FlyCapture2 C

2.12.3.2

Generated by Doxygen 1.7.5

Tue Jan 9 2018 21:59:32

Contents

Chapter 1

Software Licensing Information

Component	License
FlyCapture2	Copyright © 2017 FLIR Integrated Imaging Solutions, Inc. All Rights Reserved. This software is the confidential and proprietary information of FLIR Integrated Imaging Solutions, Inc. ("Confidential Information"). You shall not disclose such Confidential Information and shall use it only in accordance with the terms of the license agreement you entered into with FLIR Integrated Imaging Solutions, Inc. (FLIR). FLIR MAKES NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THE SOFTWARE, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, OR NON-INFRINGEMENT. FLIR SHALL NOT BE LIABLE FOR ANY DAMAGES SUFFERED BY LICENSEE AS A RESULT OF USING, MODIFYING OR DISTRIBUTING THIS SOFTWARE OR ITS DERIVATIVES.
AdapterList	The Code Project Open License (CPOL) http://www.codeproject.-com/info/cpol10.aspx
Boost	Boost Software License http://www.boost.-org/users/license.html
FFMPEG	LGPv2.1 License https://www.-ffmpeg.org/legal.html
FreeImage	FreeImage public license http-://freeimage.sourceforge.-net/freeimage-license.txt
GTK	LGPv2.1 License http://www.gnu.-org/licenses/old-licenses/lgpl-2.-1.txt
Libusb	LGPLv2.1 License http://www.gnu.-org/licenses/old-licenses/lgpl-2.-1.txt
Libraw1394	LGPLv2.0 License http://www.gnu.-org/licenses/old-licenses/lgpl-2.-0.txt

Table 1.1: License table

Chapter 2

Deprecated List

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

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

Chapter 3

Module Index

3.1 Modules

Here is a list of all modules:

Chapter 4

Data Structure Index

4.1 Data Structures

Here are the data structures with brief descriptions:

Chapter 5

File Index

5.1 File List

Here is a list of all files with brief descriptions:

Chapter 6

Module Documentation

6.1 Bus Manager Operation

The functions in this section provide access to BusManager operations.

Functions

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

  Fire a bus reset.
- FLYCAPTURE2_C_API fc2Error fc2GetNumOfCameras (fc2Context context, unsigned int *pNumCameras)

  Gets the number of cameras attached to the PC.
- FLYCAPTURE2_C_API fc2Error fc2GetCameraFromIPAddress (fc2Context context, fc2IPAddress ipAddress, fc2PGRGuid *pGuid)

  Gets the PGRGuid for a camera with the specified IPv4 address.
- FLYCAPTURE2_C_API fc2Error fc2GetCameraFromIndex (fc2Context context, unsigned int index, fc2PGRGuid *pGuid)

  Gets the PGRGuid for a camera on the PC.
- FLYCAPTURE2_C_API fc2Error fc2GetCameraFromSerialNumber (fc2Context context, unsigned int 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 fc2ReadPhyRegister (fc2Context context, fc2PGRGuid guid, unsigned int page, unsigned int port, unsigned int address, unsigned int *pValue)

  Read a phy register on the specified device.
- FLYCAPTURE2_C_API fc2Error fc2WritePhyRegister (fc2Context context, fc2PGRGuid guid, unsigned int page, unsigned int port, unsigned int address, unsigned int value)

  Write a phy register on the specified device.
- FLYCAPTURE2_C_API fc2Error fc2GetUsbLinkInfo (fc2Context context, fc2PGRGuid guid, unsigned int *pValue)

  Read usb link info for the port that the specified device is connected to.
- FLYCAPTURE2_C_API fc2Error fc2GetUsbPortStatus (fc2Context context, fc2PGRGuid guid, unsigned int *pValue)

  Read usb port status for the port that the specified device is connected to.
- FLYCAPTURE2_C_API fc2Error fc2GetTopology (fc2Context context, fc2TopologyNodeContext *pTopologyNodeContext)

  Gets the topology information for the PC.
- FLYCAPTURE2_C_API fc2Error fc2RegisterCallback (fc2Context context, fc2BusEventCallback enumCallback, fc2BusCallbackType callbackType, void *pParameter, fc2CallbackHandle *pCallbackHandle)

  Register a callback function that will be called when the specified callback event occurs.
- FLYCAPTURE2_C_API fc2Error fc2UnregisterCallback (fc2Context context, fc2CallbackHandle callbackHandle)

  Unregister a callback function.
- FLYCAPTURE2_C_API fc2Error fc2RescanBus (fc2Context context)

  Force a rescan of the buses.
- FLYCAPTURE2_C_API fc2Error fc2ForceIPAddressToCamera (fc2Context context, fc2MACAddress macAddress, fc2IPAddress ipAddress, fc2IPAddress subnetMask, fc2IPAddress defaultGateway)

  Force the camera with the specific MAC address to the specified IP address, subnet mask and default gateway.
- FLYCAPTURE2_C_API fc2Error fc2ForceAllIPAddressesAutomatically ()

  Force all cameras on the network to be assigned sequential IP addresses on the same subnet as the network adapters that they are connected to.
- FLYCAPTURE2_C_API fc2Error fc2ForceIPAddressAutomatically (unsigned int serialNumber)

  Force cameras on the network to be assigned sequential IP addresses on the same subnet as the network adapters that it is connected to.
- FLYCAPTURE2_C_API fc2Error fc2DiscoverGigECameras (fc2Context context, fc2CameraInfo *gigECameras, unsigned int *arraySize)

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

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

  Query whether a GigE camera is controllable.

6.1.1 Detailed Description

The functions in this section provide access to BusManager operations.

6.1.2 Function Documentation

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

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

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

Parameters

context	The fc2Context to be used.
gigECameras	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.

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

Fire a bus reset.

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

Parameters

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.

6.1.2.3 FLYCAPTURE2_C_API fc2Error fc2ForceAllIPAddressesAutomatically ( )

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

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

Returns

An Error indicating the success or failure of the function.

6.1.2.4 FLYCAPTURE2_C_API fc2Error fc2ForceIPAddressAutomatically (unsigned int *serialNumber*)

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

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

Returns

An Error indicating the success or failure of the function.

6.1.2.5 FLYCAPTURE2_C_API fc2Error fc2ForceIPAddressToCamera (fc2Context *context,* fc2MACAddress *macAddress,* fc2IPAddress *ipAddress,* fc2IPAddress *subnetMask,* fc2IPAddress *defaultGateway*)

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

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

Parameters

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.

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

Gets the PGRGuid for a camera on the PC.

It uniquely identifies the camera specified by the index and is used to identify the camera during a fc2Connect() call.

Parameters

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.

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

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

Parameters

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

Returns

A fc2Error indicating the success or failure of the function.

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

Gets the PGRGuid for a camera on the PC.

It uniquely identifies the camera specified by the serial number and is used to identify the camera during a 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.

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

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

Parameters

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.

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

Gets the PGRGuid for a device.

It uniquely identifies the device specified by the index.

Parameters

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.

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

Gets the interface type associated with a PGRGuid.

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

Parameters

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

Returns

6.1.2.12 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.
pNum-Cameras	Number of cameras detected.

Returns

A fc2Error indicating the success or failure of the function.

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

Gets the number of devices.

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

Parameters

context	The fc2Context to be used.
pNum-Devices	The number of devices found.

Returns

An Error indicating the success or failure of the function.

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

Gets the topology information for the PC.

Parameters

context	The fc2Context to be used.
pTopology-Node-Context	A Topology Node context that will contain the topology information

Returns

An Error indicating the success or failure of the function.

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

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

Parameters

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

Returns

An Error indicating the success or failure of the function.

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

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

Parameters

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

Returns

An Error indicating the success or failure of the function.

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

Query whether a GigE camera is controllable.

Parameters

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

Returns

A fc2Error indicating the success or failure of the function.

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

Read a phy register on the specified device.

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

Parameters

context	The fc2Context to be used.
guid	Unique PGRGuid for the device.
page	Page to read from.
port	Port to read from.
address	Address to read from.
pValue	Value read from the phy register.

Returns

An Error indicating the success or failure of the function.

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

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

Parameters

context	The fc2Context to be used.
enum-Callback	Pointer to function that will receive the callback.
callbackType	Type of callback to register for.
pParameter	Callback parameter to be passed to callback.
pCallback-Handle	Unique callback handle used for unregistering callback.

Returns

A fc2Error indicating the success or failure of the function.

6.1.2.20 FLYCAPTURE2_C_API fc2Error fc2RescanBus (fc2Context *context*)

Force a rescan of the buses.

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

Returns

An Error indicating the success or failure of the function.

6.1.2.21 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.

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

Write a phy register on the specified device.

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

Parameters

context	The fc2Context to be used.
guid	Unique PGRGuid for the device.
page	Page to write to.
port	Port to write to.
address	Address to write to.
value	Value to write to phy register.

Returns

An Error indicating the success or failure of the function.

6.2 Connection and Image Retrieval

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

Functions

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

  Connects the fc2Context to the camera specified by the GUID.
- FLYCAPTURE2_C_API fc2Error fc2Disconnect (fc2Context context)

  Disconnects the fc2Context from the camera.
- FLYCAPTURE2_C_API BOOL fc2IsConnected (fc2Context context)

  Checks if the fc2Context is connected to a physical camera specified by a GUID.
- FLYCAPTURE2_C_API fc2Error fc2SetCallback (fc2Context context, fc2ImageEventCallback pCallbackFn, void *pCallbackData)

  Sets the callback data to be used on completion of image transfer.
- FLYCAPTURE2_C_API fc2Error fc2StartCapture (fc2Context context)

  Starts isochronous image capture.
- FLYCAPTURE2_C_API fc2Error fc2StartCaptureCallback (fc2Context context, fc2ImageEventCallback pCallbackFn, void *pCallbackData)

  Starts isochronous image capture.
- FLYCAPTURE2_C_API fc2Error fc2StartSyncCapture (unsigned int numCameras, fc2Context *pContexts)

  Starts synchronized isochronous image capture on multiple cameras.
- FLYCAPTURE2_C_API fc2Error fc2StartSyncCaptureCallback (unsigned int numCameras, fc2Context *pContexts, fc2ImageEventCallback *pCallbackFns, void **pCallbackDataArray)

  Starts synchronized isochronous image capture on multiple cameras.
- FLYCAPTURE2_C_API fc2Error fc2RetrieveBuffer (fc2Context context, fc2Image *pImage)

  Retrieves the next image object containing the next image.
- FLYCAPTURE2_C_API fc2Error fc2StopCapture (fc2Context context)

  Stops isochronous image transfer and cleans up all associated resources.
- FLYCAPTURE2_C_API fc2Error fc2WaitForBufferEvent (fc2Context context, fc2Image *pImage, unsigned int eventNumber)

  Retrieves the next image event containing the next part of the image.
- FLYCAPTURE2_C_API fc2Error fc2SetUserBuffers (fc2Context context, unsigned char *const ppMemBuffers, int size, int nNumBuffers)

  Specify user allocated buffers to use as image data buffers.
- FLYCAPTURE2_C_API fc2Error fc2GetConfiguration (fc2Context context, fc2Config *config)

  Get the configuration associated with the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetConfiguration (fc2Context context, fc2Config *config)

  Set the configuration associated with the camera.

6.2.1 Detailed Description

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

6.2.2 Function Documentation

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

Connects the fc2Context to the camera specified by the GUID.

Be aware that calling fc2CreateGUIContext() releases the CCP acquired for GigE cameras in fc2Connect(). Consider calling fc2Connect() after fc2CreateGUIContext().

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.

6.2.2.2 FLYCAPTURE2_C_API fc2Error fc2Disconnect (fc2Context *context*)

Disconnects the fc2Context from the camera.

This allows another physical camera specified by a GUID to be connected to the fc2Context.

See also

fc2Connect()

Parameters

context	The fc2Context to be used.

Returns

A fc2Error indicating the success or failure of the function.

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

Get the configuration associated with the camera.

See also

fc2SetConfiguration()

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.

6.2.2.4 FLYCAPTURE2_C_API BOOL fc2IsConnected (fc2Context *context*)

Checks if the fc2Context is connected to a physical camera specified by a GUID.

See also

fc2Connect()
fc2Disconnect()

Parameters

context	The fc2Context to be used.

Returns

Whether fc2Connect() was called on the fc2Context.

6.2.2.5 FLYCAPTURE2_C_API fc2Error fc2RetrieveBuffer (fc2Context *context*, fc2Image * *pImage*)

Retrieves the next image object containing the next image.

See also

fc2StartCapture()
fc2StopCapture()

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.

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

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

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

See also

fc2StartCapture()

Parameters

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.

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

Set the configuration associated with the camera.

See also

fc2GetConfiguration()

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.

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

Specify user allocated buffers to use as image data buffers.

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

See also

fc2StartCapture()
fc2RetrieveBuffer()
fc2StopCapture()

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.

6.2.2.9 FLYCAPTURE2_C_API fc2Error fc2StartCapture (fc2Context *context*)

Starts isochronous image capture.

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

See also

fc2RetrieveBuffer()
fc2StartSyncCapture()
fc2StopCapture()

Parameters

context	The fc2Context to be used.

Returns

A fc2Error indicating the success or failure of the function.

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

Starts isochronous image capture.

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

See also

fc2RetrieveBuffer()
fc2StartSyncCapture()
fc2StopCapture()

Parameters

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.

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

Starts synchronized isochronous image capture on multiple cameras.

This function is only used for firewire cameras.

See also

fc2RetrieveBuffer()
fc2StartCapture()
fc2StopCapture()

Parameters

numCameras	Number of fc2Contexts in the ppCameras array.
pContexts	Array of fc2Contexts.

Returns

A fc2Error indicating the success or failure of the function.

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

Starts synchronized isochronous image capture on multiple cameras.

This function is only used for firewire cameras.

See also

fc2RetrieveBuffer()
fc2StartCapture()
fc2StopCapture()

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.

6.2.2.13 FLYCAPTURE2_C_API fc2Error fc2StopCapture (fc2Context *context*)

Stops isochronous image transfer and cleans up all associated resources.

See also

fc2StartCapture()
fc2RetrieveBuffer()

Parameters

context	The fc2Context to be used.

Returns

A fc2Error indicating the success or failure of the function.

6.2.2.14 FLYCAPTURE2_C_API fc2Error fc2WaitForBufferEvent (fc2Context *context*, fc2Image * *pImage*, unsigned int *eventNumber*)

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

See also

fc2StartCapture()
fc2RetrieveBuffer()
fc2StopCapture()

Parameters

context	The fc2Context to be used.
pImage	Pointer to fc2Image to store image data.
event-Number	The event number to wait for.

Returns

An Error indicating the success or failure of the function.

6.3 Information and Properties

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

Functions

- FLYCAPTURE2_C_API fc2Error fc2GetCameraInfo (fc2Context context, fc2CameraInfo *pCameraInfo)

  Retrieves information from the camera such as serial number, model name and other camera information.
- FLYCAPTURE2_C_API fc2Error fc2GetPropertyInfo (fc2Context context, fc2PropertyInfo *propInfo)

  Retrieves information about the specified camera property.
- FLYCAPTURE2_C_API fc2Error fc2GetProperty (fc2Context context, fc2Property *prop)

  Reads the settings for the specified property from the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetProperty (fc2Context context, fc2Property *prop)

  Writes the settings for the specified property to the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetPropertyBroadcast (fc2Context context, fc2Property *prop)

  Writes the settings for the specified property to the camera.

6.3.1 Detailed Description

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

6.3.2 Function Documentation

6.3.2.1 FLYCAPTURE2_C_API fc2Error fc2GetCameraInfo (fc2Context *context*, fc2CameraInfo * *pCameraInfo*)

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

Parameters

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.

6.3.2.2 FLYCAPTURE2_C_API fc2Error fc2GetProperty (fc2Context *context*, fc2Property * *prop*)

Reads the settings for the specified property from the camera.

The property type must be specified in the fc2Property structure passed into the function in order for the function to succeed. If auto is on, the integer and abs values returned may not be consistent with each other.

See also

fc2GetPropertyInfo()
fc2SetProperty()

Parameters

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.

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

Retrieves information about the specified camera property.

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

See also

fc2GetProperty()
fc2SetProperty()

Parameters

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.

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

Writes the settings for the specified property to the camera.

The property type must be specified in the Property structure passed into the function in order for the function to succeed. The absControl flag controls whether the absolute or integer value is written to the camera. Use fc2GetPropertyInfo() to query which options are available for a specific property.

See also

fc2GetPropertyInfo()
fc2GetProperty()

Parameters

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.

6.3.2.5 FLYCAPTURE2_C_API fc2Error fc2SetPropertyBroadcast (fc2Context *context*, fc2Property * *prop*)

Writes the settings for the specified property to the camera.

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

Parameters

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.

6.4 General Purpose Input / Output

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

Functions

- FLYCAPTURE2_C_API fc2Error fc2GetGPIOPinDirection (fc2Context context, unsigned int pin, unsigned int *pDirection)

  Get the GPIO pin direction for the specified pin.
- FLYCAPTURE2_C_API fc2Error fc2SetGPIOPinDirection (fc2Context context, unsigned int pin, unsigned int direction)

  Set the GPIO pin direction for the specified pin.
- FLYCAPTURE2_C_API fc2Error fc2SetGPIOPinDirectionBroadcast (fc2Context context, unsigned int pin, unsigned int direction)

  Set the GPIO pin direction for the specified pin.

6.4.1 Detailed Description

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

6.4.2 Function Documentation

6.4.2.1 FLYCAPTURE2_C_API fc2Error fc2GetGPIOPinDirection (fc2Context *context*, unsigned int *pin*, unsigned int * *pDirection*)

Get the GPIO pin direction for the specified pin.

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

See also

fc2SetGPIOPinDirection()
fc2SetGPIOPinDirectionBroadcast()

Parameters

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.

6.4.2.2 FLYCAPTURE2_C_API fc2Error fc2SetGPIOPinDirection (fc2Context *context*, unsigned int *pin*, unsigned int *direction*)

Set the GPIO pin direction for the specified pin.

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

See also

fc2GetGPIOPinDirection()
fc2SetGPIOPinDirectionBroadcast()

Parameters

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.

6.4.2.3 FLYCAPTURE2_C_API fc2Error fc2SetGPIOPinDirectionBroadcast (fc2Context *context*, unsigned int *pin*, unsigned int *direction*)

Set the GPIO pin direction for the specified pin.

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

See also

fc2GetGPIOPinDirection()

Parameters

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.

6.5 Trigger

These functions deal with trigger control on the camera.

Functions

- 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)

  Fire the software trigger according to the DCAM specifications.
- FLYCAPTURE2_C_API fc2Error fc2FireSoftwareTriggerBroadcast (fc2Context context)

  Fire the software trigger according to the DCAM specifications.
- FLYCAPTURE2_C_API fc2Error fc2GetTriggerDelayInfo (fc2Context context, fc2TriggerDelayInfo *triggerDelayInfo)

  Retrieve trigger delay information from the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetTriggerDelay (fc2Context context, fc2TriggerDelay *triggerDelay)

  Retrieve current trigger delay settings from the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetTriggerDelay (fc2Context context, fc2TriggerDelay *triggerDelay)

  Set the specified trigger delay settings to the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetTriggerDelayBroadcast (fc2Context context, fc2TriggerDelay *triggerDelay)

  Set the specified trigger delay settings to the camera.

6.5.1 Detailed Description

These functions deal with trigger control on the camera.

6.5.2 Function Documentation

6.5.2.1 FLYCAPTURE2_C_API fc2Error fc2FireSoftwareTrigger (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.

6.5.2.2 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.

6.5.2.3 FLYCAPTURE2_C_API fc2Error fc2GetTriggerDelay (fc2Context *context*, fc2TriggerDelay * *triggerDelay*)

Retrieve current trigger delay settings from the camera.

See also

fc2GetTriggerModeInfo()
fc2GetTriggerMode()
fc2SetTriggerMode()
fc2GetTriggerDelayInfo()
fc2SetTriggerDelay()
fc2SetTriggerDelayBroadcast()

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.

6.5.2.4 FLYCAPTURE2_C_API fc2Error fc2GetTriggerDelayInfo (fc2Context *context*, fc2TriggerDelayInfo * *triggerDelayInfo*)

Retrieve trigger delay information from the camera.

See also

fc2GetTriggerModeInfo()
fc2GetTriggerMode()
fc2SetTriggerMode()
fc2GetTriggerDelay()
fc2SetTriggerDelay()
fc2SetTriggerDelayBroadcast()

Parameters

context	The fc2Context to be used.
triggerDelayInfo	Structure to receive trigger delay information.

Returns

A fc2Error indicating the success or failure of the function.

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

Retrieve current trigger settings from the camera.

See also

fc2GetTriggerModeInfo()
fc2SetTriggerMode()
fc2SetTriggerModeBroadcast()
fc2GetTriggerDelayInfo()
fc2GetTriggerDelay()
fc2SetTriggerDelay()

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.

6.5.2.6 FLYCAPTURE2_C_API fc2Error fc2GetTriggerModeInfo (fc2Context *context*, fc2TriggerModeInfo * *triggerModeInfo*)

Retrieve trigger information from the camera.

See also

fc2GetTriggerMode()
fc2SetTriggerMode()
fc2SetTriggerModeBroadcast()
fc2GetTriggerDelayInfo()
fc2GetTriggerDelay()
fc2SetTriggerDelay()

Parameters

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

Returns

A fc2Error indicating the success or failure of the function.

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

Set the specified trigger delay settings to the camera.

See also

fc2GetTriggerModeInfo()
fc2GetTriggerMode()
fc2SetTriggerMode()
fc2GetTriggerDelayInfo()
fc2GetTriggerDelay()
fc2SetTriggerDelayBroadcast()

Parameters

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

Returns

A fc2Error indicating the success or failure of the function.

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

Set the specified trigger delay settings to the camera.

See also

fc2GetTriggerModeInfo()
fc2GetTriggerMode()
fc2SetTriggerMode()
fc2GetTriggerDelayInfo()
fc2GetTriggerDelay()
fc2SetTriggerDelay()

Parameters

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

Returns

A fc2Error indicating the success or failure of the function.

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

Set the specified trigger settings to the camera.

See also

fc2GetTriggerModeInfo()
fc2GetTriggerMode()
fc2GetTriggerDelayInfo()
fc2GetTriggerDelay()
fc2SetTriggerDelay()
fc2SetTriggerModeBroadcast()

Parameters

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

Returns

A fc2Error indicating the success or failure of the function.

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

Set the specified trigger settings to the camera.

See also

fc2GetTriggerModeInfo()
fc2GetTriggerMode()
fc2GetTriggerDelayInfo()
fc2GetTriggerDelay()
fc2SetTriggerDelay()
fc2SetTriggerMode()

Parameters

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

Returns

A fc2Error indicating the success or failure of the function.

6.6 Strobe

These functions deal with strobe control on the camera.

Functions

- FLYCAPTURE2_C_API fc2Error fc2GetStrobeInfo (fc2Context context, fc2StrobeInfo *strobeInfo)

  Retrieve strobe information from the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetStrobe (fc2Context context, fc2StrobeControl *strobeControl)

  Retrieve current strobe settings from the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetStrobe (fc2Context context, fc2StrobeControl *strobeControl)

  Set current strobe settings to the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetStrobeBroadcast (fc2Context context, fc2StrobeControl *strobeControl)

  Set current strobe settings to the camera.

6.6.1 Detailed Description

These functions deal with strobe control on the camera.

6.6.2 Function Documentation

6.6.2.1 FLYCAPTURE2_C_API fc2Error fc2GetStrobe (fc2Context *context*, fc2StrobeControl * *strobeControl*)

Retrieve current strobe settings from the camera.

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

See also

fc2GetStrobeInfo()
fc2SetStrobe()
fc2SetStrobeBroadcast()

Parameters

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

Returns

A fc2Error indicating the success or failure of the function.

6.6.2.2 FLYCAPTURE2_C_API fc2Error fc2GetStrobeInfo (fc2Context *context*, fc2StrobeInfo * *strobeInfo*)

Retrieve strobe information from the camera.

See also

fc2GetStrobe()
fc2SetStrobe()
fc2SetStrobeBroadcast()

Parameters

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

Returns

A fc2Error indicating the success or failure of the function.

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

Set current strobe settings to the camera.

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

See also

fc2GetStrobeInfo()
fc2GetStrobe()
fc2SetStrobeBroadcast()

Parameters

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

Returns

A fc2Error indicating the success or failure of the function.

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

Set current strobe settings to the camera.

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

See also

fc2GetStrobeInfo()
fc2GetStrobe()
fc2SetStrobe()

Parameters

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

Returns

A fc2Error indicating the success or failure of the function.

6.7 Look Up Table

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

Functions

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

  Query if LUT support is available on the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetLUTBankInfo (fc2Context context, unsigned int bank, BOOL *pReadSupported, BOOL *pWriteSupported)

  Query the read/write status of a single LUT bank.
- FLYCAPTURE2_C_API fc2Error fc2GetActiveLUTBank (fc2Context context, unsigned int *pActiveBank)

  Get the LUT bank that is currently being used.
- FLYCAPTURE2_C_API fc2Error fc2SetActiveLUTBank (fc2Context context, unsigned int activeBank)

  Set the LUT bank that will be used.
- FLYCAPTURE2_C_API fc2Error fc2EnableLUT (fc2Context context, BOOL on)

  Enable or disable LUT functionality on the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetLUTChannel (fc2Context context, unsigned int bank, unsigned int channel, unsigned int sizeEntries, unsigned int *pEntries)

  Get the LUT channel settings from the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetLUTChannel (fc2Context context, unsigned int bank, unsigned int channel, unsigned int sizeEntries, unsigned int *pEntries)

  Set the LUT channel settings to the camera.

6.7.1 Detailed Description

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

6.7.2 Function Documentation

6.7.2.1 FLYCAPTURE2_C_API fc2Error fc2EnableLUT (fc2Context *context*, BOOL *on*)

Enable or disable LUT functionality on the camera.

See also

fc2GetLUTInfo()
fc2GetLUTChannel()
fc2SetLUTChannel()

Parameters

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

Returns

A fc2Error indicating the success or failure of the function.

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

Get the LUT bank that is currently being used.

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

Parameters

context	The fc2Context to be used.
pActiveBank	The currently active bank.

Returns

A fc2Error indicating the success or failure of the function.

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

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

Parameters

context	The fc2Context to be used.
bank	The bank to query.
pReadSupported	Whether reading from the bank is supported.
pWriteSupported	Whether writing to the bank is supported.

Returns

A fc2Error indicating the success or failure of the function.

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

Get the LUT channel settings from the camera.

See also

fc2GetLUTInfo()
fc2EnableLUT()
fc2SetLUTChannel()

Parameters

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.

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

Query if LUT support is available on the camera.

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

See also

fc2EnableLUT()
fc2GetLUTChannel()
fc2SetLUTChannel()

Parameters

context	The fc2Context to be used.
pData	The LUT structure to be filled.

Returns

A fc2Error indicating the success or failure of the function.

6.7.2.6 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.

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

Set the LUT channel settings to the camera.

See also

fc2GetLUTInfo()
fc2EnableLUT()
fc2GetLUTChannel()

Parameters

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.

6.8 Memory Channels

These functions deal with memory channel control on the camera.

Functions

- 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 specified current memory channel.
- FLYCAPTURE2_C_API fc2Error fc2RestoreFromMemoryChannel (fc2Context context, unsigned int channel)

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

6.8.1 Detailed Description

These functions deal with memory channel control on the camera.

6.8.2 Function Documentation

6.8.2.1 FLYCAPTURE2_C_API fc2Error fc2GetEmbeddedImageInfo (fc2Context *context*, fc2EmbeddedImageInfo * *pInfo*)

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

See also

fc2SetEmbeddedImageInfo()

Parameters

context	The fc2Context to be used.
pInfo	Structure to be filled.

Returns

A fc2Error indicating the success or failure of the function.

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

Retrieve the current memory channel from the camera.

See also

fc2SaveToMemoryChannel()
fc2RestoreFromMemoryChannel()
fc2GetMemoryChannelInfo()

Parameters

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

Returns

A fc2Error indicating the success or failure of the function.

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

Query the camera for memory channel support.

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

See also

fc2GetMemoryChannel()
fc2SaveToMemoryChannel()
fc2RestoreFromMemoryChannel()

Parameters

context	The fc2Context to be used.
pNum-Channels	Number of memory channels supported.

Returns

A fc2Error indicating the success or failure of the function.

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

Restore the specified current memory channel.

See also

fc2GetMemoryChannel()
fc2SaveToMemoryChannel()
fc2GetMemoryChannelInfo()

Parameters

context	The fc2Context to be used.
channel	Memory channel to restore from.

Returns

A fc2Error indicating the success or failure of the function.

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

Save the current settings to the specified current memory channel.

See also

fc2GetMemoryChannel()
fc2RestoreFromMemoryChannel()
fc2GetMemoryChannelInfo()

Parameters

context	The fc2Context to be used.
channel	Memory channel to save to.

Returns

A fc2Error indicating the success or failure of the function.

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

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

See also

fc2GetEmbeddedImageInfo()

Parameters

context	The fc2Context to be used.
pInfo	Structure to be used.

Returns

A fc2Error indicating the success or failure of the function.

6.9 Register Operation

These functions deal with register operation on the camera.

Functions

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

  Write to the specified register on the camera.
- FLYCAPTURE2_C_API fc2Error fc2ReadRegister (fc2Context context, unsigned int address, unsigned int ∗pValue)

  Read the specified register from the camera.
- FLYCAPTURE2_C_API fc2Error fc2WriteRegisterBroadcast (fc2Context context, unsigned int address, unsigned int value)

  Write to the specified register on the camera with broadcast.
- FLYCAPTURE2_C_API fc2Error fc2WriteRegisterBlock (fc2Context context, unsigned short addressHigh, unsigned int addressLow, const unsigned int ∗pBuffer, unsigned int length)

  Write to the specified register block on the camera.
- FLYCAPTURE2_C_API fc2Error fc2ReadRegisterBlock (fc2Context context, unsigned short addressHigh, unsigned int addressLow, unsigned int ∗pBuffer, unsigned int length)

  Write to the specified register block on the camera.
- FLYCAPTURE2_C_API const char ∗ fc2GetRegisterString (unsigned int registerVal)

  Returns a text representation of the register value.

6.9.1 Detailed Description

These functions deal with register operation on the camera.

6.9.2 Function Documentation

6.9.2.1 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.

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

Read the specified register from the camera.

See also

fc2WriteRegister()

Parameters

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.

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

Write to the specified register block on the camera.

See also

fc2WriteRegisterBlock()

Parameters

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.

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

Write to the specified register on the camera.

See also

fc2ReadRegister()

Parameters

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.

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

Write to the specified register block on the camera.

See also

fc2ReadRegisterBlock()

Parameters

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.

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

Write to the specified register on the camera with broadcast.

See also

fc2ReadRegisterBlock()

Parameters

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.

6.10 DCAM Formats

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

Functions

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

  Query the camera to determine if the specified video mode and frame rate is supported.
- FLYCAPTURE2_C_API fc2Error fc2GetVideoModeAndFrameRate (fc2Context context, fc2VideoMode *videoMode, fc2FrameRate *frameRate)

  Get the current video mode and frame rate from the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetVideoModeAndFrameRate (fc2Context context, fc2VideoMode videoMode, fc2FrameRate frameRate)

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

6.10.1 Detailed Description

These functions deal with DCAM video mode and frame rate on the camera. This is only used for firewire and usb2 cameras.

6.10.2 Function Documentation

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

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

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

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

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

Parameters

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.

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

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

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

Parameters

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.

6.11 Format7

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

Functions

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

  Retrieve the availability of Format7 custom image mode and the camera capabilities for the specified Format7 mode.
- FLYCAPTURE2_C_API fc2Error fc2ValidateFormat7Settings (fc2Context context, fc2Format7ImageSettings *imageSettings, BOOL *settingsAreValid, fc2Format7PacketInfo *packetInfo)

  Validates Format7ImageSettings structure and returns valid packet size information if the image settings are valid.
- FLYCAPTURE2_C_API fc2Error fc2GetFormat7Configuration (fc2Context context, fc2Format7ImageSettings *imageSettings, unsigned int *packetSize, float *percentage)

  Get the current Format7 configuration from the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetFormat7ConfigurationPacket (fc2Context context, fc2Format7ImageSettings *imageSettings, unsigned int packetSize)

  Set the current Format7 configuration to the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetFormat7Configuration (fc2Context context, fc2Format7ImageSettings *imageSettings, float percentSpeed)

  Set the current Format7 configuration to the camera.

6.11.1 Detailed Description

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

6.11.2 Function Documentation

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

Get the current Format7 configuration from the camera.

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

Parameters

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.

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

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

The mode must be specified in the Format7Info structure in order for the function to succeed.

Parameters

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.

6.11.2.3 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.
image-Settings	Image settings to be written to the camera.
percent-Speed	Packet size as a percentage to be written to the camera.

Returns

A fc2Error indicating the success or failure of the function.

6.11.2.4 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.

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

Validates Format7ImageSettings structure and returns valid packet size information if the image settings are valid.

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

Parameters

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.

6.12 GVCP Register Operation

These functions deal with GVCP register operation on the camera.

Functions

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

  Write a GVCP register.
- FLYCAPTURE2_C_API fc2Error fc2WriteGVCPRegisterBroadcast (fc2Context context, unsigned int address, unsigned int value)

  Write a GVCP register with broadcast.
- FLYCAPTURE2_C_API fc2Error fc2ReadGVCPRegister (fc2Context context, unsigned int address, unsigned int ∗pValue)

  Read a GVCP register.
- FLYCAPTURE2_C_API fc2Error fc2WriteGVCPRegisterBlock (fc2Context context, unsigned int address, const unsigned int ∗pBuffer, unsigned int length)

  Write a GVCP register block.
- FLYCAPTURE2_C_API fc2Error fc2ReadGVCPRegisterBlock (fc2Context context, unsigned int address, unsigned int ∗pBuffer, unsigned int length)

  Read a GVCP register block.
- FLYCAPTURE2_C_API fc2Error fc2WriteGVCPMemory (fc2Context context, unsigned int address, const unsigned char ∗pBuffer, unsigned int length)

  Write a GVCP memory block.
- FLYCAPTURE2_C_API fc2Error fc2ReadGVCPMemory (fc2Context context, unsigned int address, unsigned char ∗pBuffer, unsigned int length)

  Read a GVCP memory block.

6.12.1 Detailed Description

These functions deal with GVCP register operation on the camera.

6.12.2 Function Documentation

6.12.2.1 FLYCAPTURE2_C_API fc2Error fc2ReadGVCPMemory (fc2Context *context*, unsigned int *address*, unsigned char ∗ *pBuffer*, unsigned int *length*)

Read a GVCP memory block.

Parameters

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.

6.12.2.2 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.

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

Read a GVCP register block.

Parameters

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.

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

Write a GVCP memory block.

Parameters

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.

6.12.2.5 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.

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

Write a GVCP register block.

Parameters

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.

6.12.2.7 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.

6.13 GigE property manipulation

These functions deal with GigE properties.

Functions

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

  Get the specified GigEProperty.
- FLYCAPTURE2_C_API fc2Error fc2SetGigEProperty (fc2Context context, const fc2GigEProperty *pGigEProp)

  Set the specified GigEProperty.
- FLYCAPTURE2_C_API fc2Error fc2DiscoverGigEPacketSize (fc2Context context, unsigned int *packetSize)

  Discover the largest packet size that works for the network link between the PC and the camera.

6.13.1 Detailed Description

These functions deal with GigE properties.

6.13.2 Function Documentation

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

Discover the largest packet size that works for the network link between the PC and the camera.

This is useful in cases where there may be multiple links between the PC and the camera and there is a possibility of a component not supporting the recommended jumbo frame packet size of 9000.

Parameters

context	The fc2Context to be used.
packetSize	The maximum packet size supported by the link.

Returns

An Error indicating the success or failure of the function.

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

Get the specified GigEProperty.

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

Parameters

context	The fc2Context to be used.
pGigEProp	The GigE property to get.

Returns

An Error indicating the success or failure of the function.

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

Set the specified GigEProperty.

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

Parameters

context	The fc2Context to be used.
pGigEProp	The GigE property to set.

Returns

An Error indicating the success or failure of the function.

6.14 GigE image settings

These functions deal with GigE image setting.

Functions

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

  Check if the particular imaging mode is supported by the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetGigEImagingMode (fc2Context context, fc2Mode *mode)

  Get the current imaging mode on the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetGigEImagingMode (fc2Context context, fc2Mode mode)

  Set the current imaging mode to the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetGigEImageSettingsInfo (fc2Context context, fc2GigEImageSettingsInfo *pInfo)

  Get information about the image settings possible on the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetGigEImageSettings (fc2Context context, fc2GigEImageSettings *pImageSettings)

  Get the current image settings on the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetGigEImageSettings (fc2Context context, const fc2GigEImageSettings *pImageSettings)

  Set the image settings specified to the camera.

6.14.1 Detailed Description

These functions deal with GigE image setting.

6.14.2 Function Documentation

6.14.2.1 FLYCAPTURE2_C_API fc2Error fc2GetGigEImageSettings (fc2Context *context*, fc2GigEImageSettings * *pImageSettings*)

Get the current image settings on the camera.

Parameters

context	The fc2Context to be used.
pImage-Settings	Current image settings on camera.

Returns

An Error indicating the success or failure of the function.

6.14.2.2 FLYCAPTURE2_C_API fc2Error fc2GetGigEImageSettingsInfo (fc2Context *context*, fc2GigEImageSettingsInfo * *pInfo*)

Get information about the image settings possible on the camera.

Parameters

context	The fc2Context to be used.
pInfo	Image settings information.

Returns

An Error indicating the success or failure of the function.

6.14.2.3 FLYCAPTURE2_C_API fc2Error fc2GetGigEImagingMode (fc2Context *context*, fc2Mode * *mode*)

Get the current imaging mode on the camera.

Parameters

context	The fc2Context to be used.
mode	Current imaging mode on the camera.

Returns

An Error indicating the success or failure of the function.

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

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

Parameters

context	The fc2Context to be used.
mode	The mode to check.
isSupported	Whether the mode is supported.

Returns

An Error indicating the success or failure of the function.

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

Set the image settings specified to the camera.

Parameters

context	The fc2Context to be used.
pImage-Settings	Image settings to set to camera.

Returns

An Error indicating the success or failure of the function.

6.14.2.6 FLYCAPTURE2_C_API fc2Error fc2SetGigEImagingMode (fc2Context *context*, fc2Mode *mode*)

Set the current imaging mode to the camera.

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

Parameters

context	The fc2Context to be used.
mode	Imaging mode to set to the camera.

Returns

An Error indicating the success or failure of the function.

6.15 GigE image binning settings

These functions deal with GigE image binning settings.

Functions

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

  Get the current binning settings on the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetGigEImageBinningSettings (fc2Context context, unsigned int horzBinnningValue, unsigned int vertBinnningValue)

  Set the specified binning values to the camera.

6.15.1 Detailed Description

These functions deal with GigE image binning settings.

6.15.2 Function Documentation

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

Get the current binning settings on the camera.

Parameters

context	The fc2Context to be used.
horz-Binnning-Value	Current horizontal binning value.
vert-Binnning-Value	Current vertical binning value.

Returns

An Error indicating the success or failure of the function.

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

Set the specified binning values to the camera.

It is recommended that GetGigEImageSettingsInfo() be called after this function succeeds to retrieve the new image settings information for the new binning mode.

Parameters

context	The fc2Context to be used.
horz-Binnning-Value	Horizontal binning value.
vert-Binnning-Value	Vertical binning value.

Returns

An Error indicating the success or failure of the function.

6.16 GigE image stream configuration

These functions deal with GigE image stream configuration.

Functions

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

  Get the number of stream channels present on the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetGigEStreamChannelInfo (fc2Context context, unsigned int channel, fc2GigEStreamChannel *pChannel)

  Get the stream channel information for the specified channel.
- FLYCAPTURE2_C_API fc2Error fc2SetGigEStreamChannelInfo (fc2Context context, unsigned int channel, fc2GigEStreamChannel *pChannel)

  Set the stream channel information for the specified channel.
- FLYCAPTURE2_C_API fc2Error fc2GetGigEConfig (fc2Context context, fc2GigEConfig *pConfig)

  Get the current gige config on the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetGigEConfig (fc2Context context, const fc2GigEConfig *pConfig)

  Set the gige config specified to the camera.

6.16.1 Detailed Description

These functions deal with GigE image stream configuration.

6.16.2 Function Documentation

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

Get the current gige config on the camera.

Parameters

context	The fc2Context to be used.
pGigEConfig	Current configuration on camera.

Returns

An Error indicating the success or failure of the function.

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

Get the stream channel information for the specified channel.

Parameters

context	The fc2Context to be used.
channel	Channel number to use.
pChannel	Stream channel information for the specified channel.

Returns

An Error indicating the success or failure of the function.

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

Get the number of stream channels present on the camera.

Parameters

context	The fc2Context to be used.
numChannels	Number of stream channels present.

Returns

An Error indicating the success or failure of the function.

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

Set the gige config specified to the camera.

Parameters

context	The fc2Context to be used.
pGigEConfig	configuration to set to camera.

Returns

An Error indicating the success or failure of the function.

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

Set the stream channel information for the specified channel.

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

Parameters

context	The fc2Context to be used.
channel	Channel number to use.
pChannel	Stream channel information to use for the specified channel.

Returns

An Error indicating the success or failure of the function.

6.17 Image Operation

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

Functions

- FLYCAPTURE2_C_API fc2Error fc2SetDefaultColorProcessing (fc2ColorProcessingAlgorithm 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 fc2CreateImage (fc2Image *pImage)

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

  Destroy the fc2Image.
- FLYCAPTURE2_C_API fc2Error fc2SetImageDimensions (fc2Image *pImage, unsigned int rows, unsigned int cols, unsigned int stride, fc2PixelFormat pixelFormat, fc2BayerTileFormat bayerFormat)

  Sets the dimensions of the image object.
- FLYCAPTURE2_C_API fc2Error fc2GetImageDimensions (fc2Image *pImage, unsigned int *pRows, unsigned int *pCols, unsigned int *pStride, fc2PixelFormat *pPixelFormat, fc2BayerTileFormat *pBayerFormat)

  Get the image dimensions associated with the image object.
- FLYCAPTURE2_C_API fc2Error fc2SetImageColorProcessing (fc2Image *pImage, fc2ColorProcessingAlgorithm colorProc)

  Set the color processing algorithm.
- FLYCAPTURE2_C_API fc2Error fc2GetImageColorProcessing (fc2Image *pImage, fc2ColorProcessingAlgorithm *pColorProc)

  Get the current color processing algorithm.
- 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 fc2GetImageData (fc2Image *pImage, unsigned char **ppData)

Get a pointer to the data associated with the image.

- FLYCAPTURE2_C_API fc2Error fc2GetImageMetadata (fc2Image *pImage, fc2ImageMetadata *pImageMetaData)

  Get the metadata associated with the image.
- FLYCAPTURE2_C_API fc2TimeStamp fc2GetImageTimeStamp (fc2Image *pImage)

  Get the timestamp data associated with the image.
- FLYCAPTURE2_C_API fc2Error fc2SaveImage (fc2Image *pImage, const char *pFilename, 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 fc2CalculateImageStatistics (fc2Image *pImage, fc2ImageStatisticsContext *pImageStatisticsContext)

  Calculate statistics associated with the image.

6.17.1 Detailed Description

The Image operations are used to retrieve images from a camera, convert between multiple pixel formats and save images to disk. Operations on images are not guaranteed to be thread safe. It is recommended that operations on images be protected by thread synchronization constructs such as mutexes.

6.17.2 Function Documentation

6.17.2.1 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.

6.17.2.2 FLYCAPTURE2_C_API fc2Error fc2ConvertImage (fc2Image * *pImageIn*, fc2Image * *pImageOut*)

Parameters

pImageIn	
pImageOut	

Returns

A fc2Error indicating the success or failure of the function.

6.17.2.3 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.

6.17.2.4 FLYCAPTURE2_C_API fc2Error fc2CreateImage (fc2Image * *pImage*)

Create a fc2Image.

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

See also

fc2SetImageData()

Parameters

pImage	Pointer to image to be created.

Returns

A fc2Error indicating the success or failure of the function.

6.17.2.5 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.

6.17.2.6 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.

6.17.2.7 FLYCAPTURE2_C_API fc2Error fc2GetDefaultColorProcessing (fc2ColorProcessingAlgorithm * *pDefaultMethod*)

Get the default color processing algorithm.

Parameters

pDefaultMethod	The default color processing algorithm.

Returns

A fc2Error indicating the success or failure of the function.

6.17.2.8 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.

6.17.2.9 FLYCAPTURE2_C_API fc2Error fc2GetImageColorProcessing (fc2Image * *pImage*, fc2ColorProcessingAlgorithm * *pColorProc*)

Get the current color processing algorithm.

Parameters

pImage	The fc2Image to be used.

See also

fc2SetColorProcessing()

Returns

The current color processing algorithm.

6.17.2.10 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.

6.17.2.11 FLYCAPTURE2_C_API fc2Error fc2GetImageDimensions (fc2Image * *pImage*, unsigned int * *pRows*, unsigned int * *pCols*, unsigned int * *pStride*, fc2PixelFormat * *pPixelFormat*, fc2BayerTileFormat * *pBayerFormat*)

Get the image dimensions associated with the image object.

Parameters

pImage	The fc2Image to be used.
pRows	Number of rows.
pCols	Number of columns.
pStride	The stride.
pPixel-Format	Pixel format.
pBayer-Format	Bayer tile format.

6.17.2.12 FLYCAPTURE2_C_API fc2Error fc2GetImageMetadata (fc2Image * *pImage*, fc2ImageMetadata * *pImageMetaData*)

Get the metadata associated with the image.

This includes embedded image information.

Parameters

pImage	The fc2Image to be used.

Returns

Metadata associated with the image.

6.17.2.13 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.

6.17.2.14 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.

6.17.2.15 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.

6.17.2.16 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.

6.17.2.17 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.

6.17.2.18 FLYCAPTURE2_C_API fc2Error fc2SetImageColorProcessing (fc2Image * *pImage*, fc2ColorProcessingAlgorithm *colorProc*)

Set the color processing algorithm.

This should be set on the input image object.

Parameters

pImage	The fc2Image to be used.
colorProc	The color processing algorithm to use.

See also

fc2GetColorProcessing()

Returns

An Error indicating the success or failure of the function.

6.17.2.19 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.

6.17.2.20 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.

6.18 Image Statistics Operation

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

Functions

- FLYCAPTURE2_C_API fc2Error fc2CreateImageStatistics (fc2ImageStatisticsContext *pImageStatisticsContext)

  Create a statistics context.
- FLYCAPTURE2_C_API fc2Error fc2DestroyImageStatistics (fc2ImageStatisticsContext imageStatisticsContext)

  Destroy a statistics context.
- FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableAll (fc2ImageStatisticsContext imageStatisticsContext)

  Enable all channels.
- FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsDisableAll (fc2ImageStatisticsContext imageStatisticsContext)

  Disable all channels.
- FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableGreyOnly (fc2ImageStatisticsContext imageStatisticsContext)

  Enable only the grey channel.
- FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableRGBOnly (fc2ImageStatisticsContext imageStatisticsContext)

  Enable only the RGB channels.
- FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableHSLOnly (fc2ImageStatisticsContext imageStatisticsContext)

  Enable only the HSL channels.
- FLYCAPTURE2_C_API fc2Error fc2GetChannelStatus (fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, BOOL *pEnabled)

  Get the status of a statistics channel.
- FLYCAPTURE2_C_API fc2Error fc2SetChannelStatus (fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, BOOL enabled)

  Set the status of a statistics channel.
- FLYCAPTURE2_C_API fc2Error fc2GetChannelRange (fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, unsigned int *pMin, unsigned int *pMax)

  Get the range of a statistics channel.
- FLYCAPTURE2_C_API fc2Error fc2GetChannelPixelValueRange (fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, unsigned int *pPixelValueMin, unsigned int *pPixelValueMax)

  Get the range of a statistics channel.
- FLYCAPTURE2_C_API fc2Error fc2GetChannelNumPixelValues (fc2ImageStatisticsContext imageStatisticsContext, fc2StatisticsChannel channel, unsigned int *pNumPixelValues)

Get the number of unique pixel values in the image.

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

  Get the mean of the image.

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

  Get the histogram for the image.

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

  Get all statistics for the image.

6.18.1 Detailed Description

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

6.18.2 Function Documentation

6.18.2.1 FLYCAPTURE2_C_API fc2Error fc2CreateImageStatistics (fc2ImageStatisticsContext * *pImageStatisticsContext*)

Create a statistics context.

Parameters

pImageStatisticsContext	A statistics context.

Returns

A fc2Error indicating the success or failure of the function.

6.18.2.2 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.

6.18.2.3 FLYCAPTURE2_C_API fc2Error fc2GetChannelHistogram (fc2ImageStatisticsContext *imageStatisticsContext,* fc2StatisticsChannel *channel,* int ∗∗ *ppHistogram*)

Get the histogram for the image.

Parameters

imageStatisticsContext	A statistics context.
channel	The statistics channel.
ppHistogram	Pointer to an array containing the histogram.

Returns

An Error indicating the success or failure of the function.

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

Get the mean of the image.

Parameters

imageStatisticsContext	A statistics context.
channel	The statistics channel.
pPixelValueMean	The mean of the image.

Returns

An Error indicating the success or failure of the function.

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

Get the number of unique pixel values in the image.

Parameters

imageStatisticsContext	A statistics context.
channel	The statistics channel.
pNumPixelValues	The number of unique pixel values.

Returns

An Error indicating the success or failure of the function.

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

Get the range of a statistics channel.

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

Parameters

imageStatisticsContext	A statistics context.
channel	The statistics channel.
pPixelValueMin	The minimum pixel value.
pPixelValueMax	The maximum pixel value.

Returns

An Error indicating the success or failure of the function.

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

Get the range of a statistics channel.

The values returned are the maximum possible values for any given pixel in the image. This is generally 0-255 for 8 bit images, and 0-65535 for 16 bit images.

Parameters

imageStatisticsContext	A statistics context.
channel	The statistics channel.
pMin	The minimum possible value.
pMax	The maximum possible value.

Returns

An Error indicating the success or failure of the function.

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

Get the status of a statistics channel.

See also

fc2SetChannelStatus()

Parameters

imageStatisticsContext	A statistics context.
channel	The statistics channel.
pEnabled	Whether the channel is enabled.

Returns

An Error indicating the success or failure of the function.

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

Get all statistics for the image.

Parameters

imageStatisticsContext	The statistics context.

channel	The statistics channel.
pRangeMin	The minimum possible value.
pRangeMax	The maximum possible value.
pPixelValue-Min	The minimum pixel value.
pPixelValue-Max	The maximum pixel value.
pNumPixel-Values	The number of unique pixel values.
pPixelValue-Mean	The mean of the image.
ppHistogram	Pointer to an array containing the histogram.

Returns

A fc2Error indicating the success or failure of the function.

6.18.2.10 FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsDisableAll (fc2ImageStatisticsContext *imageStatisticsContext*)

Disable all channels.

Parameters

image-Statistics-Context	A statistics context.

Returns

An Error indicating the success or failure of the function.

6.18.2.11 FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableAll (fc2ImageStatisticsContext *imageStatisticsContext*)

Enable all channels.

Parameters

image-Statistics-Context	A statistics context.

Returns

An Error indicating the success or failure of the function.

6.18.2.12 FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableGreyOnly (fc2ImageStatisticsContext *imageStatisticsContext*)

Enable only the grey channel.

Parameters

imageStatisticsContext	A statistics context.

Returns

An Error indicating the success or failure of the function.

6.18.2.13 FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableHSLOnly (fc2ImageStatisticsContext *imageStatisticsContext*)

Enable only the HSL channels.

Parameters

imageStatisticsContext	A statistics context.

Returns

An Error indicating the success or failure of the function.

6.18.2.14 FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableRGBOnly (fc2ImageStatisticsContext *imageStatisticsContext*)

Enable only the RGB channels.

Parameters

imageStatisticsContext	A statistics context.

Returns

An Error indicating the success or failure of the function.

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

Set the status of a statistics channel.

See also

fc2GetChannelStatus()

Parameters

imageStatisticsContext	A statistics context.
channel	The statistics channel.
enabled	Whether the channel should be enabled.

Returns

An Error indicating the success or failure of the function.

6.19 AVI Recording Operation

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

Functions

- FLYCAPTURE2_C_API fc2Error fc2CreateAVI (fc2AVIContext *pAVIContext)

  Create a AVI context.
- FLYCAPTURE2_C_API fc2Error fc2AVIOpen (fc2AVIContext AVIContext, const char *pFileName, 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 video 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 fc2AVISetMaximumSize (fc2AVIContext AVIContext, unsigned int size)

  Set the maximum file size (in megabytes) of a AVI/MP4 file.
- FLYCAPTURE2_C_API fc2Error fc2AVIClose (fc2AVIContext AVIContext)

  Close the AVI file.
- FLYCAPTURE2_C_API fc2Error fc2DestroyAVI (fc2AVIContext AVIContext)

  Destroy a AVI context.

6.19.1 Detailed Description

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

6.19.2 Function Documentation

6.19.2.1 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.

6.19.2.2 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.

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

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

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

Parameters

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

See also

SetMaximumAVISize()
fc2AVIClose()
fc2AVIOption

Returns

A fc2Error indicating the success or failure of the function.

6.19.2.4 FLYCAPTURE2_C_API fc2Error fc2AVISetMaximumSize (fc2AVIContext *AVIContext,* unsigned int *size*)

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

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

Parameters

AVIContext	The AVI context to use.
size	The maximum AVI file size in MB.

Returns

A fc2Error indicating the success or failure of the function.

6.19.2.5 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.

6.19.2.6 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.

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

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

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

Parameters

pFileName	The filename of the video file.
pOption	H.264 options to apply to the video file.

See also

fc2AVIClose()
fc2H264Option

Returns

A fc2Error indicating the success or failure of the function.

6.19.2.8 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.

See also

fc2AVIClose()
fc2MJPGOption

Returns

A fc2Error indicating the success or failure of the function.

6.20 TopologyNode Operation

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

Functions

- FLYCAPTURE2_C_API fc2Error fc2CreateTopologyNode (fc2TopologyNodeContext *pTopologyNodeContext)

  Create a TopologyNode context.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetGuid (fc2TopologyNodeContext TopologyNodeContext, fc2PGRGuid *pGuid)

  Get the PGRGuid associated with the node.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetDeviceId (fc2TopologyNodeContext TopologyNodeContext, int *pID)

  Get the device ID associated with the node.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetNodeType (fc2TopologyNodeContext TopologyNodeContext, fc2NodeType *pNodeType)

  Get the node type associated with the node.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetInterfaceType (fc2TopologyNodeContext TopologyNodeContext, fc2InterfaceType *pInterfaceType)

  Get the interface type associated with the node.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetNumChildren (fc2TopologyNodeContext TopologyNodeContext, unsigned int *pNumChildNodes)

  Get the number of child nodes.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetChild (fc2TopologyNodeContext TopologyNodeContext, unsigned int position, fc2TopologyNodeContext *pChildTopologyNodeContext)

  Get child node located at the specified position.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeAddChild (fc2TopologyNodeContext TopologyNodeContext, fc2TopologyNodeContext TopologyNodeChildContext)

  Add the specified TopologyNode as a child of the node.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetNumPorts (fc2TopologyNodeContext TopologyNodeContext, unsigned int *pNumPorts)

  Get the number of ports.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetPortType (fc2TopologyNodeContext TopologyNodeContext, unsigned int position, fc2PortType *pPortType)

  Get type of port located at the specified position.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeAddPortType (fc2TopologyNodeContext TopologyNodeContext, fc2PortType portType)

  Add the specified PortType as a port of the node.

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

  Assign a PGRGuid and device ID to the node.
- FLYCAPTURE2_C_API BOOL fc2TopologyNodeAssignGuidToNodeEx (fc2TopologyNodeContext TopologyNodeContext, fc2PGRGuid guid, int deviceId, fc2NodeType nodeType)

  Assign a PGRGuid, device ID and nodeType to the node.
- FLYCAPTURE2_C_API fc2Error fc2DestroyTopologyNode (fc2TopologyNodeContext TopologyNodeContext)

  Destroy a TopologyNode context.

6.20.1 Detailed Description

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

6.20.2 Function Documentation

6.20.2.1 FLYCAPTURE2_C_API fc2Error fc2CreateTopologyNode (fc2TopologyNodeContext * *pTopologyNodeContext*)

Create a TopologyNode context.

Parameters

pTopologyNodeContext	A Topology Node context.

Returns

A fc2Error indicating the success or failure of the function.

6.20.2.2 FLYCAPTURE2_C_API fc2Error fc2DestroyTopologyNode (fc2TopologyNodeContext *TopologyNodeContext*)

Destroy a TopologyNode context.

Parameters

TopologyNodeContext	A Topology Node context.

Returns

A fc2Error indicating the success or failure of the function.

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

Add the specified TopologyNode as a child of the node.

Parameters

TopologyNodeContext	The Topology Node context to use.
TopologyNodeChildContext	The TopologyNode child context to add.

Returns

A fc2Error indicating the success or failure of the function.

6.20.2.4 FLYCAPTURE2_C_API fc2Error fc2TopologyNodeAddPortType (fc2TopologyNodeContext *TopologyNodeContext,* fc2PortType *portType*)

Add the specified PortType as a port of the node.

Parameters

TopologyNodeContext	The Topology Node context to use.
portType	childPort The port to add.

Returns

A fc2Error indicating the success or failure of the function.

6.20.2.5 FLYCAPTURE2_C_API BOOL fc2TopologyNodeAssignGuidToNode (fc2TopologyNodeContext *TopologyNodeContext,* fc2PGRGuid *guid,* int *deviceId*)

Assign a PGRGuid and device ID to the node.

Parameters

TopologyNodeContext	The Topology Node context to use.
guid	PGRGuid to be assigned.
deviceId	Device ID to be assigned.

Returns

A fc2Error indicating the success or failure of the function.

6.20.2.6 FLYCAPTURE2_C_API BOOL fc2TopologyNodeAssignGuidToNodeEx (fc2TopologyNodeContext *TopologyNodeContext,* fc2PGRGuid *guid,* int *deviceId,* fc2NodeType *nodeType*)

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

Parameters

TopologyNodeContext	The Topology Node context to use.
guid	PGRGuid to be assigned.
deviceId	Device ID to be assigned.
nodeType	NodeType to be assigned

Returns

A fc2Error indicating the success or failure of the function.

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

Get child node located at the specified position.

Parameters

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

Returns

A fc2Error indicating the success or failure of the function.

6.20.2.8 FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetDeviceId (fc2TopologyNodeContext *TopologyNodeContext,* int * *pID*)

Get the device ID associated with the node.

Parameters

TopologyNodeContext	The Topology Node context to use.
pID	Device ID of the node.

Returns

A fc2Error indicating the success or failure of the function.

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

Get the PGRGuid associated with the node.

Parameters

TopologyNodeContext	The Topology Node context to use.
pGuid	The unique identifier associated with the node.

Returns

A fc2Error indicating the success or failure of the function.

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

Get the interface type associated with the node.

Parameters

TopologyNodeContext	The Topology Node context to use.

pInterface-Type	Interface type of the node.

Returns

A fc2Error indicating the success or failure of the function.

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

Get the node type associated with the node.

Parameters

Topology-Node-Context	The Topology Node context to use.
pNodeType	Node type of the node.

Returns

A fc2Error indicating the success or failure of the function.

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

Get the number of child nodes.

Parameters

Topology-Node-Context	The Topology Node context to use.
pNumChild-Nodes	Number of child nodes.

Returns

A fc2Error indicating the success or failure of the function.

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

Get the number of ports.

Parameters

TopologyNodeContext	The Topology Node context to use.
pNumPorts	Number of ports.

Returns

A fc2Error indicating the success or failure of the function.

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

Get type of port located at the specified position.

Parameters

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

Returns

A fc2Error indicating the success or failure of the function.

6.21 Utilities

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

Functions

- FLYCAPTURE2_C_API fc2Error fc2CheckDriver (const fc2PGRGuid *pGuid)

  Check for driver compatibility for the given camera guid.
- FLYCAPTURE2_C_API fc2Error fc2GetDriverDeviceName (const fc2PGRGuid *pGuid, char *pDeviceName, size_t *deviceNameLength)

  Get the driver's name for a device.
- FLYCAPTURE2_C_API fc2Error fc2GetSystemInfo (fc2SystemInfo *pSystemInfo)

  Get system information.
- FLYCAPTURE2_C_API fc2Error fc2GetLibraryVersion (fc2Version *pVersion)

  Get library version.
- FLYCAPTURE2_C_API fc2Error fc2LaunchBrowser (const char *pAddress)

  Launch a URL in the system default browser.
- FLYCAPTURE2_C_API fc2Error fc2LaunchHelp (const char *pFileName)

  Open a CHM file in the system default CHM viewer.
- FLYCAPTURE2_C_API fc2Error fc2LaunchCommand (const char *pCommand)

  Execute a command in the terminal.
- FLYCAPTURE2_C_API fc2Error fc2LaunchCommandAsync (const char *pCommand, fc2AsyncCommandCallback pCallback, void *pUserData)

  Execute a command in the terminal.
- FLYCAPTURE2_C_API const char * fc2ErrorToDescription (fc2Error error)

  Get a string representation of an error.

6.21.1 Detailed Description

The utility operations are used to query for general system information such as operating system, available memory etc. It can also be used to launch browsers, CHM viewers or terminal commands.

6.21.2 Function Documentation

6.21.2.1 FLYCAPTURE2_C_API fc2Error fc2CheckDriver (const fc2PGRGuid * *pGuid*)

Check for driver compatibility for the given camera guid.

Parameters

pGuid	The PGRGuid of the device to check.

Returns

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

6.21.2.2 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.

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

Get the driver's name for a device.

Parameters

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

Returns

An Error indicating the success or failure of the function.

6.21.2.4 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.

6.21.2.5 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.

6.21.2.6 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.

6.21.2.7 FLYCAPTURE2_C_API fc2Error fc2LaunchCommand (const char * *pCommand*)

Execute a command in the terminal.

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

Parameters

pCommand	Command to execute.

Returns

A fc2Error indicating the success or failure of the function.

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

Execute a command in the terminal.

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

Parameters

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.

6.21.2.9 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.

6.22 TypeDefs

Data Structures

- struct fc2PGRGuid

  A GUID to the camera.

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 * fc2TopologyNodeContext

  A context referring to the TopologyNode object.

6.22.1 Define Documentation

6.22.1.1 #define FALSE 0

6.22.1.2 #define FULL_32BIT_VALUE 0x7FFFFFFF

6.22.1.3 #define MAX_STRING_LENGTH 512

6.22.1.4 #define TRUE 1

6.22.2 Typedef Documentation

6.22.2.1 typedef int BOOL

6.22.2.2 typedef void* fc2AVIContext

A context referring to the AVI recorder object.

6.22.2.3 typedef void* fc2Context

A context to the FlyCapture2 C library.

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

6.22.2.4 typedef void* fc2GuiContext

A context to the FlyCapture2 C GUI library.

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

6.22.2.5 typedef void* fc2ImageImpl

An internal pointer used in the fc2Image structure.

6.22.2.6 typedef void* fc2ImageStatisticsContext

A context referring to the ImageStatistics object.

6.22.2.7 typedef void* fc2TopologyNodeContext

A context referring to the TopologyNode object.

6.23 Enumerations

Enumerations

- enum fc2Error { FC2_ERROR_UNDEFINED = -1, FC2_ERROR_OK, FC2_ERROR_FAILED, FC2_ERROR_NOT_IMPLEMENTED, FC2_ERROR_FAILED_BUS_MASTER_CONNECTION, FC2_ERROR_NOT_CONNECTED, FC2_ERROR_INIT_FAILED, FC2_ERROR_NOT_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_INCOMPATIBLE_DRIVER, FC2_ERROR_FORCE_32BITS = FULL_32BIT_VALUE }

  The error types returned by functions.

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

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

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

  The grab strategy employed during image transfer.

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

  Timeout options for grabbing images.

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

  Bandwidth allocation options for 1394 devices.

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

  Interfaces that a camera may use to communicate with a host.
- enum fc2PropertyType { FC2_BRIGHTNESS, FC2_AUTO_EXPOSURE, FC2_SHARPNESS, FC2_WHITE_BALANCE, FC2_HUE, FC2_SATURATION, FC2_GAMMA, FC2_IRIS, FC2_FOCUS, FC2_ZOOM, FC2_PAN, FC2_TILT, FC2_SHUTTER, FC2_GAIN, FC2_TRIGGER_MODE, FC2_TRIGGER_DELAY, FC2_FRAME_RATE, FC2_TEMPERATURE, FC2_UNSPECIFIED_PROPERTY_TYPE, FC2_PROPERTY_TYPE_FORCE_32BITS = FULL_32BIT_VALUE }

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

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

  DCAM video modes.
- enum fc2Mode { FC2_MODE_0 = 0, FC2_MODE_1, FC2_MODE_2, FC2_MODE_3, FC2_MODE_4, FC2_MODE_5, FC2_MODE_6, FC2_MODE_7, FC2_MODE_8, FC2_MODE_9, FC2_MODE_10, FC2_MODE_11, FC2_MODE_12, FC2_MODE_13, FC2_MODE_14, FC2_MODE_15, FC2_MODE_16, FC2_MODE_17, FC2_MODE_18, FC2_MODE_19, FC2_MODE_20, FC2_MODE_21, FC2_MODE_22, FC2_MODE_23, FC2_MODE_24, FC2_MODE_25, FC2_MODE_26, FC2_MODE_27, FC2_MODE_28, FC2_MODE_29, FC2_MODE_30, FC2_MODE_31, FC2_NUM_MODES, FC2_MODE_FORCE_32BITS = FULL_32BIT_VALUE }

  Camera modes for DCAM formats as well as Format7.
- enum fc2PixelFormat { FC2_PIXEL_FORMAT_MONO8 = 0x80000000, FC2_PIXEL_FORMAT_411YUV8 = 0x40000000, FC2_PIXEL_FORMAT_422YUV8 = 0x20000000, FC2_PIXEL_FORMAT_444YUV8 = 0x10000000, FC2_PIXEL_FORMAT_RGB8 = 0x08000000, FC2_PIXEL_FORMAT_MONO16 = 0x04000000, FC2_PIXEL_FORMAT_RGB16 = 0x02000000, FC2_PIXEL_FORMAT_S_MONO16 = 0x01000000, FC2_PIXEL_FORMAT_S_RGB16 =

0x00800000, FC2_PIXEL_FORMAT_RAW8 = 0x00400000, FC2_PIXEL_FORMAT_RAW16 = 0x00200000, FC2_PIXEL_FORMAT_MONO12 = 0x00100000, FC2_PIXEL_FORMAT_RAW12 = 0x00080000, FC2_PIXEL_FORMAT_BGR = 0x80000008, FC2_PIXEL_FORMAT_BGRU = 0x40000008, FC2_PIXEL_FORMAT_RGB = FC2_PIXEL_FORMAT_RGB8, FC2_PIXEL_FORMAT_RGBU = 0x40000002, FC2_PIXEL_FORMAT_BGR16 = 0x02000001, FC2_PIXEL_FORMAT_BGRU16 = 0x02000002, FC2_PIXEL_FORMAT_422YUV8_JPEG = 0x40000001, FC2_NUM_PIXEL_FORMATS = 20, FC2_UNSPECIFIED_PIXEL_FORMAT = 0 }

Pixel formats available for Format7 modes.

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

  Bus speeds.
- enum 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 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_GIGE_LWF, FC2_DRIVER_UNKNOWN = -1, FC2_DRIVER_FORCE_32BITS = FULL_32BIT_VALUE }

  Types of low level drivers that FlyCapture uses.
- enum fc2ColorProcessingAlgorithm { FC2_DEFAULT, FC2_NO_COLOR_PROCESSING, FC2_NEAREST_NEIGHBOR_FAST, FC2_EDGE_SENSING, × FC2_HQ_LINEAR, FC2_RIGOROUS, FC2_IPP, FC2_DIRECTIONAL, FC2_WEIGHTED_DIRECTIONAL, FC2_COLOR_PROCESSING_ALGORITHM_FORCE_32BITS = FULL_32BIT_VALUE }

  Color processing algorithms.
- enum fc2BayerTileFormat { FC2_BT_NONE, FC2_BT_RGGB, FC2_BT_GRBG, FC2_BT_GBRG, FC2_BT_BGGR, FC2_BT_FORCE_32BITS = FULL_32BIT_VALUE }

  Bayer tile formats.
- enum fc2ImageFileFormat { FC2_FROM_FILE_EXT = -1, FC2_PGM, FC2_PPM, FC2_BMP, FC2_JPEG, FC2_JPEG2000, FC2_TIFF, FC2_PNG, FC2_RAW, FC2_IMAGE_FILE_FORMAT_FORCE_32BITS = FULL_32BIT_VALUE }

  File formats to be used for saving images to disk.

6.23.1 Enumeration Type Documentation

6.23.1.1 enum fc2BandwidthAllocation

Bandwidth allocation options for 1394 devices.

Enumerator:

FC2_BANDWIDTH_ALLOCATION_OFF Do not allocate bandwidth.

FC2_BANDWIDTH_ALLOCATION_ON Allocate bandwidth. This is the default setting.

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

FC2_BANDWIDTH_ALLOCATION_UNSPECIFIED Not specified. This leaves the current setting unchanged.

FC2_BANDWIDTH_ALLOCATION_FORCE_32BITS

6.23.1.2 enum fc2BayerTileFormat

Bayer tile formats.

Enumerator:

FC2_BT_NONE No bayer tile format.

FC2_BT_RGGB Red-Green-Green-Blue.

FC2_BT_GRBG Green-Red-Blue-Green.

FC2_BT_GBRG Green-Blue-Red-Green.

FC2_BT_BGGR Blue-Green-Green-Red.

FC2_BT_FORCE_32BITS

6.23.1.3 enum fc2BusCallbackType

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

Enumerator:

FC2_BUS_RESET Register for all bus events.

FC2_ARRIVAL Register for arrivals only.

FC2_REMOVAL Register for removals only.

FC2_CALLBACK_TYPE_FORCE_32BITS

6.23.1.4 enum fc2BusSpeed

Bus speeds.

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

6.23.1.5 enum fc2ColorProcessingAlgorithm

Color processing algorithms.

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

Enumerator:

FC2_DEFAULT Default method.

FC2_NO_COLOR_PROCESSING No color processing.

FC2_NEAREST_NEIGHBOR_FAST Fastest but lowest quality. Equivalent to FLYCAPTURE_NEAREST_NEIGHBOR_FAST in FlyCapture.

FC2_EDGE_SENSING Weights surrounding pixels based on localized edge orientation.

FC2_HQ_LINEAR Well-balanced speed and quality.

FC2_RIGOROUS Slowest but produces good results.

FC2_IPP Multithreaded with similar results to edge sensing.

FC2_DIRECTIONAL Best quality but much faster than rigorous.

FC2_WEIGHTED_DIRECTIONAL Weighted pixel average from different directions.

FC2_COLOR_PROCESSING_ALGORITHM_FORCE_32BITS

6.23.1.6 enum fc2DriverType

Types of low level drivers that FlyCapture uses.

Enumerator:

FC2_DRIVER_1394_CAM PGRCam.sys.

FC2_DRIVER_1394_PRO PGR1394.sys.

FC2_DRIVER_1394_JUJU firewire_core.

FC2_DRIVER_1394_VIDEO1394 video1394.

FC2_DRIVER_1394_RAW1394 raw1394.

FC2_DRIVER_USB_NONE No usb driver used just BSD stack. (Linux only)

FC2_DRIVER_USB_CAM PGRUsbCam.sys.

FC2_DRIVER_USB3_PRO 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_GIGE_LWF PgrLwf.sys.

FC2_DRIVER_UNKNOWN Unknown driver type.

FC2_DRIVER_FORCE_32BITS

6.23.1.7 enum fc2Error

The error types returned by functions.

Enumerator:

FC2_ERROR_UNDEFINED Undefined.

FC2_ERROR_OK Function returned with no errors.

FC2_ERROR_FAILED General failure.

FC2_ERROR_NOT_IMPLEMENTED Function has not been implemented.

FC2_ERROR_FAILED_BUS_MASTER_CONNECTION Could not connect to - Bus Master.

FC2_ERROR_NOT_CONNECTED Camera has not been connected.

FC2_ERROR_INIT_FAILED Initialization failed.

FC2_ERROR_NOT_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_INCOMPATIBLE_DRIVER The installed driver is not compatible with the library.

FC2_ERROR_FORCE_32BITS

6.23.1.8 enum fc2FrameRate

Frame rates in frames per second.

Enumerator:

FC2_FRAMERATE_1_875 1.875 fps.

FC2_FRAMERATE_3_75 3.75 fps.

FC2_FRAMERATE_7_5 7.5 fps.

FC2_FRAMERATE_15 15 fps.

FC2_FRAMERATE_30 30 fps.

FC2_FRAMERATE_60 60 fps.

FC2_FRAMERATE_120 120 fps.

FC2_FRAMERATE_240 240 fps.

FC2_FRAMERATE_FORMAT7 Custom frame rate for Format7 functionality.

FC2_NUM_FRAMERATES Number of possible camera frame rates.

FC2_FRAMERATE_FORCE_32BITS

6.23.1.9 enum fc2GrabMode

The grab strategy employed during image transfer.

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

Enumerator:

FC2_DROP_FRAMES Grabs the newest image in the user buffer each time the RetrieveBuffer() function is called. Older images are dropped instead of accumulating in the user buffer. Grabbing blocks if the camera has not finished transmitting the next available image. If the camera is transmitting images faster than the application can grab them, images may be dropped and only the most recent image is stored for grabbing. Note that this mode is the equivalent of flycaptureLockLatest in earlier versions of the FlyCapture SDK.

FC2_BUFFER_FRAMES Images accumulate in the user buffer, and the oldest image is grabbed for handling before being discarded. This member can be used to guarantee that each image is seen. However, image processing time must not exceed transmission time from the camera to the buffer. Grabbing

blocks if the camera has not finished transmitting the next available image. The buffer size is controlled by the numBuffers parameter in the FC2Config struct. Note that this mode is the equivalent of flycaptureLockNext in earlier versions of the FlyCapture SDK.

FC2_UNSPECIFIED_GRAB_MODE Unspecified grab mode.

FC2_GRAB_MODE_FORCE_32BITS

6.23.1.10 enum fc2GrabTimeout

Timeout options for grabbing images.

Enumerator:

FC2_TIMEOUT_NONE Non-blocking wait.

FC2_TIMEOUT_INFINITE Wait indefinitely.

FC2_TIMEOUT_UNSPECIFIED Unspecified timeout setting.

FC2_GRAB_TIMEOUT_FORCE_32BITS

6.23.1.11 enum fc2ImageFileFormat

File formats to be used for saving images to disk.

Enumerator:

FC2_FROM_FILE_EXT Determine file format from file extension.

FC2_PGM Portable gray map.

FC2_PPM Portable pixmap.

FC2_BMP Bitmap.

FC2_JPEG JPEG.

FC2_JPEG2000 JPEG 2000.

FC2_TIFF Tagged image file format.

FC2_PNG Portable network graphics.

FC2_RAW Raw data.

FC2_IMAGE_FILE_FORMAT_FORCE_32BITS

6.23.1.12 enum fc2InterfaceType

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

Enumerator:

FC2_INTERFACE_IEEE1394 IEEE-1394 (Includes 1394a and 1394b).

FC2_INTERFACE_USB_2 USB 2.0.

FC2_INTERFACE_USB_3 USB 3.0.

FC2_INTERFACE_GIGE GigE.

FC2_INTERFACE_UNKNOWN Unknown interface.

FC2_INTERFACE_TYPE_FORCE_32BITS

6.23.1.13 enum fc2Mode

Camera modes for DCAM formats as well as Format7.

Enumerator:

FC2_MODE_0

FC2_MODE_1

FC2_MODE_2

FC2_MODE_3

FC2_MODE_4

FC2_MODE_5

FC2_MODE_6

FC2_MODE_7

FC2_MODE_8

FC2_MODE_9

FC2_MODE_10

FC2_MODE_11

FC2_MODE_12

FC2_MODE_13

FC2_MODE_14

FC2_MODE_15

FC2_MODE_16

FC2_MODE_17

FC2_MODE_18

FC2_MODE_19

FC2_MODE_20

FC2_MODE_21

FC2_MODE_22

FC2_MODE_23

FC2_MODE_24

FC2_MODE_25

FC2_MODE_26

FC2_MODE_27

FC2_MODE_28

FC2_MODE_29

FC2_MODE_30

FC2_MODE_31

FC2_NUM_MODES Number of modes.

FC2_MODE_FORCE_32BITS

6.23.1.14 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.

6.23.1.15 enum fc2PixelFormat

Pixel formats available for Format7 modes.

Enumerator:

FC2_PIXEL_FORMAT_MONO8 8 bits of mono information.

FC2_PIXEL_FORMAT_411YUV8 YUV 4:1:1.

FC2_PIXEL_FORMAT_422YUV8 YUV 4:2:2.

FC2_PIXEL_FORMAT_444YUV8 YUV 4:4:4.

FC2_PIXEL_FORMAT_RGB8 R = G = B = 8 bits.

FC2_PIXEL_FORMAT_MONO16 16 bits of mono information.

FC2_PIXEL_FORMAT_RGB16 R = G = B = 16 bits.

FC2_PIXEL_FORMAT_S_MONO16 16 bits of signed mono information.

FC2_PIXEL_FORMAT_S_RGB16 R = G = B = 16 bits signed.

FC2_PIXEL_FORMAT_RAW8 8 bit raw data output of sensor.

FC2_PIXEL_FORMAT_RAW16 16 bit raw data output of sensor.

FC2_PIXEL_FORMAT_MONO12 12 bits of mono information.

FC2_PIXEL_FORMAT_RAW12 12 bit raw data output of sensor.

FC2_PIXEL_FORMAT_BGR 24 bit BGR.

FC2_PIXEL_FORMAT_BGRU 32 bit BGRU.

FC2_PIXEL_FORMAT_RGB 24 bit RGB.

FC2_PIXEL_FORMAT_RGBU 32 bit RGBU.

FC2_PIXEL_FORMAT_BGR16 R = G = B = 16 bits.

FC2_PIXEL_FORMAT_BGRU16 64 bit BGRU.

FC2_PIXEL_FORMAT_422YUV8_JPEG JPEG compressed stream.

FC2_NUM_PIXEL_FORMATS Number of pixel formats.

FC2_UNSPECIFIED_PIXEL_FORMAT Unspecified pixel format.

6.23.1.16 enum fc2PropertyType

Camera properties.

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

Enumerator:

FC2_BRIGHTNESS

FC2_AUTO_EXPOSURE

FC2_SHARPNESS

FC2_WHITE_BALANCE

FC2_HUE

FC2_SATURATION

FC2_GAMMA

FC2_IRIS

FC2_FOCUS

FC2_ZOOM

FC2_PAN

FC2_TILT

FC2_SHUTTER

FC2_GAIN

FC2_TRIGGER_MODE

FC2_TRIGGER_DELAY

FC2_FRAME_RATE

FC2_TEMPERATURE

FC2_UNSPECIFIED_PROPERTY_TYPE

FC2_PROPERTY_TYPE_FORCE_32BITS

6.23.1.17 enum fc2VideoMode

DCAM video modes.

Enumerator:

FC2_VIDEOMODE_160x120YUV444 160x120 YUV444.

FC2_VIDEOMODE_320x240YUV422 320x240 YUV422.

FC2_VIDEOMODE_640x480YUV411 640x480 YUV411.

FC2_VIDEOMODE_640x480YUV422 640x480 YUV422.

FC2_VIDEOMODE_640x480RGB 640x480 24-bit RGB.

FC2_VIDEOMODE_640x480Y8 640x480 8-bit.

FC2_VIDEOMODE_640x480Y16 640x480 16-bit.

FC2_VIDEOMODE_800x600YUV422 800x600 YUV422.

FC2_VIDEOMODE_800x600RGB 800x600 RGB.

FC2_VIDEOMODE_800x600Y8 800x600 8-bit.

FC2_VIDEOMODE_800x600Y16 800x600 16-bit.

FC2_VIDEOMODE_1024x768YUV422 1024x768 YUV422.

FC2_VIDEOMODE_1024x768RGB 1024x768 RGB.

FC2_VIDEOMODE_1024x768Y8 1024x768 8-bit.

FC2_VIDEOMODE_1024x768Y16 1024x768 16-bit.

FC2_VIDEOMODE_1280x960YUV422 1280x960 YUV422.

FC2_VIDEOMODE_1280x960RGB 1280x960 RGB.

FC2_VIDEOMODE_1280x960Y8 1280x960 8-bit.

FC2_VIDEOMODE_1280x960Y16 1280x960 16-bit.

FC2_VIDEOMODE_1600x1200YUV422 1600x1200 YUV422.

FC2_VIDEOMODE_1600x1200RGB 1600x1200 RGB.

FC2_VIDEOMODE_1600x1200Y8 1600x1200 8-bit.

FC2_VIDEOMODE_1600x1200Y16 1600x1200 16-bit.

FC2_VIDEOMODE_FORMAT7 Custom video mode for Format7 functionality.

FC2_NUM_VIDEOMODES Number of possible video modes.

FC2_VIDEOMODE_FORCE_32BITS

6.24 GigE specific enumerations

These enumerations are specific to GigE camera operation only.

Enumerations

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

  Possible properties that can be queried from the camera.

6.24.1 Detailed Description

These enumerations are specific to GigE camera operation only.

6.24.2 Enumeration Type Documentation

6.24.2.1 enum fc2GigEPropertyType

Possible properties that can be queried from the camera.

Enumerator:

FC2_HEARTBEAT

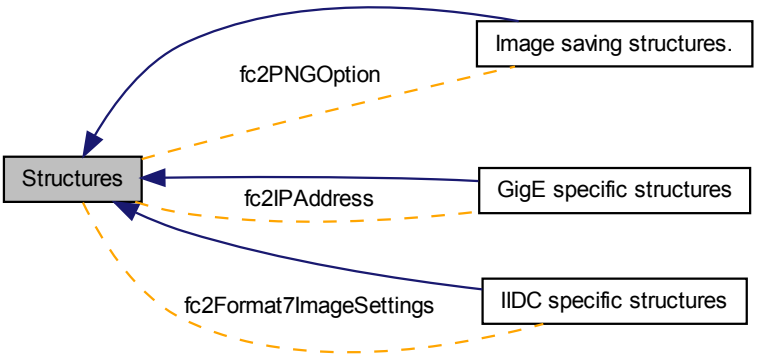
FC2_HEARTBEAT_TIMEOUT

PACKET_SIZE

PACKET_DELAY

6.25 Structures

Collaboration diagram for Structures:



Data Structures

- struct fc2Image
- struct fc2SystemInfo

  Description of the system.
- struct fc2Version

  The current version of the library.
- struct fc2IPAddress

  IPv4 address.
- struct fc2Format7ImageSettings

  Format 7 image settings.
- struct fc2Config

  Configuration for a camera.
- struct fc2TriggerDelayInfo

  Information about a specific camera property.
- struct fc2TriggerDelay

  A specific camera property.
- struct fc2TriggerModeInfo

  Information about a camera trigger property.
- struct fc2TriggerMode

  A camera trigger.
- struct fc2StrobeInfo

  A camera strobe property.

- struct fc2StrobeControl

  A camera strobe.
- struct fc2TimeStamp

  Timestamp information.
- struct fc2ConfigROM

  Camera configuration ROM.
- struct fc2CameraInfo

  Camera information.
- struct fc2EmbeddedImageInfoProperty

  Properties of a single embedded image info property.
- struct fc2EmbeddedImageInfo

  Properties of the possible embedded image information.
- struct fc2ImageMetadata

  Metadata related to an image.
- struct fc2LUTData

  Information about the camera's look up table.
- struct fc2CameraStats

  Camera diagnostic information.
- struct fc2PNGOption

  Options for saving PNG images.

Modules

- GigE specific structures

  These structures are specific to GigE camera operation only.
- IIDC specific structures

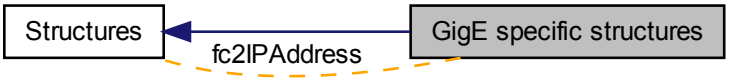
  These structures are specific to IIDC camera operation only.
- Image saving structures.

  These structures define various parameters used for saving images.

6.26 GigE specific structures

These structures are specific to GigE camera operation only.

Collaboration diagram for GigE specific structures:



Data Structures

- struct fc2IPAddress

  IPv4 address.
- struct fc2MACAddress

  MAC address.
- struct fc2GigEProperty

  A GigE property.
- struct fc2GigEStreamChannel

  Information about a single GigE stream channel.
- struct fc2GigEConfig

  Configuration for a GigE camera.
- struct fc2GigEImageSettingsInfo

  Format 7 information for a single mode.
- struct fc2GigEImageSettings

  Image settings for a GigE camera.

6.26.1 Detailed Description

These structures are specific to GigE camera operation only.

6.27 IIDC specific structures

These structures are specific to IIDC camera operation only.

Collaboration diagram for IIDC specific structures:

Data Structures

- struct fc2Format7ImageSettings

  Format 7 image settings.
- struct fc2Format7Info

  Format 7 information for a single mode.
- struct fc2Format7PacketInfo

  Format 7 packet information.

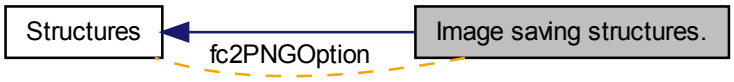
6.27.1 Detailed Description

These structures are specific to IIDC camera operation only.

6.28 Image saving structures.

These structures define various parameters used for saving images.

Collaboration diagram for Image saving structures.:



Data Structures

- struct fc2PNGOption

  Options for saving PNG images.
- struct fc2PPMOption

  Options for saving PPM images.
- struct fc2PGMOption

  Options for saving PGM images.
- struct fc2TIFFOption

  Options for saving TIFF images.
- struct fc2JPEGOption

  Options for saving JPEG image.
- struct fc2JPG2Option

  Options for saving JPEG2000 image.
- struct fc2BMPOption

  Options for saving Bitmap image.
- struct fc2MJPGOption

  Options for saving MJPG files.
- struct fc2H264Option

  Options for saving H264 files.
- struct fc2AVIOption

  Options for saving AVI files.
- struct fc2EventOptions

  Options for enabling device event registration.
- struct fc2EventCallbackData

Typedefs

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

Enumerations

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

6.28.1 Detailed Description

These structures define various parameters used for saving images.

6.28.2 Typedef Documentation

6.28.2.1 typedef void(* fc2AsyncCommandCallback)(fc2Error retError, void *pUserData)

6.28.2.2 typedef void(* fc2BusEventCallback)(void *pParameter, unsigned int serialNumber)

6.28.2.3 typedef void* fc2CallbackHandle

6.28.2.4 typedef void(* fc2CameraEventCallback)(void *pCallbackData)

6.28.2.5 typedef void(* fc2ImageEventCallback)(fc2Image *image, void *pCallbackData)

6.28.3 Enumeration Type Documentation

6.28.3.1 enum fc2TIFFCompressionMethod

Enumerator:

FC2_TIFF_NONE Save without any compression.

FC2_TIFF_PACKBITS Save using PACKBITS compression.

FC2_TIFF_DEFLATE Save using DEFLATE compression (ZLIB compression).

FC2_TIFF_ADOBE_DEFLATE Save using ADOBE DEFLATE compression.

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

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

FC2_TIFF_LZW Save using LZW compression.

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

Chapter 7

Data Structure Documentation

7.1 fc2AVIOption Struct Reference

Options for saving AVI files.

Data Fields

- float frameRate

  Frame rate of the stream.
- unsigned int reserved [256]

  Reserved for future use.

7.1.1 Detailed Description

Options for saving AVI files.

7.1.2 Field Documentation

7.1.2.1 float frameRate

Frame rate of the stream.

7.1.2.2 unsigned int reserved[256]

Reserved for future use.

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

- FlyCapture2Defs_C.h

7.2 fc2BMPOption Struct Reference

Options for saving Bitmap image.

Data Fields

- BOOL indexedColor_8bit
- unsigned int reserved [16]

  Reserved for future use.

7.2.1 Detailed Description

Options for saving Bitmap image.

7.2.2 Field Documentation

7.2.2.1 BOOL indexedColor_8bit

7.2.2.2 unsigned int reserved[16]

Reserved for future use.

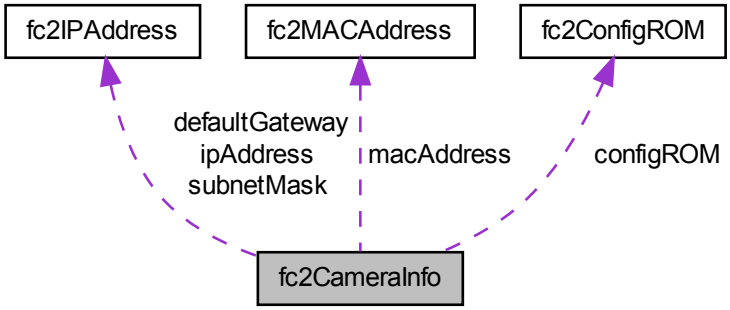
The documentation for this struct was generated from the following file:

- FlyCapture2Defs_C.h

7.3 fc2CameraInfo Struct Reference

Camera information.

Collaboration diagram for fc2CameraInfo:



Data Fields

- unsigned int serialNumber

  Device serial number.
- fc2InterfaceType interfaceType

  Interface type.
- fc2DriverType driverType

  Driver type.
- BOOL isColorCamera

  Flag indicating if this is a color camera.
- char modelName [MAX_STRING_LENGTH]

  Device model name.
- char vendorName [MAX_STRING_LENGTH]

  Device vendor name.
- char sensorInfo [MAX_STRING_LENGTH]

  String detailing the sensor information.
- char sensorResolution [MAX_STRING_LENGTH]

  String providing the sensor resolution.
- char driverName [MAX_STRING_LENGTH]

  Driver name of driver being used.
- char firmwareVersion [MAX_STRING_LENGTH]

  Firmware version of camera.
- char firmwareBuildTime [MAX_STRING_LENGTH]

  Firmware build time.
- fc2BusSpeed maximumBusSpeed

Maximum bus speed.

- fc2BayerTileFormat bayerTileFormat

  Bayer tile format.

- fc2PCIeBusSpeed pcieBusSpeed

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

- unsigned short nodeNumber

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

- unsigned short busNumber

  PCIe Bus Speed, set to PCIE_BUSSPEED_UNKNOWN for unsupported drivers.

- unsigned int reserved [16]

  Reserved for future use.

IIDC specific information

- unsigned int iidcVer

  DCAM version.

- fc2ConfigROM configROM

  Configuration ROM data.

GigE specific information

- unsigned int gigEMajorVersion

  GigE Vision version.

- unsigned int gigEMinorVersion

  GigE Vision minor version.

- char userDefinedName [MAX_STRING_LENGTH]

  User defined name.

- char xmlURL1 [MAX_STRING_LENGTH]

  XML URL 1.

- char xmlURL2 [MAX_STRING_LENGTH]

  XML URL 2.

- fc2MACAddress macAddress

  MAC address.

- fc2IPAddress ipAddress

  IP address.

- fc2IPAddress subnetMask

  Subnet mask.

- fc2IPAddress defaultGateway

  Default gateway.

- unsigned int ccpStatus

  Status/Content of CCP register.

- unsigned int applicationIPAddress

  Local Application IP Address.

- unsigned int applicationPort

  Local Application port.

7.3.1 Detailed Description

Camera information.

7.3.2 Field Documentation

7.3.2.1 unsigned int applicationIPAddress

Local Application IP Address.

7.3.2.2 unsigned int applicationPort

Local Application port.

7.3.2.3 fc2BayerTileFormat bayerTileFormat

Bayer tile format.

7.3.2.4 unsigned short busNumber

PCIe Bus Speed, set to PCIE_BUSSPEED_UNKNOWN for unsupported drivers.

7.3.2.5 unsigned int ccpStatus

Status/Content of CCP register.

7.3.2.6 fc2ConfigROM configROM

Configuration ROM data.

7.3.2.7 fc2IPAddress defaultGateway

Default gateway.

7.3.2.8 char driverName[MAX_STRING_LENGTH]

Driver name of driver being used.

7.3.2.9 fc2DriverType driverType

Driver type.

7.3.2.10 char firmwareBuildTime[MAX_STRING_LENGTH]

Firmware build time.

7.3.2.11 char firmwareVersion[MAX_STRING_LENGTH]

Firmware version of camera.

7.3.2.12 unsigned int gigEMajorVersion

GigE Vision version.

7.3.2.13 unsigned int gigEMinorVersion

GigE Vision minor version.

7.3.2.14 unsigned int iidcVer

DCAM version.

7.3.2.15 fc2InterfaceType interfaceType

Interface type.

7.3.2.16 fc2IPAddress ipAddress

IP address.

7.3.2.17 BOOL isColorCamera

Flag indicating if this is a color camera.

7.3.2.18 fc2MACAddress macAddress

MAC address.

7.3.2.19 fc2BusSpeed maximumBusSpeed

Maximum bus speed.

7.3.2.20 char modelName[MAX_STRING_LENGTH]

Device model name.

7.3.2.21 unsigned short nodeNumber

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

7.3.2.22 fc2PCIeBusSpeed pcieBusSpeed

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

7.3.2.23 unsigned int reserved[16]

Reserved for future use.

7.3.2.24 char sensorInfo[MAX_STRING_LENGTH]

String detailing the sensor information.

7.3.2.25 char sensorResolution[MAX_STRING_LENGTH]

String providing the sensor resolution.

7.3.2.26 unsigned int serialNumber

Device serial number.

7.3.2.27 fc2IPAddress subnetMask

Subnet mask.

7.3.2.28 char userDefinedName[MAX_STRING_LENGTH]

User defined name.

7.3.2.29 char vendorName[MAX_STRING_LENGTH]

Device vendor name.

7.3.2.30 char xmlURL1[MAX_STRING_LENGTH]

XML URL 1.

7.3.2.31 char xmlURL2[MAX_STRING_LENGTH]

XML URL 2.

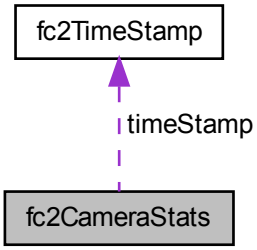
The documentation for this struct was generated from the following file:

- FlyCapture2Defs_C.h

7.4 fc2CameraStats Struct Reference

Camera diagnostic information.

Collaboration diagram for fc2CameraStats:



Data Fields

- unsigned int imageDropped
- unsigned int imageCorrupt
- unsigned int imageXmitFailed
- unsigned int imageDriverDropped
- unsigned int regReadFailed
- unsigned int regWriteFailed
- unsigned int portErrors
- BOOL cameraPowerUp
- float cameraVoltages [8]
- unsigned int numVoltages

  The number of voltage registers available.
- float cameraCurrents [8]
- unsigned int numCurrents

  The number of current registers available.

- unsigned int temperature
- unsigned int timeSinceInitialization
- unsigned int timeSinceBusReset
- fc2TimeStamp timeStamp
- unsigned int numResendPacketsRequested
- unsigned int numResendPacketsReceived
- unsigned int reserved [16]

  Reserved for future use.

7.4.1 Detailed Description

Camera diagnostic information.

7.4.2 Field Documentation

7.4.2.1 float cameraCurrents[8]

7.4.2.2 BOOL cameraPowerUp

7.4.2.3 float cameraVoltages[8]

7.4.2.4 unsigned int imageCorrupt

7.4.2.5 unsigned int imageDriverDropped

7.4.2.6 unsigned int imageDropped

7.4.2.7 unsigned int imageXmitFailed

7.4.2.8 unsigned int numCurrents

The number of current registers available.

0: the values in cameraCurrents[] are invalid.

7.4.2.9 unsigned int numResendPacketsReceived

7.4.2.10 unsigned int numResendPacketsRequested

7.4.2.11 unsigned int numVoltages

The number of voltage registers available.

0: the values in cameraVoltages[] are invalid.

7.4.2.12 unsigned int portErrors

7.4.2.13 unsigned int regReadFailed

7.4.2.14 unsigned int regWriteFailed

7.4.2.15 unsigned int reserved[16]

Reserved for future use.

7.4.2.16 unsigned int temperature

7.4.2.17 unsigned int timeSinceBusReset

7.4.2.18 unsigned int timeSinceInitialization

7.4.2.19 fc2TimeStamp timeStamp

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

- FlyCapture2Defs_C.h

7.5 fc2Config Struct Reference

Configuration for a camera.

Data Fields

- unsigned int numBuffers

  Number of buffers used by the FlyCapture2 library to grab images.
- unsigned int numImageNotifications

  Number of notifications per image.
- unsigned int minNumImageNotifications

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

  Time in milliseconds that RetrieveBuffer() and WaitForBufferEvent() will wait for an image before timing out and returning.
- fc2GrabMode grabMode

  Grab mode for the camera.
- BOOL highPerformanceRetrieveBuffer

  This parameter enables RetrieveBuffer to run in high performance mode.
- fc2BusSpeed isochBusSpeed

  Isochronous bus speed.
- fc2BusSpeed asyncBusSpeed

Asynchronous bus speed.

- fc2BandwidthAllocation bandwidthAllocation

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

- unsigned int registerTimeoutRetries

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

- unsigned int registerTimeout

  Register read/write timeout value, in microseconds.

- unsigned int reserved [16]

  Reserved for future use.

7.5.1 Detailed Description

Configuration for a camera.

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

7.5.2 Field Documentation

7.5.2.1 fc2BusSpeed asyncBusSpeed

Asynchronous bus speed.

7.5.2.2 fc2BandwidthAllocation bandwidthAllocation

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

7.5.2.3 fc2GrabMode grabMode

Grab mode for the camera.

The default is DROP_FRAMES.

7.5.2.4 int grabTimeout

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

7.5.2.5 BOOL highPerformanceRetrieveBuffer

This parameter enables RetrieveBuffer to run in high performance mode.

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

7.5.2.6 fc2BusSpeed isochBusSpeed

Isochronous bus speed.

7.5.2.7 unsigned int minNumImageNotifications

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

Read-only value.

7.5.2.8 unsigned int numBuffers

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

7.5.2.9 unsigned int numImageNotifications

Number of notifications per image.

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

There are 4 general scenarios:

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

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

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

7.5.2.10 unsigned int registerTimeout

Register read/write timeout value, in microseconds.

The default value is dependent on the interface type.

7.5.2.11 unsigned int registerTimeoutRetries

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

The default value is 0.

7.5.2.12 unsigned int reserved[16]

Reserved for future use.

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

- FlyCapture2Defs_C.h

7.6 fc2ConfigROM Struct Reference

Camera configuration ROM.

Data Fields

- unsigned int nodeVendorId
  Vendor ID of a node.
- unsigned int chipIdHi
  Chip ID (high part).
- unsigned int chipIdLo
  Chip ID (low part).
- unsigned int unitSpecId
  Unit Spec ID, usually 0xa02d.
- unsigned int unitSWVer
  Unit software version.
- unsigned int unitSubSWVer
  Unit sub software version.
- unsigned int vendorUniqueInfo_0
  Vendor unique info 0.
- unsigned int vendorUniqueInfo_1
  Vendor unique info 1.
- unsigned int vendorUniqueInfo_2
  Vendor unique info 2.

- unsigned int vendorUniqueInfo_3

  Vendor unique info 3.
- char pszKeyword [MAX_STRING_LENGTH]

  Keyword.
- unsigned int reserved [16]

  Reserved for future use.

7.6.1 Detailed Description

Camera configuration ROM.

7.6.2 Field Documentation

7.6.2.1 unsigned int chipIdHi

Chip ID (high part).

7.6.2.2 unsigned int chipIdLo

Chip ID (low part).

7.6.2.3 unsigned int nodeVendorId

Vendor ID of a node.

7.6.2.4 char pszKeyword[MAX_STRING_LENGTH]

Keyword.

7.6.2.5 unsigned int reserved[16]

Reserved for future use.

7.6.2.6 unsigned int unitSpecId

Unit Spec ID, usually 0xa02d.

7.6.2.7 unsigned int unitSubSWVer

Unit sub software version.

7.6.2.8 unsigned int unitSWVer

Unit software version.

7.6.2.9 unsigned int vendorUniqueInfo_0

Vendor unique info 0.

7.6.2.10 unsigned int vendorUniqueInfo_1

Vendor unique info 1.

7.6.2.11 unsigned int vendorUniqueInfo_2

Vendor unique info 2.

7.6.2.12 unsigned int vendorUniqueInfo_3

Vendor unique info 3.

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

- FlyCapture2Defs_C.h

7.7 fc2EmbeddedImageInfo Struct Reference

Properties of the possible embedded image information.

Collaboration diagram for fc2EmbeddedImageInfo:

Data Fields

- fc2EmbeddedImageInfoProperty timestamp
- fc2EmbeddedImageInfoProperty gain
- fc2EmbeddedImageInfoProperty shutter
- fc2EmbeddedImageInfoProperty brightness
- fc2EmbeddedImageInfoProperty exposure
- fc2EmbeddedImageInfoProperty whiteBalance
- fc2EmbeddedImageInfoProperty frameCounter
- fc2EmbeddedImageInfoProperty strobePattern
- fc2EmbeddedImageInfoProperty GPIOPinState
- fc2EmbeddedImageInfoProperty ROIPosition

7.7.1 Detailed Description

Properties of the possible embedded image information.

7.7.2 Field Documentation

7.7.2.1 fc2EmbeddedImageInfoProperty brightness

7.7.2.2 fc2EmbeddedImageInfoProperty exposure

7.7.2.3 fc2EmbeddedImageInfoProperty frameCounter

7.7.2.4 fc2EmbeddedImageInfoProperty gain

7.7.2.5 fc2EmbeddedImageInfoProperty GPIOPinState

7.7.2.6 fc2EmbeddedImageInfoProperty ROIPosition

7.7.2.7 fc2EmbeddedImageInfoProperty shutter

7.7.2.8 fc2EmbeddedImageInfoProperty strobePattern

7.7.2.9 fc2EmbeddedImageInfoProperty timestamp

7.7.2.10 fc2EmbeddedImageInfoProperty whiteBalance

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

- FlyCapture2Defs_C.h

7.8 fc2EmbeddedImageInfoProperty Struct Reference

Properties of a single embedded image info property.

Data Fields

- BOOL available

  Whether this property is available.
- BOOL onOff

  Whether this property is on or off.

7.8.1 Detailed Description

Properties of a single embedded image info property.

7.8.2 Field Documentation

7.8.2.1 BOOL available

Whether this property is available.

7.8.2.2 BOOL onOff

Whether this property is on or off.

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

- FlyCapture2Defs_C.h

7.9 fc2EventCallbackData Struct Reference

Data Fields

- void * EventUserData

  Pointer to the user-supplied data struct.
- size_t EventUserDataSize

  Size of the user data supplied to the RegisterEvent() function.
- const char * EventName

  The event name used to register the event.
- unsigned long long EventID

  The device register which EventName maps to.
- unsigned long long EventTimestamp

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

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

  The size of the structure pointed to by EventData.

7.9.1 Field Documentation

7.9.1.1 void* EventData

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

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

7.9.1.2 size_t EventDataSize

The size of the structure pointed to by EventData.

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

7.9.1.3 unsigned long long EventID

The device register which EventName maps to.

Provides an alternate means of indexing into different event types.

7.9.1.4 const char* EventName

The event name used to register the event.

Provided so the user knows which event triggered the callback.

7.9.1.5 unsigned long long EventTimestamp

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

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

7.9.1.6 void* EventUserData

Pointer to the user-supplied data struct.

7.9.1.7 size_t EventUserDataSize

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

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

- FlyCapture2Defs_C.h

7.10 fc2EventOptions Struct Reference

Options for enabling device event registration.

Data Fields

- fc2CameraEventCallback EventCallbackFcn

Callback function pointer.

- const char * EventName

  Event name to register.

- const void * EventUserData

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

- size_t EventUserDataSize

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

7.10.1 Detailed Description

Options for enabling device event registration.

7.10.2 Field Documentation

7.10.2.1 fc2CameraEventCallback EventCallbackFcn

Callback function pointer.

7.10.2.2 const char* EventName

Event name to register.

7.10.2.3 const void* EventUserData

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

7.10.2.4 size_t EventUserDataSize

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

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

- FlyCapture2Defs_C.h

7.11 fc2Format7ImageSettings Struct Reference

Format 7 image settings.

Data Fields

- fc2Mode mode

  Format 7 mode.

- unsigned int offsetX

  Horizontal image offset.
- unsigned int offsetY

  Vertical image offset.
- unsigned int width

  Width of image.
- unsigned int height

  Height of image.
- fc2PixelFormat pixelFormat

  Pixel format of image.
- unsigned int reserved [8]

  Reserved for future use.

7.11.1 Detailed Description

Format 7 image settings.

7.11.2 Field Documentation

7.11.2.1 unsigned int height

Height of image.

7.11.2.2 fc2Mode mode

Format 7 mode.

7.11.2.3 unsigned int offsetX

Horizontal image offset.

7.11.2.4 unsigned int offsetY

Vertical image offset.

7.11.2.5 fc2PixelFormat pixelFormat

Pixel format of image.

7.11.2.6 unsigned int reserved[8]

Reserved for future use.

7.11.2.7 unsigned int width

Width of image.

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

- FlyCapture2Defs_C.h

7.12 fc2Format7Info Struct Reference

Format 7 information for a single mode.

Data Fields

- fc2Mode mode

  Format 7 mode.
- unsigned int maxWidth

  Maximum image width.
- unsigned int maxHeight

  Maximum image height.
- unsigned int offsetHStepSize

  Horizontal step size for the offset.
- unsigned int offsetVStepSize

  Vertical step size for the offset.
- unsigned int imageHStepSize

  Horizontal step size for the image.
- unsigned int imageVStepSize

  Vertical step size for the image.
- unsigned int pixelFormatBitField

  Supported pixel formats in a bit field.
- unsigned int vendorPixelFormatBitField

  Vendor unique pixel formats in a bit field.
- unsigned int packetSize

  Current packet size in bytes.
- unsigned int minPacketSize

  Minimum packet size in bytes for current mode.
- unsigned int maxPacketSize

  Maximum packet size in bytes for current mode.
- float percentage

  Current packet size as a percentage of maximum packet size.
- unsigned int reserved [16]

  Reserved for future use.

7.12.1 Detailed Description

Format 7 information for a single mode.

7.12.2 Field Documentation

7.12.2.1 unsigned int imageHStepSize

Horizontal step size for the image.

7.12.2.2 unsigned int imageVStepSize

Vertical step size for the image.

7.12.2.3 unsigned int maxHeight

Maximum image height.

7.12.2.4 unsigned int maxPacketSize

Maximum packet size in bytes for current mode.

7.12.2.5 unsigned int maxWidth

Maximum image width.

7.12.2.6 unsigned int minPacketSize

Minimum packet size in bytes for current mode.

7.12.2.7 fc2Mode mode

Format 7 mode.

7.12.2.8 unsigned int offsetHStepSize

Horizontal step size for the offset.

7.12.2.9 unsigned int offsetVStepSize

Vertical step size for the offset.

7.12.2.10 unsigned int packetSize

Current packet size in bytes.

7.12.2.11 float percentage

Current packet size as a percentage of maximum packet size.

7.12.2.12 unsigned int pixelFormatBitField

Supported pixel formats in a bit field.

7.12.2.13 unsigned int reserved[16]

Reserved for future use.

7.12.2.14 unsigned int vendorPixelFormatBitField

Vendor unique pixel formats in a bit field.

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

- FlyCapture2Defs_C.h

7.13 fc2Format7PacketInfo Struct Reference

Format 7 packet information.

Data Fields

- unsigned int recommendedBytesPerPacket
  Recommended bytes per packet.
- unsigned int maxBytesPerPacket
  Maximum bytes per packet.
- unsigned int unitBytesPerPacket
  Minimum bytes per packet.
- unsigned int reserved [8]
  Reserved for future use.

7.13.1 Detailed Description

Format 7 packet information.

7.13.2 Field Documentation

7.13.2.1 unsigned int maxBytesPerPacket

Maximum bytes per packet.

7.13.2.2 unsigned int recommendedBytesPerPacket

Recommended bytes per packet.

7.13.2.3 unsigned int reserved[8]

Reserved for future use.

7.13.2.4 unsigned int unitBytesPerPacket

Minimum bytes per packet.

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

- FlyCapture2Defs_C.h

7.14 fc2GigEConfig Struct Reference

Configuration for a GigE camera.

Data Fields

- BOOL enablePacketResend

  Turn on/off packet resend functionality.
- unsigned int registerTimeoutRetries

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

  Register read/write timeout value, in microseconds.
- unsigned int reserved [8]

7.14.1 Detailed Description

Configuration for a GigE camera.

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

7.14.2 Field Documentation

7.14.2.1 BOOL enablePacketResend

Turn on/off packet resend functionality.

7.14.2.2 unsigned int registerTimeout

Register read/write timeout value, in microseconds.

The default value is dependent on the interface type.

7.14.2.3 unsigned int registerTimeoutRetries

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

The default value is 0.

7.14.2.4 unsigned int reserved[8]

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

- FlyCapture2Defs_C.h

7.15 fc2GigEImageSettings Struct Reference

Image settings for a GigE camera.

Data Fields

- unsigned int offsetX

  Horizontal image offset.
- unsigned int offsetY

  Vertical image offset.
- unsigned int width

  Width of image.
- unsigned int height

  Height of image.
- fc2PixelFormat pixelFormat

  Pixel format of image.
- unsigned int reserved [8]

  Reserved for future use.

7.15.1 Detailed Description

Image settings for a GigE camera.

7.15.2 Field Documentation

7.15.2.1 unsigned int height

Height of image.

7.15.2.2 unsigned int offsetX

Horizontal image offset.

7.15.2.3 unsigned int offsetY

Vertical image offset.

7.15.2.4 fc2PixelFormat pixelFormat

Pixel format of image.

7.15.2.5 unsigned int reserved[8]

Reserved for future use.

7.15.2.6 unsigned int width

Width of image.

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

- FlyCapture2Defs_C.h

7.16 fc2GigEImageSettingsInfo Struct Reference

Format 7 information for a single mode.

Data Fields

- unsigned int maxWidth

  Maximum image width.

- unsigned int maxHeight

  Maximum image height.
- unsigned int offsetHStepSize

  Horizontal step size for the offset.
- unsigned int offsetVStepSize

  Vertical step size for the offset.
- unsigned int imageHStepSize

  Horizontal step size for the image.
- unsigned int imageVStepSize

  Vertical step size for the image.
- unsigned int pixelFormatBitField

  Supported pixel formats in a bit field.
- unsigned int vendorPixelFormatBitField

  Vendor unique pixel formats in a bit field.
- unsigned int reserved [16]

  Reserved for future use.

7.16.1 Detailed Description

Format 7 information for a single mode.

7.16.2 Field Documentation

7.16.2.1 unsigned int imageHStepSize

Horizontal step size for the image.

7.16.2.2 unsigned int imageVStepSize

Vertical step size for the image.

7.16.2.3 unsigned int maxHeight

Maximum image height.

7.16.2.4 unsigned int maxWidth

Maximum image width.

7.16.2.5 unsigned int offsetHStepSize

Horizontal step size for the offset.

7.16.2.6 unsigned int offsetVStepSize

Vertical step size for the offset.

7.16.2.7 unsigned int pixelFormatBitField

Supported pixel formats in a bit field.

7.16.2.8 unsigned int reserved[16]

Reserved for future use.

7.16.2.9 unsigned int vendorPixelFormatBitField

Vendor unique pixel formats in a bit field.

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

- FlyCapture2Defs_C.h

7.17 fc2GigEProperty Struct Reference

A GigE property.

Data Fields

- fc2GigEPropertyType propType
  The type of property.
- BOOL isReadable
  Whether the property is readable.
- BOOL isWritable
  Whether the property is writable.
- unsigned int min
  Minimum value.
- unsigned int max
  Maximum value.
- unsigned int value
  Current value.
- unsigned int reserved [8]

7.17.1 Detailed Description

A GigE property.

7.17.2 Field Documentation

7.17.2.1 BOOL isReadable

Whether the property is readable.

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

7.17.2.2 BOOL isWritable

Whether the property is writable.

7.17.2.3 unsigned int max

Maximum value.

7.17.2.4 unsigned int min

Minimum value.

7.17.2.5 fc2GigEPropertyType propType

The type of property.

7.17.2.6 unsigned int reserved[8]

7.17.2.7 unsigned int value

Current value.

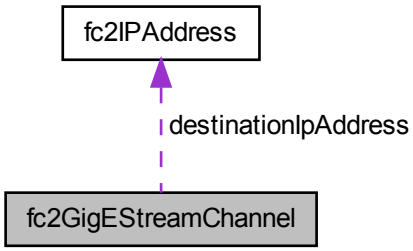
The documentation for this struct was generated from the following file:

- FlyCapture2Defs_C.h

7.18 fc2GigEStreamChannel Struct Reference

Information about a single GigE stream channel.

Collaboration diagram for fc2GigEStreamChannel:



Data Fields

- unsigned int networkInterfaceIndex

  Network interface index used (or to use).
- unsigned int hostPort

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

  Disable IP fragmentation of packets.
- unsigned int packetSize

  Packet size, in bytes.
- unsigned int interPacketDelay

  Inter packet delay, in timestamp counter units.
- fc2IPAddress destinationIpAddress

  Destination IP address.
- unsigned int sourcePort

  Source UDP port of the stream channel.
- unsigned int reserved [8]

7.18.1 Detailed Description

Information about a single GigE stream channel.

7.18.2 Field Documentation

7.18.2.1 fc2IPAddress destinationIpAddress

Destination IP address.

It can be a multicast or unicast address.

7.18.2.2 BOOL doNotFragment

Disable IP fragmentation of packets.

7.18.2.3 unsigned int hostPort

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

7.18.2.4 unsigned int interPacketDelay

Inter packet delay, in timestamp counter units.

7.18.2.5 unsigned int networkInterfaceIndex

Network interface index used (or to use).

7.18.2.6 unsigned int packetSize

Packet size, in bytes.

7.18.2.7 unsigned int reserved[8]

7.18.2.8 unsigned int sourcePort

Source UDP port of the stream channel.

Read only.

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

- FlyCapture2Defs_C.h

7.19 fc2H264Option Struct Reference

Options for saving H264 files.

Data Fields

- float frameRate

  Frame rate of the stream.
- unsigned int width

Width of source image.

- unsigned int height

  Height of source image.

- unsigned int bitrate

  Bitrate to encode at.

- unsigned int reserved [256]

  Reserved for future use.

7.19.1 Detailed Description

Options for saving H264 files.

7.19.2 Field Documentation

7.19.2.1 unsigned int bitrate

Bitrate to encode at.

7.19.2.2 float frameRate

Frame rate of the stream.

7.19.2.3 unsigned int height

Height of source image.

7.19.2.4 unsigned int reserved[256]

Reserved for future use.

7.19.2.5 unsigned int width

Width of source image.

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

- FlyCapture2Defs_C.h

7.20 fc2Image Struct Reference

Data Fields

- unsigned int rows

- unsigned int cols
- unsigned int stride
- unsigned char * pData
- unsigned int dataSize
- unsigned int receivedDataSize
- fc2PixelFormat format
- fc2BayerTileFormat bayerFormat
- fc2ImageImpl imageImpl

7.20.1 Field Documentation

7.20.1.1 fc2BayerTileFormat bayerFormat

7.20.1.2 unsigned int cols

7.20.1.3 unsigned int dataSize

7.20.1.4 fc2PixelFormat format

7.20.1.5 fc2ImageImpl imageImpl

7.20.1.6 unsigned char* pData

7.20.1.7 unsigned int receivedDataSize

7.20.1.8 unsigned int rows

7.20.1.9 unsigned int stride

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

- FlyCapture2Defs_C.h

7.21 fc2ImageMetadata Struct Reference

Metadata related to an image.

Data Fields

- unsigned int embeddedTimeStamp

  Embedded timestamp.
- unsigned int embeddedGain

  Embedded gain.
- unsigned int embeddedShutter

Embedded shutter.

- unsigned int embeddedBrightness
  Embedded brightness.
- unsigned int embeddedExposure
  Embedded exposure.
- unsigned int embeddedWhiteBalance
  Embedded white balance.
- unsigned int embeddedFrameCounter
  Embedded frame counter.
- unsigned int embeddedStrobePattern
  Embedded strobe pattern.
- unsigned int embeddedGPIOPinState
  Embedded GPIO pin state.
- unsigned int embeddedROIPosition
  Embedded ROI position.
- unsigned int reserved [31]
  Reserved for future use.

7.21.1 Detailed Description

Metadata related to an image.

7.21.2 Field Documentation

7.21.2.1 unsigned int embeddedBrightness

Embedded brightness.

7.21.2.2 unsigned int embeddedExposure

Embedded exposure.

7.21.2.3 unsigned int embeddedFrameCounter

Embedded frame counter.

7.21.2.4 unsigned int embeddedGain

Embedded gain.

7.21.2.5 unsigned int embeddedGPIOPinState

Embedded GPIO pin state.

7.21.2.6 unsigned int embeddedROIPosition

Embedded ROI position.

7.21.2.7 unsigned int embeddedShutter

Embedded shutter.

7.21.2.8 unsigned int embeddedStrobePattern

Embedded strobe pattern.

7.21.2.9 unsigned int embeddedTimeStamp

Embedded timestamp.

7.21.2.10 unsigned int embeddedWhiteBalance

Embedded white balance.

7.21.2.11 unsigned int reserved[31]

Reserved for future use.

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

- FlyCapture2Defs_C.h

7.22 fc2InternalContext Struct Reference

Data Fields

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

7.22.1 Field Documentation

7.22.1.1 FlyCapture2::BusManager* pBusMgr

7.22.1.2 FlyCapture2::CameraBase* pCamera

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

- FlyCapture2Internal_C.h

7.23 fc2InternalGuiContext Struct Reference

Data Fields

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

7.23.1 Field Documentation

7.23.1.1 FlyCapture2::CameraControlDlg* pCameraControlDlg

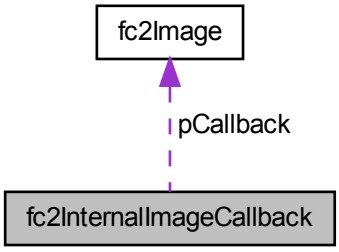
7.23.1.2 FlyCapture2::CameraSelectionDlg* pCameraSelectionDlg

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

- FlyCapture2Internal_C.h

7.24 fc2InternalImageCallback Struct Reference

Collaboration diagram for fc2InternalImageCallback:



Data Fields

- fc2ImageEventCallback pCallback
- void * pCallbackData

7.24.1 Field Documentation

7.24.1.1 fc2ImageEventCallback pCallback

7.24.1.2 void* pCallbackData

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

- FlyCapture2Internal_C.h

7.25 fc2IPAddress Struct Reference

IPv4 address.

Data Fields

- unsigned char octets [4]

7.25.1 Detailed Description

IPv4 address.

7.25.2 Field Documentation

7.25.2.1 unsigned char octets[4]

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

- FlyCapture2Defs_C.h

7.26 fc2JPEGOption Struct Reference

Options for saving JPEG image.

Data Fields

- BOOL progressive

  Whether to save as a progressive JPEG file.
- unsigned int quality

  JPEG image quality in range (0-100).
- unsigned int reserved [16]

  Reserved for future use.

7.26.1 Detailed Description

Options for saving JPEG image.

7.26.2 Field Documentation

7.26.2.1 BOOL progressive

Whether to save as a progressive JPEG file.

7.26.2.2 unsigned int quality

JPEG image quality in range (0-100).

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

7.26.2.3 unsigned int reserved[16]

Reserved for future use.

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

- FlyCapture2Defs_C.h

7.27 fc2JPG2Option Struct Reference

Options for saving JPEG2000 image.

Data Fields

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

7.27.1 Detailed Description

Options for saving JPEG2000 image.

7.27.2 Field Documentation

7.27.2.1 unsigned int quality

JPEG saving quality in range (1-512).

7.27.2.2 unsigned int reserved[16]

Reserved for future use.

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

- FlyCapture2Defs_C.h

7.28 fc2LUTData Struct Reference

Information about the camera's look up table.

Data Fields

- BOOL supported

  Flag indicating if LUT is supported.
- BOOL enabled

  Flag indicating if LUT is enabled.
- unsigned int numBanks

  The number of LUT banks available (Always 1 for PGR LUT).
- unsigned int numChannels

  The number of LUT channels per bank available.
- unsigned int inputBitDepth

  The input bit depth of the LUT.
- unsigned int outputBitDepth

  The output bit depth of the LUT.
- unsigned int numEntries

  The number of entries in the LUT.
- unsigned int reserved [8]

  Reserved for future use.

7.28.1 Detailed Description

Information about the camera's look up table.

7.28.2 Field Documentation

7.28.2.1 BOOL enabled

Flag indicating if LUT is enabled.

7.28.2.2 unsigned int inputBitDepth

The input bit depth of the LUT.

7.28.2.3 unsigned int numBanks

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

7.28.2.4 unsigned int numChannels

The number of LUT channels per bank available.

7.28.2.5 unsigned int numEntries

The number of entries in the LUT.

7.28.2.6 unsigned int outputBitDepth

The output bit depth of the LUT.

7.28.2.7 unsigned int reserved[8]

Reserved for future use.

7.28.2.8 BOOL supported

Flag indicating if LUT is supported.

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

- FlyCapture2Defs_C.h

7.29 fc2MACAddress Struct Reference

MAC address.

Data Fields

- unsigned char octets [6]

7.29.1 Detailed Description

MAC address.

7.29.2 Field Documentation

7.29.2.1 unsigned char octets[6]

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

- FlyCapture2Defs_C.h

7.30 fc2MJPGOption Struct Reference

Options for saving MJPG files.

Data Fields

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

7.30.1 Detailed Description

Options for saving MJPG files.

7.30.2 Field Documentation

7.30.2.1 float frameRate

Frame rate of the stream.

7.30.2.2 unsigned int quality

Image quality (1-100)

7.30.2.3 unsigned int reserved[256]

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

- FlyCapture2Defs_C.h

7.31 fc2PGMOption Struct Reference

Options for saving PGM images.

Data Fields

- BOOL binaryFile

  Whether to save the PPM as a binary file.
- unsigned int reserved [16]

  Reserved for future use.

7.31.1 Detailed Description

Options for saving PGM images.

7.31.2 Field Documentation

7.31.2.1 BOOL binaryFile

Whether to save the PPM as a binary file.

7.31.2.2 unsigned int reserved[16]

Reserved for future use.

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

- FlyCapture2Defs_C.h

7.32 fc2PGRGuid Struct Reference

A GUID to the camera.

Data Fields

- unsigned int value [4]

7.32.1 Detailed Description

A GUID to the camera.

It is used to uniquely identify a camera.

7.32.2 Field Documentation

7.32.2.1 unsigned int value[4]

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

- FlyCapture2Defs_C.h

7.33 fc2PNGOption Struct Reference

Options for saving PNG images.

Data Fields

- BOOL interlaced

  Whether to save the PNG as interlaced.
- unsigned int compressionLevel

  Compression level (0-9).
- unsigned int reserved [16]

  Reserved for future use.

7.33.1 Detailed Description

Options for saving PNG images.

7.33.2 Field Documentation

7.33.2.1 unsigned int compressionLevel

Compression level (0-9).

0 is no compression, 9 is best compression.

7.33.2.2 BOOL interlaced

Whether to save the PNG as interlaced.

7.33.2.3 unsigned int reserved[16]

Reserved for future use.

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

- FlyCapture2Defs_C.h

7.34 fc2PPMOption Struct Reference

Options for saving PPM images.

Data Fields

- BOOL binaryFile

  Whether to save the PPM as a binary file.
- unsigned int reserved [16]

  Reserved for future use.

7.34.1 Detailed Description

Options for saving PPM images.

7.34.2 Field Documentation

7.34.2.1 BOOL binaryFile

Whether to save the PPM as a binary file.

7.34.2.2 unsigned int reserved[16]

Reserved for future use.

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

- FlyCapture2Defs_C.h

7.35 fc2StrobeControl Struct Reference

A camera strobe.

Data Fields

- unsigned int source

  Source value.
- BOOL onOff

  Flag controlling on/off.
- unsigned int polarity

  Signal polarity.
- float delay

  Signal delay (in ms).
- float duration

  Signal duration (in ms).
- unsigned int reserved [8]

  Reserved for future use.

7.35.1 Detailed Description

A camera strobe.

7.35.2 Field Documentation

7.35.2.1 float delay

Signal delay (in ms).

7.35.2.2 float duration

Signal duration (in ms).

7.35.2.3 BOOL onOff

Flag controlling on/off.

7.35.2.4 unsigned int polarity

Signal polarity.

7.35.2.5 unsigned int reserved[8]

Reserved for future use.

7.35.2.6 unsigned int source

Source value.

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

- FlyCapture2Defs_C.h

7.36 fc2StrobeInfo Struct Reference

A camera strobe property.

Data Fields

- unsigned int source

  Source value.
- BOOL present

  Presence of strobe.
- BOOL readOutSupported

  Flag indicating if strobe value can be read out.
- BOOL onOffSupported

  Flag indicating if on/off is supported.
- BOOL polaritySupported

  Flag indicating if polarity is supported.
- float minValue

  Minimum value.
- float maxValue

  Maximum value.
- unsigned int reserved [8]

  Reserved for future use.

7.36.1 Detailed Description

A camera strobe property.

7.36.2 Field Documentation

7.36.2.1 float maxValue

Maximum value.

7.36.2.2 float minValue

Minimum value.

7.36.2.3 BOOL onOffSupported

Flag indicating if on/off is supported.

7.36.2.4 BOOL polaritySupported

Flag indicating if polarity is supported.

7.36.2.5 BOOL present

Presence of strobe.

7.36.2.6 BOOL readOutSupported

Flag indicating if strobe value can be read out.

7.36.2.7 unsigned int reserved[8]

Reserved for future use.

7.36.2.8 unsigned int source

Source value.

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

- FlyCapture2Defs_C.h

7.37 fc2SystemInfo Struct Reference

Description of the system.

Data Fields

- fc2OSType osType
  Operating system type as described by OSType.
- char osDescription [MAX_STRING_LENGTH]
  Detailed description of the operating system.

- fc2ByteOrder byteOrder

  Byte order of the system.
- size_t sysMemSize

  Amount of memory available on the system.
- char cpuDescription [MAX_STRING_LENGTH]

  Detailed description of the CPU.
- size_t numCpuCores

  Number of cores on all CPUs on the system.
- char driverList [MAX_STRING_LENGTH]

  List of drivers used.
- char libraryList [MAX_STRING_LENGTH]

  List of libraries used.
- char gpuDescription [MAX_STRING_LENGTH]

  Detailed description of the GPU.
- size_t screenWidth

  Screen resolution width in pixels.
- size_t screenHeight

  Screen resolution height in pixels.
- unsigned int reserved [16]

  Reserved for future use.

7.37.1 Detailed Description

Description of the system.

7.37.2 Field Documentation

7.37.2.1 fc2ByteOrder byteOrder

Byte order of the system.

7.37.2.2 char cpuDescription[MAX_STRING_LENGTH]

Detailed description of the CPU.

7.37.2.3 char driverList[MAX_STRING_LENGTH]

List of drivers used.

7.37.2.4 char gpuDescription[MAX_STRING_LENGTH]

Detailed description of the GPU.

7.37.2.5 char libraryList[MAX_STRING_LENGTH]

List of libraries used.

7.37.2.6 size_t numCpuCores

Number of cores on all CPUs on the system.

7.37.2.7 char osDescription[MAX_STRING_LENGTH]

Detailed description of the operating system.

7.37.2.8 fc2OSType osType

Operating system type as described by OSType.

7.37.2.9 unsigned int reserved[16]

Reserved for future use.

7.37.2.10 size_t screenHeight

Screen resolution height in pixels.

7.37.2.11 size_t screenWidth

Screen resolution width in pixels.

7.37.2.12 size_t sysMemSize

Amount of memory available on the system.

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

- FlyCapture2Defs_C.h

7.38 fc2TIFFOption Struct Reference

Options for saving TIFF images.

Data Fields

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

7.38.1 Detailed Description

Options for saving TIFF images.

7.38.2 Field Documentation

7.38.2.1 fc2TIFFCompressionMethod compression

Compression method to use for encoding TIFF images.

7.38.2.2 unsigned int reserved[16]

Reserved for future use.

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

- FlyCapture2Defs_C.h

7.39 fc2TimeStamp Struct Reference

Timestamp information.

Data Fields

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

7.39.1 Detailed Description

Timestamp information.

7.39.2 Field Documentation

7.39.2.1 unsigned int cycleCount

1394 cycle time count.

7.39.2.2 unsigned int cycleOffset

1394 cycle time offset.

7.39.2.3 unsigned int cycleSeconds

1394 cycle time seconds.

7.39.2.4 unsigned int microSeconds

Microseconds.

7.39.2.5 unsigned int reserved[8]

Reserved for future use.

7.39.2.6 long long seconds

Seconds.

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

- FlyCapture2Defs_C.h

7.40 fc2TriggerDelay Struct Reference

A specific camera property.

Data Fields

- fc2PropertyType type
  Property info type.

- BOOL present

  Flag indicating if the property is present.
- BOOL absControl

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

  Flag controlling one push.
- BOOL onOff

  Flag controlling on/off.
- BOOL autoManualMode

  Flag controlling auto.
- unsigned int valueA

  Value A (integer).
- unsigned int valueB

  Value B (integer).
- float absValue

  Floating point value.
- unsigned int reserved [8]

  Reserved for future use.

7.40.1 Detailed Description

A specific camera property.

For example, to set the gain to 12dB, set the following values:

- *type* - `GAIN`
- *absControl* - `true`
- *onePush* - `false`
- *onOff* - `true`
- *autoManualMode* - `false`
- *absValue* - `12.0`

7.40.2 Field Documentation

7.40.2.1 BOOL absControl

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

7.40.2.2 float absValue

Floating point value.

Used to configure properties in absolute mode.

7.40.2.3 BOOL autoManualMode

Flag controlling auto.

7.40.2.4 BOOL onePush

Flag controlling one push.

7.40.2.5 BOOL onOff

Flag controlling on/off.

7.40.2.6 BOOL present

Flag indicating if the property is present.

7.40.2.7 unsigned int reserved[8]

Reserved for future use.

7.40.2.8 fc2PropertyType type

Property info type.

7.40.2.9 unsigned int valueA

Value A (integer).

Used to configure properties in non-absolute mode.

7.40.2.10 unsigned int valueB

Value B (integer).

For white balance, value B applies to the blue value and value A applies to the red value.

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

- FlyCapture2Defs_C.h

7.41 fc2TriggerDelayInfo Struct Reference

Information about a specific camera property.

Data Fields

- fc2PropertyType type

  Property info type.
- BOOL present

  Flag indicating if the property is present.
- BOOL autoSupported

  Flag indicating if auto is supported.
- BOOL manualSupported

  Flag indicating if manual is supported.
- BOOL onOffSupported

  Flag indicating if on/off is supported.
- BOOL onePushSupported

  Flag indicating if one push is supported.
- BOOL absValSupported

  Flag indicating if absolute mode is supported.
- BOOL readOutSupported

  Flag indicating if property value can be read out.
- unsigned int min

  Minimum value (as an integer).
- unsigned int max

  Maximum value (as an integer).
- float absMin

  Minimum value (as a floating point value).
- float absMax

  Maximum value (as a floating point value).
- char pUnits [MAX_STRING_LENGTH]

  Textual description of units.
- char pUnitAbbr [MAX_STRING_LENGTH]

  Abbreviated textual description of units.
- unsigned int reserved [8]

  Reserved for future use.

7.41.1 Detailed Description

Information about a specific camera property.

This structure is also also used as the TriggerDelayInfo structure.

7.41.2 Field Documentation

7.41.2.1 float absMax

Maximum value (as a floating point value).

7.41.2.2 float absMin

Minimum value (as a floating point value).

7.41.2.3 BOOL absValSupported

Flag indicating if absolute mode is supported.

7.41.2.4 BOOL autoSupported

Flag indicating if auto is supported.

7.41.2.5 BOOL manualSupported

Flag indicating if manual is supported.

7.41.2.6 unsigned int max

Maximum value (as an integer).

7.41.2.7 unsigned int min

Minimum value (as an integer).

7.41.2.8 BOOL onePushSupported

Flag indicating if one push is supported.

7.41.2.9 BOOL onOffSupported

Flag indicating if on/off is supported.

7.41.2.10 BOOL present

Flag indicating if the property is present.

7.41.2.11 char pUnitAbbr[MAX_STRING_LENGTH]

Abbreviated textual description of units.

7.41.2.12 char pUnits[MAX_STRING_LENGTH]

Textual description of units.

7.41.2.13 BOOL readOutSupported

Flag indicating if property value can be read out.

7.41.2.14 unsigned int reserved[8]

Reserved for future use.

7.41.2.15 fc2PropertyType type

Property info type.

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

- FlyCapture2Defs_C.h

7.42 fc2TriggerMode Struct Reference

A camera trigger.

Data Fields

- BOOL onOff
  Flag controlling on/off.
- unsigned int polarity
  Polarity value.
- unsigned int source
  Source value.
- unsigned int mode
  Mode value.
- unsigned int parameter
  Parameter value.
- unsigned int reserved [8]
  Reserved for future use.

7.42.1 Detailed Description

A camera trigger.

7.42.2 Field Documentation

7.42.2.1 unsigned int mode

Mode value.

7.42.2.2 BOOL onOff

Flag controlling on/off.

7.42.2.3 unsigned int parameter

Parameter value.

7.42.2.4 unsigned int polarity

Polarity value.

7.42.2.5 unsigned int reserved[8]

Reserved for future use.

7.42.2.6 unsigned int source

Source value.

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

- FlyCapture2Defs_C.h

7.43 fc2TriggerModeInfo Struct Reference

Information about a camera trigger property.

Data Fields

- BOOL present

  Presence of trigger mode.

- BOOL readOutSupported

  Flag indicating if trigger value can be read out.
- BOOL onOffSupported

  Flag indicating if on/off is supported.
- BOOL polaritySupported

  Flag indicating if polarity is supported.
- BOOL valueReadable

  Flag indicating if the value is readable.
- unsigned int sourceMask

  Source mask.
- BOOL softwareTriggerSupported

  Flag indicating if software trigger is supported.
- unsigned int modeMask

  Mode mask.
- unsigned int reserved [8]

  Reserved for future use.

7.43.1 Detailed Description

Information about a camera trigger property.

7.43.2 Field Documentation

7.43.2.1 unsigned int modeMask

Mode mask.

7.43.2.2 BOOL onOffSupported

Flag indicating if on/off is supported.

7.43.2.3 BOOL polaritySupported

Flag indicating if polarity is supported.

7.43.2.4 BOOL present

Presence of trigger mode.

7.43.2.5 BOOL readOutSupported

Flag indicating if trigger value can be read out.

7.43.2.6 unsigned int reserved[8]

Reserved for future use.

7.43.2.7 BOOL softwareTriggerSupported

Flag indicating if software trigger is supported.

7.43.2.8 unsigned int sourceMask

Source mask.

7.43.2.9 BOOL valueReadable

Flag indicating if the value is readable.

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

- FlyCapture2Defs_C.h

7.44 fc2Version Struct Reference

The current version of the library.

Data Fields

- unsigned int major

  Major version number.
- unsigned int minor

  Minor version number.
- unsigned int type

  Type version number.
- unsigned int build

  Build version number.

7.44.1 Detailed Description

The current version of the library.

7.44.2 Field Documentation

7.44.2.1 unsigned int build

Build version number.

7.44.2.2 unsigned int major

Major version number.

7.44.2.3 unsigned int minor

Minor version number.

7.44.2.4 unsigned int type

Type version number.

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

- FlyCapture2Defs_C.h

Chapter 8

File Documentation

8.1 FlyCapture2_C.h File Reference

Functions

- FLYCAPTURE2_C_API fc2Error fc2CreateContext (fc2Context *pContext)

  Create a FC2 context for IIDC camera.
- 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 fc2GetCameraFromIPAddress (fc2Context context, fc2IPAddress ipAddress, fc2PGRGuid *pGuid)

  Gets the PGRGuid for a camera with the specified IPv4 address.
- FLYCAPTURE2_C_API fc2Error fc2GetCameraFromIndex (fc2Context context, unsigned int index, fc2PGRGuid *pGuid)

  Gets the PGRGuid for a camera on the PC.
- FLYCAPTURE2_C_API fc2Error fc2GetCameraFromSerialNumber (fc2Context context, unsigned int 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 fc2ReadPhyRegister (fc2Context context, fc2PGRGuid guid, unsigned int page, unsigned int port, unsigned int address, unsigned int *pValue)

  Read a phy register on the specified device.
- FLYCAPTURE2_C_API fc2Error fc2WritePhyRegister (fc2Context context, fc2PGRGuid guid, unsigned int page, unsigned int port, unsigned int address, unsigned int value)

  Write a phy register on the specified device.
- FLYCAPTURE2_C_API fc2Error fc2GetUsbLinkInfo (fc2Context context, fc2PGRGuid guid, unsigned int *pValue)

  Read usb link info for the port that the specified device is connected to.
- FLYCAPTURE2_C_API fc2Error fc2GetUsbPortStatus (fc2Context context, fc2PGRGuid guid, unsigned int *pValue)

  Read usb port status for the port that the specified device is connected to.
- FLYCAPTURE2_C_API fc2Error fc2GetTopology (fc2Context context, fc2TopologyNodeContext *pTopologyNodeContext)

  Gets the topology information for the PC.
- FLYCAPTURE2_C_API fc2Error fc2RegisterCallback (fc2Context context, fc2BusEventCallback enumCallback, fc2BusCallbackType callbackType, void *pParameter, fc2CallbackHandle *pCallbackHandle)

  Register a callback function that will be called when the specified callback event occurs.
- FLYCAPTURE2_C_API fc2Error fc2UnregisterCallback (fc2Context context, fc2CallbackHandle callbackHandle)

  Unregister a callback function.
- FLYCAPTURE2_C_API fc2Error fc2RescanBus (fc2Context context)

  Force a rescan of the buses.
- FLYCAPTURE2_C_API fc2Error fc2ForceIPAddressToCamera (fc2Context context, fc2MACAddress macAddress, fc2IPAddress ipAddress, fc2IPAddress subnetMask, fc2IPAddress defaultGateway)

  Force the camera with the specific MAC address to the specified IP address, subnet mask and default gateway.
- FLYCAPTURE2_C_API fc2Error fc2ForceAllIPAddressesAutomatically ()

  Force all cameras on the network to be assigned sequential IP addresses on the same subnet as the network adapters that they are connected to.
- FLYCAPTURE2_C_API fc2Error fc2ForceIPAddressAutomatically (unsigned int serialNumber)

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

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

  Discover all cameras connected to the network even if they reside on a different subnet.
- FLYCAPTURE2_C_API fc2Error fc2IsCameraControlable (fc2Context context, fc2PGRGuid *pGuid, BOOL *pControlable)

  Query whether a GigE camera is controllable.
- FLYCAPTURE2_C_API fc2Error fc2Connect (fc2Context context, fc2PGRGuid *guid)

  Connects the fc2Context to the camera specified by the GUID.
- FLYCAPTURE2_C_API fc2Error fc2Disconnect (fc2Context context)

  Disconnects the fc2Context from the camera.
- FLYCAPTURE2_C_API BOOL fc2IsConnected (fc2Context context)

  Checks if the fc2Context is connected to a physical camera specified by a GUID.
- FLYCAPTURE2_C_API fc2Error fc2SetCallback (fc2Context context, fc2ImageEventCallback pCallbackFn, void *pCallbackData)

  Sets the callback data to be used on completion of image transfer.
- FLYCAPTURE2_C_API fc2Error fc2StartCapture (fc2Context context)

  Starts isochronous image capture.
- FLYCAPTURE2_C_API fc2Error fc2StartCaptureCallback (fc2Context context, fc2ImageEventCallback pCallbackFn, void *pCallbackData)

  Starts isochronous image capture.
- FLYCAPTURE2_C_API fc2Error fc2StartSyncCapture (unsigned int numCameras, fc2Context *pContexts)

  Starts synchronized isochronous image capture on multiple cameras.
- FLYCAPTURE2_C_API fc2Error fc2StartSyncCaptureCallback (unsigned int numCameras, fc2Context *pContexts, fc2ImageEventCallback *pCallbackFns, void **pCallbackDataArray)

  Starts synchronized isochronous image capture on multiple cameras.
- FLYCAPTURE2_C_API fc2Error fc2RetrieveBuffer (fc2Context context, fc2Image *pImage)

  Retrieves the next image object containing the next image.
- FLYCAPTURE2_C_API fc2Error fc2StopCapture (fc2Context context)

  Stops isochronous image transfer and cleans up all associated resources.
- FLYCAPTURE2_C_API fc2Error fc2WaitForBufferEvent (fc2Context context, fc2Image *pImage, unsigned int eventNumber)

  Retrieves the next image event containing the next part of the image.
- FLYCAPTURE2_C_API fc2Error fc2SetUserBuffers (fc2Context context, unsigned char *const ppMemBuffers, int size, int nNumBuffers)

  Specify user allocated buffers to use as image data buffers.
- FLYCAPTURE2_C_API fc2Error fc2GetConfiguration (fc2Context context, fc2Config *config)

  Get the configuration associated with the camera.

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

  Set the configuration associated with the camera.
- 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)

  Fire the software trigger according to the DCAM specifications.
- FLYCAPTURE2_C_API fc2Error fc2FireSoftwareTriggerBroadcast (fc2Context context)

  Fire the software trigger according to the DCAM specifications.

- FLYCAPTURE2_C_API fc2Error fc2GetTriggerDelayInfo (fc2Context context, fc2TriggerDelayInfo *triggerDelayInfo)

  Retrieve trigger delay information from the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetTriggerDelay (fc2Context context, fc2TriggerDelay *triggerDelay)

  Retrieve current trigger delay settings from the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetTriggerDelay (fc2Context context, fc2TriggerDelay *triggerDelay)

  Set the specified trigger delay settings to the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetTriggerDelayBroadcast (fc2Context context, fc2TriggerDelay *triggerDelay)

  Set the specified trigger delay settings to the camera.
- 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 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 specified current memory channel.

- FLYCAPTURE2_C_API fc2Error fc2RestoreFromMemoryChannel (fc2Context context, unsigned int channel)

  Restore the specified current memory channel.

- FLYCAPTURE2_C_API fc2Error fc2GetMemoryChannelInfo (fc2Context context, unsigned int *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 fc2Error fc2WriteRegister (fc2Context context, unsigned int address, unsigned int value)

  Write to the specified register on the camera.

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

  Read the specified register from the camera.

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

  Write to the specified register on the camera with broadcast.

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

  Write to the specified register block on the camera.

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

  Write to the specified register block on the camera.

- FLYCAPTURE2_C_API const char * fc2GetRegisterString (unsigned int registerVal)

  Returns a text representation of the register value.

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

  Get cycle time from camera.

- FLYCAPTURE2_C_API fc2Error fc2GetStats (fc2Context context, fc2CameraStats *pCameraStats)

Returns the camera diagnostic information.

- FLYCAPTURE2_C_API fc2Error ResetStats ()
- FLYCAPTURE2_C_API fc2Error fc2RegisterEvent (fc2Context context, fc2EventOptions *pOpts)

  Register the camera to issue a custom callback function call for a specific device event.
- FLYCAPTURE2_C_API fc2Error fc2DeregisterEvent (fc2Context context, fc2EventOptions *pOpts)

  De-register an event previously registered with the camera.
- FLYCAPTURE2_C_API fc2Error fc2RegisterAllEvents (fc2Context context, fc2EventOptions *pOpts)

  Register the camera to issue a custom callback function call for a specific device event.
- FLYCAPTURE2_C_API fc2Error fc2DeregisterAllEvents (fc2Context context)
- 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 fc2DiscoverGigEPacketSize (fc2Context context, unsigned int *packetSize)

  Discover the largest packet size that works for the network link between the PC and the camera.
- FLYCAPTURE2_C_API fc2Error fc2QueryGigEImagingMode (fc2Context context, fc2Mode mode, BOOL *isSupported)

  Check if the particular imaging mode is supported by the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetGigEImagingMode (fc2Context context, fc2Mode *mode)

  Get the current imaging mode on the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetGigEImagingMode (fc2Context context, fc2Mode mode)

  Set the current imaging mode to the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetGigEImageSettingsInfo (fc2Context context, fc2GigEImageSettingsInfo *pInfo)

  Get information about the image settings possible on the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetGigEImageSettings (fc2Context context, fc2GigEImageSettings *pImageSettings)

  Get the current image settings on the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetGigEImageSettings (fc2Context context, const fc2GigEImageSettings *pImageSettings)

Set the image settings specified to the camera.

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

  Get the current binning settings on the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetGigEImageBinningSettings (fc2Context context, unsigned int horzBinnningValue, unsigned int vertBinnningValue)

  Set the specified binning values to the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetNumStreamChannels (fc2Context context, unsigned int *numChannels)

  Get the number of stream channels present on the camera.
- FLYCAPTURE2_C_API fc2Error fc2GetGigEStreamChannelInfo (fc2Context context, unsigned int channel, fc2GigEStreamChannel *pChannel)

  Get the stream channel information for the specified channel.
- FLYCAPTURE2_C_API fc2Error fc2SetGigEStreamChannelInfo (fc2Context context, unsigned int channel, fc2GigEStreamChannel *pChannel)

  Set the stream channel information for the specified channel.
- FLYCAPTURE2_C_API fc2Error fc2GetGigEConfig (fc2Context context, fc2GigEConfig *pConfig)

  Get the current gige config on the camera.
- FLYCAPTURE2_C_API fc2Error fc2SetGigEConfig (fc2Context context, const fc2GigEConfig *pConfig)

  Set the gige config specified to the camera.
- 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 fc2CreateImage (fc2Image *pImage)

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

  Destroy the fc2Image.
- FLYCAPTURE2_C_API fc2Error fc2SetImageDimensions (fc2Image *pImage, unsigned int rows, unsigned int cols, unsigned int stride, fc2PixelFormat pixelFormat, fc2BayerTileFormat bayerFormat)

  Sets the dimensions of the image object.

- FLYCAPTURE2_C_API fc2Error fc2GetImageDimensions (fc2Image *pImage, unsigned int *pRows, unsigned int *pCols, unsigned int *pStride, fc2PixelFormat *pPixelFormat, fc2BayerTileFormat *pBayerFormat)

  Get the image dimensions associated with the image object.
- FLYCAPTURE2_C_API fc2Error fc2SetImageColorProcessing (fc2Image *pImage, fc2ColorProcessingAlgorithm colorProc)

  Set the color processing algorithm.
- FLYCAPTURE2_C_API fc2Error fc2GetImageColorProcessing (fc2Image *pImage, fc2ColorProcessingAlgorithm *pColorProc)

  Get the current color processing algorithm.
- 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 fc2GetImageData (fc2Image *pImage, unsigned char **ppData)

  Get a pointer to the data associated with the image.
- FLYCAPTURE2_C_API fc2Error fc2GetImageMetadata (fc2Image *pImage, fc2ImageMetadata *pImageMetaData)

  Get the metadata associated with the image.
- FLYCAPTURE2_C_API fc2TimeStamp fc2GetImageTimeStamp (fc2Image *pImage)

  Get the timestamp data associated with the image.
- FLYCAPTURE2_C_API fc2Error fc2SaveImage (fc2Image *pImage, const char *pFilename, 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 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 fc2ImageStatisticsEnableAll (fc2ImageStatisticsContext imageStatisticsContext)

Enable all channels.

- FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsDisableAll (fc2ImageStatisticsContext imageStatisticsContext)

  Disable all channels.

- FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableGreyOnly (fc2ImageStatisticsContext imageStatisticsContext)

  Enable only the grey channel.

- FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableRGBOnly (fc2ImageStatisticsContext imageStatisticsContext)

  Enable only the RGB channels.

- FLYCAPTURE2_C_API fc2Error fc2ImageStatisticsEnableHSLOnly (fc2ImageStatisticsContext imageStatisticsContext)

  Enable only the HSL channels.

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

  Get the status of a statistics channel.

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

  Set the status of a statistics channel.

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

  Get the range of a statistics channel.

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

  Get the range of a statistics channel.

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

  Get the number of unique pixel values in the image.

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

  Get the mean of the image.

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

  Get the histogram for the image.

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

  Get all statistics for the image.

- 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 video 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 fc2AVISetMaximumSize (fc2AVIContext AVIContext, unsigned int size)

  Set the maximum file size (in megabytes) of a AVI/MP4 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 fc2CreateTopologyNode (fc2TopologyNodeContext *pTopologyNodeContext)

  Create a TopologyNode context.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetGuid (fc2TopologyNodeContext TopologyNodeContext, fc2PGRGuid *pGuid)

  Get the PGRGuid associated with the node.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetDeviceId (fc2TopologyNodeContext TopologyNodeContext, int *pID)

  Get the device ID associated with the node.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetNodeType (fc2TopologyNodeContext TopologyNodeContext, fc2NodeType *pNodeType)

  Get the node type associated with the node.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetInterfaceType (fc2TopologyNodeContext TopologyNodeContext, fc2InterfaceType *pInterfaceType)

  Get the interface type associated with the node.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetNumChildren (fc2TopologyNodeContext TopologyNodeContext, unsigned int *pNumChildNodes)

  Get the number of child nodes.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetChild (fc2TopologyNodeContext TopologyNodeContext, unsigned int position, fc2TopologyNodeContext *pChildTopologyNodeContext)

  Get child node located at the specified position.

- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeAddChild (fc2TopologyNodeContext TopologyNodeContext, fc2TopologyNodeContext TopologyNodeChildContext)

  Add the specified TopologyNode as a child of the node.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetNumPorts (fc2TopologyNodeContext TopologyNodeContext, unsigned int *pNumPorts)

  Get the number of ports.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeGetPortType (fc2TopologyNodeContext TopologyNodeContext, unsigned int position, fc2PortType *pPortType)

  Get type of port located at the specified position.
- FLYCAPTURE2_C_API fc2Error fc2TopologyNodeAddPortType (fc2TopologyNodeContext TopologyNodeContext, fc2PortType portType)

  Add the specified PortType as a port of the node.
- FLYCAPTURE2_C_API BOOL fc2TopologyNodeAssignGuidToNode (fc2TopologyNodeContext TopologyNodeContext, fc2PGRGuid guid, int deviceId)

  Assign a PGRGuid and device ID to the node.
- FLYCAPTURE2_C_API BOOL fc2TopologyNodeAssignGuidToNodeEx (fc2TopologyNodeContext TopologyNodeContext, fc2PGRGuid guid, int deviceId, fc2NodeType nodeType)

  Assign a PGRGuid, device ID and nodeType to the node.
- FLYCAPTURE2_C_API fc2Error fc2DestroyTopologyNode (fc2TopologyNodeContext TopologyNodeContext)

  Destroy a TopologyNode context.
- FLYCAPTURE2_C_API fc2Error fc2CheckDriver (const fc2PGRGuid *pGuid)

  Check for driver compatibility for the given camera guid.
- FLYCAPTURE2_C_API fc2Error fc2GetDriverDeviceName (const fc2PGRGuid *pGuid, char *pDeviceName, size_t *deviceNameLength)

  Get the driver's name for a device.
- FLYCAPTURE2_C_API fc2Error fc2GetSystemInfo (fc2SystemInfo *pSystemInfo)

  Get system information.
- FLYCAPTURE2_C_API fc2Error fc2GetLibraryVersion (fc2Version *pVersion)

  Get library version.
- FLYCAPTURE2_C_API fc2Error fc2LaunchBrowser (const char *pAddress)

  Launch a URL in the system default browser.
- FLYCAPTURE2_C_API fc2Error fc2LaunchHelp (const char *pFileName)

  Open a CHM file in the system default CHM viewer.
- FLYCAPTURE2_C_API fc2Error fc2LaunchCommand (const char *pCommand)

  Execute a command in the terminal.
- FLYCAPTURE2_C_API fc2Error fc2LaunchCommandAsync (const char *pCommand, fc2AsyncCommandCallback pCallback, void *pUserData)

  Execute a command in the terminal.
- FLYCAPTURE2_C_API const char * fc2ErrorToDescription (fc2Error error)

  Get a string representation of an error.

8.1.1 Function Documentation

8.1.1.1 FLYCAPTURE2_C_API fc2Error fc2CreateContext (fc2Context * *pContext*)

Create a FC2 context for IIDC camera.

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

See also

fc2DestroyContext()

Parameters

pContext	A pointer to the fc2Context to be created.

Returns

A fc2Error indicating the success or failure of the function.

8.1.1.2 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.

See also

fc2DestroyContext()

Parameters

pContext	A pointer to the fc2Context to be created.

Returns

A fc2Error indicating the success or failure of the function.

8.1.1.3 FLYCAPTURE2_C_API fc2Error fc2DeregisterAllEvents (fc2Context *context*)

8.1.1.4 FLYCAPTURE2_C_API fc2Error fc2DeregisterEvent (fc2Context *context,* fc2EventOptions * *pOpts*)

De-register an event previously registered with the camera.

Parameters

context	The fc2Context to be used.
pOpts	Pointer to the EventOptions structure which defines the callback function to use, the event for which to register the device, and a pointer to user data (optional) to be passed to the callback function. The callback function and user data elements of the EventOptions structure are ignored in this call, and just the event name within the structure is used with this function call.

Returns

An Error indicating the success or failure of the function.

8.1.1.5 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.

See also

fc2CreateContext()

Parameters

context	The context to be destroyed.

Returns

A fc2Error indicating the success or failure of the function.

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

Get cycle time from camera.

Parameters

context	The fc2Context to be used.
Timestamp	struct.

Returns

A fc2Error indicating the success or failure of the function.

8.1.1.7 FLYCAPTURE2_C_API fc2Error fc2GetStats (fc2Context *context*, fc2CameraStats * *pCameraStats*)

Returns the camera diagnostic information.

Parameters

context	The fc2Context to be used.
pCamera-Stats	Pointer to the fc2CameraStats structure.

Returns

A fc2Error indicating the success or failure of the function.

8.1.1.8 FLYCAPTURE2_C_API fc2Error fc2RegisterAllEvents (fc2Context *context*, fc2EventOptions * *pOpts*)

Register the camera to issue a custom callback function call for a specific device event.

Parameters

context	The fc2Context to be used.
pOpts	Pointer to the EventOptions structure which defines the callback function to use, the event for which to register the device, and a pointer to user data (optional) to be passed to the callback function. The event name element of the structure is ignored with this function call. If a single event has already been registered via RegisterEvent(), this call will fail, as the user could accidentally change the the internal callback function pointer for a queued event. The user will need to de-register all registered events, then call this function again.

Returns

An Error indicating the success or failure of the function.

8.1.1.9 FLYCAPTURE2_C_API fc2Error fc2RegisterEvent (fc2Context *context*, fc2EventOptions * *pOpts*)

Register the camera to issue a custom callback function call for a specific device event.

Parameters

context	The fc2Context to be used.
pOpts	Pointer to the EventOptions structure which defines the callback function to use, the event for which to register the device, and a pointer to user data (optional) to be passed to the callback function.

Returns

An Error indicating the success or failure of the function.

8.1.1.10 FLYCAPTURE2_C_API fc2Error ResetStats ( )

8.2 FlyCapture2Defs_C.h File Reference

Data Structures

- struct fc2PGRGuid

  A GUID to the camera.
- struct fc2Image
- struct fc2SystemInfo

  Description of the system.
- struct fc2Version

  The current version of the library.
- struct fc2IPAddress

  IPv4 address.
- struct fc2MACAddress

  MAC address.
- struct fc2GigEProperty

  A GigE property.
- struct fc2GigEStreamChannel

  Information about a single GigE stream channel.
- struct fc2GigEConfig

  Configuration for a GigE camera.
- struct fc2GigEImageSettingsInfo

  Format 7 information for a single mode.
- struct fc2GigEImageSettings

  Image settings for a GigE camera.
- struct fc2Format7ImageSettings

  Format 7 image settings.
- struct fc2Format7Info

  Format 7 information for a single mode.
- struct fc2Format7PacketInfo

  Format 7 packet information.
- struct fc2Config

  Configuration for a camera.
- struct fc2TriggerDelayInfo

  Information about a specific camera property.
- struct fc2TriggerDelay

  A specific camera property.

- struct fc2TriggerModeInfo

  Information about a camera trigger property.
- struct fc2TriggerMode

  A camera trigger.
- struct fc2StrobeInfo

  A camera strobe property.
- struct fc2StrobeControl

  A camera strobe.
- struct fc2TimeStamp

  Timestamp information.
- struct fc2ConfigROM

  Camera configuration ROM.
- struct fc2CameraInfo

  Camera information.
- struct fc2EmbeddedImageInfoProperty

  Properties of a single embedded image info property.
- struct fc2EmbeddedImageInfo

  Properties of the possible embedded image information.
- struct fc2ImageMetadata

  Metadata related to an image.
- struct fc2LUTData

  Information about the camera's look up table.
- struct fc2CameraStats

  Camera diagnostic information.
- struct fc2PNGOption

  Options for saving PNG images.
- struct fc2PPMOption

  Options for saving PPM images.
- struct fc2PGMOption

  Options for saving PGM images.
- struct fc2TIFFOption

  Options for saving TIFF images.
- struct fc2JPEGOption

  Options for saving JPEG image.
- struct fc2JPG2Option

  Options for saving JPEG2000 image.
- struct fc2BMPOption

  Options for saving Bitmap image.
- struct fc2MJPGOption

  Options for saving MJPG files.
- struct fc2H264Option

  Options for saving H264 files.
- struct fc2AVIOption

Options for saving AVI files.

- struct fc2EventOptions

  Options for enabling device event registration.

- struct fc2EventCallbackData

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 * fc2TopologyNodeContext

  A context referring to the TopologyNode object.

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

Enumerations

- enum fc2Error { FC2_ERROR_UNDEFINED = -1, FC2_ERROR_OK, FC2_ERROR_FAILED, FC2_ERROR_NOT_IMPLEMENTED, FC2_ERROR_FAILED_BUS_MASTER_CONNECTION, FC2_ERROR_NOT_CONNECTED, FC2_ERROR_INIT_FAILED, FC2_ERROR_NOT_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_INCOMPATIBLE_DRIVER, FC2_ERROR_FORCE_32BITS = FULL_32BIT_VALUE }

The error types returned by functions.

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

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

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

  The grab strategy employed during image transfer.

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

  Timeout options for grabbing images.

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

  Bandwidth allocation options for 1394 devices.

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

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

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

Camera properties.

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

Frame rates in frames per second.

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

DCAM video modes.

- enum fc2Mode { FC2_MODE_0 = 0, FC2_MODE_1, FC2_MODE_2, FC2_MODE_3, FC2_MODE_4, FC2_MODE_5, FC2_MODE_6, FC2_MODE_7, FC2_MODE_8, FC2_MODE_9, FC2_MODE_10, FC2_MODE_11, FC2_MODE_12, FC2_MODE_13, FC2_MODE_14, FC2_MODE_15, FC2_MODE_16, FC2_MODE_17, FC2_MODE_18, FC2_MODE_19, FC2_MODE_20, FC2_MODE_21, FC2_MODE_22, FC2_MODE_23, FC2_MODE_24, FC2_MODE_25, FC2_MODE_26, FC2_MODE_27, FC2_MODE_28, FC2_MODE_29, FC2_MODE_30, FC2_MODE_31, FC2_NUM_MODES, FC2_MODE_FORCE_32BITS = FULL_32BIT_VALUE }

Camera modes for DCAM formats as well as Format7.

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

Pixel formats available for Format7 modes.

- enum fc2BusSpeed { FC2_BUSSPEED_S100, FC2_BUSSPEED_S200, FC2_BUSSPEED_S400, FC2_BUSSPEED_S480, FC2_BUSSPEED_S800, FC2-

_BUSSPEED_S1600, FC2_BUSSPEED_S3200, FC2_BUSSPEED_S5000, × FC2_BUSSPEED_10BASE_T, FC2_BUSSPEED_100BASE_T, FC2_BUSSPEED_1000BASE_T, FC2_BUSSPEED_10000BASE_T, FC2_BUSSPEED_S_FASTEST, FC2_BUSSPEED_ANY, FC2_BUSSPEED_SPEED_UNKNOWN = -1, FC2_BUSSPEED_FORCE_32BITS = FULL_32BIT_VALUE }

Bus speeds.

- enum 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 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_GIGE_LWF, FC2_DRIVER_UNKNOWN = -1, FC2_DRIVER_FORCE_32BITS = FULL_32BIT_VALUE }

  Types of low level drivers that FlyCapture uses.
- enum fc2ColorProcessingAlgorithm { FC2_DEFAULT, FC2_NO_COLOR_PROCESSING, FC2_NEAREST_NEIGHBOR_FAST, FC2_EDGE_SENSING, × FC2_HQ_LINEAR, FC2_RIGOROUS, FC2_IPP, FC2_DIRECTIONAL, FC2_WEIGHTED_DIRECTIONAL, FC2_COLOR_PROCESSING_ALGORITHM_FORCE_32BITS = FULL_32BIT_VALUE }

  Color processing algorithms.
- enum fc2BayerTileFormat { FC2_BT_NONE, FC2_BT_RGGB, FC2_BT_GRBG, FC2_BT_GBRG, FC2_BT_BGGR, FC2_BT_FORCE_32BITS = FULL_32BIT_VALUE }

  Bayer tile formats.
- enum fc2ImageFileFormat { FC2_FROM_FILE_EXT = -1, FC2_PGM, FC2_PPM, FC2_BMP, FC2_JPEG, FC2_JPEG2000, FC2_TIFF, FC2_PNG, FC2_RAW, FC2_IMAGE_FILE_FORMAT_FORCE_32BITS = FULL_32BIT_VALUE }

  File formats to be used for saving images to disk.
- enum fc2GigEPropertyType { FC2_HEARTBEAT, FC2_HEARTBEAT_TIMEOUT, PACKET_SIZE, PACKET_DELAY }

  Possible properties that can be queried from the camera.
- enum fc2StatisticsChannel { FC2_STATISTICS_GREY, FC2_STATISTICS_RED, FC2_STATISTICS_GREEN, FC2_STATISTICS_BLUE, FC2_STATISTICS_HUE, FC2_STATISTICS_SATURATION, FC2_STATISTICS_LIGHTNESS, FC2_STATISTICS_FORCE_32BITS = FULL_32BIT_VALUE }

  Channels that allow statistics to be calculated.
- enum fc2OSType { FC2_WINDOWS_X86, FC2_WINDOWS_X64, FC2_LINUX_X86, FC2_LINUX_X64, FC2_MAC, FC2_UNKNOWN_OS, FC2_OSTYPE_FORCE_32BITS = FULL_32BIT_VALUE }

  Possible operating systems.
- enum fc2ByteOrder { FC2_BYTE_ORDER_LITTLE_ENDIAN, FC2_BYTE_ORDER_BIG_ENDIAN, FC2_BYTE_ORDER_FORCE_32BITS = FULL_32BIT_VALUE }

Possible byte orders.

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

  Possible states of a port on a node.

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

  Type of node.

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

8.2.1 Enumeration Type Documentation

8.2.1.1 enum fc2ByteOrder

Possible byte orders.

Enumerator:

FC2_BYTE_ORDER_LITTLE_ENDIAN
FC2_BYTE_ORDER_BIG_ENDIAN
FC2_BYTE_ORDER_FORCE_32BITS

8.2.1.2 enum fc2NodeType

Type of node.

Enumerator:

COMPUTER
BUS
CAMERA
NODE

8.2.1.3 enum fc2OSType

Possible operating systems.

Enumerator:

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

8.2.1.4 enum fc2PortType

Possible states of a port on a node.

Enumerator:

NOT_CONNECTED

CONNECTED_TO_PARENT

CONNECTED_TO_CHILD

8.2.1.5 enum fc2StatisticsChannel

Channels that allow statistics to be calculated.

Enumerator:

FC2_STATISTICS_GREY

FC2_STATISTICS_RED

FC2_STATISTICS_GREEN

FC2_STATISTICS_BLUE

FC2_STATISTICS_HUE

FC2_STATISTICS_SATURATION

FC2_STATISTICS_LIGHTNESS

FC2_STATISTICS_FORCE_32BITS

8.3 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 fc2GUIDisconnect (fc2GuiContext context)

  Disconnect GUI context from camera.
- FLYCAPTURE2_C_API void fc2Disonnect (fc2GuiContext context) __attribute__((deprecated))

  Disconnect GUI context from camera.

- FLYCAPTURE2_C_API void fc2Show (fc2GuiContext context)

  Show the GUI.
- FLYCAPTURE2_C_API void fc2Hide (fc2GuiContext context)

  Hide the GUI.
- FLYCAPTURE2_C_API BOOL fc2IsVisible (fc2GuiContext context)

  Check if the GUI is visible.
- FLYCAPTURE2_C_API void fc2ShowModal (fc2GuiContext context, BOOL *pOkSelected, fc2PGRGuid *guidArray, unsigned int *size)

  Show the camera selection dialog.

8.3.1 Function Documentation

8.3.1.1 FLYCAPTURE2_C_API fc2Error fc2CreateGUIContext (fc2GuiContext * *pContext*)

Create a GUI context.

Any GigE cameras that were connected prior to this call will lose CCP after the call. Consider creating a GUI context prior to connecting any GigE cameras or calling connect on any outstanding GigE camera context.

Parameters

pContext	Pointer to context to be created.

Returns

An Error indicating the success or failure of the function.

8.3.1.2 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.

8.3.1.3 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.

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

8.3.1.4 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.

8.3.1.5 FLYCAPTURE2_C_API void fc2GUIDisconnect (fc2GuiContext *context*)

Disconnect GUI context from camera.

Parameters

context	GUI context to disconnect.

Returns

An Error indicating the success or failure of the function.

8.3.1.6 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.

8.3.1.7 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.

8.3.1.8 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.

8.3.1.9 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.

8.4 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)

8.4.1 Function Documentation

8.4.1.1 bool IsContextValid (fc2Context *context*) [inline]

8.4.1.2 bool IsGuiContextValid (fc2GuiContext *context*) [inline]

8.4.1.3 void SyncCppImageToStruct (fc2Image * *pImage*) [inline]

8.5 FlyCapture2Platform_C.h File Reference

Defines

- #define FLYCAPTURE2_C_API
- #define FLYCAPTURE2_C_CALL_CONVEN

8.5.1 Define Documentation

8.5.1.1 #define FLYCAPTURE2_C_API

8.5.1.2 #define FLYCAPTURE2_C_CALL_CONVEN

8.6 FlyCapture2Private_C.h File Reference

Functions

- FLYCAPTURE2_C_API void * GetInternal (unsigned int index)

8.6.1 Function Documentation

8.6.1.1 FLYCAPTURE2_C_API void* GetInternal (unsigned int *index*)

8.7 Licensing.dox File Reference

8.8 MultiSyncLibrary_C.h File Reference

Functions

- MULTISYNCLIBRARY_C_API syncError syncCreateContext (syncContext *p-Context)

  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 (sync-Context 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 syncQueryCrossPCSynchronization-Setting (syncContext context)

  Query cross pc synchronizaion support status.

8.8.1 Function Documentation

8.8.1.1 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.

8.8.1.2 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.

8.8.1.3 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

8.8.1.4 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

8.8.1.5 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.

8.8.1.6 MULTISYNCLIBRARY_C_API double syncGetTimeSinceSynced (syncContext *context*)

Time since sync started.

Parameters

context	The syncContext to be used.

Returns

Time sinced synced.

8.8.1.7 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

8.8.1.8 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

8.8.1.9 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.

8.8.1.10 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.

8.8.1.11 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.

8.9 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 }

8.9.1 Define Documentation

8.9.1.1 #define FALSE 0

8.9.1.2 #define FULL_32BIT_VALUE 0x7FFFFFFF

8.9.1.3 #define MAX_STRING_LENGTH 512

8.9.1.4 #define TRUE 1

8.9.2 Typedef Documentation

8.9.2.1 typedef int BOOL

8.9.2.2 typedef void* syncContext

A context to the MultiSync C library.

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

8.9.3 Enumeration Type Documentation

8.9.3.1 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

8.9.3.2 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

8.10 MultiSyncLibraryPlatform_C.h File Reference

Defines

- #define MULTISYNCLIBRARY_C_API
- #define MULTISYNCLIBRARY_C_CALL_CONVEN

8.10.1 Define Documentation

8.10.1.1 #define MULTISYNCLIBRARY_C_API

8.10.1.2 #define MULTISYNCLIBRARY_C_CALL_CONVEN

Index