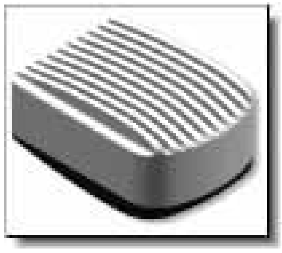
Lumenera USB Camera API Reference Manual

Release 5.0



Lumenera Corporation • 7 Capella Court, Ottawa, ON, Canada • K2E 8A7 • (t) 1.613.736-4077 • (f) 1.613.736-4071 • www.lumenera.com

Table of Contents

1 Introduction

1.1 The Lumenera API

The Lumenera USB Camera API (LuCam 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, Visual C#, Borland C++ Builder and any COM enabled compiler language, in a matter of minutes you are able to create an application to command and control the camera properties, retrieve, display, and capture image data.

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

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

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

If you have purchased the SDK, our Technical Assistance Center (TAC) 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 TAC group at:

support@lumenera.com

Please visit our support website to review our FAQs and browse through our extensive Knowledge Base. Go to www.lumenera.com and follow the Support links for more information.

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 is 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 is 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 is processed.
LucamAdjustDisplayWindow	Allows the user to scale (zoom in/out) the video stream into the preview window.
LucamAdjustWhiteBalanceFromSnapshot	Calculates the appropriate color gain values for snapshot mode.
LucamAutoFocusQueryProgress	Provides the status of the auto focus calibration.
LucamAutoFocusStart	Starts an auto focus calibration.
LucamAutoFocusStop	Stops the auto focus calibration.
LucamAutoFocusWait	Waits for the completion of the auto focus calibration.
LucamAutoRoiGet	Returns the region of interest used for the auto functions.
LucamAutoRoiSet	Sets a region of interest that will be used by the auto functions.
LucamCameraClose	Closes the connection to Lumenera camera.
LucamCameraOpen	Opens a connection to a Lumenera camera.

Function	Description
LucamCameraReset	Resets camera to power-on default state.
LucamCancelTakeFastFrame	Cancels a call to LucamTakeFastFrame(), LucamForceTakeFastFrame(), LucamTakeFastFrameNoTrigger() or LucamTakeSnapshot() made with another programming thread.
LucamCancelTakeVideo	Cancels a call to LucamTakeVideo() or LucamTakeVideoEx() made with another programming thread.
LucamConvertBmp24ToRgb24	Converts a frame of data from the format returned by LucamConvertFrameToRgb24 (BGR) to standard format (RGB).
LucamConvertFrameToGreyscale8	Converts an 8 bit raw frame of data obtained with LucamTakeVideo or LucamTakeSnapshot to a fully processed monochrome frame suitable for display or saving.
LucamConvertFrameToGreyscale8Ex	Converts an 8 bit raw frame of data obtained with LucamTakeVideo or LucamTakeSnapshot to a fully processed monochrome frame suitable for display or saving.
LucamConvertFrameToGreyscale16	Converts a 16 bit raw frame of data obtained with LucamTakeVideo or LucamTakeSnapshot to a fully processed monochrome frame suitable for display or saving.
LucamConvertFrameToGreyscale16Ex	Converts a 16 bit raw frame of data obtained with LucamTakeVideo or LucamTakeSnapshot to a fully processed monochrome frame suitable for display or saving.
LucamConvertFrameToRGB24	Converts a raw frame of data obtained with LucamTakeVideo or LucamTakeSnapshot to a fully processed RGB24 frame suitable for display or saving
LucamConvertFrameToRGB24Ex	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

Function	Description
LucamConvertFrameToRGB32Ex	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).
LucamConvertFrameToRgb48Ex	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).
LucamConvertRawAVIToStdVideo	Converts a raw AVI video file obtained with LucamStreamVideoControlAVI to a standard video format. (e.g. Standard 24-bit AVI).
LucamCreateDisplayWindow	Creates a display window, which is managed by the API, for displaying video.
LucamDestroyDisplayWindow	Destroys the display window created with LucamCreateDisplayWindow.
LucamDigitalWhiteBalance	Performs a single (one iteration) digital color gain adjustment on the video stream in an attempt to color balance the image.
LucamDigitalWhiteBalanceEx	Performs a single (one iteration) digital color gain adjustment on the video stream in an attempt to color balance the image to a specific target color.
LucamDisableFastFrames	Disables the fast snapshot capture mode.
LucamDisableSynchronousSnapshots	Disables the simultaneous snapshot capture mode.
LucamDisplayPropertyPage	Pops up a DirectShow dialog with the camera properties.
LucamDisplayVideoFormatPage	Pops up a DirectShow 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.

Function	Description
LucamEnumCameras	Returns the version information and serial numbers for all Lumenera cameras attached to the computer.
LucamForceTakeFastFrame	Captures a SW trigger snapshot while the camera is in HW triggered Fast Frames mode.
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.
LucamGetImageIntensity	Gets the pixel intensity value of a given image.
LucamGetLastError	Gets the specific error code for the last error that occurred when calling an API function.
LucamGetLastErrorFromCamera	Gets the specific error code for the last error that occurred when calling an API function for a given camera.
LucamGetProperty	Gets the value of the specified camera property.
LucamGetStillImageFormat	Returns the snapshot image format used to capture a snapshot.
LucamGetTruePixelDepth	Gets the actual pixel depth of the camera. This is used when using the camera in 16 bit mode.
LucamGetVideoImageFormat	Returns the video image format used to capture a video frame.
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.
LucamInitAutoLens	Initialize and calibrate the focus and iris positions of the camera lens.
LucamNumCameras	Returns the number of Lumenera cameras attached to the computer.

Function	Description
LucamOneShotAutoExposure	Performs a single (one iteration) exposure adjustment on the video stream in an attempt to reach the auto-exposure target.
LucamOneShotAutoWhiteBalance	Performs a single (one iteration) color gain adjustment on the video stream in an attempt to color balance the image.
LucamOneShotAutoWhiteBalanceEx	Performs a single (one iteration) color gain adjustment on the video stream in an attempt to color balance the image to a specific target color.
LucamPerformDualTapCorrection	Performs an additional correction on a captured image from cameras that have more than one sensor readout taps.
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.
LucamPreviewAVIClose	Closes the connection to an AVI file.
LucamPreviewAVIControl	Controls the previewing of a raw AVI video.
LucamPreviewAVIGetDuration	Returns the length of an open AVI file.
LucamPreviewAVIGetFormat	Returns the AVI file information.
LucamPreviewAVIGetFrameCount	Returns the total number of frames within the opened AVI file.
LucamPreviewAVIGetFrameRate	Returns the recorded frame rate of the AVI file.
LucamPreviewAVIGetPositionFrame	Returns the current frame based position within the AVI file.
LucamPreviewAVIGetPositionTime	Returns the current time based position within the AVI file.
LucamPreviewAVISetPositionFrame	Sets the current frame based position within the AVI file.
LucamPreviewAVISetPositionTime	Sets the current time based position within the AVI file.
LucamPreviewAVIOpen	Opens an AVI file for previewing. The control of the video playback can be done through the LucamPreviewAVIControl function.
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.

Function	Description
LucamQueryRgbPreviewPixelFormat	Returns the pixel format for the preview window.
LucamQueryVersion	Returns version information about the camera.
LucamReadRegister	Reads the internal camera registers.
LucamRegisterEventNotification	Registers an event handle with the LuCam API
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.
LucamSaveImageEx	Saves a single image or video frame to disk in one of several formats. This function takes into consideration the camera’s Endianness for 16 bit data.
LucamSaveImageW	Exactly like LucamSaveImage but the input filename string is in Unicode (wide character) format.
LucamSaveImageWEx	Exactly like LucamSaveImage but the input filename string is in Unicode (wide character) format. This function takes into consideration the camera’s Endianness for 16 bit data.
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.
LucamSetTriggerMode	Toggles between SW triggered and HW triggered snapshots in Fast Frames mode.
LucamSetTimeout	Updates the timeout value that was originally set for LucamTakeVideo() or the value set in the LUCAM_SNAPSHOT structure while the camera is in Fast Frames mode.

Function	Description
LucamSetup8bitsColorLUT	Populates the 8 bit LUT inside the camera for each color channel.
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.
LucamStreamVideoControlAVI	Controls the capture of the video to an AVI file.
LucamTakeFastFrame	Takes a single image from the camera, using the camera’s still imaging or video mode.
LucamTakeFastFrameNoTrigger	Retrieves a previously taken single image from the camera, using the camera’s still imaging 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.
LucamTriggerFastFrame	Initiates the request to take a snapshot.
LucamUnregisterEventNotification	Deregisters an event with the Lucam API
LucamWriteRegister	Writes to the internal camera registers.

2.2 API Function Summary Grouped by Task

2.2.1 Initialization and Termination

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.

Function	Description
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.
LucamGetLastErrorFromCamera	Gets the specific error code for the last error that occurred when calling an API function for a given camera.

2.2.2 Camera Settings

Function	Description
LucamAutoRoiGet	Returns the region of interest used for the auto functions.
LucamAutoRoiSet	Sets a region of interest that will be used by the auto functions.
LucamGetCameraID	Gets the camera model ID number.
LucamGetProperty	Gets the value of the specified camera property.
LucamGetStillImageFormat	Returns the snapshot image format used to capture a snapshot.
LucamGetVideoImageFormat	Returns the video image format used to capture a video frame.
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 on the video stream in an attempt to reach the autoexposure target.
LucamOneShotAutoWhiteBalance	Performs a single (one iteration) on-chip, analog color gain adjustment on the video stream in an attempt to color balance the image.

Function	Description
LucamOneShotAutoWhiteBalanceEx	Performs a single (one iteration) on-chip, analog color gain adjustment on the video stream in an attempt to color balance the image to a specific target color.
LucamDigitalWhiteBalance	Performs a single (one iteration) digital color gain adjustment on the video stream in an attempt to color balance the image.
LucamDigitalWhiteBalanceEx	Performs a single (one iteration) digital color gain adjustment on the video stream in an attempt to color balance the image to a specific target color.
LucamAdjustWhiteBalanceFromSnapshot	Calculates the appropriate color gain values for snapshot mode.
LucamSetTimeout	Updates the timeout value that was originally set for LucamTakeVideo() or the value set in the LUCAM_SNAPSHOT structure while the camera is in Fast Frames mode.
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.
LucamSetup8bitsColorLUT	Populates the 8 bit LUT inside the camera for each color channel.
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.

Function	Description
LucamCreateDisplayWindow	Creates a display window, which is managed by the API, for displaying video.
LucamAdjustDisplayWindow	Allows the user to scale (zoom in/out) the video stream into the preview window.
LucamDestroyDisplayWindow	Destroys the display window created with LucamCreateDisplayWindow.
LucamDisplayPropertyPage	Pops up a DirectShow dialog with the camera properties.
LucamDisplayVideoFormatPage	Pops up a DirectShow 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 Capture, Conversion and Preview of AVI Video Files

Function	Description
LucamConvertRawAVIToStdVideo	Converts a raw AVI video file obtained with LucamStreamVideoControlAVI to a standard video format. (e.g. Standard 24-bit AVI).
LucamPreviewAVIClose	Closes the connection to an AVI file.
LucamPreviewAVIControl	Controls the previewing of a raw AVI video.
LucamPreviewAVIGetDuration	Returns the length of an open AVI file.
LucamPreviewAVIGetFormat	Returns the AVI file information.
LucamPreviewAVIGetFrameCount	Returns the total number of frames within the opened AVI file.
LucamPreviewAVIGetFrameRate	Returns the recorded frame rate of the AVI file.
LucamPreviewAVIGetPositionFrame	Returns the current frame based position within the AVI file.
LucamPreviewAVIGetPositionTime	Returns the current time based position within the AVI file.
LucamPreviewAVISetPositionFrame	Sets the current frame based position within the AVI file.
LucamPreviewAVISetPositionTime	Sets the current time based position within the AVI file.
LucamPreviewAVIOpen	Opens an AVI file for previewing. The

Function	Description
	control of the video playback can be done through the LucamPreviewAVIControl function.
LucamStreamVideoControlAVI	Controls the capture of the video to an AVI file.

2.2.5 Image Capture

Function	Description
LucamGetImageIntensity	Gets the pixel intensity value of a given image.
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.
LucamForceTakeFastFrame	Captures a SW trigger snapshot while the camera is in HW triggered Fast Frames mode.
LucamTakeFastFrameNoTrigger	Retrieves a previously taken single image from the camera, using the camera's still imaging mode.
LucamSetTriggerMode	Toggles between SW triggered and HW triggered snapshots in Fast Frames 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.
LucamTriggerFastFrame	Initiates the request to take a snapshot.

2.2.6 Image Saving

Function	Description
LucamConvertBmp24ToRgb24	Converts a frame of data from the format returned by LucamConvertFrameToRgb24 (BGR) to standard format (RGB).
LucamConvertFrameToGreyscale8	Converts an 8 bit raw frame of data obtained with LucamTakeVideo or LucamTakeSnapshot to a fully processed monochrome frame suitable for display or saving.
LucamConvertFrameToGreyscale8Ex	Converts an 8 bit raw frame of data obtained with LucamTakeVideo or LucamTakeSnapshot to a fully processed monochrome frame suitable for display or saving.
LucamConvertFrameToGreyscale16	Converts a 16 bit raw frame of data obtained with LucamTakeVideo or LucamTakeSnapshot to a fully processed monochrome frame suitable for display or saving.
LucamConvertFrameToGreyscale16Ex	Converts a 16 bit raw frame of data obtained with LucamTakeVideo or LucamTakeSnapshot to a fully processed monochrome frame suitable for display or saving.
LucamConvertFrameToRGB24	Converts a raw frame of data obtained with LucamTakeVideo or LucamTakeSnapshot to a fully processed RGB24 frame suitable for display or saving
LucamConvertFrameToRGB24Ex	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
LucamConvertFrameToRGB32Ex	Converts a raw frame of data obtained with LucamTakeVideo or LucamTakeSnapshot to a fully processed RGB32 frame suitable for display or saving

Function	Description
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).
LucamConvertFrameToRgb48Ex	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).
LucamSaveImage	Saves a single image or video frame to disk in one of several formats.
LucamSaveImageEx	Saves a single image or video frame to disk in one of several formats. This function takes into consideration the camera's Endianness for 16 bit data.
LucamSaveImageW	Exactly like LucamSaveImage but the input filename string is in Unicode (wide character) format.
LucamSaveImageWEx	Exactly like LucamSaveImage but the input filename string is in Unicode (wide character) format. This function takes into consideration the camera's Endianness for 16 bit data.

2.2.7 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 is 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 is processed.
LucamQueryRgbPreviewPixelFormat	Returns the pixel format for the preview window.
LucamRemoveStreamingCallback	Removes the specified video filter callback

Function	Description
	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 is processed.
LucamRemoveRgbPreviewCallback	Removes the specified video filter callback function registered using the function LucamAddRgbPreviewCallback.

2.2.8 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.

2.2.9 Lens Control

Function	Description
LucamAutoFocusStart	Starts the auto focus feature on the camera
LucamAutoFocusWait	Waits for the termination of the auto-focus feature.
LucamAutoFocusStop	Stops the auto-focus feature.
LucamAutoFocusQueryProgress	Queries the progress of the auto-focus feature.
LucamInitAutoLens	Initialize and calibrate the focus and iris positions of the camera lens.
LucamOneShotAutoIris	Performs a single (one iteration) iris adjustment on the video stream in an attempt to reach the auto-exposure target.

3

Application Programming Interface User's Guide

3.1 General Overview

The LuCam API is composed of various functions that control, interrogate, and acquire data from the camera. There are two groups of functions, namely, a basic group and an advanced group.

The functions in the basic group are simple to use and easy to understand. The majority of an application's functionality can be realized quickly using only these functions.

The functions of the advanced group are more powerful and provide greater flexibility and tighter control over the camera's operation; however, they require a more in-depth understanding of the inner workings of the camera. This understanding can be gained from the information in the previous sections, in conjunction with the source code samples provided with the SDK. If you have questions that are not covered in this manual, you can e-mail our TAC group at:

support@lumenera.com

The following section provides guidance for performing specific and common tasks using the LuCam API. Refer to the source code provided with the SDK for specific examples of these tasks.

3.2 Basic Tasks

3.2.1 Connecting and Disconnecting

In order to communicate with a camera you must first open a connection to it and obtain its handle. This handle is used as an input parameter to most of the other API functions. The function used for this task is ***LucamCameraOpen()***. When you are completely finished with the camera, you should terminate the connection using ***LucamCameraClose()***.

Multiple cameras may be attached to the computer at one time. You can obtain the number of cameras attached prior to connecting with any of them using the ***LucamNumCameras()*** function. The ***LucamEnumCameras()*** function can be used to obtain the unique serial numbers and camera type for all cameras attached to the computer. Thus, it is possible to identify and open a specific camera.

3.2.2 Query the Camera

Several functions exist to obtain information about the camera. To obtain the version information for the camera, its driver and the API, you can use the ***LucamQueryVersion()*** function.

To determine the USB connection type (e.g. USB2.0 or USB1.1) you can call ***LucamQueryExternInterface()***.

To determine the available frame rates for the camera you can use the ***LucamEnumAvailableFrameRates()*** function.

3.2.3 Preview Video

Once a camera has been opened, a video preview can be displayed simply by calling ***LucamStreamVideoControl()***. This function takes three parameters;

1. The handle to the camera
2. A control flag (set to START_DISPLAY)
3. A window handle (if NULL a window is automatically created for the video display)

More advanced applications may require their own display window. The handle to this window can be passed to the function and the video will be previewed in it.

Pausing or stopping the preview can be achieved using the same function with the control flag set to PAUSE_STREAM or STOP_STREAMING. When the video stream is stopped, the display window is removed (unless you have created your own display window.)

While video is previewing, the displayed frame rate can be obtained using the function ***LucamQueryDisplayFrameRate()***.

3.2.4 Adjusting the Video

The API allows for the simple adjustment of several of the video properties by providing the ability to pop up one of two pre-defined dialogs. These dialogs are based on the DirectShow libraries. ***LucamDisplayPropertyPage()*** will pop up a dialog for adjusting image properties (exposure, gain, etc.) ***LucamDisplayVideoFormatPage()*** will pop up a dialog for adjusting the video format (resolution, pixel bit depth, etc.)

3.2.5 Configuring Video Format

At any time when the camera is not streaming video, you can configure the video format using the ***LucamSetFormat()*** function. This will allow you to select the following video properties:

1. Subwindow size
2. Subwindow position
3. Subsampling/binning mode
4. Pixel format
5. Video frame rate

There are some constraints and limitations on and associated with the above properties.

1. The subwindow size is provided as the window width and height. Both the width and height must be a multiple of 8.
2. The subwindow position is provided as the x and y coordinates of the top left corner of the subwindow. The y position is constrained to multiples of 8.
3. Subsampling must be the same for both x and y directions.
4. The pixel format is provided as LUCAM_PF_8 or LUCAM_PF_16 and indicates the bit depth of the data. Using LUCAM_PF_16 doubles the amount of data coming from the camera and will cause the maximum frame rate to be cut in half. Although there are 16 bits per pixel in this mode, only 10, 12, or 14 bits of data are valid (depending on the maximum bit depth of the particular camera being used). To determine the Endianness of a camera call ***LucamGetProperty()*** function with the LUCAM_PROP_COLOR_FORMAT property. The flags parameter will state the endianness. If it is equal to LUCAM_PROP_FLAG_LITTLE_ENDIAN then the camera uses Little Endian format. Otherwise, assume that the camera is in Big Endian.
5. The video frame rate provided may not be exactly as requested. The closest lower available rate will be provided.

3.2.6 Grab Video Data

As long as video streaming is turned on (with the ***LucamStreamVideoControl()*** function), video data can be grabbed from the camera. The function to use for this is ***LucamTakeVideo()***. The number of frames must be specified as well as a pointer to a buffer that will accept the data. The data returned from the camera is in its raw form with each byte (for LUCAM_PF_8) or word (for LUCAM_PF_16) returned from the camera representing a single pixel in the image. For color cameras, the data is in the Bayer format as described in a previous section.

It’s not necessary to be previewing the data to capture it. Grabbing video while the preview is turned off is much more efficient and leaves the CPU free to do other processing tasks.

3.2.7 Take a Snapshot (or many)

Single snapshots can be taken using one of several functions. The ***LucamTakeSnapshot()*** function can be used any time a camera is open to asynchronously take a single snapshot using the camera’s snapshot mode. Even if the camera is streaming video, the function will handle the stream, stopping and then restarting it as necessary. The overhead associated with managing the stream means this function is not extremely fast. However, it is a very simple way to obtain snapshots when time is not critical.

For quicker snapshot capturing, individual functions exist to enable the snapshot mode (***LucamEnableFastFrame()***), take snapshots (***LucamTakeFastFrames()***) and then disable the snapshot mode (***LucamDisableFastFrames()***) separately. The use of these functions helps eliminate overhead associated with managing the streaming snapshot mode. When the ***LucamEnableFastFrame()*** function is called, streaming is stopped, and then snapshots can be captured repeatedly thru the ***LucamTakeFastFrames()*** function at a faster rate than with ***LucamTakeSnapshot()***.

As with the ***LucamTakeVideo()*** functions, the data returned from the camera is in its raw, unprocessed form.

3.2.8 Processing Images

For monochrome cameras, the data that arrives from the camera is ready for user processing, display or capture to disk.

Data from color cameras, on the other hand, is in the raw Bayer format and will typically need to be converted to 24-bit RGB before being user processed, displayed or captured to disk. This conversion can be accomplished using the ***LucamConvertFrameToRgb24()*** function.

The two additional functions ***LucamConvertFrameToRgb32()*** and ***LucamConvertFrameToRgb48()*** are available to convert the raw Bayer data into 32 and 48 bit RGB data respectively.

The ***LucamConvertBmp24ToRgb24()*** function will convert the byte order for each pixel from .bmp file standard (or DIB, Device Independent Bitmap) Blue, Green, Red to Red, Green, Blue. Also the image is flipped horizonally so that the first row of data is found at the starting of the buffer.

The color data can also be converted to 8 bit and 16 bit monochrome (Greyscale) using the ***LucamConvertFrameToGreyscale8()*** and ***LucamConvertFrameToGreyscale16()*** respectively.

3.2.9 Save Image to Disk

Saving images to disk can be accomplished with the ***LucamSaveImage()*** or ***LucamSaveImageW()*** functions. Once a frame of data has been taken from the camera (video or snapshot) it can be saved in one of several image formats. (For color cameras, first convert to 24-bit or 48 bit RGB to obtain a proper color image.)

The available formats are:

1. Raw Data (.raw) - no header, no compression
2. Bitmap (.bmp) – Windows bitmap file, no compression
3. TIFF (.tif) – Tagged Image File Format, lossless compression
4. JPEG (.jpg) – JPEG compression, lossy compression

The JPEG and Bitmap formats are not capable of storing the camera output when the camera is set to 16 bit. Only TIFF and Raw data formats support the 16 bit output.
The ***LucamSaveImageEx()*** and ***LucamSaveImageWEx()*** account for the differences in Endianness between the Lu-based and Lw-based cameras when saving 16 bit data to files.

3.2.10 Setting and Getting Camera Properties

Camera properties are items such as exposure, gain, contrast, etc. There are only three functions associated with camera properties. ***LucamSetProperty()*** is used to set the value of a camera property. ***LucamGetProperty()*** is used to get the value of a camera property. ***LucamPropertyRange()*** will return the valid range and default value for a given camera property. It is important to note that not all properties are available for every camera supported by the API. The table below indicates which properties are available for each camera model.

Table 1 – Property Availability By Camera Model

Property	Lu050	Lu070	Lu080	Lu100	Lu110	Lu120	Lu130	Lu160	Lu170
Brightness	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Contrast	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Hue	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Saturation	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Sharpness	No	No	No	No	No	No	No	No	No
Gamma	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Exposure	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Gain	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Red Gain	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Blue Gain	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Green1 Gain	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes
Green2 Gain	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes

Property	Lu200	Lw230	Lu270	Lw290	Lu330	Lu370	Lw560	Lw570	Lw620
Brightness	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Contrast	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Hue	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Saturation	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Sharpness	No	No	No	No	No	No	No	No	No
Gamma	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Exposure	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Gain	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Red Gain	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Blue Gain	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Green1 Gain	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Green2 Gain	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes

3.3 Advanced Tasks

3.3.1 Manage Your Own Video Display Window

When the ***LucamStreamVideoControl()*** is used to start previewing video data, the window is created automatically. Using ***LucamCreateDisplayWindow()*** you can create your own display window and specify its size and location on the screen.

3.3.2 Custom Color Correction Matrix

For color cameras, the data returned from the camera is in the Bayer format. However, this data needs to be color corrected to obtain true color. Normally, one of the standard matrices would be selected from the available ones in the API, but there may be instances where a custom matrix is desired. For example, a color image can be converted to monochrome with the appropriate matrix. Changing the Hue and Saturation can also be performed using a custom matrix. In order to apply a custom matrix, it must first be defined. This is done using the ***LucamSetupCustomMatrix()*** function.

To make use of this custom matrix in video mode, you set the LUCAM_PROP_CORRECTION_MATRIX property to LUCAM_CM_CUSTOM using ***LucamSetProperty()***.

If you want the custom matrix to be used when converting raw images from the camera to RGB24 using ***LucamConvertFrameToRgb24()***, set the CorrectionMatrix field of the LUCAM_CONVERSION structure to LUCAM_CM_CUSTOM.

3.3.3 Custom Look-Up-Tables (LUT)

The camera has a built-in 8 bit LUT, which is used by the Brightness, Contrast and Gamma properties. You can provide your own customized LUT, which will be applied to all data coming from the camera, using the ***LucamSetup8bitsLUT*** function. If you provide your own LUT, the Brightness, Contrast and Gamma properties should not be used; otherwise they will overwrite the custom LUT.

A separate LUT can be applied to individual color channels of a color camera, using the ***LucamSetup8bitsColorLUT()*** function.

3.3.4 Video Callback functions

The Video Callback feature allows you to supply your own function that will be called for every frame of data that arrives from the camera, giving you access to the data stream for frame-by-frame processing. This can be used to provide a graphic overlay, for example, or false color to a monochrome image, etc.

There are two functions that can be used to register your callback. The first is named ***LucamAddStreamingCallback()***. The second is named ***LucamAddRgbPreviewCallback()***. The only difference between them is the data format of the frame of data that is passed to them. The first function will be passed the raw data as it comes out of the camera. The second function will be passed the data after it has been processed into RGB data. (For monochrome cameras, there will be no difference between the data passed for either function.)

The ***LucamQueryRgbPreviewPixelFormat()*** function can be used to determine the bit depth of the RGB data (either 24 bpp or 32 bpp).

The callback function can be removed using either ***LucamRemoveStreamingCallback()*** or ***LucamRemoveRgbPreviewCallback()*** depending on which function was used to add it.

3.3.5 Snapshot Callback Functions

The Snapshot Callback feature allows you to supply your own function that will be called for every snapshot frame that arrives from the camera, giving you access to the snapshot stream for frame-by-frame processing. This can be used to provide a graphic overlay, for example, or false color to a monochrome image etc. The ***LucamAddSnapshotCallback()*** function can be used to register your callback. The callback function can be removed by using the ***LucamRemoveSnapshotCallback()*** function.

3.3.6 Multiple Camera, Simultaneous Image Capture

Several cameras can be connected to the same computer and used to capture an image at the same time. This is accomplished using a set of three API functions. ***LucamEnableSynchronousSnapshots()*** enables this mode for all the cameras. ***LucamTakeSynchronousSnapshots()*** is used to perform the snapshot capture. ***LucamDisableSynchronousSnapshots()*** is used to disable this mode for all the cameras.

3.3.7 Non-Volatile User Accessible Camera Memory

The camera contains a 1024-byte area of non-volatile memory that is user-accessible. The ***LucamPermanentBufferWrite()*** function can be used to write arbitrary data to the memory area that won't get erased when the camera is powered down. The ***LucamPermanentBufferRead()*** function can be used to read the memory area. There is a limit of 100,000 times that the memory area can be overwritten reliably.

3.4 SDK Sample Code Descriptions

Included in the SDK are several sample applications that demonstrate the use of many LuCam API functions. This section describes the sample applications provided.

If you have not purchased the SDK, compiled executables of all the sample applications are provided "as-is" in the Sample Code directory.

3.4.1 AutoLens Sample Application

Application type:	Windows Dialog
Compiler:	Visual C++.Net
Description:	This sample application demonstrates how to control the lenses used with the Lw11059 based cameras. It provides controls for the focus and iris of this lens.

Functions Used:	LucamAddRgbPreviewCallback()
	LucamAdjustDisplayWindow()
	LucamCameraClose()
	LucamCameraOpen()
	LucamConvertFrameToGreyscale8()
	LucamConvertFrameToGreyscale16()
	LucamConvertFrameToRgb24()
	LucamConvertFrameToRgb48()
	LucamDestroyDisplayWindow()
	LucamDigitalWhiteBalance()
	LucamDisableFastFrames()
	LucamEnableFastFrames()
	LucamGetCameraId()
	LucamGetFormat()
	LucamGetLastError()
	LucamGetProperty()

	LucamGetPropertyRange()
	LucamOneShotAutoWhiteBalance()
	LucamQueryVersion()
	LucamRemoveRgbPreviewCallback()
	LucamSaveImage()
	LucamSetProperty()
	LucamStreamVideoControl()

3.4.2 AVISample Sample Application

Application type:	Windows Dialog
Compiler:	Visual C++.Net
Description:	This sample application demonstrates how to use the AVI capture and playback functions.

Functions Used:	LucamAddRgbPreviewCallback()
	LucamAdjustDisplayWindow()
	LucamCameraClose()
	LucamCameraOpen()
	LucamConvertFrameToGreyscale8()
	LucamConvertFrameToGreyscale16()
	LucamConvertFrameToRgb24()
	LucamConvertFrameToRgb48()
	LucamConvertRawAVIToStdVideo()
	LucamDestroyDisplayWindow()
	LucamDigitalWhiteBalance()
	LucamDisableFastFrames()
	LucamEnableFastFrames()
	LucamGetCameraId()
	LucamGetCurrentMatrix()
	LucamGetFormat()
	LucamGetLastError()
	LucamGetProperty()
	LucamGetPropertyRange()
	LucamNumCameras()
	LucamOneShotAutoWhiteBalance()
	LucamPreviewAVIClose()
	LucamPreviewAVIControl()
	LucamPreviewAVIOpen()
	LucamQueryVersion()
	LucamRemoveRgbPreviewCallback()
	LucamSaveImage()
	LucamSetFormat()
	LucamSetProperty()
	LucamSetupCustomMatrix()
	LucamStreamVideoControl()

	LucamStreamVideoControlAVI()
	LucamTakeSnapshot()
	LucamTakeVideo()
	LucamTakeVideoEx()

3.4.3 BlankCamera Sample Application

Application type:	Windows Dialog
Compiler:	Visual C++.Net
Description:	This sample application demonstrates how to use the AVI capture and playback functions.

Functions Used:	LucamAddRgbPreviewCallback()
	LucamAdjustDisplayWindow()
	LucamCameraClose()
	LucamCameraOpen()
	LucamConvertFrameToGreyscale8()
	LucamConvertFrameToGreyscale16()
	LucamConvertFrameToRgb24()
	LucamConvertFrameToRgb48()
	LucamConvertRawAVIToStdVideo()
	LucamDestroyDisplayWindow()
	LucamDigitalWhiteBalance()
	LucamGetCameraId()
	LucamGetCurrentMatrix()
	LucamGetFormat()
	LucamGetLastError()
	LucamGetProperty()
	LucamGetTruePixelDepth()
	LucamNumCameras()
	LucamOneShotAutoWhiteBalance()
	LucamPreviewAVIClose()
	LucamPreviewAVIControl()
	LucamPreviewAVIOpen()
	LucamQueryVersion()
	LucamRegisterEventNotification()
	LucamRemoveRgbPreviewCallback()
	LucamSaveImage()
	LucamSetFormat()
	LucamSetProperty()
	LucamSetupCustomMatrix()
	LucamStreamVideoControl()
	LucamStreamVideoControlAVI()
	LucamTakeSnapshot()
	LucamTakeVideo()

	LucamTakeVideoEx()
	LucamUnregisterEventNotification()

3.4.4 Callback Sample Application

Application type:	Windows Dialog
Compiler:	Visual C++ .Net
Description:	This sample application demonstrates how to create a callback function for both a snapshot callback and a preview callback. The application measures the number of frames captured by the computer for both video frames and snapshot frames. It calculates the capture time of each frame and the average frame rate.

Functions Used:	LucamAddRgbPreviewCallback()
	LucamAddSnapshotCallback()
	LucamCameraClose()
	LucamCameraOpen()
	LucamCreateDisplayWindow()
	LucamDestroyDisplayWindow()
	LucamDisableFastFrames()
	LucamEnableFastFrames()
	LucamGetCameraId()
	LucamGetFormat()
	LucamGetLastError()
	LucamGetProperty()
	LucamQueryVersion()
	LucamRemoveRgbPreviewCallback()
	LucamRemoveSnapshotCallback()
	LucamStreamVideoControl()

3.4.5 CaptureToFile Sample Application

Application type:	Windows Dialog
Compiler:	Visual C++.Net
Description:	This sample application demonstrates how to convert the pixel data into ASCII text and saves this data to either a text file or to an MS Excel® spreadsheet.

Functions Used:	LucamCameraClose()
	LucamCameraOpen()
	LucamConvetFrameToRgb24()
	LucamConvertFrameToRgb48()
	LucamDestroyDisplayWindow()
	LucamGetFormat()
	LucamGetLastError()

	LucamGetProperty()
	LucamSaveImage()
	LucamSetProperty()
	LucamStreamVideoControl()
	LucamTakeVideo()

3.4.6 ClickCrop Sample Application

Application type:	Windows Dialog
Compiler:	Visual C++ .Net
Description:	This sample application demonstrates how to use the callback function to apply an overlay to the video stream. Either a rectangle or elliptical overlay can be selected and placed onto the preview window. The size of the shapes can also be defined. The position can be selected by clicking with the mouse on a location in the preview window. A snapshot can be taken based on the full field of view or just the overlay area.

Functions Used:	LucamAddRgbPreviewCallback()
	LucamCameraClose()
	LucamCameraOpen()
	LucamDestroyDisplayWindow()
	LucamDigitalWhiteBalance()
	LucamGetFormat()
	LucamGetLastError()
	LucamGetProperty()
	LucamOneShotAutoWhiteBalance()
	LucamQueryVersion()
	LucamRemoveRgbPreviewCallback()
	LucamSaveImage()
	LucamSetProperty()
	LucamStreamVideoControl()
	LucamTakeSnapshot()

3.4.7 CSharp Sample Application

Application type:	Windows Dialog
Compiler:	Visual C#.Net
Description:	This sample application demonstrates how to access the LuCamAPICOM object in a C# environment. This application demonstrates how to preview video from the camera, take a snapshot and save it to a file.

Functions Used:	LucamCameraClose()
	LucamCameraOpen()
	LucamConvetFrameToRgb24()

	LucamDestroyDisplayWindow()
	LucamGetFormat()
	LucamGetProperty()
	LucamSaveImage()
	LucamSetFormat()
	LucamSetProperty()
	LucamStreamVideoControl()
	LucamTakeSnapshot()

3.4.8 DirectShow Callback Sample Application

Application type:	Windows Dialog
Compiler:	Visual C++ .Net
Description:	This sample application demonstrates how to setup a callback function using the camera's DirectX interface. The callback function applies a gamma function to the video data through a LUT (Look Up Table).
Functions Used:	NA

3.4.9 DirectX Sample Application

Application type:	Windows Dialog
Compiler:	Visual C++.Net
Description:	This sample application demonstrates how to access the camera through its DirectX interface using Visual C++.Net. It provides controls to start and stop the video stream, preview the video data and control the demosaicing method, control the exposure, gamma, contrast and brightness values. It also demonstrates how to access the permanent buffer storage on the camera.
Functions Used:	NA

3.4.10 DirectX Snapshot Sample Application

Application type:	Windows Dialog
Compiler:	Visual C++ .Net
Description:	This sample application demonstrates how to acquire a snapshot through the DirectX interface. It demonstrates how to change the exposure and gains values, use the strobe output and toggle the trigger input between a SW trigger and HW trigger.
Functions Used:	NA

3.4.11 DualSlope Sample Application

Application type:	Windows Dialog
Compiler:	Visual C++ .Net
Description:	This sample application demonstrates how to use the dual slope feature of the Lu120 and Lw620 cameras.

Functions Used:	LucamCameraClose()
	LucamCameraOpen()
	LucamConvetFrameToRgb24()
	LucamConvetFrameToRgb48()
	LucamDestroyDisplayWindow()
	LucamGetCameraId()
	LucamGetFormat()
	LucamGetLastError()
	LucamGetProperty()
	LucamPropertyRange()
	LucamQueryVersion()
	LucamSaveImage()
	LucamSetFormat()
	LucamSetProperty()
	LucamStreamVideoControl()
	LucamTakeVideo()
	LucamTakeSnapshot()

3.4.12 DX Control Net Sample Application

Application type:	Windows Console
Compiler:	Visual C++.Net
Description:	This sample application is a console based application that uses the DirectX interface of the camera.
Functions Used:	NA

3.4.13 EnumFrameRates Sample Application

Application type:	Windows Console
Compiler:	Visual C++.Net
Description:	This sample application is a console based application that lists the available frames rates for the camera.

Functions Used:	LucamCameraClose()
	LucamCameraOpen()
	LucamEnumAvailableFrameRates()
	LucamGetCameraId()

LucamGetFormat()
LucamQueryVersion()

3.4.14 FastSynchSnaps Sample Application

Application type:	Windows Dialog
Compiler:	Visual C++ 6.0 and Visual C++ .Net
Description:	This sample application demonstrates how to do fast synchronous snapshots from multiple cameras.

Functions Used:	LucamCameraClose()
	LucamCameraOpen()
	LucamConvetFrameToRgb24()
	LucamDisableSynchronousSnapshots()
	LucamEnableSynchronousSnapshots()
	LucamGetLastError()
	LucamQueryVersion()
	LucamTakeSynchronousSnapshots()
	LucamSaveImage()

3.4.15 Flipping Sample Application

Application type:	Windows Dialog
Compiler:	Visual C++ .Net
Description:	This sample demonstrates how to flip and mirror the video preview.

Functions Used:	LucamCameraClose()
	LucamCameraOpen()
	LucamDestroyDisplayWindow()
	LucamGetCameraId()
	LucamGetFormat()
	LucamGetProperty()
	LucamQueryVersion()
	LucamSetFormat()
	LucamSetProperty()
	LucamStreamVideoControl()

3.4.16 FrameRate Sample Application

Application type:	Windows Dialog
Compiler:	Visual C++ .Net
Description:	This sample demonstrates how to read the various available frame rates.

Functions Used:	LucamCameraClose()
	LucamCameraOpen()
	LucamCreateDisplayWindow()
	LucamDestroyDisplayWindow()
	LucamGetCameraId()
	LucamGetFormat()
	LucamSetFormat()
	LucamSetProperty()
	LucamStreamVideoControl()

3.4.17 Get16BitInfo Sample Application

Application type:	Windows Console
Compiler:	Visual C++.Net
Description:	This sample application is a console based application that provides information on the 16 bit mode of the camera such as its bit depth and endianness.

Functions Used:	LucamCameraClose()
	LucamCameraOpen()
	LucamGetCameraId()
	LucamGetFormat()
	LucamGetTruePixelDepth()
	LucamQueryVersion()

3.4.18 GetRanges Sample Application

Application type:	Windows Dialog
Compiler:	Visual C++ .Net
Description:	This sample demonstrates how to read and write the camera properties and get their value ranges.

Functions Used:	LucamCameraClose()
	LucamCameraOpen()
	LucamDestroyDisplayWindow()
	LucamGetProperty()
	LucamPropertyRange()
	LucamSetProperty()
	LucamStreamVideoControl()

3.4.19 GPI Event Signalling Sample Application

Application type:	Windows Dialog
Compiler:	Visual C++ .Net
Description:	This sample application demonstrates how to link an event to the camera’s GPI events.

Functions Used:	LucamCameraClose()
	LucamCameraOpen()
	LucamRegisterEventNotification()
	LucamUnregisterEventNotification()

3.4.20 GpioTest Sample Application

Application type:	Windows Dialog
Compiler:	Visual C++ .Net
Description:	This sample application demonstrates how to read the GPI port of the camera and write to the GPO port.

Functions Used:	LucamCameraClose()
	LucamCameraOpen()
	LucamCreateDisplayWindow()
	LucamGetLastError()
	LucamGpioRead()
	LucamGpioWrite()
	LucamGpoSelect()
	LucamQueryVersion()
	LucamStreamVideoControl()

3.4.21 Histogram Sample Application

Application type:	Windows Dialog
Compiler:	Visual C++ .Net
Description:	This sample application demonstrates how to link an event to the camera’s GPI events.

Functions Used:	LucamAddRgbPreviewCallback()
	LucamAdjustDisplayWindow()
	LucamCameraClose()
	LucamCameraOpen()
	LucamConvertFrameToRgb24()
	LucamConvertFrameToRgb48()
	LucamConvertRawAVIToStdVideo()
	LucamDestroyDisplayWindow()
	LucamDigitalWhiteBalance()
	LucamGetCameraId()
	LucamGetCurrentMatrix()
	LucamGetFormat()
	LucamGetLastError()
	LucamGetProperty()
	LucamGetTruePixelDepth()
	LucamNumCameras()
	LucamOneShotAutoWhiteBalance()

	LucamQueryVersion()
	LucamPreviewAVIClose()
	LucamPreviewAVIControl()
	LucamPreviewAVIOpen()
	LucamRemoveRgbPreviewCallback()
	LucamSaveImage()
	LucamStreamVideoControl()
	LucamStreamVideoControlAVI()
	LucamSetFormat()
	LucamSetProperty()
	LucamSetupCustomMatrix()
	LucamStreamVideoControl()
	LucamTakeVideo()
	LucamTakeVideoEx()

3.4.22 HwTrigCount Sample Application

Application type:	Windows Dialog
Compiler:	Visual C++ .Net
Description:	This sample demonstrates how to configure the camera to use the HW trigger to capture snapshots.

Functions Used:	LucamAddSnapshotCallback()
	LucamCameraClose()
	LucamCameraOpen()
	LucamCreateDisplayWindow()
	LucamDestroyDisplayWindow()
	LucamDisableFastFrames()
	LucamEnableFastFrames()
	LucamGetCameraId()
	LucamGetFormat()
	LucamQueryVersion()
	LucamRemoveSnapshotCallback()
	LucamSetFormat()
	LucamStreamVideoControl()

3.4.23 InfinityTest Sample Application

Application type:	Windows Dialog
Compiler:	Visual C++ .Net
Description:	This sample demonstrates how to capture DeltaVu type snapshots with the InfinityX-21 camera.

Functions Used:	LucamBuildHighResImage()
	LucamCameraClose()
	LucamCameraOpen()

	LucamConvertFrameToRgb24()
	LucamCreateDisplayWindow()
	LucamDestroyDisplayWindow()
	LucamGetCameraId()
	LucamQueryVersion()
	LucamSaveImage()
	LucamSetFormat()
	LucamSetProperty()
	LucamStreamVideoControl()
	LucamTakeVideo()

3.4.24 Lucam Capture Sample Application

Application type:	Windows Dialog
Compiler:	Visual C++ .Net
Description:	This sample is the source code for the LuCam Capture application that is included with the LuCam Software.

Functions Used:	LucamAddStreamingCallback()
	LucamBuildHighResImage()
	LucamCameraClose()
	LucamCameraOpen()
	LucamConvertBmp24ToRgb24()
	LucamConvertFrameToRgb24()
	LucamDestroyDisplayWindow()
	LucamEnumCameras()
	LucamGetCameraId()
	LucamGetFormat()
	LucamGetProperty()
	LucamGetLastError()
	LucamGetProperty()
	LucamNumCameras()
	LucamOneShotWhiteBalance()
	LucamQueryExternalInterface()
	LucamQueryDisplayFrameRate()
	LucamQueryVersion()
	LucamPropertyRange()
	LucamReadRegister()
	LucamRemoveRgbPreviewCallback()
	LucamRemoveStreamingCallback()
	LucamSaveImage()
	LucamSaveImageEx()
	LucamSaveImageW()
	LucamSetFormat()
	LucamSetProperty()
	LucamSetupCustomMatrix()

LucamStreamVideoControl()
LucamTakeSnapshot()
LucamTakeVideo()
LucamWriteRegister()

3.4.25 LucamX Sample Application

Application type:	Windows Dialog
Compiler:	Visual C++ .Net
Description:	This sample application is similar to the LuCam Sample application. This version adds support for the InfinityX-21 camera. This code may not support the same features as the original LuCam Sample application.

Functions Used:	LucamAddStreamingCallback()
	LucamBuildHighResImage()
	LucamCameraClose()
	LucamCameraOpen()
	LucamConvertBmp24ToRgb24()
	LucamConvertFrameToRgb24()
	LucamDestroyDisplayWindow()
	LucamEnumCameras()
	LucamGetCameraId()
	LucamGetFormat()
	LucamGetProperty()
	LucamGetLastError()
	LucamGetProperty()
	LucamNumCameras()
	LucamOneShotWhiteBalance()
	LucamQueryExternalInterface()
	LucamQueryDisplayFrameRate()
	LucamQueryVersion()
	LucamPropertyRange()
	LucamReadRegister()
	LucamRemoveRgbPreviewCallback()
	LucamRemoveStreamingCallback()
	LucamSaveImage()
	LucamSaveImageEx()
	LucamSaveImageW()
	LucamSetFormat()
	LucamSetProperty()
	LucamSetupCustomMatrix()
	LucamStreamVideoControl()
	LucamTakeSnapshot()
	LucamTakeVideo()
	LucamWriteRegister()

3.4.26 MonoCheck Sample Application

Application type:	Windows Dialog
Compiler:	Visual C++ .Net
Description:	Example on how to check for mono or color versions of camera

Functions Used:	LucamCameraClose()
	LucamCameraOpen()
	LucamGetProperty()

3.4.27 MultiSnapshot Sample Application

Application type:	Windows Dialog
Compiler:	Visual C++ .Net
Description:	This sample demonstrates how to take snapshots from different cameras.

Functions Used:	LucamCameraClose()
	LucamCameraOpen()
	LucamConvertFrameToRgb24()
	LucamCreateDisplayWindow()
	LucamDestroyDisplayWindow()
	LucamDisableFastFrames()
	LucamEnableFastFrames()
	LucamGetCameraId()
	LucamGetFormat()
	LucamGetLastError()
	LucamQueryVersion()
	LucamSaveImage()
	LucamSetFormat()
	LucamStreamVideoControl()
	LucamTakeFastFrame()

3.4.28 PermStorage Sample Application

Application type:	Windows Dialog
Compiler:	Visual C++ .Net
Description:	This sample demonstrates how to access and use the permanent storage buffer on the camera.

Functions Used:	LucamCameraClose()
	LucamCameraOpen()
	LucamPermanentBufferRead()
	LucamPermanentBufferWrite()

3.4.29 ResetAndFF Sample Application

Application type:	Windows Console
Compiler:	Visual C++.Net
Description:	This sample application is a console based application that demonstrates how to reset the camera and configure it to perform Fast Frame snapshots.

Functions Used:	LucamCameraClose()
	LucamCameraOpen()
	LucamCameraReset()
	LucamConvertFrameToRgb24()
	LucamConvertFrameToRgb48()
	LucamDisableFastFrames()
	LucamEnableFastFrames()
	LucamGetCameraId()
	LucamGetFormat()
	LucamQueryVersion()
	LucamSaveImage()
	LucamSetProperty(()
	LucamTakeFastFrame()

3.4.30 ScrollingPreview Sample Application

Application type:	Windows Dialog
Compiler:	Visual C++ .Net
Description:	Example on how to create a scrolling preview window

Functions Used:	LucamAdjustDisplayWindow()
	LucamCameraClose()
	LucamCameraOpen()
	LucamGetFormat()
	LucamStreamVideoControl()

3.4.31 Snapshot Sample Application

Application type:	Windows Dialog
Compiler:	Visual C++ .Net
Description:	This example demonstrates how to take snapshots.

Functions Used:	LucamCameraClose()
	LucamCameraOpen()
	LucamConvertFrameToRgb24()
	LucamConvertFrameToRgb48()
	LucamCreateDisplayWindow()
	LucamDestroyDisplayWindow()

	LucamDisableFastFrames()
	LucamEnableFastFrames()
	LucamForceTakeFastFrame()
	LucamGetCameraId()
	LucamGetFormat()
	LucamGetLastError()
	LucamGetProperty()
	LucamGetTruePixelDepth()
	LucamQueryVersion()
	LucamSaveImage()
	LucamSetFormat()
	LucamSetProperty()
	LucamStreamVideoControl()
	LucamTakeFastFrame()
	LucamTakeSnapshot()

3.4.32 Threshold Sample Application

Application type:	Windows Dialog
Compiler:	Visual C++ .Net
Description:	This sample demonstrates how to setup the camera to work in threshold mode. In this mode, the camera will only return pixel data that is higher than the threshold value. The data returned include the pixel intensity and its X and Y coordinates.

Functions Used:	LucamAddStreamingCallback()
	LucamCameraClose()
	LucamCameraOpen()
	LucamCreateDisplayWindow()
	LucamDestroyDisplayWindow()
	LucamGetFormat()
	LucamGetLastError()
	LucamGetProperty()
	LucamSetFormat()
	LucamSetProperty()
	LucamStreamVideoControl()

3.4.33 VB Picture Flip Sample Application

Application type:	Windows Dialog
Compiler:	Visual Basic 6.0
Description:	This sample demonstrates how to flip and mirror the video preview.

Functions Used:	LucamCameraClose()
	LucamCameraOpen()
	LucamGetFormat()
	LucamGetProperty()
	LucamSetFormat()
	LucamSetProperty()
	LucamStreamVideoControl()

3.4.34 VB Sync Snaps Sample Application

Application type:	Windows Dialog
Compiler:	Visual Basic 6.0
Description:	This sample demonstrates how to do synchronous snapshot captures from 2 cameras.

Functions Used:	LucamCameraClose()
	LucamCameraOpen()
	LucamDisableSynchronousSnapshots()
	LucamEnableSynchronousSnapshots()
	LucamGetLastError()
	LucamTakeSynchronousSnapshots()

3.4.35 VBlucamCOMSample Application

Application type:	Windows Dialog
Compiler:	Visual Basic.Net
Description:	This sample application demonstrates how to access the camera features using the LuCamAPICOM COM object.
Functions Used:	NA

3.4.36 VBNet Sample Application

Application type:	Windows Dialog
Compiler:	Visual Basic.Net
Description:	This sample application demonstrates how to access the camera using VB.Net. This application is similar to the VB Sample application described earlier.

Functions Used:	LucamAddRgbPreviewCallback()
	LucamCameraClose()
	LucamCameraOpen()
	LucamConvertFrameToGreyscale8()
	LucamConvertFrameToRgb24()
	LucamDestroyDisplayWindow()
	LucamDisplayPropertyPage()

LucamDisableSynchronousSnapshots()
LucamEnableSynchronousSnapshots()
LucamEnumCameras()
LucamGetFormat()
LucamGetLastError()
LucamGetProperty()
LucamGpioRead()
LucamGpioWrite()
LucamGpoSelect()
LucamOneShotWhiteBalance()
LucamRemoveRgbPreviewCallback()
LucamSaveImage()
LucamSetFormat()
LucamSetPropert()
LucamStreamVideoControl()
LucamTakeSnapshot()
LucamTakeSynchronousSnapshots()
LucamTakeVideo()

4

Camera Support for Third Party Software

This section describes the various third party software interfaces that are supported by the Lumenera USB 2.0 cameras. These software interfaces are more of a programmatic interface and require some knowledge of the LuCam API interface and the LuCam SDK.

4.1 MatLab Camera Plug-In

The MatLab plug-in that was designed for the Lumenera USB 2.0 cameras supports all USB based cameras. The plug-in provides two interfaces:

- An Image Acquisition adapter interface
- A LuCam API Dynamic Link Library (DLL) wrapper interface

Please refer to the readme.txt file that is included with the MatLab Plug-In Software Package for installation instructions for both interfaces. This package is available for download from our website at www.lumenera.com/support/download.php.

Note: The MatLab camera plug-in requires the MathWorks IMAQ Toolbox to use either interface. You can acquire this interface directly from MathWorks' website at www.mathworks.com.

4.1.1 The Image Acquisition Adapter Interface

This interface supports the MatLab Image Acquistion (IMAQ) Toolbox adapter specification. It provides an interface to the camera that can be accessed via the MatLab Image Acquisition Toolbox engine. A video input object is provided by this interface and allows users to access the camera video stream and camera properties using standard IMAQ functions. This interface allows you to quickly port your existing IMAQ based applications to work with the Lumenera family of USB 2.0 cameras.

This is a new interface for Lumenera cameras, and development is on-going. Currently, a subset of the most frequently used API function calls has been implemented, and additional capability will be added in future releases.

4.1.2 LuCam API Wrapper Interface

The LuCam API Wrapper Interface provides several MatLab script files (.m files) that mimic the LuCam API functions described in this manual. These script files redirect the function calls to a Dispatcher DLL which, in turn, calls the LuCam API interface. There is no setup or initialization required to use this interface.

Script files differ in their implementation in that the function parameter list for each API function is in reverse order from the way it is documented in this manual. So if an API function takes Parameter1, Parameter2 and Parameter3 in this order, the MatLab script associated to this function would input the parameters as Parameter3, Parameter2 and Parameter1.

E.g.:

LuCam API function notation:

```
LucamSetProperty(handle, property, value, flag);
```

MatLab script notation:

```
LucamSetProperty (flag, value, property, handle);
```

Not all the LuCam API functions have an associated script file. Please refer to Detailed API Description chapter for more information on the functions that are currently supported.

Furthermore, the Constants and Structures Descriptions chapter contains definitions for the camera properties, values and flags used by the camera. A more complete list and their associated values can be found in the lucamapi.h file that is included with the LuCam SDK.

4.2 LabVIEW Camera Plug-In

The LabVIEW plug-in that was designed for the Lumenera USB 2.0 cameras supports all USB based cameras. The plug-in provides a graphical, icon-based interface that closely resembles the LuCam API interface described in this manual. It is built and designed to work with NI's Vision Builder for Automated Inspection software.

Each of the LuCam API functions has an associated .vi module. The inputs and outputs for each .vi correspond closely with the functions described in this manual.

Not all the LuCam API functions have an associated .vi module . Please refer to Detailed API Description chapter for more information on the functions that are currently supported.

Furthermore, the Constants and Structures Descriptions chapter contains definitions and values for the camera properties, values and flags used by the camera. Many of these values have been defined by the Lumenera Camera plug-in. In the event that some are not defined, a more complete list can be found in the lucamapi.h file that is included with the LuCam SDK.

5

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	[in/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.

The declaration for the preview callback function is as follows:

```
void __stdcall PreviewCallback(VOID *pContext,
          BYTE *pData,
          ULONG dataLength,
          ULONG unused);
```

Where,

pContext	can be a pointer to a global storage area that contains application context information such as a pointer to a controlling object.
pData	is the video frame that was just received
dataLength	is the size of this video frame

This function is not supported currently by the MatLab plug-in.

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 LucamAddSnapshotCallback(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
*pContex	[in] pointer to the context data
*pData	[in/out] snapshot frame data returned from the camera
dataLength	[in] snapshot 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

The declaration for the snapshot callback function is as follows:

```
void __stdcall SnapshotCallback(VOID *pContext,
          BYTE *pData,
          ULONG dataLength);
```

Where,

pContext	can be a pointer to a global storage area that contains application context information such as a pointer to a controlling object.
pData	is the snapshot that was just received
dataLength	is the size of this snapshot frame

This function is not supported currently by the MatLab plug-in.

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	[in/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

The declaration for the streaming callback function is as follows:

```
void __stdcall StreamingCallback(VOID *pContext,
          BYTE *pData,
          ULONG dataLength);
```

Where,

pContext	can be a pointer to a global storage area that contains application context information such as a pointer to a controlling object.
pData	is the video frame that was just received
dataLength	is the size of this video frame

This function is not supported currently by the MatLab plug-in.

LucamAdjustDisplayWindow

Allows the user to scale (zoom in/out) the video stream into the preview window.

Usage

```
LONG LucamAdjustDisplayWindow(HANDLE hCamera,
          LPCTSTR lpTitle,
          int x,
          int y,
          int width,
          int height);
```

Parameters

hCamera	[in] handle to the camera
lpTitle	[in] title of window that appears in window frame
x	[in] x coordinate of pixel in video stream that will appear in upper left corner of display window (default is 0)
y	[in] y coordinate of pixel in video stream that will appear in upper left corner of display window (default is 0)
width	[in] width of scaled video stream in pixels (default is 0)
height	[in] height of scaled video stream in pixels (default is 0)

Return Values

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

Remarks

For example, to zoom in by a factor of two, the width and height would be set to twice the actual width and height of the video stream.
The x and y values are used to pan the display window across the video stream. Negative values for x and y will pan the window down and to the right, respectively.

This function is not supported currently by the MatLab plug-in.

LucamAdjustWhiteBalanceFromSnapshot

Calculates the appropriate color balances for a snapshot image based on the snapshot settings provided.

Usage

```
BOOL LucamAdjustWhiteBalanceFromSnapshot(
          HANDLE hCamera,
          LUCAM_SNAPSHOT *pSettings,
          BYTE *pData,
          float redOverGreen,
          float blueOverGreen,
          ULONG startX,
          ULONG startY,
          ULONG width,
          ULONG height);
```

Parameters

hCamera	[in] handle to the camera
pSettings	[in/out] snapshot settings used to capture original snapshot. New color gains are set in this structure upon completion of this function
pData	[in] snapshot captured with pSettings snapshot settings
redOverGreen	[in] red pixel value of the desired color divided by the green pixel value
blueOverGreen	[in] blue pixel value of the desired color divided by the green pixel value
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 color balance
height	[in] height of window to color balance

Return Values

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

Remarks

To use this function, first take a snapshot image using either LucamTakeSnapshot() or LucamTakeFastFrames(). Take the same LUCAM_SNAPSHOT structured used to acquire the image and pass it along with the snapshot image buffer into this function. This function will calculate the appropriate color gains based on the redOverGreen and blueOverGreen values provided and update the pSettings LUCAM_SNAPSHOT structure provided with these new gain values.

If this function call was performed while the camera is in Fast Frames mode, it will also update the current color gains used for all subsequent snapshots.

This function is not supported currently by the MatLab plug-in.

LucamAutoFocusQueryProgress

Provides the status of the auto focus calibration. The value returned states the progress of the auto focus calibration.

Usage

```
BOOL LucamAutoFocusQueryProgress(HANDLE hCamera,
         FLOAT *pPercentageCompleted);
```

Parameters

hCamera	[in] handle to the camera
pPercentageCompleted	[out] progress value in percent

Return Values

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

Remarks

This function can be called to update a status bar while performing an auto focus step. This function is only available with cameras that can control a motorized lens.

LucamAutoFocusStart

Starts an auto focus calibration.

Usage

```
BOOL LucamAutoFocusStart(HANDLE hCamera,
          ULONG startX,
          ULONG startY,
          ULONG width,
          ULONG height,
          FLOAT putZeroThere1,
          FLOAT putZeroThere2,
          FLOAT putZeroThere3,
          BOOL (__stdcall * ProgressCallback)(void
               *context, FLOAT percentageCompleted),
          void *contextForCallback);
```

Parameters

hCamera	[in] handle to the camera
startX	[in] X position of top left corner of window to auto focus
startY	[in] Y position of top left corner of window to auto focus
width	[in] width of window to auto focus
height	[in] height of window to auto focus
putZeroThere1	[in] reserved value and should be set to 0
putZeroThere2	[in] reserved value and should be set to 0
putZeroThere3	[in] reserved value and should be set to 0
ProgressCallback	[in] optional callback function pointer
contextForCallback	[in] context parameter for callback function

Return Values

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

Remarks

If a callback function is provided, it will be called periodically with the current progress of the auto focus. If a callback function is not used, the current auto focus status can be requested through the LucamAutoFocusQueryProgress() function.

The declaration for the progress callback function is as follows:

```
BOOL (__stdcall * ProgressCallback)(void *context,
          FLOAT percentageCompleted)
```

Where,

pContext	can be a pointer to a global storage area that contains application context information such as a pointer to a controlling object.
percentageCompleted	is the current auto focus progress in percent

LucamAutoFocusStop

Stops the auto focus calibration.

Usage

```
BOOL LucamAutoFocusStop(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

This function stops an auto focus calibration prematurely. By default, the auto focus calibration will continue until proper focus is achieved within the provided region of interest. If proper focus is not achieved within a preset number of iterations, the LuCam API will terminate the auto focus calibration with the closest focus value. This function is not required to terminate the auto focus if the auto focus calibration completed normally.

LucamAutoFocusWait

Waits for the completion of the auto focus calibration.

Usage

```
BOOL LucamAutoFocusWait(HANDLE hCamera, DWORD timeout);
```

Parameters

hCamera	[in] handle to the camera
timeout	[in] timeout value for the auto focus calibration will run before terminating if the proper focus value is not found

Return Values

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

Remarks

This is a blocking function. It returns when either the auto focus calibration is complete or when the timeout is reached. This function does not stop the auto focus calibration process when the timeout is reached. To stop the calibration call the LucamAutoFocusStop() function.

LucamAutoRoiGet

Returns the region of interest used for the auto functions.

Usage

```
BOOL LucamAutoRoiGet(HANDLE hCamera,
          LONG *pStartX,
          LONG *pStartY,
          LONG *pWidth,
          LONG *pHeight);
```

Parameters

hCamera	[in] handle to the camera
*pStartX	[out] the starting X offset of the top left corner of the ROI
*pStartY	[out] the starting Y offset of the top left corner of the ROI
*pWidth	[out] the width of the ROI
*pHeight	[out] the height of the ROI

Return Values

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

Remarks

This function only applies currently to Lw230 based cameras.

LucamAutoRoiSet

Sets a region of interest that will be used by the auto functions.

Usage

```
BOOL LucamAutoRoiSet(HANDLE hCamera,
          LONG startX,
          LONG startY,
          LONG width,
          LONG height);
```

Parameters

hCamera	[in] handle to the camera
startX	[in] the starting X offset of the top left corner of the ROI
startY	[in] the starting Y offset of the top left corner of the ROI
width	[in] the width of the ROI
height	[in] the height of the ROI

Return Values

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

Remarks

The ROI dimensions should not exceed the current window size set in the camera. This function only applies currently to Lw230 based cameras.

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 computer. Valid camera numbers are in the range 1 through the value returned from *LucamNumCameras()*.

LucamCameraReset

Resets the camera to its 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

This function is not supported currently by the MatLab plug-in.

LucamCancelTakeFastFrame

Cancels a call to LucamTakeFastFrame(), LucamForceTakeFastFrame(), LucamTakeFastFrameNoTrigger() or LucamTakeSnapshot() made with another programming thread.

Usage

```
BOOL LucamCancelTakeFastFrame(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

The camera handle provided to this function must be the same one that was used in the associated LucamTakeFastFrame(), LucamForceTakeFastFrame(), LucamTakeFastFrameNoTrigger() or LucamTakeSnapshot() function being cancelled. The cancelled function will return FALSE and the LucamGetLastError() function will return `LucamCancelled`.

This function is not supported currently by the MatLab plug-in.

LucamCancelTakeVideo

Cancels a call to LucamTakeVideo() or LucamTakeVideoEx() made with another programming thread.

Usage

```
BOOL LucamCancelTakeVideo(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

The camera handle provided to this function must be the same one that was used in the associated LucamTakeVideo() or LucamTakeVideoEx() function being cancelled. The cancelled function will return FALSE and the LucamGetLastError() function will return `LucamCancelled`.

This function is not supported currently by the MatLab plug-in.

LucamConvertBmp24ToRgb24

Converts a frame of data from the format returned by LucamConvertFrametoRGB24() (BGR) to standard format (RGB).

Usage

```
void LucamConvertBmp24ToRgb24(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

None.

Remarks

This function is not supported currently by the MatLab plug-in.

LucamConvertFrameToGreyscale8

Converts an 8 bit raw Bayer frame of data obtained with LucamTakeVideo(), LucamTakeFastFrames() or LucamTakeSnapshot() to a fully processed monochrome frame suitable for display or saving.

Usage

```
BOOL LucamConvertFrameToGreyscale8(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 monochrome image data in 8 bit format
*pSrc	[in] image data to be processed from LucamTakeVideo(), LucamTakeFastFrames() 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 Section 6.2.4. The pixel format should be LUCAM_PF_8. The output frame consists of 8 bit pixels of greyscale data.

This function is not supported currently by the MatLab plug-in.

LucamConvertFrameToGreyscale8Ex

Converts an 8 bit raw Bayer frame of data obtained with LucamTakeVideo(), LucamTakeFastFrames() or LucamTakeSnapshot() to a fully processed monochrome frame suitable for display or saving.

Usage

```
BOOL LucamConvertFrameToGreyscale8Ex(HANDLE hCamera,
          BYTE *pDest,
          const BYTE *pSrc,
          LUCAM_IMAGE_FORMAT *pImageFormat,
          LUCAM_CONVERSION_PARAMS *pParams);
```

Parameters

hCamera	[in] handle to the camera
*pDest	[out] processed monochrome image data in 8 bit format
*pSrc	[in] image data to be processed from LucamTakeVideo(), LucamTakeFastFrames() or LucamTakeSnapshot()
pImageFormat	[in] structure containing the image format properties
*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

This function can be used to convert a previously saved raw image.

The LUCAM_IMAGE_FORMAT structure is described in Section 6.2.6. This structure is populated by calling either LucamGetVideoImageFormat() or LucamGetStillImageFormat() after capturing the Bayer image to be converted.

The LUCAM_CONVERSION_PARAMS structure is described in Section 6.2.5. The output frame consists of 8 bit pixels of greyscale data.

This function is not supported currently by the MatLab plug-in.

LucamConvertFrameToGreyscale16

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

Usage

```
BOOL LucamConvertFrameToGreyscale16(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 monochrome image data in 16 bit format
*pSrc	[in] image data to be processed from LucamTakeVideo(), LucamTakeFastFrames() 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 Section 6.2.4. The pixel format should be LUCAM_PF_16. The output frame consists of 16 bit pixels of greyscale data.

This function is not supported currently by the MatLab plug-in.

LucamConvertFrameToGreyscale16Ex

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

Usage

```
BOOL LucamConvertFrameToGreyscale16Ex(HANDLE hCamera,
          USHORT *pDest,
          const USHORT *pSrc,
          LUCAM_IMAGE_FORMAT *pImageFormat,
          LUCAM_CONVERSION_PARAMS *pParams);
```

Parameters

hCamera	[in] handle to the camera
*pDest	[out] processed monochrome image data in 16 bit format
*pSrc	[in] image data to be processed from LucamTakeVideo(), LucamTakeFastFrames() or LucamTakeSnapshot()
pImageFormat	[in] structure containing the image format properties
*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

This function can be used to convert a previously saved raw image.

The LUCAM_IMAGE_FORMAT structure is described in Section 6.2.6. This structure is populated by calling either LucamGetVideoImageFormat() or LucamGetStillImageFormat() after capturing the Bayer image to be converted.

The LUCAM_CONVERSION_PARAMS structure is described in Section 6.2.5. The output frame consists of 16 bit pixels of greyscale data.

This function is not supported currently by the MatLab plug-in.

LucamConvertFrameToRGB24

Converts a raw frame of data obtained with LucamTakeVideo(), LucamTakeFastFrames() 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(), LucamTakeFastFrames() 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 Section 6.2.4. The available pixel formats are listed in Section 6.1.3. The RGB24 data format has 24 bits per pixel. The three bytes of each pixel are the Blue, Green, Red values in that order.

This function is not supported currently by the MatLab plug-in.

LucamConvertFrameToRGB24Ex

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

Usage

```
BOOL LucamConvertFrameToRgb24Ex(HANDLE hCamera,
          BYTE *pDest,
          const BYTE *pSrc,
          const LUCAM_IMAGE_FORMAT *pImageFormat,
          const LUCAM_CONVERSION_PARAMS *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(), LucamTakeFastFrames() or LucamTakeSnapshot()
pImageFormat	[in] structure containing the image format properties
*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

This function can be used to convert a previously saved raw image.

The LUCAM_IMAGE_FORMAT structure is described in Section 6.2.6. This structure is populated by calling either LucamGetVideoImageFormat() or LucamGetStillImageFormat() after capturing the Bayer image to be converted.

The LUCAM_CONVERSION_PARAMS structure is described in Section 6.2.5. The RGB24 data format has 24 bits per pixel. The three bytes of each pixel are the Blue, Green, Red values in that order.

This function is not supported currently by the MatLab plug-in.

LucamConvertFrameToRGB32

Converts a raw frame of data obtained with LucamTakeVideo(), LucamTakeFastFrames() 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(), LucamTakeFastFrames() 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 Section 6.2.4. The available pixel formats are listed in Section 6.1.3. 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 can contain the assigned Alpha channel value.

This function is not supported currently by the MatLab plug-in.

LucamConvertFrameToRGB32Ex

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

Usage

```
BOOL LucamConvertFrameToRgb32Ex(HANDLE hCamera,
          BYTE *pDest,
          const BYTE *pSrc,
          const LUCAM_IMAGE_FORMAT *pImageFormat,
          const LUCAM_CONVERSION_PARAMS *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(), LucamTakeFastFrames() or LucamTakeSnapshot()
pImageFormat	[in] structure containing the image format properties
*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

This function can be used to convert a previously saved raw image.

The LUCAM_IMAGE_FORMAT structure is described in Section 6.2.6. This structure is populated by calling either LucamGetVideoImageFormat() or LucamGetStillImageFormat() after capturing the Bayer image to be converted.

The LUCAM_CONVERSION_PARAMS structure is described in Section 6.2.5. 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.

This function is not supported currently by the MatLab plug-in.

LucamConvertFrameToRGB48

Converts a raw frame of data obtained with LucamTakeVideo(), LucamTakeFastFrames() 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 RGB48 format
*pSrc	[in] image data to be processed from LucamTakeVideo(), LucamTakeFastFrames() 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 Section 6.2.4. The pixel format should be LUCAM_PF_16. The RGB48 data format has 48 bits per pixel. The three words (2 bytes) of each pixel are the Blue, Green, Red values in that order.

This function is not supported currently by the MatLab plug-in.

LucamConvertFrameToRGB48Ex

Converts a raw frame of data obtained with LucamTakeVideo(), LucamTakeFastFrames() 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 LucamConvertFrameToRgb48Ex(HANDLE hCamera,
          USHORT *pDest,
          const USHORT *pSrc,
          const LUCAM_IMAGE_FORMAT *pImageFormat,
          const LUCAM_CONVERSION_PARAMS *pParams);
```

Parameters

hCamera	[in] handle to the camera
*pDest	[out] processed image data in RGB48 format
*pSrc	[in] image data to be processed from LucamTakeVideo(), LucamTakeFastFrames() or LucamTakeSnapshot()
pImageFormat	[in] structure containing the image format properties
*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

This function can be used to convert a previously saved raw image.

The LUCAM_IMAGE_FORMAT structure is described in Section 6.2.6. This structure is populated by calling either LucamGetVideoImageFormat() or LucamGetStillImageFormat() after capturing the Bayer image to be converted.

The LUCAM_CONVERSION_PARAMS structure is described in Section 6.2.5. The RGB48 data format has 48 bits per pixel. The three words (2 bytes) of each pixel are the Blue, Green, Red values in that order.

This function is not supported currently by the MatLab plug-in.

LucamConvertRawAVIToStdVideo

Converts a raw AVI video (8 bit) obtained with LucamStreamVideoControlAVI() to a fully processed standard video format. (e.g. Standard 24-bit AVI).

Usage

```
BOOL LucamConvertRawAVIToStdVideo(HANDLE hCamera,
          WCHAR *pOutputFileName,
          WCHAR *pInputFileName,
          ULONG outputType);
```

Parameters

hCamera	[in] handle to the camera
* pOutputFileName	[out] processed video file name
* pInputFileName	[in] raw AVI video file name to be processed
outputType	[in] format of the video output

Return Values

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

Remarks

The outputType must be one of AVI_STANDARD_24 or AVI_STANDARD_32. The input file name should be different than the output file name. Note that the output file could be 3 to 4 times larger than the original raw AVI file.

This function is not supported currently by the MatLab plug-in.

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 is not automatically resized to the video frame size whenever the video frame size is changed. You must destroy the window and then recreate it.

This function is not supported currently by the MatLab plug-in.

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

This function is not supported currently by the MatLab plug-in.

LucamDigitalWhiteBalance

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

Usage

```
BOOL LucamDigitalWhiteBalance(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

The width and height are the width and height of the video stream after any sub-sampling or binning.
The on-chip analog gain values are not changed.

LucamDigitalWhiteBalanceEx

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

Usage

```
BOOL LucamDigitalWhiteBalanceEx(HANDLE hCamera,
          FLOAT redOverGreen,
          FLOAT blueOverGreen,
          ULONG startX,
          ULONG startY,
          ULONG width,
          ULONG height);
```

Parameters

hCamera	[in] handle to the camera
redOverGreen	[in] red pixel value of the desired color divided by the green pixel value
blueOverGreen	[in] blue pixel value of the desired color divided by the green pixel value
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

The width and height are the width and height of the video stream after any sub-sampling or binning.
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.
The on-chip analog gain values are not changed.

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.

This function is not supported currently by the MatLab plug-in.

LucamDisableSynchronousSnapshots

Disables the simultaneous snapshot capture mode.

Usage

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

Parameters

syncSnapsHandle	[in] handle returned from LucamEnable*SynchronousSnaphots()* 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

This function is not supported currently by the MatLab plug-in.

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 Section 6.2.1. 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.

This function is not supported currently by the MatLab plug-in.

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 the number of available frame rates on the camera.
If the function fails, the return value is 0.

Remarks

Frame rates are in frames per second. Call this function with entryCount equal to 0. The function will return the number of available frame rates. Allocate the necessary memory to store all the frame rate values and call the function again with entryCount equal to the value returned in the last call and pAvailableFrameRates pointing to the allocated memory.

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

A call to LucamNumCameras() should be called prior to calling this function and allocating memory for pVersionsArray to know how many cameras are connected to the computer.

LucamForceTakeFastFrame

Forces a SW triggered snapshot while in HW triggered Fast Frames mode.

Usage

```
BOOL LucamForceTakeFastFrame(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

This function forces a snapshot capture when the camera is set in Fast Frames mode with the HW trigger enabled. This function will return a snapshot frame without waiting for the next HW trigger.

This function is not supported currently by the MatLab plug-in.

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 (Discontinued)	0x091
Lu050C, Lu055C (Discontinued)	0x095
Lu056C (Discontinued)	0x093
Lu070M, Lu075M, Lu070C, Lu075C	0x08C
Lw070M, Lw075M, Lw070C, Lw075C	0x18C
Lm075M, Lm075C	0x28C
Lu080M, Lu085M, Lu080C, Lu085C	0x085
Lu100M, Lu105M, Lu100C, Lu105C	0x092
Lu110M, Lu115M, Lu110C, Lu115C (Discontinued)	0x094
Lu120M, Lu125M, Lu120C, Lu125C	0x096
Lu130M, Lu135M, Lu130C, Lu135C	0x09A
Lw130M, Lw135M, Lw130C, Lw135C	0x19A
Lm135M, Lm135C	0x29A
Lu160M, Lu165M, Lu160C, Lu165C	0x08A
Lw160M, Lw165M, Lw160C, Lw165C	0x18A
Lm165M, Lm165C	0x28A
Lu170M, Lu175M, Lu170C, Lu175C	0x09E
Lu176C	0x082
Lu200C, Lu205C	0x097
Lw230M, Lw235M, Lw230C, Lw235C	0x180
Lu270C, Lu275C	0x08D

Camera Model	ID
Lw290C, Lw295C	0x1CD
Lu330C, Lu335C	0x09B
Lw330C, Lw335C	0x19B
Lu370C, Lu375C	0x08B
Lw570C, Lw575	0x1C5
Lw620M, Lw625M, Lw620C, Lw625C	0x186
Lw11050C, Lw11056C, Lw11057C, Lw11058C, Lw11059C	0x1C8
InfinityX-21	0x0A0
Infinity1-1, Infinity 1	0x0A1
Infinity1-3, Infinity 3	0x0A3
Infinity1-5	0x1Ac
Infinity1-6	0x1A6
Infinity 2	0x0A2

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.

This function is not supported currently by the MatLab plug-in.

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 Section 6.2.2. This function can be called immediately after *LucamOpenCamera()* to get the default values for the video format parameters.

LucamGetImageIntensity

Returns the pixel intensity value of a given image.

Usage

```
BOOL LucamGetImageIntensity(HANDLE hCamera,
          BYTE *pFrame,
          LUCAM_IMAGE_FORMAT *pImageFormat,
          ULONG startX,
          ULONG startY,
          ULONG width,
          ULONG height,
          FLOAT *pIntensity,
          FLOAT *pRedIntensity,
          FLOAT *pGreen1Intensity,
          FLOAT *pGreen2Intensity,
          FLOAT *pBlueIntensity);
```

Parameters

hCamera	[in] handle to the camera
*pFrame	[in] frame to be analysed
*pImageFormat	[in] image format of the frame
startX	[in] X position of top left corner of region of interest (ROI) for analysis
startY	[in] Y position of top left corner of ROI for analysis
width	[in] width of ROI for analysis
height	[in] height of ROI for analysis
*pIntensity	[out] average global intensity of all pixels in the frame
*pRedIntensity	[out] average global intensity of all red pixels in the frame
*pGreen1Intensity	[out] average global intensity of all green1 (red-green) pixels in the frame
*pGreen2Intensity	[out] average global intensity of all green2 (blue-green) pixels in the frame
*pBlueIntensity	[out] average global intensity of all blue pixels in the frame

Return Values

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

Remarks

If both pFrame and pImageFormat are NULL, the function will wait for the next video frame and analyse it based on the specified ROI.

If pFrame is not NULL and pImageFormat is NULL, the function immediatly computes the intensities using the current video format settings.
If both pFrame and pImageFormat are not NULL, the function computes the intensities using the format settings contained it the *pImageFormat structure.
For monochrome cameras, individual colour intensities will return the same value as the global intensity, pIntensity.

This function is not supported currently by the MatLab plug-in.

LucamGetLastError

Returns 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 lucamerr.h file.

This function is not supported currently by the MatLab plug-in.

LucamGetLastErrorForCamera

Returns the specific error code for the last error that occurred when calling an API function for a given camera.

Usage

```
ULONG LucamGetLastErrorForCamera(HANDLE hCamera);
```

Parameters

hCamera	[in] handle to the camera

Return Values

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

Remarks

Error codes can be found in the lucamerr.h file. Error codes that are not caused by the camera, such as converting a frame, do not update this error code, but will update the error code returned by LucamGetLastError().

This function is not supported currently by the MatLab plug-in.

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 Section 6.1.1. 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 Section 6.1.2.

LucamGetStillImageFormat

Returns the snapshot image format used to capture a snapshot.

Usage

```
BOOL LucamGetStillImageFormat(HANDLE hCamera,
          LUCAM_IMAGE_FORMAT *pImageFormat);
```

Parameters

hCamera	[in] handle to the camera
pImageFormat	[out] structure containing the image format properties

Return Values

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

Remarks

The LUCAM_IMAGE_FORMAT structure is described in Section 6.2.6.

This function returns the image format properties needed to convert a raw frame to either color or greyscale. The image format information can be saved with the raw image so that it can be converted at a later date using any of the LucamConvertFrame***Ex() based functions.

This function is not supported currently by the MatLab plug-in.

LucamGetTruePixelDepth

Gets the actual pixel depth when running the camera in 16 bit mode.

Usage

```
BOOL LucamGetTruePixelDepth(HANDLE hCamera,
          ULONG *pCount);
```

Parameters

hCamera	[in] handle to the camera
*pCount	[out] pixel depth of the 16 bit data provided by the camera

Return Values

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

Remarks

None.

LucamGetVideoImageFormat

Returns the video image format used to capture a video frame.

Usage

```
BOOL LucamGetVideoImageFormat(HANDLE hCamera,
          LUCAM_IMAGE_FORMAT *pImageFormat);
```

Parameters

hCamera	[in] handle to the camera
pImageFormat	[out] structure containing the image format properties

Return Values

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

Remarks

The LUCAM_IMAGE_FORMAT structure is described in Section 6.2.6.

This function returns the image format properties needed to convert a raw frame to either color or greyscale. The image format information can be saved with the raw image so that it can be converted at a later date using any of the LucamConvertFrame***Ex() based functions.

This function is not supported currently by the MatLab plug-in.

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.

LucamGpioConfigure

Configures the direction of a bi-directional GPIO pin.

Usage

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

Parameters

hCamera	[in] handle to the camera
gpoEnable	[in] bit flags used to disable/enable the output on a GPIO

Return Values

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

Remarks

This function is currently only supported on Lm-based cameras.

Setting the appropriate bit to 1 configures a GPIO pin as an output.
- Bit 0: configures GPO1 direction
- Bit 1: configures GPO2 direction
- Bit 2: configures GPO3 direction
- Bit 3: configures GPO4 direction

Setting the bit to 0 puts the GPIO pin into its default input mode.

LucamGpoSelect

Enables and disables the alternate GPO functionality.

Usage

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

Parameters

hCamera	[in] handle to the camera
gpoEnable	[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 for manual toggling
Bit 1: enables GPO2 for manual toggling
Bit 2: enables GPO3 for manual toggling
Bit 3: enables GPO4 for manual toggling

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

Typical default GPO functionality is:

GPO1: Strobe output ACTIVE LOW (Snapshot mode only)
GPO2: Strobe output ACTIVE HIGH (Snapshot mode only)
GPO3: N/A
GPO4: SOF (Start of Frame) signal (Video mode only)

LucamInitAutoLens

Initialize and calibrate the focus and iris positions of the camera lens.

Usage

```
BOOL LucamInitAutoLens(HANDLE hCamera,
          BOOL force);
```

Parameters

hCamera	[in] handle to the camera
force	[in] force a recalibration of the lens parameters

Return Values

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

Remarks

A call to this function is required to initialize and calibrate the focus and iris properties of the camera.

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

```
BOOL 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

The declaration for the porgress callback function is as follows:

```
void __stdcall SnapshotCallback(VOID *pContext,
          FLOAT percentageCompleted);
```

Where,

pContext	can be a pointer to a global storage area that contains application context information such as a pointer to a controlling object.
percentageCompleted	is a value that describes the current progress as a percentage

LucamOneShotAutoIris

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

Usage

```
BOOL LucamOneShotAutoIris(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) on-chip analog gain adjustment on the video stream 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

The width and height are the width and height of the video stream after any sub-sampling or binning.

LucamOneShotAutoWhiteBalanceEx

Performs a single (one iteration) on-chip analog gain adjustment on the video stream 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
redOverGreen	[in] red pixel value of the desired color divided by the green pixel value
blueOverGreen	[in] blue pixel value of the desired color divided by the green pixel value
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

The width and height are the width and height of the video stream after any sub-sampling or binning.
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.

LucamPerformDualTapCorrection

Performs an additional correction on a captured image from cameras that have more than one sensor readout taps.

Usage

```
BOOL LucamPerformDualTapCorrection(HANDLE hCamera,
          BYTE *pFrame,
          const LUCAM_IMAGE_FORMAT *pImageFormat);
```

Parameters

hCamera	[in] handle to the camera
*pFrame	[in/out] pointer to dual tap raw frame
*pImageFormat	[in] pointer to image format

Return Values

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

Remarks

Currently, only the Lw1105x based cameras support this function.

This function is not supported currently by the MatLab plug-in.

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 2048 bytes long.

This function is not supported currently by the MatLab plug-in.

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 2048 bytes long. This area is limited to 100,000 writes.

This function is not supported currently by the MatLab plug-in.

LucamPreviewAVIClose

Closes the controller to an AVI file.

Usage

```
BOOL LucamPreviewAVIClose(HANDLE hAVI);
```

Parameters

hAVI	[in] handle of the AVI controller

Return Values

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

Remarks

This function is not supported currently by the MatLab plug-in.

LucamPreviewAVIControl

Controls the previewing of an AVI video.

Usage

```
BOOL LucamPreviewAVIControl(HANDLE hAVI,
          ULONG previewControlType,
          HWND previewWindow);
```

Parameters

hAVI	[in] handle of the AVI controller
previewControlType	[in] control type parameter
previewWindow	[in] handle to the window to preview 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_AVI, START_AVI and PAUSE_AVI.
START_AVI will start the video preview in the specified window. This can be the window created with *LucamCreateDisplayWindow()* or the application's own window.

This function is not supported currently by the MatLab plug-in.

LucamPreviewAVIGetDuration

Returns the length of an open AVI file.

Usage

```
BOOL LucamPreviewAVIGetDuration(HANDLE hAVI,
          LONGLONG *pDurationMinutes,
          LONGLONG *pDurationSeconds,
          LONGLONG *pDurationMilliseconds,
          LONGLONG *pDurationMicroSeconds);
```

Parameters

hAVI	[in] handle of the AVI controller
pDurationMinutes	[out] minute portion of AVI duration
pDurationSeconds	[out] second portion of AVI duration
pDurationMilliseconds	[out] millisecond portion of AVI duration
pDurationMicroSeconds	[out] microsecond portion of AVI duration

Return Values

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

Remarks

This function is not supported currently by the MatLab plug-in.

LucamPreviewAVIGetFormat

Returns the AVI file information.

Usage

```
BOOL LucamPreviewAVIGetFormat(HANDLE hAVI,
          LONG *width,
          LONG *height,
          LONG *fileType,
          LONG *bitDepth);
```

Parameters

hAVI	[in] handle of the AVI controller
width	[out] width of the video AVI file
height	[out] height of the video AVI file
fileType	[out] file type of the video AVI file
bitDepth	[out] bit depth of the video AVI file

Return Values

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

Remarks

Valid file types that can be played by the LuCam API include AVI_RAW and AVI_STANDARD24.

This function is not supported currently by the MatLab plug-in.

LucamPreviewAVIGetFrameCount

Returns the total number of frames within the opened AVI file.

Usage

```
BOOL LucamPreviewAVIGetFrameCount(HANDLE hAVI,
          LONGLONG *pFrameCount);
```

Parameters

hAVI	[in] handle of the AVI controller
pFrameCount	[out] number of frames in AVI

Return Values

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

Remarks

This function is not supported currently by the MatLab plug-in.

LucamPreviewAVIGetFrameRate

Returns the recorded frame rate of the AVI file.

Usage

```
BOOL LucamPreviewAVIGetFrameRate(HANDLE hAVI,
          FLOAT *pFrameRate);
```

Parameters

hAVI	[in] handle of the AVI controller
pFrameRate	[out] frame rate of AVI

Return Values

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

Remarks

This function is not supported currently by the MatLab plug-in.

LucamPreviewAVIGetPositionFrame

Returns the current frame based position within the AVI file.

Usage

```
BOOL LucamPreviewAVIGetPositionFrame(HANDLE hAVI,
          LONGLONG *pPositionCurrentFrame);
```

Parameters

hAVI	[in] handle of the AVI controller
pPositionCurrentFrame	[out] frame number of current position

Return Values

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

Remarks

This function is not supported currently by the MatLab plug-in.

LucamPreviewAVIGetPositionTime

Returns the current time based position within the AVI file.

Usage

```
BOOL LucamPreviewAVIGetPositionTime(HANDLE hAVI,
          LONGLONG *pPositionMinutes,
          LONGLONG *pPositionSeconds,
          LONGLONG *pPositionMilliSeconds,
          LONGLONG *pPositionMicroSeconds);
```

Parameters

hAVI	[in] handle of the AVI controller
pDurationMinutes	[out] minute portion of current position
pDurationSeconds	[out] second portion of current position
pDurationMilliseconds	[out] millisecond portion of current position
pDurationMicroSeconds	[out] microsecond portion of current position

Return Values

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

Remarks

This function is not supported currently by the MatLab plug-in.

LucamPreviewAVISetPositionFrame

Sets the current frame based position within the AVI file.

Usage

```
BOOL LucamPreviewAVISetPositionFrame(HANDLE hAVI,
          LONGLONG pPositionFrame);
```

Parameters

hAVI	[in] handle of the AVI controller
pPositionCurrentFrame	[in] frame number of current position

Return Values

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

Remarks

This function is not supported currently by the MatLab plug-in.

LucamPreviewAVISetPositionTime

Sets the current time based position within the AVI file.

Usage

```
BOOL LucamPreviewAVISetPositionTime(HANDLE hAVI,
          LONGLONG pPositionMinutes,
          LONGLONG pPositionSeconds,
          LONGLONG pPositionMilliSeconds,
          LONGLONG pPositionMicroSeconds);
```

Parameters

hAVI	[in] handle of the AVI controller
pDurationMinutes	[in] minute portion of current position
pDurationSeconds	[in] second portion of current position
pDurationMilliseconds	[in] millisecond portion of current position
pDurationMicroSeconds	[in] microsecond portion of current position

Return Values

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

Remarks

This function is not supported currently by the MatLab plug-in.

LucamPreviewAVIOpen

Opens an AVI file for previewing (including an 8 bit raw AVI file). The control of the video is handled with the LucamPreviewAVIControl() function.

Usage

```
HANDLE LucamPreviewAVIOpen(WCHAR *pFileName);
```

Parameters

pFileName	[in] raw AVI file name to be previewed

Return Values

The function returns the HANDLE to the AVI controller used for previewing.
If the function fails, the return value is NULL.

Remarks

This function is not supported currently by the MatLab plug-in.

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 Section 6.1.1. 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 Section 6.1.2.

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

This function is not supported currently by the MatLab plug-in.

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 Section 6.1.7.

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 Section 6.1.3

This function is not supported currently by the MatLab plug-in.

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 Section 6.2.3.

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

This function is not supported currently by the MatLab plug-in.

LucamRegisterEventNotification

Registers an event handle with the LuCam API.

Usage

```
PVOID LucamRegisterEventNotification(HANDLE hCamera,
          DWORD eventId,
          HANDLE hEvent);
```

Parameters

hCamera	[in] handle to the camera
eventId	[in] type of event
hEvent	[in] handle to an event

Return Values

If the function succeeds, the return value is a non-NULL handle to a LuCam API event.
If the function fails, the return value is NULL.

Remarks

An event should be created before calling this function and its handle should be provided through the hEvent parameter. See Section 6.1.13 for more information on the types of event notifications that are available.

This function is not supported currently by the MatLab plug-in.

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 LucamAddRgbPreviewCallback()

Return Values

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

Remarks

This function is not supported currently by the MatLab plug-in.

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

This function is not supported currently by the MatLab plug-in.

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

This function is not supported currently by the MatLab plug-in.

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] 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), Joint Photograhic Experts Group (.jpg), Tagged Image File Format (.tif) and Raw (.raw).
The available pixel formats are listed in Section 6.1.3.
If an unsupported file type (indicated by its extension) is provided, the function will fail.

This function is not supported currently by the MatLab plug-in.

LucamSaveImageEx

Saves a single image or video frame to disk in one of several formats. This function will take into consideration the format of the camera output (big-endian, little-endian) when using 16 bit data.

Usage

```
BOOL LucamSaveImageEx(HANDLE hCamera,
          ULONG width,
          ULONG height,
          ULONG pixelFormat,
          BYTE *pData,
          CHAR *pFilename);
```

Parameters

hCamera	[in] handle to the camera
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] 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 this function will use to save the data.
Supported image formats are Windows bitmap (.bmp), Joint Photograhic Experts Group (.jpg), Tagged Image File Format (.tif) and Raw (.raw).
The available pixel formats are listed in Section 6.1.3.
If an unsupported file type (indicated by its extension) is provided, the function will fail.

This function is not supported currently by the MatLab plug-in.

LucamSaveImageW

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

Usage

```
BOOL LucamSaveImageW(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] 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), Joint Photograhic Experts Group (.jpg), Tagged Image File Format (.tif) and Raw (.raw).
The available pixel formats are listed in Section 6.1.3.
If an unsupported file type (indicated by its extension) is provided, the function will fail.

This function is not supported currently by the MatLab plug-in.

LucamSaveImageWEx

Saves a single image or video frame to disk in one of several formats. This function will take into consideration the format of the camera output (big-endian, little-endian) when using 16 bit data.

Usage

```
BOOL LucamSaveImageEx(HANLDE hCamera,
          ULONG width,
          ULONG height,
          ULONG pixelFormat,
          BYTE *pData,
          WCHAR *pFilename);
```

Parameters

hCamera	[in] handle to the camera
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] 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), Joint Photograhic Experts Group (.jpg), Tagged Image File Format (.tif) and Raw (.raw).
The available pixel formats are listed in Section 6.1.3.
If an unsupported file type (indicated by its extension) is provided, the function will fail.

This function is not supported currently by the MatLab plug-in.

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 Section 6.2.2.
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 Section 6.1.1. 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 Section 6.1.2. If a capability flag is not supported by the property, it is silently ignored.

LucamSetTimeout

Updates the timeout value that was originally set for LucamTakeVideo() or the value set in the LUCAM_SNAPSHOT structure while the camera is in Fast Frames mode.

Usage

```
BOOL LucamSetTimeout(HANDLE hCamera,
          BOOL still,
          FLOAT timeout);
```

Parameters

hCamera	[in] handle to the camera
still	[in] mode to apply new value to
timeout	[in] timeout value

Return Values

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

Remarks

To update the video frame capture timeout value used in LucamTakeVideo() function, set the still parameter to FALSE. Setting the still parameter to TRUE will affect the snapshot mode time out value.

This function is not supported currently by the MatLab plug-in.

LucamSetTriggerMode

Sets the trigger mode used for snapshots while in Fast Frames mode.

Usage

```
BOOL LucamSetTriggerMode(HANDLE hCamera,
          BOOL useHwTrigger);
```

Parameters

hCamera	[in] handle to the camera
useHwTrigger	[in] trigger mode to use

Return Values

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

Remarks

This function allows the toggling of the trigger (either HW or SW trigger) input used to capture snapshots while in Fast Frames mode. Set the useHwTrigger to set the camera to use the HW trigger input to capture snapshots.

This function is not supported currently by the MatLab plug-in.

LucamSetup8bitsColorLUT

Populates the 8 bit Color LUT inside the camera. The LUT provided is for only one color channel at a time. You can use the same LUT for 1 or many color channels by setting the appropriate parameters.

Usage

```
BOOL LucamSetup8bitsColorLUT(HANDLE hCamera,
          UCHAR *pLut,
          ULONG length,
          BOOL applyOnRed,
          BOOL applyOnGreen1,
          BOOL applyOnGreen2,
          BOOL applyOnBlue);
```

Parameters

hCamera	[in] handle to the camera
*pLut	[in] pointer to LUT values
length	[in] number of LUT values
applyOnRed	[in] apply LUT on red channel
applyOnGreen1	[in] apply LUT on green1 channel
applyOnGreen2	[in] apply LUT on green2 channel
applyOnBlue	[in] apply LUT on blue channel

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 to enable it.

This function is not supported currently by the MatLab plug-in.

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 to enable it.

This function is not supported currently by the MatLab plug-in.

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.

This function is not supported currently by the MatLab plug-in.

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.

This function is not supported currently by the MatLab plug-in. There are two temporary MatLab scripts used for previewing video, LucamShowPreview.m and LucamHidePreview.m. These scripts start and stop a preview from a given camera, respectively.

LucamStreamVideoControlAVI

Controls the capture of the video in a raw 8 bit AVI file.

Usage

```
BOOL LucamStreamVideoControlAVI(HANDLE hCamera,
          ULONG controlType,
          LPCWSTR pFileName,
          HWND hWnd);
```

Parameters

hCamera	[in] handle to the camera
controlType	[in] control type parameter
pFileName	[in] file name where to put the AVI
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 starts the capture of the AVI video and displays it in the specified window. This can be the window created with *LucamCreateDisplayWindow()* or the user's own window.
START_STREAMING captures the video without displaying it. Using this alternative gives an AVI video file with higher quality and frame rate.

This function is not supported currently by the MatLab plug-in.

LucamTakeFastFrame

Takes a single image from the camera, using the camera's still imaging 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

To use this function, the camera should be in Fast Frames mode using the LucamEnableFastFrames() function.

This function is not supported currently by the MatLab plug-in.

LucamTakeFastFrameNoTrigger

Retrieves a previously taken single image from the camera, using the camera's still imaging mode.

Usage

```
BOOL LucamTakeFastFrameNoTrigger(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

To use this function, the camera should be in Fast Frames mode using the LucamEnableFastFrames() function. If the camera was set to use a HW trigger to initiate a snapshot, this function can retrieve a previously captured image from the API without sending an new snapshot request and waiting for the next snapshot.

This function is not supported currently by the MatLab plug-in.

LucamTakeSnapshot

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

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 Section 6.2.1. 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

The camera's video stream should be started with a call to LucamStreamVideoControl() before calling this function.

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

This function is not supported currently by the MatLab plug-in.

LucamTriggerFastFrame

Initiates the request to take a snapshot.

Usage

```
BOOL LucamTriggerFastFrame(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

To use this function, the camera should be in Fast Frames mode using the LucamEnableFastFrames() function. This function will not wait for the return of the snapshot. To retrieve the snapshot, call either LucamTakeFastFrame() or LucamTakeFastFrameNoTrigger() functions.

This function is not supported currently by the MatLab plug-in.

LucamUnregisterEventNotification

Deregisters an event handle with the LuCam API.

Usage

```
BOOL LucamUnregisterEventNotification(HANDLE hCamera,
          PVOID pEventInformation);
```

Parameters

hCamera	[in] handle to the camera
pEventInformation	[in] handle to LuCam API event

Return Values

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

Remarks

The pEventInformation parameter of the LuCam API Event handle that was returned from the LucamRegisterEventNotification() function.

This function is not supported currently by the MatLab plug-in.

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

This function should only be used by knowledgeable users on the advice of Lumenera. Accessing camera registers could cause the camera to malfunction or damage it.

This function is not supported currently by the MatLab plug-in.

6

Constants and Structures Descriptions

6.1 Constants Definitions

Many of the parameters passed to API functions have been defined as constants in the API header file, lucamapi.h, which is included in the <LuCam Software>\SDK directory. The names of these constants and their definition are described in the following sections.

6.1.1 Camera Properties

These properties are use to access the camera settings.

LUCAM_PROP_BRIGHTNESS:	Controls the brightness parameter. This property can take values between -100 and 100.
LUCAM_PROP_CONTRAST:	Thjs property controls the contrast parameter. This property can take values between 0 and 100.
LUCAM_PROP_HUE:	Controls the hue parameter. This property can take values between -180 and 180.
LUCAM_PROP_SATURATION:	Controls the saturation parameter. This property can take values between 0.00 and 2.00.
LUCAM_PROP_SHARPNESS:	Controls the sharpness parameter. This property may not be available on all cameras.
LUCAM_PROP_GAMMA:	Controls the gamma property. This property can take values between -0.01 and 5.00.
LUCAM_PROP_PAN:	Controls the pan property. This property is only provided for compatibility with DirectShow and is not used by the cameras.
LUCAM_PROP_TILT:	Controls the tilt property. This property is only provided for compatibility with DirectShow and is not used by the cameras.
LUCAM_PROP_ROLL:	Controls the roll property. This property is only provided for compatibility with DirectShow and is not used by the cameras.

LUCAM_PROP_ZOOM: Controls the zoom property. This property is only provided for compatibility with DirectShow and is not used by the cameras.
LUCAM_PROP_EXPOSURE: Controls the exposure parameter.
LUCAM_PROP_IRIS: Controls the iris property. The values of this property correspond to the F-Stops of the lens.
LUCAM_PROP_FOCUS: Controls the focus property. The focus value is an arbitrary value relative to the location of the focal point of the lens when the camera was originally powered and/or the lens was connected. To select an absolute focal point use LUCAM_PROP_ABS_FOCUS property.
LUCAM_PROP_GAIN: Controls the analog gain parameter.
LUCAM_PROP_GAIN_RED: Controls the analog red gain parameter.
LUCAM_PROP_GAIN_BLUE: Controls the analog blue gain parameter.
LUCAM_PROP_GAIN_GREEN1: Controls the analog green1 gain parameter.
LUCAM_PROP_GAIN_GREEN2: Controls the analog green2 gain parameter.
LUCAM_PROP_GAIN_MAGENTA: Controls the analog magenta gain parameter.
LUCAM_PROP_GAIN_CYAN: Controls the analog cyan gain parameter.
LUCAM_PROP_GAIN_YELLOW1: Controls the analog yellow1 gain parameter.
LUCAM_PROP_GAIN_YELLOW2: Controls the analog yellow2 gain parameter.
LUCAM_PROP_DEMOSAICING_METHOD: This property sets the demosaicing methods used for the video preview. Refer to Section 6.1.4 for available demosaicing methods.
LUCAM_PROP_CORRECTION_MATRIX: This property sets the correction matrix used for the video preview. Refer to Section 6.1.5 for available demosaicing methods.
LUCAM_PROP_FLIPPING: This property sets the flip and mirror parameters.
LUCAM_PROP_DIGITAL_WHITEBALANCE_U: Controls the digital white balance U parameter. Can take a value between -100 and 100.
LUCAM_PROP_DIGITAL_WHITEBALANCE_V: Controls the digital white balance V parameter. Can take a value between -100 and 100.
LUCAM_PROP_DIGITAL_GAIN: Controls the digital gain parameter. Can take a value between 0 and 2.0.
LUCAM_PROP_DIGITAL_GAIN_RED: Controls the digital red gain parameter. Can take a value between 0 and 2.5.
LUCAM_PROP_DIGITAL_GAIN_GREEN: Controls the digital green gain parameter. Can take a value between 0 and 2.5.
LUCAM_PROP_DIGITAL_GAIN_BLUE: Controls the digital blue gain parameter. Can take a value between 0 and 2.5.
LUCAM_PROP_COLOR_FORMAT: Identifies the Bayer color format of the camera. This is a read-only property.

LUCAM_PROP_MAX_WIDTH: Identifies the max width allowable for the camera. This is a read-only property.
LUCAM_PROP_MAX_HEIGHT: Identifies the max height allowable for the camera. This is a read-only property.
LUCAM_PROP_ABS_FOCUS: Controls the focus property using an absolute value. In order to use this property, the camera needs to have its focus and iris properties calibrated by calling LucamInitAutoLens() function.
LUCAM_PROP_BLACK_LEVEL: Controls the black level of the camera's sensor. Currently only available for the Lu120 series cameras.
LUCAM_PROP_STILL_KNEE1_EXPOSURE: Controls the still knee1 point for multislope captures.
LUCAM_PROP_STILL_KNEE2_EXPOSURE: Controls the still knee2 point for multislope captures.
LUCAM_PROP_STILL_KNEE3_EXPOSURE: Controls the still knee3 point for multislope captures.
LUCAM_PROP_VIDEO_KNEE: Controls the video knee point for multislope captures.
LUCAM_PROP_THRESHOLD: Controls the pixel intensity parameter that the camera uses when in count or filter pixel format mode.
LUCAM_PROP_AUTO_EXP_TARGET: Controls the pixel intensity that is used as the auto-exposure target. This value can be between 0-255. It corresponds to the desired luminance (average pixel value) of the frames in the video stream.
LUCAM_PROP_TIMESTAMPS: Controls the application of a timestamp in each received video frame. The first two bytes of the frame contain the timestamp when this feature is enabled.
LUCAM_PROP_SNAPSHOT_CLOCK_SPEED: Controls the snapshot clock speed used to read out the snapshots. Lower values setup faster snapshot clock speed.
LUCAM_PROP_AUTO_EXP_MAXIMUM: Identifies the maximum exposure value that will be set by the continuous autoexposure algorithm.
LUCAM_PROP_TEMPERATURE: Controls the temperature parameter for cooling-enabled cameras.
LUCAM_PROP_TRIGGER: Controls the HW trigger input polarity.
LUCAM_PROP_STILL_STROBE_DURATION: Controls the length of the strobe output signal when this signal is active.
LUCAM_PROP_FAN: Controls the cooling fans for a cooling-enabled camera.
LUCAM_PROP_SYNC_MODE: Controls the master/slave relationship between two cameras that are synchronized in video mode through a shared pixel clock.
LUCAM_PROP_SNAPSHOT_COUNT: Defines the number of consecutive snapshots to acquire on cameras that support fast multiple snapshots.

LUCAM_PROP_LSC_X: Applies a lens shading correction in the X direction. Used with large format cameras.
LUCAM_PROP_LSC_Y: Applies a lens shading correction in the Y direction. Used with large format cameras.

LUCAM_PROP_STILL_EXPOSURE: Controls the exposure while in Fast Frames (fast snapshot) mode.
LUCAM_PROP_STILL_GAIN: Controls the analog gain while in Fast Frames (fast snapshot) mode.
LUCAM_PROP_STILL_GAIN_RED: Controls the analog red gain while in Fast Frames (fast snapshot) mode.
LUCAM_PROP_STILL_GAIN_GREEN1: Controls the analog green1 gain while in Fast Frames (fast snapshot) mode.
LUCAM_PROP_STILL_GAIN_GREEN2: Controls the analog green2 gain while in Fast Frames (fast snapshot) mode.
LUCAM_PROP_STILL_GAIN_BLUE: Controls the analog blue gain while in Fast Frames (fast snapshot) mode.
LUCAM_PROP_STILL_GAIN_MAGENTA: Controls the analog magenta gain while in Fast Frames (fast snapshot) mode.
LUCAM_PROP_STILL_GAIN_YELLOW1: Controls the analog yellow1 gain while in Fast Frames (fast snapshot) mode.
LUCAM_PROP_STILL_GAIN_YELLOW2: Controls the analog yellow2 gain while in Fast Frames (fast snapshot) mode.
LUCAM_PROP_STILL_GAIN_CYAN: Controls the analog cyan gain while in Fast Frames (fast snapshot) mode.
LUCAM_PROP_JPEG_QUALITY: Controls the compression ratio used with JPEG images

6.1.2 Capability Flags

These flags are used to enable or disable a particular camera function.

LUCAM_PROP_FLAG_USE: Identifies that a particular property will be used. Is typically used to enable the auto features of the camera.
LUCAM_PROP_FLAG_AUTO: Identifies that a particular property's auto function will be controlled.
LUCAM_PROP_FLAG_STROBE_FROM_START_OF_EXPOSURE: Identifies to signal the strobe (and any strobe delay) from the start of exposure of the sensor. Typically, the strobe is signal from the point where the sensor array is completely exposed. With global shutters, the start of exposure and the point that the sensor is completely exposed is identical. For the rolling shutters, these values will be different.
LUCAM_PROP_FLAG_USE_FOR_SNAPSHOTS: Identifies to use the property value for snapshots. This is used only with the

	LUCAM_PROP_IRIS property. This flag allows the iris to be opened when the snapshot starts exposing.
LUCAM_PROP_FLAG_POLARITY:	Identifies that when accessing the LUCAM_PROP_TRIGGER that we are changing the HW trigger input polarity.
LUCAM_PROP_FLAG_LITTLE_ENDIAN:	Identifies that the camera uses Little Endian data format for representing 16 bit data.
LUCAM_PROP_FLAG_ALTERNATE:	Keeps the iris open before the first snapshot of a quick multiple snapshot capture.
LUCAM_PROP_FLAG_READONLY:	Identifies that a property is a read only property. This flag is used in the pFlags parameter of the LucamPropertyRange() function.
LUCAM_FRAME_FORMAT_FLAGS_BINNING:	Identifies that binning will be used instead of subsampling.

6.1.3 Pixel Formats

These properties set the camera to run into a specific mode or define the type of data that is present in a memory buffer. The following formats cannot be combined.

LUCAM_PF_8:	Selects or defines 8 bit raw data or 8 bit monochrome data.
LUCAM_PF_16:	Selects or defines 16 bit raw data or 16 bit monochrome data.
LUCAM_PF_24:	Selects or defines 24 bit color data; 8 bits for red, green and blue channels.
LUCAM_PF_YUV422:	Selects or defines 16 bit YUV data.
LUCAM_PF_32:	Selects or defines 32 bit color data; 8 bits for red, green, blue and alpha channels.
LUCAM_PF_48:	Selects or defines 48 bit color data; 16 bits for red, green and blue channels.
LUCAM_PF_COUNT:	Sets the camera to run in count mode. In this mode, the camera counts the number of pixels that are above a predefined pixel intensity value.
LUCAM_PF_FILTER:	Sets the camera to run in filter mode. In this mode, the camera only returns pixels that are above a predefined pixel intensity value.

6.1.4 Demosaic Methods

These values state which demosaicing method will be used to convert the raw Bayer data to color data.

LUCAM_DM_NONE:	Identifies that no demosacing method is used.
LUCAM_DM_FAST:	Identifies to use the fast demosaicing mehod. This method can be completed quickly but does not provide high quality images.

LUCAM_DM_HIGH_QUALITY: Identifies to use the high quality demosaicing mehod. This method provides high quality images at a medium rate of speed.
LUCAM_DM_HIGHER_QUALITY: Identifies to use the higher quality demosaicing mehod. This method provides higher quality images at a reduced rate of speed.
LUCAM_DM_SIMPLE: Identifies to use the fastest demosaicing mehod. This method can be completed very quickly by sacrificing image quality.

6.1.5 Correction Matrices

These values state which color correction matrix will be used to correct the color response of the pixel data.

LUCAM_CM_NONE: Identifies that no correction matrix is used.
LUCAM_CM_FLUORESCENT: Identifies to use the correction matrix to correct for fluorescent (office) lighting.
LUCAM_CM_DAYLIGHT: Identifies to use the correction matrix to correct for daylight (sunlight).
LUCAM_CM_INCANDESCENT: Identifies to use the correction matrix to correct for incandescent (home) lighting.
LUCAM_CM_XENON_FLASH: Identifies to use the correction matrix to correct for xenon flash lighting.
LUCAM_CM_HALOGEN: Identifies to use the correction matrix to correct for halogen lighting.
LUCAM_CM_IDENTITY:: Identifies to use the identity matrix to do color correction.
LUCAM_CM_CUSTOM: Identifies to use the custom correction matrix to do the color correction.

6.1.6 Color Formats

These values state what Bayer format used by the camera's sensor.

LUCAM_CF_MONO: Identifies that the camera is a monochrome camera.
LUCAM_CF_BAYER_RGGB: Identifies that the camera is a color camera where the Bayer data is Red-Green, Green-Blue.
LUCAM_CF_BAYER_GRBG: Identifies that the camera is a color camera where the Bayer data is Green-Red, Blue- Green.
LUCAM_CF_BAYER_GBRG: Identifies that the camera is a color camera where the Bayer data is Green-Blue, Red-Green.
LUCAM_CF_BAYER_BGGR: Identifies that the camera is a color camera where the Bayer data is Blue-Green, Green-Red.
LUCAM_CF_BAYER_CYYM: Identifies that the camera is a color camera where the Bayer data is Cyan-Yellow, Yellow-Magenta.
LUCAM_CF_BAYER_YCMY: Identifies that the camera is a color camera where the Bayer data is Yellow-Cyan, Magenta-Yellow.

LUCAM_CF_BAYER_YMCY: Identifies that the camera is a color camera where the Bayer data is Yellow-Magenta, Cyan-Yellow.
LUCAM_CF_BAYER_MYYC: Identifies that the camera is a color camera where the Bayer data is Magenta-Yellow, Yellow-Cyan.

6.1.7 External Interfaces

These values state which external USB interface associated with the camera.

LUCAM_EXTERN_INTERFACE_USB1: Identifies that the camera is connected to a USB 1.1 interface.
LUCAM_EXTERN_INTERFACE_USB2: Identifies that the camera is connected to a USB 2.0 interface.

6.1.8 Shutter Types

These flags state which shutter type to use for snapshot captures.

LUCAM_SHUTTER_TYPE_GLOBAL: Identifies to use the camera's global shutter to capture the snapshots.
LUCAM_SHUTTER_TYPE_ROLLING: Identifies to use the camera's rolling shutter to capture the snapshots.

6.1.9 Image Flipping

These flags state the flipping mode to use for the preview. These properties can be applied to the captured images. They will not take affect until the raw image data is converted to color through the LucamConvertFrameToRgbXX() functions or to greyscale through the LucamConvertFrameToGreyscallXX() functions.

LUCAM_PROP_FLIPPING_NONE: Sets the camera to not use flipping.
LUCAM_PROP_FLIPPING_X: Controls the flipping in the X direction (mirror) only.
LUCAM_PROP_FLIPPING_Y Controls the flipping in the Y direction only.
LUCAM_PROP_FLIPPING_XY Controls the flipping in the X and Y directions.

6.1.10 Video streaming modes

These flags control the state of the video stream.

STOP_STREAMING:	Stops the video stream or video preview.
START_STREAMING:	Starts the video stream with no video preview.
START_DISPLAY:	Starts the video stream with video preview.
PAUSE_STREAM:	Pauses the video stream or video preview.

6.1.11 AVI video preview controls

These flags control the stat of the AVI playback.

STOP_AVI:	Stops the AVI playback.

START_AVI:	Starts the AVI playback.
PAUSE_AVI:	Pause the AVI playback.

6.1.12 Video file conversion formats

These flags define the AVI file pixel format.

AVI_RAW_LUMENERA:	Defines that the pixel format of the captured AVI file is in RAW pixel format.
AVI_STANDARD_24:	Defines that the pixel format of the captured AVI file is in standard 24 bit color pixel format.
AVI_STANDARD_32:	Defines that the pixel format of the captured AVI file is in standard 32 bit color pixel format.

6.1.13 Event Notification Types

These values state the types of notifications that can be waited on using LucamRegisterEventNotification() function:

LUCAM_EVENT_GPI1_CHANGED:	Identifies that the event should be signalled on changes to GPI1.
LUCAM_EVENT_GPI2_CHANGED:	Identifies that the event should be signalled on changes to GPI2.
LUCAM_EVENT_GPI3_CHANGED:	Identifies that the event should be signalled on changes to GPI3.
LUCAM_EVENT_GPI4_CHANGED:	Identifies that the event should be signalled on changes to GPI4.
LUCAM_EVENT_DEVICE_SURPRISE_REMOVAL:	Identifies that the event should be signalled when the camera is disconnected from the USB bus.
LUCAM_EVENT_START_OF_READOUT:	Identifies that the event should be signaled when the camera has completed its exposure and is starting to readout the image from the sensor. This notification is only supported on the Lm075, Lm135, Lm165, Lu070, Lu130, Lw130, Lw160, Lw230 and Lw11050 based cameras.

6.2 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.

6.2.1 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)
```

6.2.2 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 {
  USHORT subSampleX; // sub-sample ratio in x direction in
                           pixels (x:1)
  USHORT binningX;  // binning ratio in x direction in
                   pixels (x:1)
}
USHORT flagsX;     // binning flag for x direction
union {
   USHORT subSampleY;  // sub-sample ratio in y direction
                   in pixels (y:1)
  USHORT binningY;  // binning ratio in y direction in
                   pixels (y:1)
}
USHORT flagsY;     // binning flag for y direction
```

6.2.3 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
```

6.2.4 LUCAM_CONVERSION Structure

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

6.2.5 LUCAM_CONVERSION_PARAMS Structure

```
ULONG Size;    // Size of this structure
ULONG DemosaicMethod;    // Demsaicing method in use
ULONG CorrectionMatrix;  // Correction matrix in use
BOOL FlipX;    // Flip X mode in use
BOOL FlipY;    // Flip Y mode in use
FLOAT Hue;     // Hue value in use
FLOAT Saturation;   // Saturation value in use
BOOL UseColorGainsOverWb; // Defines which structure to use
          in union
union
{
  struct
  {
    FLOAT DigitalGain;   // Digital gain value in use
    FLOAT DigitalWhiteBalanceU; // Digital white balance U
          value in use
    FLOAT DigitalWhiteBalanceV; // Digital white balance Y
          value in use
  };
  struct
  {
    FLOAT DigitalGainRed; // Digital red gain value in use
    FLOAT DigitalGainGreen; // Digital green gain value in
          use
    FLOAT DigitalGainBlue; // Digital blue gain value in
          use
  };
};
```

6.2.6 LUCAM_IMAGE_FORMAT Structure

```
ULONG Size;    // Size of this structure
ULONG Width;   // Width value in use
ULONG Height;  // Height value in use
ULONG PixelFormat;  // Pixel format value in use
ULONG ImageSize;    // Current image size
ULONG LucamReserved[8];  // Reserved for future use
```