



# Lumenera USB Camera API Reference Manual

## Release 3.3



The contents of this document may not be copied nor duplicated in any form, in whole or in part, without prior written consent from Lumenera Corporation. Lumenera makes no warranties as to the accuracy of the information contained in this document or its suitability for any purpose. The information in this document is subject to change without notice.  
Copyright © 2004 Lumenera Corporation. All rights reserved.

**License Agreement (Software):**

This Agreement states the terms and conditions upon which Lumenera Corporation ("Lumenera") offers to license to you (the "Licensee") the software together with all related documentation and accompanying items including, but not limited to, the executable programs, drivers, libraries, and data files associated with such programs (collectively, the "Software").

The Software is licensed, not sold, to you for use only under the terms of this Agreement.

Lumenera grants to you the right to use all or a portion of this Software provided that the Software is used only in conjunction with Lumenera's family of products.

In using the Software you agree not to:

- a) decompile, disassemble, reverse engineer, or otherwise attempt to derive the source code for any Product (except to the extent applicable laws specifically prohibit such restriction);
- b) remove or obscure any trademark or copyright notices.

**Limited Warranty (Hardware and Software):**

ANY USE OF THE SOFTWARE OR HARDWARE IS AT YOUR OWN RISK. THE SOFTWARE IS PROVIDED FOR USE ONLY WITH LUMENERA'S HARDWARE AND OTHER RELATED SOFTWARE. THE SOFTWARE IS PROVIDED FOR USE "AS IS" WITHOUT WARRANTY OF ANY KIND. TO THE MAXIMUM EXTENT PERMITTED BY LAW, LUMENERA DISCLAIMS ALL WARRANTIES OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, WITHOUT LIMITATION, IMPLIED WARRANTIES OR CONDITIONS OF MERCHANTABILITY, QUALITY AND FITNESS FOR A PARTICULAR PURPOSE. LUMENERA IS NOT OBLIGATED TO PROVIDE ANY UPDATES OR UPGRADES TO THE SOFTWARE OR ANY RELATED HARDWARE.

**Limited Liability (Hardware and Software):**

In no event shall Lumenera or its Licensor's be liable for any damages whatsoever (including, without limitation, incidental, direct, indirect, special or consequential damages, damages for loss of business profits, business interruption, loss of business information, or other pecuniary loss) arising out of the use or inability to use this Software or related Hardware, including, but not limited to, any of Lumenera's family of products.

# Table of Contents

<i>Introduction</i> .....	6
1.1 The Lumenera API.....	6
<i>Summary of Functions</i> .....	7
2.1 Alphabetical Summary of Functions .....	7
2.2 API Function Summary Grouped by Task.....	10
<i>Detailed API Description</i> .....	15
LucamAddRgbPreviewCallback.....	15
LucamAddSnapshotCallback.....	16
LucamAddStreamingCallback.....	16
LucamCameraClose .....	17
LucamCameraOpen .....	18
LucamCameraReset .....	18
LucamConvertBmp24ToRgb24 .....	19
LucamConvertFrameToRGB24 .....	19
LucamConvertFrameToRGB32 .....	20
LucamConvertFrameToRGB48 .....	21
LucamCreateDisplayWindow .....	22
LucamDestroyDisplayWindow .....	23
LucamDisableFastFrames .....	23
LucamDisableSynchronousSnapshots.....	23
LucamDisplayPropertyPage.....	24
LucamDisplayVideoFormatPage .....	24
LucamEnableFastFrames.....	25
LucamEnableSynchronousSnapshots.....	26
LucamEnumAvailableFrameRates.....	27
LucamEnumCameras .....	27
LucamGetCameraId .....	28
LucamGetCurrentMatrix .....	29

<b>LucamGetFormat</b> .....	29
<b>LucamGetLastError</b> .....	30
<b>LucamGetProperty</b> .....	30
<b>LucamGpioRead</b> .....	31
<b>LucamGpioWrite</b> .....	32
<b>LucamGpoSelect</b> .....	32
<b>LucamNumCameras</b> .....	33
<b>LucamOneShotAutoExposure</b> .....	33
<b>LucamOneShotAutoWhiteBalance</b> .....	34
<b>LucamOneShotAutoWhiteBalanceEx</b> .....	35
<b>LucamPermanentBufferRead</b> .....	35
<b>LucamPermanentBufferWrite</b> .....	36
<b>LucamPropertyRange</b> .....	37
<b>LucamQueryDisplayFrameRate</b> .....	37
<b>LucamQueryExternInterface</b> .....	38
<b>LucamQueryRgbPreviewPixelFormat</b> .....	38
<b>LucamQueryVersion</b> .....	39
<b>LucamReadRegister</b> .....	39
<b>LucamRemoveRgbPreviewCallback</b> .....	40
<b>LucamRemoveSnapshotCallback</b> .....	41
<b>LucamRemoveStreamingCallback</b> .....	41
<b>LucamSaveImage</b> .....	42
<b>LucamSaveImageW</b> .....	42
<b>LucamSetFormat</b> .....	43
<b>LucamSetProperty</b> .....	44
<b>LucamSetup8bitsLUT</b> .....	44
<b>LucamSetupCustomMatrix</b> .....	45
<b>LucamStreamVideoControl</b> .....	45
<b>LucamTakeFastFrame</b> .....	46
<b>LucamTakeSnapshot</b> .....	47
<b>LucamTakeSynchronousSnapshots</b> .....	47
<b>LucamTakeVideo</b> .....	48

---

<b>LucamTakeVideoEx</b> .....	<b>48</b>
<b>LucamWriteRegister</b> .....	<b>49</b>
<i>Appendix I</i> .....	<i>51</i>
<b>4.1 Constants Definitions</b> .....	<b>51</b>
<b>4.2 Data Structure Definitions</b> .....	<b>54</b>



# Introduction

## 1.1 The Lumenera API

The Lumenera USB Camera API provides a comprehensive set of functions allowing you to control the operation of any Lumenera USB camera or camera module.

Directly callable from Visual C++, Visual Basic and Borland Builder, in a matter of minutes you can create an application to command and control the camera properties, retrieve, display, and capture image data.

Advanced functions allow for some very powerful features such as, video overlay, simultaneous image capture from multiple cameras, and complete control over the processing of the image data.

Although the list of functions is quite extensive, only a small number of functions are required to do the basics of image display, capture and camera control.

This document is only a reference for each of the API functions, describing how to call them. The sample code included with the Developer's Kit, including the full source code for the Lucam Capture application, provides many examples of how to use the functions in real-world applications.

And, of course, if you have purchased the Developer's Kit, our technical support group is available to provide assistance in the use of the API so you can get the most out of your camera application.

You can e-mail our technical support group at:

[support@lumenera.com](mailto:support@lumenera.com)

# 2

## Summary of Functions

### 2.1 Alphabetical Summary of Functions

Function	Description
LucamAddRgbPreviewCallback	Allows the user to add a video filter callback function, which is called after each frame of streaming video is returned from the camera and after it's processed.
LucamAddSnapshotCallback	Allows the user to add a data filter callback function, which is called after each hardware triggered snapshot is returned from the camera but before it's processed.
LucamAddStreamingCallback	Allows the user to add a video filter callback function, which is called after each frame of raw streaming video is returned from the camera but before it's processed.
LucamCameraClose	Closes the connection to Lumenera camera.
LucamCameraOpen	Opens a connection to a Lumenera camera.
LucamCameraReset	Resets camera to power-on default state.
LucamConvertBmp24ToRgb24	Converts a frame of data from the format returned by LucamConvertRgb24 (BGR) to standard format (RGB).
LucamConvertFrameToRGB24	Converts a raw frame of data obtained with LucamTakeVideo or LucamTakeSnapshot to a fully processed RGB24 frame suitable for display or saving.
LucamConvertFrameToRGB32	Converts a raw frame of data obtained with LucamTakeVideo or LucamTakeSnapshot to a fully processed RGB32 frame suitable for display or saving.
LucamConvertFrameToRgb48	Converts a raw frame of data obtained with LucamTakeVideo or LucamTakeSnapshot to a fully processed RGB48 frame suitable for saving in an image format that supports 16 bits

	per color channel (e.g. TIFF format).
LucamCreateDisplayWindow	Creates a display window, which is managed by the API, for displaying video.
LucamDestroyDisplayWindow	Destroys the display window created with LucamCreateDisplayWindow.
LucamDisableFastFrames	Disables the fast snapshot capture mode.
LucamDisableSynchronousSnapshots	Disables the simultaneous snapshot capture mode.
LucamDisplayPropertyPage	Pops up a Direct Show dialog with the camera properties.
LucamDisplayVideoFormatPage	Pops up a Direct Show dialog with the video properties.
LucamEnableFastFrames	Enables the fast snapshot capture mode.
LucamEnableSynchronousSnapshots	Enables the simultaneous snapshot capture mode.
LucamEnumAvailableFrameRates	Returns an array containing the available frame rates for the camera based on the clock rates available on the camera.
LucamEnumCameras	Returns the version information and serial numbers for all Lumenera cameras attached to the computer.
LucamGetCameraID	Gets the camera model ID number.
LucamGetCurrentMatrix	Gets the current color correction matrix being applied for video preview.
LucamGetFormat	Gets the video frame format (subwindow position and size, subsampling, pixel format) and desired frame rate for the video data.
LucamGetLastError	Gets the specific error code for the last error that occurred when calling an API function.
LucamGetProperty	Gets the value of the specified camera property.
LucamGpioRead	Reads the General Purpose I/O register for the external header status.
LucamGpioWrite	Writes to the General Purpose I/O register to trigger the external header output.
LucamGpoSelect	Enables and disables the alternate GPO functionality.
LucamNumCameras	Returns the number of Lumenera cameras attached to the computer.
LucamOneShotAutoExposure	Performs a single (one iteration) exposure adjustment in an attempt to reach the autoexposure target.
LucamOneShotAutoWhiteBalance	Performs a single (one iteration) gain adjustment in an attempt to color balance the

	image.
LucamOneShotAutoWhiteBalanceEx	Performs a single (one iteration) gain adjustment in an attempt to color balance the image to a specific target color.
LucamPermanentBufferRead	Reads data from the user-defined non-volatile memory area of the camera.
LucamPermanentBufferWrite	Writes data to the user-defined non-volatile memory area of the camera.
LucamPropertyRange	Returns the range of valid values for a camera property and its default value.
LucamQueryDisplayFrameRate	Returns the actual displayed frame rate of the camera.
LucamQueryExternInterface	Returns the type of interface between the camera and the computer.
LucamQueryRgbPreviewPixelFormat	Returns the pixel format for the preview window.
LucamQueryVersion	Returns version information about the camera.
LucamReadRegister	Reads the internal camera registers.
LucamRemoveRgbPreviewCallback	Removes the specified video filter callback function registered using the function LucamAddRgbPreviewCallback.
LucamRemoveSnapshotCallback	Removes the specified data filter callback function registered using the function LucamAddSnapshotCallback.
LucamRemoveStreamingCallback	Removes the specified video filter callback function registered using the function LucamAddStreamingCallback.
LucamSaveImage	Saves a single image or video frame to disk in one of several formats.
LucamSaveImageW	Exactly like LucamSaveImage but the input filename string is in Unicode (wide character) format.
LucamSetFormat	Sets the video frame format (subwindow position and size, subsampling, pixel format) and desired frame rate for the video data.
LucamSetProperty	Sets the value of the specified camera property.
LucamSetup8bitsLUT	Populates the 8-bit LUT inside the camera.
LucamSetupCustomMatrix	Defines the color correction matrix values to use when converting raw data to RGB24 with the correction matrix parameter LUCAM_CM_CUSTOM.
LucamStreamVideoControl	Controls the streaming video.
LucamTakeFastFrame	Takes a single image from the camera, using

	the camera's still imaging or video mode.
LucamTakeSnapshot	Takes a single image from the camera, using the camera's still imaging or video mode.
LucamTakeSynchronousSnapshots	Simultaneously takes a single image from each of several cameras.
LucamTakeVideo	Takes video frames from the camera, using the camera's video mode.
LucamTakeVideoEx	Takes video data greater than a specified threshold, from the camera, using the camera's video mode and returns their coordinates.
LucamWriteRegister	Writes to the internal camera registers.

## 2.2 API Function Summary Grouped by Task

### 2.2.1 Initialization

Function	Description
LucamNumCameras	Returns the number of Lumenera cameras attached to the computer.
LucamEnumCameras	Returns the version information and serial numbers for all Lumenera cameras attached to the computer.
LucamCameraOpen	Opens a connection to a Lumenera camera.
LucamCameraClose	Closes the connection to Lumenera camera.
LucamCameraReset	Resets camera to power-on default state.
LucamGetLastError	Gets the specific error code for the last error that occurred when calling an API function.

### 2.2.2 Camera Settings

Function	Description
LucamGetCameraID	Gets the camera model ID number.
LucamGetProperty	Gets the value of the specified camera property.
LucamSetProperty	Sets the value of the specified camera property.
LucamPropertyRange	Returns the range of valid values for a camera property and its default value.
LucamPermanentBufferRead	Reads data from the user-defined non-volatile memory area of the camera.
LucamPermanentBufferWrite	Writes data to the user-defined non-volatile memory area of the camera.

LucamOneShotAutoExposure	Performs a single (one iteration) exposure adjustment in an attempt to reach the autoexposure target.
LucamOneShotAutoWhiteBalance	Performs a single (one iteration) gain adjustment in an attempt to color balance the image.
LucamOneShotAutoWhiteBalanceEx	Performs a single (one iteration) gain adjustment in an attempt to color balance the image to a specific target color.
LucamSetupCustomMatrix	Defines the color correction matrix values to use when converting raw data to RGB24 with the correction matrix parameter LUCAM_CM_CUSTOM.
LucamSetup8bitsLUT	Populates the 8-bit LUT inside the camera.
LucamQueryVersion	Returns version information about the camera.
LucamQueryExternInterface	Returns the type of interface between the camera and the computer.

### 2.2.3 Video Control

Function	Description
LucamGetFormat	Gets the video frame format (subwindow position and size, subsampling, pixel format) and desired frame rate for the video data.
LucamSetFormat	Sets the video frame format (subwindow position and size, subsampling, pixel format) and desired frame rate for the video data.
LucamStreamVideoControl	Controls the streaming video.
LucamCreateDisplayWindow	Creates a display window, which is managed by the API, for displaying video.
LucamDestroyDisplayWindow	Destroys the display window created with LucamCreateDisplayWindow.
LucamDisplayPropertyPage	Pops up a Direct Show dialog with the camera properties.
LucamDisplayVideoFormatPage	Pops up a Direct Show dialog with the video properties.
LucamGetCurrentMatrix	Gets the current color correction matrix being applied for video preview.
LucamEnumAvailableFrameRates	Returns an array containing the available frame rates for the camera based on the clock rates available on the camera.
LucamQueryDisplayFrameRate	Returns the actual displayed frame rate of the camera.

## 2.2.4 Image Capture

Function	Description
LucamTakeVideo	Takes video frames from the camera, using the camera's video mode.
LucamTakeVideoEx	Takes video data greater than a specified threshold, from the camera, using the camera's video mode and returns their coordinates.
LucamTakeSnapshot	Takes a single image from the camera, using the camera's still imaging or video mode.
LucamEnableFastFrames	Enables the fast snapshot capture mode.
LucamTakeFastFrame	Takes a single image from the camera, using the camera's still imaging or video mode.
LucamDisableFastFrames	Disables the fast snapshot capture mode.
LucamEnableSynchronousSnapshots	Enables the simultaneous snapshot capture mode.
LucamTakeSynchronousSnapshots	Simultaneously takes a single image from each of several cameras.
LucamDisableSynchronousSnapshots	Disables the simultaneous snapshot capture mode.

## 2.2.5 Image Saving

Function	Description
LucamConvertFrameToRGB24	Converts a raw frame of data obtained with LucamTakeVideo or LucamTakeSnapshot to a fully processed RGB24 frame suitable for display or saving
LucamConvertFrameToRGB32	Converts a raw frame of data obtained with LucamTakeVideo or LucamTakeSnapshot to a fully processed RGB32 frame suitable for display or saving
LucamConvertFrameToRgb48	Converts a raw frame of data obtained with LucamTakeVideo or LucamTakeSnapshot to a fully processed RGB48 frame suitable for saving in an image format that supports 16 bits per color channel (e.g. TIFF format).
LucamConvertBmp24ToRgb24	Converts a frame of data from the format returned by LucamConvertRgb24 (BGR) to standard format (RGB).
LucamSaveImage	Saves a single image or video frame to disk in one of several formats.

LucamSaveImageW	Exactly like LucamSaveImage but the input filename string is in Unicode (wide character) format.
-----------------	--

## 2.2.6 Callback Handling

Function	Description
LucamAddSnapshotCallback	Allows the user to add a data filter callback function, which is called after each hardware triggered snapshot is returned from the camera but before it's processed.
LucamRemoveSnapshotCallback	Removes the specified data filter callback function registered using the function LucamAddSnapshotCallback.
LucamAddStreamingCallback	Allows the user to add a video filter callback function, which is called after each frame of raw streaming video is returned from the camera but before it's processed.
LucamQueryRgbPreviewPixelFormat	Returns the pixel format for the preview window.
LucamRemoveStreamingCallback	Removes the specified video filter callback function registered using the function LucamAddStreamingCallback.
LucamAddRgbPreviewCallback	Allows the user to add a video filter callback function, which is called after each frame of streaming video is returned from the camera and after it's processed.
LucamRemoveRgbPreviewCallback	Removes the specified video filter callback function registered using the function LucamAddRgbPreviewCallback.

## 2.2.7 Register and External I/O Access

Function	Description
LucamGpioRead	Reads the General Purpose I/O register for the external header status.
LucamGpioWrite	Writes to the General Purpose I/O register to trigger the external header output.
LucamGpoSelect	Enables and disables the alternate GPO functionality.
LucamReadRegister	Reads the internal camera registers.
LucamWriteRegister	Writes to the internal camera registers.





## Detailed API Description

### LucamAddRgbPreviewCallback

Allows the user to add a video filter callback function, which is called after each frame of streaming video is returned from the camera and after it's processed.

#### Usage

```
LONG LucamAddRgbPreviewCallback(HANDLE hCamera,
    VOID (__stdcall *VideoFilter)(
    VOID *pContext,
    BYTE *pData,
    ULONG dataLength,
    ULONG unused),
    VOID *pCBContext,
    ULONG rgbPixelFormat);
```

#### Parameters

hCamera	[in] handle to the camera
*VideoFilter	[in] pointer to the callback function
*pContext	[in] pointer to the context data
*pData	[out] video frame data returned from the camera
dataLength	[in] size of video frame in bytes
unused	[in] reserved for future use
*pCBContext	[in] pointer to the callback context data
rgbPixelFormat	[in] pixel format of data

#### Return Values

If the function succeeds, the return value is the unique callback registration number.

If the function fails, the return value is -1.

#### Remarks

The pixel format is one of LUCAM\_PF\_24 or LUCAM\_PF\_32 and should match the format of the video. You can use `LucamQueryRgbPreviewPixelFormat` to get the video pixel format.

## LucamAddSnapshotCallback

Allows the user to add a data filter callback function, which is called after each hardware triggered snapshot is returned from the camera but before it's processed.

### Usage

```
LONG LucamAddRgbPreviewCallback( HANDLE hCamera,
    VOID (__stdcall *SnapshotCallback) (
    VOID *pContext,
    BYTE *pData,
    ULONG dataLength),
    VOID *pCBContext);
```

### Parameters

hCamera	[in] handle to the camera
*SnapshotCallback	[in] pointer to the callback function
*pContext	[in] pointer to the context data
*pData	[out] video frame data returned from the camera
dataLength	[in] size of video frame in bytes
*pCBContext	[in] pointer to the callback context data

### Return Values

If the function succeeds, the return value is the unique callback registration number.  
If the function fails, the return value is -1.

### Remarks

None.

## LucamAddStreamingCallback

Allows the user to add a video filter callback function, which is called after each frame of streaming video is returned from the camera.

### Usage

```
LONG LucamAddStreamingCallback(HANDLE hCamera,
    VOID (__stdcall *VideoFilter) (
```

```
VOID *pContext,  
BYTE *pData,  
ULONG dataLength),  
VOID *pCBContext);
```

### Parameters

hCamera [in] handle to the camera  
\*VideoFilter [in] pointer to the callback function  
\*pContext [in] pointer to the context data  
\*pData [out] video frame data returned from the camera  
dataLength [in] size of video frame in bytes  
\*pCBContext [in] pointer to the callback context data

### Return Values

If the function succeeds, the return value is the unique callback registration number.  
If the function fails, the return value is -1.

### Remarks

None.

## LucamCameraClose

Closes a connection to a Lumenera camera.

### Usage

```
BOOL LucamCameraClose(HANDLE hCamera);
```

### Parameters

hCamera [in] handle to the camera

### Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

### Remarks

None.

## LucamCameraOpen

Opens a connection to a Lumenera camera.

### Usage

```
HANDLE LucamCameraOpen(ULONG cameraNumber);
```

### Parameters

cameraNumber [in] camera number

### Return Values

If the function succeeds, the return value is a handle to the Lumenera camera attached to the computer.

If the function fails, the return value is NULL.

### Remarks

*LucamNumCameras* may be called to determine the number of cameras connected to the bus. Valid camera numbers are in the range 1 through the value returned from *LucamNumCameras*.

## LucamCameraReset

Resets camera to power-on default state.

### Usage

```
BOOL LucamCameraReset(HANDLE hCamera);
```

### Parameters

hCamera [in] handle to the camera

### Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

### Remarks

None.

## LucamConvertBmp24ToRgb24

Converts a frame of data from the format returned by LucamConvertRgb24 (BGR) to standard format (RGB).

### Usage

```
BOOL LucamConvertRgb24ToRgb24 (UCHAR *pFrame,  
                               ULONG width,  
                               ULONG height);
```

### Parameters

*pFrame	[in] pointer to the buffer containing the frame of data
width	[in] width in pixels for frame of data
height	[in] height in pixels for frame of data

### Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

### Remarks

None.

## LucamConvertFrameToRGB24

Converts a raw frame of data obtained with LucamTakeVideo or LucamTakeSnapshot to a fully processed RGB24 frame suitable for display or saving.

### Usage

```
BOOL LucamConvertFrameToRgb24 (HANDLE hCamera,  
                               BYTE *pDest,  
                               BYTE *pSrc,  
                               ULONG width,  
                               ULONG height,  
                               ULONG pixelFormat,  
                               LUCAM_CONVERSION *pParams);
```

### Parameters

hCamera	[in] handle to the camera
*pDest	[out] processed image data in RGB24 format

\*pSrc [in] image data to be processed from LucamTakeVideo or LucamTakeSnapshot  
width [in] width in pixels for frame of data  
height [in] height in pixels for frame of data  
pixelFormat [in] pixel format of source data  
\*pParams [in] structure containing the options for converting the data

### Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

### Remarks

The LUCAM\_CONVERSION structure is described in Appendix I. The available pixel formats are listed in Appendix I. The RGB24 data format has 24 bits per pixel. The three bytes of each pixel are the Blue, Green, Red values in that order.

## LucamConvertFrameToRGB32

Converts a raw frame of data obtained with LucamTakeVideo or LucamTakeSnapshot to a fully processed RGB32 frame suitable for display or saving.

### Usage

```
BOOL LucamConvertFrameToRgb32 (HANDLE hCamera,
                               BYTE *pDest,
                               BYTE *pSrc,
                               ULONG width,
                               ULONG height,
                               ULONG pixelFormat,
                               LUCAM_CONVERSION *pParams);
```

### Parameters

hCamera [in] handle to the camera  
\*pDest [out] processed image data in RGB32 format  
\*pSrc [in] image data to be processed from LucamTakeVideo or LucamTakeSnapshot  
width [in] width in pixels for frame of data  
height [in] height in pixels for frame of data  
pixelFormat [in] pixel format of source data  
\*pParams [in] structure containing the options for converting the data

## Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

## Remarks

The LUCAM\_CONVERSION structure is described in Appendix I. The available pixel formats are listed in Appendix I. The RGB32 data format has 32 bits per pixel. The first three bytes of each pixel are the Blue, Green, Red values respectively. The last byte is the Alpha channel and is set to zero.

## LucamConvertFrameToRGB48

Converts a raw frame of data obtained with LucamTakeVideo or LucamTakeSnapshot to a fully processed RGB48 frame suitable for saving in an image format that supports 16 bits per color channel (e.g. TIFF format).

## Usage

```
BOOL LucamConvertFrameToRgb48 (HANDLE hCamera,  
                                USHORT *pDest,  
                                USHORT *pSrc,  
                                ULONG width,  
                                ULONG height,  
                                ULONG pixelFormat,  
                                LUCAM_CONVERSION *pParams);
```

## Parameters

hCamera	[in] handle to the camera
*pDest	[out] processed image data in RGB24 format
*pSrc	[in] image data to be processed from LucamTakeVideo or LucamTakeSnapshot
width	[in] width in pixels for frame of data
height	[in] height in pixels for frame of data
pixelFormat	[in] pixel format of source data
*pParams	[in] structure containing the options for converting the data

## Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

**Remarks**

The LUCAM\_CONVERSION structure is described in Appendix I. The pixel format should be LUCAM\_PF\_16.

**LucamCreateDisplayWindow**

Creates a display window, managed by the API, for displaying video.

**Usage**

```
BOOL LucamCreateDisplayWindow(HANDLE hCamera,
                              LPCTSTR lpTitle,
                              ULONG dwStyle,
                              int x,
                              int y,
                              int width,
                              int height,
                              HWND parentWnd,
                              HMENU childId);
```

**Parameters**

hCamera	[in] handle to the camera
lpTitle	[in] title of window that appears in window frame
dwStyle	[in] window style (default is WS_OVERLAPPEDWINDOW WS_VISIBLE)
x	[in] x coordinate on desktop where upper left corner of window will appear (default is 0)
y	[in] y coordinate on desktop where upper left corner of window will appear (default is 0)
width	[in] width of window in pixels (default is 0)
height	[in] height of window in pixels (default is 0)
parentWnd	[in] handle to the parent window for the dialog (default is NULL)
childId	[in] id of child menu (default is NULL)

**Return Values**

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

**Remarks**

The window will automatically resize to the video frame size whenever the video frame size is changed.

## LucamDestroyDisplayWindow

Destroys the display window created with *LucamCreateDisplayWindow*.

### Usage

```
BOOL LucamDestroyDisplayWindow(HANDLE hCamera);
```

### Parameters

hCamera [in] handle to the camera

### Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

### Remarks

None.

## LucamDisableFastFrames

Disables the fast snapshot capture mode.

### Usage

```
BOOL LucamDisableFastFrames(HANDLE hCamera);
```

### Parameters

hCamera [in] handle to the camera

### Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

### Remarks

If the camera was streaming when *LucamEnableFastFrames* was called, streaming will be restored when *LucamDisableFastFrames* is called.

## LucamDisableSynchronousSnapshots

Disables the simultaneous snapshot capture mode.

## Usage

```
BOOL LucamDisableSynchronousSnapshots(  
    HANDLE syncSnapsHandle);
```

## Parameters

syncSnapsHandle [in] handle returned from  
*LucamEnableSynchronousSnapshots* function

## Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

## Remarks

None.

## LucamDisplayPropertyPage

Pops up a Direct Show dialog with the camera properties.

## Usage

```
BOOL LucamDisplayPropertyPage(HANDLE hCamera,  
    HWND parentWnd);
```

## Parameters

hCamera [in] handle to the camera  
parentWnd [in] handle to the parent window for the dialog

## Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

## Remarks

None.

## LucamDisplayVideoFormatPage

Pops up a Direct Show dialog with the video properties.

## Usage

```
BOOL LucamDisplayVideoFormatPage(HANDLE hCamera,  
                                 HWND parentWnd);
```

## Parameters

hCamera [in] handle to the camera  
parentWnd [in] handle to the parent window for the dialog

## Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

## Remarks

None.

## LucamQueryDisplayFrameRate

Returns the actual displayed frame rate of the camera.

## Usage

```
BOOL LucamQueryDisplayFrameRate(HANDLE hCamera,  
                                FLOAT *pValue);
```

## Parameters

hCamera [in] handle to the camera  
\*pValue [out] actual frame rate being displayed in display window

## Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

## Remarks

None.

## LucamEnableFastFrames

Enables the fast snapshot capture mode.

## Usage

```
BOOL LucamEnableFastFrames(HANDLE hCamera
                           LUCAM_SNAPSHOT *pSettings);
```

## Parameters

hCamera [in] handle to the camera  
 \*pSettings [in] structure containing settings to use for the snapshot

## Return Values

If the function succeeds, the return value is TRUE.  
 If the function fails, the return value is FALSE.

## Remarks

The LUCAM\_SNAPSHOT structure is described in Appendix I. If video is streaming when a snapshot is taken, the stream will automatically be stopped (pausing video in the display window if present) before the snapshot is taken. It is not restarted after the snapshot is taken.

## LucamEnableSynchronousSnapshots

Enables the simultaneous snapshot capture mode.

## Usage

```
HANDLE LucamEnableSynchronousSnapshots(
        ULONG numberOfCameras,
        HANDLE *phCameras,
        LUCAM_SNAPSHOT **ppSettings);
```

## Parameters

NumberOfCameras [in] number of cameras to synchronously capture  
 \*phCamera [in] handles to the cameras  
 \*\*ppSettings [in] array of pointers to structures containing settings to use for the snapshot of each camera

## Return Values

If the function succeeds, the return value is a handle.  
 If the function fails, the return value is NULL.

## Remarks

None.

## LucamEnumAvailableFrameRates

Returns an array containing the available frame rates for the camera based on the clock rates available on the camera.

### Usage

```
ULONG LucamEnumAvailableFrameRates(HANDLE hCamera,  
                                     ULONG entryCount,  
                                     FLOAT *pAvailableFrameRates);
```

### Parameters

hCamera	[in] handle to the camera
entrycount	[out] number of available frame rates in array
*pAvailableFrameRates	[out] array of available frame rates

### Return Values

If the function succeeds, the return value is 1.

If the function fails, the return value is 0.

### Remarks

Frame rates are in frames per second.

## LucamEnumCameras

Returns the version information and serial numbers for all Lumenera cameras attached to the computer.

### Usage

```
ULONG LucamEnumCameras(LUCAM_VERSION *pVersionsArray,  
                        ULONG arrayCount);
```

### Parameters

*pVersionArray	[out] pointer to array of version structures
arrayCount	[in] number of version structures to return

**Return Values**

If the function succeeds, the return value is the number of version structures that contain valid information.  
If the function fails, the return value is -1.

**Remarks**

None.

**LucamGetCameraId**

Gets the camera model ID number.

**Usage**

```
BOOL LucamGetCameraId(HANDLE hCamera, ULONG *pId);
```

**Parameters**

hCamera                   [in] handle to the camera  
\*pId                       [out] pointer to the camera model ID

**Return Values**

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

**Remarks**

The following table shows the IDs for each camera model:

Camera Model	ID
Lu050M, Lu055M	0x91
Lu050C, Lu055C	0x95
Lu065C	0x93
Lu070M, Lu075M, Lu070C, Lu075C	0x8C
Lu100M, Lu105M, Lu100C, Lu105C	0x92
Lu110M, Lu115M, Lu110C, Lu115C	0x94
Lu120M, Lu125M, Lu120C, Lu125C	0x96
Lu130M, Lu135M, Lu130C, Lu135C	0x9A
Lu160M, Lu165M, Lu160C, Lu165C	0x8A
Lu170M, Lu175M, Lu170C, Lu175C	0x9E
Lu200C, Lu205C	0x97
Lu270C, Lu275C	0x8D
Lu330C, Lu335C	0x9B
Lu370C, Lu375C	0x8B

Infinity X	0xA0
Infinity 1	0xA1
Infinity 2	0xA2
Infinity 3	0xA3
Infinity 4	0xA4

## LucamGetCurrentMatrix

Gets the current color correction matrix being applied for video preview.

### Usage

```
BOOL LucamGetCurrentMatrix(HANDLE hCamera, FLOAT *pMatrix);
```

### Parameters

hCamera                   [in] handle to the camera  
\*pMatrix                   [out] pointer to array of coefficients

### Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

### Remarks

The matrix is a 9-element array in a 3 x 3 format.

## LucamGetFormat

Gets the video frame format (subwindow position and size, subsampling, pixel format) and desired frame rate for the video data.

### Usage

```
BOOL LucamGetFormat(HANDLE hCamera,  
                    LUCAM_FRAME_FORMAT *format,  
                    FLOAT *pFrameRate);
```

### Parameters

hCamera                   [in] handle to the camera  
\*format                   [out] video frame format  
\*pFrameRate               [out] frame rate for streaming video

## Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

## Remarks

The origin of the imager is the top left corner. The LUCAM\_FRAME\_FORMAT structure is described in Appendix I. This function can be called immediately after *LucamOpenCamera* to get the default values for the video format parameters.

## LucamGetLastError

Gets the specific error code for the last error that occurred when calling an API function.

## Usage

```
ULONG LucamGetLastError(void);
```

## Parameters

None.

## Return Values

The last error that occurred for a call to an API function is returned.

## Remarks

Error codes can be found in the *lucamerror.h* file.

## LucamGetProperty

Gets the value of the specified camera property.

## Usage

```
BOOL LucamGetProperty(HANDLE hCamera,  
                     ULONG property,  
                     FLOAT *pValue,  
                     LONG *pFlags);
```

## Parameters

hCamera	[in] handle to the camera
property	[in] camera property
*pValue	[out] value of camera property
*pFlags	[out] capability flags for property

## Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

## Remarks

The allowable properties are listed in Appendix I. Not all properties are supported by all cameras. If a property is unsupported the function will return a fail condition (FALSE) and the value of \*pValue will be undefined. The allowable capability flags are listed in Appendix I.

## LucamGpioRead

Reads the General Purpose I/O register to obtain the external header status.

## Usage

```
BOOL LucamGpioRead( HANDLE hCamera,  
                   BYTE *pGpoValues,  
                   BYTE *pGpiValues);
```

## Parameters

hCamera	[in] handle to the camera
*pGpoValues	[out] value of the output bits of the register
*pGpiValues	[out] value of the input bits of the register

## Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

## Remarks

None.

## LucamGpioWrite

Writes to the General Purpose I/O register to trigger the external header output.

### Usage

```
BOOL LucamGpioWrite(HANDLE hCamera,  
                    BYTE GpoValues);
```

### Parameters

HCamera [in] handle to the camera  
GpoValues [in] value of the output bits of the register

### Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

### Remarks

None.

## LucamGpoSelect

Enables and disables the alternate GPO functionality.

### Usage

```
BOOL LucamGpoSelect(HANDLE hCamera,  
                    BYTE gpoEnable);
```

### Parameters

hCamera [in] handle to the camera  
pGpoEnable [in] bit flags used to disable/enable alternate functionality

### Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

### Remarks

Setting the appropriate bit to 1 enables the manual toggling of the specific GPO using the LucamGPIOWrite function.

Bit 0: enables GPO1

Bit 1: enables GPO2

Bit 2: enables GPO3

Bit 3: enables GPO4

Setting the bit to 0 puts the GPO into its default mode, which will automatically output a signal, based on its underlying definition. The definitions of the GPOs are described in the User's Manual.

## LucamNumCameras

Returns the number of Lumenera cameras attached to the computer.

### Usage

```
LONG LucamNumCameras(void);
```

### Parameters

None.

### Return Values

If the function succeeds, the return value is the number of Lumenera cameras attached to the computer.

If the function fails, the return value is -1.

### Remarks

None.

## LucamOneShotAutoExposure

Performs a single (one iteration) exposure adjustment in an attempt to reach the autoexposure target.

### Usage

```
LONG LucamOneShotAutoExposure(HANDLE hCamera,  
                                UCHAR target,  
                                ULONG startX,  
                                ULONG startY,  
                                ULONG width,  
                                ULONG height);
```

### Parameters

hCamera	[in] handle to the camera
target	[in] target average brightness (0-255)

startX	[in] X position of top left corner of window to auto expose
startY	[in] Y position of top left corner of window to auto expose
width	[in] width of window to auto expose
height	[in] height of window to auto expose

**Return Values**

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

**Remarks**

None.

## LucamOneShotAutoWhiteBalance

Performs a single (one iteration) gain adjustment in an attempt to color balance the image.

**Usage**

```
LONG LucamOneShotAutoWhiteBalance( HANDLE hCamera,
                                     ULONG startX,
                                     ULONG startY,
                                     ULONG width,
                                     ULONG height);
```

**Parameters**

hCamera	[in] handle to the camera
startX	[in] X position of top left corner of window to auto white balance
startY	[in] Y position of top left corner of window to auto white balance
width	[in] width of window to auto white balance
height	[in] height of window to auto white balance

**Return Values**

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

**Remarks**

None.

## LucamOneShotAutoWhiteBalanceEx

Performs a single (one iteration) gain adjustment in an attempt to color balance the image to a specific target color.

### Usage

```
LONG LucamOneShotAutoWhiteBalanceEx (HANDLE hCamera,  
                                       FLOAT redOverGreen,  
                                       FLOAT blueOverGreen,  
                                       ULONG startX,  
                                       ULONG startY,  
                                       ULONG width,  
                                       ULONG height);
```

### Parameters

hCamera	[in] handle to the camera
startX	[in] X position of top left corner of window to auto white balance
startY	[in] Y position of top left corner of window to auto white balance
width	[in] width of window to auto white balance
height	[in] height of window to auto white balance

### Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

### Remarks

Sometimes it is desirable to perform a color balance to achieve some non-white target. An example is on a microscope where the background may be slightly yellow or blue depending on the light source. In order to ensure the camera images match what is seen down the eyepiece, set redOverGreen and blueOverGreen to values that match the Red over Green and Blue over Green components of the background color. For example, if the background color has R, G & B values of 255, 250, 230, set redOverGreen to 1.02 and blueOverGreen to 0.92.

To balance to white, set redOverGreen and blueOverGreen to 1.0.

## LucamPermanentBufferRead

Reads data from the user-defined non-volatile memory area of the camera.

**Usage**

```

BOOL LucamPermanentBufferRead(HANDLE hCamera,
                               UCHAR *pBuf,
                               ULONG offset,
                               ULONG length);

```

**Parameters**

hCamera	[in] handle to the camera
*pBuf	[out] buffer to return data to
offset	[in] offset in bytes from start of memory area
length	[in] length of data buffer to read

**Return Values**

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

**Remarks**

The non-volatile memory area is 1024 bytes long.

## LucamPermanentBufferWrite

Writes data to the user-defined non-volatile memory area of the camera.

**Usage**

```

BOOL LucamPermanentBufferWrite(HANDLE hCamera,
                                UCHAR *pBuf,
                                ULONG offset,
                                ULONG length);

```

**Parameters**

hCamera	[in] handle to the camera
*pBuf	[in] buffer containing data to write into memory
offset	[in] offset in bytes from start of memory area
length	[in] length of data buffer to write

**Return Values**

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

## Remarks

The non-volatile memory area is 1024 bytes long.

## LucamPropertyRange

Returns the range of valid values for a camera property and its default value.

## Usage

```
BOOL LucamPropertyRange(HANDLE hCamera,  
                        ULONG property,  
                        FLOAT *pMin,  
                        FLOAT *pMax,  
                        FLOAT *pDefault,  
                        LONG *pFlags);
```

## Parameters

hCamera	[in] handle to the camera
property	[in] camera property
*pMin	[out] minimum valid value of camera property
*pMax	[out] maximum valid value of camera property
*pDefault	[out] default value of camera property
*pFlags	[out] capability flags for property

## Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

## Remarks

The allowable properties are listed in Appendix I. Not all properties are supported by all cameras. If a property is unsupported the function will return a fail condition (FALSE). The allowable capability flags are listed in Appendix I.

## LucamQueryDisplayFrameRate

Returns the actual average displayed frame rate of the camera since preview was started.

## Usage

```
BOOL LucamQueryDisplayFrameRate ( HANDLE hCamera,
```

```
    FLOAT *pValue);
```

### Parameters

hCamera [in] handle to the camera  
\*pValue [out] average frame rate in frames per second

### Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

### Remarks

None

## LucamQueryExternInterface

Returns the type of interface between the camera and the computer.

### Usage

```
BOOL LucamQueryExternInterface(HANDLE hCamera,  
                               ULONG *pExternInterface);
```

### Parameters

hCamera [in] handle to the camera  
\*pExternInterface [out] pointer containing the external interface type

### Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

### Remarks

The External Interfaces are listed in Appendix I.

## LucamQueryRgbPreviewPixelFormat

Returns the pixel format for the preview window.

### Usage

```
BOOL LucamQueryRgbPreviewPixelFormat(HANDLE hCamera,
```

```
ULONG *pRgbPixelFormat);
```

### Parameters

hCamera [in] handle to the camera  
\*pRgbPixelFormat [out] pointer containing the preview pixel format

### Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

### Remarks

This pixel format is used when registering Preview Callbacks using LucamAddRgbPreviewCallback. The pixel formats are listed in Appendix I.

## LucamQueryVersion

Returns version information about the camera.

### Usage

```
BOOL LucamQueryVersion(HANDLE hCamera,  
                       LUCAM_VERSION *pVersion);
```

### Parameters

hCamera [in] handle to the camera  
\*pVersion [out] pointer to a structure containing version information

### Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

### Remarks

The LUCAM\_VERSION structure is described in Appendix I.

## LucamReadRegister

Reads the internal camera registers.

## Usage

```
BOOL LucamReadRegister(HANDLE hCamera,  
                      LONG address,  
                      LONG numReg,  
                      LONG *pValue);
```

## Parameters

hCamera	[in] handle to the camera
address	[in] starting register address
numReg	[in] number of contiguous registers to read
*pValue	[out] value(s) of register(s)

## Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

## Remarks

None.

## LucamRemoveRgbPreviewCallback

Removes the specified video filter callback function registered using the function *LucamAddRgbPreviewCallback*.

## Usage

```
BOOL LucamRemoveRgbPreviewCallback(HANDLE hCamera,  
                                   LONG callbackId);
```

## Parameters

hCamera	[in] handle to the camera
callbackId	[in] callback ID returned from <i>LucamAddRgbPreviewCallback</i>

## Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

## Remarks

None.

## LucamRemoveSnapshotCallback

Removes the specified data filter callback function registered using the function `LucamAddSnapshotCallback`.

### Usage

```
BOOL LucamRemoveSnapshotCallback( HANDLE hCamera,  
                                  LONG callbackId);
```

### Parameters

`hCamera` [in] handle to the camera  
`callbackId` [in] callback ID returned from *LucamAddSnapshotCallback*

### Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

### Remarks

None.

## LucamRemoveStreamingCallback

Removes the specified video filter callback function registered using the function `LucamAddStreamingCallback`.

### Usage

```
BOOL LucamRemoveStreamingCallback(HANDLE hCamera,  
                                   LONG callbackId);
```

### Parameters

`hCamera` [in] handle to the camera  
`callbackId` [in] callback ID returned from *LucamAddStreamingCallback*

### Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

### Remarks

None.

## LucamSaveImage

Saves a single image or video frame to disk in one of several formats.

### Usage

```
BOOL LucamSaveImage(ULONG width,  
                    ULONG height,  
                    ULONG pixelformat,  
                    BYTE *pData,  
                    CHAR *pFilename);
```

### Parameters

width	[in] width of image in pixels
height	[in] height of image in pixels
pixelFormat	[in] pixel format of image data
*pData	[in] image data to save
*pFilename	[in/out] filename for saved image

### Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

### Remarks

The filename extension indicates the format the file will be saved in. Supported image formats are Windows bitmap(.bmp), Tagged Image File Format (.tif) and Raw (.raw). The available pixel formats are listed in Appendix I. If an unsupported file type (indicated by its extension) is provided, the function will fail.

## LucamSaveImageW

Saves a single image or video frame to disk in one of several formats.

### Usage

```
BOOL LucamSaveImage(ULONG width,  
                    ULONG height,  
                    ULONG pixelformat,  
                    BYTE *pData,  
                    WCHAR *pFilename);
```

## Parameters

width [in] width of image in pixels  
height [in] height of image in pixels  
pixelFormat [in] pixel format of image data  
\*pData [in] image data to save  
\*pFilename [in/out] filename for saved image in Unicode string format

## Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

## Remarks

The filename extension indicates the format the file will be saved in.

Supported image formats are Windows bitmap(.bmp), Tagged Image File Format (.tif) and Raw (.raw).

The available pixel formats are listed in Appendix I.

If an unsupported file type (indicated by its extension) is provided, the function will fail.

## LucamSetFormat

Sets the video frame format (subwindow position and size, subsampling, pixel format) and desired frame rate for the video data.

## Usage

```
BOOL LucamSetFormat(HANDLE hCamera,  
                    LUCAM_FRAME_FORMAT *format,  
                    FLOAT frameRate);
```

## Parameters

hCamera [in] handle to the camera  
\*format [in] video frame format  
frameRate [in] frame rate for streaming video

## Return Values

If the function succeeds, the return value is TRUE.

If the function fails, the return value is FALSE.

## Remarks

The origin of the imager is the top left corner. The LUCAM\_FRAME\_FORMAT structure is described in Appendix I.

Each dimension of the subwindow must be evenly divisible by 8.

## LucamSetProperty

Sets the value of the specified camera property.

### Usage

```
BOOL LucamSetProperty(HANDLE hCamera,  
                      ULONG property,  
                      FLOAT value,  
                      LONG flags);
```

### Parameters

hCamera	[in] handle to the camera
property	[in] camera property
value	[in] value of camera property
flags	[in] capability flags for property

### Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

### Remarks

The allowable properties are listed in Appendix I. Not all properties are supported by all cameras. If a property is unsupported the function will return a fail condition (FALSE). The allowable capability flags are listed in Appendix I. If a capability flag is not supported by the property, it is silently ignored.

## LucamSetup8bitsLUT

Populates the 8-bit LUT inside the camera.

### Usage

```
BOOL LucamSetup8bitsLUT( HANDLE hCamera,  
                        UCHAR *pLut,  
                        ULONG length);
```

### Parameters

hCamera	[in] handle to the camera
---------	---------------------------

\*pLut           [in] pointer to LUT values  
length         [in] number of LUT values

### Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

### Remarks

The length of the LUT must be 0 (to disable it) or 256.

## LucamSetupCustomMatrix

Defines the color correction matrix values to use when converting raw data to RGB24 with the correction matrix parameter LUCAM\_CM\_CUSTOM.

### Usage

```
BOOL LucamSetupCustomMatrix( HANDLE hCamera,  
                             FLOAT *pMatrix);
```

### Parameters

hCamera       [in] handle to the camera  
\*pMatrix      [in] processed image data in RGB24 format

### Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

### Remarks

The *LucamConvertFrameToRgb24* function requires a color correction matrix parameter. The pre-defined ones may be used, but when a specific matrix is required, the LUCAM\_CM\_CUSTOM parameter can be passed and the values defined using this function (*LucamSetupCustomMatrix*) will be used.

## LucamStreamVideoControl

Controls the streaming video.

## Usage

```
BOOL LucamStreamVideoControl(HANDLE hCamera,  
                             ULONG controlType,  
                             HWND hWnd);
```

## Parameters

hCamera [in] handle to the camera  
controlType [in] control type parameter  
hWnd [in] handle to the window to stream video to

## Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

## Remarks

Valid control types are STOP\_STREAMING, PAUSE\_STREAM, START\_STREAMING and START\_DISPLAY. START\_DISPLAY will start the video streaming and display it in the specified window. This can be the window created with *LucamCreateDisplayWindow* or the user's own window. START\_STREAMING simply causes video to stream without being displayed.

## LucamTakeFastFrame

Takes a single image from the camera, using the camera's still imaging or video mode.

## Usage

```
BOOL LucamTakeFastFrame(HANDLE hCamera,  
                        BYTE *pData);
```

## Parameters

hCamera [in] handle to the camera  
\*pData [out] image data returned from the camera

## Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

## Remarks

None.

## LucamTakeSnapshot

Takes a single image from the camera, using the camera's still imaging or video mode.

## Usage

```
BOOL LucamTakeSnapshot(HANDLE hCamera,  
                       LUCAM_SNAPSHOT *pSettings,  
                       BYTE *pData);
```

## Parameters

hCamera [in] handle to the camera  
\*pSettings [in] structure containing settings to use for the snapshot  
\*pData [out] image data returned from the camera

## Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

## Remarks

The LUCAM\_SNAPSHOT structure is described in Appendix I. If video is streaming when a snapshot is taken, the stream will automatically be stopped (pausing video in the display window if present) before the snapshot is taken and then restarted after the snapshot is taken.

This function is equivalent to calling the following three functions in succession: *LucamEnableFastFrames*, *LucamTakeFastFrame*, *LucamDisableFastFrames*

## LucamTakeSynchronousSnapshots

Simultaneously takes a single image from each of several cameras.

## Usage

```
BOOL LucamTakeSynchronousSnapshots(  
                                     HANDLE syncSnapsHandle,  
                                     BYTE **ppBuffers);
```

## Parameters

syncSnapsHandle [in] handle to the camera  
\*\*ppBuffers [out] array of pointers to image data returned from the camera

## Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

## Remarks

None.

## LucamTakeVideo

Takes video frames from the camera, using the camera's video mode.

## Usage

```
BOOL LucamTakeVideo(HANDLE hCamera,  
                    LONG numFrames,  
                    BYTE *pData);
```

## Parameters

hCamera [in] handle to the camera  
numFrames [in] number of video frames to take  
\*pData [out] video data returned from the camera

## Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

## Remarks

None.

## LucamTakeVideoEx

Takes video data greater than a specified threshold, from the camera, using the camera's video mode, and returns their coordinates.

## Usage

```
BOOL LucamTakeVideoEx(HANDLE hCamera,  
                      BYTE *pDataCoords,  
                      ULONG *pLength,  
                      ULONG timeout);
```

## Parameters

hCamera	[in] handle to the camera
*pDataCoords	[out] coordinates of video data returned from the camera
*pLength	[out] number of bytes of pData
timeout	[in] maximum length of time in milliseconds to wait before returning, if no data is returned

## Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

## Remarks

The returned data is formatted with one data set per pixel that is above the threshold, plus one byte at end of frame

x-location[7:0]  
x-location[15:8]  
y-location[7:0]  
y-location[15:8]  
pixel value[7:0]  
pixel value[15:8] (always 0)  
...  
x-location[7:0]  
x-location[15:8]  
y-location[7:0]  
y-location[15:8]  
pixel value[7:0]  
pixel value[15:8] (always 0)  
...  
0x55

## LucamWriteRegister

Writes the internal camera registers.

## Usage

```
BOOL LucamWriteRegister(HANDLE hCamera,  
                        LONG address,  
                        LONG numReg,  
                        LONG *pValue);
```

## Parameters

hCamera	[in] handle to the camera
address	[in] starting register address
numReg	[in] number of contiguous registers to write
*pValue	[in] value(s) of register(s)

## Return Values

If the function succeeds, the return value is TRUE.  
If the function fails, the return value is FALSE.

## Remarks

None.

## 4

# Appendix I

## 4.1 Constants Definitions

Many of the parameters passed to API functions have been defined as constants in the API header file. The names of these constants and their definition are described in the following sections.

### 4.1.1 Camera Properties

```
LUCAM_PROP_BRIGHTNESS // Brightness
LUCAM_PROP_CONTRAST // Contrast
LUCAM_PROP_HUE // Hue
LUCAM_PROP_SATURATION // Saturation
LUCAM_PROP_SHARPNESS // Sharpness
LUCAM_PROP_GAMMA // Gamma
LUCAM_PROP_PAN // Pan
LUCAM_PROP_TILT // Tilt
LUCAM_PROP_ROLL // Roll
LUCAM_PROP_ZOOM // Zoom
LUCAM_PROP_EXPOSURE // Exposure (mS)
LUCAM_PROP_IRIS // Iris
LUCAM_PROP_FOCUS // Focus
LUCAM_PROP_WHITE_BALANCE_U // White Balance
LUCAM_PROP_WHITE_BALANCE_V // White Balance
LUCAM_PROP_GAIN // Global Gain (factor)
LUCAM_PROP_GAIN_RED // Red Gain (factor)
LUCAM_PROP_GAIN_BLUE // Blue Gain (factor)
LUCAM_PROP_GAIN_GREEN1 // Green 1 Gain (factor)
LUCAM_PROP_GAIN_GREEN2 // Green 2 Gain (factor)
LUCAM_PROP_GAIN_MAGENTA // Magenta Gain (factor)
LUCAM_PROP_GAIN_CYAN // Cyan Gain (factor)
LUCAM_PROP_GAIN_YELLOW1 // Yellow 1 Gain (factor)
LUCAM_PROP_GAIN_YELLOW2 // Yellow 2 Gain (factor)
LUCAM_PROP_COLOR_FORMAT // Color Format (Read Only)
LUCAM_PROP_STILL_DYN_RANGE // Dynamic Range
```

```

LUCAM_PROP_DEMOSAICING_METHOD // Demosaicing Method
LUCAM_PROP_CORRECTION_MATRIX // Color Correction
LUCAM_PROP_FLIPPING           // Flipping Mode
LUCAM_PROP_STILL_KNEE1_EXPOSURE // Knee 1 for multi-
                                // slope integration
LUCAM_PROP_STILL_KNEE2_EXPOSURE // Knee 2 for multi-
                                // slope integration
LUCAM_PROP_STILL_KNEE3_EXPOSURE // Knee 1 for multi-
                                // slope integration
LUCAM_PROP_VIDEO_KNEE         // Knee point for multi-
                                // slope integration in
                                // video mode

LUCAM_PROP_THRESHOLD
LUCAM_PROP_AUTO_EXP_TARGET    // Auto Exposure Target
LUCAM_PROP_TIMESTAMPS
LUCAM_PROP_SNAPSHOT_SETTING
LUCAM_PROP_AUTO_EXP_MAXIMUM   // Maximum exposure in
                                // AEC mode

LUCAM_PROP_STILL_EXPOSURE

```

#### 4.1.2 Capability Flags

```

LUCAM_PROP_FLAG_USE
LUCAM_PROP_FLAG_AUTO           // Automatic mode
LUCAM_PROP_FLAG_STROBE_FROM_START_OF_EXPOSURE
LUCAM_FRAME_FORMAT_FLAGS_BINNING // Binning enabled

```

#### 4.1.3 Pixel Formats

```

LUCAM_PF_8 // 8 bits per pixel (Raw data)
LUCAM_PF_16 // 16 bits per pixel (Raw data)
LUCAM_PF_24 // 24 bits per pixel Color
LUCAM_PF_YUV422 // 16 bits per pixel Color
LUCAM_PF_32 // 32 bits per pixel Color
LUCAM_PF_48 // 48 bits per pixel Color
LUCAM_PF_COUNT //
LUCAM_PF_FILTER //

```

#### 4.1.4 Demosaic Methods

```

LUCAM_DM_NONE // No demosaicing is done
LUCAM_DM_FAST // Fast method (low quality)
LUCAM_DM_HIGH_QUALITY // High Quality (med. speed)

```

```
LUCAM_DM_HIGHER_QUALITY // Higher Quality (slow. speed)
```

#### 4.1.5 Correction Matrices

```
LUCAM_CM_NONE // No color correction performed  
LUCAM_CM_FLUORESCENT // Fluorescent (Office) lighting  
LUCAM_CM_DAYLIGHT // Outdoor lighting  
LUCAM_CM_INCANDESCENT // Incandescent (Home) lighting  
LUCAM_CM_XENON_FLASH // Flash bulb lighting  
LUCAM_CM_HALOGEN // Halogen lighting  
LUCAM_CM_CUSTOM // User defined correction matrix
```

#### 4.1.6 Color Formats

```
LUCAM_CF_MONO  
LUCAM_CF_BAYER_RGGB  
LUCAM_CF_BAYER_GRBG  
LUCAM_CF_BAYER_GBRG  
LUCAM_CF_BAYER_BGGR  
LUCAM_CF_BAYER_CYYM  
LUCAM_CF_BAYER_YCMY  
LUCAM_CF_BAYER_YMCY  
LUCAM_CF_BAYER_MYYC
```

#### 4.1.7 External Interfaces

```
LUCAM_EXTERN_INTERFACE_USB1 // USB1 Interface  
LUCAM_EXTERN_INTERFACE_USB2 // USB2 Interface
```

#### 4.1.8 Shutter Types

```
LUCAM_SHUTTER_TYPE_GLOBAL // Global exposure shutter  
LUCAM_SHUTTER_TYPE_ROLLING // Rolling exposure shutter
```

#### 4.1.9 Image Flipping

```
LUCAM_PROP_FLIPPING_NONE // No flipping  
LUCAM_PROP_FLIPPING_X // Horizontal Flip (Mirror)  
LUCAM_PROP_FLIPPING_Y // Vertical Flip  
LUCAM_PROP_FLIPPING_XY // Horiz. & Vert. Flip
```

### 4.1.10 Video streaming modes

```

STOP_STREAMING      // Stop streaming video
START_STREAMING     // Start streaming video
START_DISPLAY       // Start streaming video and Display
PAUSE_STREAM        // Pause streaming

```

## 4.2 Data Structure Definitions

Several of the parameters passed to API functions have been defined as data structures in the API header file. The names of these structures and the description of their contents are described in the following sections.

### 4.2.1 LUCAM\_SNAPSHOT Structure

```

FLOAT exposure;     // Exposure in milliseconds
FLOAT gain;         // Overall gain as a multiplicative
                   // factor
union {
  struct {
    FLOAT gainRed;   // Gain for Red pixels as multiplicative
                   // factor
    FLOAT gainBlue; // Gain for Blue pixels as multiplicative
                   // factor
    FLOAT gainGrn1; // Gain for Green pixels on Red rows as
                   // multiplicative factor
    FLOAT gainGrn2; // Gain for Green pixels on Blue rows as
                   // multiplicative factor
  }
  struct {
    FLOAT gainMag;   // Gain for Magenta pixels as
                   // multiplicative factor
    FLOAT gainCyan; // Gain for Cyan pixels as multiplicative
                   // factor
    FLOAT gainYel1; // Gain for Yellow pixels on Magenta rows
                   // as multiplicative factor
    FLOAT gainYel2; // Gain for Yellow pixels on Cyan rows as
                   // multiplicative factor
  }
}
union {
  BOOL useStrobe;   // use a flash (backward compatibility)
  ULONG strobeFlags; // use LUCAM_PROP_FLAG_USE and/or

```

```

        LUCAM_PROP_FLAG_STROBE_FROM_START_OF_EX
        POSURE
    }

    FLOAT strobeDelay; // time interval from when exposure
                      // starts to time the flash is fired in
                      // milliseconds
    BOOL useHwTrigger; // wait for hardware trigger flag
    FLOAT timeout;     // maximum time to wait for hardware
                      // trigger prior to returning from function
                      // in milliseconds

    LUCAM_FRAME_FORMAT format; // frame format for data
    ULONG shutterType; // Shutter mode of the camera
    FLOAT exposureDelay; // time interval from when the trigger
                        // occurs to when the exposure starts

    union {
        ULONG ulReserved1; // (backwards compatibility)
        BOOL bufferlastframe; // set to TRUE if you want
                              // TakeFastFrame to return an already
                              // received frame
    };

    ULONG ulReserved2; // Reserved for future use (Must be set
                      // to zero)
    FLOAT flReserved1; // Reserved for future use (Must be set
                      // to zero)
    FLOAT flReserved2; // Reserved for future use (Must be set
                      // to zero)

```

#### 4.2.2 LUCAM\_FRAME\_FORMAT Structure

```

    ULONG xOffset; // x coordinate on imager of top left
                  // corner of subwindow in pixels
    ULONG yOffset; // y coordinate on imager of top left
                  // corner of subwindow in pixels
    ULONG width; // width in pixels of subwindow
    ULONG height; // height in pixels of subwindow
    ULONG pixelFormat; // pixel format for data
    union {
        struct {
            USHORT subSampleX; // sub-sample ratio in x direction in
                              // pixels (x:1)
        }
        struct {

```

```
    USHORT binningX; // binning ratio in x direction in
                    // pixels (x:1)
}
}
USHORT flagsX; // binning flag for x direction
union {
    struct {
        USHORT subSampleY; // sub-sample ratio in y direction
                        // in pixels (y:1)
    }
    struct {
        USHORT binningY; // binning ratio in y direction in
                        // pixels (y:1)
    }
}
USHORT flagsY; // binning flag for y direction
```

#### 4.2.3 LUCAM\_VERSION Structure

```
ULONG firmware; // Firmware version
ULONG fpga; // FPGA version
ULONG api; // API version
ULONG driver; // Device driver version
ULONG serialnumber; // Camera's unique serial number
ULONG reserved; // Reserved for future use
```

#### 4.2.4 LUCAM\_CONVERSION Structure

```
ULONG DemosaicMethod; // Demosaic method to convert
                    // Bayer data to full color
ULONG CorrectionMatrix; // Color correction matrix
```