStats Struct Reference

Camera diagnostic information.

Collaboration diagram for fc2CameraStats:

Data Fields

- 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](#)
- [fc2TimeStamp](#) timeStamp
- unsigned int [numResendPacketsRequested](#)
- unsigned int [numResendPacketsReceived](#)
- unsigned int [reserved](#) [16]
 - Reserved for future use.*

6.4.1 Detailed Description

Camera diagnostic information.

6.4.2 Field Documentation

6.4.2.1 float [cameraCurrents](#)[8]

6.4.2.2 **BOOL** [cameraPowerUp](#)

6.4.2.3 float [cameraVoltages](#)[8]

6.4.2.4 unsigned int [imageCorrupt](#)

6.4.2.5 unsigned int [imageDriverDropped](#)

6.4.2.6 unsigned int [imageDropped](#)

6.4.2.7 unsigned int [imageXmitFailed](#)

6.4.2.8 unsigned int [numCurrents](#)

The number of current registers available.

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

6.4.2.9 unsigned int numResendPacketsReceived

6.4.2.10 unsigned int numResendPacketsRequested

6.4.2.11 unsigned int numVoltages

The number of voltage registers available.

0: the values in cameraVoltages[] are invalid.

6.4.2.12 unsigned int portErrors

6.4.2.13 unsigned int regReadFailed

6.4.2.14 unsigned int regWriteFailed

6.4.2.15 unsigned int reserved[16]

Reserved for future use.

6.4.2.16 unsigned int temperature

6.4.2.17 unsigned int timeSinceBusReset

6.4.2.18 unsigned int timeSinceInitialization

6.4.2.19 fc2TimeStamp timeStamp

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

- [FlyCapture2Defs_C.h](#)

6.5 fc2Config Struct Reference

Configuration for a camera.

Data Fields

- 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 re-turning.
- `fc2GrabMode` `grabMode`
Grab mode for the camera.
- `BOOL` `highPerformanceRetrieveBuffer`
This parameter enables `RetrieveBuffer` to run in high performance mode.
- `fc2BusSpeed` `isochBusSpeed`
Isochronous bus speed.
- `fc2BusSpeed` `asyncBusSpeed`
Asynchronous bus speed.
- `fc2BandwidthAllocation` `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.

6.5.1 Detailed Description

Configuration for a camera.

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

6.5.2 Field Documentation

6.5.2.1 `fc2BusSpeed` `asyncBusSpeed`

Asynchronous bus speed.

6.5.2.2 `fc2BandwidthAllocation` `bandwidthAllocation`

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

6.5.2.3 `fc2GrabMode` `grabMode`

Grab mode for the camera.

The default is `DROP_FRAMES`.

6.5.2.4 int grabTimeout

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

6.5.2.5 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.

6.5.2.6 fc2BusSpeed isochBusSpeed

Isochronous bus speed.

6.5.2.7 unsigned int minNumImageNotifications

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

Read-only value.

6.5.2.8 unsigned int numBuffers

Number of buffers used by the FlyCapture2 library to grab images.

6.5.2.9 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.

6.5.2.10 unsigned int registerTimeout

Register read/write timeout value, in microseconds.

The default value is dependent on the interface type.

6.5.2.11 unsigned int registerTimeoutRetries

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

The default value is 0.

6.5.2.12 unsigned int reserved[16]

Reserved for future use.

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

- [FlyCapture2Defs_C.h](#)

6.6 fc2ConfigROM Struct Reference

Camera configuration ROM.

Data Fields

- 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](#) [MAX_STRING_LENGTH]
Keyword.
- unsigned int [reserved](#) [16]
Reserved for future use.

6.6.1 Detailed Description

Camera configuration ROM.

6.6.2 Field Documentation

6.6.2.1 unsigned int chipIdHi

Chip ID (high part).

6.6.2.2 unsigned int chipIdLo

Chip ID (low part).

6.6.2.3 unsigned int nodeVendorId

Vendor ID of a node.

6.6.2.4 char pszKeyword[MAX_STRING_LENGTH]

Keyword.

6.6.2.5 unsigned int reserved[16]

Reserved for future use.

6.6.2.6 unsigned int unitSpecId

Unit Spec ID, usually 0xa02d.

6.6.2.7 unsigned int unitSubSWVer

Unit sub software version.

6.6.2.8 unsigned int unitSWVer

Unit software version.

6.6.2.9 unsigned int vendorUniqueInfo_0

Vendor unique info 0.

6.6.2.10 unsigned int vendorUniqueInfo_1

Vendor unique info 1.

6.6.2.11 unsigned int vendorUniqueInfo_2

Vendor unique info 2.

6.6.2.12 unsigned int vendorUniqueInfo_3

Vendor unique info 3.

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

- [FlyCapture2Defs_C.h](#)

6.7 fc2EmbeddedImageInfo Struct Reference

Properties of the possible embedded image information.

Collaboration diagram for fc2EmbeddedImageInfo:

Data Fields

- [fc2EmbeddedImageInfoProperty timestamp](#)
- [fc2EmbeddedImageInfoProperty gain](#)
- [fc2EmbeddedImageInfoProperty shutter](#)
- [fc2EmbeddedImageInfoProperty brightness](#)
- [fc2EmbeddedImageInfoProperty exposure](#)
- [fc2EmbeddedImageInfoProperty whiteBalance](#)
- [fc2EmbeddedImageInfoProperty frameCounter](#)
- [fc2EmbeddedImageInfoProperty strobePattern](#)
- [fc2EmbeddedImageInfoProperty GPIOPinState](#)
- [fc2EmbeddedImageInfoProperty ROIPosition](#)

6.7.1 Detailed Description

Properties of the possible embedded image information.

6.7.2 Field Documentation

6.7.2.1 fc2EmbeddedImageInfoProperty brightness

6.7.2.2 fc2EmbeddedImageInfoProperty exposure

6.7.2.3 fc2EmbeddedImageInfoProperty frameCounter

6.7.2.4 fc2EmbeddedImageInfoProperty gain

6.7.2.5 fc2EmbeddedImageInfoProperty GPIOPinState

6.7.2.6 fc2EmbeddedImageInfoProperty ROIPosition

6.7.2.7 fc2EmbeddedImageInfoProperty shutter

6.7.2.8 fc2EmbeddedImageInfoProperty strobePattern

6.7.2.9 fc2EmbeddedImageInfoProperty timestamp

6.7.2.10 fc2EmbeddedImageInfoProperty whiteBalance

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

- [FlyCapture2Defs_C.h](#)

6.8 fc2EmbeddedImageInfoProperty Struct Reference

Properties of a single embedded image info property.

Data Fields

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

6.8.1 Detailed Description

Properties of a single embedded image info property.

6.8.2 Field Documentation

6.8.2.1 BOOL available

Whether this property is available.

6.8.2.2 BOOL onOff

Whether this property is on or off.

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

- [FlyCapture2Defs_C.h](#)

6.9 fc2EventCallbackData Struct Reference

Data Fields

- void * [EventUserData](#)
Pointer to the user-supplied data struct.
- size_t [EventUserDataSize](#)
Size of the user data supplied to the RegisterEvent() function.
- const char * [EventName](#)
The event name used to register the event.
- unsigned long long [EventID](#)
The device register which EventName maps to.
- unsigned long long [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.

6.9.1 Field Documentation

6.9.1.1 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.

6.9.1.2 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.

6.9.1.3 unsigned long long EventID

The device register which EventName maps to.

Provides an alternate means of indexing into different event types.

6.9.1.4 const char* EventName

The event name used to register the event.

Provided so the user knows which event triggered the callback.

6.9.1.5 unsigned long long EventTimestamp

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

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

6.9.1.6 void* EventUserData

Pointer to the user-supplied data struct.

6.9.1.7 size_t EventUserDataSize

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

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

- [FlyCapture2Defs_C.h](#)

6.10 fc2EventOptions Struct Reference

Options for enabling device event registration.

Data Fields

- [fc2CameraEventCallback 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.

6.10.1 Detailed Description

Options for enabling device event registration.

6.10.2 Field Documentation

6.10.2.1 `fc2CameraEventCallback` `EventCallbackFcn`

Callback function pointer.

6.10.2.2 `const char*` `EventName`

Event name to register.

6.10.2.3 `const void*` `EventUserData`

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

6.10.2.4 `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_C.h](#)

6.11 `fc2Format7ImageSettings` Struct Reference

Format 7 image settings.

Data Fields

- `fc2Mode` `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.
- `fc2PixelFormat` `pixelFormat`
Pixel format of image.
- unsigned int `reserved` [8]
Reserved for future use.

6.11.1 Detailed Description

Format 7 image settings.

6.11.2 Field Documentation

6.11.2.1 unsigned int height

Height of image.

6.11.2.2 fc2Mode mode

Format 7 mode.

6.11.2.3 unsigned int offsetX

Horizontal image offset.

6.11.2.4 unsigned int offsetY

Vertical image offset.

6.11.2.5 fc2PixelFormat pixelFormat

Pixel format of image.

6.11.2.6 unsigned int reserved[8]

Reserved for future use.

6.11.2.7 unsigned int width

Width of image.

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

- [FlyCapture2Defs_C.h](#)

6.12 fc2Format7Info Struct Reference

Format 7 information for a single mode.

Data Fields

- [fc2Mode 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.

6.12.1 Detailed Description

Format 7 information for a single mode.

6.12.2 Field Documentation

6.12.2.1 unsigned int imageHStepSize

Horizontal step size for the image.

6.12.2.2 unsigned int imageVStepSize

Vertical step size for the image.

6.12.2.3 unsigned int maxHeight

Maximum image height.

6.12.2.4 unsigned int maxPacketSize

Maximum packet size in bytes for current mode.

6.12.2.5 unsigned int maxWidth

Maximum image width.

6.12.2.6 unsigned int minPacketSize

Minimum packet size in bytes for current mode.

6.12.2.7 fc2Mode mode

Format 7 mode.

6.12.2.8 unsigned int offsetHStepSize

Horizontal step size for the offset.

6.12.2.9 unsigned int offsetVStepSize

Vertical step size for the offset.

6.12.2.10 unsigned int packetSize

Current packet size in bytes.

6.12.2.11 float percentage

Current packet size as a percentage of maximum packet size.

6.12.2.12 unsigned int pixelFormatBitField

Supported pixel formats in a bit field.

6.12.2.13 unsigned int reserved[16]

Reserved for future use.

6.12.2.14 unsigned int vendorPixelFormatBitField

Vendor unique pixel formats in a bit field.

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

- [FlyCapture2Defs_C.h](#)

6.13 fc2Format7PacketInfo Struct Reference

Format 7 packet information.

Data Fields

- 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.

6.13.1 Detailed Description

Format 7 packet information.

6.13.2 Field Documentation

6.13.2.1 unsigned int maxBytesPerPacket

Maximum bytes per packet.

6.13.2.2 unsigned int recommendedBytesPerPacket

Recommended bytes per packet.

6.13.2.3 unsigned int reserved[8]

Reserved for future use.

6.13.2.4 unsigned int unitBytesPerPacket

Minimum bytes per packet.

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

- [FlyCapture2Defs_C.h](#)

6.14 fc2GigEConfig Struct Reference

Configuration for a GigE camera.

Data Fields

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

6.14.1 Detailed Description

Configuration for a GigE camera.

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

6.14.2 Field Documentation

6.14.2.1 BOOL enablePacketResend

Turn on/off packet resend functionality.

6.14.2.2 unsigned int registerTimeout

Register read/write timeout value, in microseconds.

The default value is dependent on the interface type.

6.14.2.3 unsigned int registerTimeoutRetries

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

The default value is 0.

6.14.2.4 unsigned int reserved[8]

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

- [FlyCapture2Defs_C.h](#)

6.15 fc2GigEImageSettings Struct Reference

Image settings for a GigE camera.

Data Fields

- unsigned int [offsetX](#)
Horizontal image offset.
- unsigned int [offsetY](#)
Vertical image offset.
- unsigned int [width](#)
Width of image.
- unsigned int [height](#)
Height of image.
- [fc2PixelFormat](#) [pixelFormat](#)
Pixel format of image.
- unsigned int [reserved](#) [8]
Reserved for future use.

6.15.1 Detailed Description

Image settings for a GigE camera.

6.15.2 Field Documentation

6.15.2.1 unsigned int height

Height of image.

6.15.2.2 unsigned int offsetX

Horizontal image offset.

6.15.2.3 unsigned int offsetY

Vertical image offset.

6.15.2.4 fc2PixelFormat pixelFormat

Pixel format of image.

6.15.2.5 unsigned int reserved[8]

Reserved for future use.

6.15.2.6 unsigned int width

Width of image.

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

- [FlyCapture2Defs_C.h](#)

6.16 fc2GigElmageSettingsInfo Struct Reference

Format 7 information for a single mode.

Data Fields

- 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.

6.16.1 Detailed Description

Format 7 information for a single mode.

6.16.2 Field Documentation

6.16.2.1 unsigned int imageHStepSize

Horizontal step size for the image.

6.16.2.2 unsigned int imageVStepSize

Vertical step size for the image.

6.16.2.3 unsigned int maxHeight

Maximum image height.

6.16.2.4 unsigned int maxWidth

Maximum image width.

6.16.2.5 unsigned int offsetHStepSize

Horizontal step size for the offset.

6.16.2.6 unsigned int offsetVStepSize

Vertical step size for the offset.

6.16.2.7 unsigned int pixelFormatBitField

Supported pixel formats in a bit field.

6.16.2.8 unsigned int reserved[16]

Reserved for future use.

6.16.2.9 unsigned int vendorPixelFormatBitField

Vendor unique pixel formats in a bit field.

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

- [FlyCapture2Defs_C.h](#)

6.17 fc2GigEProperty Struct Reference

A GigE property.

Data Fields

- [fc2GigEPropertyType 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.
- unsigned int [reserved](#) [8]

6.17.1 Detailed Description

A GigE property.

6.17.2 Field Documentation

6.17.2.1 BOOL isReadable

Whether the property is readable.

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

6.17.2.2 BOOL isWritable

Whether the property is writable.

6.17.2.3 unsigned int max

Maximum value.

6.17.2.4 unsigned int min

Minimum value.

6.17.2.5 fc2GigEPropertyType propType

The type of property.

6.17.2.6 unsigned int reserved[8]

6.17.2.7 unsigned int value

Current value.

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

- [FlyCapture2Defs_C.h](#)

6.18 fc2GigEStreamChannel Struct Reference

Information about a single GigE stream channel.

Collaboration diagram for fc2GigEStreamChannel:

Data Fields

- 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.
- [fc2IPAddress destinationIpAddress](#)
Destination IP address.
- unsigned int [sourcePort](#)
Source UDP port of the stream channel.
- unsigned int [reserved](#) [8]

6.18.1 Detailed Description

Information about a single GigE stream channel.

6.18.2 Field Documentation

6.18.2.1 **fc2IPAddress** destinationIpAddress

Destination IP address.

It can be a multicast or unicast address.

6.18.2.2 **BOOL** doNotFragment

Disable IP fragmentation of packets.

6.18.2.3 **unsigned int** hostPort

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

6.18.2.4 **unsigned int** interPacketDelay

Inter packet delay, in timestamp counter units.

6.18.2.5 **unsigned int** networkInterfaceIndex

Network interface index used (or to use).

6.18.2.6 **unsigned int** packetSize

Packet size, in bytes.

6.18.2.7 **unsigned int** reserved[8]

6.18.2.8 **unsigned int** sourcePort

Source UDP port of the stream channel.

Read only.

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

- [FlyCapture2Defs_C.h](#)

6.19 fc2H264Option Struct Reference

Options for saving H264 files.

Data Fields

- 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.

6.19.1 Detailed Description

Options for saving H264 files.

6.19.2 Field Documentation

6.19.2.1 unsigned int bitrate

Bitrate to encode at.

6.19.2.2 float frameRate

Frame rate of the stream.

6.19.2.3 unsigned int height

Height of source image.

6.19.2.4 unsigned int reserved[256]

Reserved for future use.

6.19.2.5 unsigned int width

Width of source image.

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

- [FlyCapture2Defs_C.h](#)

6.20 fc2Image Struct Reference

Data Fields

- unsigned int [rows](#)
- unsigned int [cols](#)
- unsigned int [stride](#)
- unsigned char * [pData](#)
- unsigned int [dataSize](#)
- unsigned int [receivedDataSize](#)
- [fc2PixelFormat](#) format
- [fc2BayerTileFormat](#) bayerFormat
- [fc2ImageImpl](#) imageImpl

6.20.1 Field Documentation

6.20.1.1 [fc2BayerTileFormat](#) bayerFormat

6.20.1.2 unsigned int cols

6.20.1.3 unsigned int dataSize

6.20.1.4 [fc2PixelFormat](#) format

6.20.1.5 [fc2ImageImpl](#) imageImpl

6.20.1.6 unsigned char* pData

6.20.1.7 unsigned int receivedDataSize

6.20.1.8 unsigned int rows

6.20.1.9 unsigned int stride

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

- [FlyCapture2Defs_C.h](#)

6.21 fc2ImageMetadata Struct Reference

Metadata related to an image.

Data Fields

- 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.

6.21.1 Detailed Description

Metadata related to an image.

6.21.2 Field Documentation

6.21.2.1 unsigned int `embeddedBrightness`

Embedded brightness.

6.21.2.2 unsigned int `embeddedExposure`

Embedded exposure.

6.21.2.3 unsigned int `embeddedFrameCounter`

Embedded frame counter.

6.21.2.4 unsigned int `embeddedGain`

Embedded gain.

6.21.2.5 unsigned int embeddedGPIOPinState

Embedded GPIO pin state.

6.21.2.6 unsigned int embeddedROIPosition

Embedded ROI position.

6.21.2.7 unsigned int embeddedShutter

Embedded shutter.

6.21.2.8 unsigned int embeddedStrobePattern

Embedded strobe pattern.

6.21.2.9 unsigned int embeddedTimeStamp

Embedded timestamp.

6.21.2.10 unsigned int embeddedWhiteBalance

Embedded white balance.

6.21.2.11 unsigned int reserved[31]

Reserved for future use.

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

- [FlyCapture2Defs_C.h](#)

6.22 fc2InternalContext Struct Reference

Data Fields

- FlyCapture2::BusManager * [pBusMgr](#)
- FlyCapture2::CameraBase * [pCamera](#)

6.22.1 Field Documentation

6.22.1.1 FlyCapture2::BusManager* pBusMgr

6.22.1.2 FlyCapture2::CameraBase* pCamera

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

- [FlyCapture2Internal_C.h](#)

6.23 fc2InternalGuiContext Struct Reference

Data Fields

- FlyCapture2::CameraSelectionDlg * [pCameraSelectionDlg](#)
- FlyCapture2::CameraControlDlg * [pCameraControlDlg](#)

6.23.1 Field Documentation

6.23.1.1 FlyCapture2::CameraControlDlg* pCameraControlDlg

6.23.1.2 FlyCapture2::CameraSelectionDlg* pCameraSelectionDlg

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

- [FlyCapture2Internal_C.h](#)

6.24 fc2InternalImageCallback Struct Reference

Collaboration diagram for fc2InternalImageCallback:

Data Fields

- [fc2ImageEventCallback](#) pCallback
- void * [pCallbackData](#)

6.24.1 Field Documentation

6.24.1.1 fc2ImageEventCallback pCallback

6.24.1.2 void* pCallbackData

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

- [FlyCapture2Internal_C.h](#)

6.25 fc2IPAddress Struct Reference

IPv4 address.

Data Fields

- unsigned char [octets](#) [4]

6.25.1 Detailed Description

IPv4 address.

6.25.2 Field Documentation

6.25.2.1 unsigned char octets[4]

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

- [FlyCapture2Defs_C.h](#)

6.26 fc2JPEGOption Struct Reference

Options for saving JPEG image.

Data Fields

- [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.

6.26.1 Detailed Description

Options for saving JPEG image.

6.26.2 Field Documentation

6.26.2.1 BOOL progressive

Whether to save as a progressive JPEG file.

6.26.2.2 unsigned int quality

JPEG image quality in range (0-100).

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

6.26.2.3 unsigned int reserved[16]

Reserved for future use.

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

- [FlyCapture2Defs_C.h](#)

6.27 fc2JPG2Option Struct Reference

Options for saving JPEG2000 image.

Data Fields

- unsigned int [quality](#)
JPEG saving quality in range (1-512).
- unsigned int [reserved](#) [16]
Reserved for future use.

6.27.1 Detailed Description

Options for saving JPEG2000 image.

6.27.2 Field Documentation

6.27.2.1 unsigned int quality

JPEG saving quality in range (1-512).

6.27.2.2 unsigned int reserved[16]

Reserved for future use.

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

- [FlyCapture2Defs_C.h](#)

6.28 fc2LUTData Struct Reference

Information about the camera's look up table.

Data Fields

- [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.

6.28.1 Detailed Description

Information about the camera's look up table.

6.28.2 Field Documentation

6.28.2.1 BOOL enabled

Flag indicating if LUT is enabled.

6.28.2.2 unsigned int inputBitDepth

The input bit depth of the LUT.

6.28.2.3 unsigned int numBanks

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

6.28.2.4 unsigned int numChannels

The number of LUT channels per bank available.

6.28.2.5 unsigned int numEntries

The number of entries in the LUT.

6.28.2.6 unsigned int outputBitDepth

The output bit depth of the LUT.

6.28.2.7 unsigned int reserved[8]

Reserved for future use.

6.28.2.8 BOOL supported

Flag indicating if LUT is supported.

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

- [FlyCapture2Defs_C.h](#)

6.29 fc2MACAddress Struct Reference

MAC address.

Data Fields

- unsigned char [octets](#) [6]

6.29.1 Detailed Description

MAC address.

6.29.2 Field Documentation

6.29.2.1 unsigned char octets[6]

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

- [FlyCapture2Defs_C.h](#)

6.30 fc2MJPEGOption Struct Reference

Options for saving MJPG files.

Data Fields

- float [frameRate](#)
Frame rate of the stream.
- unsigned int [quality](#)
Image quality (1-100)
- unsigned int [reserved](#) [256]

6.30.1 Detailed Description

Options for saving MJPG files.

6.30.2 Field Documentation

6.30.2.1 float frameRate

Frame rate of the stream.

6.30.2.2 unsigned int quality

Image quality (1-100)

6.30.2.3 unsigned int reserved[256]

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

- [FlyCapture2Defs_C.h](#)

6.31 fc2PGMOption Struct Reference

Options for saving PGM images.

Data Fields

- [BOOL binaryFile](#)
Whether to save the PPM as a binary file.
- unsigned int [reserved](#) [16]
Reserved for future use.

6.31.1 Detailed Description

Options for saving PGM images.

6.31.2 Field Documentation

6.31.2.1 **BOOL** binaryFile

Whether to save the PPM as a binary file.

6.31.2.2 unsigned int reserved[16]

Reserved for future use.

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

- [FlyCapture2Defs_C.h](#)

6.32 fc2PGRGuid Struct Reference

A GUID to the camera.

Data Fields

- unsigned int [value](#) [4]

6.32.1 Detailed Description

A GUID to the camera.

It is used to uniquely identify a camera.

6.32.2 Field Documentation

6.32.2.1 unsigned int value[4]

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

- [FlyCapture2Defs_C.h](#)

6.33 fc2PNGOption Struct Reference

Options for saving PNG images.

Data Fields

- [BOOL interlaced](#)
Whether to save the PNG as interlaced.
- unsigned int [compressionLevel](#)
Compression level (0-9).
- unsigned int [reserved](#) [16]
Reserved for future use.

6.33.1 Detailed Description

Options for saving PNG images.

6.33.2 Field Documentation

6.33.2.1 unsigned int compressionLevel

Compression level (0-9).

0 is no compression, 9 is best compression.

6.33.2.2 BOOL interlaced

Whether to save the PNG as interlaced.

6.33.2.3 unsigned int reserved[16]

Reserved for future use.

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

- [FlyCapture2Defs_C.h](#)

6.34 fc2PPMOption Struct Reference

Options for saving PPM images.

Data Fields

- [BOOL binaryFile](#)
Whether to save the PPM as a binary file.
- unsigned int [reserved](#) [16]
Reserved for future use.

6.34.1 Detailed Description

Options for saving PPM images.

6.34.2 Field Documentation

6.34.2.1 **BOOL** binaryFile

Whether to save the PPM as a binary file.

6.34.2.2 unsigned int reserved[16]

Reserved for future use.

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

- [FlyCapture2Defs_C.h](#)

6.35 fc2StrobeControl Struct Reference

A camera strobe.

Data Fields

- 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.

6.35.1 Detailed Description

A camera strobe.

6.35.2 Field Documentation

6.35.2.1 float delay

Signal delay (in ms).

6.35.2.2 float duration

Signal duration (in ms).

6.35.2.3 BOOL onOff

Flag controlling on/off.

6.35.2.4 unsigned int polarity

Signal polarity.

6.35.2.5 unsigned int reserved[8]

Reserved for future use.

6.35.2.6 unsigned int source

Source value.

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

- [FlyCapture2Defs_C.h](#)

6.36 fc2StrobeInfo Struct Reference

A camera strobe property.

Data Fields

- 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.

6.36.1 Detailed Description

A camera strobe property.

6.36.2 Field Documentation

6.36.2.1 float [maxValue](#)

Maximum value.

6.36.2.2 float [minValue](#)

Minimum value.

6.36.2.3 [BOOL onOffSupported](#)

Flag indicating if on/off is supported.

6.36.2.4 [BOOL polaritySupported](#)

Flag indicating if polarity is supported.

6.36.2.5 [BOOL present](#)

Presence of strobe.

6.36.2.6 **BOOL** readOutSupported

Flag indicating if strobe value can be read out.

6.36.2.7 **unsigned int** reserved[8]

Reserved for future use.

6.36.2.8 **unsigned int** source

Source value.

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

- [FlyCapture2Defs_C.h](#)

6.37 fc2SystemInfo Struct Reference

Description of the system.

Data Fields

- [fc2OSType](#) **osType**
Operating system type as described by OSType.
- **char** [osDescription](#) [MAX_STRING_LENGTH]
Detailed description of the operating system.
- [fc2ByteOrder](#) **byteOrder**
Byte order of the system.
- **size_t** [sysMemSize](#)
Amount of memory available on the system.
- **char** [cpuDescription](#) [MAX_STRING_LENGTH]
Detailed description of the CPU.
- **size_t** [numCpuCores](#)
Number of cores on all CPUs on the system.
- **char** [driverList](#) [MAX_STRING_LENGTH]
List of drivers used.
- **char** [libraryList](#) [MAX_STRING_LENGTH]
List of libraries used.
- **char** [gpuDescription](#) [MAX_STRING_LENGTH]
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.

6.37.1 Detailed Description

Description of the system.

6.37.2 Field Documentation

6.37.2.1 `fc2ByteOrder` `byteOrder`

Byte order of the system.

6.37.2.2 `char cpuDescription[MAX_STRING_LENGTH]`

Detailed description of the CPU.

6.37.2.3 `char driverList[MAX_STRING_LENGTH]`

List of drivers used.

6.37.2.4 `char gpuDescription[MAX_STRING_LENGTH]`

Detailed description of the GPU.

6.37.2.5 `char libraryList[MAX_STRING_LENGTH]`

List of libraries used.

6.37.2.6 `size_t numCpuCores`

Number of cores on all CPUs on the system.

6.37.2.7 `char osDescription[MAX_STRING_LENGTH]`

Detailed description of the operating system.

6.37.2.8 `fc2OSType` `osType`

Operating system type as described by `OSType`.

6.37.2.9 `unsigned int reserved[16]`

Reserved for future use.

6.37.2.10 size_t screenHeight

Screen resolution height in pixels.

6.37.2.11 size_t screenWidth

Screen resolution width in pixels.

6.37.2.12 size_t sysMemSize

Amount of memory available on the system.

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

- [FlyCapture2Defs_C.h](#)

6.38 fc2TIFFOption Struct Reference

Options for saving TIFF images.

Data Fields

- [fc2TIFFCompressionMethod compression](#)
Compression method to use for encoding TIFF images.
- unsigned int [reserved](#) [16]
Reserved for future use.

6.38.1 Detailed Description

Options for saving TIFF images.

6.38.2 Field Documentation

6.38.2.1 fc2TIFFCompressionMethod compression

Compression method to use for encoding TIFF images.

6.38.2.2 unsigned int reserved[16]

Reserved for future use.

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

- [FlyCapture2Defs_C.h](#)

6.39 fc2TimeStamp Struct Reference

Timestamp information.

Data Fields

- 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.

6.39.1 Detailed Description

Timestamp information.

6.39.2 Field Documentation

6.39.2.1 unsigned int cycleCount

1394 cycle time count.

6.39.2.2 unsigned int cycleOffset

1394 cycle time offset.

6.39.2.3 unsigned int cycleSeconds

1394 cycle time seconds.

6.39.2.4 unsigned int microSeconds

Microseconds.

6.39.2.5 unsigned int reserved[8]

Reserved for future use.

6.39.2.6 long long seconds

Seconds.

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

- [FlyCapture2Defs_C.h](#)

6.40 fc2TriggerDelay Struct Reference

A specific camera property.

Data Fields

- [fc2PropertyType](#) 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.

6.40.1 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

6.40.2 Field Documentation

6.40.2.1 **BOOL** absControl

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

6.40.2.2 **float** absValue

Floating point value.

Used to configure properties in absolute mode.

6.40.2.3 **BOOL** autoManualMode

Flag controlling auto.

6.40.2.4 **BOOL** onePush

Flag controlling one push.

6.40.2.5 **BOOL** onOff

Flag controlling on/off.

6.40.2.6 **BOOL** present

Flag indicating if the property is present.

6.40.2.7 **unsigned int** reserved[8]

Reserved for future use.

6.40.2.8 **fc2PropertyType** type

Property info type.

6.40.2.9 **unsigned int** valueA

Value A (integer).

Used to configure properties in non-absolute mode.

6.40.2.10 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_C.h](#)

6.41 fc2TriggerDelayInfo Struct Reference

Information about a specific camera property.

Data Fields

- [fc2PropertyType](#) 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](#) [MAX_STRING_LENGTH]
Textual description of units.
- char [pUnitAbbr](#) [MAX_STRING_LENGTH]
Abbreviated textual description of units.
- unsigned int [reserved](#) [8]
Reserved for future use.

6.41.1 Detailed Description

Information about a specific camera property.

This structure is also also used as the TriggerDelayInfo structure.

6.41.2 Field Documentation

6.41.2.1 float absMax

Maximum value (as a floating point value).

6.41.2.2 float absMin

Minimum value (as a floating point value).

6.41.2.3 BOOL absValSupported

Flag indicating if absolute mode is supported.

6.41.2.4 BOOL autoSupported

Flag indicating if auto is supported.

6.41.2.5 BOOL manualSupported

Flag indicating if manual is supported.

6.41.2.6 unsigned int max

Maximum value (as an integer).

6.41.2.7 unsigned int min

Minimum value (as an integer).

6.41.2.8 BOOL onePushSupported

Flag indicating if one push is supported.

6.41.2.9 BOOL onOffSupported

Flag indicating if on/off is supported.

6.41.2.10 BOOL present

Flag indicating if the property is present.

6.41.2.11 char pUnitAbbr[MAX_STRING_LENGTH]

Abbreviated textual description of units.

6.41.2.12 char pUnits[MAX_STRING_LENGTH]

Textual description of units.

6.41.2.13 BOOL readOutSupported

Flag indicating if property value can be read out.

6.41.2.14 unsigned int reserved[8]

Reserved for future use.

6.41.2.15 fc2PropertyType type

Property info type.

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

- [FlyCapture2Defs_C.h](#)

6.42 fc2TriggerMode Struct Reference

A camera trigger.

Data Fields

- **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.

6.42.1 Detailed Description

A camera trigger.

6.42.2 Field Documentation

6.42.2.1 unsigned int mode

Mode value.

6.42.2.2 BOOL onOff

Flag controlling on/off.

6.42.2.3 unsigned int parameter

Parameter value.

6.42.2.4 unsigned int polarity

Polarity value.

6.42.2.5 unsigned int reserved[8]

Reserved for future use.

6.42.2.6 unsigned int source

Source value.

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

- [FlyCapture2Defs_C.h](#)

6.43 fc2TriggerModelInfo Struct Reference

Information about a camera trigger property.

Data Fields

- [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.

6.43.1 Detailed Description

Information about a camera trigger property.

6.43.2 Field Documentation

6.43.2.1 unsigned int modeMask

Mode mask.

6.43.2.2 **BOOL** onOffSupported

Flag indicating if on/off is supported.

6.43.2.3 **BOOL** polaritySupported

Flag indicating if polarity is supported.

6.43.2.4 **BOOL** present

Presence of trigger mode.

6.43.2.5 **BOOL** readOutSupported

Flag indicating if trigger value can be read out.

6.43.2.6 **unsigned int** reserved[8]

Reserved for future use.

6.43.2.7 **BOOL** softwareTriggerSupported

Flag indicating if software trigger is supported.

6.43.2.8 **unsigned int** sourceMask

Source mask.

6.43.2.9 **BOOL** valueReadable

Flag indicating if the value is readable.

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

- [FlyCapture2Defs_C.h](#)

6.44 **fc2Version** Struct Reference

The current version of the library.

Data Fields

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

6.44.1 Detailed Description

The current version of the library.

6.44.2 Field Documentation

6.44.2.1 unsigned int build

Build version number.

6.44.2.2 unsigned int major

Major version number.

6.44.2.3 unsigned int minor

Minor version number.

6.44.2.4 unsigned int type

Type version number.

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

- [FlyCapture2Defs_C.h](#)

Chapter 7

File Documentation

7.1 FlyCapture2_C.h File Reference

Include dependency graph for FlyCapture2_C.h:

7.2 FlyCapture2Defs_C.h File Reference

Include dependency graph for FlyCapture2Defs_C.h: This graph shows which files directly or indirectly include this file:

Data Structures

- struct [fc2PGRGuid](#)
A GUID to the camera.
- struct [fc2Image](#)
- struct [fc2SystemInfo](#)
Description of the system.
- struct [fc2Version](#)
The current version of the library.
- struct [fc2IPAddress](#)
IPv4 address.
- struct [fc2MACAddress](#)
MAC address.
- struct [fc2GigEProperty](#)
A GigE property.
- struct [fc2GigEStreamChannel](#)
Information about a single GigE stream channel.
- struct [fc2GigEConfig](#)
Configuration for a GigE camera.
- struct [fc2GigEImageSettingsInfo](#)
Format 7 information for a single mode.
- struct [fc2GigEImageSettings](#)
Image settings for a GigE camera.

- struct [fc2Format7ImageSettings](#)
Format 7 image settings.
- struct [fc2Format7Info](#)
Format 7 information for a single mode.
- struct [fc2Format7PacketInfo](#)
Format 7 packet information.
- struct [fc2Config](#)
Configuration for a camera.
- struct [fc2TriggerDelayInfo](#)
Information about a specific camera property.
- struct [fc2TriggerDelay](#)
A specific camera property.
- struct [fc2TriggerModelInfo](#)
Information about a camera trigger property.
- struct [fc2TriggerMode](#)
A camera trigger.
- struct [fc2StrobeInfo](#)
A camera strobe property.
- struct [fc2StrobeControl](#)
A camera strobe.
- struct [fc2TimeStamp](#)
Timestamp information.
- struct [fc2ConfigROM](#)
Camera configuration ROM.
- struct [fc2CameraInfo](#)
Camera information.
- struct [fc2EmbeddedImageInfoProperty](#)
Properties of a single embedded image info property.
- struct [fc2EmbeddedImageInfo](#)
Properties of the possible embedded image information.
- struct [fc2ImageMetadata](#)
Metadata related to an image.
- struct [fc2LUTData](#)
Information about the camera's look up table.
- struct [fc2CameraStats](#)
Camera diagnostic information.
- struct [fc2PNGOption](#)
Options for saving PNG images.
- struct [fc2PPMOption](#)
Options for saving PPM images.
- struct [fc2PGMOption](#)
Options for saving PGM images.
- struct [fc2TIFFOption](#)
Options for saving TIFF images.
- struct [fc2JPEGOption](#)
Options for saving JPEG image.
- struct [fc2JPG2Option](#)
Options for saving JPEG2000 image.
- struct [fc2BMPOption](#)
Options for saving Bitmap image.
- struct [fc2MJPGOption](#)

Options for saving MJPG files.

- struct [fc2H264Option](#)

Options for saving H264 files.

- struct [fc2AVIOption](#)

Options for saving AVI files.

- struct [fc2EventOptions](#)

Options for enabling device event registration.

- struct [fc2EventCallbackData](#)

Macros

- #define [FALSE](#) 0
- #define [TRUE](#) 1
- #define [FULL_32BIT_VALUE](#) 0xFFFFFFFF
- #define [MAX_STRING_LENGTH](#) 512

Typedefs

- typedef int [BOOL](#)
- typedef void * [fc2Context](#)
A context to the FlyCapture2 C library.
- typedef void * [fc2GuiContext](#)
A context to the FlyCapture2 C GUI library.
- typedef void * [fc2ImageImpl](#)
An internal pointer used in the [fc2Image](#) structure.
- typedef void * [fc2AVIContext](#)
A context referring to the AVI recorder object.
- typedef void * [fc2ImageStatisticsContext](#)
A context referring to the ImageStatistics object.
- typedef void * [fc2TopologyNodeContext](#)
A context referring to the TopologyNode object.
- typedef void * [fc2CallbackHandle](#)
- typedef void(* [fc2BusEventCallback](#)) (void *pParameter, unsigned int serialNumber)
- typedef void(* [fc2ImageEventCallback](#)) ([fc2Image](#) *image, void *pCallbackData)
- typedef void(* [fc2AsyncCommandCallback](#)) ([fc2Error](#) retError, void *pUserData)
- typedef void(* [fc2CameraEventCallback](#)) (void *pCallbackData)

Enumerations

- enum `fc2Error` {
`FC2_ERROR_UNDEFINED` = -1,
`FC2_ERROR_OK`,
`FC2_ERROR_FAILED`,
`FC2_ERROR_NOT_IMPLEMENTED`,
`FC2_ERROR_FAILED_BUS_MASTER_CONNECTION`,
`FC2_ERROR_NOT_CONNECTED`,
`FC2_ERROR_INIT_FAILED`,
`FC2_ERROR_NOT_INITIALIZED`,
`FC2_ERROR_INVALID_PARAMETER`,
`FC2_ERROR_INVALID_SETTINGS`,
`FC2_ERROR_INVALID_BUS_MANAGER`,
`FC2_ERROR_MEMORY_ALLOCATION_FAILED`,
`FC2_ERROR_LOW_LEVEL_FAILURE`,
`FC2_ERROR_NOT_FOUND`,
`FC2_ERROR_FAILED_GUID`,
`FC2_ERROR_INVALID_PACKET_SIZE`,
`FC2_ERROR_INVALID_MODE`,
`FC2_ERROR_NOT_IN_FORMAT7`,
`FC2_ERROR_NOT_SUPPORTED`,
`FC2_ERROR_TIMEOUT`,
`FC2_ERROR_BUS_MASTER_FAILED`,
`FC2_ERROR_INVALID_GENERATION`,
`FC2_ERROR_LUT_FAILED`,
`FC2_ERROR_IIDC_FAILED`,
`FC2_ERROR_STROBE_FAILED`,
`FC2_ERROR_TRIGGER_FAILED`,
`FC2_ERROR_PROPERTY_FAILED`,
`FC2_ERROR_PROPERTY_NOT_PRESENT`,
`FC2_ERROR_REGISTER_FAILED`,
`FC2_ERROR_READ_REGISTER_FAILED`,
`FC2_ERROR_WRITE_REGISTER_FAILED`,
`FC2_ERROR_ISOCH_FAILED`,
`FC2_ERROR_ISOCH_ALREADY_STARTED`,
`FC2_ERROR_ISOCH_NOT_STARTED`,
`FC2_ERROR_ISOCH_START_FAILED`,
`FC2_ERROR_ISOCH_RETRIEVE_BUFFER_FAILED`,
`FC2_ERROR_ISOCH_STOP_FAILED`,
`FC2_ERROR_ISOCH_SYNC_FAILED`,
`FC2_ERROR_ISOCH_BANDWIDTH_EXCEEDED`,
`FC2_ERROR_IMAGE_CONVERSION_FAILED`,
`FC2_ERROR_IMAGE_LIBRARY_FAILURE`,
`FC2_ERROR_BUFFER_TOO_SMALL`,
`FC2_ERROR_IMAGE_CONSISTENCY_ERROR`,
`FC2_ERROR_INCOMPATIBLE_DRIVER`,
`FC2_ERROR_FORCE_32BITS` = `FULL_32BIT_VALUE` }

The error types returned by functions.

- enum `fc2BusCallbackType` {
`FC2_BUS_RESET`,
`FC2_ARRIVAL`,
`FC2_REMOVAL`,
`FC2_CALLBACK_TYPE_FORCE_32BITS` = `FULL_32BIT_VALUE` }

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

- enum `fc2GrabMode` {
`FC2_DROP_FRAMES`,
`FC2_BUFFER_FRAMES`,
`FC2_UNSPECIFIED_GRAB_MODE`,

```
FC2_GRAB_MODE_FORCE_32BITS = FULL_32BIT_VALUE }
```

The grab strategy employed during image transfer.

- enum `fc2GrabTimeout` {
`FC2_TIMEOUT_NONE` = 0,
`FC2_TIMEOUT_INFINITE` = -1,
`FC2_TIMEOUT_UNSPECIFIED` = -2,
`FC2_GRAB_TIMEOUT_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Timeout options for grabbing images.

- enum `fc2BandwidthAllocation` {
`FC2_BANDWIDTH_ALLOCATION_OFF` = 0,
`FC2_BANDWIDTH_ALLOCATION_ON` = 1,
`FC2_BANDWIDTH_ALLOCATION_UNSUPPORTED` = 2,
`FC2_BANDWIDTH_ALLOCATION_UNSPECIFIED` = 3,
`FC2_BANDWIDTH_ALLOCATION_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Bandwidth allocation options for 1394 devices.

- enum `fc2InterfaceType` {
`FC2_INTERFACE_IEEE1394`,
`FC2_INTERFACE_USB_2`,
`FC2_INTERFACE_USB_3`,
`FC2_INTERFACE_GIGE`,
`FC2_INTERFACE_UNKNOWN`,
`FC2_INTERFACE_TYPE_FORCE_32BITS` = `FULL_32BIT_VALUE` }

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

- enum `fc2PropertyType` {
`FC2_BRIGHTNESS`,
`FC2_AUTO_EXPOSURE`,
`FC2_SHARPNESS`,
`FC2_WHITE_BALANCE`,
`FC2_HUE`,
`FC2_SATURATION`,
`FC2_GAMMA`,
`FC2_IRIS`,
`FC2_FOCUS`,
`FC2_ZOOM`,
`FC2_PAN`,
`FC2_TILT`,
`FC2_SHUTTER`,
`FC2_GAIN`,
`FC2_TRIGGER_MODE`,
`FC2_TRIGGER_DELAY`,
`FC2_FRAME_RATE`,
`FC2_TEMPERATURE`,
`FC2_UNSPECIFIED_PROPERTY_TYPE`,
`FC2_PROPERTY_TYPE_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Camera properties.

- enum `fc2FrameRate` {
`FC2_FRAMERATE_1_875`,
`FC2_FRAMERATE_3_75`,
`FC2_FRAMERATE_7_5`,
`FC2_FRAMERATE_15`,
`FC2_FRAMERATE_30`,
`FC2_FRAMERATE_60`,
`FC2_FRAMERATE_120`,
`FC2_FRAMERATE_240`,
`FC2_FRAMERATE_FORMAT7`,
`FC2_NUM_FRAMERATES`,
`FC2_FRAMERATE_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Frame rates in frames per second.

- enum `fc2VideoMode` {
 - `FC2_VIDEOMODE_160x120YUV444,`
 - `FC2_VIDEOMODE_320x240YUV422,`
 - `FC2_VIDEOMODE_640x480YUV411,`
 - `FC2_VIDEOMODE_640x480YUV422,`
 - `FC2_VIDEOMODE_640x480RGB,`
 - `FC2_VIDEOMODE_640x480Y8,`
 - `FC2_VIDEOMODE_640x480Y16,`
 - `FC2_VIDEOMODE_800x600YUV422,`
 - `FC2_VIDEOMODE_800x600RGB,`
 - `FC2_VIDEOMODE_800x600Y8,`
 - `FC2_VIDEOMODE_800x600Y16,`
 - `FC2_VIDEOMODE_1024x768YUV422,`
 - `FC2_VIDEOMODE_1024x768RGB,`
 - `FC2_VIDEOMODE_1024x768Y8,`
 - `FC2_VIDEOMODE_1024x768Y16,`
 - `FC2_VIDEOMODE_1280x960YUV422,`
 - `FC2_VIDEOMODE_1280x960RGB,`
 - `FC2_VIDEOMODE_1280x960Y8,`
 - `FC2_VIDEOMODE_1280x960Y16,`
 - `FC2_VIDEOMODE_1600x1200YUV422,`
 - `FC2_VIDEOMODE_1600x1200RGB,`
 - `FC2_VIDEOMODE_1600x1200Y8,`
 - `FC2_VIDEOMODE_1600x1200Y16,`
 - `FC2_VIDEOMODE_FORMAT7,`
 - `FC2_NUM_VIDEOMODES,`
 - `FC2_VIDEOMODE_FORCE_32BITS = FULL_32BIT_VALUE }`

DCAM video modes.

- enum `fc2Mode` {

```

FC2_MODE_0 = 0,
FC2_MODE_1,
FC2_MODE_2,
FC2_MODE_3,
FC2_MODE_4,
FC2_MODE_5,
FC2_MODE_6,
FC2_MODE_7,
FC2_MODE_8,
FC2_MODE_9,
FC2_MODE_10,
FC2_MODE_11,
FC2_MODE_12,
FC2_MODE_13,
FC2_MODE_14,
FC2_MODE_15,
FC2_MODE_16,
FC2_MODE_17,
FC2_MODE_18,
FC2_MODE_19,
FC2_MODE_20,
FC2_MODE_21,
FC2_MODE_22,
FC2_MODE_23,
FC2_MODE_24,
FC2_MODE_25,
FC2_MODE_26,
FC2_MODE_27,
FC2_MODE_28,
FC2_MODE_29,
FC2_MODE_30,
FC2_MODE_31,
FC2_NUM_MODES,
FC2_MODE_FORCE_32BITS = FULL_32BIT_VALUE }

```

Camera modes for DCAM formats as well as Format7.

```

• enum fc2PixelFormat {
    FC2_PIXEL_FORMAT_MONO8 = 0x80000000,
    FC2_PIXEL_FORMAT_411YUV8 = 0x40000000,
    FC2_PIXEL_FORMAT_422YUV8 = 0x20000000,
    FC2_PIXEL_FORMAT_444YUV8 = 0x10000000,
    FC2_PIXEL_FORMAT_RGB8 = 0x08000000,
    FC2_PIXEL_FORMAT_MONO16 = 0x04000000,
    FC2_PIXEL_FORMAT_RGB16 = 0x02000000,
    FC2_PIXEL_FORMAT_S_MONO16 = 0x01000000,
    FC2_PIXEL_FORMAT_S_RGB16 = 0x00800000,
    FC2_PIXEL_FORMAT_RAW8 = 0x00400000,
    FC2_PIXEL_FORMAT_RAW16 = 0x00200000,
    FC2_PIXEL_FORMAT_MONO12 = 0x00100000,
    FC2_PIXEL_FORMAT_RAW12 = 0x00080000,
    FC2_PIXEL_FORMAT_BGR = 0x80000008,
    FC2_PIXEL_FORMAT_BGRU = 0x40000008,
    FC2_PIXEL_FORMAT_RGB = FC2_PIXEL_FORMAT_RGB8,
    FC2_PIXEL_FORMAT_RGBU = 0x40000002,
    FC2_PIXEL_FORMAT_BGR16 = 0x02000001,
    FC2_PIXEL_FORMAT_BGRU16 = 0x02000002,
    FC2_PIXEL_FORMAT_422YUV8_JPEG = 0x40000001,
    FC2_NUM_PIXEL_FORMATS = 20,
    FC2_UNSPECIFIED_PIXEL_FORMAT = 0 }

```

Pixel formats available for Format7 modes.

- enum `fc2BusSpeed` {
`FC2_BUSSPEED_S100`,
`FC2_BUSSPEED_S200`,
`FC2_BUSSPEED_S400`,
`FC2_BUSSPEED_S480`,
`FC2_BUSSPEED_S800`,
`FC2_BUSSPEED_S1600`,
`FC2_BUSSPEED_S3200`,
`FC2_BUSSPEED_S5000`,
`FC2_BUSSPEED_10BASE_T`,
`FC2_BUSSPEED_100BASE_T`,
`FC2_BUSSPEED_1000BASE_T`,
`FC2_BUSSPEED_10000BASE_T`,
`FC2_BUSSPEED_S_FASTEST`,
`FC2_BUSSPEED_ANY`,
`FC2_BUSSPEED_SPEED_UNKNOWN` = -1,
`FC2_BUSSPEED_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Bus speeds.

- enum `fc2PCleBusSpeed` {
`FC2_PCIE_BUSSPEED_2_5`,
`FC2_PCIE_BUSSPEED_5_0`,
`FC2_PCIE_BUSSPEED_UNKNOWN` = -1,
`FC2_PCIE_BUSSPEED_FORCE_32BITS` = `FULL_32BIT_VALUE` }
- enum `fc2DriverType` {
`FC2_DRIVER_1394_CAM`,
`FC2_DRIVER_1394_PRO`,
`FC2_DRIVER_1394_JUUU`,
`FC2_DRIVER_1394_VIDEO1394`,
`FC2_DRIVER_1394_RAW1394`,
`FC2_DRIVER_USB_NONE`,
`FC2_DRIVER_USB_CAM`,
`FC2_DRIVER_USB3_PRO`,
`FC2_DRIVER_GIGE_NONE`,
`FC2_DRIVER_GIGE_FILTER`,
`FC2_DRIVER_GIGE_PRO`,
`FC2_DRIVER_GIGE_LWF`,
`FC2_DRIVER_UNKNOWN` = -1,
`FC2_DRIVER_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Types of low level drivers that FlyCapture uses.

- enum `fc2ColorProcessingAlgorithm` {
`FC2_DEFAULT`,
`FC2_NO_COLOR_PROCESSING`,
`FC2_NEAREST_NEIGHBOR_FAST`,
`FC2_EDGE_SENSING`,
`FC2_HQ_LINEAR`,
`FC2_RIGOROUS`,
`FC2_IPP`,
`FC2_DIRECTIONAL`,
`FC2_COLOR_PROCESSING_ALGORITHM_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Color processing algorithms.

- enum `fc2BayerTileFormat` {
`FC2_BT_NONE`,
`FC2_BT_RGGB`,
`FC2_BT_GRGB`,
`FC2_BT_GBRG`,
`FC2_BT_BGGR`,
`FC2_BT_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Bayer tile formats.

- enum `fc2ImageFileFormat` {
`FC2_FROM_FILE_EXT` = -1,
`FC2_PGM`,
`FC2_PPM`,
`FC2_BMP`,
`FC2_JPEG`,
`FC2_JPEG2000`,
`FC2_TIFF`,
`FC2_PNG`,
`FC2_RAW`,
`FC2_IMAGE_FILE_FORMAT_FORCE_32BITS` = `FULL_32BIT_VALUE` }

File formats to be used for saving images to disk.

- enum `fc2GigEPropertyType` {
`FC2_HEARTBEAT`,
`FC2_HEARTBEAT_TIMEOUT`,
`PACKET_SIZE`,
`PACKET_DELAY` }

Possible properties that can be queried from the camera.

- enum `fc2StatisticsChannel` {
`FC2_STATISTICS_GREY`,
`FC2_STATISTICS_RED`,
`FC2_STATISTICS_GREEN`,
`FC2_STATISTICS_BLUE`,
`FC2_STATISTICS_HUE`,
`FC2_STATISTICS_SATURATION`,
`FC2_STATISTICS_LIGHTNESS`,
`FC2_STATISTICS_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Channels that allow statistics to be calculated.

- enum `fc2OSType` {
`FC2_WINDOWS_X86`,
`FC2_WINDOWS_X64`,
`FC2_LINUX_X86`,
`FC2_LINUX_X64`,
`FC2_MAC`,
`FC2_UNKNOWN_OS`,
`FC2_OSTYPE_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Possible operating systems.

- enum `fc2ByteOrder` {
`FC2_BYTE_ORDER_LITTLE_ENDIAN`,
`FC2_BYTE_ORDER_BIG_ENDIAN`,
`FC2_BYTE_ORDER_FORCE_32BITS` = `FULL_32BIT_VALUE` }

Possible byte orders.

- enum `fc2PortType` {
`NOT_CONNECTED` = 1,
`CONNECTED_TO_PARENT`,
`CONNECTED_TO_CHILD` }

Possible states of a port on a node.

- enum `fc2NodeType` {
`COMPUTER`,
`BUS`,
`CAMERA`,
`NODE` }

Type of node.

- enum `fc2TIFFCompressionMethod` {
`FC2_TIFF_NONE` = 1,
`FC2_TIFF_PACKBITS`,
`FC2_TIFF_DEFLATE`,
`FC2_TIFF_ADOBE_DEFLATE`,
`FC2_TIFF_CCITTFAX3`,
`FC2_TIFF_CCITTFAX4`,
`FC2_TIFF_LZW`,
`FC2_TIFF_JPEG` }

7.2.1 Enumeration Type Documentation

7.2.1.1 enum `fc2ByteOrder`

Possible byte orders.

Enumerator

FC2_BYTE_ORDER_LITTLE_ENDIAN
FC2_BYTE_ORDER_BIG_ENDIAN
FC2_BYTE_ORDER_FORCE_32BITS

7.2.1.2 enum `fc2NodeType`

Type of node.

Enumerator

COMPUTER
BUS
CAMERA
NODE

7.2.1.3 enum `fc2OSType`

Possible operating systems.

Enumerator

FC2_WINDOWS_X86 All Windows 32-bit variants.
FC2_WINDOWS_X64 All Windows 64-bit variants.
FC2_LINUX_X86 All Linux 32-bit variants.
FC2_LINUX_X64 All Linux 32-bit variants.
FC2_MAC Mac OSX.
FC2_UNKNOWN_OS Unknown operating system.
FC2_OSTYPE_FORCE_32BITS

7.2.1.4 enum fc2PortType

Possible states of a port on a node.

Enumerator

NOT_CONNECTED
CONNECTED_TO_PARENT
CONNECTED_TO_CHILD

7.2.1.5 enum fc2StatisticsChannel

Channels that allow statistics to be calculated.

Enumerator

FC2_STATISTICS_GREY
FC2_STATISTICS_RED
FC2_STATISTICS_GREEN
FC2_STATISTICS_BLUE
FC2_STATISTICS_HUE
FC2_STATISTICS_SATURATION
FC2_STATISTICS_LIGHTNESS
FC2_STATISTICS_FORCE_32BITS

7.3 FlyCapture2GUI_C.h File Reference

Include dependency graph for FlyCapture2GUI_C.h:

Functions

- [FLYCAPTURE2_C_API](#) [fc2Error](#) [fc2CreateGUIContext](#) ([fc2GuiContext](#) *pContext)
Create a GUI context.
- [FLYCAPTURE2_C_API](#) [fc2Error](#) [fc2DestroyGUIContext](#) ([fc2GuiContext](#) context)
Destroy a GUI context.
- [FLYCAPTURE2_C_API](#) void [fc2GUIConnect](#) ([fc2GuiContext](#) context, [fc2Context](#) cameraContext)
Connect GUI context to a camera context.
- [FLYCAPTURE2_C_API](#) void [fc2GUIDisconnect](#) ([fc2GuiContext](#) context)
Disconnect GUI context from camera.
- [FLYCAPTURE2_C_API](#) void [fc2Disonnnect](#) ([fc2GuiContext](#) context) [__attribute__\(\(deprecated\)\)](#)
Disconnect GUI context from camera.
- [FLYCAPTURE2_C_API](#) void [fc2Show](#) ([fc2GuiContext](#) context)
Show the GUI.
- [FLYCAPTURE2_C_API](#) void [fc2Hide](#) ([fc2GuiContext](#) context)
Hide the GUI.
- [FLYCAPTURE2_C_API](#) [BOOL](#) [fc2IsVisible](#) ([fc2GuiContext](#) context)
Check if the GUI is visible.
- [FLYCAPTURE2_C_API](#) void [fc2ShowModal](#) ([fc2GuiContext](#) context, [BOOL](#) *pOkSelected, [fc2PGRGuid](#) *guidArray, unsigned int *size)
Show the camera selection dialog.

7.3.1 Function Documentation

7.3.1.1 FLYCAPTURE2_C_API `fc2Error fc2CreateGUIContext (fc2GuiContext * pContext)`

Create a GUI context.

Parameters

<i>pContext</i>	Pointer to context to be created.
-----------------	-----------------------------------

Returns

An Error indicating the success or failure of the function.

7.3.1.2 FLYCAPTURE2_C_API `fc2Error fc2DestroyGUIContext (fc2GuiContext context)`

Destroy a GUI context.

Parameters

<i>context</i>	Context to be destroyed.
----------------	--------------------------

Returns

An Error indicating the success or failure of the function.

7.3.1.3 FLYCAPTURE2_C_API `void fc2Disconnect (fc2GuiContext context)`

Disconnect GUI context from camera.

Parameters

<i>context</i>	GUI context to disconnect.
----------------	----------------------------

Returns

An Error indicating the success or failure of the function.

Deprecated This method is deprecated and will be removed in a future FlyCapture2 release. Please use `fc2GUI↔IDisconnect` instead.

7.3.1.4 FLYCAPTURE2_C_API `void fc2GUIConnect (fc2GuiContext context, fc2Context cameraContext)`

Connect GUI context to a camera context.

Parameters

<i>context</i>	GUI context to connect.
<i>cameraContext</i>	Camera context to connect.

Returns

An Error indicating the success or failure of the function.

7.3.1.5 FLYCAPTURE2_C_API void fc2GUIDisconnect (fc2GuiContext *context*)

Disconnect GUI context from camera.

Parameters

<i>context</i>	GUI context to disconnect.
----------------	----------------------------

Returns

An Error indicating the success or failure of the function.

7.3.1.6 FLYCAPTURE2_C_API void fc2Hide (fc2GuiContext *context*)

Hide the GUI.

Parameters

<i>context</i>	Pointer to context to hide.
----------------	-----------------------------

Returns

An Error indicating the success or failure of the function.

7.3.1.7 FLYCAPTURE2_C_API BOOL fc2IsVisible (fc2GuiContext *context*)

Check if the GUI is visible.

Parameters

<i>context</i>	Pointer to context to show.
----------------	-----------------------------

Returns

Whether the GUI is visible.

7.3.1.8 FLYCAPTURE2_C_API void fc2Show (fc2GuiContext *context*)

Show the GUI.

Parameters

<i>context</i>	Pointer to context to show.
----------------	-----------------------------

Returns

An Error indicating the success or failure of the function.

7.3.1.9 FLYCAPTURE2_C_API void fc2ShowModal (fc2GuiContext *context*, BOOL * *pOkSelected*, fc2PGRGuid * *guidArray*, unsigned int * *size*)

Show the camera selection dialog.

Parameters

<i>context</i>	Pointer to context to show.
<i>pOkSelected</i>	Whether Ok (true) or Cancel (false) was clicked.
<i>guidArray</i>	Array of PGRGuids containing the selected cameras.
<i>size</i>	Size of PGRGuid array.

7.4 FlyCapture2Internal_C.h File Reference

Include dependency graph for FlyCapture2Internal_C.h:

Data Structures

- struct [fc2InternalContext](#)
- struct [fc2InternalGuiContext](#)
- struct [fc2InternalImageCallback](#)

Functions

- bool [IsContextValid](#) (fc2Context *context*)
- bool [IsGuiContextValid](#) (fc2GuiContext *context*)
- void [SyncCpplImageToStruct](#) (fc2Image **pImage*)

7.4.1 Function Documentation

7.4.1.1 `bool IsContextValid (fc2Context context)` `[inline]`

7.4.1.2 `bool IsGuiContextValid (fc2GuiContext context)` `[inline]`

7.4.1.3 `void SyncCpplImageToStruct (fc2Image * plmage)` `[inline]`

7.5 FlyCapture2Platform_C.h File Reference

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

Macros

- `#define FLYCAPTURE2_C_API`
- `#define FLYCAPTURE2_C_CALL_CONVEN`

7.5.1 Macro Definition Documentation

7.5.1.1 `#define FLYCAPTURE2_C_API`

7.5.1.2 `#define FLYCAPTURE2_C_CALL_CONVEN`

7.6 FlyCapture2Private_C.h File Reference

Include dependency graph for FlyCapture2Private_C.h:

Functions

- `FLYCAPTURE2_C_API void * GetInternal (unsigned int index)`

7.6.1 Function Documentation

7.6.1.1 `FLYCAPTURE2_C_API void* GetInternal (unsigned int index)`

7.7 MultiSyncLibrary_C.h File Reference

Include dependency graph for MultiSyncLibrary_C.h:

Functions

- [MULTISYNCLIBRARY_C_API syncError syncCreateContext \(syncContext *pContext\)](#)
Create a Sync context for MultiSync Library.
- [MULTISYNCLIBRARY_C_API syncError syncDestroyContext \(syncContext context\)](#)
Destory the sync context.
- [MULTISYNCLIBRARY_C_API syncError syncStart \(syncContext context\)](#)
Start the sync progress.
- [MULTISYNCLIBRARY_C_API syncError syncStop \(syncContext context\)](#)
Stop the sync progress.
- [MULTISYNCLIBRARY_C_API syncError syncRescanMasterTimingBus \(syncContext context\)](#)
Scan newly connected or removed timing bus (for corss-PC syncing only)
- [MULTISYNCLIBRARY_C_API syncMessage syncGetStatus \(syncContext context\)](#)
Start the sync progress.
- [MULTISYNCLIBRARY_C_API double syncGetTimeSinceSynced \(syncContext context\)](#)
Time since sync started.
- [MULTISYNCLIBRARY_C_API BOOL syncIsTimingBusConnected \(syncContext context\)](#)
Whether syncing across PCs.
- [MULTISYNCLIBRARY_C_API BOOL syncEnableCrossPCSynchronization \(syncContext context\)](#)
Enable across pc synchronization support.
- [MULTISYNCLIBRARY_C_API BOOL syncDisableCrossPCSynchronization \(syncContext context\)](#)
Disable across pc synchronization support.
- [MULTISYNCLIBRARY_C_API BOOL syncQueryCrossPCSynchronizationSetting \(syncContext context\)](#)
Query cross pc synchronizaion support status.

7.7.1 Function Documentation

7.7.1.1 MULTISYNCLIBRARY_C_API syncError syncCreateContext (syncContext * pContext)

Create a Sync context for MultiSync Library.

This call must be made before any other calls that use a context will succeed.

Parameters

<i>pContext</i>	A pointer to the syncContext to be created.
-----------------	---

Returns

A syncError indicating the success or failure of the function.

7.7.1.2 MULTISYNCLIBRARY_C_API syncError syncDestroyContext (syncContext context)

Destory the sync context.

This must be called when the user is finished with the context in order to prevent memory leaks.

Parameters

<i>context</i>	The syncContext to be destroyed.
----------------	----------------------------------

Returns

A syncError indicating the success or failure of the function.

7.7.1.3 MULTISYNCLIBRARY_C_API BOOL syncDisableCrossPCSynchronization (syncContext *context*)

Disable across pc synchronization support.

Parameters

<i>context</i>	The syncContext to be used.
----------------	-----------------------------

Returns

True if operation was successful

7.7.1.4 MULTISYNCLIBRARY_C_API BOOL syncEnableCrossPCSynchronization (syncContext *context*)

Enable across pc synchronization support.

Parameters

<i>context</i>	The syncContext to be used.
----------------	-----------------------------

Returns

True if operation was successful

7.7.1.5 MULTISYNCLIBRARY_C_API syncMessage syncGetStatus (syncContext *context*)

Start the sync progress.

Parameters

<i>context</i>	The syncContext to be used.
----------------	-----------------------------

Returns

A syncMessage indicating the sync status.

7.7.1.6 MULTISYNCLIBRARY_C_API double syncGetTimeSinceSynced (syncContext context)

Time since sync started.

Parameters

<i>context</i>	The syncContext to be used.
----------------	-----------------------------

Returns

Time since sync started.

7.7.1.7 MULTISYNCLIBRARY_C_API BOOL syncIsTimingBusConnected (syncContext context)

Whether syncing across PCs.

Parameters

<i>context</i>	The syncContext to be used.
----------------	-----------------------------

Returns

True if its syncing across PC

7.7.1.8 MULTISYNCLIBRARY_C_API BOOL syncQueryCrossPCSynchronizationSetting (syncContext context)

Query cross pc synchronizaion support status.

Parameters

<i>context</i>	The syncContext to be used.
----------------	-----------------------------

Returns

True if cross pc synchronization was supported

7.7.1.9 MULTISYNCLIBRARY_C_API syncError syncRescanMasterTimingBus (syncContext context)

Scan newly connected or removed timing bus (for corss-PC syncing only)

Parameters

<i>context</i>	The syncContext to be used.
----------------	-----------------------------

Returns

A syncError indicating the success or failure of the function.

7.7.1.10 MULTISYNCLIBRARY_C_API syncError syncStart (syncContext context)

Start the sync progress.

Parameters

<i>context</i>	The syncContext to be used.
----------------	-----------------------------

Returns

A syncError indicating the success or failure of the function.

7.7.1.11 MULTISYNCLIBRARY_C_API syncError syncStop (syncContext context)

Stop the sync progress.

Parameters

<i>context</i>	The syncContext to be used.
----------------	-----------------------------

Returns

A syncError indicating the success or failure of the function.

7.8 MultiSyncLibraryDefs_C.h File Reference

Include dependency graph for MultiSyncLibraryDefs_C.h: This graph shows which files directly or indirectly include this file:

Macros

- #define [FALSE](#) 0
- #define [TRUE](#) 1
- #define [FULL_32BIT_VALUE](#) 0x7FFFFFFF
- #define [MAX_STRING_LENGTH](#) 512

Typedefs

- typedef int [BOOL](#)
- typedef void * [syncContext](#)
A context to the MultiSync C library.

Enumerations

- enum `syncError` {
`SYNC_ERROR_OK` = 0,
`SYNC_ERROR_FAILED`,
`SYNC_ERROR_ALREADY_STARTED`,
`SYNC_ERROR_ALREADY_STOPPED`,
`SYNC_ERROR_CONTEXT_NOT_INITIALIZED`,
`SYNC_ERROR_UNKNOWN_ERROR` }
- enum `syncMessage` {
`SYNC_MESSAGE_OK` = 0,
`SYNC_MESSAGE_FAILED`,
`SYNC_MESSAGE_STARTED`,
`SYNC_MESSAGE_STOPPED`,
`SYNC_MESSAGE_SYNCING`,
`SYNC_MESSAGE_NOMASTER`,
`SYNC_MESSAGE_THREAD_ERROR`,
`SYNC_MESSAGE_DEVICE_ERROR`,
`SYNC_MESSAGE_NOT_ENOUGH_DEVICES`,
`SYNC_MESSAGE_BUS_RESET`,
`SYNC_MESSAGE_NOT_INITIALIZED`,
`SYNC_MESSAGE_UNKNOWN_ERROR` }

7.8.1 Macro Definition Documentation

7.8.1.1 `#define FALSE 0`

7.8.1.2 `#define FULL_32BIT_VALUE 0x7FFFFFFF`

7.8.1.3 `#define MAX_STRING_LENGTH 512`

7.8.1.4 `#define TRUE 1`

7.8.2 Typedef Documentation

7.8.2.1 `typedef int BOOL`

7.8.2.2 `typedef void* syncContext`

A context to the MultiSync C library.

It must be created before performing any calls to the library.

7.8.3 Enumeration Type Documentation

7.8.3.1 `enum syncError`

Enumerator

`SYNC_ERROR_OK`
`SYNC_ERROR_FAILED`
`SYNC_ERROR_ALREADY_STARTED`
`SYNC_ERROR_ALREADY_STOPPED`
`SYNC_ERROR_CONTEXT_NOT_INITIALIZED`
`SYNC_ERROR_UNKNOWN_ERROR`

7.8.3.2 enum syncMessage

Enumerator

SYNC_MESSAGE_OK
SYNC_MESSAGE_FAILED
SYNC_MESSAGE_STARTED
SYNC_MESSAGE_STOPPED
SYNC_MESSAGE_SYNCING
SYNC_MESSAGE_NOMASTER
SYNC_MESSAGE_THREAD_ERROR
SYNC_MESSAGE_DEVICE_ERROR
SYNC_MESSAGE_NOT_ENOUGH_DEVICES
SYNC_MESSAGE_BUS_RESET
SYNC_MESSAGE_NOT_INITIALIZED
SYNC_MESSAGE_UNKNOWN_ERROR

7.9 MultiSyncLibraryPlatform_C.h File Reference

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

Macros

- #define [MULTISYNCLIBRARY_C_API](#)
- #define [MULTISYNCLIBRARY_C_CALL_CONVEN](#)

7.9.1 Macro Definition Documentation

7.9.1.1 #define MULTISYNCLIBRARY_C_API

7.9.1.2 #define MULTISYNCLIBRARY_C_CALL_CONVEN

Index

- AVI Recording Operation, [84](#)
 - [fc2AVIAppend, 84](#)
 - [fc2AVIClose, 84](#)
 - [fc2AVIOpen, 85](#)
 - [fc2CreateAVI, 85](#)
 - [fc2DestroyAVI, 85](#)
 - [fc2H264Open, 86](#)
 - [fc2MJPEGOpen, 86](#)
- absControl
 - [fc2TriggerDelay, 170](#)
- absMax
 - [fc2TriggerDelayInfo, 172](#)
- absMin
 - [fc2TriggerDelayInfo, 172](#)
- absValSupported
 - [fc2TriggerDelayInfo, 172](#)
- absValue
 - [fc2TriggerDelay, 170](#)
- applicationIPAddress
 - [fc2CameraInfo, 124](#)
- applicationPort
 - [fc2CameraInfo, 124](#)
- asyncBusSpeed
 - [fc2Config, 130](#)
- autoManualMode
 - [fc2TriggerDelay, 170](#)
- autoSupported
 - [fc2TriggerDelayInfo, 172](#)
- available
 - [fc2EmbeddedImageInfoProperty, 136](#)
- BOOL
 - [MultiSyncLibraryDefs_C.h, 198](#)
 - [TypeDefs, 98](#)
- BUS
 - [FlyCapture2Defs_C.h, 188](#)
- bandwidthAllocation
 - [fc2Config, 130](#)
- bayerFormat
 - [fc2Image, 151](#)
- bayerTileFormat
 - [fc2CameraInfo, 124](#)
- binaryFile
 - [fc2PGMOption, 160](#)
 - [fc2PPMOption, 162](#)
- bitrate
 - [fc2H264Option, 150](#)
- brightness
 - [fc2EmbeddedImageInfo, 135](#)
- build
 - [fc2Version, 177](#)
- Bus Manager Operation, [9](#)
 - [fc2DiscoverGigECameras, 10](#)
 - [fc2FireBusReset, 11](#)
 - [fc2ForceAllIPAddressesAutomatically, 11](#)
 - [fc2ForceIPAddressAutomatically, 11](#)
 - [fc2ForceIPAddressToCamera, 11](#)
 - [fc2GetCameraFromIPAddress, 12](#)
 - [fc2GetCameraFromIndex, 12](#)
 - [fc2GetCameraFromSerialNumber, 13](#)
 - [fc2GetCameraSerialNumberFromIndex, 13](#)
 - [fc2GetDeviceFromIndex, 13](#)
 - [fc2GetInterfaceTypeFromGuid, 14](#)
 - [fc2GetNumOfCameras, 14](#)
 - [fc2GetNumOfDevices, 14](#)
 - [fc2GetTopology, 15](#)
 - [fc2GetUsbLinkInfo, 15](#)
 - [fc2GetUsbPortStatus, 15](#)
 - [fc2IsCameraControlable, 16](#)
 - [fc2ReadPhyRegister, 16](#)
 - [fc2RegisterCallback, 16](#)
 - [fc2RescanBus, 17](#)
 - [fc2UnregisterCallback, 17](#)
 - [fc2WritePhyRegister, 17](#)
- busNumber
 - [fc2CameraInfo, 124](#)
- byteOrder
 - [fc2SystemInfo, 166](#)
- CAMERA
 - [FlyCapture2Defs_C.h, 188](#)
- COMPUTER
 - [FlyCapture2Defs_C.h, 188](#)
- CONNECTED_TO_CHILD
 - [FlyCapture2Defs_C.h, 189](#)
- CONNECTED_TO_PARENT
 - [FlyCapture2Defs_C.h, 189](#)
- cameraCurrents
 - [fc2CameraStats, 128](#)
- cameraPowerUp
 - [fc2CameraStats, 128](#)
- cameraVoltages
 - [fc2CameraStats, 128](#)
- ccpStatus
 - [fc2CameraInfo, 124](#)
- chipIdHi
 - [fc2ConfigROM, 133](#)
- chipIdLo
 - [fc2ConfigROM, 133](#)
- cols

- fc2Image, 151
- compression
 - fc2TIFFOption, 167
- compressionLevel
 - fc2PNGOption, 161
- configROM
 - fc2CameraInfo, 124
- Connection and Image Retrieval, 19
 - fc2Connect, 19
 - fc2Disconnect, 20
 - fc2GetConfiguration, 20
 - fc2RetrieveBuffer, 20
 - fc2SetCallback, 21
 - fc2SetConfiguration, 21
 - fc2SetUserBuffers, 21
 - fc2StartCapture, 22
 - fc2StartCaptureCallback, 22
 - fc2StartSyncCapture, 23
 - fc2StartSyncCaptureCallback, 23
 - fc2StopCapture, 24
 - fc2WaitForBufferEvent, 24
- cpuDescription
 - fc2SystemInfo, 166
- cycleCount
 - fc2TimeStamp, 168
- cycleOffset
 - fc2TimeStamp, 168
- cycleSeconds
 - fc2TimeStamp, 168
- DCAM Formats, 51
 - fc2GetVideoModeAndFrameRate, 51
 - fc2GetVideoModeAndFrameRateInfo, 51
 - fc2SetVideoModeAndFrameRate, 52
- dataSize
 - fc2Image, 151
- defaultGateway
 - fc2CameraInfo, 125
- delay
 - fc2StrobeControl, 163
- destinationIpAddress
 - fc2GigEStreamChannel, 149
- doNotFragment
 - fc2GigEStreamChannel, 149
- driverList
 - fc2SystemInfo, 166
- driverName
 - fc2CameraInfo, 125
- driverType
 - fc2CameraInfo, 125
- duration
 - fc2StrobeControl, 163
- embeddedBrightness
 - fc2ImageMetadata, 152
- embeddedExposure
 - fc2ImageMetadata, 152
- embeddedFrameCounter
 - fc2ImageMetadata, 152
- embeddedGPIOPinState
 - fc2ImageMetadata, 152
- embeddedGain
 - fc2ImageMetadata, 152
- embeddedROIPosition
 - fc2ImageMetadata, 153
- embeddedShutter
 - fc2ImageMetadata, 153
- embeddedStrobePattern
 - fc2ImageMetadata, 153
- embeddedTimeStamp
 - fc2ImageMetadata, 153
- embeddedWhiteBalance
 - fc2ImageMetadata, 153
- enablePacketResend
 - fc2GigEConfig, 143
- enabled
 - fc2LUTData, 157
- Enumerations, 100
 - FC2_ARRIVAL, 106
 - FC2_AUTO_EXPOSURE, 112
 - FC2_BANDWIDTH_ALLOCATION_FORCE_32←BITS, 105
 - FC2_BANDWIDTH_ALLOCATION_OFF, 105
 - FC2_BANDWIDTH_ALLOCATION_ON, 105
 - FC2_BANDWIDTH_ALLOCATION_UNSPECIFI←ED, 105
 - FC2_BANDWIDTH_ALLOCATION_UNSUPPO←RTED, 105
 - FC2_BMP, 109
 - FC2_BRIGHTNESS, 112
 - FC2_BT_BGGR, 105
 - FC2_BT_FORCE_32BITS, 105
 - FC2_BT_GBRG, 105
 - FC2_BT_GRBG, 105
 - FC2_BT_NONE, 105
 - FC2_BT_RGB, 105
 - FC2_BUFFER_FRAMES, 109
 - FC2_BUS_RESET, 106
 - FC2_BUSSPEED_10000BASE_T, 106
 - FC2_BUSSPEED_1000BASE_T, 106
 - FC2_BUSSPEED_100BASE_T, 106
 - FC2_BUSSPEED_10BASE_T, 106
 - FC2_BUSSPEED_ANY, 106
 - FC2_BUSSPEED_FORCE_32BITS, 106
 - FC2_BUSSPEED_S100, 106
 - FC2_BUSSPEED_S1600, 106
 - FC2_BUSSPEED_S200, 106
 - FC2_BUSSPEED_S3200, 106
 - FC2_BUSSPEED_S400, 106
 - FC2_BUSSPEED_S480, 106
 - FC2_BUSSPEED_S5000, 106
 - FC2_BUSSPEED_S800, 106
 - FC2_BUSSPEED_S_FASTEST, 106
 - FC2_BUSSPEED_SPEED_UNKNOWN, 106
 - FC2_CALLBACK_TYPE_FORCE_32BITS, 106
 - FC2_COLOR_PROCESSING_ALGORITHM_F←ORCE_32BITS, 106

- FC2_DEFAULT, 106
- FC2_DIRECTIONAL, 106
- FC2_DRIVER_1394_CAM, 107
- FC2_DRIVER_1394_JUJU, 107
- FC2_DRIVER_1394_PRO, 107
- FC2_DRIVER_1394_RAW1394, 107
- FC2_DRIVER_1394_VIDEO1394, 107
- FC2_DRIVER_FORCE_32BITS, 107
- FC2_DRIVER_GIGE_FILTER, 107
- FC2_DRIVER_GIGE_LWF, 107
- FC2_DRIVER_GIGE_NONE, 107
- FC2_DRIVER_GIGE_PRO, 107
- FC2_DRIVER_UNKNOWN, 107
- FC2_DRIVER_USB3_PRO, 107
- FC2_DRIVER_USB_CAM, 107
- FC2_DRIVER_USB_NONE, 107
- FC2_DROP_FRAMES, 109
- FC2_EDGE_SENSING, 106
- FC2_ERROR_BUFFER_TOO_SMALL, 108
- FC2_ERROR_BUS_MASTER_FAILED, 108
- FC2_ERROR_FAILED_BUS_MASTER_CONNECTION, 107
- FC2_ERROR_FAILED_GUID, 107
- FC2_ERROR_FAILED, 107
- FC2_ERROR_FORCE_32BITS, 108
- FC2_ERROR_IIDC_FAILED, 108
- FC2_ERROR_IMAGE_CONSISTENCY_ERROR, 108
- FC2_ERROR_IMAGE_CONVERSION_FAILED, 108
- FC2_ERROR_IMAGE_LIBRARY_FAILURE, 108
- FC2_ERROR_INCOMPATIBLE_DRIVER, 108
- FC2_ERROR_INIT_FAILED, 107
- FC2_ERROR_INVALID_BUS_MANAGER, 107
- FC2_ERROR_INVALID_GENERATION, 108
- FC2_ERROR_INVALID_MODE, 107
- FC2_ERROR_INVALID_PACKET_SIZE, 107
- FC2_ERROR_INVALID_PARAMETER, 107
- FC2_ERROR_INVALID_SETTINGS, 107
- FC2_ERROR_ISOCH_ALREADY_STARTED, 108
- FC2_ERROR_ISOCH_BANDWIDTH_EXCEEDED, 108
- FC2_ERROR_ISOCH_FAILED, 108
- FC2_ERROR_ISOCH_NOT_STARTED, 108
- FC2_ERROR_ISOCH_RETRIEVE_BUFFER_FAILED, 108
- FC2_ERROR_ISOCH_START_FAILED, 108
- FC2_ERROR_ISOCH_STOP_FAILED, 108
- FC2_ERROR_ISOCH_SYNC_FAILED, 108
- FC2_ERROR_LOW_LEVEL_FAILURE, 107
- FC2_ERROR_LUT_FAILED, 108
- FC2_ERROR_MEMORY_ALLOCATION_FAILED, 107
- FC2_ERROR_NOT_CONNECTED, 107
- FC2_ERROR_NOT_FOUND, 107
- FC2_ERROR_NOT_IMPLEMENTED, 107
- FC2_ERROR_NOT_IN_FORMAT7, 107
- FC2_ERROR_NOT_INITIALIZED, 107
- FC2_ERROR_NOT_SUPPORTED, 107
- FC2_ERROR_OK, 107
- FC2_ERROR_PROPERTY_FAILED, 108
- FC2_ERROR_PROPERTY_NOT_PRESENT, 108
- FC2_ERROR_READ_REGISTER_FAILED, 108
- FC2_ERROR_REGISTER_FAILED, 108
- FC2_ERROR_STROBE_FAILED, 108
- FC2_ERROR_TIMEOUT, 107
- FC2_ERROR_TRIGGER_FAILED, 108
- FC2_ERROR_UNDEFINED, 107
- FC2_ERROR_WRITE_REGISTER_FAILED, 108
- FC2_FOCUS, 112
- FC2_FRAME_RATE, 112
- FC2_FRAMERATE_120, 108
- FC2_FRAMERATE_15, 108
- FC2_FRAMERATE_1_875, 108
- FC2_FRAMERATE_240, 108
- FC2_FRAMERATE_30, 108
- FC2_FRAMERATE_3_75, 108
- FC2_FRAMERATE_60, 108
- FC2_FRAMERATE_7_5, 108
- FC2_FRAMERATE_FORCE_32BITS, 108
- FC2_FRAMERATE_FORMAT7, 108
- FC2_FROM_FILE_EXT, 109
- FC2_GAIN, 112
- FC2_GAMMA, 112
- FC2_GRAB_MODE_FORCE_32BITS, 109
- FC2_GRAB_TIMEOUT_FORCE_32BITS, 109
- FC2_HQ_LINEAR, 106
- FC2_HUE, 112
- FC2_IMAGE_FILE_FORMAT_FORCE_32BITS, 109
- FC2_INTERFACE_GIGE, 110
- FC2_INTERFACE_IEEE1394, 110
- FC2_INTERFACE_TYPE_FORCE_32BITS, 110
- FC2_INTERFACE_UNKNOWN, 110
- FC2_INTERFACE_USB_2, 110
- FC2_INTERFACE_USB_3, 110
- FC2_IPP, 106
- FC2_IRIS, 112
- FC2_JPEG2000, 109
- FC2_JPEG, 109
- FC2_MODE_0, 110
- FC2_MODE_1, 110
- FC2_MODE_10, 110
- FC2_MODE_11, 110
- FC2_MODE_12, 110
- FC2_MODE_13, 110
- FC2_MODE_14, 110
- FC2_MODE_15, 110
- FC2_MODE_16, 110
- FC2_MODE_17, 110
- FC2_MODE_18, 110
- FC2_MODE_19, 110
- FC2_MODE_2, 110
- FC2_MODE_20, 110
- FC2_MODE_21, 110
- FC2_MODE_22, 110

FC2_MODE_23, 110
 FC2_MODE_24, 110
 FC2_MODE_25, 110
 FC2_MODE_26, 110
 FC2_MODE_27, 110
 FC2_MODE_28, 110
 FC2_MODE_29, 110
 FC2_MODE_3, 110
 FC2_MODE_30, 110
 FC2_MODE_31, 110
 FC2_MODE_4, 110
 FC2_MODE_5, 110
 FC2_MODE_6, 110
 FC2_MODE_7, 110
 FC2_MODE_8, 110
 FC2_MODE_9, 110
 FC2_MODE_FORCE_32BITS, 110
 FC2_NEAREST_NEIGHBOR_FAST, 106
 FC2_NO_COLOR_PROCESSING, 106
 FC2_NUM_FRAMERATES, 108
 FC2_NUM_MODES, 110
 FC2_NUM_PIXEL_FORMATS, 111
 FC2_NUM_VIDEOMODES, 113
 FC2_PAN, 112
 FC2_PCIE_BUSSPEED_2_5, 111
 FC2_PCIE_BUSSPEED_5_0, 111
 FC2_PCIE_BUSSPEED_FORCE_32BITS, 111
 FC2_PCIE_BUSSPEED_UNKNOWN, 111
 FC2_PGM, 109
 FC2_PIXEL_FORMAT_411YUV8, 111
 FC2_PIXEL_FORMAT_422YUV8, 111
 FC2_PIXEL_FORMAT_422YUV8_JPEG, 111
 FC2_PIXEL_FORMAT_444YUV8, 111
 FC2_PIXEL_FORMAT_BGR16, 111
 FC2_PIXEL_FORMAT_BGRU16, 111
 FC2_PIXEL_FORMAT_BGRU, 111
 FC2_PIXEL_FORMAT_BGR, 111
 FC2_PIXEL_FORMAT_MONO12, 111
 FC2_PIXEL_FORMAT_MONO16, 111
 FC2_PIXEL_FORMAT_MONO8, 111
 FC2_PIXEL_FORMAT_RAW12, 111
 FC2_PIXEL_FORMAT_RAW16, 111
 FC2_PIXEL_FORMAT_RAW8, 111
 FC2_PIXEL_FORMAT_RGB16, 111
 FC2_PIXEL_FORMAT_RGB8, 111
 FC2_PIXEL_FORMAT_RGBU, 111
 FC2_PIXEL_FORMAT_RGB, 111
 FC2_PIXEL_FORMAT_S_MONO16, 111
 FC2_PIXEL_FORMAT_S_RGB16, 111
 FC2_PNG, 109
 FC2_PPM, 109
 FC2_PROPERTY_TYPE_FORCE_32BITS, 112
 FC2_RAW, 109
 FC2_REMOVAL, 106
 FC2_RIGOROUS, 106
 FC2_SATURATION, 112
 FC2_SHARPNESS, 112
 FC2_SHUTTER, 112
 FC2_TEMPERATURE, 112
 FC2_TIFF, 109
 FC2_TILT, 112
 FC2_TIMEOUT_INFINITE, 109
 FC2_TIMEOUT_NONE, 109
 FC2_TIMEOUT_UNSPECIFIED, 109
 FC2_TRIGGER_DELAY, 112
 FC2_TRIGGER_MODE, 112
 FC2_UNSPECIFIED_GRAB_MODE, 109
 FC2_UNSPECIFIED_PIXEL_FORMAT, 111
 FC2_UNSPECIFIED_PROPERTY_TYPE, 112
 FC2_VIDEOMODE_1024x768RGB, 112
 FC2_VIDEOMODE_1024x768Y16, 113
 FC2_VIDEOMODE_1024x768Y8, 113
 FC2_VIDEOMODE_1024x768YUV422, 112
 FC2_VIDEOMODE_1280x960RGB, 113
 FC2_VIDEOMODE_1280x960Y16, 113
 FC2_VIDEOMODE_1280x960Y8, 113
 FC2_VIDEOMODE_1280x960YUV422, 113
 FC2_VIDEOMODE_1600x1200RGB, 113
 FC2_VIDEOMODE_1600x1200Y16, 113
 FC2_VIDEOMODE_1600x1200Y8, 113
 FC2_VIDEOMODE_1600x1200YUV422, 113
 FC2_VIDEOMODE_160x120YUV444, 112
 FC2_VIDEOMODE_320x240YUV422, 112
 FC2_VIDEOMODE_640x480RGB, 112
 FC2_VIDEOMODE_640x480Y16, 112
 FC2_VIDEOMODE_640x480Y8, 112
 FC2_VIDEOMODE_640x480YUV411, 112
 FC2_VIDEOMODE_640x480YUV422, 112
 FC2_VIDEOMODE_800x600RGB, 112
 FC2_VIDEOMODE_800x600Y16, 112
 FC2_VIDEOMODE_800x600Y8, 112
 FC2_VIDEOMODE_800x600YUV422, 112
 FC2_VIDEOMODE_FORCE_32BITS, 113
 FC2_VIDEOMODE_FORMAT7, 113
 FC2_WHITE_BALANCE, 112
 FC2_ZOOM, 112
 fc2BandwidthAllocation, 105
 fc2BayerTileFormat, 105
 fc2BusCallbackType, 105
 fc2BusSpeed, 106
 fc2ColorProcessingAlgorithm, 106
 fc2DriverType, 106
 fc2Error, 107
 fc2FrameRate, 108
 fc2GrabMode, 108
 fc2GrabTimeout, 109
 fc2ImageFileFormat, 109
 fc2InterfaceType, 109
 fc2Mode, 110
 fc2PCleBusSpeed, 110
 fc2PixelFormat, 111
 fc2PropertyType, 111
 fc2VideoMode, 112
 EventCallbackFcn
 fc2EventOptions, 138
 EventData

- fc2EventCallbackData, [136](#)
- EventDataSize
 - fc2EventCallbackData, [136](#)
- EventID
 - fc2EventCallbackData, [136](#)
- EventName
 - fc2EventCallbackData, [137](#)
 - fc2EventOptions, [138](#)
- EventTimestamp
 - fc2EventCallbackData, [137](#)
- EventUserData
 - fc2EventCallbackData, [137](#)
 - fc2EventOptions, [138](#)
- EventUserDataSize
 - fc2EventCallbackData, [137](#)
 - fc2EventOptions, [138](#)
- exposure
 - fc2EmbeddedImageInfo, [135](#)
- FALSE
 - MultiSyncLibraryDefs_C.h, [198](#)
 - TypeDefs, [98](#)
- FC2_ARRIVAL
 - Enumerations, [106](#)
- FC2_AUTO_EXPOSURE
 - Enumerations, [112](#)
- FC2_BANDWIDTH_ALLOCATION_FORCE_32BITS
 - Enumerations, [105](#)
- FC2_BANDWIDTH_ALLOCATION_OFF
 - Enumerations, [105](#)
- FC2_BANDWIDTH_ALLOCATION_ON
 - Enumerations, [105](#)
- FC2_BANDWIDTH_ALLOCATION_UNSPECIFIED
 - Enumerations, [105](#)
- FC2_BANDWIDTH_ALLOCATION_UNSUPPORTED
 - Enumerations, [105](#)
- FC2_BMP
 - Enumerations, [109](#)
- FC2_BRIGHTNESS
 - Enumerations, [112](#)
- FC2_BT_BGGR
 - Enumerations, [105](#)
- FC2_BT_FORCE_32BITS
 - Enumerations, [105](#)
- FC2_BT_GBRG
 - Enumerations, [105](#)
- FC2_BT_GRBG
 - Enumerations, [105](#)
- FC2_BT_NONE
 - Enumerations, [105](#)
- FC2_BT_RGGG
 - Enumerations, [105](#)
- FC2_BUFFER_FRAMES
 - Enumerations, [109](#)
- FC2_BUS_RESET
 - Enumerations, [106](#)
- FC2_BUSSPEED_10000BASE_T
 - Enumerations, [106](#)
- FC2_BUSSPEED_1000BASE_T
 - Enumerations, [106](#)
- Enumerations, [106](#)
- FC2_BUSSPEED_100BASE_T
 - Enumerations, [106](#)
- FC2_BUSSPEED_10BASE_T
 - Enumerations, [106](#)
- FC2_BUSSPEED_ANY
 - Enumerations, [106](#)
- FC2_BUSSPEED_FORCE_32BITS
 - Enumerations, [106](#)
- FC2_BUSSPEED_S100
 - Enumerations, [106](#)
- FC2_BUSSPEED_S1600
 - Enumerations, [106](#)
- FC2_BUSSPEED_S200
 - Enumerations, [106](#)
- FC2_BUSSPEED_S3200
 - Enumerations, [106](#)
- FC2_BUSSPEED_S400
 - Enumerations, [106](#)
- FC2_BUSSPEED_S480
 - Enumerations, [106](#)
- FC2_BUSSPEED_S5000
 - Enumerations, [106](#)
- FC2_BUSSPEED_S800
 - Enumerations, [106](#)
- FC2_BUSSPEED_S_FASTEST
 - Enumerations, [106](#)
- FC2_BUSSPEED_SPEED_UNKNOWN
 - Enumerations, [106](#)
- FC2_BYTE_ORDER_BIG_ENDIAN
 - FlyCapture2Defs_C.h, [188](#)
- FC2_BYTE_ORDER_FORCE_32BITS
 - FlyCapture2Defs_C.h, [188](#)
- FC2_BYTE_ORDER_LITTLE_ENDIAN
 - FlyCapture2Defs_C.h, [188](#)
- FC2_CALLBACK_TYPE_FORCE_32BITS
 - Enumerations, [106](#)
- FC2_COLOR_PROCESSING_ALGORITHM_FORCE_32BITS
 - Enumerations, [106](#)
- FC2_DEFAULT
 - Enumerations, [106](#)
- FC2_DIRECTIONAL
 - Enumerations, [106](#)
- FC2_DRIVER_1394_CAM
 - Enumerations, [107](#)
- FC2_DRIVER_1394_JUJU
 - Enumerations, [107](#)
- FC2_DRIVER_1394_PRO
 - Enumerations, [107](#)
- FC2_DRIVER_1394_RAW1394
 - Enumerations, [107](#)
- FC2_DRIVER_1394_VIDEO1394
 - Enumerations, [107](#)
- FC2_DRIVER_FORCE_32BITS
 - Enumerations, [107](#)
- FC2_DRIVER_GIGE_FILTER
 - Enumerations, [107](#)

- FC2_DRIVER_GIGE_LWF
 - Enumerations, [107](#)
- FC2_DRIVER_GIGE_NONE
 - Enumerations, [107](#)
- FC2_DRIVER_GIGE_PRO
 - Enumerations, [107](#)
- FC2_DRIVER_UNKNOWN
 - Enumerations, [107](#)
- FC2_DRIVER_USB3_PRO
 - Enumerations, [107](#)
- FC2_DRIVER_USB_CAM
 - Enumerations, [107](#)
- FC2_DRIVER_USB_NONE
 - Enumerations, [107](#)
- FC2_DROP_FRAMES
 - Enumerations, [109](#)
- FC2_EDGE_SENSING
 - Enumerations, [106](#)
- FC2_ERROR_BUFFER_TOO_SMALL
 - Enumerations, [108](#)
- FC2_ERROR_BUS_MASTER_FAILED
 - Enumerations, [108](#)
- FC2_ERROR_FAILED_BUS_MASTER_CONNECTION
 - Enumerations, [107](#)
- FC2_ERROR_FAILED_GUID
 - Enumerations, [107](#)
- FC2_ERROR_FAILED
 - Enumerations, [107](#)
- FC2_ERROR_FORCE_32BITS
 - Enumerations, [108](#)
- FC2_ERROR_IIDC_FAILED
 - Enumerations, [108](#)
- FC2_ERROR_IMAGE_CONSISTENCY_ERROR
 - Enumerations, [108](#)
- FC2_ERROR_IMAGE_CONVERSION_FAILED
 - Enumerations, [108](#)
- FC2_ERROR_IMAGE_LIBRARY_FAILURE
 - Enumerations, [108](#)
- FC2_ERROR_INCOMPATIBLE_DRIVER
 - Enumerations, [108](#)
- FC2_ERROR_INIT_FAILED
 - Enumerations, [107](#)
- FC2_ERROR_INVALID_BUS_MANAGER
 - Enumerations, [107](#)
- FC2_ERROR_INVALID_GENERATION
 - Enumerations, [108](#)
- FC2_ERROR_INVALID_MODE
 - Enumerations, [107](#)
- FC2_ERROR_INVALID_PACKET_SIZE
 - Enumerations, [107](#)
- FC2_ERROR_INVALID_PARAMETER
 - Enumerations, [107](#)
- FC2_ERROR_INVALID_SETTINGS
 - Enumerations, [107](#)
- FC2_ERROR_ISOCH_ALREADY_STARTED
 - Enumerations, [108](#)
- FC2_ERROR_ISOCH_BANDWIDTH_EXCEEDED
 - Enumerations, [108](#)
- FC2_ERROR_ISOCH_FAILED
 - Enumerations, [108](#)
- FC2_ERROR_ISOCH_NOT_STARTED
 - Enumerations, [108](#)
- FC2_ERROR_ISOCH_RETRIEVE_BUFFER_FAILED
 - Enumerations, [108](#)
- FC2_ERROR_ISOCH_START_FAILED
 - Enumerations, [108](#)
- FC2_ERROR_ISOCH_STOP_FAILED
 - Enumerations, [108](#)
- FC2_ERROR_ISOCH_SYNC_FAILED
 - Enumerations, [108](#)
- FC2_ERROR_LOW_LEVEL_FAILURE
 - Enumerations, [107](#)
- FC2_ERROR_LUT_FAILED
 - Enumerations, [108](#)
- FC2_ERROR_MEMORY_ALLOCATION_FAILED
 - Enumerations, [107](#)
- FC2_ERROR_NOT_CONNECTED
 - Enumerations, [107](#)
- FC2_ERROR_NOT_FOUND
 - Enumerations, [107](#)
- FC2_ERROR_NOT_IMPLEMENTED
 - Enumerations, [107](#)
- FC2_ERROR_NOT_IN_FORMAT7
 - Enumerations, [107](#)
- FC2_ERROR_NOT_INITIALIZED
 - Enumerations, [107](#)
- FC2_ERROR_NOT_SUPPORTED
 - Enumerations, [107](#)
- FC2_ERROR_OK
 - Enumerations, [107](#)
- FC2_ERROR_PROPERTY_FAILED
 - Enumerations, [108](#)
- FC2_ERROR_PROPERTY_NOT_PRESENT
 - Enumerations, [108](#)
- FC2_ERROR_READ_REGISTER_FAILED
 - Enumerations, [108](#)
- FC2_ERROR_REGISTER_FAILED
 - Enumerations, [108](#)
- FC2_ERROR_STROBE_FAILED
 - Enumerations, [108](#)
- FC2_ERROR_TIMEOUT
 - Enumerations, [107](#)
- FC2_ERROR_TRIGGER_FAILED
 - Enumerations, [108](#)
- FC2_ERROR_UNDEFINED
 - Enumerations, [107](#)
- FC2_ERROR_WRITE_REGISTER_FAILED
 - Enumerations, [108](#)
- FC2_FOCUS
 - Enumerations, [112](#)
- FC2_FRAME_RATE
 - Enumerations, [112](#)
- FC2_FRAMERATE_120
 - Enumerations, [108](#)
- FC2_FRAMERATE_15
 - Enumerations, [108](#)

- FC2_FRAMERATE_1_875
 - Enumerations, [108](#)
- FC2_FRAMERATE_240
 - Enumerations, [108](#)
- FC2_FRAMERATE_30
 - Enumerations, [108](#)
- FC2_FRAMERATE_3_75
 - Enumerations, [108](#)
- FC2_FRAMERATE_60
 - Enumerations, [108](#)
- FC2_FRAMERATE_7_5
 - Enumerations, [108](#)
- FC2_FRAMERATE_FORCE_32BITS
 - Enumerations, [108](#)
- FC2_FRAMERATE_FORMAT7
 - Enumerations, [108](#)
- FC2_FROM_FILE_EXT
 - Enumerations, [109](#)
- FC2_GAIN
 - Enumerations, [112](#)
- FC2_GAMMA
 - Enumerations, [112](#)
- FC2_GRAB_MODE_FORCE_32BITS
 - Enumerations, [109](#)
- FC2_GRAB_TIMEOUT_FORCE_32BITS
 - Enumerations, [109](#)
- FC2_HEARTBEAT_TIMEOUT
 - GigE specific enumerations, [114](#)
- FC2_HEARTBEAT
 - GigE specific enumerations, [114](#)
- FC2_HQ_LINEAR
 - Enumerations, [106](#)
- FC2_HUE
 - Enumerations, [112](#)
- FC2_IMAGE_FILE_FORMAT_FORCE_32BITS
 - Enumerations, [109](#)
- FC2_INTERFACE_GIGE
 - Enumerations, [110](#)
- FC2_INTERFACE_IEEE1394
 - Enumerations, [110](#)
- FC2_INTERFACE_TYPE_FORCE_32BITS
 - Enumerations, [110](#)
- FC2_INTERFACE_UNKNOWN
 - Enumerations, [110](#)
- FC2_INTERFACE_USB_2
 - Enumerations, [110](#)
- FC2_INTERFACE_USB_3
 - Enumerations, [110](#)
- FC2_IPP
 - Enumerations, [106](#)
- FC2_IRIS
 - Enumerations, [112](#)
- FC2_JPEG2000
 - Enumerations, [109](#)
- FC2_JPEG
 - Enumerations, [109](#)
- FC2_LINUX_X64
 - FlyCapture2Defs_C.h, [188](#)
- FC2_LINUX_X86
 - FlyCapture2Defs_C.h, [188](#)
- FC2_MAC
 - FlyCapture2Defs_C.h, [188](#)
- FC2_MODE_0
 - Enumerations, [110](#)
- FC2_MODE_1
 - Enumerations, [110](#)
- FC2_MODE_10
 - Enumerations, [110](#)
- FC2_MODE_11
 - Enumerations, [110](#)
- FC2_MODE_12
 - Enumerations, [110](#)
- FC2_MODE_13
 - Enumerations, [110](#)
- FC2_MODE_14
 - Enumerations, [110](#)
- FC2_MODE_15
 - Enumerations, [110](#)
- FC2_MODE_16
 - Enumerations, [110](#)
- FC2_MODE_17
 - Enumerations, [110](#)
- FC2_MODE_18
 - Enumerations, [110](#)
- FC2_MODE_19
 - Enumerations, [110](#)
- FC2_MODE_2
 - Enumerations, [110](#)
- FC2_MODE_20
 - Enumerations, [110](#)
- FC2_MODE_21
 - Enumerations, [110](#)
- FC2_MODE_22
 - Enumerations, [110](#)
- FC2_MODE_23
 - Enumerations, [110](#)
- FC2_MODE_24
 - Enumerations, [110](#)
- FC2_MODE_25
 - Enumerations, [110](#)
- FC2_MODE_26
 - Enumerations, [110](#)
- FC2_MODE_27
 - Enumerations, [110](#)
- FC2_MODE_28
 - Enumerations, [110](#)
- FC2_MODE_29
 - Enumerations, [110](#)
- FC2_MODE_3
 - Enumerations, [110](#)
- FC2_MODE_30
 - Enumerations, [110](#)
- FC2_MODE_31
 - Enumerations, [110](#)
- FC2_MODE_4
 - Enumerations, [110](#)

- FC2_MODE_5
 - Enumerations, [110](#)
- FC2_MODE_6
 - Enumerations, [110](#)
- FC2_MODE_7
 - Enumerations, [110](#)
- FC2_MODE_8
 - Enumerations, [110](#)
- FC2_MODE_9
 - Enumerations, [110](#)
- FC2_MODE_FORCE_32BITS
 - Enumerations, [110](#)
- FC2_NEAREST_NEIGHBOR_FAST
 - Enumerations, [106](#)
- FC2_NO_COLOR_PROCESSING
 - Enumerations, [106](#)
- FC2_NUM_FRAMERATES
 - Enumerations, [108](#)
- FC2_NUM_MODES
 - Enumerations, [110](#)
- FC2_NUM_PIXEL_FORMATS
 - Enumerations, [111](#)
- FC2_NUM_VIDEOMODES
 - Enumerations, [113](#)
- FC2_OSTYPE_FORCE_32BITS
 - FlyCapture2Defs_C.h, [188](#)
- FC2_PAN
 - Enumerations, [112](#)
- FC2_PCIE_BUSSPEED_2_5
 - Enumerations, [111](#)
- FC2_PCIE_BUSSPEED_5_0
 - Enumerations, [111](#)
- FC2_PCIE_BUSSPEED_FORCE_32BITS
 - Enumerations, [111](#)
- FC2_PCIE_BUSSPEED_UNKNOWN
 - Enumerations, [111](#)
- FC2_PGM
 - Enumerations, [109](#)
- FC2_PIXEL_FORMAT_411YUV8
 - Enumerations, [111](#)
- FC2_PIXEL_FORMAT_422YUV8
 - Enumerations, [111](#)
- FC2_PIXEL_FORMAT_422YUV8_JPEG
 - Enumerations, [111](#)
- FC2_PIXEL_FORMAT_444YUV8
 - Enumerations, [111](#)
- FC2_PIXEL_FORMAT_BGR16
 - Enumerations, [111](#)
- FC2_PIXEL_FORMAT_BGRU16
 - Enumerations, [111](#)
- FC2_PIXEL_FORMAT_BGRU
 - Enumerations, [111](#)
- FC2_PIXEL_FORMAT_BGR
 - Enumerations, [111](#)
- FC2_PIXEL_FORMAT_MONO12
 - Enumerations, [111](#)
- FC2_PIXEL_FORMAT_MONO16
 - Enumerations, [111](#)
- FC2_PIXEL_FORMAT_MONO8
 - Enumerations, [111](#)
- FC2_PIXEL_FORMAT_RAW12
 - Enumerations, [111](#)
- FC2_PIXEL_FORMAT_RAW16
 - Enumerations, [111](#)
- FC2_PIXEL_FORMAT_RAW8
 - Enumerations, [111](#)
- FC2_PIXEL_FORMAT_RGB16
 - Enumerations, [111](#)
- FC2_PIXEL_FORMAT_RGB8
 - Enumerations, [111](#)
- FC2_PIXEL_FORMAT_RGBU
 - Enumerations, [111](#)
- FC2_PIXEL_FORMAT_RGB
 - Enumerations, [111](#)
- FC2_PIXEL_FORMAT_S_MONO16
 - Enumerations, [111](#)
- FC2_PIXEL_FORMAT_S_RGB16
 - Enumerations, [111](#)
- FC2_PNG
 - Enumerations, [109](#)
- FC2_PPM
 - Enumerations, [109](#)
- FC2_PROPERTY_TYPE_FORCE_32BITS
 - Enumerations, [112](#)
- FC2_RAW
 - Enumerations, [109](#)
- FC2_REMOVAL
 - Enumerations, [106](#)
- FC2_RIGOROUS
 - Enumerations, [106](#)
- FC2_SATURATION
 - Enumerations, [112](#)
- FC2_SHARPNESS
 - Enumerations, [112](#)
- FC2_SHUTTER
 - Enumerations, [112](#)
- FC2_STATISTICS_BLUE
 - FlyCapture2Defs_C.h, [189](#)
- FC2_STATISTICS_FORCE_32BITS
 - FlyCapture2Defs_C.h, [189](#)
- FC2_STATISTICS_GREEN
 - FlyCapture2Defs_C.h, [189](#)
- FC2_STATISTICS_GREY
 - FlyCapture2Defs_C.h, [189](#)
- FC2_STATISTICS_HUE
 - FlyCapture2Defs_C.h, [189](#)
- FC2_STATISTICS_LIGHTNESS
 - FlyCapture2Defs_C.h, [189](#)
- FC2_STATISTICS_RED
 - FlyCapture2Defs_C.h, [189](#)
- FC2_STATISTICS_SATURATION
 - FlyCapture2Defs_C.h, [189](#)
- FC2_TEMPERATURE
 - Enumerations, [112](#)
- FC2_TIFF_ADOBE_DEFLATE
 - Image saving structures., [120](#)

- FC2_TIFF_CCITTFAX3
 - Image saving structures., [120](#)
- FC2_TIFF_CCITTFAX4
 - Image saving structures., [120](#)
- FC2_TIFF_DEFLATE
 - Image saving structures., [120](#)
- FC2_TIFF_JPEG
 - Image saving structures., [120](#)
- FC2_TIFF_LZW
 - Image saving structures., [120](#)
- FC2_TIFF_NONE
 - Image saving structures., [120](#)
- FC2_TIFF_PACKBITS
 - Image saving structures., [120](#)
- FC2_TIFF
 - Enumerations, [109](#)
- FC2_TILT
 - Enumerations, [112](#)
- FC2_TIMEOUT_INFINITE
 - Enumerations, [109](#)
- FC2_TIMEOUT_NONE
 - Enumerations, [109](#)
- FC2_TIMEOUT_UNSPECIFIED
 - Enumerations, [109](#)
- FC2_TRIGGER_DELAY
 - Enumerations, [112](#)
- FC2_TRIGGER_MODE
 - Enumerations, [112](#)
- FC2_UNKNOWN_OS
 - FlyCapture2Defs_C.h, [188](#)
- FC2_UNSPECIFIED_GRAB_MODE
 - Enumerations, [109](#)
- FC2_UNSPECIFIED_PIXEL_FORMAT
 - Enumerations, [111](#)
- FC2_UNSPECIFIED_PROPERTY_TYPE
 - Enumerations, [112](#)
- FC2_VIDEOMODE_1024x768RGB
 - Enumerations, [112](#)
- FC2_VIDEOMODE_1024x768Y16
 - Enumerations, [113](#)
- FC2_VIDEOMODE_1024x768Y8
 - Enumerations, [113](#)
- FC2_VIDEOMODE_1024x768YUV422
 - Enumerations, [112](#)
- FC2_VIDEOMODE_1280x960RGB
 - Enumerations, [113](#)
- FC2_VIDEOMODE_1280x960Y16
 - Enumerations, [113](#)
- FC2_VIDEOMODE_1280x960Y8
 - Enumerations, [113](#)
- FC2_VIDEOMODE_1280x960YUV422
 - Enumerations, [113](#)
- FC2_VIDEOMODE_1600x1200RGB
 - Enumerations, [113](#)
- FC2_VIDEOMODE_1600x1200Y16
 - Enumerations, [113](#)
- FC2_VIDEOMODE_1600x1200Y8
 - Enumerations, [113](#)
- FC2_VIDEOMODE_1600x1200YUV422
 - Enumerations, [113](#)
- FC2_VIDEOMODE_160x120YUV444
 - Enumerations, [112](#)
- FC2_VIDEOMODE_320x240YUV422
 - Enumerations, [112](#)
- FC2_VIDEOMODE_640x480RGB
 - Enumerations, [112](#)
- FC2_VIDEOMODE_640x480Y16
 - Enumerations, [112](#)
- FC2_VIDEOMODE_640x480Y8
 - Enumerations, [112](#)
- FC2_VIDEOMODE_640x480YUV411
 - Enumerations, [112](#)
- FC2_VIDEOMODE_640x480YUV422
 - Enumerations, [112](#)
- FC2_VIDEOMODE_800x600RGB
 - Enumerations, [112](#)
- FC2_VIDEOMODE_800x600Y16
 - Enumerations, [112](#)
- FC2_VIDEOMODE_800x600Y8
 - Enumerations, [112](#)
- FC2_VIDEOMODE_800x600YUV422
 - Enumerations, [112](#)
- FC2_VIDEOMODE_FORCE_32BITS
 - Enumerations, [113](#)
- FC2_VIDEOMODE_FORMAT7
 - Enumerations, [113](#)
- FC2_WHITE_BALANCE
 - Enumerations, [112](#)
- FC2_WINDOWS_X64
 - FlyCapture2Defs_C.h, [188](#)
- FC2_WINDOWS_X86
 - FlyCapture2Defs_C.h, [188](#)
- FC2_ZOOM
 - Enumerations, [112](#)
- FLYCAPTURE2_C_API
 - FlyCapture2Platform_C.h, [193](#)
- FLYCAPTURE2_C_CALL_CONVEN
 - FlyCapture2Platform_C.h, [193](#)
- FULL_32BIT_VALUE
 - MultiSyncLibraryDefs_C.h, [198](#)
 - TypeDefs, [98](#)
- fc2AVIAppend
 - AVI Recording Operation, [84](#)
- fc2AVIClose
 - AVI Recording Operation, [84](#)
- fc2AVIContext
 - TypeDefs, [98](#)
- fc2AVIOpen
 - AVI Recording Operation, [85](#)
- fc2AVIOption, [121](#)
 - frameRate, [121](#)
 - reserved, [121](#)
- fc2AsyncCommandCallback
 - Image saving structures., [120](#)
- fc2BMPOption, [122](#)
 - indexedColor_8bit, [122](#)

- reserved, 122
- fc2BandwidthAllocation
 - Enumerations, 105
- fc2BayerTileFormat
 - Enumerations, 105
- fc2BusCallbackType
 - Enumerations, 105
- fc2BusEventCallback
 - Image saving structures., 120
- fc2BusSpeed
 - Enumerations, 106
- fc2ByteOrder
 - FlyCapture2Defs_C.h, 188
- fc2CalculateImageStatistics
 - Image Operation, 71
- fc2CallbackHandle
 - Image saving structures., 120
- fc2CameraEventCallback
 - Image saving structures., 120
- fc2CameraInfo, 122
 - applicationIPAddress, 124
 - applicationPort, 124
 - bayerTileFormat, 124
 - busNumber, 124
 - ccpStatus, 124
 - configROM, 124
 - defaultGateway, 125
 - driverName, 125
 - driverType, 125
 - firmwareBuildTime, 125
 - firmwareVersion, 125
 - gigEMajorVersion, 125
 - gigEMinorVersion, 125
 - iidcVer, 125
 - interfaceType, 125
 - ipAddress, 125
 - isColorCamera, 126
 - macAddress, 126
 - maximumBusSpeed, 126
 - modelName, 126
 - nodeNumber, 126
 - pcieBusSpeed, 126
 - reserved, 126
 - sensorInfo, 126
 - sensorResolution, 126
 - serialNumber, 126
 - subnetMask, 127
 - userDefinedName, 127
 - vendorName, 127
 - xmlURL1, 127
 - xmlURL2, 127
- fc2CameraStats, 127
 - cameraCurrents, 128
 - cameraPowerUp, 128
 - cameraVoltages, 128
 - imageCorrupt, 128
 - imageDriverDropped, 128
 - imageDropped, 128
 - imageXmitFailed, 128
 - numCurrents, 128
 - numResendPacketsReceived, 128
 - numResendPacketsRequested, 129
 - numVoltages, 129
 - portErrors, 129
 - regReadFailed, 129
 - regWriteFailed, 129
 - reserved, 129
 - temperature, 129
 - timeSinceBusReset, 129
 - timeSinceInitialization, 129
 - timeStamp, 129
- fc2CheckDriver
 - Utilities, 93
- fc2ColorProcessingAlgorithm
 - Enumerations, 106
- fc2Config, 129
 - asyncBusSpeed, 130
 - bandwidthAllocation, 130
 - grabMode, 130
 - grabTimeout, 130
 - highPerformanceRetrieveBuffer, 131
 - isochBusSpeed, 131
 - minNumImageNotifications, 131
 - numBuffers, 131
 - numImageNotifications, 131
 - registerTimeout, 131
 - registerTimeoutRetries, 132
 - reserved, 132
- fc2ConfigROM, 132
 - chipIdHi, 133
 - chipIdLo, 133
 - nodeVendorId, 133
 - pszKeyword, 133
 - reserved, 133
 - unitSWVer, 133
 - unitSpecId, 133
 - unitSubSWVer, 133
 - vendorUniqueInfo_0, 133
 - vendorUniqueInfo_1, 133
 - vendorUniqueInfo_2, 134
 - vendorUniqueInfo_3, 134
- fc2Connect
 - Connection and Image Retrieval, 19
- fc2Context
 - TypeDefs, 98
- fc2ConvertImage
 - Image Operation, 71
- fc2ConvertImageTo
 - Image Operation, 71
- fc2CreateAVI
 - AVI Recording Operation, 85
- fc2CreateGUIContext
 - FlyCapture2GUI_C.h, 190
- fc2CreateImage
 - Image Operation, 72
- fc2CreateImageStatistics

- Image Statistics Operation, [78](#)
- fc2CreateTopologyNode
 - TopologyNode Operation, [88](#)
- fc2DestroyAVI
 - AVI Recording Operation, [85](#)
- fc2DestroyGUIContext
 - FlyCapture2GUI_C.h, [190](#)
- fc2DestroyImage
 - Image Operation, [72](#)
- fc2DestroyImageStatistics
 - Image Statistics Operation, [78](#)
- fc2DestroyTopologyNode
 - TopologyNode Operation, [88](#)
- fc2DetermineBitsPerPixel
 - Image Operation, [72](#)
- fc2Disconnect
 - Connection and Image Retrieval, [20](#)
- fc2DiscoverGigECameras
 - Bus Manager Operation, [10](#)
- fc2DiscoverGigEPacketSize
 - GigE property manipulation, [60](#)
- fc2Disonnect
 - FlyCapture2GUI_C.h, [190](#)
- fc2DriverType
 - Enumerations, [106](#)
- fc2EmbeddedImageInfo, [134](#)
 - brightness, [135](#)
 - exposure, [135](#)
 - frameCounter, [135](#)
 - GPIOPinState, [135](#)
 - gain, [135](#)
 - ROIPosition, [135](#)
 - shutter, [135](#)
 - strobePattern, [135](#)
 - timestamp, [135](#)
 - whiteBalance, [135](#)
- fc2EmbeddedImageInfoProperty, [135](#)
 - available, [136](#)
 - onOff, [136](#)
- fc2EnableLUT
 - Look Up Table, [40](#)
- fc2Error
 - Enumerations, [107](#)
- fc2ErrorToDescription
 - Utilities, [94](#)
- fc2EventCallbackData, [136](#)
 - EventData, [136](#)
 - EventDataSize, [136](#)
 - EventID, [136](#)
 - EventName, [137](#)
 - EventTimestamp, [137](#)
 - EventUserData, [137](#)
 - EventUserDataSize, [137](#)
- fc2EventOptions, [137](#)
 - EventCallbackFcn, [138](#)
 - EventName, [138](#)
 - EventUserData, [138](#)
 - EventUserDataSize, [138](#)
- fc2FireBusReset
 - Bus Manager Operation, [11](#)
- fc2FireSoftwareTrigger
 - Trigger, [31](#)
- fc2FireSoftwareTriggerBroadcast
 - Trigger, [32](#)
- fc2ForceAllIPAddressesAutomatically
 - Bus Manager Operation, [11](#)
- fc2ForceIPAddressAutomatically
 - Bus Manager Operation, [11](#)
- fc2ForceIPAddressToCamera
 - Bus Manager Operation, [11](#)
- fc2Format7ImageSettings, [138](#)
 - height, [139](#)
 - mode, [139](#)
 - offsetX, [139](#)
 - offsetY, [139](#)
 - pixelFormat, [139](#)
 - reserved, [139](#)
 - width, [139](#)
- fc2Format7Info, [139](#)
 - imageHStepSize, [140](#)
 - imageVStepSize, [140](#)
 - maxHeight, [140](#)
 - maxPacketSize, [141](#)
 - maxWidth, [141](#)
 - minPacketSize, [141](#)
 - mode, [141](#)
 - offsetHStepSize, [141](#)
 - offsetVStepSize, [141](#)
 - packetSize, [141](#)
 - percentage, [141](#)
 - pixelFormatBitField, [141](#)
 - reserved, [141](#)
 - vendorPixelFormatBitField, [142](#)
- fc2Format7PacketInfo, [142](#)
 - maxBytesPerPacket, [142](#)
 - recommendedBytesPerPacket, [142](#)
 - reserved, [142](#)
 - unitBytesPerPacket, [143](#)
- fc2FrameRate
 - Enumerations, [108](#)
- fc2GUIConnect
 - FlyCapture2GUI_C.h, [190](#)
- fc2GUIDisconnect
 - FlyCapture2GUI_C.h, [191](#)
- fc2GetActiveLUTBank
 - Look Up Table, [41](#)
- fc2GetCameraFromIPAddress
 - Bus Manager Operation, [12](#)
- fc2GetCameraFromIndex
 - Bus Manager Operation, [12](#)
- fc2GetCameraFromSerialNumber
 - Bus Manager Operation, [13](#)
- fc2GetCameraInfo
 - Information and Properties, [26](#)
- fc2GetCameraSerialNumberFromIndex
 - Bus Manager Operation, [13](#)

- fc2GetChannelHistogram
 - Image Statistics Operation, [78](#)
- fc2GetChannelMean
 - Image Statistics Operation, [79](#)
- fc2GetChannelNumPixelValues
 - Image Statistics Operation, [79](#)
- fc2GetChannelPixelValueRange
 - Image Statistics Operation, [79](#)
- fc2GetChannelRange
 - Image Statistics Operation, [80](#)
- fc2GetChannelStatus
 - Image Statistics Operation, [80](#)
- fc2GetConfiguration
 - Connection and Image Retrieval, [20](#)
- fc2GetDefaultColorProcessing
 - Image Operation, [73](#)
- fc2GetDefaultOutputFormat
 - Image Operation, [73](#)
- fc2GetDeviceFromIndex
 - Bus Manager Operation, [13](#)
- fc2GetDriverDeviceName
 - Utilities, [94](#)
- fc2GetEmbeddedImageInfo
 - Memory Channels, [44](#)
- fc2GetFormat7Configuration
 - Format7, [53](#)
- fc2GetFormat7Info
 - Format7, [53](#)
- fc2GetGPIOPinDirection
 - General Purpose Input / Output, [29](#)
- fc2GetGigEConfig
 - GigE image stream configuration, [67](#)
- fc2GetGigEImageBinningSettings
 - GigE image binning settings, [65](#)
- fc2GetGigEImageSettings
 - GigE image settings, [62](#)
- fc2GetGigEImageSettingsInfo
 - GigE image settings, [62](#)
- fc2GetGigEImagingMode
 - GigE image settings, [63](#)
- fc2GetGigEProperty
 - GigE property manipulation, [60](#)
- fc2GetGigEStreamChannelInfo
 - GigE image stream configuration, [67](#)
- fc2GetImageData
 - Image Operation, [73](#)
- fc2GetImageStatistics
 - Image Statistics Operation, [80](#)
- fc2GetImageTimeStamp
 - Image Operation, [74](#)
- fc2GetInterfaceTypeFromGuid
 - Bus Manager Operation, [14](#)
- fc2GetLUTBankInfo
 - Look Up Table, [41](#)
- fc2GetLUTChannel
 - Look Up Table, [41](#)
- fc2GetLUTInfo
 - Look Up Table, [42](#)
- fc2GetLibraryVersion
 - Utilities, [94](#)
- fc2GetMemoryChannel
 - Memory Channels, [45](#)
- fc2GetMemoryChannelInfo
 - Memory Channels, [45](#)
- fc2GetNumOfCameras
 - Bus Manager Operation, [14](#)
- fc2GetNumOfDevices
 - Bus Manager Operation, [14](#)
- fc2GetNumStreamChannels
 - GigE image stream configuration, [68](#)
- fc2GetProperty
 - Information and Properties, [26](#)
- fc2GetPropertyInfo
 - Information and Properties, [27](#)
- fc2GetRegisterString
 - Register Operation, [48](#)
- fc2GetStrobe
 - Strobe, [37](#)
- fc2GetStrobeInfo
 - Strobe, [37](#)
- fc2GetSystemInfo
 - Utilities, [94](#)
- fc2GetTopology
 - Bus Manager Operation, [15](#)
- fc2GetTriggerDelay
 - Trigger, [32](#)
- fc2GetTriggerDelayInfo
 - Trigger, [32](#)
- fc2GetTriggerMode
 - Trigger, [33](#)
- fc2GetTriggerModelInfo
 - Trigger, [33](#)
- fc2GetUsbLinkInfo
 - Bus Manager Operation, [15](#)
- fc2GetUsbPortStatus
 - Bus Manager Operation, [15](#)
- fc2GetVideoModeAndFrameRate
 - DCAM Formats, [51](#)
- fc2GetVideoModeAndFrameRateInfo
 - DCAM Formats, [51](#)
- fc2GigEConfig, [143](#)
 - enablePacketResend, [143](#)
 - registerTimeout, [143](#)
 - registerTimeoutRetries, [143](#)
 - reserved, [144](#)
- fc2GigEImageSettings, [144](#)
 - height, [144](#)
 - offsetX, [144](#)
 - offsetY, [144](#)
 - pixelFormat, [145](#)
 - reserved, [145](#)
 - width, [145](#)
- fc2GigEImageSettingsInfo, [145](#)
 - imageHStepSize, [146](#)
 - imageVStepSize, [146](#)
 - maxHeight, [146](#)

- maxWidth, [146](#)
- offsetHStepSize, [146](#)
- offsetVStepSize, [146](#)
- pixelFormatBitField, [146](#)
- reserved, [146](#)
- vendorPixelFormatBitField, [146](#)
- fc2GigEProperty, [147](#)
 - isReadable, [147](#)
 - isWritable, [147](#)
 - max, [147](#)
 - min, [148](#)
 - propType, [148](#)
 - reserved, [148](#)
 - value, [148](#)
- fc2GigEPropertyType
 - GigE specific enumerations, [114](#)
- fc2GigEStreamChannel, [148](#)
 - destinationIpAddress, [149](#)
 - doNotFragment, [149](#)
 - hostPort, [149](#)
 - interPacketDelay, [149](#)
 - networkInterfaceIndex, [149](#)
 - packetSize, [149](#)
 - reserved, [149](#)
 - sourcePort, [149](#)
- fc2GrabMode
 - Enumerations, [108](#)
- fc2GrabTimeout
 - Enumerations, [109](#)
- fc2GuiContext
 - TypeDefs, [99](#)
- fc2H264Open
 - AVI Recording Operation, [86](#)
- fc2H264Option, [150](#)
 - bitrate, [150](#)
 - frameRate, [150](#)
 - height, [150](#)
 - reserved, [150](#)
 - width, [150](#)
- fc2Hide
 - FlyCapture2GUI_C.h, [191](#)
- fc2IPAddress, [155](#)
 - octets, [155](#)
- fc2Image, [151](#)
 - bayerFormat, [151](#)
 - cols, [151](#)
 - dataSize, [151](#)
 - format, [151](#)
 - imageImpl, [151](#)
 - pData, [151](#)
 - receivedDataSize, [151](#)
 - rows, [151](#)
 - stride, [151](#)
- fc2ImageEventCallback
 - Image saving structures., [120](#)
- fc2ImageFileFormat
 - Enumerations, [109](#)
- fc2ImageImpl
 - TypeDefs, [99](#)
- fc2ImageMetadata, [151](#)
 - embeddedBrightness, [152](#)
 - embeddedExposure, [152](#)
 - embeddedFrameCounter, [152](#)
 - embeddedGPIOPinState, [152](#)
 - embeddedGain, [152](#)
 - embeddedROIPosition, [153](#)
 - embeddedShutter, [153](#)
 - embeddedStrobePattern, [153](#)
 - embeddedTimeStamp, [153](#)
 - embeddedWhiteBalance, [153](#)
 - reserved, [153](#)
- fc2ImageStatisticsContext
 - TypeDefs, [99](#)
- fc2ImageStatisticsDisableAll
 - Image Statistics Operation, [81](#)
- fc2ImageStatisticsEnableAll
 - Image Statistics Operation, [81](#)
- fc2ImageStatisticsEnableGreyOnly
 - Image Statistics Operation, [81](#)
- fc2ImageStatisticsEnableHSLOnly
 - Image Statistics Operation, [82](#)
- fc2ImageStatisticsEnableRGBOnly
 - Image Statistics Operation, [82](#)
- fc2InterfaceType
 - Enumerations, [109](#)
- fc2InternalContext, [153](#)
 - pBusMgr, [154](#)
 - pCamera, [154](#)
- fc2InternalGuiContext, [154](#)
 - pCameraControlDlg, [154](#)
 - pCameraSelectionDlg, [154](#)
- fc2InternalImageCallback, [154](#)
 - pCallback, [154](#)
 - pCallbackData, [154](#)
- fc2IsCameraControlable
 - Bus Manager Operation, [16](#)
- fc2IsVisible
 - FlyCapture2GUI_C.h, [191](#)
- fc2JPEGOption, [155](#)
 - progressive, [155](#)
 - quality, [155](#)
 - reserved, [156](#)
- fc2JPG2Option, [156](#)
 - quality, [156](#)
 - reserved, [156](#)
- fc2LUTData, [157](#)
 - enabled, [157](#)
 - inputBitDepth, [157](#)
 - numBanks, [157](#)
 - numChannels, [158](#)
 - numEntries, [158](#)
 - outputBitDepth, [158](#)
 - reserved, [158](#)
 - supported, [158](#)
- fc2LaunchBrowser
 - Utilities, [96](#)

- fc2LaunchCommand
 - Utilities, [96](#)
- fc2LaunchCommandAsync
 - Utilities, [96](#)
- fc2LaunchHelp
 - Utilities, [97](#)
- fc2MACAddress, [158](#)
 - octets, [159](#)
- fc2MJPEGOpen
 - AVI Recording Operation, [86](#)
- fc2MJPEGOption, [159](#)
 - frameRate, [159](#)
 - quality, [159](#)
 - reserved, [159](#)
- fc2Mode
 - Enumerations, [110](#)
- fc2NodeType
 - FlyCapture2Defs_C.h, [188](#)
- fc2OSType
 - FlyCapture2Defs_C.h, [188](#)
- fc2PCleBusSpeed
 - Enumerations, [110](#)
- fc2PGMOption, [160](#)
 - binaryFile, [160](#)
 - reserved, [160](#)
- fc2PGRGuid, [160](#)
 - value, [161](#)
- fc2PNGOption, [161](#)
 - compressionLevel, [161](#)
 - interlaced, [161](#)
 - reserved, [161](#)
- fc2PPMOption, [162](#)
 - binaryFile, [162](#)
 - reserved, [162](#)
- fc2PixelFormat
 - Enumerations, [111](#)
- fc2PortType
 - FlyCapture2Defs_C.h, [188](#)
- fc2PropertyType
 - Enumerations, [111](#)
- fc2QueryGigEImagingMode
 - GigE image settings, [63](#)
- fc2ReadGVCPMemory
 - GVCP Register Operation, [56](#)
- fc2ReadGVCPRegister
 - GVCP Register Operation, [57](#)
- fc2ReadGVCPRegisterBlock
 - GVCP Register Operation, [57](#)
- fc2ReadPhyRegister
 - Bus Manager Operation, [16](#)
- fc2ReadRegister
 - Register Operation, [48](#)
- fc2ReadRegisterBlock
 - Register Operation, [49](#)
- fc2RegisterCallback
 - Bus Manager Operation, [16](#)
- fc2RescanBus
 - Bus Manager Operation, [17](#)
- fc2RestoreFromMemoryChannel
 - Memory Channels, [45](#)
- fc2RetrieveBuffer
 - Connection and Image Retrieval, [20](#)
- fc2SavelImage
 - Image Operation, [74](#)
- fc2SavelImageWithOptions
 - Image Operation, [74](#)
- fc2SaveToMemoryChannel
 - Memory Channels, [46](#)
- fc2SetActiveLUTBank
 - Look Up Table, [42](#)
- fc2SetCallback
 - Connection and Image Retrieval, [21](#)
- fc2SetChannelStatus
 - Image Statistics Operation, [82](#)
- fc2SetConfiguration
 - Connection and Image Retrieval, [21](#)
- fc2SetDefaultColorProcessing
 - Image Operation, [74](#)
- fc2SetDefaultOutputFormat
 - Image Operation, [75](#)
- fc2SetEmbeddedImageInfo
 - Memory Channels, [46](#)
- fc2SetFormat7Configuration
 - Format7, [54](#)
- fc2SetFormat7ConfigurationPacket
 - Format7, [54](#)
- fc2SetGPIOPinDirection
 - General Purpose Input / Output, [29](#)
- fc2SetGPIOPinDirectionBroadcast
 - General Purpose Input / Output, [30](#)
- fc2SetGigEConfig
 - GigE image stream configuration, [68](#)
- fc2SetGigEImageBinningSettings
 - GigE image binning settings, [65](#)
- fc2SetGigEImageSettings
 - GigE image settings, [63](#)
- fc2SetGigEImagingMode
 - GigE image settings, [64](#)
- fc2SetGigEProperty
 - GigE property manipulation, [61](#)
- fc2SetGigEStreamChannelInfo
 - GigE image stream configuration, [68](#)
- fc2SetImageData
 - Image Operation, [75](#)
- fc2SetImageDimensions
 - Image Operation, [76](#)
- fc2SetLUTChannel
 - Look Up Table, [42](#)
- fc2SetProperty
 - Information and Properties, [27](#)
- fc2SetPropertyBroadcast
 - Information and Properties, [28](#)
- fc2SetStrobe
 - Strobe, [38](#)
- fc2SetStrobeBroadcast
 - Strobe, [38](#)

- fc2SetTriggerDelay
 - Trigger, [34](#)
- fc2SetTriggerDelayBroadcast
 - Trigger, [34](#)
- fc2SetTriggerMode
 - Trigger, [35](#)
- fc2SetTriggerModeBroadcast
 - Trigger, [35](#)
- fc2SetUserBuffers
 - Connection and Image Retrieval, [21](#)
- fc2SetVideoModeAndFrameRate
 - DCAM Formats, [52](#)
- fc2Show
 - FlyCapture2GUI_C.h, [191](#)
- fc2ShowModal
 - FlyCapture2GUI_C.h, [192](#)
- fc2StartCapture
 - Connection and Image Retrieval, [22](#)
- fc2StartCaptureCallback
 - Connection and Image Retrieval, [22](#)
- fc2StartSyncCapture
 - Connection and Image Retrieval, [23](#)
- fc2StartSyncCaptureCallback
 - Connection and Image Retrieval, [23](#)
- fc2StatisticsChannel
 - FlyCapture2Defs_C.h, [189](#)
- fc2StopCapture
 - Connection and Image Retrieval, [24](#)
- fc2StrobeControl, [162](#)
 - delay, [163](#)
 - duration, [163](#)
 - onOff, [163](#)
 - polarity, [163](#)
 - reserved, [163](#)
 - source, [163](#)
- fc2StrobeInfo, [163](#)
 - maxValue, [164](#)
 - minValue, [164](#)
 - onOffSupported, [164](#)
 - polaritySupported, [164](#)
 - present, [164](#)
 - readOutSupported, [164](#)
 - reserved, [165](#)
 - source, [165](#)
- fc2SystemInfo, [165](#)
 - byteOrder, [166](#)
 - cpuDescription, [166](#)
 - driverList, [166](#)
 - gpuDescription, [166](#)
 - libraryList, [166](#)
 - numCpuCores, [166](#)
 - osDescription, [166](#)
 - osType, [166](#)
 - reserved, [166](#)
 - screenHeight, [166](#)
 - screenWidth, [167](#)
 - sysMemSize, [167](#)
- fc2TIFFCompressionMethod
 - Image saving structures., [120](#)
- fc2TIFFOption, [167](#)
 - compression, [167](#)
 - reserved, [167](#)
- fc2TimeStamp, [168](#)
 - cycleCount, [168](#)
 - cycleOffset, [168](#)
 - cycleSeconds, [168](#)
 - microSeconds, [168](#)
 - reserved, [168](#)
 - seconds, [168](#)
- fc2TopologyNodeAddChild
 - TopologyNode Operation, [88](#)
- fc2TopologyNodeAddPortType
 - TopologyNode Operation, [89](#)
- fc2TopologyNodeAssignGuidToNode
 - TopologyNode Operation, [89](#)
- fc2TopologyNodeAssignGuidToNodeEx
 - TopologyNode Operation, [89](#)
- fc2TopologyNodeContext
 - TypeDefs, [99](#)
- fc2TopologyNodeGetChild
 - TopologyNode Operation, [90](#)
- fc2TopologyNodeGetDeviceld
 - TopologyNode Operation, [90](#)
- fc2TopologyNodeGetGuid
 - TopologyNode Operation, [90](#)
- fc2TopologyNodeGetInterfaceType
 - TopologyNode Operation, [90](#)
- fc2TopologyNodeGetNodeType
 - TopologyNode Operation, [91](#)
- fc2TopologyNodeGetNumChildren
 - TopologyNode Operation, [91](#)
- fc2TopologyNodeGetNumPorts
 - TopologyNode Operation, [91](#)
- fc2TopologyNodeGetPortType
 - TopologyNode Operation, [92](#)
- fc2TriggerDelay, [169](#)
 - absControl, [170](#)
 - absValue, [170](#)
 - autoManualMode, [170](#)
 - onOff, [170](#)
 - onePush, [170](#)
 - present, [170](#)
 - reserved, [170](#)
 - type, [170](#)
 - valueA, [170](#)
 - valueB, [170](#)
- fc2TriggerDelayInfo, [171](#)
 - absMax, [172](#)
 - absMin, [172](#)
 - absValSupported, [172](#)
 - autoSupported, [172](#)
 - manualSupported, [172](#)
 - max, [172](#)
 - min, [172](#)
 - onOffSupported, [172](#)
 - onePushSupported, [172](#)

- pUnitAbbr, [173](#)
 - pUnits, [173](#)
 - present, [173](#)
 - readOutSupported, [173](#)
 - reserved, [173](#)
 - type, [173](#)
- fc2TriggerMode, [173](#)
 - mode, [174](#)
 - onOff, [174](#)
 - parameter, [174](#)
 - polarity, [174](#)
 - reserved, [174](#)
 - source, [174](#)
- fc2TriggerModelInfo, [175](#)
 - modeMask, [175](#)
 - onOffSupported, [175](#)
 - polaritySupported, [176](#)
 - present, [176](#)
 - readOutSupported, [176](#)
 - reserved, [176](#)
 - softwareTriggerSupported, [176](#)
 - sourceMask, [176](#)
 - valueReadable, [176](#)
- fc2UnregisterCallback
 - Bus Manager Operation, [17](#)
- fc2ValidateFormat7Settings
 - Format7, [54](#)
- fc2Version, [176](#)
 - build, [177](#)
 - major, [177](#)
 - minor, [177](#)
 - type, [177](#)
- fc2VideoMode
 - Enumerations, [112](#)
- fc2WaitForBufferEvent
 - Connection and Image Retrieval, [24](#)
- fc2WriteGVCPMemory
 - GVCP Register Operation, [57](#)
- fc2WriteGVCPRegister
 - GVCP Register Operation, [58](#)
- fc2WriteGVCPRegisterBlock
 - GVCP Register Operation, [58](#)
- fc2WriteGVCPRegisterBroadcast
 - GVCP Register Operation, [58](#)
- fc2WritePhyRegister
 - Bus Manager Operation, [17](#)
- fc2WriteRegister
 - Register Operation, [49](#)
- fc2WriteRegisterBlock
 - Register Operation, [50](#)
- fc2WriteRegisterBroadcast
 - Register Operation, [50](#)
- firmwareBuildTime
 - fc2CameraInfo, [125](#)
- firmwareVersion
 - fc2CameraInfo, [125](#)
- FlyCapture2_C.h, [179](#)
- FlyCapture2Defs_C.h, [179](#)
- BUS, [188](#)
- CAMERA, [188](#)
- COMPUTER, [188](#)
- CONNECTED_TO_CHILD, [189](#)
- CONNECTED_TO_PARENT, [189](#)
- FC2_BYTE_ORDER_BIG_ENDIAN, [188](#)
- FC2_BYTE_ORDER_FORCE_32BITS, [188](#)
- FC2_BYTE_ORDER_LITTLE_ENDIAN, [188](#)
- FC2_LINUX_X64, [188](#)
- FC2_LINUX_X86, [188](#)
- FC2_MAC, [188](#)
- FC2_OSTYPE_FORCE_32BITS, [188](#)
- FC2_STATISTICS_BLUE, [189](#)
- FC2_STATISTICS_FORCE_32BITS, [189](#)
- FC2_STATISTICS_GREEN, [189](#)
- FC2_STATISTICS_GREY, [189](#)
- FC2_STATISTICS_HUE, [189](#)
- FC2_STATISTICS_LIGHTNESS, [189](#)
- FC2_STATISTICS_RED, [189](#)
- FC2_STATISTICS_SATURATION, [189](#)
- FC2_UNKNOWN_OS, [188](#)
- FC2_WINDOWS_X64, [188](#)
- FC2_WINDOWS_X86, [188](#)
- fc2ByteOrder, [188](#)
- fc2NodeType, [188](#)
- fc2OSType, [188](#)
- fc2PortType, [188](#)
- fc2StatisticsChannel, [189](#)
- NODE, [188](#)
- NOT_CONNECTED, [189](#)
- FlyCapture2GUI_C.h, [189](#)
 - fc2CreateGUIContext, [190](#)
 - fc2DestroyGUIContext, [190](#)
 - fc2Disconnect, [190](#)
 - fc2GUIConnect, [190](#)
 - fc2GUIDisconnect, [191](#)
 - fc2Hide, [191](#)
 - fc2IsVisible, [191](#)
 - fc2Show, [191](#)
 - fc2ShowModal, [192](#)
- FlyCapture2Internal_C.h, [192](#)
 - IsContextValid, [193](#)
 - IsGuiContextValid, [193](#)
 - SyncCpplImageToStruct, [193](#)
- FlyCapture2Platform_C.h, [193](#)
 - FLYCAPTURE2_C_API, [193](#)
 - FLYCAPTURE2_C_CALL_CONVEN, [193](#)
- FlyCapture2Private_C.h, [193](#)
 - GetInternal, [193](#)
- format
 - fc2Image, [151](#)
- Format7, [53](#)
 - fc2GetFormat7Configuration, [53](#)
 - fc2GetFormat7Info, [53](#)
 - fc2SetFormat7Configuration, [54](#)
 - fc2SetFormat7ConfigurationPacket, [54](#)
 - fc2ValidateFormat7Settings, [54](#)
- frameCounter

- fc2EmbeddedImageInfo, 135
- frameRate
 - fc2AVIOption, 121
 - fc2H264Option, 150
 - fc2MJPGOption, 159
- GPIOPinState
 - fc2EmbeddedImageInfo, 135
- GVCP Register Operation, 56
 - fc2ReadGVCPMemory, 56
 - fc2ReadGVCPRegister, 57
 - fc2ReadGVCPRegisterBlock, 57
 - fc2WriteGVCPMemory, 57
 - fc2WriteGVCPRegister, 58
 - fc2WriteGVCPRegisterBlock, 58
 - fc2WriteGVCPRegisterBroadcast, 58
- gain
 - fc2EmbeddedImageInfo, 135
- General Purpose Input / Output, 29
 - fc2GetGPIOPinDirection, 29
 - fc2SetGPIOPinDirection, 29
 - fc2SetGPIOPinDirectionBroadcast, 30
- GetInternal
 - FlyCapture2Private_C.h, 193
- GigE image binning settings, 65
 - fc2GetGigEImageBinningSettings, 65
 - fc2SetGigEImageBinningSettings, 65
- GigE image settings, 62
 - fc2GetGigEImageSettings, 62
 - fc2GetGigEImageSettingsInfo, 62
 - fc2GetGigEImagingMode, 63
 - fc2QueryGigEImagingMode, 63
 - fc2SetGigEImageSettings, 63
 - fc2SetGigEImagingMode, 64
- GigE image stream configuration, 67
 - fc2GetGigEConfig, 67
 - fc2GetGigEStreamChannelInfo, 67
 - fc2GetNumStreamChannels, 68
 - fc2SetGigEConfig, 68
 - fc2SetGigEStreamChannelInfo, 68
- GigE property manipulation, 60
 - fc2DiscoverGigEPacketSize, 60
 - fc2GetGigEProperty, 60
 - fc2SetGigEProperty, 61
- GigE specific enumerations, 114
 - FC2_HEARTBEAT_TIMEOUT, 114
 - FC2_HEARTBEAT, 114
 - fc2GigEPropertyType, 114
 - PACKET_DELAY, 114
 - PACKET_SIZE, 114
- GigE specific structures, 117
- gigEMajorVersion
 - fc2CameraInfo, 125
- gigEMinorVersion
 - fc2CameraInfo, 125
- gpuDescription
 - fc2SystemInfo, 166
- grabMode
 - fc2Config, 130
- grabTimeout
 - fc2Config, 130
- height
 - fc2Format7ImageSettings, 139
 - fc2GigEImageSettings, 144
 - fc2H264Option, 150
- highPerformanceRetrieveBuffer
 - fc2Config, 131
- hostPort
 - fc2GigEStreamChannel, 149
- IIDC specific structures, 118
- iidcVer
 - fc2CameraInfo, 125
- Image Operation, 70
 - fc2CalculateImageStatistics, 71
 - fc2ConvertImage, 71
 - fc2ConvertImageTo, 71
 - fc2CreateImage, 72
 - fc2DestroyImage, 72
 - fc2DetermineBitsPerPixel, 72
 - fc2GetDefaultColorProcessing, 73
 - fc2GetDefaultOutputFormat, 73
 - fc2GetImageData, 73
 - fc2GetImageTimeStamp, 74
 - fc2SaveImage, 74
 - fc2SaveImageWithOptions, 74
 - fc2SetDefaultColorProcessing, 74
 - fc2SetDefaultOutputFormat, 75
 - fc2SetImageData, 75
 - fc2SetImageDimensions, 76
- Image saving structures., 119
 - FC2_TIFF_ADOBE_DEFLATE, 120
 - FC2_TIFF_CCITTFAX3, 120
 - FC2_TIFF_CCITTFAX4, 120
 - FC2_TIFF_DEFLATE, 120
 - FC2_TIFF_JPEG, 120
 - FC2_TIFF_LZW, 120
 - FC2_TIFF_NONE, 120
 - FC2_TIFF_PACKBITS, 120
 - fc2AsyncCommandCallback, 120
 - fc2BusEventCallback, 120
 - fc2CallbackHandle, 120
 - fc2CameraEventCallback, 120
 - fc2ImageEventCallback, 120
 - fc2TIFFCompressionMethod, 120
- Image Statistics Operation, 77
 - fc2CreateImageStatistics, 78
 - fc2DestroyImageStatistics, 78
 - fc2GetChannelHistogram, 78
 - fc2GetChannelMean, 79
 - fc2GetChannelNumPixelValues, 79
 - fc2GetChannelPixelValueRange, 79
 - fc2GetChannelRange, 80
 - fc2GetChannelStatus, 80
 - fc2GetImageStatistics, 80
 - fc2ImageStatisticsDisableAll, 81
 - fc2ImageStatisticsEnableAll, 81

- fc2ImageStatisticsEnableGreyOnly, 81
 - fc2ImageStatisticsEnableHSLOnly, 82
 - fc2ImageStatisticsEnableRGBOnly, 82
 - fc2SetChannelStatus, 82
- imageCorrupt
 - fc2CameraStats, 128
- imageDriverDropped
 - fc2CameraStats, 128
- imageDropped
 - fc2CameraStats, 128
- imageHStepSize
 - fc2Format7Info, 140
 - fc2GigEImageSettingsInfo, 146
- imageImpl
 - fc2Image, 151
- imageVStepSize
 - fc2Format7Info, 140
 - fc2GigEImageSettingsInfo, 146
- imageXmitFailed
 - fc2CameraStats, 128
- indexedColor_8bit
 - fc2BMPOption, 122
- Information and Properties, 26
 - fc2GetCameraInfo, 26
 - fc2GetProperty, 26
 - fc2GetPropertyInfo, 27
 - fc2SetProperty, 27
 - fc2SetPropertyBroadcast, 28
- inputBitDepth
 - fc2LUTData, 157
- interPacketDelay
 - fc2GigEStreamChannel, 149
- interfaceType
 - fc2CameraInfo, 125
- interlaced
 - fc2PNGOption, 161
- ipAddress
 - fc2CameraInfo, 125
- isColorCamera
 - fc2CameraInfo, 126
- IsContextValid
 - FlyCapture2Internal_C.h, 193
- IsGuiContextValid
 - FlyCapture2Internal_C.h, 193
- isReadable
 - fc2GigEProperty, 147
- isWritable
 - fc2GigEProperty, 147
- isochBusSpeed
 - fc2Config, 131
- libraryList
 - fc2SystemInfo, 166
- Look Up Table, 40
 - fc2EnableLUT, 40
 - fc2GetActiveLUTBank, 41
 - fc2GetLUTBankInfo, 41
 - fc2GetLUTChannel, 41
 - fc2GetLUTInfo, 42
 - fc2SetActiveLUTBank, 42
 - fc2SetLUTChannel, 42
- MAX_STRING_LENGTH
 - MultiSyncLibraryDefs_C.h, 198
 - TypeDefs, 98
- MULTISYNCLIBRARY_C_API
 - MultiSyncLibraryPlatform_C.h, 199
- MULTISYNCLIBRARY_C_CALL_CONVEN
 - MultiSyncLibraryPlatform_C.h, 199
- macAddress
 - fc2CameraInfo, 126
- major
 - fc2Version, 177
- manualSupported
 - fc2TriggerDelayInfo, 172
- max
 - fc2GigEProperty, 147
 - fc2TriggerDelayInfo, 172
- maxBytesPerPacket
 - fc2Format7PacketInfo, 142
- maxHeight
 - fc2Format7Info, 140
 - fc2GigEImageSettingsInfo, 146
- maxPacketSize
 - fc2Format7Info, 141
- maxValue
 - fc2StrobeInfo, 164
- maxWidth
 - fc2Format7Info, 141
 - fc2GigEImageSettingsInfo, 146
- maximumBusSpeed
 - fc2CameraInfo, 126
- Memory Channels, 44
 - fc2GetEmbeddedImageInfo, 44
 - fc2GetMemoryChannel, 45
 - fc2GetMemoryChannelInfo, 45
 - fc2RestoreFromMemoryChannel, 45
 - fc2SaveToMemoryChannel, 46
 - fc2SetEmbeddedImageInfo, 46
- microSeconds
 - fc2TimeStamp, 168
- min
 - fc2GigEProperty, 148
 - fc2TriggerDelayInfo, 172
- minNumImageNotifications
 - fc2Config, 131
- minPacketSize
 - fc2Format7Info, 141
- minValue
 - fc2StrobeInfo, 164
- minor
 - fc2Version, 177
- mode
 - fc2Format7ImageSettings, 139
 - fc2Format7Info, 141
 - fc2TriggerMode, 174
- modeMask
 - fc2TriggerModelInfo, 175

- modelName
 - fc2CameraInfo, [126](#)
- MultiSyncLibrary_C.h, [193](#)
 - syncCreateContext, [194](#)
 - syncDestroyContext, [194](#)
 - syncDisableCrossPCSynchronization, [195](#)
 - syncEnableCrossPCSynchronization, [195](#)
 - syncGetStatus, [195](#)
 - syncGetTimeSinceSynced, [195](#)
 - syncIsTimingBusConnected, [196](#)
 - syncQueryCrossPCSynchronizationSetting, [196](#)
 - syncRescanMasterTimingBus, [196](#)
 - syncStart, [197](#)
 - syncStop, [197](#)
- MultiSyncLibraryDefs_C.h, [197](#)
 - BOOL, [198](#)
 - FALSE, [198](#)
 - FULL_32BIT_VALUE, [198](#)
 - MAX_STRING_LENGTH, [198](#)
 - SYNC_ERROR_ALREADY_STARTED, [198](#)
 - SYNC_ERROR_ALREADY_STOPPED, [198](#)
 - SYNC_ERROR_CONTEXT_NOT_INITIALIZED, [198](#)
 - SYNC_ERROR_FAILED, [198](#)
 - SYNC_ERROR_OK, [198](#)
 - SYNC_ERROR_UNKNOWN_ERROR, [198](#)
 - SYNC_MESSAGE_BUS_RESET, [199](#)
 - SYNC_MESSAGE_DEVICE_ERROR, [199](#)
 - SYNC_MESSAGE_FAILED, [199](#)
 - SYNC_MESSAGE_NOMASTER, [199](#)
 - SYNC_MESSAGE_NOT_ENOUGH_DEVICES, [199](#)
 - SYNC_MESSAGE_NOT_INITIALIZED, [199](#)
 - SYNC_MESSAGE_OK, [199](#)
 - SYNC_MESSAGE_STARTED, [199](#)
 - SYNC_MESSAGE_STOPPED, [199](#)
 - SYNC_MESSAGE_SYNCING, [199](#)
 - SYNC_MESSAGE_THREAD_ERROR, [199](#)
 - SYNC_MESSAGE_UNKNOWN_ERROR, [199](#)
 - syncContext, [198](#)
 - syncError, [198](#)
 - syncMessage, [198](#)
 - TRUE, [198](#)
- MultiSyncLibraryPlatform_C.h, [199](#)
 - MULTISYNCLIBRARY_C_API, [199](#)
 - MULTISYNCLIBRARY_C_CALL_CONVEN, [199](#)
- NODE
 - FlyCapture2Defs_C.h, [188](#)
- NOT_CONNECTED
 - FlyCapture2Defs_C.h, [189](#)
- networkInterfaceIndex
 - fc2GigEStreamChannel, [149](#)
- nodeNumber
 - fc2CameraInfo, [126](#)
- nodeVendorId
 - fc2ConfigROM, [133](#)
- numBanks
 - fc2LUTData, [157](#)
- numBuffers
 - fc2Config, [131](#)
- numChannels
 - fc2LUTData, [158](#)
- numCpuCores
 - fc2SystemInfo, [166](#)
- numCurrents
 - fc2CameraStats, [128](#)
- numEntries
 - fc2LUTData, [158](#)
- numImageNotifications
 - fc2Config, [131](#)
- numResendPacketsReceived
 - fc2CameraStats, [128](#)
- numResendPacketsRequested
 - fc2CameraStats, [129](#)
- numVoltages
 - fc2CameraStats, [129](#)
- octets
 - fc2IPAddress, [155](#)
 - fc2MACAddress, [159](#)
- offsetHStepSize
 - fc2Format7Info, [141](#)
 - fc2GigEImageSettingsInfo, [146](#)
- offsetVStepSize
 - fc2Format7Info, [141](#)
 - fc2GigEImageSettingsInfo, [146](#)
- offsetX
 - fc2Format7ImageSettings, [139](#)
 - fc2GigEImageSettings, [144](#)
- offsetY
 - fc2Format7ImageSettings, [139](#)
 - fc2GigEImageSettings, [144](#)
- onOff
 - fc2EmbeddedImageInfoProperty, [136](#)
 - fc2StrobeControl, [163](#)
 - fc2TriggerDelay, [170](#)
 - fc2TriggerMode, [174](#)
- onOffSupported
 - fc2StrobeInfo, [164](#)
 - fc2TriggerDelayInfo, [172](#)
 - fc2TriggerModeInfo, [175](#)
- onePush
 - fc2TriggerDelay, [170](#)
- onePushSupported
 - fc2TriggerDelayInfo, [172](#)
- osDescription
 - fc2SystemInfo, [166](#)
- osType
 - fc2SystemInfo, [166](#)
- outputBitDepth
 - fc2LUTData, [158](#)
- PACKET_DELAY
 - GigE specific enumerations, [114](#)
- PACKET_SIZE
 - GigE specific enumerations, [114](#)
- pBusMgr

- fc2InternalContext, 154
- pCallback
 - fc2InternalImageCallback, 154
- pCallbackData
 - fc2InternalImageCallback, 154
- pCamera
 - fc2InternalContext, 154
- pCameraControlDlg
 - fc2InternalGuiContext, 154
- pCameraSelectionDlg
 - fc2InternalGuiContext, 154
- pData
 - fc2Image, 151
- pUnitAbbr
 - fc2TriggerDelayInfo, 173
- pUnits
 - fc2TriggerDelayInfo, 173
- packetSize
 - fc2Format7Info, 141
 - fc2GigEStreamChannel, 149
- parameter
 - fc2TriggerMode, 174
- pcieBusSpeed
 - fc2CameraInfo, 126
- percentage
 - fc2Format7Info, 141
- pixelFormat
 - fc2Format7ImageSettings, 139
 - fc2GigEImageSettings, 145
- pixelFormatBitField
 - fc2Format7Info, 141
 - fc2GigEImageSettingsInfo, 146
- polarity
 - fc2StrobeControl, 163
 - fc2TriggerMode, 174
- polaritySupported
 - fc2StrobeInfo, 164
 - fc2TriggerModelInfo, 176
- portErrors
 - fc2CameraStats, 129
- present
 - fc2StrobeInfo, 164
 - fc2TriggerDelay, 170
 - fc2TriggerDelayInfo, 173
 - fc2TriggerModelInfo, 176
- progressive
 - fc2JPEGOption, 155
- propType
 - fc2GigEProperty, 148
- pszKeyword
 - fc2ConfigROM, 133
- quality
 - fc2JPEGOption, 155
 - fc2JPG2Option, 156
 - fc2MJPGOption, 159
- ROIPosition
 - fc2EmbeddedImageInfo, 135
- readOutSupported
 - fc2StrobeInfo, 164
 - fc2TriggerDelayInfo, 173
 - fc2TriggerModelInfo, 176
- receivedDataSize
 - fc2Image, 151
- recommendedBytesPerPacket
 - fc2Format7PacketInfo, 142
- regReadFailed
 - fc2CameraStats, 129
- regWriteFailed
 - fc2CameraStats, 129
- Register Operation, 48
 - fc2GetRegisterString, 48
 - fc2ReadRegister, 48
 - fc2ReadRegisterBlock, 49
 - fc2WriteRegister, 49
 - fc2WriteRegisterBlock, 50
 - fc2WriteRegisterBroadcast, 50
- registerTimeout
 - fc2Config, 131
 - fc2GigEConfig, 143
- registerTimeoutRetries
 - fc2Config, 132
 - fc2GigEConfig, 143
- reserved
 - fc2AVIOption, 121
 - fc2BMPOption, 122
 - fc2CameraInfo, 126
 - fc2CameraStats, 129
 - fc2Config, 132
 - fc2ConfigROM, 133
 - fc2Format7ImageSettings, 139
 - fc2Format7Info, 141
 - fc2Format7PacketInfo, 142
 - fc2GigEConfig, 144
 - fc2GigEImageSettings, 145
 - fc2GigEImageSettingsInfo, 146
 - fc2GigEProperty, 148
 - fc2GigEStreamChannel, 149
 - fc2H264Option, 150
 - fc2ImageMetadata, 153
 - fc2JPEGOption, 156
 - fc2JPG2Option, 156
 - fc2LUTData, 158
 - fc2MJPGOption, 159
 - fc2PGMOption, 160
 - fc2PNGOption, 161
 - fc2PPMOption, 162
 - fc2StrobeControl, 163
 - fc2StrobeInfo, 165
 - fc2SystemInfo, 166
 - fc2TIFFOption, 167
 - fc2TimeStamp, 168
 - fc2TriggerDelay, 170
 - fc2TriggerDelayInfo, 173
 - fc2TriggerMode, 174
 - fc2TriggerModelInfo, 176

- rows
 - fc2Image, [151](#)
- SYNC_ERROR_ALREADY_STARTED
 - MultiSyncLibraryDefs_C.h, [198](#)
- SYNC_ERROR_ALREADY_STOPPED
 - MultiSyncLibraryDefs_C.h, [198](#)
- SYNC_ERROR_CONTEXT_NOT_INITIALIZED
 - MultiSyncLibraryDefs_C.h, [198](#)
- SYNC_ERROR_FAILED
 - MultiSyncLibraryDefs_C.h, [198](#)
- SYNC_ERROR_OK
 - MultiSyncLibraryDefs_C.h, [198](#)
- SYNC_ERROR_UNKNOWN_ERROR
 - MultiSyncLibraryDefs_C.h, [198](#)
- SYNC_MESSAGE_BUS_RESET
 - MultiSyncLibraryDefs_C.h, [199](#)
- SYNC_MESSAGE_DEVICE_ERROR
 - MultiSyncLibraryDefs_C.h, [199](#)
- SYNC_MESSAGE_FAILED
 - MultiSyncLibraryDefs_C.h, [199](#)
- SYNC_MESSAGE_NOMASTER
 - MultiSyncLibraryDefs_C.h, [199](#)
- SYNC_MESSAGE_NOT_ENOUGH_DEVICES
 - MultiSyncLibraryDefs_C.h, [199](#)
- SYNC_MESSAGE_NOT_INITIALIZED
 - MultiSyncLibraryDefs_C.h, [199](#)
- SYNC_MESSAGE_OK
 - MultiSyncLibraryDefs_C.h, [199](#)
- SYNC_MESSAGE_STARTED
 - MultiSyncLibraryDefs_C.h, [199](#)
- SYNC_MESSAGE_STOPPED
 - MultiSyncLibraryDefs_C.h, [199](#)
- SYNC_MESSAGE_SYNCING
 - MultiSyncLibraryDefs_C.h, [199](#)
- SYNC_MESSAGE_THREAD_ERROR
 - MultiSyncLibraryDefs_C.h, [199](#)
- SYNC_MESSAGE_UNKNOWN_ERROR
 - MultiSyncLibraryDefs_C.h, [199](#)
- screenHeight
 - fc2SystemInfo, [166](#)
- screenWidth
 - fc2SystemInfo, [167](#)
- seconds
 - fc2TimeStamp, [168](#)
- sensorInfo
 - fc2CameraInfo, [126](#)
- sensorResolution
 - fc2CameraInfo, [126](#)
- serialNumber
 - fc2CameraInfo, [126](#)
- shutter
 - fc2EmbeddedImageInfo, [135](#)
- softwareTriggerSupported
 - fc2TriggerModelInfo, [176](#)
- source
 - fc2StrobeControl, [163](#)
 - fc2StrobeInfo, [165](#)
 - fc2TriggerMode, [174](#)
- sourceMask
 - fc2TriggerModelInfo, [176](#)
- sourcePort
 - fc2GigEStreamChannel, [149](#)
- stride
 - fc2Image, [151](#)
- Strobe, [37](#)
 - fc2GetStrobe, [37](#)
 - fc2GetStrobeInfo, [37](#)
 - fc2SetStrobe, [38](#)
 - fc2SetStrobeBroadcast, [38](#)
- strobePattern
 - fc2EmbeddedImageInfo, [135](#)
- Structures, [115](#)
- subnetMask
 - fc2CameraInfo, [127](#)
- supported
 - fc2LUTData, [158](#)
- syncContext
 - MultiSyncLibraryDefs_C.h, [198](#)
- SyncCplImageToStruct
 - FlyCapture2Internal_C.h, [193](#)
- syncCreateContext
 - MultiSyncLibrary_C.h, [194](#)
- syncDestroyContext
 - MultiSyncLibrary_C.h, [194](#)
- syncDisableCrossPCsSynchronization
 - MultiSyncLibrary_C.h, [195](#)
- syncEnableCrossPCsSynchronization
 - MultiSyncLibrary_C.h, [195](#)
- syncError
 - MultiSyncLibraryDefs_C.h, [198](#)
- syncGetStatus
 - MultiSyncLibrary_C.h, [195](#)
- syncGetTimeSinceSynced
 - MultiSyncLibrary_C.h, [195](#)
- syncIsTimingBusConnected
 - MultiSyncLibrary_C.h, [196](#)
- syncMessage
 - MultiSyncLibraryDefs_C.h, [198](#)
- syncQueryCrossPCsSynchronizationSetting
 - MultiSyncLibrary_C.h, [196](#)
- syncRescanMasterTimingBus
 - MultiSyncLibrary_C.h, [196](#)
- syncStart
 - MultiSyncLibrary_C.h, [197](#)
- syncStop
 - MultiSyncLibrary_C.h, [197](#)
- sysMemSize
 - fc2SystemInfo, [167](#)
- TRUE
 - MultiSyncLibraryDefs_C.h, [198](#)
 - TypeDefs, [98](#)
- temperature
 - fc2CameraStats, [129](#)
- timeSinceBusReset
 - fc2CameraStats, [129](#)
- timeSinceInitialization

- fc2CameraStats, 129
- timeStamp
 - fc2CameraStats, 129
- timestamp
 - fc2EmbeddedImageInfo, 135
- TopologyNode Operation, 87
 - fc2CreateTopologyNode, 88
 - fc2DestroyTopologyNode, 88
 - fc2TopologyNodeAddChild, 88
 - fc2TopologyNodeAddPortType, 89
 - fc2TopologyNodeAssignGuidToNode, 89
 - fc2TopologyNodeAssignGuidToNodeEx, 89
 - fc2TopologyNodeGetChild, 90
 - fc2TopologyNodeGetDevicId, 90
 - fc2TopologyNodeGetGuid, 90
 - fc2TopologyNodeGetInterfaceType, 90
 - fc2TopologyNodeGetNodeType, 91
 - fc2TopologyNodeGetNumChildren, 91
 - fc2TopologyNodeGetNumPorts, 91
 - fc2TopologyNodeGetPortType, 92
- Trigger, 31
 - fc2FireSoftwareTrigger, 31
 - fc2FireSoftwareTriggerBroadcast, 32
 - fc2GetTriggerDelay, 32
 - fc2GetTriggerDelayInfo, 32
 - fc2GetTriggerMode, 33
 - fc2GetTriggerModelInfo, 33
 - fc2SetTriggerDelay, 34
 - fc2SetTriggerDelayBroadcast, 34
 - fc2SetTriggerMode, 35
 - fc2SetTriggerModeBroadcast, 35
- type
 - fc2TriggerDelay, 170
 - fc2TriggerDelayInfo, 173
 - fc2Version, 177
- TypeDefs, 98
 - BOOL, 98
 - FALSE, 98
 - FULL_32BIT_VALUE, 98
 - fc2AVIContext, 98
 - fc2Context, 98
 - fc2GuiContext, 99
 - fc2ImageImpl, 99
 - fc2ImageStatisticsContext, 99
 - fc2TopologyNodeContext, 99
 - MAX_STRING_LENGTH, 98
 - TRUE, 98
- unitBytesPerPacket
 - fc2Format7PacketInfo, 143
- unitSWVer
 - fc2ConfigROM, 133
- unitSpecId
 - fc2ConfigROM, 133
- unitSubSWVer
 - fc2ConfigROM, 133
- userDefinedName
 - fc2CameraInfo, 127
- Utilities, 93
 - fc2CheckDriver, 93
 - fc2ErrorToDescription, 94
 - fc2GetDriverDeviceName, 94
 - fc2GetLibraryVersion, 94
 - fc2GetSystemInfo, 94
 - fc2LaunchBrowser, 96
 - fc2LaunchCommand, 96
 - fc2LaunchCommandAsync, 96
 - fc2LaunchHelp, 97
- value
 - fc2GigEProperty, 148
 - fc2PGRGuid, 161
- valueReadable
 - fc2TriggerModelInfo, 176
- valueA
 - fc2TriggerDelay, 170
- valueB
 - fc2TriggerDelay, 170
- vendorName
 - fc2CameraInfo, 127
- vendorPixelFormatBitField
 - fc2Format7Info, 142
 - fc2GigEImageSettingsInfo, 146
- vendorUniqueInfo_0
 - fc2ConfigROM, 133
- vendorUniqueInfo_1
 - fc2ConfigROM, 133
- vendorUniqueInfo_2
 - fc2ConfigROM, 134
- vendorUniqueInfo_3
 - fc2ConfigROM, 134
- whiteBalance
 - fc2EmbeddedImageInfo, 135
- width
 - fc2Format7ImageSettings, 139
 - fc2GigEImageSettings, 145
 - fc2H264Option, 150
- xmlURL1
 - fc2CameraInfo, 127
- xmlURL2
 - fc2CameraInfo, 127