POINT GREY RESEARCH

FlyCapture

API Programming Reference

Version 1.6 Release Candidate 10
Revised October 3, 2006

Point Grey Research Inc.
8866 Hudson Street • Vancouver, BC • Canada • V6P 4N2 • T (604) 730-9937 • www.ptgrey.com

Software Warranty

The FlyCapture® Software Development Kit (the "Software") is owned and copyrighted by Point Grey Research, Inc. All rights are reserved. The Original Purchaser is granted a license to use the Software subject to the following restrictions and limitations.

1. The license is to the Original Purchaser only, and is nontransferable unless you have received written permission of Point Grey Research, Inc.
2. The Original Purchaser may use the Software only with Point Grey Research, Inc. cameras owned by the Original Purchaser, including but not limited to, Firefly®, Firefly®2, Firefly® MV, Flea®, Scorpion™, Dragonfly®, Dragonfly®2 or Dragonfly Express™ Camera Modules.
3. The Original Purchaser may make back-up copies of the Software for his or her own use only, subject to the use limitations of this license.
4. Subject to s.5 below, the Original Purchaser may not engage in, nor permit third parties to engage in, any of the following:
   A. Providing or disclosing the Software to third parties.
   B. Making alterations or copies of any kind of the Software (except as specifically permitted in s.3 above).
   C. Attempting to un-assemble, de-compile or reverse engineer the Software in any way.
   D. Granting sublicenses, leases or other rights in the Software to others.
5. Original Purchasers who are Original Equipment Manufacturers may make Derivative Products with the Software. Derivative Products are new software products developed, in whole or in part, using the Software and other Point Grey Research, Inc. products. Point Grey Research, Inc. hereby grants a license to Original Equipment Manufacturers to incorporate and distribute the libraries found in the Software with the Derivative Products. The components of any Derivative Product that contain the Software libraries may only be used with Point Grey Research, Inc. products, or images derived from such products.
5.1 By the distribution of the Software libraries with Derivative Products, Original Purchasers agree to:
   A. not permit further redistribution of the Software libraries by end-user customers;
   B. include a valid copyright notice on any Derivative Product; and
   C. indemnify, hold harmless, and defend Point Grey Research, Inc. from and against any claims or lawsuits, including attorney's fees, that arise or result from the use or distribution of any Derivative Product.

Point Grey Research, Inc. reserves the right to terminate this license if there are any violations of its terms or if there is a default committed by the Original Purchaser. Upon termination, for any reason, all copies of the Software must be immediately returned to Point Grey Research, Inc. and the Original Purchaser shall be liable to Point Grey Research, Inc. for any and all damages suffered as a result of the violation or default.

Software License Agreement

The FlyCapture® Software Development Kit (the "Software") is owned and copyrighted by Point Grey Research, Inc. All rights are reserved. The Original Purchaser is granted a license to use the Software subject to the following restrictions and limitations.

1. The license is to the Original Purchaser only, and is nontransferable unless you have received written permission of Point Grey Research, Inc.
2. The Original Purchaser may use the Software only with Point Grey Research, Inc. cameras owned by the Original Purchaser, including but not limited to, Firefly®, Firefly®2, Firefly® MV, Flea®, Scorpion™, Dragonfly®, Dragonfly®2 or Dragonfly Express™ Camera Modules.
3. The Original Purchaser may make back-up copies of the Software for his or her own use only, subject to the use limitations of this license.
4. Subject to s.5 below, the Original Purchaser may not engage in, nor permit third parties to engage in, any of the following:
   A. Providing or disclosing the Software to third parties.
   B. Making alterations or copies of any kind of the Software (except as specifically permitted in s.3 above).
   C. Attempting to un-assemble, de-compile or reverse engineer the Software in any way.
   D. Granting sublicenses, leases or other rights in the Software to others.
5. Original Purchasers who are Original Equipment Manufacturers may make Derivative Products with the Software. Derivative Products are new software products developed, in whole or in part, using the Software and other Point Grey Research, Inc. products. Point Grey Research, Inc. hereby grants a license to Original Equipment Manufacturers to incorporate and distribute the libraries found in the Software with the Derivative Products. The components of any Derivative Product that contain the Software libraries may only be used with Point Grey Research, Inc. products, or images derived from such products.
5.1 By the distribution of the Software libraries with Derivative Products, Original Purchasers agree to:
   A. not permit further redistribution of the Software libraries by end-user customers;
   B. include a valid copyright notice on any Derivative Product; and
   C. indemnify, hold harmless, and defend Point Grey Research, Inc. from and against any claims or lawsuits, including attorney's fees, that arise or result from the use or distribution of any Derivative Product.

Point Grey Research, Inc. reserves the right to terminate this license if there are any violations of its terms or if there is a default committed by the Original Purchaser. Upon termination, for any reason, all copies of the Software must be immediately returned to Point Grey Research, Inc. and the Original Purchaser shall be liable to Point Grey Research, Inc. for any and all damages suffered as a result of the violation or default.

Table of Contents

1 Header Files

The information for this API Programming Reference is derived from the commented sections of the following FlyCapture header (.h) files, located in the C:\Program Files\Point Grey Research\PGR FlyCapture\include directory:

- pgrcameragui.h
- PGRFlyCapture.h
- pgrflycaptureregui.h
- PGRFlyCaptureMessaging.h
- PGRFlyCapturePlus.h

2 FlyCapture API

2.1. 1394 Bus Functions

2.1.1. flycaptureBusCameraCount

This function returns the number of 1394 cameras attached to the machine.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureBusCameraCount(
                        unsigned int* puiCount )
```

Parameters

puiCount	The number of cameras on the bus.

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.1.2. flycaptureBusEnumerateCamerasEx

This function enumerates all of the cameras found on the machine. It fills an array of FlyCaptureInfoEx structures with all of the pertinent information from the attached cameras. The index of a given FlyCaptureInfoEx structure in the array parInfo is the device number.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureBusEnumerateCamerasEx(
                                FlyCaptureInfoEx*  arInfo,
                                unsigned int*      puiSize )
```

Parameters

arInfo	An array of FlyCaptureInfoEx structures, at least as large as the number of cameras on the bus.
puiSize	The size of the array passed in. The number of cameras detected is passed back in this argument also.

Return Value
A FlyCaptureError indicating the success or failure of the function.

See Also
flycaptureBusCameraCount()

2.1.3. flycaptureModifyCallback

This function registers or deregisters a bus callback function. When the state of the bus changes, the registered callback function will be called with a FLYCAPTURE_MESSAGE_X parameter indicating the type of event. Please see the FlyCap example for more information on how to use callback functionality.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureModifyCallback(
                        FlyCaptureContext   context,
                        FlyCaptureCallback* pfnCallback,
                        void*               pParam,
                        bool                bAdd )
```

Parameters

context	The FlyCapture context to access.
pfnCallback	A pointer to an externally defined callback function.
pParam	A user-specified parameter to be passed back to the callback function. Can be NULL.
bAdd	True if the callback is to be added to the list of callbacks, false if the callback is to be removed.

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.2. Camera Property Functions

2.2.1. flycaptureEnableLookUpTable

This function turns the look up table on or off.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureEnableLookUpTable(
                            FlyCaptureContext context,
                            bool              bOn )
```

Parameters

context	The context associated with the camera.
bOn	true to enable, false to disable.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks

The look up table on some cameras can not be turned off. These cameras return FLYCAPTURE_NOT_IMPLEMENTED if bOn is false. See the description of flycaptureQueryLookUpTable() for help on identifying these cameras.

2.2.2. flycaptureGetCameraAbsProperty

Allows the user to get the current absolute value for a given parameter from the camera if it is supported.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureGetCameraAbsProperty(
                                FlyCaptureContext   context,
                                FlyCaptureProperty  cameraProperty,
                                float*              pfValue )
```

Parameters

context	The FlyCapture context to query.
cameraProperty	A FlyCaptureProperty indicating which property to query.
pfValue	A pointer to a float that will contain the result.

Return Value
A FlyCaptureError indicating the success or failure of the operation.

2.2.3. flycaptureGetCameraAbsPropertyEx

Allows the user to get the current absolute value for a given parameter from the camera if it is supported. This function also allows the user to query the states of the one push, on/off, and auto controls in the property's standard register.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureGetCameraAbsPropertyEx(
                                FlyCaptureContext   context,
                                FlyCaptureProperty  cameraProperty,
                                bool*               pbOnePush,
                                bool*               pbOnOff,
                                bool*               pbAuto,
                                float*              pfValue )
```

Parameters

context	The FlyCapture context to query.
cameraProperty	A FlyCaptureProperty indicating which property to query.
pbOnePush	A valid pointer to a bool that will store the one push state
pbOnOff	A valid pointer to a bool that will store the on/off state.
pbAuto	A valid pointer to a bool that will store the auto state

pfValue	A pointer to a float that will contain the result.

Return Value
A FlyCaptureError indicating the success or failure of the operation.

Remarks
The data returned by this function is extracted by a series of two register reads.

2.2.4. flycaptureGetCameraAbsPropertyRange

Allows the user to determine the presence and range of the absolute value registers for the camera

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureGetCameraAbsPropertyRange(
                                    FlyCaptureContext  context,
                                    FlyCaptureProperty cameraProperty,
                                    bool*              pbPresent,
                                    float*             pfMin,
                                    float*             pfMax,
                                    const char**       ppszUnits,
                                    const char**       ppszUnitAbbr )
```

Parameters

context	The Flycapture context to query.
cameraProperty	A FlyCaptureProperty indicating which property to query.
pbPresent	Whether or not this register has absolute value support.
pfMin	The minimum value that this register can handle.
pfMax	The maximum value that this register can handle.
ppszUnits	A string indicating the units of the register.
ppszUnitAbbr	An abbreviation of the units

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.2.5. flycaptureGetCameraProperty

Allows the user to query the current value of the given property.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureGetCameraProperty(
                            FlyCaptureContext   context,
                            FlyCaptureProperty  cameraProperty,
                            long*               plValueA,
                            long*               plValueB,
                            bool*               pbAuto )
```

Parameters

context	The FlyCapture context to extract the properties from.
cameraProperty	A FlyCaptureProperty indicating the property to query.
plValueA	A pointer to storage space for the “A”, or first value associated with this property.
plValueB	A pointer to storage space for the “B”, or second value associated with this property.
pbAuto	A pointer to a bool that will store the current Auto value of the property.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
Pass NULL for any pointer argument to ignore that argument.

2.2.6. flycaptureGetCameraPropertyEx

Replaces flycaptureGetCameraProperty() and provides better access to camera features.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureGetCameraPropertyEx(
                              FlyCaptureContext   context,
                              FlyCaptureProperty  cameraProperty,
                              bool*               pbOnePush,
                              bool*               pbOnOff,
                              bool*               pbAuto,
                              int*                piValueA,
                              int*                piValueB )
```

Parameters

context	The FlyCapture context to extract the properties from.
cameraProperty	A FlyCaptureProperty indicating the property to query.
pbOnePush	The value of the one push bit.
pbOnOff	The value of the On/Off bit.
pbAuto	The value of the Auto bit.
piValueA	The current value of this property.
piValueB	The current secondary value of this property. (only used for the two whitebalance values.)

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
Pass NULL for any pointer argument to ignore that argument.

2.2.7. flycaptureGetCameraPropertyRange

Allows the user to examine the default, minimum, maximum, and auto characteristics for the given property.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureGetCameraPropertyRange(
                                  FlyCaptureContext  context,
                                  FlyCaptureProperty cameraProperty,
                                  bool*              pbPresent,
                                  long*              plMin,
                                  long*              plMax,
                                  long*              plDefault,
                                  bool*              pbAuto,
                                  bool*              pbManual )
```

Parameters

context	The FlyCapture context to extract the properties from.
cameraProperty	A FlyCaptureProperty indicating the property to examine.
pbPresent	A pointer to a bool that will contain whether or not camera property is present.
plMin	A pointer to a long that will contain the minimum property value.
plMax	A pointer to a long that will contain the maximum property value.
plDefault	A pointer to a long that will contain the default property value.
pbAuto	A pointer to a bool that will contain whether or not the Auto setting is available for this property.
pbManual	A pointer to a bool that will contain whether or not this property may be manually adjusted.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
Pass NULL for any pointer argument to ignore that argument.

2.2.8. flycaptureGetCameraPropertyRangeEx

Replaces flycaptureGetCameraPropertyRange() and provides better access to camera features.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureGetCameraPropertyRangeEx(
                                    FlyCaptureContext    context,
                                    FlyCaptureProperty   cameraProperty,
                                    bool*                pbPresent,
                                    bool*                pbOnePush,
                                    bool*                pbReadOut,
                                    bool*                pbOnOff,
```

```
                                bool*               pbAuto,
                                bool*               pbManual,
                                int*                piMin,
                                int*                piMax )
```

Parameters

context	The FlyCapture context to extract the properties from.
cameraProperty	A FlyCaptureProperty indicating the property to examine.
pbPresent	Indicates the presence of this property on the camera.
pbOnePush	Indicates the availability of the one push feature.
pbReadOut	Indicates the ability to read out the value of this property.
pbOnOff	Indicates the ability to turn this property on and off.
pbAuto	Indicates the availability of auto mode for this property.
pbManual	Indicates the ability to manually control this property.
piMin	The minimum value of the property is returned in this argument.
piMax	The maximum value of the property is returned in this argument.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
Pass NULL for any pointer argument to ignore that argument.

2.2.9. flycaptureGetCameraRegister

This function allows the user to get any of camera's registers.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureGetCameraRegister(
                            FlyCaptureContext context,
                            unsigned long     ulRegister,
                            unsigned long*    pulValue )
```

Parameters

context	The FlyCaptureContext associated with the camera to be queried.
ulRegister	The 32 bit register location to query.
pulValue	The 32 bit value currently stored in the register.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks

The ulRegister value is actually an offset applied to a base address. Typically this base addess is 0xFFFFF0F00000 but it is not constant. Refer to the "Unit Dependent Directory" section of the DCAM spec for more information.

2.2.10. flycaptureGetCameraTrigger

Deprecated. Please use flycaptureGetTrigger().

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureGetCameraTrigger(
                           FlyCaptureContext context,
                           unsigned int*     puiPresence,
                           unsigned int*     puiOnOff,
                           unsigned int*     puiPolarity,
                           unsigned int*     puiTriggerMode )
```

Return Value
FLYCAPTURE_DEPRECATED.

2.2.11. flycaptureGetLookUpTableChannel

This function will retrieve the specified look up table on the camera.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureGetLookUpTableChannel(
                                FlyCaptureContext context,
                                unsigned int      uiChannel,
                                unsigned int*     puiArray )
```

Parameters

context	The context associated with the camera.
uiChannel	The channel to retrieve
puiArray	a valid array of "numberOfEntries" unsigned ints

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.2.12. flycaptureGetMemoryChannel

This function will query the camera to see what the currently set memory channel is and/or what the maximum valid channel is. At least one pointer must be valid.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureGetMemoryChannel(
                              FlyCaptureContext context,
                              unsigned int*     puiCurrentChannel,
```

```
                                    unsigned int*     puiNumChannels = NULL )
```

Parameters

context	The FlyCaptureContext associated with the camera
puiNumChannels	NULL or a valid pointer to an unsigned int to store the maximum valid memory channel. Zero indicates no user channels.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
Refer to your camera's technical reference for the registers affected by the memory channels. Use flycaptureGetMemoryChannel() to check the current and/or maximum number of channels available.

2.2.13. flycaptureGetStrobe

This function allows the user to query the state of one of the camera's strobe sources. Only for use with cameras which support DCAM v1.31 compliant strobes.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureGetStrobe(
                    FlyCaptureContext context,
                    int               iSource,
                    bool*             pbOnOff,
                    int*              piPolarity,
                    int*              piDelay,
                    int*              piDuration )
```

Parameters

context	The context associated with the camera to be queried.
iSource	The strobe source to be queried.
pbOnOff	The current on/off status is returned in this paramaeter.
piPolarity	The current polarity of the strobe is returned. 1 or 0.
piDelay	The current delay is returned in this parameter.
piDuration	The current duration is returned in this parameter.

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.2.14. flycaptureGetTrigger

This function allows the user to query the state of the camera's trigger functionality. This function replaces the deprecated flycaptureGetCameraTrigger() function.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureGetTrigger(
                     FlyCaptureContext context,
                     bool*             pbOnOff,
                     int*              piPolarity,
                     int*              piSource,
                     int*              piRawValue,
                     int*              piMode,
                     int*              piParameter )
```

Parameters

context	The context associated with the camera to be queried.
pbOnOff	The On/Off state is returned in this parameter.
piPolarity	The polarity value is returned in this parameter.
piSource	The source value is returned in this parameter.
piRawValue	The raw signal value is returned in this parameter.
piMode	The trigger mode is returned in this parameter.
piParameter	The parameter for the trigger function is returned in this parameter.

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.2.15. flycaptureQueryLookUpTable

This function queries the availability and state of the camera's look up table.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureQueryLookUpTable(
                          FlyCaptureContext context,
                          bool*             pbAvailable,
                          unsigned int*     puiNumChannels,
                          bool*             pbOn,
                          unsigned int*     puiBitDepth,
                          unsigned int*     puiNumEntries )
```

Parameters

context	The context associated with the camera to be queried.
pbAvailable	NULL or a parameter which indicates if the LUT is supported
puiNumChannels	NULL or a parameter which indicates the number of available channels. NOTE: some cameras will return available, but zero channels. Typically, these cameras will have a single channel and not support turning the LUT off.
pbOn	NULL or a parameter which indicates whether the LUT is currently on
puiBitDepth	NULL or a parameter which indicates the bit depth of the LUT (this will be the number

	of bits in the output values).
puiNumEntries	NULL or a parameter which indicates the number of entries in the table (this will be the number of input values).

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
At least one parameter besides the context must be valid or an error will be returned.

2.2.16. flycaptureQueryStrobe

This function queries the abilities and available settings for a particular strobe source. Only for use with cameras which support DCAM v1.31 compliant strobes.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureQueryStrobe(
                      FlyCaptureContext context,
                      int               iSource,
                      bool*             pbAvailable,
                      bool*             pbReadOut,
                      bool*             pbOnOff,
                      bool*             pbPolarity,
                      int*              piMinValue,
                      int*              piMaxValue )
```

Parameters

context	The context associated with the camera to be queried.
iSource	The strobe source to be queried.
pbReadOut	Describes whether the source allows reading of the current value.
pbOnOff	Describes whether the source can be turned on or off.
pbPolarity	Describes whether the source's polarity can be changed.
piMinValue	This parameter holds the minimum value of the delay and duration.
piMaxValue	This parameter holds the maximum value of the delay and duration.

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.2.17. flycaptureQueryTrigger

This function allows the user to query the trigger functionality of the camera.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureQueryTrigger(
                       FlyCaptureContext  context,
```

```
                        bool*              pbPresent,
                        bool*              pbReadOut,
                        bool*              pbOnOff,
                        bool*              pbPolarity,
                        bool*              pbValueRead,
                        unsigned int*      puiSourceMask,
                        bool*              pbSoftwareTrigger,
                        unsigned int*      puiModeMask )
```

Parameters

context	The context associated with the camera to be queried.
pbPresent	Whether or not the camera has trigger functionality.
pbReadOut	Whether or not the user can read values in the trigger functionality.
pbOnOff	Whether or not the functionality can be turned on or off.
pbPolarity	Whether or not the polarity can be changed.
pbValueRead	Whether or not the raw trigger input can be read.
puiSourceMask	A bit field indicating which trigger sources are available.
pbSoftwareTrigger	Whether or not software triggering is available.
puiModeMask	A bit field indicating which trigger modes are available.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
Polarity and trigger sources are camera dependant.

2.2.18. flycaptureReadRegisterBlock

Provides block-read (asynchronous) access to the entire register space of the camera.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureReadRegisterBlock(
                            FlyCaptureContext   context,
                            unsigned short      usAddrHigh,
                            unsigned long       ulAddrLow,
                            unsigned long*      pulBuffer,
                            unsigned long       ulLength )
```

Parameters

context	The context associated with the camera to be accessed.
usAddrHigh	The top 16 bits of the 48-bit absolute address to read.
ulAddrLow	The bottom 32 bits of the 48-bit absolute addresss to read.

pulBuffer	The buffer that will receive the data. Must be of size ulLength.
ulLength	The length, in quadlets, of the block to read.

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.2.19. flycaptureRestoreFromMemoryChannel

This function will restore a group of register settings from the specified memory channel on the camera. This will make the specified channel the current channel.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureRestoreFromMemoryChannel(
                                    FlyCaptureContext context,
                                    unsigned long     ulChannel )
```

Parameters

context	The FlyCaptureContext associated with the camera
ulChannel	The channel to change to

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
Refer to your camera's technical reference for the registers affected by the memory channels. Use flycaptureGetMemoryChannel() to check the current and/or maximum number of channels available.

2.2.20. flycaptureSaveToMemoryChannel

This function will save a group of the current camera registers to the specified memory channel on the camera.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureSaveToMemoryChannel(
                               FlyCaptureContext context,
                               unsigned long     ulChannel )
```

Parameters

context	The FlyCaptureContext associated with the camera
ulChannel	The channel to store the values in

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks

Refer to your camera's technical reference for the registers affected by the memory channels. Use flycaptureGetMemoryChannel() to check the current and/or maximum number of channels available.

2.2.21. flycaptureSetCameraAbsProperty

Allows the user to set the absolute value of the given parameter if the mode is supported.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureSetCameraAbsProperty(
                              FlyCaptureContext  context,
                              FlyCaptureProperty cameraProperty,
                              float              fValue )
```

Parameters

context	The FlyCapture context to query.
cameraProperty	A FlyCaptureProperty indicating which property to query.
fValue	A float containing the new value of the parameter.

Return Value
A FlyCaptureError indicating the success or failure of the operation.

2.2.22. flycaptureSetCameraAbsPropertyBroadcast

Allows the user to set the absolute value of the given parameter to all cameras on the current bus.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureSetCameraAbsPropertyBroadcast(
                                        FlyCaptureContext  context,
                                        FlyCaptureProperty cameraProper
ty,
                                        float              fValue )
```

Parameters

context	The FlyCapture context to query.
cameraProperty	A FlyCaptureProperty indicating which property to query.
fValue	A float containing the new value of the parameter.

Return Value
A FlyCaptureError indicating the success or failure of the operation.

Remarks
This function will set the given property for all the cameras on the 1394 bus that are associated with the context passed. If multiple busses (i.e. more than one 1394 card) exist, a call to this function must be made for each bus using a context representing a camera on that bus.

2.2.23. flycaptureSetCameraAbsPropertyBroadcastEx

Allows the user to set the absolute value of the given parameter if the mode is supported. This function also allows the user to specify the one push, on/off, and auto settings of the same property.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureSetCameraAbsPropertyBroadcastEx(
                                FlyCaptureContext  context,
                                FlyCaptureProperty cameraProperty,
                                bool               bOnePush,
                                bool               bOnOff,
                                bool               bAuto,
                                float              fValue )
```

Parameters

context	The FlyCapture context to query.
cameraProperty	A FlyCaptureProperty indicating which property to query.
bOnePush	A bool indicating if one push should be enabled.
bOnOff	A bool indicating if the property should be on or off.
bAuto	A bool indicating if the property should be automatically controlled by the camera.
fValue	A float containing the new value of the parameter.

Return Value
A FlyCaptureError indicating the success or failure of the operation.

Remarks
This function will set the given property for all the cameras on the 1394 bus that are associated with the context passed. If multiple busses (i.e. more than one 1394 card) exist, a call to this function must be made for each bus using a context representing a camera on that bus.

2.2.24. flycaptureSetCameraAbsPropertyEx

Allows the user to set the absolute value of the given parameter if the mode is supported. This function also allows the user to specify the one push, on/off, and auto settings of the same property.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureSetCameraAbsPropertyEx(
                                FlyCaptureContext  context,
                                FlyCaptureProperty cameraProperty,
                                bool               bOnePush,
                                bool               bOnOff,
                                bool               bAuto,
                                float              fValue )
```

Parameters

context	The FlyCapture context to query.
cameraProperty	A FlyCaptureProperty indicating which property to query.
bOnePush	A bool indicating if one push should be enabled.
bOnOff	A bool indicating if the property should be on or off.
bAuto	A bool indicating if the property should be automatically controlled by the camera.
fValue	A float containing the new value of the parameter.

Return Value
A FlyCaptureError indicating the success or failure of the operation.

2.2.25. flycaptureSetCameraProperty

Allows the user to set the given property.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureSetCameraProperty(
                            FlyCaptureContext  context,
                            FlyCaptureProperty cameraProperty,
                            long               lValueA,
                            long               lValueB,
                            bool               bAuto )
```

Parameters

context	The FlyCaptureContext to set the properties in.
cameraProperty	A FlyCaptureProperty indicating the property to set.
lValueA	A long containing the “A”, or first new value of the property.
lValueB	A long containing the “B”, or second new value of the property.
bAuto	A boolean containing the new 'auto' state of the property.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
Calling this function with either of FLYCAPTURE_SOFTWARE_WHITEBALANCE as the cameraProperty parameter and 'true' for the bAuto parameter will invoke a single shot auto white balance method. The assumption is that flycaptureGrabImage() has been called previously with a white object centered in the field of view. This will only work if the camera is a color camera and in RGB mode. The Red and Blue whitebalance parameters only affect cameras that do offboard color calculation such as the Dragonfly.

2.2.26. flycaptureSetCameraPropertyBroadcast

Allows the user to set the given property for all cameras on the bus.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureSetCameraPropertyBroadcast(
                                    FlyCaptureContext  context,
                                    FlyCaptureProperty cameraProperty,
                                    long               lValueA,
                                    long               lValueB,
                                    bool               bAuto )
```

Parameters

context	The FlyCaptureContext to set the properties in.
cameraProperty	A FlyCaptureProperty indicating the property to set.
lValueA	A long containing the “A”, or first new value of the property.
lValueB	A long containing the “B”, or second new value of the property.
bAuto	A boolean containing the new 'auto' state of the property.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
This function will set the given property for all the cameras on the 1394 bus. If you are using multiple busses (ie, more than one 1394 card) you must call this function for each bus, on a context representing a camera on that bus.

2.2.27. flycaptureSetCameraPropertyBroadcastEx

Replaces flycaptureSetCameraPropertyBroadcast() and provides better access to camera features.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureSetCameraPropertyBroadcastEx(
                                      FlyCaptureContext    context,
                                      FlyCaptureProperty   cameraPrope
rty,
                                      bool                 bOnePush,
                                      bool                 bOnOff,
                                      bool                 bAuto,
                                      int                  iValueA,
                                      int                  iValueB )
```

Parameters

context	The FlyCaptureContext to set the properties in.
cameraProperty	A FlyCaptureProperty indicating the property to set.

bOnePush	Set the one push bit.
bOnOff	Set the on/off bit.
bAuto	Set the auto bit.
iValueA	The value to set.
iValueB	The secondary value to set. (only used for the two whitebalance values.)

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
This function will set the given property for all the cameras on the 1394 bus. If you are using multiple busses (ie, more than one 1394 card) you must call this function for each bus, on a context representing a camera on that bus.

2.2.28. flycaptureSetCameraPropertyEx

Replaces flycaptureSetCameraPropertyEx() and provides better access to camera features.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureSetCameraPropertyEx(
                              FlyCaptureContext    context,
                              FlyCaptureProperty   cameraProperty,
                              bool                 bOnePush,
                              bool                 bOnOff,
                              bool                 bAuto,
                              int                  iValueA,
                              int                  iValueB )
```

Parameters

context	The FlyCaptureContext to set the properties in.
cameraProperty	A FlyCaptureProperty indicating the property to set.
bOnePush	Set the one push bit.
bOnOff	Set the on/off bit.
bAuto	Set the auto bit.
iValueA	The value to set.
iValueB	The secondary value to set. (only used for the two whitebalance values.)

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.2.29. flycaptureSetCameraRegister

This function allows the user to set any of the camera's registers.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureSetCameraRegister(
                            FlyCaptureContext context,
                            unsigned long     ulRegister,
                            unsigned long     ulValue )
```

Parameters

context	The FlyCaptureContext associated with the camera to be queried.
ulRegister	The 32 bit register location to set.
ulValue	The 32 bit value to store in the register.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
The ulRegister value is actually an offset applied to a base address. Typically this base addess is 0xFFFFF0F00000 but it is not constant. Refer to the “Unit Dependent Directory” section of the DCAM spec for more information.

2.2.30. flycaptureSetCameraRegisterBroadcast

This function allows the user to set any register for all cameras on the bus.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureSetCameraRegisterBroadcast(
                                     FlyCaptureContext context,
                                     unsigned long     ulRegister,
                                     unsigned long     ulValue )
```

Parameters

context	The FlyCaptureContext associated with the camera to be queried.
ulRegister	The 32 bit register location to set.
ulValue	The 32 bit value to store in the register.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
The ulRegister value is actually an offset applied to a base address. Typically this base addess is 0xFFFFF0F00000 but it is not constant. Refer to the “Unit Dependent Directory” section of the DCAM spec for more information.

2.2.31. flycaptureSetCameraTrigger

Deprecated. Please use flycaptureSetTrigger().

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureSetCameraTrigger(
                            FlyCaptureContext context,
                            unsigned int      uiOnOff,
                            unsigned int      uiPolarity,
                            unsigned int      uiTriggerMode )
```

Return Value
FLYCAPTURE_DEPRECATED.

2.2.32. flycaptureSetCameraTriggerBroadcast

Deprecated. Please use flycaptureSetTriggerBroadcast().

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureSetCameraTriggerBroadcast(
                                     FlyCaptureContext context,
                                     unsigned char     ucOnOff,
                                     unsigned char     ucPolarity,
                                     unsigned char     ucTriggerMode )
```

Return Value
FLYCAPTURE_DEPRECATED.

2.2.33. flycaptureSetLookUpTableChannel

This function will set the specified look up table on the camera.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureSetLookUpTableChannel(
                                 FlyCaptureContext   context,
                                 unsigned int        uiChannel,
                                 const unsigned int* puiArray )
```

Parameters

context	The context associated with the camera.
uiChannel	The channel to set
puiArray	a valid array of “numberOfEntries” unsigned ints with values less than 2^(“bitDepth”)

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.2.34. flycaptureSetStrobe

This function allows the user to set the state of one of the camera's strobe sources. Only for use with cameras which support DCAM v1.31 compliant strobes.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureSetStrobe(
                    FlyCaptureContext context,
                    int               iSource,
                    bool              bOnOff,
                    int               iPolarity,
                    int               iDelay,
                    int               iDuration )
```

Parameters

context	The context associated with the camera to be queried.
iSource	The strobe source to be set.
bOnOff	Describes whether to turn the strobe on or off.
iPolarity	The polarity of the strobe. 1 or 0.
iDelay	The delay of the strobe.
iDuration	The duration of the strobe.

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.2.35. flycaptureSetStrobeBroadcast

This function duplicates the functionality of flycaptureSetStrobe() but broadcasts the settings to all cameras on the bus. Only for use with cameras which support DCAM v1.31 compliant strobes.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureSetStrobeBroadcast(
                    FlyCaptureContext context,
                    int               iSource,
                    bool              bOnOff,
                    int               iPolarity,
                    int               iDelay,
                    int               iDuration )
```

Parameters

context	The context associated with the camera to be queried.
iSource	The strobe source to be set.
bOnOff	Describes whether to turn the strobe on or off.

iPolarity	The polarity of the strobe. 1 or 0.
iDelay	The delay of the strobe.
iDuration	The duration of the strobe.

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.2.36. flycaptureSetTrigger

This function allows the user to set the state of the camera's trigger functionality. THis function replaces the deprecated flycaptureSetCameraTrigger() function.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureSetTrigger(
                    FlyCaptureContext context,
                    bool              bOnOff,
                    int               iPolarity,
                    int               iSource,
                    int               iMode,
                    int               iParameter )
```

Parameters

context	The context associated with the camera to be queried.
bOnOff	Turn the trigger on or off.
iPolarity	The polarity of the trigger. 1 or 0.
iSource	The new trigger source. Corresponds to the source mask.
iMode	The new trigger mode. Corresponds to the mode mask.
iParameter	The (optional) parameter to the trigger function, if required.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
If you have set a grab timeout using flycaptureSetGrabTimeoutEx(), this timeout will be used in asynchronous trigger mode as well: flycaptureGrabImage*() will return with the image when you either trigger the camera, or the timeout value expires.

2.2.37. flycaptureSetTriggerBroadcast

This function duplicates the functionality of flycaptureSetTrigger, except it broadcasts changes to all cameras on the bus.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
```

```
flycaptureSetTriggerBroadcast(
                                FlyCaptureContext context,
                                bool              bOnOff,
                                int               iPolarity,
                                int               iSource,
                                int               iMode,
                                int               iParameter )
```

Parameters

context	The context associated with the camera to be queried.
bOnOff	Turn the trigger on or off.
iPolarity	The polarity of the trigger. 1 or 0.
iSource	The new trigger source. Corresponds to the source mask.
iMode	The new trigger mode. Corresponds to the mode mask.
iParameter	The (optional) parameter to the trigger function, if required.

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.2.38. flycaptureWriteRegisterBlock

Provides block-write (asynchronous) access to the entire register space of the camera.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureWriteRegisterBlock(
                              FlyCaptureContext      context,
                              unsigned short         usAddrHigh,
                              unsigned long          ulAddrLow,
                              const unsigned long*   pulBuffer,
                              unsigned long          ulLength )
```

Parameters

context	The context associated with the camera to be accessed.
usAddrHigh	The top 16 bits of the 48-bit absolute address to write.
ulAddrLow	The bottom 32 bits of the 48-bit absolute addresss to write.
pulBuffer	The buffer that contains the data to be written.
ulLength	The length, in quadlets, of the block to write.

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.3. Construction/Destruction Functions

2.3.1. flycaptureCreateContext

This function creates a FlyCaptureContext and allocates all of the memory that it requires. The purpose of the FlyCaptureContext is to act as a handle to one of the cameras attached to the system. This call must be made before any other calls involving the context will work.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureCreateContext(
                        FlyCaptureContext* pContext )
```

Parameters

pContext	A pointer to the FlyCaptureContext to be created.

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.3.2. flycaptureDestroyContext

Destroys the given FlyCaptureContext. In order to prevent memory leaks from occurring, this function must be called when the user is finished with the FlyCaptureContext.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureDestroyContext(
                         FlyCaptureContext context )
```

Parameters

context	The FlyCaptureContext to be destroyed.

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.3.3. flycaptureGetBusSpeed

This function gets the current asynchronous and isochronous bus speeds. Asynchronous data transmission is primarily register reads and writes. Isochronous data transmission is reserved for image transmission.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureGetBusSpeed(
                      FlyCaptureContext   context,
                      FlyCaptureBusSpeed* pAsyncBusSpeed,
                      FlyCaptureBusSpeed* pIsochBusSpeed )
```

Parameters

context	The FlyCaptureContext associated with the camera to be queried.
pAsyncBusSpeed	The current asynchronous bus speed.
pIsochBusSpeed	The current isochronous bus speed.

Return Value
A FlyCaptureError indicating the success or failure of the function.

See Also
flycaptureSetBusSpeed()

2.3.4. flycaptureGetCameraInfo

Retrieves information about the camera.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureGetCameraInfo(
                        FlyCaptureContext context,
                        FlyCaptureInfoEx* pInfo )
```

Parameters

context	The FlyCaptureContext associated with the camera.
pInfo	Receives the camera information.

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.3.5. flycaptureInitialize

This function initializes one of the cameras on the bus and associates it with the provided FlyCaptureContext. This call must be made after a flycaptureCreateContext() command and prior to a flycaptureStart() command in order for images to be grabbed. Users can also use the flycaptureInitializeFromSerialNumber() command to initialize a context with a specific serial number.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureInitialize(
                     FlyCaptureContext context,
                     unsigned long     ulDevice )
```

Parameters

context	The FlyCaptureContext to be associated with the camera being initialized.
ulDevice	The device index of the FlyCapture camera to be initialized (as indicated by flycaptureBusEnumerateCamerasEx()).

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
If there is only one device on the bus, its index is generally 0.

See Also
flycaptureInitializeFromSerialNumber(), flycaptureCreateContext(), flycaptureStart(), flycaptureBusEnumerateCamerasEx()

2.3.6. flycaptureInitializeFromSerialNumber

Similar to the flycaptureInitialize() command, this function initializes one of the cameras on the bus and associates it with the given FlyCaptureContext. This function differs from its counterpart in that it takes a serial number rather than a bus index.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureInitializeFromSerialNumber(
                                    FlyCaptureContext            conte
xt,
                                    FlyCaptureCameraSerialNumber seria
lNumber )
```

Parameters

context	The FlyCaptureContext to be associated with the camera being initialized.
serialNumber	The serial number of the FlyCapture camera system to be initialized.

Return Value
A FlyCaptureError indicating the success or failure of the function.

See Also
flycaptureInitialize(), flycaptureCreateContext(), flycaptureStart()

2.3.7. flycaptureSetBusSpeed

This function sets the asynchronous and isochronous transmit and receive bus speeds.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureSetBusSpeed(
                     FlyCaptureContext  context,
                     FlyCaptureBusSpeed asyncBusSpeed,
                     FlyCaptureBusSpeed isochBusSpeed )
```

Parameters

context	The FlyCaptureContext associated with the camera to be queried.
asyncBusSpeed	The desired asynchronous data communication speed.
isochBusSpeed	The desired isochronous data communication speed.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
If only one of asyncBusSpeed or isochBusSpeed is required, set the other parameter to FLYCAPTURE_ANY.

See Also
flycaptureGetBusSpeed()

2.4. Construction/Destruction

2.4.1. flycaptureInitializePlus

Identical behaviour to flycaptureInitialize(), except that the user has the option of specifying the number of buffers to use, and optionally allocate those buffers outside the library.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureInitializePlus(
                        FlyCaptureContext   context,
                        unsigned long       ulBusIndex,
                        unsigned long       ulNumBuffers,
                        unsigned char**     arpBuffers )
```

Parameters

context	The context associated with the camera to be accessed.
ulBusIndex	The zero-based device index of the camera to be initialized.
ulNumBuffers	The number of buffers to expect or allocate. For lock next mode, the minimum number of buffers is 2. For lock latest mode, the minimum number of buffers is 4. The maximum number of buffers is only limited by system memory.
arpBuffers	An array of pointers to buffers. If this argument is NULL the library will allocate and free the buffers internally, otherwise the caller is responsible for allocation and deallocation. No boundary checking is done on these images, if you are supplying your own buffers, they must be large enough to hold the largest image you are expecting. When allocating your own buffers, you must take padding into account. The maximum amount of padding required is 1 packet, which can be up to 4096 bytes for 1394a and 8192 bytes for 1394b. Adding this padding to the image size will ensure the buffer is large enough to accomodate the image.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
If you wish to use the camera serial number to initialize, or you don't care about the number of buffers being allocated, use either of the other initialize methods in pgrflycapture.h.

See Also
flycaptureInitialize()

2.5. Control Functions

2.5.1. flycaptureCheckVideoMode

This function allows the user to check if a given mode is supported by the camera.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureCheckVideoMode(
                          FlyCaptureContext   context,
                          FlyCaptureVideoMode videoMode,
                          FlyCaptureFrameRate frameRate,
                          bool*               pbSupported )
```

Parameters

context	An initialized FlyCaptureContext.
videoMode	The video mode to check.
frameRate	The frame rate to check.
pbSupported	A pointer to a bool that will store whether or not the mode is supported.

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.5.2. flycaptureGetColorProcessingMethod

This function allows users to check the current color processing method.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureGetColorProcessingMethod(
                                    FlyCaptureContext        context,
                                    FlyCaptureColorMethod*   pMethod )
```

Parameters

context	The FlyCapture context to access.
pMethod	A pointer to a FlyCaptureColorMethod that will store the current color processing method.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks

This function is only applicable when using the SDK and driver with cameras that do not do on board color processing. See the definition of FlyCaptureColorMethod for detailed descriptions of the available modes.

See Also
flycaptureSetColorProcessingMethod()

2.5.3. flycaptureGetColorTileFormat

This function allows users to check the current color tile destippling format.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureGetColorTileFormat(
                              FlyCaptureContext          context,
                              FlyCaptureStippledFormat*  pformat )
```

Parameters

context	The FlyCapture context to access.
pformat	A pointer to a FlyCaptureStippledFormat that will store the current color tile format.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
The color tile format indicates the format of the stippled image the camera returns. This function is only applicable to cameras that do not do onboard color processing.

See Also
flycaptureSetColorTileFormat()

2.5.4. flycaptureGetCurrentVideoMode

This function allows the user to request the camera's current video mode and frame rate.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureGetCurrentVideoMode(
                              FlyCaptureContext    context,
                              FlyCaptureVideoMode* pVideoMode,
                              FlyCaptureFrameRate* pFrameRate )
```

Parameters

context	An initialized FlyCaptureContext.
pVideoMode	A pointer to a video mode to be filled in.
pFrameRate	A pointer to a frame rate to be filled in.

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.5.5. flycaptureInitializeNotify

Initializes partial image notification mode. Partial image notification allows the grabbing program to be notified several times during a single image grab. Processing on an image can then begin even before the entire image has been acquired. This function must be called after a camera initialization function like flycaptureInitialize() or flycaptureInitializePlus(), and before a start function like flycaptureStartLockNext(). flycaptureLockNextEvent(), flycaptureWaitForImageEvent() and flycaptureUnlockEvent() are the only image acquisition functions that can be used when in partial image notification mode. Please see the ImageEventEx example for more information.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureInitializeNotify(
                          FlyCaptureContext    context,
                          unsigned long        ulNumEvents,
                          FlyCaptureImageEvent arpEvents[] )
```

Parameters

context	The context associated with the camera to be accessed.
ulNumEvents	The number of desired image events per image. The maximum number of events is camera-dependant.
arpEvents	An array of uiNumEvents event structures. The uiSizeBytes member must be filled, which indicates which portion of the image each event is for. The image portions need not be equal sized. No other members need to be filled.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
This will not work unless you have a hotfix to the 1394 subsystem from Microsoft. Please see PGR knowledge base article 153, http://www.ptgrey.com/support/kb/details.asp?id=153, for more information. Partial image notification is not available for "lock latest" functionality. This is PGR bug 2126.

See Also
flycaptureGetPacketInfo(), flycaptureGetCustomImagePacketInfo()

2.5.6. flycaptureQueryCustomImage

This function queries the options available for the advanced Custom Image or DCAM Format 7 functionality.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureQueryCustomImage(
                          FlyCaptureContext context,
                          unsigned int      uiMode,
                          bool*             pbAvailable,
                          unsigned int*     puiMaxImagePixelsWidth,
                          unsigned int*     puiMaxImagePixelsHeight,
                          unsigned int*     puiPixelUnitHorz,
```

```
                                unsigned int*     puiPixelUnitVert,
                                unsigned int*     puiPixelFormats )
```

Parameters

context	The FlyCaptureContext to start grabbing.
uiMode	The mode to query (0-7).
pbAvailable	Indicates the availability of this mode.
puiMaxImagePixelsWidth	Maximum horizonal pixels.
puiMaxImagePixelsHeight	Maximum vertical pixels.
puiPixelUnitHorz	Indicates the horizonal “step size” of the custom image.
puiPixelUnitVert	Indicates the vertical “step size” of the custom image.
puiPixelFormats	A bit field indicating the supported pixel formats of this mode.

Return Value
A FlyCaptureError indicating the success or failure of the function.

See Also
flycaptureStartCustomImage()

2.5.7. flycaptureQueryCustomImageEx

This function queries the options available for the advanced Custom Image or DCAM Format 7functionality. This function differs from flycaptureStartCustomImage() in that it allows the user to retrieve the offset unit size as well (which may be different than the image unit size).

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureQueryCustomImageEx(
                            FlyCaptureContext context,
                            unsigned int      uiMode,
                            bool*             pbAvailable,
                            unsigned int*     puiMaxImagePixelsWidth,
                            unsigned int*     puiMaxImagePixelsHeight,
                            unsigned int*     puiPixelUnitHorz,
                            unsigned int*     puiPixelUnitVert,
                            unsigned int*     puiOffsetUnitHorz,
                            unsigned int*     puiOffsetUnitVert,
                            unsigned int*     puiPixelFormats )
```

Parameters

context	The FlyCaptureContext to start grabbing.
uiMode	The mode to query (0-7).
pbAvailable	Indicates the availability of this mode.
puiMaxImagePixelsWidth	Maximum horizonal pixels.

puiMaxImagePixelsHeight	Maximum vertical pixels.
puiPixelUnitHorz	Indicates the horizontal “step size” of the custom image.
puiPixelUnitVert	Indicates the vertical “step size” of the custom image.
puiOffsetUnitHorz	Indicates the horizontal “step size” of the offset in the custom image.
puiOffsetUnitVert	Indicates the vertical “step size” of the offset in the custom image.
puiPixelFormats	A bit field indicating the supported pixel formats of this mode.

Return Value
A FlyCaptureError indicating the success or failure of the function.

See Also
flycaptureStartCustomImage()

2.5.8. flycaptureSetColorProcessingMethod

This function allows users to select the method used for color processing.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureSetColorProcessingMethod(
                                    FlyCaptureContext        context,
                                    FlyCaptureColorMethod    method )
```

Parameters

context	The FlyCapture context to access.
method	A variable of type FlyCaptureColorMethod indicating the color processing method to be used.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
This function is only applicable when using the SDK and driver with cameras that do not do on board color processing. See the definition of FlyCaptureColorMethod for detailed descriptions of the available modes.

See Also
flycaptureGetColorProcessingMethod()

2.5.9. flycaptureSetColorTileFormat

This function sets the color tile destippling format.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureSetColorTileFormat(
                              FlyCaptureContext        context,
                              FlyCaptureStippledFormat format )
```

Parameters

context	The FlyCapture context to access.
format	The FlyCaptureStippledFormat to set.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
The color tile format indicates the format of the stippled image the camera returns. This function is only applicable to cameras that do not do onboard color processing.

See Also
flycaptureGetColorTileFormat()

2.5.10. flycaptureStart

This function starts the image grabbing process. It should be called after flycaptureCreateContext() and flycaptureInitialize().

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureStart(
                FlyCaptureContext   context,
                FlyCaptureVideoMode videoMode,
                FlyCaptureFrameRate frameRate )
```

Parameters

context	The FlyCaptureContext to start grabbing.
videoMode	The video mode to start the camera in.
frameRate	The frame rate to start the camera at.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
It is during this and other related start calls where driver level image buffer allocation occurs.

See Also
flycaptureCreateContext(), flycaptureInitialize(), flycaptureInitializeFromSerialNumber(), flycaptureStartCustomImage(), flycaptureStop()

2.5.11. flycaptureStartCustomImage

This function starts the image grabbing process with "custom image" (DCAM Format 7) functionality, which allows the user to select a custom image size and/or region of interest.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureStartCustomImage(
                           FlyCaptureContext       context,
```

```
                                unsigned int            uiMode,
                                unsigned int            uiImagePosLeft,
                                unsigned int            uiImagePosTop,
                                unsigned int            uiWidth,
                                unsigned int            uiHeight,
                                float                   fBandwidth,
                                FlyCapturePixelFormat   format )
```

Parameters

context	The FlyCaptureContext to start grabbing.
uiMode	The camera-specific mode. (0-7).
uiImagePosLeft	The left position of the (sub)image.
uiImagePosTop	Top top position of the (sub)image.
uiWidth	The width of the (sub)image.
uiHeight	The height of the (sub)image.
fBandwidth	A number between 1.0 and 100.0 which represents the percentage of the camera's maximum bandwidth to use for transmission.
format	The pixel format to be used.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
It is during this and other related start calls where driver level image buffer allocation occurs.

See Also
flycaptureStartCustomImagePacket(), flycaptureQueryCustomImage()

2.5.12. flycaptureStartCustomImagePacket

This function is identical to flycaptureStartCustomImage() except it takes in a packet size rather than a float bandwidth parameter.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureStartCustomImagePacket(
                                FlyCaptureContext       context,
                                unsigned long           ulMode,
                                unsigned long           ulImagePosLeft
,
                                unsigned long           ulImagePosTop,
                                unsigned long           ulWidth,
                                unsigned long           ulHeight,
                                unsigned long           ulPacketSizeBy
tes,
                                FlyCapturePixelFormat   format )
```

Parameters

context	The FlyCaptureContext to start grabbing.
ulMode	The camera-specific mode. (0-7).
ulImagePosLeft	The left position of the (sub)image.
ulImagePosTop	The top position of the (sub)image.
ulWidth	The width of the (sub)image.
ulHeight	The height of the (sub)image.
ulPacketSizeBytes	The number of packets to send per isochronous period. A larger packet size will result in faster image transmission and increased bandwidth requirements. This number should be a multiple of 4 and fit within the values defined by flycaptureGetCustomImageMaxPacketSize().
format	The pixel format to be used.

Return Value
A FlyCaptureError indicating the success or failure of the function.

See Also
flycaptureStartCustomImage(), flycaptureQueryCustomImage(), flycaptureGetCustomImagePacketInfo()

2.5.13. flycaptureStartLockNext

Starts the camera streaming and initializes the library for "lock next" functionality. This function needs to used instead of flycaptureStart() for the following "lock next" functions.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureStartLockNext(
                        FlyCaptureContext    context,
                        FlyCaptureVideoMode  videoMode,
                        FlyCaptureFrameRate  frameRate )
```

Parameters

context	The context associated with the camera to be started.
videoMode	The video mode to start the camera in.
frameRate	The frame rate to start the camera at.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
For "lock latest" functionality, use flycaptureStart() and the flycaptureLockLatest().

2.5.14. flycaptureStartLockNextCustomImage

This function is identical to flycaptureStartLockNext(), except that it will start the camera in custom image mode. See flycaptureStartCustomImage().

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureStartLockNextCustomImage(
                                    FlyCaptureContext     context,
                                    unsigned long         ulMode,
                                    unsigned long         ulImagePosLeft
,
                                    unsigned long         ulImagePosTop,
                                    unsigned long         ulWidth,
                                    unsigned long         ulHeight,
                                    float                 fBandwidth,
                                    FlyCapturePixelFormat format )
```

Parameters

context	The context associated with the camera to be started.
ulMode	The camera-specific mode. (0-7).
ulImagePosLeft	The left position of the (sub)image.
ulImagePosTop	The top position of the (sub)image.
ulWidth	The width of the (sub)image.
ulHeight	The height of the (sub)image.
fBandwidth	The bandwidth to assign to this camera. 100.0 indicates full bandwidth.
format	The pixel format to be used.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
For “lock latest” functionality, use flycaptureStart() and the flycaptureLockLatest().

See Also
flycaptureQueryCustomImage(), flycaptureStartCustomImage().

2.5.15. flycaptureStartLockNextCustomImagePacket

This function is identical to flycaptureStartLockNextCustomImage(), except that it takes a packet size in bytes, instead of a floating point bandwidth estimation.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureStartLockNextCustomImagePacket(
                                          FlyCaptureContext     context,
                                          unsigned long         ulMode,
```

```
                                                unsigned long         ulImageP
osLeft,
                                                unsigned long         ulImageP
osTop,
                                                unsigned long         ulWidth,
                                                unsigned long         ulHeight
,
                                                unsigned long         ulPacket
SizeBytes,
                                                FlyCapturePixelFormat format )
```

Parameters

context	The context associated with the camera to be started.
ulMode	The camera-specific mode. (0-7).
ulImagePosLeft	The left position of the (sub)image.
ulImagePosTop	The top position of the (sub)image.
ulWidth	The width of the (sub)image.
ulHeight	The height of the (sub)image.
ulPacketSizeBytes	The number of packets to send per isochronous period. A larger packet size will result in faster image transmission and increased bandwidth requirements. This number should be a multiple of 4 and fit within the values defined by flycaptureGetCustomImageMaxPacketSize().
format	The pixel format to be used.

Return Value
A FlyCaptureError indicating the success or failure of the function.

See Also
flycaptureQueryCustomImage(), flycaptureStartCustomImage().

2.5.16. flycaptureStop

This function halts all image grabbing for the specified FlyCaptureContext.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureStop(
               FlyCaptureContext context )
```

Parameters

context	The FlyCaptureContext to stop.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
This function invalidates all buffers returned by flycaptureLockNext() and flycaptureLockLatest().

2.6. General Functions

2.6.1. flycaptureErrorToString

This function returns a description of the provided FlyCaptureError.

Declaration

```
const char* PGRFLYCAPTURE_CALL_CONVEN
flycaptureErrorToString(
                        FlyCaptureError error )
```

Parameters

error	The FlyCapture error to be parsed.

Return Value
A null-terminated character string that describes the FlyCapture error.

2.6.2. flycaptureGetLibraryVersion

This function returns the version of the library defined at the top of this header file (PGRFLYCAPTURE_VERSION), which is in the format 100*(major version)+(minor version).

Declaration

```
int PGRFLYCAPTURE_CALL_CONVEN
flycaptureGetLibraryVersion()
```

Return Value
An integer indicating the current version of the library.

2.6.3. flycaptureRegisterToString

This function returns a description of the provided register number.

Declaration

```
const char* PGRFLYCAPTURE_CALL_CONVEN
flycaptureRegisterToString(
                           unsigned long ulRegister  )
```

Parameters

ulRegister	The register to be translated.

Return Value
A null-terminated character string that describes the register.

2.7. Image Related Functions

2.7.1. flycaptureConvertImage

Convert an arbitrary image format to another format.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureConvertImage(
                      FlyCaptureContext        context,
                      const FlyCaptureImage*   pimageSrc,
                      FlyCaptureImage*         pimageDest )
```

Parameters

context	The FlyCapture context to access.
pimageSrc	The source image to convert
pimageDest	The destination image to convert. The pData member must be initialized to an output buffer of sufficient size, and the pixelFormat member indicates the desired output format. Only BGR and BGRU are currently supported.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
This function replaces flycaptureConvertToBGR24(), flycaptureStippledToBGR24(), and flycaptureStippledToBGRU32().

2.7.2. flycaptureGetCustomImagePacketInfo

Returns isochronous packet size information for the indicated custom image mode and image size. The maximum packet size is useful for determining a minimum image event notification size. This function is very similar to flycaptureGetPacketSize() but should be used when dealing with custom image modes.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureGetCustomImagePacketInfo(
                                  FlyCaptureContext     context,
                                  unsigned long         ulMode,
                                  unsigned long         ulWidth,
                                  unsigned long         ulHeight,
                                  FlyCapturePixelFormat format,
                                  FlyCapturePacketInfo* pinfo )
```

Parameters

context	The context associated with the camera to be accessed.
ulMode	The camera-specific mode. (0-7).

ulWidth	The width of the (sub)image.
ulHeight	The height of the (sub)image.
format	The pixel format to be used.
pinfo	Returned packet size information.

Return Value
A FlyCaptureError indicating the success or failure of the function.

See Also
flycaptureGetPacketInfo(), flycaptureStartCustomImagePacket()

2.7.3. flycaptureGetImageFilters

Retrieves the currently active filters. The returned number is a bitmap corresponding to the FLYCAPTURE_IMAGE_FILTER_* values.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureGetImageFilters(
                          FlyCaptureContext   context,
                          unsigned int*       puiFilters )
```

Parameters

context	The context associated with the camera to be accessed.
puiFilters	The filter bitmap is returned in this value.

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.7.4. flycaptureGetImageTimestamping

Retrieves the status of camera-generated image timestamping.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureGetImageTimestamping(
                               FlyCaptureContext  context,
                               bool*              pbOn )
```

Parameters

context	The context associated with the camera to be accessed.
pbOn	Whether or not the camera is producing image timestamps.

Return Value

Value	Meaning
FLYCAPTURE_OK	If the time stamping status was read correctly.

FLYCAPTURE_NOT_IMPLEMENTED	If the camera does not support image timestamping.

2.7.5. flycaptureGetPacketInfo

Returns the isochronous packet size for the indicated video mode and frame rate. This number is useful when deciding the amount of data for each image event notification. The size of each image event has to be a multiple of the packet size. It is also useful for determining the amount of bandwidth required to run a camera at a given mode and frame rate.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureGetPacketInfo(
                        FlyCaptureContext       context,
                        FlyCaptureVideoMode     videoMode,
                        FlyCaptureFrameRate     frameRate,
                        FlyCapturePacketInfo*   pinfo )
```

Parameters

context	The context associated with the camera to be accessed.
videoMode	Required video mode.
frameRate	Required frame rate.
pinfo	Returned packet size information.

Return Value
A FlyCaptureError indicating the success or failure of the function.

See Also
flycaptureGetCustomImagePacketInfo()

2.7.6. flycaptureGrabImage

This function grabs the newest image from the FlyCapture camera system and passes the image buffer and information to the user.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureGrabImage(
                    FlyCaptureContext     context,
                    unsigned char**       ppImageBuffer,
                    int*                  piRows,
                    int*                  piCols,
                    int*                  piRowInc,
                    FlyCaptureVideoMode*  pVideoMode )
```

Parameters

context	The FlyCapture context to lock the image in.
ppImageBuffer	Pointer to the returned image buffer pointer.

piRows	Pointer to the returned rows.
piCols	Pointer to the returned columns.
piRowInc	Pointer to the returned row increment (number of bytes per row.)
pVideoMode	Pointer to the returned video mode.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
This function will block until a new image is available. You can optionally set the timeout value for the wait using the flycaptureSetGrabTimeoutEx() function (by default the wait time is infinite.) Setting the timeout value should normally not be necessary.

See Also
flycaptureStart(), flycaptureGrabImage2(), flycaptureSetGrabTimeoutEx()

2.7.7. flycaptureGrabImage2

This function is identical to flycaptureGrabImage() except that it returns a FlyCaptureImage structure.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureGrabImage2(
                    FlyCaptureContext    context,
                    FlyCaptureImage*     pimage )
```

Parameters

context	The FlyCapture context to lock the image in.
pimage	A pointer to a FlyCaptureImage structure that will contain the image information.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
See remarks for flycaptureGrabImage().

See Also
flycaptureStart(), flycaptureGrabImage(), flycaptureSetGrabTimeoutEx()

2.7.8. flycaptureInplaceRGB24toBGR24

Changes the input image buffer from 24-bit RGB to windows-displayable 24-bit BGR.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureInplaceRGB24toBGR24(
                             unsigned char* pImageBuffer,
                             int            iImagePixels )
```

Parameters

pImageBuffer	Pointer to the image contents.
iImagePixels	Size of the image, in pixels.

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.7.9. flycaptureInplaceWhiteBalance

This function performs an inplace software based white balance on the provided image.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureInplaceWhiteBalance(
                              FlyCaptureContext context,
                              unsigned char*    pData,
                              int               iRows,
                              int               iCols )
```

Parameters

context	The FlyCapture context.
pData	The BGR24 image data.
iRows	Image rows.
iCols	Image columns.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
The image must be in BGR24 format. I.e., the output from one of the above functions. This function has no effect on cameras that are detected to have hardware whitebalance.

2.7.10. flycaptureLockLatest

Lock the “latest” image that has not been seen. If there is an unseen image waiting, this function will return immediately with that image, otherwise it will block until the next image has been received. The difference in the sequence numbers of images returned by consecutive calls to this function indicates the number of missed images between calls.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureLockLatest(
                     FlyCaptureContext    context,
                     FlyCaptureImagePlus* pimage )
```

Parameters

context	The context associated with the camera to be accessed.

pimage	The returned FlyCapturePlus.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
flycaptureUnlock() must be called using the buffer index returned in pimage when processing on this image has been completed. This function behaves identically to flycaptureGrabImage(), except it doesn't implicitly unlock the previously seen image first. The camera must have been started using flycaptureStart() in order for this function to succeed.

2.7.11. flycaptureLockNext

Lock the "next" image that has not been seen. Provided that the previous image processing time is not greater than the time taken for the camera to transmit images to the available unlocked buffers, this function can be called repeatedly to guarantee that each image will be seen. If the camera has not finished transmitting the next image, this function will block. Users can verify image sequentiality by comparing sequence numbers of sequential images.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureLockNext(
                  FlyCaptureContext      context,
                  FlyCapturePlus*        pimage )
```

Parameters

context	The context associated with the camera to be accessed.
pimage	The returned FlyCapturePlus.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
flycaptureUnlock() must be called using the buffer index returned in pimage when processing on this image has been completed. The camera must have been started using flycaptureStartLockNext() for this function to succeed.

2.7.12. flycaptureLockNextEvent

When in partial image notification mode (flycaptureInitializeNotify()), this function will fill an array of FlyCaptureImageEvent structures corresponding to the requested events for each received image. This function will not block.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureLockNextEvent(
                       FlyCaptureContext    context,
                       FlyCaptureImage*     pimage,
                       FlyCaptureImageEvent arpEvents[] )
```

Parameters

context	The context associated with the camera to be accessed.
pimage	The returned FlyCaptureImage corresponding to the image that the events are for.
arpEvents	An array of event structures that will be filled by this function. The number of events in this array must be the number passed in to flycaptureInitializeNotify(). This array can contain the same events that were passed into flycaptureInitializeNotify(), or it can be a new array if you wish to retain ownership of the image buffer.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
This will not work unless you have a hotfix to the 1394 subsystem from Microsoft. Please see PGR knowledge base article 153, http://www.ptgrey.com/support/kb/details.asp?id=153, for more information.

2.7.13. flycaptureParseImageTimestamp

Parses the first 4 bytes of an image generated with image timestamping on to retrieve 1394 timestamp information.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureParseImageTimestamp(
                              FlyCaptureContext    context,
                              const unsigned char* pData,
                              unsigned int*        puiSeconds,
                              unsigned int*        puiCount,
                              unsigned int*        puiOffset )
```

Parameters

context	The context associated with the camera to be accessed.
pData	The image data to be parsed.
puiSeconds	The seconds component of the 1394 timestamp.
puiCount	The count component of the 1394 timestamp.
puiOffset	The offset component of the 1394 timestamp.

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.7.14. flycaptureSaveImage

Writes the specified image buffer to disk.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureSaveImage(
                    FlyCaptureContext          context,
```

```
                        const FlyCaptureImage*      pImage,
                        const char*                 pszPath,
                        FlyCaptureImageFileFormat   format )
```

Parameters

context	The FlyCapture context to access.
pImage	The image to save. This can be populated by the user, by only filling out the pData, size, and pixel format information, or can be the structure returned by flycaptureConvertImage().
pszPath	The name of the file to write to.
format	The file format to write.

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.7.15. flycaptureSetGrabTimeoutEx

This function allows the user to set the timeout value for flycaptureGrabImage*() and flycaptureLockLatest(). This is not normally necessary but can be useful in specific applications. For example, setting uiTimeout to be 0 will result in non-blocking grab call.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureSetGrabTimeoutEx(
                          FlyCaptureContext context,
                          unsigned long     ulTimeout )
```

Parameters

context	The FlyCaptureContext associated with the camera to be queried.
ulTimeout	The timeout value, in milliseconds. A value of FLYCAPTURE_INFINITE indicates an infinite wait. A value of zero indicates a nonblocking grab call.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
The default grab timeout value is “infinite.” It is not normally necessary to set this value.

2.7.16. flycaptureSetImageFilters

Sets the active filters. The returned number is a bitmap corresponding to the FLYCAPTURE_IMAGE_FILTER_* values.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureSetImageFilters(
                          FlyCaptureContext   context,
                          unsigned int        uiFilters )
```

Parameters

context	The context associated with the camera to be accessed.
uiFilters	The filters to set. Use FLYCAPTURE_IMAGE_FILTER_NONE to disable image filtering.

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.7.17. flycaptureSetImageTimestamping

Sets image timestamping. If image timestamping is on, the first 4 bytes of the image will contain camera-generated timestamp information, and the cycle seconds, count, and offset returned in FlyCaptureTimestamp will use the data.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureSetImageTimestamping(
                              FlyCaptureContext  context,
                              bool               bOn )
```

Parameters

context	The context associated with the camera to be accessed.
bOn	On or off flag for image timestamping.

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.7.18. flycaptureSyncForLockNext

Takes in an array of contexts attached to multiple cameras that are already synchronized in hardware and assures that the next time lockNext() is called for all contexts, the images locked will correspond to one another. Note that this function only needs to be called once after the contexts have been started. The contexts should be started in the same order that they are listed in arContexts before this function is called.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureSyncForLockNext(
                         FlyCaptureContext* arContexts,
                         unsigned long      ulContexts )
```

Parameters

arContexts	An array of contexts attached to the cameras to synchronize.
ulContexts	The number of contexts in arContext.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks

This function operates by skipping the appropriate number of images in contexts that were started "after" the reference context (position 0 in the array). If this function fails it does not necessarily mean the cameras are out of sync. This is still experimental. Note also that this function will turn on image timestamping. Please contact PGR support for more information.

2.7.19. flycaptureUnlock

Returns a buffer into the pool to be filled by the camera driver. This must be called for each image locked using the above lock functions after processing on that image has been completed.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureUnlock(
                FlyCaptureContext  context,
                unsigned long      ulBufferIndex )
```

Parameters

context	The context associated with the camera to be accessed.
ulBufferIndex	The buffer to unlock.

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.7.20. flycaptureUnlockAll

Unlocks all locked images. This is equivalent to maintaining a list of locked buffers and calling flycaptureUnlock() for each.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureUnlockAll(
                  FlyCaptureContext  context )
```

Parameters

context	The context associated with the camera to be accessed.

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.7.21. flycaptureUnlockEvent

This function will release ownership of the buffers in the set of event structures. It has the same functionality as flycaptureUnlock(), except that it unlocks the buffers in the correct order.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureUnlockEvent(
                    FlyCaptureContext      context,
                    FlyCaptureImageEvent   arpEvents[] )
```

Parameters

context	The context associated with the camera to be accessed.
arpEvents	An array of event structures that will be unlocked by this function. The number of events in this array must be the number passed in to flycaptureInitializeNotify(). This array should be the same one filled by flycaptureLockNextEvent().

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
This will not work unless you have a hotfix to the 1394 subsystem from Microsoft. Please see PGR knowledge base article 153, http://www.ptgrey.com/support/kb/details.asp?id=153, for more information.

2.7.22. flycaptureWaitForImageEvent

This function waits for a single partial image image event, as defined by the sizes specified by flycaptureInitializeNotify(). If the event has already been triggered (the image part has already been received) this function will return immediately. It is not necessary to call this function for all the events in an image. The events from a single image will be triggered in order. To verify that no images have been missed, call this function on all the events of all the images received and verify the sequence numbers are contiguous.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureWaitForImageEvent(
                            FlyCaptureContext      context,
                            FlyCaptureImageEvent*  pevent,
                            unsigned long          ulTimeout )
```

Parameters

context	The context associated with the camera to be accessed.
pevent	The event structure corresponding to the part of the image to wait for. This should be one of the structures filled in by flycaptureLockNextEvent(). At this point, the sequence number of the event is filled.
ulTimeout	The time, in milliseconds, to wait for the image event to be received. Can be FLYCAPTURE_INFINITE.

Return Value
A FlyCaptureError indicating the success or failure of the function.

Remarks
This will not work unless you have a hotfix to the 1394 subsystem from Microsoft. Please see PGR knowledge base article 153, http://www.ptgrey.com/support/kb/details.asp?id=153, for more information.

2.8. Messaging

2.8.1. flycaptureBusErrorToString

This function provides the user with a mechanism for decoding the error code member returned as part of a FlycaptureMessage FLYCAPTURE_BUS_ERROR. It returns a string containing a description of the provided error.

Declaration

```
const char* PGRFLYCAPTURE_CALL_CONVEN
flycaptureBusErrorToString(
                           unsigned long ulErrorCode  )
```

Parameters

ulErrorCode	The error code to be translated.

Return Value
A string containing a human readable interpretation of the error code.

2.8.2. flycaptureCloseMessaging

This function closes messaging for a specific camera or all buses.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureCloseMessaging(
                         FlyCaptureContext context,
                         ULONG             ulSerialNumber )
```

Parameters

context	The FlyCaptureContext associated with the camera.
ulSerialNumber	The serial number of the camera of which to close messaging for. Use FLYCAPTURE_BUS_MESSAGE as the serial number to close bus event messaging.

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.8.3. flycaptureGetMessageLoggingStatus

This function returns the status of message logging.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureGetMessageLoggingStatus(
                                  FlyCaptureContext context,
                                  bool* pbEnabled )
```

Parameters

context	The FlyCaptureContext associated with the camera.

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.8.4. flycaptureInitializeMessaging

This function initializes messaging for a specific camera or all buses.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureInitializeMessaging(
                              FlyCaptureContext context,
                              ULONG             ulSerialNumber )
```

Parameters

context	The FlyCaptureContext associated with the camera.
ulSerialNumber	The serial number of the camera of which to initialize messaging for. Use FLYCAPTURE_BUS_MESSAGE as the serial number to initialize bus event messages.

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.8.5. flycaptureReceiveMessage

This function is used to receive messages from cameras or bus events.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureReceiveMessage(
                         FlyCaptureContext  context,
                         ULONG              ulSerialNumber,
                         FlyCaptureMessage* pMessage,
                         OVERLAPPED*        polRead )
```

Parameters

context	The FlyCaptureContext associated with the camera.
ulSerialNumber	The serial number of the camera of which to close messaging for. Use FLYCAPTURE_BUS_MESSAGE as the serial number to receive bus event messages.
pMessage	A pointer to the message which will be filled when a message is received.
polRead	A pointer to an overlapped I/O structure. The event handle of this structure is set when a message is received.

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.8.6. flycaptureSetMessageLoggingStatus

This function turns message logging on and off.

Declaration

```
FlyCaptureError PGRFLYCAPTURE_CALL_CONVEN
flycaptureSetMessageLoggingStatus(
                                  FlyCaptureContext context,
                                  bool              bEnable )
```

Parameters

context	The FlyCaptureContext associated with the camera.
bEnable	TRUE turns message logging on, FALSE turns it off.

Return Value
A FlyCaptureError indicating the success or failure of the function.

2.9. Type Definitions

2.9.1. CameraGUIContext

Declaration

```
typedef void*  CameraGUIContext
```

Remarks
A user level handle for the PGRGui object.

2.9.2. CameraGUIError

Declaration

```
enum CameraGUIError
{
   PGRCAMGUI_OK,
   PGRCAMGUI_FAILED,
   PGRCAMGUI_COULD_NOT_CREATE_DIALOG,
   PGRCAMGUI_INVALID_ARGUMENT,
   PGRCAMGUI_INVALID_CONTEXT,
   PGRCAMGUI_MEMORY_ALLOCATION_ERROR,
   PGRCAMGUI_INTERNAL_CAMERA_ERROR,

}
```

Elements

PGRCAMGUI_COULD_NOT_CREATE_DIALOG	Was unable to create dialog.
PGRCAMGUI_FAILED	Function failed.
PGRCAMGUI_INTERNAL_CAMERA_ERROR	There has been an internal camera error - call getLastError()

PGRCAMGUI_INVALID_ARGUMENT	An invalid argument was passed.
PGRCAMGUI_INVALID_CONTEXT	An invalid context was passed.
PGRCAMGUI_MEMORY_ALLOCATION_ERROR	Memory allocation error.
PGRCAMGUI_OK	Function completed successfully.

Remarks
Error codes returned from all PGRCameraGUI functions.

2.9.3. CameraGUIType

Declaration

```
enum CameraGUIType
{
   PGRCAMGUI_TYPE_PGRFLYCAPTURE,
   PGRCAMGUI_TYPE_DIGICLOPS,
   PGRCAMGUI_TYPE_LADYBUG,

}
```

Elements

PGRCAMGUI_TYPE_DIGICLOPS	Digiclops Settings.
PGRCAMGUI_TYPE_LADYBUG	Ladybug settings.
PGRCAMGUI_TYPE_PGRFLYCAPTURE	PGRFlyCapture settings.

Remarks
Type of PGRCameraGUI settings dialog to display. This enum will be deprecated in the next version. Please use the SDK-specific .LIB and .DLL (see note above) and the new pgrcamguiInitializeSettingsDialog() instead of pgrcamguiCreateSettingsDialog().

2.9.4. FlyCaptureBusEvent

An enumeration of the different type of bus events.

Declaration

```
enum FlyCaptureBusEvent
{
   FLYCAPTURE_MESSAGE_BUS_RESET = 0x02,
   FLYCAPTURE_MESSAGE_DEVICE_ARRIVAL,
   FLYCAPTURE_MESSAGE_DEVICE_REMOVAL,

}
```

Elements

FLYCAPTURE_MESSAGE_BUS_RESET	A message returned from the bus callback mechanism indicating a bus reset.
FLYCAPTURE_MESSAGE_DEVICE_ARRIVAL	A message returned from the bus callback mechanism indicating a device has arrived on the bus.
FLYCAPTURE_MESSAGE_DEVICE_REMOVAL	A message returned from the bus callback mechanism indicating a device has been removed from the bus.

2.9.5. FlyCaptureBusSpeed

An enumeration used to describe the bus speed

Declaration

```
enum FlyCaptureBusSpeed
{
   FLYCAPTURE_S100,
   FLYCAPTURE_S200,
   FLYCAPTURE_S400,
   FLYCAPTURE_S800,
   FLYCAPTURE_S1600,
   FLYCAPTURE_S3200,
   FLYCAPTURE_S_FASTEST,
   FLYCAPTURE_ANY,
   FLYCAPTURE_SPEED_UNKNOWN = -1,
   FLYCAPTURE_SPEED_FORCE_QUADLET   = 0x7FFFFFFF,

}
```

Elements

FLYCAPTURE_ANY	Any speed that is available.
FLYCAPTURE_S100	100Mbits/sec.
FLYCAPTURE_S1600	1600Mbits/sec.
FLYCAPTURE_S200	200Mbits/sec.
FLYCAPTURE_S3200	3200Mbits/sec.
FLYCAPTURE_S400	400Mbits/sec.
FLYCAPTURE_S800	800Mbits/sec.
FLYCAPTURE_SPEED_FORCE_QUADLET	Unused member to force this enum to compile to 32 bits.
FLYCAPTURE_SPEED_UNKNOWN	
FLYCAPTURE_S_FASTEST	The fastest speed available.

2.9.6. FlyCaptureCallback

Declaration

```
typedef void __cdecl
FlyCaptureCallback( void* pParam, int iMessage, unsigned long ulParam )
```

Remarks

Function prototype for the bus callback mechanism. pParam contains the parameter passed in when registering the callback. iMessage is one of the above FLYCAPTURE_MESSAGE_* #defines and ulParam is a message-defined parameter.

See Also

flycaptureModifyCallback()

2.9.7. FlyCaptureCameraModel

An enumeration used to describe the different camera models that can be accessed through this SDK.

Declaration

```
enum FlyCaptureCameraModel
{
   FLYCAPTURE_FIREFLY,
   FLYCAPTURE_DRAGONFLY,
   FLYCAPTURE_AIM,
   FLYCAPTURE_SCORPION,
   FLYCAPTURE_TYPHOON,
   FLYCAPTURE_FLEA,
   FLYCAPTURE_DRAGONFLY_EXPRESS,
   FLYCAPTURE_FLEA2,
   FLYCAPTURE_FIREFLY_MV,
   FLYCAPTURE_DRAGONFLY2,
   FLYCAPTURE_UNKNOWN = -1,

   FCCM_FORCE_QUADLET   = 0x7FFFFFFF,
}
```

Elements

FCCM_FORCE_QUADLET	Unused member to force this enum to compile to 32 bits.
FLYCAPTURE_AIM	
FLYCAPTURE_DRAGONFLY	
FLYCAPTURE_DRAGONFLY2	
FLYCAPTURE_DRAGONFLY_EXPRESS	
FLYCAPTURE_FIREFLY	
FLYCAPTURE_FIREFLY_MV	
FLYCAPTURE_FLEA	
FLYCAPTURE_FLEA2	
FLYCAPTURE_SCORPION	
FLYCAPTURE_TYPHOON	
FLYCAPTURE_UNKNOWN	

2.9.8. FlyCaptureCameraSerialNumber

The type used to store the serial number uniquely identifying a FlyCapture camera.

Declaration

```
typedef unsigned long FlyCaptureCameraSerialNumber
```

2.9.9. FlyCaptureCameraType

An enumeration used to describe the different camera color configurations.

Declaration

```
enum FlyCaptureCameraType
{
   FLYCAPTURE_BLACK_AND_WHITE,
   FLYCAPTURE_COLOR

}
```

Elements

FLYCAPTURE_BLACK_AND_WHITE	black and white system.
FLYCAPTURE_COLOR	color system.

2.9.10. FlyCaptureColorMethod

An enumeration used to describe the different color processing methods.

Declaration

```
enum FlyCaptureColorMethod
{
   FLYCAPTURE_DISABLE,
   FLYCAPTURE_EDGE_SENSING,
   FLYCAPTURE_NEAREST_NEIGHBOR,
   FLYCAPTURE_NEAREST_NEIGHBOR_FAST,
   FLYCAPTURE_RIGOROUS,

}
```

Elements

FLYCAPTURE_DISABLE	Disable color processing.
FLYCAPTURE_EDGE_SENSING	Edge sensing de-mosaicing. This is the most accurate method that can still keep up with the camera's frame rate.
FLYCAPTURE_NEAREST_NEIGHBOR	Nearest neighbor de-mosaicing. This algorithm is significantly faster than edge sensing, at the cost of accuracy.
FLYCAPTURE_NEAREST_NEIGHBOR_FAST	Faster, less accurate nearest neighbor de-mosaicing.
FLYCAPTURE_RIGOROUS	Rigorous de-mosaicing. This provides the best quality color reproduction. This method is so processor intensive that it might not keep up with the camera's frame rate. Best used for offline processing where accurate color reproduction is required.

Remarks

This is only relevant for cameras that do not do onboard color processing, such as the Dragonfly. The FLYCAPTURE_RIGOROUS method is very slow and will not keep up with high frame rates.

2.9.11. FlyCaptureContext

Context pointer for the PGRFlyCapture library.

Declaration

```
typedef void* FlyCaptureContext
```

2.9.12. FlyCaptureError

The error codes returned by the functions in this library.

Declaration

```
enum FlyCaptureError
{
   FLYCAPTURE_OK,
   FLYCAPTURE_FAILED,
   FLYCAPTURE_INVALID_ARGUMENT,
   FLYCAPTURE_INVALID_CONTEXT,
   FLYCAPTURE_NOT_IMPLEMENTED,
   FLYCAPTURE_ALREADY_INITIALIZED,
   FLYCAPTURE_ALREADY_STARTED,
   FLYCAPTURE_CALLBACK_NOT_REGISTERED,
   FLYCAPTURE_CALLBACK_ALREADY_REGISTERED,
   FLYCAPTURE_CAMERACONTROL_PROBLEM,
   FLYCAPTURE_COULD_NOT_OPEN_FILE,
   FLYCAPTURE_COULD_NOT_OPEN_DEVICE_HANDLE,
   FLYCAPTURE_MEMORY_ALLOC_ERROR,
   FLYCAPTURE_NO_IMAGE,
   FLYCAPTURE_NOT_INITIALIZED,
   FLYCAPTURE_NOT_STARTED,
   FLYCAPTURE_MAX_BANDWIDTH_EXCEEDED,
   FLYCAPTURE_NON_PGR_CAMERA,
   FLYCAPTURE_INVALID_MODE,
   FLYCAPTURE_ERROR_UNKNOWN,
   FLYCAPTURE_INVALID_CUSTOM_SIZE,
   FLYCAPTURE_TIMEOUT,
   FLYCAPTURE_TOO_MANY_LOCKED_BUFFERS,
   FLYCAPTURE_VERSION_MISMATCH,
   FLYCAPTURE_DEVICE_BUSY,
   FLYCAPTURE_DEPRECATED,
   FLYCAPTURE_BUFFER_SIZE_TOO_SMALL,

}
```

Elements

FLYCAPTURE_ALREADY_INITIALIZED	Device already initialized.
FLYCAPTURE_ALREADY_STARTED	Grabbing has already been started.
FLYCAPTURE_BUFFER_SIZE_TOO_SMALL	Supplied User Buffer is too small.
FLYCAPTURE_CALLBACK_ALREADY_REGISTERED	Callback is already registered
FLYCAPTURE_CALLBACK_NOT_REGISTERED	Callback is not registered
FLYCAPTURE_CAMERACONTROL_PROBLEM	Problem controlling camera.
FLYCAPTURE_COULD_NOT_OPEN_DEVICE_HANDLE	Failed to open a device handle.
FLYCAPTURE_COULD_NOT_OPEN_FILE	Failed to open file.
FLYCAPTURE_DEPRECATED	Function has been deprecated. Please see documentation.
FLYCAPTURE_DEVICE_BUSY	The camera responded that it is

	currently busy.
FLYCAPTURE_ERROR_UNKNOWN	Unknown error.
FLYCAPTURE_FAILED	General failure.
FLYCAPTURE_INVALID_ARGUMENT	Invalid argument passed.
FLYCAPTURE_INVALID_CONTEXT	Invalid context passed.
FLYCAPTURE_INVALID_CUSTOM_SIZE	Invalid custom size.
FLYCAPTURE_INVALID_MODE	Invalid video mode or framerate passed or retrieved.
FLYCAPTURE_MAX_BANDWIDTH_EXCEEDED	Request would exceed maximum bandwidth.
FLYCAPTURE_MEMORY_ALLOC_ERROR	Memory allocation error
FLYCAPTURE_NON_PGR_CAMERA	Attached camera is not a PGR camera.
FLYCAPTURE_NOT_IMPLEMENTED	Function not implemented.
FLYCAPTURE_NOT_INITIALIZED	Device not initialized.
FLYCAPTURE_NOT_STARTED	flycaptureStart() not called.
FLYCAPTURE_NO_IMAGE	flycaptureGrabImage() not called.
FLYCAPTURE_OK	Function completed successfully.
FLYCAPTURE_TIMEOUT	Operation timed out.
FLYCAPTURE_TOO_MANY_LOCKED_BUFFERS	Too many image buffers are locked by the user.
FLYCAPTURE_VERSION_MISMATCH	There is a version mismatch between one of the interacting modules: pgrflycapture.dll, pgrflycaptureregui.dll, and the camera driver.

2.9.13. FlyCaptureFrameRate

Enum describing different framerates.

Declaration

```
enum FlyCaptureFrameRate
{
   FLYCAPTURE_FRAMERATE_1_875,
   FLYCAPTURE_FRAMERATE_3_75,
   FLYCAPTURE_FRAMERATE_7_5,
   FLYCAPTURE_FRAMERATE_15,
   FLYCAPTURE_FRAMERATE_30,
   FLYCAPTURE_FRAMERATE_50,
   FLYCAPTURE_FRAMERATE_60,
   FLYCAPTURE_FRAMERATE_120,
   FLYCAPTURE_FRAMERATE_240,
   FLYCAPTURE_NUM_FRAMERATES,
   FLYCAPTURE_FRAMERATE_CUSTOM,
   FLYCAPTURE_FRAMERATE_ANY,

}
```

Elements

FLYCAPTURE_FRAMERATE_120	120 fps.

FLYCAPTURE_FRAMERATE_15	15 fps.
FLYCAPTURE_FRAMERATE_1_875	1.875 fps. (Frames per second)
FLYCAPTURE_FRAMERATE_240	240 fps.
FLYCAPTURE_FRAMERATE_30	30 fps.
FLYCAPTURE_FRAMERATE_3_75	3.75 fps.
FLYCAPTURE_FRAMERATE_50	Deprecated. Please use Custom image.
FLYCAPTURE_FRAMERATE_60	60 fps.
FLYCAPTURE_FRAMERATE_7_5	7.5 fps.
FLYCAPTURE_FRAMERATE_ANY	Hook for "any usable frame rate."
FLYCAPTURE_FRAMERATE_CUSTOM	Custom frame rate. Used with custom image size functionality.
FLYCAPTURE_NUM_FRAMERATES	Number of possible camera frame rates.

2.9.14. FlyCaptureImage

This structure is used to pass image information into and out of the API.

Declaration

```
struct FlyCaptureImage
{
   int iRows;
   int iCols;
   int iRowInc;
   FlyCaptureVideoMode videoMode;
   FlyCaptureTimestamp timeStamp;
   unsigned char* pData;
   bool bStippled;
   FlyCapturePixelFormat pixelFormat;
   unsigned long  ulReserved[ 6 ];

}
```

Elements

bStippled	If the returned image is Y8, Y16, RAW8 or RAW16, this flag indicates whether it is a greyscale or stippled (bayer tiled) image. In all other modes, this flag has no meaning.
iCols	Columns, in pixels, of the image.
iRowInc	Row increment. The number of bytes per row.
iRows	Rows, in pixels, of the image.
pData	Pointer to the actual image data.
pixelFormat	The pixel format of this image.
timeStamp	Timestamp of this image.
ulReserved	Reserved for future use.
videoMode	Video mode that this image was captured with. This member is only populated when the image is returned from a grab call.

Remarks

The size of the image buffer is iRowInc * iRows, and depends on the pixel format.

2.9.15. FlyCaptureImageEvent

This structure is used for partial image notification functionality. Please see the release notes and API documentation for details.

Declaration

```
struct FlyCaptureImageEvent
{
   unsigned char* pBuffer;

   unsigned int   uiSizeBytes;

   unsigned int   uiSeqNum;

   unsigned int   uiBufferIndex;

   void*          pInternal;

   unsigned long  ulReserved[ 8 ];

}
```

Elements

pBuffer	A pointer to the start of the location inside the image buffer that this event corresponds to. This is only valid on structures coming back from flycaptureLockNextEvent() and points to a buffer allocated internally (in the case of flycaptureInitialize*(), or passed in by the user (in the case of flycaptureInitializePlus()).
pInternal	Internal bookkeeping. This can be ignored.
uiBufferIndex	The internal buffer index of this image portion. This can be ignored.
uiSeqNum	Sequence number for this image event. Populated by flycaptureWaitForImageEvent(). Sequence numbers should be contiguous if no image buffers are being dropped. If they are not, then the user level grab thread is not keeping up with the images the camera is sending, and this is a fatal error.
uiSizeBytes	The size of the image portion that this event corresponds to. This must be specified for structures being passed in to flycaptureInitializeNotify(). This is the only member that needs to be specified. The sizes passed in to flycaptureInitializeNotify() must add up to the total image size and must be whole multiples of the packet size. Appropriate packet sizes can be determined using flycaptureGetPacketSize() and flycaptureGetCustomImagePacketInfo().
ulReserved	Reserved for future use.

See Also

flycaptureInitializeNotify(), flycaptureLockNextEvent(), flycaptureWaitForImageEvent(), flycaptureUnlockEvent(). flycaptureGetPacketSize(), flycaptureGetCustomImageMaxPacketSize()

2.9.16. FlyCaptureImageFileFormat

Enumerates the image file formats that flycaptureSaveImage() can write to.

Declaration

```
enum FlyCaptureImageFileFormat
```

```
{
   FLYCAPTURE_FILEFORMAT_PGM,
   FLYCAPTURE_FILEFORMAT_PPM,
   FLYCAPTURE_FILEFORMAT_BMP,
   FLYCAPTURE_FILEFORMAT_JPG,
   FLYCAPTURE_FILEFORMAT_PNG,
   FLYCAPTURE_FILEFORMAT_RAW,

}
```

Elements

FLYCAPTURE_FILEFORMAT_BMP	3 or 4 channel RGB windows bitmap.
FLYCAPTURE_FILEFORMAT_JPG	JPEG format. Not implemented.
FLYCAPTURE_FILEFORMAT_PGM	Single channel (8 or 16 bit) greyscale portable grey map.
FLYCAPTURE_FILEFORMAT_PNG	Portable Network Graphics format. Not implemented.
FLYCAPTURE_FILEFORMAT_PPM	3 channel RGB portable pixel map.
FLYCAPTURE_FILEFORMAT_RAW	Raw data output.

2.9.17. FlyCaptureImagePlus

A wrapper for FlyCaptureImage that provides access to advanced functionality.

Declaration

```
struct FlyCaptureImagePlus
{
   FlyCaptureImage   image;

   unsigned int      uiSeqNum;

   unsigned int      uiBufferIndex;

   unsigned long     ulReserved[ 8 ];

}
```

Elements

image	The FlyCaptureImage that this FlyCaptureImagePlus structure is wrapping. Please see documentation in pgrflycapture.h.
uiBufferIndex	The internal buffer index that the image buffer contained in the FlyCaptureImage corresponds to. For functions that lock the image, this number must be passed back to the "unlock" function. If flycaptureInitializePlus() was called, this number corresponds to the position of the buffer in the buffer array passed in.
uiSeqNum	The sequence number of the image. This number is generated in the driver and sequential images should have a difference of one. If the difference is greater than one, it indicates the number of missed images since the last lock image call.
ulReserved	Reserved for future use.

2.9.18. FlyCaptureInfoEx

This structure stores a variety of different pieces of information associated with a particular camera. It is used with the flycaptureBusEnumerateCamerasEx() method. This structure has replaced FlyCaptureInfo.

Declaration

```
struct FlyCaptureInfoEx
{
   FlyCaptureCameraSerialNumber  SerialNumber;
   FlyCaptureCameraType          CameraType;
   FlyCaptureCameraModel         CameraModel;
   char  pszModelName[ 512 ];
   char  pszVendorName[ 512 ];
   char  pszSensorInfo[ 512 ];
   int   iDCAMVer;
   int   iNodeNum;
   int   iBusNum;
   FlyCaptureBusSpeed            CameraMaxBusSpeed;

   unsigned long  ulReserved[ 116 ];

}
```

Elements

CameraMaxBusSpeed	Camera max bus speed
CameraModel	Camera model.
CameraType	Type of imager (color or b&w).
SerialNumber	Camera serial number.
iBusNum	Low-level 1394 bus number for this device.
iDCAMVer	1394 DCAM compliance level. DCAM version is this value / 100. eg, 1.31.
iNodeNum	Low-level 1394 node number for this device.
pszModelName	Camera model string. Null terminated.
pszSensorInfo	Sensor info string. Null terminated.
pszVendorName	Vendor name string. Null terminated.
ulReserved	Reserved for future data.

2.9.19. FlyCaptureMessage

This structure is used to receive messages for specific cameras or bus events. The information received is message dependent.

Declaration

```
struct FlyCaptureMessage
{
   FlyCaptureMessageType     msgType;

   union
   {
      struct
      {
```

```
   int                     iBusNumber;
   FlyCaptureSystemTime    stTimeStamp;
} Reset;

struct
{
   char                    szDevice[128];
   unsigned long           ulSerialNumber;
   int                     iBusNumber;
   int                     iNodeNumber;
   FlyCaptureSystemTime    stTimeStamp;
} Arrival;

struct
{
   char                    szDevice[128];
   unsigned long           ulSerialNumber;
   int                     iBusNumber;
   int                     iNodeNumber;
   FlyCaptureSystemTime    stTimeStamp;
} Removal;

struct
{
   char                    szDevice[128];
   unsigned long           ulSerialNumber;
   int                     iBusNumber;
   int                     iNodeNumber;
   FlyCaptureSystemTime    stTimeStamp;
   unsigned long           ulErrorCode;
} BusError;

struct
{
   char                    szDevice[128];
   unsigned long           ulSerialNumber;
   int                     iBusNumber;
   int                     iNodeNumber;
   unsigned long           ulSequence;
   unsigned long           ulBytes;
   FlyCaptureSystemTime    stTimeStamp;
} Image;

struct
{
   char                    szDevice[128];
   unsigned long           ulSerialNumber;
   int                     iBusNumber;
   int                     iNodeNumber;
   unsigned long           ulRegister;
   unsigned long           ulValue;
   char                    szError[16];
} Register;

struct
```

```
      {
         char                 szDevice[128];
         unsigned long        ulSerialNumber;
         int                  iBusNumber;
         int                  iNodeNumber;
         unsigned long        ulRegister;
         unsigned long        ulNumberOfQuadlets;
         char                 szError[16];
      } RegisterBlock;
   } Msg;

   unsigned long            ulReserved[64];
}
```

Elements

Msg	The message specific details.
msgType	The type of message being received.
ulReserved	Reserved for future use.

2.9.20. FlyCaptureMessageType

Enumerates the message types that can be received.

Declaration

```
enum FlyCaptureMessageType
{
   FLYCAPTURE_BUS_RESET,
   FLYCAPTURE_DEVICE_ARRIVAL,
   FLYCAPTURE_DEVICE_REMOVAL,
   FLYCAPTURE_BUS_ERROR,
   FLYCAPTURE_GRABBED_IMAGE,
   FLYCAPTURE_REGISTER_READ,
   FLYCAPTURE_REGISTER_READ_BLOCK,
   FLYCAPTURE_REGISTER_WRITE,
   FLYCAPTURE_REGISTER_WRITE_BLOCK,
}
```

Elements

FLYCAPTURE_BUS_ERROR	A 1394b bus has experienced an error.
FLYCAPTURE_BUS_RESET	The bus was reset.
FLYCAPTURE_DEVICE_ARRIVAL	A device was connected.
FLYCAPTURE_DEVICE_REMOVAL	A device was disconnected.
FLYCAPTURE_GRABBED_IMAGE	An image has been grabbed.
FLYCAPTURE_REGISTER_READ	A register has been read.
FLYCAPTURE_REGISTER_READ_BLOCK	A block of registers has been read.
FLYCAPTURE_REGISTER_WRITE	A register has been written to.
FLYCAPTURE_REGISTER_WRITE_BLOCK	A block of registers has been written to.

2.9.21. FlyCapturePacketInfo

The rate at which data is transmitted over the 1394 bus is determined in part by the size of the datapackets. This structure describes a camera's capabilities for a given video mode.

Declaration

```
struct FlyCapturePacketInfo
{
   unsigned int   uiMinSizeBytes;
   unsigned int   uiMaxSizeBytes;
   unsigned int   uiMaxSizeEventBytes;
   unsigned long  ulReserved[ 8 ];
}
```

Elements

uiMaxSizeBytes	Maximum packet size, in bytes. Note that this ignores the OS-enforced bandwidth restrictions. The realized max packet size will be 80% of this reported value.
uiMaxSizeEventBytes	Maximum packet size, in bytes, when using events. The bandwidth note for uiMaxSizeBytes applies here too.
uiMinSizeBytes	Minimum packet size, in bytes.
ulReserved	Reserved for future use.

See Also
flycaptureGetPacketInfo()

2.9.22. FlyCapturePixelFormat

An enumeration used to indicate the pixel format of an image. This enumeration is used as a member of FlyCaptureImage and as a parameter to FlyCaptureStartCustomImage().

Declaration

```
enum FlyCapturePixelFormat
{
   FLYCAPTURE_MONO8     = 0x00000001,
   FLYCAPTURE_411YUV8   = 0x00000002,
   FLYCAPTURE_422YUV8   = 0x00000004,
   FLYCAPTURE_444YUV8   = 0x00000008,
   FLYCAPTURE_RGB8      = 0x00000010,
   FLYCAPTURE_MONO16    = 0x00000020,
   FLYCAPTURE_RGB16     = 0x00000040,
   FLYCAPTURE_S_MONO16  = 0x00000080,
   FLYCAPTURE_S_RGB16   = 0x00000100,
   FLYCAPTURE_RAW8      = 0x00000200,
   FLYCAPTURE_RAW16     = 0x00000400,
   FLYCAPTURE_BGR       = 0x10000001,
   FLYCAPTURE_BGRU      = 0x10000002,
   FCPF_FORCE_QUADLET   = 0x7FFFFFFF,

}
```

Elements

FCPF_FORCE_QUADLET	Unused member to force this enum to compile to 32 bits.
FLYCAPTURE_411YUV8	YUV 4:1:1.
FLYCAPTURE_422YUV8	YUV 4:2:2.
FLYCAPTURE_444YUV8	YUV 4:4:4.
FLYCAPTURE_BGR	24 bit BGR

FLYCAPTURE_BGRU	32 bit BGRU
FLYCAPTURE_MONO16	16 bit mono.
FLYCAPTURE_MONO8	8 bit of mono.
FLYCAPTURE_RAW16	16 bit raw data output from sensor.
FLYCAPTURE_RAW8	8 bit raw data output from sensor.
FLYCAPTURE_RGB16	RR, G and B are the same and equal 16 bits.
FLYCAPTURE_RGB8	R, G and B are the same and equal 8 bits.
FLYCAPTURE_S_MONO16	16 bit signed mono .
FLYCAPTURE_S_RGB16	RR, G and B are the same and equal 16 bits signed

2.9.23. FlyCaptureProperty

An enumeration of the different camera properties that can be set via the API.

Declaration

```
enum FlyCaptureProperty
{
   FLYCAPTURE_BRIGHTNESS,
   FLYCAPTURE_AUTO_EXPOSURE,
   FLYCAPTURE_SHARPNESS,
   FLYCAPTURE_WHITE_BALANCE,
   FLYCAPTURE_HUE,
   FLYCAPTURE_SATURATION,
   FLYCAPTURE_GAMMA,
   FLYCAPTURE_IRIS,
   FLYCAPTURE_FOCUS,
   FLYCAPTURE_ZOOM,
   FLYCAPTURE_PAN,
   FLYCAPTURE_TILT,
   FLYCAPTURE_SHUTTER,
   FLYCAPTURE_GAIN,
   FLYCAPTURE_TRIGGER_DELAY,
   FLYCAPTURE_FRAME_RATE,

   FLYCAPTURE_SOFTWARE_WHITEBALANCE,
   FLYCAPTURE_TEMPERATURE,

}
```

Elements

FLYCAPTURE_AUTO_EXPOSURE	The auto exposure property of the camera.
FLYCAPTURE_BRIGHTNESS	The brightness property of the camera.
FLYCAPTURE_FOCUS	The focus property of the camera.
FLYCAPTURE_FRAME_RATE	The frame rate property of the camera.
FLYCAPTURE_GAIN	The gain property of the camera.
FLYCAPTURE_GAMMA	The gamma property of the camera.
FLYCAPTURE_HUE	The hue property of the camera.
FLYCAPTURE_IRIS	The iris property of the camera.
FLYCAPTURE_PAN	The pan property of the camera.
FLYCAPTURE_SATURATION	The saturation property of the camera.
FLYCAPTURE_SHARPNESS	The sharpness property of the camera.

FLYCAPTURE_SHUTTER	The shutter property of the camera.
FLYCAPTURE_SOFTWARE_WHITEBALANCE	Software white balance property. Use this to manipulate the values for software whitebalance. This is only applicable to cameras that do not do onboard color processing. On these cameras, hardware white balance is disabled.
FLYCAPTURE_TEMPERATURE	The temperature property of the camera
FLYCAPTURE_TILT	The tilt property of the camera.
FLYCAPTURE_TRIGGER_DELAY	The trigger delay property of the camera.
FLYCAPTURE_WHITE_BALANCE	The hardware white balance property of the camera.
FLYCAPTURE_ZOOM	The zoom property of the camera.

Remarks
A lot of these properties are included only for completeness and future expandability, and will have no effect on a PGR camera.

2.9.24. FlyCaptureStippledFormat

An enumeration used to indicate the Bayer tile format of the stippled images passed into a destippling function.

Declaration

```
enum FlyCaptureStippledFormat
{
   FLYCAPTURE_STIPPLEDFORMAT_BGGR,
   FLYCAPTURE_STIPPLEDFORMAT_GBRG,
   FLYCAPTURE_STIPPLEDFORMAT_GRBG,
   FLYCAPTURE_STIPPLEDFORMAT_RGGB,
   FLYCAPTURE_STIPPLEDFORMAT_DEFAULT

}
```

Elements

FLYCAPTURE_STIPPLEDFORMAT_BGGR	Indicates a BGGR image.
FLYCAPTURE_STIPPLEDFORMAT_DEFAULT	Indicates the default stipple format for the Dragonfly or Firefly.
FLYCAPTURE_STIPPLEDFORMAT_GBRG	Indicates a GBRG image.
FLYCAPTURE_STIPPLEDFORMAT_GRBG	Indicates a GRBG image.
FLYCAPTURE_STIPPLEDFORMAT_RGGB	Indicates a RGGB image.

Remarks
This is only relevant for cameras that do not do onboard color processing, such as the Dragonfly. The four letters of the enum value correspond to the “top left” 2x2 section of the stippled image. For example, the first line of a BGGR image image will be BGBGBG..., and the second line will be GRGRGR....

2.9.25. FlyCaptureSystemTime

This structure is used in messages as either the timestamp of an image or a bus event time.

Declaration

```
struct FlyCaptureSystemTime
{
   unsigned short usHour;
   unsigned short usMinute;
   unsigned short usSecond;
   unsigned short usMilliseconds;
}
```

Elements

usHour	
usMilliseconds	
usMinute	
usSecond	

2.9.26. FlyCaptureTimestamp

This structure defines the format by which time is represented in the PGRFlycapture SDK. The ulSeconds and ulMicroSeconds values represent the absolute system time when the image was captured. The ulCycleSeconds and ulCycleCount are higher-precision values that have either been propagated up from the 1394 bus or extracted from the image itself. The data will be extracted from the image if image timestamping is enabled and directly (and less accurately) from the 1394 bus otherwise.

Declaration

```
struct FlyCaptureTimestamp
{
   unsigned long ulSeconds;
   unsigned long ulMicroSeconds;
   unsigned long ulCycleSeconds;
   unsigned long ulCycleCount;
   unsigned long ulCycleOffset;

}
```

Elements

ulCycleCount	The cycle time count. 0-7999. (1/8000ths of a second.)
ulCycleOffset	The cycle offset. 0-3071 (1/3072ths of a cycle count.)
ulCycleSeconds	The cycle time seconds. 0-127.
ulMicroSeconds	The microseconds component.
ulSeconds	The number of seconds since the epoch.

Remarks

The ulCycleSeconds value will wrap around after 128 seconds. The ulCycleCount represents the 1/8000 second component. Use these two values when synchronizing grabs between two computers sharing a common 1394 bus that may not have precisely synchronized system timers.

2.9.27. FlyCaptureVideoMode

Enum describing different video modes.

Declaration

```
enum FlyCaptureVideoMode
{
   FLYCAPTURE_VIDEOMODE_160x120YUV444     = 0,
   FLYCAPTURE_VIDEOMODE_320x240YUV422     = 1,
   FLYCAPTURE_VIDEOMODE_640x480YUV411     = 2,
   FLYCAPTURE_VIDEOMODE_640x480YUV422     = 3,
   FLYCAPTURE_VIDEOMODE_640x480RGB        = 4,
   FLYCAPTURE_VIDEOMODE_640x480Y8         = 5,
   FLYCAPTURE_VIDEOMODE_640x480Y16        = 6,
   FLYCAPTURE_VIDEOMODE_800x600YUV422     = 17,
   FLYCAPTURE_VIDEOMODE_800x600RGB        = 18,
   FLYCAPTURE_VIDEOMODE_800x600Y8         = 7,
   FLYCAPTURE_VIDEOMODE_800x600Y16        = 19,
   FLYCAPTURE_VIDEOMODE_1024x768YUV422    = 20,
   FLYCAPTURE_VIDEOMODE_1024x768RGB       = 21,
   FLYCAPTURE_VIDEOMODE_1024x768Y8        = 8,
   FLYCAPTURE_VIDEOMODE_1024x768Y16       = 9,
   FLYCAPTURE_VIDEOMODE_1280x960YUV422    = 22,
   FLYCAPTURE_VIDEOMODE_1280x960RGB       = 23,
   FLYCAPTURE_VIDEOMODE_1280x960Y8        = 10,
   FLYCAPTURE_VIDEOMODE_1280x960Y16       = 24,
   FLYCAPTURE_VIDEOMODE_1600x1200YUV422   = 50,
   FLYCAPTURE_VIDEOMODE_1600x1200RGB      = 51,
   FLYCAPTURE_VIDEOMODE_1600x1200Y8       = 11,
   FLYCAPTURE_VIDEOMODE_1600x1200Y16      = 52,

   FLYCAPTURE_VIDEOMODE_CUSTOM            = 15,
   FLYCAPTURE_VIDEOMODE_ANY               = 16,

   FLYCAPTURE_NUM_VIDEOMODES              = 23,

}
```

Elements

FLYCAPTURE_NUM_VIDEOMODES	Number of possible video modes.
FLYCAPTURE_VIDEOMODE_1024x768RGB	1024x768 RGB.
FLYCAPTURE_VIDEOMODE_1024x768Y16	1024x768 16-bit greyscale or bayer tiled color image.
FLYCAPTURE_VIDEOMODE_1024x768Y8	1024x768 8-bit greyscale or bayer tiled color image.
FLYCAPTURE_VIDEOMODE_1024x768YUV422	1024x768 YUV422.
FLYCAPTURE_VIDEOMODE_1280x960RGB	1280x960 RGB.
FLYCAPTURE_VIDEOMODE_1280x960Y16	1280x960 16-bit greyscale or bayer titled color image.
FLYCAPTURE_VIDEOMODE_1280x960Y8	1280x960 8-bit greyscale or bayer titled color image.
FLYCAPTURE_VIDEOMODE_1280x960YUV422	1280x960 YUV422.
FLYCAPTURE_VIDEOMODE_1600x1200RGB	1600x1200 RGB.
FLYCAPTURE_VIDEOMODE_1600x1200Y16	1600x1200 16-bit greyscale or bayer titled color image.
FLYCAPTURE_VIDEOMODE_1600x1200Y8	1600x1200 8-bit greyscale or bayer titled

	color image.
FLYCAPTURE_VIDEOMODE_1600x1200YUV422	1600x1200 YUV422.
FLYCAPTURE_VIDEOMODE_160x120YUV444	160x120 YUV444.
FLYCAPTURE_VIDEOMODE_320x240YUV422	320x240 YUV422.
FLYCAPTURE_VIDEOMODE_640x480RGB	640x480 24-bit RGB.
FLYCAPTURE_VIDEOMODE_640x480Y16	640x480 16-bit greyscale or bayer tiled color image.
FLYCAPTURE_VIDEOMODE_640x480Y8	640x480 8-bit greyscale or bayer tiled color image.
FLYCAPTURE_VIDEOMODE_640x480YUV411	640x480 YUV411.
FLYCAPTURE_VIDEOMODE_640x480YUV422	640x480 YUV422.
FLYCAPTURE_VIDEOMODE_800x600RGB	800x600 RGB.
FLYCAPTURE_VIDEOMODE_800x600Y16	800x600 16-bit greyscale or bayer tiled color image.
FLYCAPTURE_VIDEOMODE_800x600Y8	800x600 8-bit greyscale or bayer tiled color image.
FLYCAPTURE_VIDEOMODE_800x600YUV422	800x600 YUV422.
FLYCAPTURE_VIDEOMODE_ANY	Hook for “any usable video mode.”
FLYCAPTURE_VIDEOMODE_CUSTOM	Custom video mode. Used with custom image size functionality.

Remarks
The explicit numbering is to provide downward compatibility for this enum.

2.9.28. GenericCameraContext

Declaration

```
typedef void*  GenericCameraContext
```

Remarks
A generic camera context to cast SDK contexts to before passing them in.

2.10. External Functions

2.10.1. pgrcamguiCreateContext

Allocates a PGRCameraGUI handle to be used in all successive calls. This function must be called before any other functions.

Declaration

```
PGRCAMERAGUI_API CameraGUIError
pgrcamguiCreateContext(
                    CameraGUIContext* pcontext )
```

Parameters

pcontext	a pointer to a PGRCameraGUI context to be created.

Return Value

Value	Meaning
PGRCAMGUI_OK	upon successful completion.

Remarks
API Functions

2.10.2. pgrcamguiCreateGraphWindow

Create the FlyCapture specific Image Utility window. Call this function before any other calls involving the GraphWindow.

Declaration

```
PGRCAMERAGUI_API CameraGUIError
pgrcamguiCreateGraphWindow(
                                CameraGUIContext context )
```

Parameters

context	The PGRCameraGUI context to access.

Return Value

Value	Meaning
PGRCAMGUI_OK	upon successful completion.

2.10.3. pgrcamguiCreateSettingsDialog

This function is DEPRECATED. Please use pgrcamguiInitializeSettingsDialog()

Declaration

```
PGRCAMERAGUI_API CameraGUIError
pgrcamguiCreateSettingsDialog(
                             CameraGUIContext     context,
                             CameraGUIType        type,
                             GenericCameraContext camcontext )
```

2.10.4. pgrcamguiDestroyContext

Frees memory associated with a given context.

Declaration

```
PGRCAMERAGUI_API CameraGUIError
pgrcamguiDestroyContext(
                        CameraGUIContext context )
```

Parameters

context	context to destroy.

Return Value

Value	Meaning
PGRCAMGUI_OK	upon successful completion.
PGRCAMGUI_INVALID_CONTEXT	if context is null.

2.10.5. pgrcamguiGetGraphWindowState

Determines whether the Graph Window is currently displayed.

Declaration

```
PGRCAMERAGUI_API CameraGUIError
pgrcamguiGetGraphWindowState(
                                CameraGUIContext context,
                                BOOL* pShowing )
```

Parameters

context	The PGRCameraGUI context to access.
pShowing	BOOL to be filled with state information. Must not be NULL.

Return Value

Value	Meaning
PGRCAMGUI_OK	upon successful completion.
PGRCAMGUI_INVALID_ARGUMENT	if pShowing is NULL.

2.10.6. pgrcamguiGetSettingsWindowState

Retrieves the state of the settings dialog.

Declaration

```
PGRCAMERAGUI_API CameraGUIError
pgrcamguiGetSettingsWindowState(
                                CameraGUIContext   context,
                                BOOL*              pbShowing )
```

Parameters

context	The PGRCameraGUI context to access.
pbShowing	A pointer to the returned state of the settings dialog.

Return Value

Value	Meaning
PGRCAMGUI_OK	upon successful completion.
PGRCAMGUI_FAILED	if the function failed.
PGRCAMGUI_INVALID_CONTEXT	if context is null.

2.10.7. pgrcamguiInitializeSettingsDialog

Creates the settings dialog. Call this before calling the either of the other two functions dealing with the Settings dialog.

Declaration

```
PGRCAMERAGUI_API CameraGUIError
pgrcamguiInitializeSettingsDialog(
                                    CameraGUIContext       context,
                                    GenericCameraContext   camcontext )
```

Parameters

context	The PGRCameraGUI context to access.
camcontext	The SDK-level context to use.

Return Value

Value	Meaning
PGRCAMGUI_OK	upon successful completion.
PGRCAMGUI_FAILED	if the function failed.
PGRCAMGUI_INVALID_CONTEXT	if context is null.

2.10.8. pgrcamguiSetSettingsWindowHelpPrefix

Enables context sensitive help in the settings dialog by specifying the help prefix to append topic specific pages to. eg: "..\\doc\\FlyCapture SDK help.chm::/FlyCap Demo Program/Camera Control Dialog" to which strings like "/Look Up Table.html" will be appended.

Declaration

```
PGRCAMERAGUI_API CameraGUIError
pgrcamguiSetSettingsWindowHelpPrefix(
                                    CameraGUIContext context,
                                    const char* pszHelpPrefix )
```

Parameters

context	The PGRCameraGUI context to access.
pszHelpPrefix	a pointer to a string containing the desired help prefix if NULL the context sensitive help will be unavailable.

Return Value

Value	Meaning
PGRCAMGUI_OK	upon successful completion.
PGRCAMGUI_FAILED	if the function failed.
PGRCAMGUI_INVALID_CONTEXT	if context is null.

2.10.9. pgrcamguiShowCameraSelectionModal

Displays the camera selection dialog.

Declaration

```
PGRCAMERAGUI_API CameraGUIError
pgrcamguiShowCameraSelectionModal(
                                  CameraGUIContext       context,
                                  GenericCameraContext   camcontext,
                                  unsigned long*         pulSerialNumbe
r,
                                  INT_PTR*               pipDialogStatu
s )
```

Parameters

context	The PGRCameraGUI context to access.
camcontext	The SDK-specific context to use, casted appropriately. (ie, FlycaptureContext, DigiclopsContext, etc.)
pulSerialNumber	Pointer to the returned serial number of the selected camera.
pipDialogStatus	Status returned from the DoModal() call from the dialog. Use this to check for “Ok” or “Cancel.”

Return Value

Value	Meaning
PGRCAMGUI_OK	upon successful completion.
PGRCAMGUI_FAILED	if the function failed.
PGRCAMGUI_INVALID_CONTEXT	if context is NULL.

2.10.10. pgrcamguiShowInfoDlg

Displays a modal dialog that displays PGR version information.

Declaration

```
PGRCAMERAGUI_API CameraGUIError
pgrcamguiShowInfoDlg(
                     CameraGUIContext      context,
                     GenericCameraContext  camcontext,
                     HWND                  hwndParent,
                     char*                 pszAppName = NULL )
```

Parameters

context	The PGRCameraGUI context to access.
camcontext	The SDK-level context to use.
hwndParent	Handle to the parent window.

pszAppName	Pointer to a string that will be prepended to the version information and seperated by a newline.

Return Value

Value	Meaning
PGRCAMGUI_OK	upon successful completion.

2.10.11. pgrcamguiToggleGraphWindowState

Displays or hides the modeless Image Utility window.

Declaration

```
PGRCAMERAGUI_API CameraGUIError
pgrcamguiToggleGraphWindowState(
                                CameraGUIContext context,
                                HWND hwndParent )
```

Parameters

context	The PGRCameraGUI context to access.
hwndParent	A handle to the dialog's parent. Must not be NULL.

Return Value

Value	Meaning
PGRCAMGUI_OK	upon successful completion.
PGRCAMGUI_INVALID_ARGUMENT	if context or hwndParent is NULL
PGRCAMGUI_COULD_NOT_CREATE_DIALOG	if dialog could not be created
PGRCAMGUI_FAILED	if dialog could not be shown or destroyed

2.10.12. pgrcamguiToggleSettingsWindowState

Displays or hides the modeless settings dialog.

Declaration

```
PGRCAMERAGUI_API CameraGUIError
pgrcamguiToggleSettingsWindowState(
                                CameraGUIContext   context,
                                HWND               hwndParent )
```

Parameters

context	The PGRCameraGUI context to access.
hwndParent	Handle to the parent window.

Return Value

Value	Meaning

PGRCAMGUI_OK	upon successful completion.
PGRCAMGUI_FAILED	if the function failed.
PGRCAMGUI_INVALID_CONTEXT	if context is null.

2.10.13. pgrcamguiUpdateGraphWindowImage

Used to pass a new image to the Image Utility window.

Declaration

```
PGRCAMERAGUI_API CameraGUIError
pgrcamguiUpdateGraphWindowImage(
                                CameraGUIContext context,
                                FlyCaptureContext fcContext,
                                FlyCaptureImage* pImage )
```

Parameters

context	The PGRCameraGUI context to access.
fcContext	A valid FlyCaptureContext. Must not be NULL.
pImage	A valid FlyCaptureImage. Must not be NULL.

Return Value

Value	Meaning
PGRCAMGUI_OK	upon successful completion.
PGRCAMGUI_INVALID_ARGUMENT	if any argument is NULL.

2.11. Macros

2.11.1. FLYCAPTURE_BUS_MESSAGE

Declaration

```
#define FLYCAPTURE_BUS_MESSAGE   999999999
```

Remarks
This is used as the serial number when initializing or receiving bus messages.

2.11.2. FLYCAPTURE_INFINITE

Declaration

```
#define FLYCAPTURE_INFINITE   0xFFFFFFFF
```

Remarks
A value indicating an infinite wait. This macro is used primarily used with the flycaptureSetGrabTimeoutEx() in order to indicate the software should wait indefinitely for the camera to produce an image.

2.11.3. PGRCAMERAGUI_API

Declaration

```
#define PGRCAMERAGUI_API __declspec( dllexport )
```

2.11.4. PGRFLYCAPTURE_API

Declaration

```
#define PGRFLYCAPTURE_API __declspec( dllexport )
```

2.11.5. PGRFLYCAPTURE_CALL_CONVEN

Declaration

```
#define PGRFLYCAPTURE_CALL_CONVEN __cdecl
```

2.11.6. PGRFLYCAPTURE_VERSION

The version of the library.

Declaration

```
#define PGRFLYCAPTURE_VERSION 106106
```

3 Technical Support Resources

Point Grey Research Inc. endeavours to provide the highest level of technical support possible to our customers. Most support resources can be accessed through the Product Support section of our website: www.ptgrey.com/support.

3.1. Creating a Customer Login Account

The first step in accessing our technical support resources is to obtain a Customer Login Account. This requires a valid name, e-mail address, and camera serial number. To apply for a Customer Login Account go to: www.ptgrey.com/support/downloads/user_request.html.

3.2. Knowledge Base

Our on-line knowledge base contains answers to some of the most common support questions. It has information about all PGR products and was developed to help customers resolve product issues. It is constantly updated, expanded, and refined to ensure that our customers have access to the latest information. To access the knowledge base, go to: www.ptgrey.com/support/kb/.

3.3. Product Downloads

Customers with a Customer Login Account can access the latest software and firmware for their cameras from our downloads site at www.ptgrey.com/support/downloads. We encourage our customers to keep their software and firmware up-to-date by downloading and installing the latest versions. These versions include the latest bug fixes and feature enhancements.

3.4. Contacting Technical Support

Before contacting Technical Support, have you:

1. *Read the product documentation and user manual?*
2. *Searched the Knowledge Base?*
3. *Downloaded and installed the latest version of software and/or firmware?*

If you have done all the above and still can't find an answer to your question, contact our Technical Support team using our on-line web form: www.ptgrey.com/support/contact/. This will create a ticket in our Request Tracker support system, and a Technical Support representative will contact you by e-mail within one (1) business day.

4 Contacting Point Grey Research Inc.

For any questions, concerns or comments please contact us via the following methods:

Email: For all general questions about Point Grey Research please contact us at info@ptgrey.com.

For technical support (existing customers only) contact us at http://www.ptgrey.com/support/contact/.

Knowledge Base: Find answers to commonly asked questions in our knowledge base at http://www.ptgrey.com/support/kb/.

Downloads: Users can download the latest manuals and software from http://www.ptgrey.com/support/downloads/

Main Office:

Mailing Address:
Point Grey Research, Inc.
8866 Hudson Street
Vancouver, BC, Canada
V6P 4N2

Tel: +1 (604) 730-9937
Fax: +1 (604) 732-8231
sales@ptgrey.com

Distributors

USA

Mailing Address:
13749 E. Charter Oak Drive
Scottsdale, AZ USA
85259-2322

Tel: +1 (480) 391-2125
Fax: +1 (480) 391-2125
na-sales@ptgrey.com

Europe

Mailing Address:
Gerstäcker Str. 60
D-81827 München
Germany

Tel: +49 (89) 45463224
Fax: +49 (89) 45463225
eu-sales@ptgrey.com

Japan — ViewPLUS Inc. (http://www.viewplus.co.jp/)

Korea — Cylod Co. Ltd. (http://www.cylod.com)