
# FlyCapture 2.5

## C Language API Programming Reference

Revised April 13, 2013

**Point Grey Research Inc.**
12051 Riverside Way • Richmond, BC • Canada • V6W 1K7 • **T** (604) 242-9937 •
**www.ptgrey.com**

## Software Warranty

Point Grey Research warrants to the Original Purchaser, for a period of one (1) year from date of purchase that:

1. The diskette on which the Software is furnished and the accompanying documentation are not defective;
2. The Software is properly recorded upon the diskettes enclosed;
3. The documentation is substantially complete and contains all the information Point Grey Research deems necessary to use the Software;
4. The Software functions substantially as described in the documentation.

Point Grey Research, Inc.'s entire liability and the Original Purchaser's exclusive remedy shall be the replacement of any diskette or documentation not meeting these warranties. On such an occasion, a copy of the paid receipt accompanied with the faulty diskette or documentation must be returned to Point Grey Research, Inc. or an authorized dealer.

Point Grey Research, Inc. expressly disclaims and excludes all other warranties, express, implied and statutory, including, but without limitation, warranty of merchantability and fitness for a particular application or purpose. In no event shall Point Grey Research, Inc. be liable to the Original Purchaser or any third party for direct, indirect, incidental, consequential, special or accidental damages, including without limitation damages for business interruption, loss of profits, revenue, data or bodily injury or death.

## Software License Agreement

**READ CAREFULLY: This is a legal agreement between you (an individual or a single entity) ("you") and Point Grey Research, Inc. ("PGR"). Before installing and using the FlyCapture® Software Development Kit and any updates to it that we may at our discretion provide to you (collectively, the "SDK"), you should read this agreement. If you do not agree with all of the terms of this agreement, do not install or use the SDK. PGR may change this agreement at any time and it is your responsibility to review the most updated version of it on PGR's website at** http://www.ptgrey.com/support/kb/data/PGR-FlyCap-SDK-LA.pdf**. By continuing to use the SDK following such changes, you agree to be bound by them.**

1. **Grant of License:** Subject to the terms of this agreement, you are hereby granted a limited, terminable, non-transferable, non-exclusive license and right to use the SDK only in conjunction with: (a) those PGR cameras listed at http://www.ptgrey.com/products/index.asp (as such list may be amended by PGR at any time and from time to time) and owned by you; and (b) the images derived from such cameras.

2. **Free and Open Source Components:** Notwithstanding anything to the contrary herein, use, copying and distribution of components of the SDK licensed under free and open source license agreements are governed solely by the terms of those license agreements (which are contained in the electronic documentation for the SDK) and not this agreement.

3. **Restrictions:** Except as (and only if) explicitly permitted by Section 4 below, you will not, and will not permit any third parties to: (a) copy the SDK, other than a reasonable number of backup copies for your own use only, and such backup copies together with the original will be kept in your possession and control; (b) provide or disclose the SDK to any third party; (c) alter, modify, reverse engineer, decompile or disassemble the SDK, or attempt to do any of the foregoing; (d) grant sublicenses, leases, or any other rights in the SDK to any third party; or (e) remove, alter or obscure any proprietary rights notices (including any copyright and trademark notices) on and in the SDK.

4. **Additional OEM Rights:** If you are an original equipment manufacturer, then in addition to the rights set out in Section 1 above you are hereby granted a limited, terminable, non-transferable, non-exclusive license and right to use the SDK for the sole additional purpose of incorporating the libraries found in the SDK (collectively, the "**Libraries**") into new products developed by you, in whole or in part, using the SDK (collectively, the "**Derivative Products**") provided that you: (a) ensure that the components of any Derivative Product that derive functionality from any of the Libraries may only be used with PGR products, including the SDK, and images derived from such products; (b) may only redistribute drivers (.inf and .sys), dynamically linked libraries (.dlls and .so), executables (.exe) and documentation (.doc, .txt, .pdf and .chm) and only to the extent necessary to support your Derivative Products. For clarity, headers (.h), source (.c, .cpp,.cs and .vb) and statically linked libraries (.lib and .a) cannot be redistributed; (c) will prohibit any, and ensure that there is no, redistribution of any of the Libraries by any third party, including any end user customers; and (d) will include any PGR and third party proprietary rights legends or notices (including copyright and trademark notices), unaltered and unobscured, on all Derivative Products.

5. **Ownership:** PGR and third parties are the owners of and retain title to all proprietary and intellectual property rights (including all patent, copyright, trade secret and trademark

rights) in and to the SDK. You have no right, title or interest in the SDK, except as specifically set forth herein, and no rights in any trade-marks of PGR. All rights not explicitly granted herein are hereby reserved.

6. **Indemnification:** You assume the entire risk relating to, and will indemnify, hold harmless and defend PGR from and against any claims, actions, lawsuits, or proceedings, and any losses, liabilities, damages and expenses (including attorney's fees and expenses) that arise or result from your activities under this agreement, including the distribution or use of the SDK (including the Libraries) and/or the development, distribution or use of any Derivative Product (including any intellectual property infringement claims relating thereto).

7. **No Warranties:** Your use of the SDK is solely at your own risk. The SDK is provided "as is" and "as available" without warranty or condition of any kind, either express, implied or statutory, including implied warranties of merchantability, fitness for a particular application or purpose, title and non-infringement, and PGR hereby expressly disclaims all such warranties and conditions. Some jurisdictions do not allow the exclusion of implied warranties, so the above exclusion may not apply to you.

8. **Limitation of Liability:** Notwithstanding any other provision herein, to the maximum extent permitted by applicable law, PGR will not under any circumstances be liable for any direct, indirect, incidental, consequential, punitive or special losses or damages (including damages for bodily injury or death, business interruption, loss or corruption of business information or data, loss of opportunity, loss of privacy, cost of recovery, financial loss, loss of revenue or loss of profits for any reason whatsoever) whether or not PGR has been advised or should have been aware of the possibility of any such losses or damage arising, and in no event will the aggregate and cumulative liability of PGR for any matters arising under this agreement or otherwise exceed $25.

9. **Changes and Support:** PGR may at its sole discretion elect to provide you with limited support services related to the SDK for such period of time as PGR at its sole discretion elects. PGR may change or cease to provide the SDK and the limited Support at any time and from time to time without notice to you and that PGR is not obligated to provide the SDK or any support. PGR may use any technical information, feedback or ideas you provide to PGR for PGR's business purposes, including product support and development.

10. **Termination:** PGR may terminate this agreement immediately on notice to you if you violate any of the terms of this agreement. PGR may also terminate this agreement for convenience provided that 30 days notice is delivered to you. Any notice given by PGR hereunder will be effective when sent by PGR to the email address you provided to PGR when you registered for a download account. On termination, for any reason, your rights hereunder will cease and you must immediately return all copies of the SDK to PGR and on termination for violation or default, you will be liable to PGR for all damages suffered as a result of the violation or default. Sections 6-8, 10 and 12 will survive any termination hereof.

11. **Export Laws:** This agreement involves products and/or technical data that may be controlled under laws and regulations of the United States and other countries, including the United States Export Administration Regulations, or any other applicable law, regulation, rule, guideline or order (collectively, "**Export Laws**"). You will comply with all Export Laws to ensure that the SDK is not exported, directly or indirectly, in contravention of the Export Laws.

You represent and warrant to PGR that you are not a person barred from receiving the SDK under any Export Laws.

12. **General:** This agreement is the entire agreement between you and PGR with respect to the subject matter of this agreement. If you are signing on behalf of an entity such as a corporation, you represent and warrant that you have the authority to bind such entity. This agreement and the rights granted hereunder are personal to you and you may not assign this agreement to a third party without the prior written consent of PGR. This agreement is governed exclusively by and will be enforced, construed, and interpreted exclusively in accordance with the laws of British Columbia ("BC") and the laws of Canada applicable in BC. The courts of the Province of BC will have exclusive jurisdiction over any dispute arising under this agreement. You agree that termination and/or monetary damages may not be a sufficient remedy if you breach this agreement and that PGR will be entitled, without waiving any other rights or remedies, to injunctive or equitable relief as may be deemed proper by a court of competent jurisdiction in the event of a breach. If PGR does not exercise any legal right or remedy in this agreement or otherwise, this will not be taken to be a formal waiver by PGR of its rights, which rights will remain available to PGR. If any provision of this agreement is construed to be illegal or invalid, the illegal or invalid provisions will be deemed stricken and deleted herefrom to the same extent and effect as if never incorporated herein, but all other provisions hereof will continue in full force and effect.

# Contents

# Chapter 1

# Data Structure Index

## 1.1 Data Structures

Here are the data structures with brief descriptions:

# Chapter 2

# File Index

## 2.1 File List

Here is a list of all files with brief descriptions:

# Chapter 3

# Data Structure Documentation

## 3.1 fc2AVIOption Struct Reference

### Data Fields

- float frameRate
- unsigned int reserved [256]

### 3.1.1 Field Documentation

#### 3.1.1.1 float frameRate

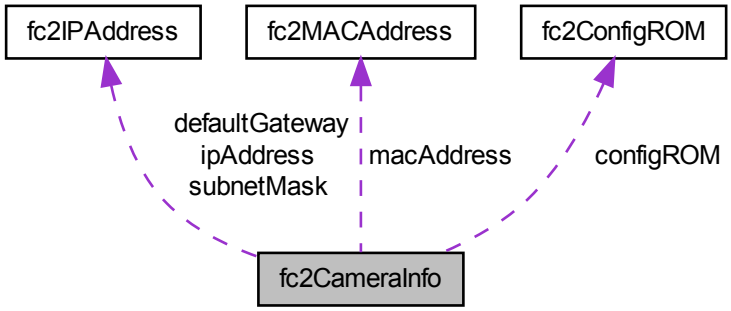
#### 3.1.1.2 unsigned int reserved[256]

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

- FlyCapture2Defs_C.h

## 3.2 fc2CameraInfo Struct Reference

Collaboration diagram for fc2CameraInfo:




### Data Fields

- unsigned int serialNumber
- fc2InterfaceType interfaceType
- fc2DriverType driverType
- BOOL isColorCamera
- char modelName [MAX_STRING_LENGTH]
- char vendorName [MAX_STRING_LENGTH]
- char sensorInfo [MAX_STRING_LENGTH]
- char sensorResolution [MAX_STRING_LENGTH]
- char driverName [MAX_STRING_LENGTH]
- char firmwareVersion [MAX_STRING_LENGTH]
- char firmwareBuildTime [MAX_STRING_LENGTH]
- fc2BusSpeed maximumBusSpeed
- fc2PCIeBusSpeed pcieBusSpeed
- fc2BayerTileFormat bayerTileFormat
- unsigned short busNumber
- unsigned short nodeNumber
- unsigned int iidcVer
- fc2ConfigROM configROM
- unsigned int gigEMajorVersion
- unsigned int gigEMinorVersion
- char userDefinedName [MAX_STRING_LENGTH]
- char xmlURL1 [MAX_STRING_LENGTH]
- char xmlURL2 [MAX_STRING_LENGTH]

- fc2MACAddress macAddress
- fc2IPAddress ipAddress
- fc2IPAddress subnetMask
- fc2IPAddress defaultGateway
- unsigned int ccpStatus

  *Status/Content of CCP register.*
- unsigned int applicationIPAddress

  *Local Application IP Address.*
- unsigned int applicationPort

  *Local Application port.*
- unsigned int reserved [16]

### 3.2.1 Field Documentation

#### 3.2.1.1 unsigned int applicationIPAddress

Local Application IP Address.

#### 3.2.1.2 unsigned int applicationPort

Local Application port.

#### 3.2.1.3 fc2BayerTileFormat bayerTileFormat

#### 3.2.1.4 unsigned short busNumber

#### 3.2.1.5 unsigned int ccpStatus

Status/Content of CCP register.

#### 3.2.1.6 fc2ConfigROM configROM

#### 3.2.1.7 fc2IPAddress defaultGateway

#### 3.2.1.8 char driverName[MAX_STRING_LENGTH]

#### 3.2.1.9 fc2DriverType driverType

#### 3.2.1.10 char firmwareBuildTime[MAX_STRING_LENGTH]

#### 3.2.1.11 char firmwareVersion[MAX_STRING_LENGTH]

#### 3.2.1.12 unsigned int gigEMajorVersion

#### 3.2.1.13 unsigned int gigEMinorVersion

#### 3.2.1.14 unsigned int iidcVer

#### 3.2.1.15 fc2InterfaceType interfaceType

#### 3.2.1.16 fc2IPAddress ipAddress

#### 3.2.1.17 BOOL isColorCamera

#### 3.2.1.18 fc2MACAddress macAddress

#### 3.2.1.19 fc2BusSpeed maximumBusSpeed

#### 3.2.1.20 char modelName[MAX_STRING_LENGTH]

#### 3.2.1.21 unsigned short nodeNumber

#### 3.2.1.22 fc2PCIeBusSpeed pcieBusSpeed

#### 3.2.1.23 unsigned int reserved[16]

#### 3.2.1.24 char sensorInfo[MAX_STRING_LENGTH]

#### 3.2.1.25 char sensorResolution[MAX_STRING_LENGTH]

#### 3.2.1.26 unsigned int serialNumber

#### 3.2.1.27 fc2IPAddress subnetMask

#### 3.2.1.28 char userDefinedName[MAX_STRING_LENGTH]

#### 3.2.1.29 char vendorName[MAX_STRING_LENGTH]

#### 3.2.1.30 char xmlURL1[MAX_STRING_LENGTH]

#### 3.2.1.31 char xmlURL2[MAX_STRING_LENGTH]

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

- FlyCapture2Defs_C.h

## 3.3 fc2Config Struct Reference

### Data Fields

- unsigned int numBuffers

- unsigned int numImageNotifications
- unsigned int minNumImageNotifications
- int grabTimeout
- fc2GrabMode grabMode
- fc2BusSpeed isochBusSpeed
- fc2BusSpeed asyncBusSpeed
- fc2BandwidthAllocation bandwidthAllocation
- unsigned int registerTimeoutRetries
- unsigned int registerTimeout
- unsigned int reserved [16]

### 3.3.1 Field Documentation

#### 3.3.1.1 fc2BusSpeed asyncBusSpeed

#### 3.3.1.2 fc2BandwidthAllocation bandwidthAllocation

#### 3.3.1.3 fc2GrabMode grabMode

#### 3.3.1.4 int grabTimeout

#### 3.3.1.5 fc2BusSpeed isochBusSpeed

#### 3.3.1.6 unsigned int minNumImageNotifications

#### 3.3.1.7 unsigned int numBuffers

#### 3.3.1.8 unsigned int numImageNotifications

#### 3.3.1.9 unsigned int registerTimeout

#### 3.3.1.10 unsigned int registerTimeoutRetries

#### 3.3.1.11 unsigned int reserved[16]

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

- FlyCapture2Defs_C.h

## 3.4 fc2ConfigROM Struct Reference

### Data Fields

- unsigned int nodeVendorId
- unsigned int chipIdHi

- unsigned int chipIdLo
- unsigned int unitSpecId
- unsigned int unitSWVer
- unsigned int unitSubSWVer
- unsigned int vendorUniqueInfo_0
- unsigned int vendorUniqueInfo_1
- unsigned int vendorUniqueInfo_2
- unsigned int vendorUniqueInfo_3
- char pszKeyword [MAX_STRING_LENGTH]
- unsigned int reserved [16]

### 3.4.1 Field Documentation

#### 3.4.1.1 unsigned int chipIdHi

#### 3.4.1.2 unsigned int chipIdLo

#### 3.4.1.3 unsigned int nodeVendorId

#### 3.4.1.4 char pszKeyword[MAX_STRING_LENGTH]

#### 3.4.1.5 unsigned int reserved[16]

#### 3.4.1.6 unsigned int unitSpecId

#### 3.4.1.7 unsigned int unitSubSWVer

#### 3.4.1.8 unsigned int unitSWVer

#### 3.4.1.9 unsigned int vendorUniqueInfo_0

#### 3.4.1.10 unsigned int vendorUniqueInfo_1

#### 3.4.1.11 unsigned int vendorUniqueInfo_2

#### 3.4.1.12 unsigned int vendorUniqueInfo_3

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

- FlyCapture2Defs_C.h

## 3.5 fc2EmbeddedImageInfo Struct Reference

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

### 3.5.1 Field Documentation

#### 3.5.1.1 fc2EmbeddedImageInfoProperty brightness

#### 3.5.1.2 fc2EmbeddedImageInfoProperty exposure

#### 3.5.1.3 fc2EmbeddedImageInfoProperty frameCounter

#### 3.5.1.4 fc2EmbeddedImageInfoProperty gain

#### 3.5.1.5 fc2EmbeddedImageInfoProperty GPIOPinState

#### 3.5.1.6 fc2EmbeddedImageInfoProperty ROIPosition

#### 3.5.1.7 fc2EmbeddedImageInfoProperty shutter

#### 3.5.1.8 fc2EmbeddedImageInfoProperty strobePattern

#### 3.5.1.9 fc2EmbeddedImageInfoProperty timestamp

#### 3.5.1.10 fc2EmbeddedImageInfoProperty whiteBalance

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

- FlyCapture2Defs_C.h

## 3.6 fc2EmbeddedImageInfoProperty Struct Reference

### Data Fields

- BOOL available
- BOOL onOff

### 3.6.1 Field Documentation

#### 3.6.1.1 BOOL available

#### 3.6.1.2 BOOL onOff

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

- FlyCapture2Defs_C.h

## 3.7 fc2Format7ImageSettings Struct Reference

### Data Fields

- fc2Mode mode

- unsigned int offsetX
- unsigned int offsetY
- unsigned int width
- unsigned int height
- fc2PixelFormat pixelFormat
- unsigned int reserved [8]

### 3.7.1 Field Documentation

#### 3.7.1.1 unsigned int height

#### 3.7.1.2 fc2Mode mode

#### 3.7.1.3 unsigned int offsetX

#### 3.7.1.4 unsigned int offsetY

#### 3.7.1.5 fc2PixelFormat pixelFormat

#### 3.7.1.6 unsigned int reserved[8]

#### 3.7.1.7 unsigned int width

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

- FlyCapture2Defs_C.h

## 3.8 fc2Format7Info Struct Reference

### Data Fields

- fc2Mode mode
- unsigned int maxWidth
- unsigned int maxHeight
- unsigned int offsetHStepSize
- unsigned int offsetVStepSize
- unsigned int imageHStepSize
- unsigned int imageVStepSize
- unsigned int pixelFormatBitField
- unsigned int vendorPixelFormatBitField
- unsigned int packetSize
- unsigned int minPacketSize
- unsigned int maxPacketSize
- float percentage
- unsigned int reserved [16]

### 3.8.1 Field Documentation

#### 3.8.1.1 unsigned int imageHStepSize

#### 3.8.1.2 unsigned int imageVStepSize

#### 3.8.1.3 unsigned int maxHeight

#### 3.8.1.4 unsigned int maxPacketSize

#### 3.8.1.5 unsigned int maxWidth

#### 3.8.1.6 unsigned int minPacketSize

#### 3.8.1.7 fc2Mode mode

#### 3.8.1.8 unsigned int offsetHStepSize

#### 3.8.1.9 unsigned int offsetVStepSize

#### 3.8.1.10 unsigned int packetSize

#### 3.8.1.11 float percentage

#### 3.8.1.12 unsigned int pixelFormatBitField

#### 3.8.1.13 unsigned int reserved[16]

#### 3.8.1.14 unsigned int vendorPixelFormatBitField

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

- FlyCapture2Defs_C.h

## 3.9 fc2Format7PacketInfo Struct Reference

### Data Fields

- unsigned int recommendedBytesPerPacket
- unsigned int maxBytesPerPacket
- unsigned int unitBytesPerPacket
- unsigned int reserved [8]

### 3.9.1 Field Documentation

#### 3.9.1.1 unsigned int maxBytesPerPacket

#### 3.9.1.2 unsigned int recommendedBytesPerPacket

#### 3.9.1.3 unsigned int reserved[8]

#### 3.9.1.4 unsigned int unitBytesPerPacket

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

- FlyCapture2Defs_C.h

## 3.10 fc2GigEConfig Struct Reference

### Data Fields

- BOOL enablePacketResend

  *Turn on/off packet resend functionality.*
- unsigned int timeoutForPacketResend

  *The number of miliseconds to wait for each requested packet.*
- unsigned int maxPacketsToResend

  *The max number of packets that can be requested to be resend.*
- unsigned int reserved [8]

### 3.10.1 Field Documentation

#### 3.10.1.1 BOOL enablePacketResend

Turn on/off packet resend functionality.

#### 3.10.1.2 unsigned int maxPacketsToResend

The max number of packets that can be requested to be resend.

#### 3.10.1.3 unsigned int reserved[8]

#### 3.10.1.4 unsigned int timeoutForPacketResend

The number of miliseconds to wait for each requested packet.

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

- FlyCapture2Defs_C.h

## 3.11 fc2GigEImageSettings Struct Reference

### Data Fields

- unsigned int offsetX
- unsigned int offsetY
- unsigned int width
- unsigned int height
- fc2PixelFormat pixelFormat
- unsigned int reserved [8]

### 3.11.1 Field Documentation

#### 3.11.1.1 unsigned int height

#### 3.11.1.2 unsigned int offsetX

#### 3.11.1.3 unsigned int offsetY

#### 3.11.1.4 fc2PixelFormat pixelFormat

#### 3.11.1.5 unsigned int reserved[8]

#### 3.11.1.6 unsigned int width

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

- FlyCapture2Defs_C.h

## 3.12 fc2GigEImageSettingsInfo Struct Reference

### Data Fields

- unsigned int maxWidth
- unsigned int maxHeight
- unsigned int offsetHStepSize
- unsigned int offsetVStepSize
- unsigned int imageHStepSize
- unsigned int imageVStepSize
- unsigned int pixelFormatBitField
- unsigned int vendorPixelFormatBitField
- unsigned int reserved [16]

### 3.12.1 Field Documentation

#### 3.12.1.1 unsigned int imageHStepSize

#### 3.12.1.2 unsigned int imageVStepSize

#### 3.12.1.3 unsigned int maxHeight

#### 3.12.1.4 unsigned int maxWidth

#### 3.12.1.5 unsigned int offsetHStepSize

#### 3.12.1.6 unsigned int offsetVStepSize

#### 3.12.1.7 unsigned int pixelFormatBitField

#### 3.12.1.8 unsigned int reserved[16]

#### 3.12.1.9 unsigned int vendorPixelFormatBitField

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

- FlyCapture2Defs_C.h

## 3.13 fc2GigEProperty Struct Reference

### Data Fields

- fc2GigEPropertyType propType
- BOOL isReadable
- BOOL isWritable
- unsigned int min
- unsigned int max
- unsigned int value
- unsigned int reserved [8]

### 3.13.1 Field Documentation

#### 3.13.1.1 BOOL isReadable

#### 3.13.1.2 BOOL isWritable

#### 3.13.1.3 unsigned int max

#### 3.13.1.4 unsigned int min

#### 3.13.1.5 fc2GigEPropertyType propType

#### 3.13.1.6 unsigned int reserved[8]

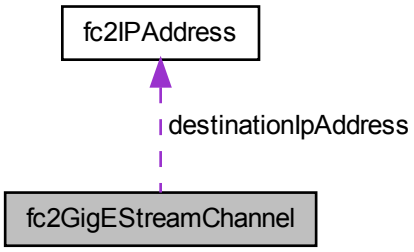
#### 3.13.1.7 unsigned int value

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

- FlyCapture2Defs_C.h

## 3.14 fc2GigEStreamChannel Struct Reference

Collaboration diagram for fc2GigEStreamChannel:




### Data Fields

- unsigned int networkInterfaceIndex
- unsigned int hostPost
- BOOL doNotFragment
- unsigned int packetSize
- unsigned int interPacketDelay
- fc2IPAddress destinationIpAddress
- unsigned int sourcePort
- unsigned int reserved [8]

### 3.14.1 Field Documentation

#### 3.14.1.1 fc2IPAddress destinationIpAddress

#### 3.14.1.2 BOOL doNotFragment

#### 3.14.1.3 unsigned int hostPost

#### 3.14.1.4 unsigned int interPacketDelay

#### 3.14.1.5 unsigned int networkInterfaceIndex

#### 3.14.1.6 unsigned int packetSize

#### 3.14.1.7 unsigned int reserved[8]

#### 3.14.1.8 unsigned int sourcePort

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

- FlyCapture2Defs_C.h

## 3.15 fc2H264Option Struct Reference

### Data Fields

- float frameRate
- unsigned int width
- unsigned int height
- unsigned int bitrate
- unsigned int reserved [256]

### 3.15.1 Field Documentation

#### 3.15.1.1 unsigned int bitrate

#### 3.15.1.2 float frameRate

#### 3.15.1.3 unsigned int height

#### 3.15.1.4 unsigned int reserved[256]

#### 3.15.1.5 unsigned int width

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

- FlyCapture2Defs_C.h

## 3.16 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

### 3.16.1 Field Documentation

#### 3.16.1.1 fc2BayerTileFormat bayerFormat

#### 3.16.1.2 unsigned int cols

#### 3.16.1.3 unsigned int dataSize

#### 3.16.1.4 fc2PixelFormat format

#### 3.16.1.5 fc2ImageImpl imageImpl

#### 3.16.1.6 unsigned char* pData

#### 3.16.1.7 unsigned int receivedDataSize

#### 3.16.1.8 unsigned int rows

#### 3.16.1.9 unsigned int stride

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

- FlyCapture2Defs_C.h

## 3.17 fc2ImageMetadata Struct Reference

### Data Fields

- unsigned int embeddedTimeStamp
- unsigned int embeddedGain
- unsigned int embeddedShutter

- unsigned int embeddedBrightness
- unsigned int embeddedExposure
- unsigned int embeddedWhiteBalance
- unsigned int embeddedFrameCounter
- unsigned int embeddedStrobePattern
- unsigned int embeddedGPIOPinState
- unsigned int embeddedROIPosition
- unsigned int reserved [31]

### 3.17.1 Field Documentation

#### 3.17.1.1 unsigned int embeddedBrightness

#### 3.17.1.2 unsigned int embeddedExposure

#### 3.17.1.3 unsigned int embeddedFrameCounter

#### 3.17.1.4 unsigned int embeddedGain

#### 3.17.1.5 unsigned int embeddedGPIOPinState

#### 3.17.1.6 unsigned int embeddedROIPosition

#### 3.17.1.7 unsigned int embeddedShutter

#### 3.17.1.8 unsigned int embeddedStrobePattern

#### 3.17.1.9 unsigned int embeddedTimeStamp

#### 3.17.1.10 unsigned int embeddedWhiteBalance

#### 3.17.1.11 unsigned int reserved[31]

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

- FlyCapture2Defs_C.h

## 3.18 fc2InternalContext Struct Reference

### Data Fields

- FlyCapture2::BusManager * pBusMgr
- FlyCapture2::CameraBase * pCamera

### 3.18.1 Field Documentation

#### 3.18.1.1 FlyCapture2::BusManager* pBusMgr

#### 3.18.1.2 FlyCapture2::CameraBase* pCamera

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

- FlyCapture2Internal_C.h

## 3.19 fc2InternalGuiContext Struct Reference

### Data Fields

- FlyCapture2::CameraSelectionDlg * pCameraSelectionDlg
- FlyCapture2::CameraControlDlg * pCameraControlDlg

### 3.19.1 Field Documentation

#### 3.19.1.1 FlyCapture2::CameraControlDlg* pCameraControlDlg

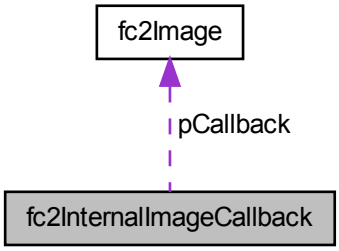
#### 3.19.1.2 FlyCapture2::CameraSelectionDlg* pCameraSelectionDlg

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

- FlyCapture2Internal_C.h

## 3.20 fc2InternalImageCallback Struct Reference

Collaboration diagram for fc2InternalImageCallback:




### Data Fields

- fc2ImageEventCallback pCallback
- void * pCallbackData

### 3.20.1 Field Documentation

#### 3.20.1.1 fc2ImageEventCallback pCallback

#### 3.20.1.2 void* pCallbackData

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

- FlyCapture2Internal_C.h

## 3.21 fc2IPAddress Struct Reference

### Data Fields

- unsigned char octets [4]

### 3.21.1 Field Documentation

#### 3.21.1.1 unsigned char octets[4]

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

- FlyCapture2Defs_C.h

## 3.22 fc2JPEGOption Struct Reference

### Data Fields

- BOOL progressive
- unsigned int quality
- unsigned int reserved [16]

### 3.22.1 Field Documentation

#### 3.22.1.1 BOOL progressive

#### 3.22.1.2 unsigned int quality

#### 3.22.1.3 unsigned int reserved[16]

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

- FlyCapture2Defs_C.h

## 3.23 fc2JPG2Option Struct Reference

### Data Fields

- unsigned int quality
- unsigned int reserved [16]

### 3.23.1 Field Documentation

#### 3.23.1.1 unsigned int quality

#### 3.23.1.2 unsigned int reserved[16]

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

- FlyCapture2Defs_C.h

## 3.24 fc2LUTData Struct Reference

### Data Fields

- BOOL supported
- BOOL enabled
- unsigned int numBanks
- unsigned int numChannels
- unsigned int inputBitDepth
- unsigned int outputBitDepth
- unsigned int numEntries
- unsigned int reserved [8]

### 3.24.1 Field Documentation

#### 3.24.1.1 BOOL enabled

#### 3.24.1.2 unsigned int inputBitDepth

#### 3.24.1.3 unsigned int numBanks

#### 3.24.1.4 unsigned int numChannels

#### 3.24.1.5 unsigned int numEntries

#### 3.24.1.6 unsigned int outputBitDepth

#### 3.24.1.7 unsigned int reserved[8]

#### 3.24.1.8 BOOL supported

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

- FlyCapture2Defs_C.h

## 3.25 fc2MACAddress Struct Reference

### Data Fields

- unsigned char octets [6]

### 3.25.1 Field Documentation

#### 3.25.1.1 unsigned char octets[6]

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

- FlyCapture2Defs_C.h

## 3.26 fc2MJPGOption Struct Reference

### Data Fields

- float frameRate
- unsigned int quality
- unsigned int reserved [256]

### 3.26.1 Field Documentation

#### 3.26.1.1 float frameRate

#### 3.26.1.2 unsigned int quality

#### 3.26.1.3 unsigned int reserved[256]

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

- FlyCapture2Defs_C.h

## 3.27 fc2PGMOption Struct Reference

### Data Fields

- BOOL binaryFile
- unsigned int reserved [16]

### 3.27.1 Field Documentation

#### 3.27.1.1 BOOL binaryFile

#### 3.27.1.2 unsigned int reserved[16]

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

- FlyCapture2Defs_C.h

## 3.28 fc2PGRGuid Struct Reference

A GUID to the camera.

### Data Fields

- unsigned int value [4]

### 3.28.1 Detailed Description

A GUID to the camera.

It is used to uniquely identify a camera.

### 3.28.2 Field Documentation

#### 3.28.2.1 unsigned int value[4]

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

- FlyCapture2Defs_C.h

## 3.29 fc2PNGOption Struct Reference

### Data Fields

- BOOL interlaced
- unsigned int compressionLevel
- unsigned int reserved [16]

### 3.29.1 Field Documentation

#### 3.29.1.1 unsigned int compressionLevel

#### 3.29.1.2 BOOL interlaced

#### 3.29.1.3 unsigned int reserved[16]

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

- FlyCapture2Defs_C.h

## 3.30 fc2PPMOption Struct Reference

### Data Fields

- BOOL binaryFile
- unsigned int reserved [16]

### 3.30.1 Field Documentation

#### 3.30.1.1 BOOL binaryFile

#### 3.30.1.2 unsigned int reserved[16]

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

- FlyCapture2Defs_C.h

## 3.31 fc2StrobeControl Struct Reference

### Data Fields

- unsigned int source
- BOOL onOff
- unsigned int polarity
- float delay
- float duration
- unsigned int reserved [8]

### 3.31.1 Field Documentation

#### 3.31.1.1 float delay

#### 3.31.1.2 float duration

#### 3.31.1.3 BOOL onOff

#### 3.31.1.4 unsigned int polarity

#### 3.31.1.5 unsigned int reserved[8]

#### 3.31.1.6 unsigned int source

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

- FlyCapture2Defs_C.h

## 3.32 fc2StrobeInfo Struct Reference

### Data Fields

- unsigned int source
- BOOL present
- BOOL readOutSupported
- BOOL onOffSupported
- BOOL polaritySupported
- float minValue
- float maxValue
- unsigned int reserved [8]

### 3.32.1 Field Documentation

#### 3.32.1.1 float maxValue

#### 3.32.1.2 float minValue

#### 3.32.1.3 BOOL onOffSupported

#### 3.32.1.4 BOOL polaritySupported

#### 3.32.1.5 BOOL present

#### 3.32.1.6 BOOL readOutSupported

#### 3.32.1.7 unsigned int reserved[8]

#### 3.32.1.8 unsigned int source

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

- FlyCapture2Defs_C.h

## 3.33 fc2SystemInfo Struct Reference

### Data Fields

- fc2OSType osType
- char osDescription [MAX_STRING_LENGTH]
- fc2ByteOrder byteOrder
- size_t sysMemSize
- char cpuDescription [MAX_STRING_LENGTH]
- size_t numCpuCores

- char driverList [MAX_STRING_LENGTH]
- char libraryList [MAX_STRING_LENGTH]
- char gpuDescription [MAX_STRING_LENGTH]
- size_t screenWidth
- size_t screenHeight
- unsigned int reserved [16]

### 3.33.1 Field Documentation

#### 3.33.1.1 fc2ByteOrder byteOrder

#### 3.33.1.2 char cpuDescription[MAX_STRING_LENGTH]

#### 3.33.1.3 char driverList[MAX_STRING_LENGTH]

#### 3.33.1.4 char gpuDescription[MAX_STRING_LENGTH]

#### 3.33.1.5 char libraryList[MAX_STRING_LENGTH]

#### 3.33.1.6 size_t numCpuCores

#### 3.33.1.7 char osDescription[MAX_STRING_LENGTH]

#### 3.33.1.8 fc2OSType osType

#### 3.33.1.9 unsigned int reserved[16]

#### 3.33.1.10 size_t screenHeight

#### 3.33.1.11 size_t screenWidth

#### 3.33.1.12 size_t sysMemSize

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

- FlyCapture2Defs_C.h

## 3.34 fc2TIFFOption Struct Reference

### Data Fields

- fc2TIFFCompressionMethod compression
- unsigned int reserved [16]

### 3.34.1 Field Documentation

#### 3.34.1.1 fc2TIFFCompressionMethod compression

#### 3.34.1.2 unsigned int reserved[16]

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

- FlyCapture2Defs_C.h

## 3.35 fc2TimeStamp Struct Reference

### Data Fields

- long long seconds
- unsigned int microSeconds
- unsigned int cycleSeconds
- unsigned int cycleCount
- unsigned int cycleOffset
- unsigned int reserved [8]

### 3.35.1 Field Documentation

#### 3.35.1.1 unsigned int cycleCount

#### 3.35.1.2 unsigned int cycleOffset

#### 3.35.1.3 unsigned int cycleSeconds

#### 3.35.1.4 unsigned int microSeconds

#### 3.35.1.5 unsigned int reserved[8]

#### 3.35.1.6 long long seconds

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

- FlyCapture2Defs_C.h

## 3.36 fc2TriggerDelay Struct Reference

### Data Fields

- fc2PropertyType type

- BOOL present
- BOOL absControl
- BOOL onePush
- BOOL onOff
- BOOL autoManualMode
- unsigned int valueA
- unsigned int valueB
- float absValue
- unsigned int reserved [8]

### 3.36.1 Field Documentation

#### 3.36.1.1 BOOL absControl

#### 3.36.1.2 float absValue

#### 3.36.1.3 BOOL autoManualMode

#### 3.36.1.4 BOOL onePush

#### 3.36.1.5 BOOL onOff

#### 3.36.1.6 BOOL present

#### 3.36.1.7 unsigned int reserved[8]

#### 3.36.1.8 fc2PropertyType type

#### 3.36.1.9 unsigned int valueA

#### 3.36.1.10 unsigned int valueB

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

- FlyCapture2Defs_C.h

## 3.37 fc2TriggerDelayInfo Struct Reference

### Data Fields

- fc2PropertyType type
- BOOL present
- BOOL autoSupported
- BOOL manualSupported
- BOOL onOffSupported

- BOOL onePushSupported
- BOOL absValSupported
- BOOL readOutSupported
- unsigned int min
- unsigned int max
- float absMin
- float absMax
- char pUnits [MAX_STRING_LENGTH]
- char pUnitAbbr [MAX_STRING_LENGTH]
- unsigned int reserved [8]

## 3.37.1 Field Documentation

### 3.37.1.1 float absMax

### 3.37.1.2 float absMin

### 3.37.1.3 BOOL absValSupported

### 3.37.1.4 BOOL autoSupported

### 3.37.1.5 BOOL manualSupported

### 3.37.1.6 unsigned int max

### 3.37.1.7 unsigned int min

### 3.37.1.8 BOOL onePushSupported

### 3.37.1.9 BOOL onOffSupported

### 3.37.1.10 BOOL present

### 3.37.1.11 char pUnitAbbr[MAX_STRING_LENGTH]

### 3.37.1.12 char pUnits[MAX_STRING_LENGTH]

### 3.37.1.13 BOOL readOutSupported

### 3.37.1.14 unsigned int reserved[8]

### 3.37.1.15 fc2PropertyType type

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

- FlyCapture2Defs_C.h

## 3.38 fc2TriggerMode Struct Reference

### Data Fields

- BOOL onOff
- unsigned int polarity
- unsigned int source
- unsigned int mode
- unsigned int parameter
- unsigned int reserved [8]

### 3.38.1 Field Documentation

#### 3.38.1.1 unsigned int mode

#### 3.38.1.2 BOOL onOff

#### 3.38.1.3 unsigned int parameter

#### 3.38.1.4 unsigned int polarity

#### 3.38.1.5 unsigned int reserved[8]

#### 3.38.1.6 unsigned int source

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

- FlyCapture2Defs_C.h

## 3.39 fc2TriggerModeInfo Struct Reference

### Data Fields

- BOOL present
- BOOL readOutSupported
- BOOL onOffSupported
- BOOL polaritySupported
- BOOL valueReadable
- unsigned int sourceMask
- BOOL softwareTriggerSupported
- unsigned int modeMask
- unsigned int reserved [8]

### 3.39.1 Field Documentation

#### 3.39.1.1 unsigned int modeMask

#### 3.39.1.2 BOOL onOffSupported

#### 3.39.1.3 BOOL polaritySupported

#### 3.39.1.4 BOOL present

#### 3.39.1.5 BOOL readOutSupported

#### 3.39.1.6 unsigned int reserved[8]

#### 3.39.1.7 BOOL softwareTriggerSupported

#### 3.39.1.8 unsigned int sourceMask

#### 3.39.1.9 BOOL valueReadable

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

- FlyCapture2Defs_C.h

## 3.40 fc2Version Struct Reference

### Data Fields

- unsigned int major
- unsigned int minor
- unsigned int type
- unsigned int build

### 3.40.1 Field Documentation

#### 3.40.1.1 unsigned int build

#### 3.40.1.2 unsigned int major

#### 3.40.1.3 unsigned int minor

#### 3.40.1.4 unsigned int type

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

- FlyCapture2Defs_C.h

# Chapter 4

# File Documentation

## 4.1 FlyCapture2_C.h File Reference

### Functions

- FLYCAPTURE2_C_API fc2Error fc2CreateContext (fc2Context *pContext)

  *Create a FC2 context for IIDC camaera.*
- FLYCAPTURE2_C_API fc2Error fc2CreateGigEContext (fc2Context *pContext)

  *Create a FC2 context for a GigE Vision camera.*
- FLYCAPTURE2_C_API fc2Error fc2DestroyContext (fc2Context context)

  *Destroy the FC2 context.*
- 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 fc2IsCameraControlable (fc2Context context, fc2PGRGuid *pGuid, BOOL *pControlable)

  *Query whether a GigE camera is controlable.*
- 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 serialNumber, 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 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 netowrk adapters that they are connected to.*
- FLYCAPTURE2_C_API fc2Error fc2ForceIPAddressAutomatically (unsigned int serialNumber)

  *Force a cameras on the network to be assigned sequential IP addresses on the same subnet as the netowrk 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 fc2WriteRegister (fc2Context context, unsigned int address, unsigned int value)

  *Write to the specified register on 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 fc2ReadRegister (fc2Context context, unsigned int address, unsigned int *pValue)

  *Read the specified register from the camera.*
- 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)

  *Write to the specified register block on the camera.*
- 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 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 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.*
- 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.*
- 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.*
- FLYCAPTURE2_C_API fc2Error fc2GetTriggerModeInfo (fc2Context context, fc2TriggerModeInfo *triggerModeInfo)

  *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)
- 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.*
- 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.*
- 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.*
- 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.*

- 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.*
- 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 fc2QueryGigEImagingMode (fc2Context context, fc2Mode mode, BOOL *isSupported)
- FLYCAPTURE2_C_API fc2Error fc2GetGigEImagingMode (fc2Context context, fc2Mode *mode)
- FLYCAPTURE2_C_API fc2Error fc2SetGigEImagingMode (fc2Context context, fc2Mode mode)
- FLYCAPTURE2_C_API fc2Error fc2GetGigEImageSettingsInfo (fc2Context context, fc2GigEImageSettingsInfo *pInfo)
- FLYCAPTURE2_C_API fc2Error fc2GetGigEImageSettings (fc2Context context, fc2GigEImageSettings *pImageSettings)
- FLYCAPTURE2_C_API fc2Error fc2SetGigEImageSettings (fc2Context context, const fc2GigEImageSettings *pImageSettings)
- FLYCAPTURE2_C_API fc2Error fc2GetGigEConfig (fc2Context context, fc2GigEConfig *pConfig)
- FLYCAPTURE2_C_API fc2Error fc2SetGigEConfig (fc2Context context, const fc2GigEConfig *pConfig)
- FLYCAPTURE2_C_API fc2Error fc2GetGigEImageBinningSettings (fc2Context context, unsigned int *horzBinnningValue, unsigned int *vertBinnningValue)

- FLYCAPTURE2_C_API fc2Error fc2SetGigEImageBinningSettings (fc2Context context, unsigned int horzBinnningValue, unsigned int vertBinnningValue)
- FLYCAPTURE2_C_API fc2Error fc2GetNumStreamChannels (fc2Context context, unsigned int *numChannels)
- FLYCAPTURE2_C_API fc2Error fc2GetGigEStreamChannelInfo (fc2Context context, unsigned int channel, fc2GigEStreamChannel *pChannel)
- FLYCAPTURE2_C_API fc2Error fc2SetGigEStreamChannelInfo (fc2Context context, unsigned int channel, fc2GigEStreamChannel *pChannel)
- 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.*
- FLYCAPTURE2_C_API fc2Error fc2GetMemoryChannel (fc2Context context, unsigned int *pCurrentChannel)

  *Retrieve the current memory channel from the camera.*
- FLYCAPTURE2_C_API fc2Error fc2SaveToMemoryChannel (fc2Context context, unsigned int channel)

  *Save the current settings to the specfied current memory channel.*
- FLYCAPTURE2_C_API fc2Error fc2RestoreFromMemoryChannel (fc2Context context, unsigned int channel)

  *Restore the specfied current memory channel.*
- FLYCAPTURE2_C_API fc2Error fc2GetMemoryChannelInfo (fc2Context context, unsigned int *pNumChannels)

  *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.*
- FLYCAPTURE2_C_API const char * fc2GetRegisterString (unsigned int registerVal)

  *Returns a text representation of the register value.*
- FLYCAPTURE2_C_API fc2Error fc2CreateImage (fc2Image *pImage)

  *Create a fc2Image.*
- FLYCAPTURE2_C_API fc2Error fc2DestroyImage (fc2Image *image)

  *Destroy the fc2Image.*
- FLYCAPTURE2_C_API fc2Error fc2SetDefaultColorProcessing (fc2ColorProcessingAlgorithm defaultMethod)

  *Set the default color processing algorithm.*
- FLYCAPTURE2_C_API fc2Error fc2GetDefaultColorProcessing (fc2ColorProcessingAlgorithm *pDefaultMethod)

  *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 *pBitsPerPixel)

  *Calculate the bits per pixel for the specified pixel format.*
- 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.*
- FLYCAPTURE2_C_API fc2Error fc2SaveImageWithOption (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 fc2GetImageData (fc2Image *pImage, unsigned char **ppData)

  *Get a pointer to the data associated with the image.*
- FLYCAPTURE2_C_API fc2Error fc2SetImageData (fc2Image *pImage, const unsigned char *pData, unsigned int dataSize)

  *Set the data of the Image object.*
- 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 fc2TimeStamp fc2GetImageTimeStamp (fc2Image *pImage)

  *Get the timestamp data associated with the image.*
- FLYCAPTURE2_C_API fc2Error fc2CalculateImageStatistics (fc2Image *pImage, fc2ImageStatisticsContext *pImageStatisticsContext)

  *Calculate statistics associated with the image.*
- 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 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.*
- FLYCAPTURE2_C_API fc2Error fc2CreateAVI (fc2AVIContext *pAVIContext)

  *Create a AVI context.*
- FLYCAPTURE2_C_API fc2Error fc2AVIOpen (fc2AVIContext AVIContext, const char *pFileName, fc2AVIOption *pOption)

  *Open an AVI file in preparation for writing Images to disk.*
- FLYCAPTURE2_C_API fc2Error fc2MJPGOpen (fc2AVIContext AVIContext, const char *pFileName, fc2MJPGOption *pOption)

  *Open an MJPEG file in preparation for writing Images to disk.*
- FLYCAPTURE2_C_API fc2Error fc2H264Open (fc2AVIContext AVIContext, const char *pFileName, fc2H264Option *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.*
- 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.*

- FLYCAPTURE2_C_API fc2Error fc2GetCycleTime (fc2Context context, fc2TimeStamp *pTimeStamp)

  *Get cycle time from camera.*

### 4.1.1 Function Documentation

#### 4.1.1.1 FLYCAPTURE2_C_API fc2Error fc2AVIAppend ( fc2AVIContext *AVIContext*, fc2Image * *pImage* )

Append an image to the AVI file.

**Parameters**

| | |
|---|---|
| *AVIContext* | The AVI context to use. |
| *pImage* | The image to append. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.2 FLYCAPTURE2_C_API fc2Error fc2AVIClose ( fc2AVIContext *AVIContext* )

Close the AVI file.

**Parameters**

| | |
|---|---|
| *AVIContext* | The AVI context to use. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.3 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**

| | |
|---|---|
| *AVIContext* | The AVI context to use. |
| *pFileName* | The filename of the AVI file. |
| *pOption* | Options to apply to the AVI file. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.4 FLYCAPTURE2_C_API fc2Error fc2CalculateImageStatistics ( fc2Image * *pImage*, fc2ImageStatisticsContext * *pImageStatisticsContext* )

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**

| | |
|---|---|
| *pImage* | The fc2Image to be used. |
| *pImage-Statistics-Context* | The fc2ImageStatisticsContext to hold the statistics. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.5 FLYCAPTURE2_C_API fc2Error fc2Connect ( fc2Context *context*, fc2PGRGuid * *guid* )

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

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *guid* | The unique identifier for a specific camera on the PC. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.6 FLYCAPTURE2_C_API fc2Error fc2ConvertImage ( fc2Image * *pImageIn*, fc2Image * *pImageOut* )

**Parameters**

| | |
|---|---|
| *pImageIn* | |
| *pImageOut* | |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.7 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.

The destination image does not need to be configured in any way before the call is made.

**Parameters**

| | |
|---|---|
| *format* | Output format of the converted image. |
| *pImageIn* | Input image. |
| *pImageOut* | Output image. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.8 FLYCAPTURE2_C_API fc2Error fc2CreateAVI ( fc2AVIContext * *pAVIContext* )

Create a AVI context.

**Parameters**

| | |
|---|---|
| *pAVIContext* | A AVI context. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.9 FLYCAPTURE2_C_API fc2Error fc2CreateContext ( fc2Context * *pContext* )

Create a FC2 context for IIDC camaera.

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

**Parameters**

| | |
|---|---|
| *pContext* | A pointer to the fc2Context to be created. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.10 FLYCAPTURE2_C_API fc2Error fc2CreateGigEContext ( fc2Context * *pContext* )

Create a FC2 context for a GigE Vision camera.

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

**Parameters**

| | |
|---|---|
| *pContext* | A pointer to the fc2Context to be created. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.11 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.

**Parameters**

| | |
|---|---|
| *pImage* | Pointer to image to be created. |

**See also**

fc2SetImageData()

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.12 FLYCAPTURE2_C_API fc2Error fc2CreateImageStatistics ( fc2ImageStatisticsContext * *pImageStatisticsContext* )

Create a statistics context.

**Parameters**

| | |
|---|---|
| *pImage-Statistics-Context* | A statistics context. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.13 FLYCAPTURE2_C_API fc2Error fc2DestroyAVI ( fc2AVIContext *AVIContext* )

Destroy a AVI context.

**Parameters**

| | |
|---|---|
| *AVIContext* | A AVI context. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.14 FLYCAPTURE2_C_API fc2Error fc2DestroyContext ( fc2Context *context* )

Destroy the FC2 context.

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

**Parameters**

| | |
|---|---|
| *context* | The context to be destroyed. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.15 FLYCAPTURE2_C_API fc2Error fc2DestroyImage ( fc2Image * *image* )

Destroy the fc2Image.

**Parameters**

| | |
|---|---|
| *image* | Pointer to image to be destroyed. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.16 FLYCAPTURE2_C_API fc2Error fc2DestroyImageStatistics ( fc2ImageStatisticsContext *imageStatisticsContext* )

Destroy a statistics context.

**Parameters**

| | |
|---|---|
| *imageStatisticsContext* | A statistics context. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.17 FLYCAPTURE2_C_API fc2Error fc2DetermineBitsPerPixel ( fc2PixelFormat *format*, unsigned int * *pBitsPerPixel* )

Calculate the bits per pixel for the specified pixel format.

**Parameters**

| | |
|---|---|
| *format* | The pixel format. |
| *pBitsPerPixel* | The bits per pixel. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.18 FLYCAPTURE2_C_API fc2Error fc2Disconnect ( fc2Context *context* )

Disconnects the fc2Context from the camera.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.19 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**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *gigE-Cameras* | Pointer to an array of CameraInfo structures. |
| *arraySize* | 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.

#### 4.1.1.20 FLYCAPTURE2_C_API fc2Error fc2EnableLUT ( fc2Context *context,* BOOL *on* )

Enable or disable LUT functionality on the camera.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *on* | Whether to enable or disable LUT. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.21 FLYCAPTURE2_C_API const char* fc2ErrorToDescription ( fc2Error *error* )

Get a string representation of an error.

**Parameters**

| | |
|---|---|
| *error* | Error to be parsed. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.22 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**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *pGuid* | PGRGuid of the camera or the device to cause bus reset. |

**Returns**

An Error indicating the success or failure of the function.

#### 4.1.1.23 FLYCAPTURE2_C_API fc2Error fc2FireSoftwareTrigger ( fc2Context *context* )

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.24 FLYCAPTURE2_C_API fc2Error fc2FireSoftwareTriggerBroadcast ( fc2Context *context* )

Fire the software trigger according to the DCAM specifications.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.25 FLYCAPTURE2_C_API fc2Error fc2ForceAllIPAddressesAutomatically ( )

Force all cameras on the network to be assigned sequential IP addresses on the same subnet as the netowrk 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.

#### 4.1.1.26 FLYCAPTURE2_C_API fc2Error fc2ForceIPAddressAutomatically ( unsigned int *serialNumber* )

Force a cameras on the network to be assigned sequential IP addresses on the same subnet as the netowrk 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.

#### 4.1.1.27 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**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *macAddress* | MAC address of the camera. |
| *ipAddress* | IP address to set on the camera. |
| *subnetMask* | Subnet mask to set on the camera. |
| *default-Gateway* | Default gateway to set on the camera. |

**Returns**

An Error indicating the success or failure of the function.

#### 4.1.1.28 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**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *pActiveBank* | The currently active bank. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.29 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**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *index* | Zero based index of camera. |
| *pGuid* | Unique PGRGuid for the camera. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.30 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 fc2Connect() call.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *serial-Number* | Serial number of camera. |
| *pGuid* | Unique PGRGuid for the camera. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.31 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**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *pCameraInfo* | Pointer to the camera information structure to be filled. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.32 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**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *index* | Zero based index of desired camera. |
| *pSerial-Number* | Serial number of camera. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.33 FLYCAPTURE2_C_API fc2Error fc2GetConfiguration ( fc2Context *context*, fc2Config * *config* )

Get the configuration associated with the camera.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *config* | Pointer to the configuration structure to be filled. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.34 FLYCAPTURE2_C_API fc2Error fc2GetCycleTime ( fc2Context *context,* fc2TimeStamp * *pTimeStamp* )

Get cycle time from camera.

**Parameters**

| | |
|---|---|
| *Timestamp* | struct. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.35 FLYCAPTURE2_C_API fc2Error fc2GetDefaultColorProcessing ( fc2ColorProcessingAlgorithm * *pDefaultMethod* )

Get the default color processing algorithm.

**Parameters**

| | |
|---|---|
| *pDefault-Method* | The default color processing algorithm. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.36 FLYCAPTURE2_C_API fc2Error fc2GetDefaultOutputFormat ( fc2PixelFormat * *pFormat* )

Get the default output pixel format.

**Parameters**

| | |
|---|---|
| *pFormat* | The default pixel format. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.37 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**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *index* | Zero based index of device. |
| *pGuid* | Unique PGRGuid for the device. |

**See also**

fc2GetNumOfDevices()

**Returns**

An Error indicating the success or failure of the function.

#### 4.1.1.38 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.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *pInfo* | Structure to be filled. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.39 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**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *imageSettings* | Current image settings. |
| *packetSize* | Current packet size. |
| *percentage* | Current packet size as a percentage. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.40 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**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *info* | Structure to be filled with the capabilities of the specified mode and the current state in the specified mode. |
| *pSupported* | Whether the specified mode is supported. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.41 FLYCAPTURE2_C_API fc2Error fc2GetGigEConfig ( fc2Context *context*, fc2GigEConfig * *pConfig* )

#### 4.1.1.42 FLYCAPTURE2_C_API fc2Error fc2GetGigEImageBinningSettings ( fc2Context *context*, unsigned int * *horzBinnningValue*, unsigned int * *vertBinnningValue* )

#### 4.1.1.43 FLYCAPTURE2_C_API fc2Error fc2GetGigEImageSettings ( fc2Context *context*, fc2GigEImageSettings * *pImageSettings* )

#### 4.1.1.44 FLYCAPTURE2_C_API fc2Error fc2GetGigEImageSettingsInfo ( fc2Context *context*, fc2GigEImageSettingsInfo * *pInfo* )

#### 4.1.1.45 FLYCAPTURE2_C_API fc2Error fc2GetGigEImagingMode ( fc2Context *context*, fc2Mode * *mode* )

#### 4.1.1.46 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**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *pGigEProp* | The GigE property to get. |

**Returns**

An Error indicating the success or failure of the function.

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

#### 4.1.1.48 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.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *pin* | Pin to get the direction for. |
| *pDirection* | Direction of the pin. 0 for input, 1 for output. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.49 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**

| | |
|---|---|
| *pImage* | The fc2Image to be used. |
| *ppData* | A pointer to the image data. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.50 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**

| | |
|---|---|
| *imageStatisticsContext* | The statistics context. |
| *channel* | The statistics channel. |
| *pRangeMin* | The minimum possible value. |
| *pRangeMax* | The maximum possible value. |
| *pPixelValueMin* | The minimum pixel value. |
| *pPixelValueMax* | The maximum pixel value. |
| *pNumPixelValues* | The number of unique pixel values. |
| *pPixelValueMean* | The mean of the image. |
| *ppHistogram* | Pointer to an array containing the histogram. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.51 FLYCAPTURE2_C_API fc2TimeStamp fc2GetImageTimeStamp ( fc2Image * *pImage* )

Get the timestamp data associated with the image.

**Parameters**

| | |
|---|---|
| *pImage* | The fc2Image to be used. |

**Returns**

Timestamp data associated with the image.

#### 4.1.1.52 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**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *pGuid* | The PGRGuid to get the interface for. |
| *pInterface-Type* | The interface type of the PGRGuid. |

**Returns**

#### 4.1.1.53 FLYCAPTURE2_C_API fc2Error fc2GetLibraryVersion ( fc2Version * *pVersion* )

Get library version.

**Parameters**

| | |
|---|---|
| *pVersion* | Structure to receive the library version. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.54 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**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *bank* | The bank to query. |
| *pRead-Supported* | Whether reading from the bank is supported. |
| *pWrite-Supported* | Whether writing to the bank is supported. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.55 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.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *bank* | Bank to retrieve. |
| *channel* | Channel to retrieve. |
| *sizeEntries* | Number of entries in LUT table to read. |
| *pEntries* | Array to store LUT entries. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.56 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.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *pData* | The LUT structure to be filled. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.57 FLYCAPTURE2_C_API fc2Error fc2GetMemoryChannel ( fc2Context *context*, unsigned int * *pCurrentChannel* )

Retrieve the current memory channel from the camera.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *pCurrent-Channel* | Current memory channel. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.58 FLYCAPTURE2_C_API fc2Error fc2GetMemoryChannelInfo ( fc2Context *context*, unsigned int * *pNumChannels* )

Query the camera for memory channel support.

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

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *pNumChannels* | Number of memory channels supported. |

**Returns**

A fc2Error indicating the success or failure of the function.

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

Gets the number of cameras attached to the PC.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *pNumCameras* | Number of cameras detected. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.60 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**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *pNumDevices* | The number of devices found. |

**Returns**

An Error indicating the success or failure of the function.

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

#### 4.1.1.62 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.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *prop* | Pointer to the Property structure to be filled. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.63 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.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *propInfo* | Pointer to the PropertyInfo structure to be filled. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.64 FLYCAPTURE2_C_API const char* fc2GetRegisterString ( unsigned int *registerVal* )

Returns a text representation of the register value.

**Parameters**

| | |
|---|---|
| *registerVal* | The register value to query. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.65 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.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *strobeControl* | Structure to receive strobe settings. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.66 FLYCAPTURE2_C_API fc2Error fc2GetStrobeInfo ( fc2Context *context*, fc2StrobeInfo * *strobeInfo* )

Retrieve strobe information from the camera.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *strobeInfo* | Structure to receive strobe information. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.67 FLYCAPTURE2_C_API fc2Error fc2GetSystemInfo ( fc2SystemInfo * *pSystemInfo* )

Get system information.

**Parameters**

| | |
|---|---|
| *pSystemInfo* | Structure to receive system information. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.68 FLYCAPTURE2_C_API fc2Error fc2GetTriggerDelay ( fc2Context *context*, fc2TriggerDelay * *triggerDelay* )

Retrieve current trigger delay settings from the camera.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *triggerDelay* | Structure to receive trigger delay settings. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.69 FLYCAPTURE2_C_API fc2Error fc2GetTriggerDelayInfo ( fc2Context *context*, fc2TriggerDelayInfo * *triggerDelayInfo* )

Retrieve trigger delay information from the camera.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *triggerDelay-Info* | Structure to receive trigger delay information. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.70 FLYCAPTURE2_C_API fc2Error fc2GetTriggerMode ( fc2Context *context*, fc2TriggerMode * *triggerMode* )

Retrieve current trigger settings from the camera.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *triggerMode* | Structure to receive trigger mode settings. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.71 FLYCAPTURE2_C_API fc2Error fc2GetTriggerModeInfo ( fc2Context *context*, fc2TriggerModeInfo * *triggerModeInfo* )

Retrieve trigger information from the camera.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *triggerModeInfo* | Structure to receive trigger information. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.72 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 FRAMERATE_FORMAT7.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *videoMode* | Current video mode. |
| *frameRate* | Current frame rate. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.73 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**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *videoMode* | Video mode to check. |
| *frameRate* | Frame rate to check. |
| *pSupported* | Whether the video mode and frame rate is supported. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.74 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**

| | |
|---|---|
| *AVIContext* | The AVI context to use. |
| *pFileName* | The filename of the AVI file. |
| *pOption* | Options to apply to the AVI file. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.75 FLYCAPTURE2_C_API fc2Error fc2IsCameraControlable ( fc2Context *context*, fc2PGRGuid * *pGuid*, BOOL * *pControlable* )

Query whether a GigE camera is controlable.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *pGuid* | Unique PGRGuid for the camera. |
| *pControlable* | True indicates camera is controllable |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.76 FLYCAPTURE2_C_API fc2Error fc2LaunchBrowser ( const char * *pAddress* )

Launch a URL in the system default browser.

**Parameters**

| | |
|---|---|
| *pAddress* | URL to open in browser. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.77 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**

| | |
|---|---|
| *pCommand* | Command to execute. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.78 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**

| | |
|---|---|
| *pCommand* | Command to execute. |
| *pCallback* | Callback to fire when command is complete. |
| *pUserData* | Data pointer to pass to callback. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.79 FLYCAPTURE2_C_API fc2Error fc2LaunchHelp ( const char * *pFileName* )

Open a CHM file in the system default CHM viewer.

**Parameters**

| | |
|---|---|
| *pFileName* | Filename of CHM file to open. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.80 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**

| | |
|---|---|
| *AVIContext* | The AVI context to use. |
| *pFileName* | The filename of the AVI file. |
| *pOption* | Options to apply to the AVI file. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.81 FLYCAPTURE2_C_API fc2Error fc2QueryGigEImagingMode ( fc2Context *context*, fc2Mode *mode*, BOOL * *isSupported* )

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

Read a GVCP memory block.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *address* | GVCP address to be read from. |
| *pBuffer* | Array containing data to be written. |
| *length* | Size of array, in quadlets. |

**Returns**

An Error indicating the success or failure of the function.

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

Read a GVCP register.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *address* | GVCP address to be read from. |
| *pValue* | The value that is read. |

**Returns**

An Error indicating the success or failure of the function.

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

Read a GVCP register block.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *address* | GVCP address to be read from. |
| *pBuffer* | Array containing data to be written. |
| *length* | Size of array, in quadlets. |

**Returns**

An Error indicating the success or failure of the function.

#### 4.1.1.85 FLYCAPTURE2_C_API fc2Error fc2ReadRegister ( fc2Context *context*, unsigned int *address*, unsigned int * *pValue* )

Read the specified register from the camera.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *address* | DCAM address to be read from. |
| *pValue* | The value that is read. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.86 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.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *addressHigh* | Top 16 bits of the 48 bit absolute address to read from. |
| *addressLow* | Bottom 32 bits of the 48 bits absolute address to read from. |
| *pBuffer* | Array to store read data. |
| *length* | Size of array, in quadlets. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.87 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**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *enumCallback* | Pointer to function that will receive the callback. |
| *callbackType* | Type of callback to register for. |
| *pParameter* | Callback parameter to be passed to callback. |
| *pCallbackHandle* | Unique callback handle used for unregistering callback. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.88 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.

#### 4.1.1.89 FLYCAPTURE2_C_API fc2Error fc2RestoreFromMemoryChannel ( fc2Context *context*, unsigned int *channel* )

Restore the specfied current memory channel.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *channel* | Memory channel to restore from. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.90 FLYCAPTURE2_C_API fc2Error fc2RetrieveBuffer ( fc2Context *context*, fc2Image * *pImage* )

Retrieves the the next image object containing the next image.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *pImage* | Pointer to fc2Image to store image data. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.91 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**

| | |
|---|---|
| *pImage* | The fc2Image to be used. |
| *pFilename* | Filename to save image with. |
| *format* | File format to save in. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.92 FLYCAPTURE2_C_API fc2Error fc2SaveImageWithOption ( fc2Image * *pImage*, const char * *pFilename*, fc2ImageFileFormat *format*, void * *pOption* )

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

**Parameters**

| | |
|---|---|
| *pImage* | The fc2Image to be used. |
| *pFilename* | Filename to save image with. |
| *format* | File format to save in. |
| *pOption* | Options for saving image. |

**Returns**

A fc2Error indicating the success or failure of the function.

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

Save the current settings to the specfied current memory channel.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *channel* | Memory channel to save to. |

**Returns**

A fc2Error indicating the success or failure of the function.

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

Set the LUT bank that will be used.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *activeBank* | The bank to be set as active. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.95 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.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *pCallbackFn* | A function to be called when a new image is received. |
| *pCallback-Data* | A pointer to data that can be passed to the callback function. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.96 FLYCAPTURE2_C_API fc2Error fc2SetConfiguration ( fc2Context *context,* fc2Config * *config* )

Set the configuration associated with the camera.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *config* | Pointer to the configuration structure to be used. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.97 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**

| | |
|---|---|
| *default-Method* | The color processing algorithm to set. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.98 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**

| | |
|---|---|
| *format* | The output pixel format to set. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.99 FLYCAPTURE2_C_API fc2Error fc2SetEmbeddedImageInfo ( fc2Context *context*, fc2EmbeddedImageInfo * *pInfo* )

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

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *pInfo* | Structure to be used. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.100 FLYCAPTURE2_C_API fc2Error fc2SetFormat7Configuration ( fc2Context *context*, fc2Format7ImageSettings * *imageSettings*, float *percentSpeed* )

Set the current Format7 configuration to the camera.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *imageSettings* | Image settings to be written to the camera. |
| *percentSpeed* | Packet size as a percentage to be written to the camera. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.101 FLYCAPTURE2_C_API fc2Error fc2SetFormat7ConfigurationPacket ( fc2Context *context*, fc2Format7ImageSettings * *imageSettings*, unsigned int *packetSize* )

Set the current Format7 configuration to the camera.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *image-Settings* | Image settings to be written to the camera. |
| *packetSize* | Packet size to be written to the camera. |

**Returns**

A fc2Error indicating the success or failure of the function.

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

#### 4.1.1.103 FLYCAPTURE2_C_API fc2Error fc2SetGigEImageBinningSettings ( fc2Context *context,* unsigned int *horzBinnningValue,* unsigned int *vertBinnningValue* )

#### 4.1.1.104 FLYCAPTURE2_C_API fc2Error fc2SetGigEImageSettings ( fc2Context *context,* const fc2GigEImageSettings * *pImageSettings* )

#### 4.1.1.105 FLYCAPTURE2_C_API fc2Error fc2SetGigEImagingMode ( fc2Context *context,* fc2Mode *mode* )

#### 4.1.1.106 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**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *pGigEProp* | The GigE property to set. |

**Returns**

An Error indicating the success or failure of the function.

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

#### 4.1.1.108 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.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *pin* | Pin to get the direction for. |
| *direction* | Direction of the pin. 0 for input, 1 for output. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.109 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.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *pin* | Pin to get the direction for. |
| *direction* | Direction of the pin. 0 for input, 1 for output. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.110 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**

| | |
|---|---|
| *pImage* | The fc2Image to be used. |
| *pData* | Pointer to the image buffer. |
| *dataSize* | Size of the image buffer. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.111 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**

| | |
|---|---|
| *pImage* | The fc2Image to be used. |
| *rows* | Number of rows to set. |
| *cols* | Number of cols to set. |
| *stride* | Stride to set. |
| *pixelFormat* | Pixel format to set. |
| *bayerFormat* | Bayer tile format to set. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.112 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.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *bank* | Bank to set. |
| *channel* | Channel to set. |
| *sizeEntries* | Number of entries in LUT table to write. This must be the same size as numEntries returned by GetLutInfo(). |
| *pEntries* | Array containing LUT entries to write. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.113 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.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *prop* | Pointer to the Property structure to be used. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.114 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**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *prop* | Pointer to the Property structure to be used. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.115 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.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *strobeControl* | Structure providing strobe settings. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.116 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.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *strobeControl* | Structure providing strobe settings. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.117 FLYCAPTURE2_C_API fc2Error fc2SetTriggerDelay ( fc2Context *context*, fc2TriggerDelay * *triggerDelay* )

Set the specified trigger delay settings to the camera.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *triggerDelay* | Structure providing trigger delay settings. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.118 FLYCAPTURE2_C_API fc2Error fc2SetTriggerDelayBroadcast ( fc2Context *context*, fc2TriggerDelay * *triggerDelay* )

Set the specified trigger delay settings to the camera.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *triggerDelay* | Structure providing trigger delay settings. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.119 FLYCAPTURE2_C_API fc2Error fc2SetTriggerMode ( fc2Context *context*, fc2TriggerMode * *triggerMode* )

Set the specified trigger settings to the camera.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *triggerMode* | Structure providing trigger mode settings. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.120 FLYCAPTURE2_C_API fc2Error fc2SetTriggerModeBroadcast ( fc2Context *context*, fc2TriggerMode * *triggerMode* )

Set the specified trigger settings to the camera.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *triggerMode* | Structure providing trigger mode settings. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.121 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.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *ppMemBuffers* | Pointer to memory buffers to be written to. The size of the data should be equal to (size * numBuffers) or larger. |
| *size* | The size of each buffer (in bytes). |
| *nNumBuffers* | Number of buffers in the array. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.122 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**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *videoMode* | Video mode to set to camera. |
| *frameRate* | Frame rate to set to camera. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.123 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.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.124 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.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *pCallbackFn* | A function to be called when a new image is received. |
| *pCallback-Data* | 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.

#### 4.1.1.125 FLYCAPTURE2_C_API fc2Error fc2StartSyncCapture ( unsigned int *numCameras,* fc2Context * *pContexts* )

Starts synchronized isochronous image capture on multiple cameras.

**Parameters**

| | |
|---|---|
| *numCameras* | Number of fc2Contexts in the ppCameras array. |
| *pContexts* | Array of fc2Contexts. |

**Returns**

A fc2Error indicating the success or failure of the function.

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

Starts synchronized isochronous image capture on multiple cameras.

**Parameters**

| | |
|---|---|
| *numCameras* | Number of fc2Contexts in the ppCameras array. |
| *pContexts* | Array of fc2Contexts. |
| *pCallbackFns* | Array of callback functions for each camera. |
| *pCallbackDataArray* | Array of callback data pointers. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.127 FLYCAPTURE2_C_API fc2Error fc2StopCapture ( fc2Context *context* )

Stops isochronous image transfer and cleans up all associated resources.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.128 FLYCAPTURE2_C_API fc2Error fc2UnregisterCallback ( fc2Context *context*, fc2CallbackHandle *callbackHandle* )

Unregister a callback function.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *callback-Handle* | Unique callback handle. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.129 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**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *image-Settings* | Structure containing the image settings. |
| *settingsAre-Valid* | Whether the settings are valid. |
| *packetInfo* | Packet size information that can be used to determine a valid packet size. |

**Returns**

A fc2Error indicating the success or failure of the function.

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

Write a GVCP memory block.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *address* | GVCP address to be write to. |
| *pBuffer* | Array containing data to be written. |
| *length* | Size of array, in quadlets. |

**Returns**

An Error indicating the success or failure of the function.

#### 4.1.1.131 FLYCAPTURE2_C_API fc2Error fc2WriteGVCPRegister ( fc2Context *context*, unsigned int *address*, unsigned int *value* )

Write a GVCP register.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *address* | GVCP address to be written to. |
| *value* | The value to be written. |

**Returns**

An Error indicating the success or failure of the function.

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

Write a GVCP register block.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *address* | GVCP address to be write to. |
| *pBuffer* | Array containing data to be written. |
| *length* | Size of array, in quadlets. |

**Returns**

An Error indicating the success or failure of the function.

#### 4.1.1.133 FLYCAPTURE2_C_API fc2Error fc2WriteGVCPRegisterBroadcast ( fc2Context *context*, unsigned int *address*, unsigned int *value* )

Write a GVCP register with broadcast.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *address* | GVCP address to be written to. |
| *value* | The value to be written. |

**Returns**

An Error indicating the success or failure of the function.

#### 4.1.1.134 FLYCAPTURE2_C_API fc2Error fc2WriteRegister ( fc2Context *context*, unsigned int *address*, unsigned int *value* )

Write to the specified register on the camera.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *address* | DCAM address to be written to. |
| *value* | The value to be written. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.135 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.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *addressHigh* | Top 16 bits of the 48 bit absolute address to write to. |
| *addressLow* | Bottom 32 bits of the 48 bits absolute address to write to. |
| *pBuffer* | Array containing data to be written. |
| *length* | Size of array, in quadlets. |

**Returns**

A fc2Error indicating the success or failure of the function.

#### 4.1.1.136 FLYCAPTURE2_C_API fc2Error fc2WriteRegisterBroadcast ( fc2Context *context*, unsigned int *address*, unsigned int *value* )

Write to the specified register on the camera with broadcast.

**Parameters**

| | |
|---|---|
| *context* | The fc2Context to be used. |
| *address* | DCAM address to be written to. |
| *value* | The value to be written. |

**Returns**

A fc2Error indicating the success or failure of the function.

## 4.2 FlyCapture2Defs_C.h File Reference

### Data Structures

- struct fc2PGRGuid

  *A GUID to the camera.*
- struct fc2Image
- struct fc2SystemInfo
- struct fc2Version
- struct fc2Config
- struct fc2TriggerDelayInfo
- struct fc2TriggerDelay
- struct fc2TriggerModeInfo
- struct fc2TriggerMode
- struct fc2StrobeInfo
- struct fc2StrobeControl
- struct fc2Format7ImageSettings
- struct fc2Format7Info
- struct fc2Format7PacketInfo
- struct fc2IPAddress
- struct fc2MACAddress
- struct fc2GigEProperty
- struct fc2GigEStreamChannel
- struct fc2GigEConfig
- struct fc2GigEImageSettingsInfo
- struct fc2GigEImageSettings

- struct fc2TimeStamp
- struct fc2ConfigROM
- struct fc2CameraInfo
- struct fc2EmbeddedImageInfoProperty
- struct fc2EmbeddedImageInfo
- struct fc2ImageMetadata
- struct fc2LUTData
- struct fc2PNGOption
- struct fc2PPMOption
- struct fc2PGMOption
- struct fc2TIFFOption
- struct fc2JPEGOption
- struct fc2JPG2Option
- struct fc2AVIOption
- struct fc2MJPGOption
- struct fc2H264Option

## Defines

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

## 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_INTITIALIZED, 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_FORCE_32BITS = FULL_32BIT_VALUE }
- enum fc2BusCallbackType { FC2_BUS_RESET, FC2_ARRIVAL, FC2_REMOVAL, FC2_CALLBACK_TYPE_FORCE_32BITS = FULL_32BIT_VALUE }
- enum fc2GrabMode { FC2_DROP_FRAMES, FC2_BUFFER_FRAMES, FC2_UNSPECIFIED_GRAB_MODE, FC2_GRAB_MODE_FORCE_32BITS = FULL_32BIT_VALUE }
- enum fc2GrabTimeout { FC2_TIMEOUT_NONE = 0, FC2_TIMEOUT_INFINITE = -1, FC2_TIMEOUT_UNSPECIFIED = -2, FC2_GRAB_TIMEOUT_FORCE_32BITS = FULL_32BIT_VALUE }
- 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 }
- 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 }
- enum fc2DriverType { FC2_DRIVER_1394_CAM, FC2_DRIVER_1394_PRO, FC2_DRIVER_1394_JUJU, 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_UNKNOWN = -1, FC2_DRIVER_FORCE_32BITS = FULL_32BIT_VALUE }

  *Types of low level drivers that flycapture uses.*

- 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 }
- 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 }
- 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 }
- 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 }
- 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 }

- 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 }
- enum fc2PCIeBusSpeed { FC2_PCIE_BUSSPEED_2_5, FC2_PCIE_BUSSPEED_5_0, FC2_PCIE_BUSSPEED_UNKNOWN = -1, FC2_PCIE_BUSSPEED_FORCE_32BITS = FULL_32BIT_VALUE }
- 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 }
- enum fc2BayerTileFormat { FC2_BT_NONE, FC2_BT_RGGB, FC2_BT_GRBG, FC2_BT_GBRG, FC2_BT_BGGR, FC2_BT_FORCE_32BITS = FULL_32BIT_VALUE }
- 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 }
- enum fc2GigEPropertyType { FC2_HEARTBEAT, FC2_HEARTBEAT_TIMEOUT }
- 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 }
- 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 }
- enum fc2ByteOrder { FC2_BYTE_ORDER_LITTLE_ENDIAN, FC2_BYTE_ORDER_BIG_ENDIAN, FC2_BYTE_ORDER_FORCE_32BITS = FULL_32BIT_VALUE }
- 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 }

### 4.2.1 Define Documentation

#### 4.2.1.1 #define FALSE 0

#### 4.2.1.2 #define FULL_32BIT_VALUE 0x7FFFFFFF

#### 4.2.1.3 #define MAX_STRING_LENGTH 512

#### 4.2.1.4 #define TRUE 1

### 4.2.2 Typedef Documentation

#### 4.2.2.1 typedef int BOOL

#### 4.2.2.2 typedef void(* fc2AsyncCommandCallback)(fc2Error retError, void *pUserData)

#### 4.2.2.3 typedef void* fc2AVIContext

A context referring to the AVI recorder object.

#### 4.2.2.4 typedef void(* fc2BusEventCallback)(void *pParameter, unsigned int serialNumber)

#### 4.2.2.5 typedef void* fc2CallbackHandle

#### 4.2.2.6 typedef void* fc2Context

A context to the FlyCapture2 C library.

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

#### 4.2.2.7 typedef void* fc2GuiContext

A context to the FlyCapture2 C GUI library.

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

#### 4.2.2.8 typedef void(* fc2ImageEventCallback)(fc2Image *image, void *pCallbackData)

#### 4.2.2.9 typedef void* fc2ImageImpl

An internal pointer used in the fc2Image structure.

#### 4.2.2.10 typedef void* fc2ImageStatisticsContext

A context referring to the ImageStatistics object.

### 4.2.3 Enumeration Type Documentation

#### 4.2.3.1 enum fc2BandwidthAllocation

**Enumerator:**

***FC2_BANDWIDTH_ALLOCATION_OFF***

***FC2_BANDWIDTH_ALLOCATION_ON***

***FC2_BANDWIDTH_ALLOCATION_UNSUPPORTED***

***FC2_BANDWIDTH_ALLOCATION_UNSPECIFIED***

***FC2_BANDWIDTH_ALLOCATION_FORCE_32BITS***

#### 4.2.3.2 enum fc2BayerTileFormat

**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***

#### 4.2.3.3 enum fc2BusCallbackType

**Enumerator:**

***FC2_BUS_RESET***
***FC2_ARRIVAL***
***FC2_REMOVAL***
***FC2_CALLBACK_TYPE_FORCE_32BITS***

#### 4.2.3.4 enum fc2BusSpeed

**Enumerator:**

***FC2_BUSSPEED_S100*** 100Mbits/sec.
***FC2_BUSSPEED_S200*** 200Mbits/sec.
***FC2_BUSSPEED_S400*** 400Mbits/sec.
***FC2_BUSSPEED_S480*** 480Mbits/sec. Only for USB2 cameras.
***FC2_BUSSPEED_S800*** 800Mbits/sec.
***FC2_BUSSPEED_S1600*** 1600Mbits/sec.
***FC2_BUSSPEED_S3200*** 3200Mbits/sec.
***FC2_BUSSPEED_S5000*** 5000Mbits/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***

#### 4.2.3.5 enum fc2ByteOrder

**Enumerator:**

***FC2_BYTE_ORDER_LITTLE_ENDIAN***

***FC2_BYTE_ORDER_BIG_ENDIAN***

***FC2_BYTE_ORDER_FORCE_32BITS***

#### 4.2.3.6 enum fc2ColorProcessingAlgorithm

**Enumerator:**

***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***

#### 4.2.3.7 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*** PGRXHCI.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_UNKNOWN*** Unknown driver type.

***FC2_DRIVER_FORCE_32BITS***

#### 4.2.3.8 enum fc2Error

**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_INTITIALIZED*** 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_FORCE_32BITS***

#### 4.2.3.9 enum fc2FrameRate

**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***

#### 4.2.3.10 enum fc2GigEPropertyType

**Enumerator:**

***FC2_HEARTBEAT***

***FC2_HEARTBEAT_TIMEOUT***

#### 4.2.3.11 enum fc2GrabMode

**Enumerator:**

***FC2_DROP_FRAMES***

***FC2_BUFFER_FRAMES***

***FC2_UNSPECIFIED_GRAB_MODE***

***FC2_GRAB_MODE_FORCE_32BITS***

#### 4.2.3.12 enum fc2GrabTimeout

**Enumerator:**

***FC2_TIMEOUT_NONE***

***FC2_TIMEOUT_INFINITE***

***FC2_TIMEOUT_UNSPECIFIED***

***FC2_GRAB_TIMEOUT_FORCE_32BITS***

#### 4.2.3.13 enum fc2ImageFileFormat

**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***

#### 4.2.3.14 enum fc2InterfaceType

**Enumerator:**

***FC2_INTERFACE_IEEE1394***

***FC2_INTERFACE_USB_2***

***FC2_INTERFACE_USB_3***

***FC2_INTERFACE_GIGE***

***FC2_INTERFACE_UNKNOWN***

***FC2_INTERFACE_TYPE_FORCE_32BITS***

#### 4.2.3.15 enum fc2Mode

**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***

#### 4.2.3.16 enum fc2OSType

**Enumerator:**

***FC2_WINDOWS_X86***

***FC2_WINDOWS_X64***

***FC2_LINUX_X86***

***FC2_LINUX_X64***

***FC2_MAC***

***FC2_UNKNOWN_OS***

***FC2_OSTYPE_FORCE_32BITS***

#### 4.2.3.17 enum fc2PCIeBusSpeed

**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.

#### 4.2.3.18 enum fc2PixelFormat

**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.

#### 4.2.3.19 enum fc2PropertyType

**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***

#### 4.2.3.20 enum fc2StatisticsChannel

**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***

#### 4.2.3.21 enum fc2TIFFCompressionMethod

**Enumerator:**

***FC2_TIFF_NONE***

***FC2_TIFF_PACKBITS***

***FC2_TIFF_DEFLATE***

***FC2_TIFF_ADOBE_DEFLATE***

***FC2_TIFF_CCITTFAX3***

***FC2_TIFF_CCITTFAX4***

***FC2_TIFF_LZW***

***FC2_TIFF_JPEG***

#### 4.2.3.22 enum fc2VideoMode

**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***

## 4.3 FlyCapture2GUI_C.h File Reference

### 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 fc2Disonnect (fc2GuiContext context)

  *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.*

### 4.3.1 Function Documentation

#### 4.3.1.1 FLYCAPTURE2_C_API fc2Error fc2CreateGUIContext ( fc2GuiContext * *pContext* )

Create a GUI context.

**Parameters**

| | |
|---|---|
| *pContext* | Pointer to context to be created. |

**Returns**

An Error indicating the success or failure of the function.

#### 4.3.1.2 FLYCAPTURE2_C_API fc2Error fc2DestroyGUIContext ( fc2GuiContext *context* )

Destroy a GUI context.

**Parameters**

| | |
|---|---|
| *context* | Context to be destroyed. |

**Returns**

An Error indicating the success or failure of the function.

#### 4.3.1.3 FLYCAPTURE2_C_API void fc2Disonnect ( fc2GuiContext *context* )

Disconnect GUI context from camera.

**Parameters**

| | |
|---|---|
| *context* | GUI context to disconnect. |

**Returns**

An Error indicating the success or failure of the function.

#### 4.3.1.4 FLYCAPTURE2_C_API void fc2GUIConnect ( fc2GuiContext *context,* fc2Context *cameraContext* )

Connect GUI context to a camera context.

**Parameters**

| | |
|---|---|
| *context* | GUI context to connect. |
| *camera-Context* | Camera context to connect. |

**Returns**

An Error indicating the success or failure of the function.

#### 4.3.1.5 FLYCAPTURE2_C_API void fc2Hide ( fc2GuiContext *context* )

Hide the GUI.

**Parameters**

| | |
|---|---|
| *context* | Pointer to context to hide. |

**Returns**

An Error indicating the success or failure of the function.

#### 4.3.1.6 FLYCAPTURE2_C_API BOOL fc2IsVisible ( fc2GuiContext *context* )

Check if the GUI is visible.

**Parameters**

| | |
|---|---|
| *context* | Pointer to context to show. |

**Returns**

Whether the GUI is visible.

#### 4.3.1.7 FLYCAPTURE2_C_API void fc2Show ( fc2GuiContext *context* )

Show the GUI.

**Parameters**

| | |
|---|---|
| *context* | Pointer to context to show. |

**Returns**

An Error indicating the success or failure of the function.

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

Show the camera selection dialog.

**Parameters**

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

## 4.4 FlyCapture2Internal_C.h File Reference

### Data Structures

- struct fc2InternalContext
- struct fc2InternalGuiContext
- struct fc2InternalImageCallback

### Functions

- bool IsContextValid (fc2Context context)
- bool IsGuiContextValid (fc2GuiContext context)
- void SyncCppImageToStruct (fc2Image *pImage)

### 4.4.1 Function Documentation

#### 4.4.1.1 bool IsContextValid ( fc2Context *context* ) [inline]

#### 4.4.1.2 bool IsGuiContextValid ( fc2GuiContext *context* ) [inline]

#### 4.4.1.3 void SyncCppImageToStruct ( fc2Image * *pImage* ) [inline]

## 4.5 FlyCapture2Platform_C.h File Reference

### Defines

- #define FLYCAPTURE2_C_API
- #define FLYCAPTURE2_C_CALL_CONVEN

### 4.5.1 Define Documentation

#### 4.5.1.1 #define FLYCAPTURE2_C_API

#### 4.5.1.2 #define FLYCAPTURE2_C_CALL_CONVEN

## 4.6 MultiSyncLibrary_C.h File Reference

### 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.*

### 4.6.1 Function Documentation

#### 4.6.1.1 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**

| | |
|---|---|
| *pContext* | A pointer to the syncContext to be created. |

**Returns**

A syncError indicating the success or failure of the function.

#### 4.6.1.2 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**

| | |
|---|---|
| *context* | The syncContext to be destoryed. |

**Returns**

A syncError indicating the success or failure of the function.

#### 4.6.1.3 MULTISYNCLIBRARY_C_API bool syncDisableCrossPCSynchronization ( syncContext *context* )

Disable across pc synchronization support.

**Parameters**

| | |
|---|---|
| *context* | The syncContext to be used. |

**Returns**

True if operation was successful

#### 4.6.1.4 MULTISYNCLIBRARY_C_API bool syncEnableCrossPCSynchronization ( syncContext *context* )

Enable across pc synchronization support.

**Parameters**

| | |
|---|---|
| *context* | The syncContext to be used. |

**Returns**

True if operation was successful

#### 4.6.1.5 MULTISYNCLIBRARY_C_API syncMessage syncGetStatus ( syncContext *context* )

Start the sync progress.

**Parameters**

| | |
|---|---|
| *context* | The syncContext to be used. |

**Returns**

A syncMessage indicating the sync status.

#### 4.6.1.6 MULTISYNCLIBRARY_C_API double syncGetTimeSinceSynced ( syncContext *context* )

Time since sync started.

**Parameters**

| | |
|---|---|
| *context* | The syncContext to be used. |

**Returns**

Time sinced synced.

#### 4.6.1.7 MULTISYNCLIBRARY_C_API bool syncIsTimingBusConnected ( syncContext *context* )

Whether syncing across PCs.

**Parameters**

| | |
|---|---|
| *context* | The syncContext to be used. |

**Returns**

True if its syncing across PC

#### 4.6.1.8 MULTISYNCLIBRARY_C_API bool syncQueryCrossPCSynchronizationSetting ( syncContext *context* )

Query cross pc synchronizaion support status.

**Parameters**

| | |
|---|---|
| *context* | The syncContext to be used. |

**Returns**

True if cross pc synchronization was supported

#### 4.6.1.9 MULTISYNCLIBRARY_C_API syncError syncRescanMasterTimingBus ( syncContext *context* )

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

**Parameters**

| | |
|---|---|
| *context* | The syncContext to be used. |

**Returns**

A syncError indicating the success or failure of the function.

#### 4.6.1.10 MULTISYNCLIBRARY_C_API syncError syncStart ( syncContext *context* )

Start the sync progress.

**Parameters**

| | |
|---|---|
| *context* | The syncContext to be used. |

**Returns**

A syncError indicating the success or failure of the function.

#### 4.6.1.11 MULTISYNCLIBRARY_C_API syncError syncStop ( syncContext *context* )

Stop the sync progress.

**Parameters**

| | |
|---|---|
| *context* | The syncContext to be used. |

**Returns**

A syncError indicating the success or failure of the function.

## 4.7 MultiSyncLibraryDefs_C.h File Reference

## Defines

- #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 }

### 4.7.1 Define Documentation

#### 4.7.1.1 #define FALSE 0

#### 4.7.1.2 #define FULL_32BIT_VALUE 0x7FFFFFFF

#### 4.7.1.3 #define MAX_STRING_LENGTH 512

#### 4.7.1.4 #define TRUE 1

### 4.7.2 Typedef Documentation

#### 4.7.2.1 typedef int BOOL

#### 4.7.2.2 typedef void* syncContext

A context to the MultiSync C library.

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

### 4.7.3 Enumeration Type Documentation

#### 4.7.3.1 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***

#### 4.7.3.2 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***

## 4.8 MultiSyncLibraryPlatform_C.h File Reference

### Defines

- #define MULTISYNCLIBRARY_C_API
- #define MULTISYNCLIBRARY_C_CALL_CONVEN

### 4.8.1 Define Documentation

#### 4.8.1.1 #define MULTISYNCLIBRARY_C_API

#### 4.8.1.2 #define MULTISYNCLIBRARY_C_CALL_CONVEN

# Index