LuCam Software Development Kit (SDK)

USER'S MANUAL

Release 2020-03-27-1666
www.teledynelumenera.com

TELEDYNE LUMENERA
Everywhereyoulook™

7 CAPELLA COURT, OTTAWA, ON, CANADA K2E 8A7 | TEL (613) 736-4077 | FAX (613) 736-4071 | LUMENERA.INFO@TELEDYNE.COM

Table of Contents

License Agreement (Software):

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

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

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

In using the Software you agree not to:

a) Decompile, disassemble, reverse engineer, or otherwise attempt to derive the source code for any Product (except to the extent applicable laws specifically prohibit such restriction);

b) Remove or obscure any trademark or copyright notices.

Limited Warranty (Hardware and Software):

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

Limited Liability (Hardware and Software):

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

1 Introduction

1.1 Teledyne Lumenera API

The Teledyne Lumenera Camera API (LuCam API) provides a comprehensive set of functions allowing you to control the operation of any Lumenera USB, or GigE, camera.

Directly callable from Visual C++, Visual Basic, Visual C#, Borland C++ Builder and any .NET 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 use them. The sample code included with the Software Developer's Kit (SDK) provides many examples of how to use the functions in real-world applications.

The Teledyne Lumenera Technical Assistance Center (TAC) team is available to provide assistance in the use of the API so you can get the most out of your camera application.

Contact the TAC team via email at: lumenera.support@teledyne.com

Please visit the Support section on the Teledyne Lumenera website for FAQs and access to the Knowledge Base. www.lumenera.com

2 Summary of Functions

2.1 Alphabetical Summary of Functions

Function	Description
LgcamGetIPConfiguration	Gets the Ethernet IP configuration for GigE cameras.
LgcamSetIPConfiguration	Sets the Ethernet IP configuration for GigE cameras.
LucamAddRgbPreviewCallback	Allows the user to add a video filter call back 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 call back function, which is called after each hardware or software triggered snapshot is returned from the camera but before it is processed.
LucamAddStreamingCallback	Allows the user to add a video filter call back 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.
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.
LucamContinuousAutoExposureEnable	Enable auto exposure control with selected ROI.
LucamContinuousAutoExposureDisable	Disable the auto exposure control that have been started with LucamContinuousAutoExposureEnable

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
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.
LucamDataLsbAlign	Shift raw image data to match camera bitdepth.
LucamDestroyDisplayWindow	Destroys the display window created with LucamCreateDisplayWindow.

Function	Description
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.
LucamEnableInterfacePowerSpecViolation	Enable the usage of USB 3.0 Y cable to power camera.
LucamEnableSynchronousSnapshots	Enables the simultaneous snapshot capture mode.
LucamEnableTimestamp	Enables insertion of timestamp information in captured images.
LucamEnumAvailableFrameRates	Returns an array containing the available frame rates for the camera based on the clock rates available on the camera.
LucamEnumCameras	Returns the version information and serial numbers for all Lumenera cameras attached to the computer.
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, sub sampling, pixel format) and desired frame rate for the video data.
LucamGetHardwareRevision	Get the current hardware revision of the camera.
LucamGetImageIntensity	Gets the pixel intensity value of a given image.
LucamIsInterfacePowerSpecViolationEnabled	Return status of USB 3.0 Y cable power usage.
LucamGetLastError	Gets the specific error code for the last error that occurred when calling an API function.
LucamGetLastErrorForCamera	Gets the specific error code for the last error that occurred when calling an API function for a given camera.
LucamGetMetaData	Gets metadata information inserted in images.
LucamGetProperty	Gets the value of the specified camera property.
LucamGetStillImageFormat	Returns the snapshot image format used to capture a snapshot.
LucamGetTimestamp	Gets actual timestamp counter from camera.
LucamGetTimestampFrequency	Gets number of counts per second from camera.
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.
LucamGpioConfigure	Configure the General Purpose I/O pins as inputs or outputs.

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.
LucamInitAutoLens	Initialize and calibrate the focus and iris positions of the camera lens.
LucamIsTimestampEnabled	Check if the timestamp function is enabled on actual camera model.
LucamLedSet	Enable or disable camera status LED.
LucamNumCameras	Returns the number of Lumenera cameras attached to the computer.
LucamOneShotAutoExposure	Performs a single (one iteration) exposure adjustment on the video stream in an attempt to reach the auto-exposure target.
LucamOneShotAutoExposureEx	Performs a single iteration exposure adjustment on the video stream in an attempt to reach the auto-exposure target for a specific lightning condition.
LucamOneshotAutoGain	Performs a single iteration gain adjustment on the video stream in an attempt to reach the image intensity target.
LucamOneshotAutoIris	Performs a single (one iteration) Iris adjustment in an attempt to reach the image intensity 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.
LucamPerformMonoGridCorrection	Performs an additional correction on a monochrome capture image from cameras.
LucamPerformMultiTapCorrection	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.

Function	Description
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.
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 call back function registered using the function LucamAddRgbPreviewCallback.
LucamRemoveSnapshotCallback	Removes the specified data filter call back function registered using the function LucamAddSnapshotCallback.
LucamRemoveStreamingCallback	Removes the specified video filter call back 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.
LucamSelectExternInterface	Selects external Interface to connect with Lumenera cameras (either USB or GigE)
LucamSetFormat	Sets the video frame format (subwindow position and size, sub sampling, pixel format) and desired frame rate for the video data.
LucamSetProperty	Sets the value of the specified camera property.
LucamSetTimestamp	Assign a value to camera timestamp counter.
LucamSetTriggerMode	Toggles between SW triggered and HW triggered snapshots in Fast Frames mode.

Function	Description
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.
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 frames from the camera, using the camera's video mode.
LucamTriggerFastFrame	Initiates the request to take a snapshot.
LucamUnregisterEventNotification	Deregisters an event with the Lucam API
LucamWriteRegister	Writes to the internal camera registers.

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.
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.
LucamGetLastErrorForCamera	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
LgcamGetIPConfiguration	Get the Ethernet IP configuration for GigE camera.
LgcamSetIPConfiguration	Set the Ethernet IP configuration for GigE camera.
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.
LucamContinuousAutoExposureEnable	Enable auto exposure control with selected ROI.
LucamContinuousAutoExposureDisable	Disable the auto exposure control that have been started with LucamContinuousAutoExposureEnable
LucamGetCameraID	Gets the camera model ID number.
LucamGetHardwareRevision	Gets the current hardware revision of the camera.
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.
LucamSelectExternInterface	Selects the external interface used to connect with the camera
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.

Function	Description
LucamOneShotAutoExposure	Performs a single (one iteration) exposure adjustment on the video stream in an attempt to reach the auto exposure target.
LucamOneShotAutoExposureEX	Performs a single iteration exposure adjustment on the video stream in an attempt to reach the auto-exposure target to a specific lightning condition.
LucamOneShotAutoGain	Performs a single iteration gain adjustment on the video stream in an attempt to reach the image intensity target.
LucamOneshotAutoIris	Performs a single (one iteration) iris adjustment in an attempt to reach the image intensity 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.
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.
LucamSequencingCancelTakeSequence	Can the active trigger sequence
LucamSequencingGetIndexForFrame	Get the frame buffer index of the trigger sequence
LucamSequencingGetStatus	Get current status of the trigger sequencing mode
LucamSequencingTakeSequence	Activate current sequence and wait for all frames of the sequence.
LucamSequencingSetup	Initialise trigger sequencing 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, sub sampling, pixel format) and desired frame rate for the video data.
LucamSetFormat	Sets the video frame format (subwindow position and size, sub sampling, pixel format) and desired frame rate for the video data.
LucamStreamVideoControl	Controls the streaming video.
LucamCreateDisplayWindow	Creates a display window, which is managed by the API, for displaying video.
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 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
LucamDataLsbAlign	Shift raw image data to match camera bitdepth.
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 frames from the camera, using the camera's video mode.
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.

Function	Description
LucamConvertFrameToRGB24	Converts a raw frame of data obtained with LucamTakeVideo or LucamTakeSnapshot to a fully processed RGB24 frame suitable for display or saving
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
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).
LucamPerformDualTapCorrection	Performs an additional correction on a captured image from cameras that have more than one sensor readout tap.
LucamPerformMonoGridCorrection	Performs an additional correction on a monochrome capture image from cameras.
LucamPerformMultiTapCorrection	Performs an additional correction on a captured image from cameras that have more than one sensor readout tap.
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 Call back Handling

Function	Description
LucamAddSnapshotCallback	Allows the user to add a data filter call back function, which is called after each snapshot (hardware or software trigger) is returned from the camera but before it is processed.
LucamRemoveSnapshotCallback	Removes the specified data filter call back function registered using the function LucamAddSnapshotCallback.
LucamAddStreamingCallback	Allows the user to add a video filter call back 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 call back function registered using the function LucamAddStreamingCallback.
LucamAddRgbPreviewCallback	Allows the user to add a video filter call back 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 call back function registered using the function LucamAddRgbPreviewCallback.

2.2.8 Register and External I/O Access

Function	Description
LucamGpioConfigure	Configure the General Purpose I/O pins as inputs or outputs.
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.

Function	Description
LucamOneShotAutoIris	Performs a single (one iteration) iris adjustment on the video stream in an attempt to reach the image Intensity target.

2.2.10 **Timestamp**

Function	Description
LucamEnableTimestamp	Enables insertion of timestamp information in captured images.
LucamGetMetaData	Gets metadata information inserted in images.
LucamGetTimestamp	Gets actual timestamp counter from camera.
LucamGetTimestampFrequency	Gets number of counts per second from camera.
LucamIsTimestampEnabled	Check if the timestamp function is enabled on actual camera model.
LucamSetTimestamp	Assign a value to camera timestamp counter.

2.2.11 **Power specification violation.**

Function	Description
LucamEnableInterfacePowerSpecViolation	Enable the usage of USB 3.0 Y cable to power camera.
LucamIsInterfacePowerSpecViolationEnabled	Return status of USB 3.0 Y cable power usage.

2.2.12 **Trigger Sequencing**

Function	Description
LucamSequencingCancelTakeSequence	Cancel the active trigger sequence
LucamSequencingGetIndexForFrame	Get the frame buffer index of the trigger sequence
LucamSequencingGetStatus	Get current status of the trigger sequencing mode
LucamSequencingTakeSequence	Activate current sequence and wait for all frames of the sequence.
LucamSequencingSetup	Initialize trigger-sequencing mode.

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 is accomplished using this subet of functions.

The functions identified in 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. Refer to the source code samples provided with the SDK to gain a more complete understanding of how the basic and advanced API functions are used. If you have questions, e-mail the TAC team 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 connection type (e.g. USB1.1 USB2.0, USB 3.0, or GigE) you can call ***LucamQueryExternInterface()***.

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

To obtain the hardware revision for the camera, you can use the ***LucamGetHardwareRevision()*** function.

3.2.3 Camera operation modes

Your camera can deliver image frames to the host in an unsynchronized mode where all frames are transferred to the computer as fast possible, without any interventions; this is also referred to as *video*

mode. The second mode is a synchronized mode where a software or hardware trigger is required in order for the camera to start an exposure. This mode is referred to as the *snapshot mode* or *still mode*.

3.2.3.1 Video Mode

To capture images with LucamTakeVideo(), a video stream need to be started. The stream can be started using LucamStreamVideoControl() and could be started to stream into memory (START_STREAMING) or showing to the screen (START_DISPLAY). The stream will remain suspended until completion of LucamTakeSnapshot() and the call of LucamEnableFastFrames() until LucamDisableFastFrames() combination is completed. It is also important to know, that when you enter to one of the streaming mode you cannot switch to the other one. In order to switch to another streaming mode, the first stream need to be shutdown using LucamStreamVideoControl() with the STOP_STREAMING flags.

3.2.3.1.1 Preview Video

Once a camera has been opened, a video preview can be displayed simply by calling ***LucamStreamVideoControl()***[section 5.124], Annex A is showing example of steps to start video stream. 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 delivered to the specified window.

Pausing or stopping the preview is 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()***.

3.2.3.1.2 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.)

3.2.3.1.3 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. Sub sampling/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 LUCAM_PROP_UNIT_WIDTH or LUCAM_PROP_UNIT_HEIGHT multiplied by the sampling factor.
2. The subwindow position is provided as the x and y coordinates of the top left corner of the subwindow. Both the width and height must be a multiple of LUCAM_PROP_UNIT_WIDTH or LUCAM_PROP_UNIT_HEIGHT multiplied by the sampling factor.
3. Sub sampling 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.

3.2.3.1.4 Grab Video Data

You have two possible modes to get images data from the Lumenera camera. The first one is the synchronize mode also referred to as still mode or snapshot mode and is described in section 3.2.3.2.1. The second method to retrieve image data is unsynchronized mode, also refered as video mode. In video mode, the camera will send as fast possible one image after the other to the host. The video stream can be controlled with the ***LucamStreamVideoControl()*** function described in section 5.124. The stream can be directed to memory or to DirectShow windows. Now that the stream is turned on, video data can be grabbed from the stream using the ***LucamTakeVideo()*** described in section 5.130. The number of frames must be specified as well as the 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 preview 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. However, if switching from memory stream to previewing stream or vice versa, it is necessary to first stop the actual stream and create the desired stream.

3.2.3.2 Snapshot mode

3.2.3.2.1 Take a Snapshot (or many)

Single snapshots can be taken using one of several functions. The ***LucamTakeSnapshot()*** function can be used any time to asynchronously capture a single image using the camera's snapshot mode. Even if the camera is streaming video, the function will suspend the video stream while the image is acquired. 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 images can be captured repeatedly using the ***LucamTakeFastFrames()*** function at a faster rate than with ***LucamTakeSnapshot()***. If a video stream is active when calling the **LucamEnableFastFrame()**, the stream will be suspended until the **LucamDisableFastFrame()** is called. While the video stream is suspended it is still possible to stop the

video stream and restart it, however the stream will be suspended until the **LucamDisableFastFrame()** is executed.

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

3.2.3.2.2 Take images in a burst (burst mode)

Burst mode enables users to take multiple snapshots as fast as possible with a single software or hardware trigger. Not all camera models support burst mode but it is possible to interrogate the camera by reading the LUCAM_PROP_SNAPSHOT_COUNT property. If the read operation returns with success then the camera supports burst mode and the return value of LucamGetProperty() represents the actual number of frames per trigger. It is possible to get the number of supported frames by using LucamPropertyRange() on the LUCAM_PROP_SNAPSHOT_COUNT property. If the camera output is set to 16-bit mode, then the maximum number of frames per trigger available will be reduce in order to be store frames into camera memory (Camera models dependent).

In order to use the camera's burst mode, you first need to adjust the LUCAM_PROP_SNAPSHOT_COUNT property to the number of desired frames per trigger (greater than 1), then enter into snapshot mode by using the LucamEnableFastFrames(). The LucamTakeFastFrame() will trigger and return the first frame of the burst mode. To extract any remaining frames from the camera, you need to execute a loop using the function LucamTakeFastFrameNoTrigger().

See Also:
LucamEnableFastFrames(), LucamDisableFastFrames(), LucamTakeFastFrame(), LucamTakeFastFrameNoTrigger(), LucamGetProperty(), LucamSetProperty(), LucamPropertyRange(), LucamGetLastError(), LucamGetLastErrorForCamera(). LucamSequencingSetup().

3.2.4 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. In addition, the image is flipped horizontally 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.

3.2.5 Processed Color format

All frames converted to color contain RGB data. Since Windows optimize it display operations for bitmap format, all our information return is in packed BGR and BGRA format. It is also important to remember that rows information is inverted. Therefore, the first the line is the last line of the BGR data.

Under Linux, the RGB data is packed as RGB.

To be a little more convenient, we have provided constant definition to identify how the RGB data has been packed.

LUCAM_RGB_FORMAT_RGB, LUCAM_RGB_FORMAT_BMP:

```
#if defined(_WIN32)
#define LUCAM_API_RGB24_FORMAT LUCAM_RGB_FORMAT_BMP
#define LUCAM_API_RGB32_FORMAT LUCAM_RGB_FORMAT_BMP
#define LUCAM_API_RGB48_FORMAT LUCAM_RGB_FORMAT_BMP
#elif defined(_linux_)
#define LUCAM_API_RGB24_FORMAT LUCAM_RGB_FORMAT_RGB
#define LUCAM_API_RGB32_FORMAT LUCAM_RGB_FORMAT_RGB
#define LUCAM_API_RGB48_FORMAT LUCAM_RGB_FORMAT_RGB
#else
#error "Unsupported platform"
#endif
```

3.2.6 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, lose 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.

3.2.7 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 Call back functions

The Video Call back 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.

Two functions can be used to register your call back. 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 call back function can be removed using either ***LucamRemoveStreamingCallback()*** or ***LucamRemoveRgbPreviewCallback()*** depending on which function was used to add it.

3.3.5 Snapshot Call back Functions

The Snapshot Call back 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 call back. The call back 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 will not be erased when the camera is powered down. The ***LucamPermanentBufferRead()*** function can be used to read the memory area. The number of write cycle to flash memory is limited to 100,000 cycles..

3.4 Camera Signals

3.4.1 GPO1 / Strobe Out:

This signal serves double duty and is also used to provide an ACTIVE LOW, 5.5 ms pulse (suitable for triggering a strobe unit) when any of the Take Snapshot API functions are used with the useStrobe option enabled. This strobe pulse can be delayed with respect to the start of frame exposure by a user selectable amount (see the Lumenera API Reference Manual for further details). This signal can be toggled using the LucamGpioWrite() function. For USB 3.0 product, the strobe signal can be programme to be used on GPIO2 or GPIO3.

3.4.2 **GPO2 / Strobe Out:**

This signal serves double duty and is also used to provide an ACTIVE HIGH, 5.5 ms pulse (suitable for triggering a strobe unit) when any of the Take Snapshot API functions are used with the useStrobe option enabled. This strobe pulse can be delayed with respect to the start of frame exposure by a user selectable amount (see the Lumenera API Reference Manual for further details). The strobe signal can be toggled using the LucamGpioWrite() function. For USB 3.0 product, the strobe signal can be programmed to be used on GPIO2 or GPIO3. When GPO2 is programmed to be used as a strobe signal, it simply inverts the polarity of GPO1, so this means that the GPO1 needs to be programmed as a strobe signal in order to generate the second strobe signal.

3.4.3 **GPO3:**

This signal can be toggled using the LucamGpioWrite() function. For USB 3.0 Lt's series, the GPO3 can also be used as the start of frame signal (SOF).

3.4.4 **GPO4 / Video SOF*:**

This signal serves double duty and is used to provide an ACTIVE HIGH, 85 **µ**s pulse each time a frame is output in video mode for most of the cameras. For some of the CCD based cameras*, the duration of the pulse reflects the exposure set in the camera and the falling edge represents the Start of Readout of the sensor. The LucamGpoSelect() API function is used to enable/disable the Video SOF signal.

This signal can be toggled using the LucamGpioWrite() function.

* Currently supported on the Lw070, Lw130, Lw160 and Lw230-based cameras.

* This functionality is reported on GPIO3 on USB 3.0 Lt's products.

3.4.5 **GPI1 / Trigger In:**

This signal serves double duty and is also used to receive an ACTIVE HIGH, LVTTL input (Vin min = 0 V, Vin max = 3.3 V) pulse which will trigger the taking of a snapshot, when any of the Take Snapshot API functions are used with the useHwTrigger option enabled. The active high pulse must have a minimum width of 0.5 **µ**s. There is no maximum limit to the trigger pulse width.

This signal is floating and **MUST** be driven at all times when being used. The signal status can be obtained by using the LucamGpioRead() function.

3.4.6 **GPI2:**

This signal is floating and MUST be driven at all times when being used. The signal status can be obtained by using the LucamGpioRead()function.

3.4.7 **GPI3:**

This signal is floating and MUST be driven at all times when being used. The signal status can be obtained by using the LucamGpioRead() function.

3.4.8 **GPI4:**

This signal is floating and MUST be driven at all times when being used. The signal status can be obtained by using the LucamGpioRead() function.

3.4.9 **VCC Output:**

This optional feature allows the camera to output a 3.3 V DC signal on Pin 16. The camera can source up to 50 mA of current from this pin. This feature is only available on Lw-based cameras that have been ordered with this option available. This feature is not available on existing Lu-based cameras. The large format cameras, mini cameras and GigE Vision cameras do not have this signal available on the external I/O connector.

3.4.10 **Ready Signal with snapshot operations:**

The ready signal is an output signal that is raised when camera is ready to accept trigger. This signal only has meaning when using camera with hardware trigger, however it will still produce when using camera in software trigger. The signal will be available on GPO2, so GPIO2 pin need to be configure as an output. In order for the signal to work, the trigger polarity need to be set so trigger are intercepted on rising edge. This signal does not interfere with the SOF signal, so it is possible to have the Ready signal and the SOF signal at the same time.

The signal is supported by listed product:

Lt345, Lt545, Lt945, Lt1245, Lt365, Lt665, Lt965, Lt1265, Lt16059, Lt29059, Lt425

To enable the feature use the existing API:
LucamGpioConfigure(hCamera, 0x02); // Setup GPO2 as output
LucamGpoSelect(hCamera, 0); // Disable GPIO on GPO2

3.5 Multithreading with the API

The Lumenera camera models can be use in multithreading environment. As some API functions require the usage of internal resources that are not sharable between threads, it is required to be aware of some restrictions. We have grouped functions that need to be running from same threads.

The first collection composes the camera connection functions. The execution of these functions needs to occur on the same thread. If another thread need to call any of these functions then calling LucamSelectExternInterface (0) should release all thread local objects maintained by the API.

- LucamNumCameras
- LucamEnumCameras
- LucamCameraOpen
- LucamSelectExternInterface

The function LucamCameraClose can be call on any thread. However, we still need to confirm that it is the case or the GigE camera.

The next cluster of functions integrates the camera functions that control the camera stream. These functions have to run from the same thread but not necessary from the thread that ran the camera connection functions.

- LucamStreamVideoControl
- LucamStartPreview
- LucamEnableFastFrames
- LucamDisableFastFrames
- LucamStreamVideoControlAVI
- LucamCreateDisplayWindow
- LucamDestoryDisplayWindow
- LucamAdjustDisplayWindow
-

It is important to note the function “LucamCreateDisplayWindow” and “LucamStreamVideoControl (START_DISPLAY)” needs to have a message loop.

Even if we recommend using one thread, it is possible calling “LucamTakeVideo” or “LucamTakeFastFrame” from any thread. However, multiple calls of these frames capture will return the same frame to these threads. If using the buffer last frame option in snapshot mode, then it is limited to run on one thread.

Any application will benefit from using callback functions (Streaming, snapshot or Preview). While a callback function runs, it is required to avoid usage of blocking functions. Still in the execution of the callback, it is not recommend calling API functions. However, some of the API call will run without any issues such as “LucamSetProperty”. Accessing to a graphical user interface control directly may generate a dead lock as this blocks the main thread. To avoid the dead lock, actions should be post notification to the GUI threads.

Please note the API assumes passed handle from the application is valid. The API will not perform any verification on the handle and it is to the application responsibility to synchronize a call to “LucamCameraClose”.

On Windows operating system and with USB products, it is possible to create multiple handle per camera and only one handle at the time can use camera to preview. On the same note, the creation of camera handle is limited to one per camera with GigE product or other operating system (Linux, MAC).

3.6 SDK Sample Code Desciption

Included in the SDK are several sample applications that demonstrate the use of many LuCam API functions. The sample applications are installed at same location of the SDK.

On 64bit Windows it is installed in "C:\Program Files (x86)\Lumenera Corporation\Lumenera Camera SDK\Sample Applications"

On 32bit Windows it is installed in "C:\Program Files\Lumenera Corporation\Lumenera Camera SDK\Sample Applications"

All samples projects are based on Visual Studio 2010 (C++, C# and VB.NET).

4 Camera Support for Third Party Software

This section describes the various third party software interfaces that are supported by all Lumenera 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 was designed to support all Lumenera 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 the Teledyne Lumenera website at:

www.lumenera.com/support/downloads/industrial-downloads.php.

Or

www.lumenera.com/support/downloads/microscopy-downloads.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 Acquisition (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 all Lumenera 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 that, 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. Therefore, 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 the **Detailed API Description** Chapter for more information on supported functions.

The LabVIEW plug-in designed for the Lumenera USB 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.

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

The Detailed API Description Chapter includes more information on the functions that are currently supported.

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.

The LabVIEW plug-in can be accessed from either of these locations:

www.lumenera.com/support/downloads/industrial-downloads.php

Or

www.lumenera.com/support/downloads/microscopy-downloads.php

5 Detailed API Description

5.1 LgcamGetIPConfiguration

Get the IP Configuration for a GigE camera.

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☒	MATLAB	☐

Usage

```
BOOL LgcamGetIPConfiguration(ULONG index,
                             UCHAR cameraMac[6],
                       LGCAM_IP_CONFIGURATION *pCameraConfiguration,
                             UCHAR hostMac[6],
                             LGCAM_IP_CONFIGURATION *pHostConfiguration);
```

Parameters

Index	[in] Camera identification number
cameraMac[6]	[out] Camera MAC Address.
*pCameraConfiguration	[out] Ethernet IP configuration.
hostMac[6]	[out] Host MAC address.
*pHostConfiguration	[out] Host Ethernet IP configuration.

Return Values

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

Remarks

The Ethernet IP configuration structure is describe in section 7.2.7

5.2 LgcamSetIPConfiguration

Set the IP Configuration for GigE camera.

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☒	MATLAB	☐

Usage

```
BOOL LgcamSetIPConfiguration(ULONG index,
                        LGCAM_IP_CONFIGURATION *pCameraConfiguration);
```

Parameters

Index [in] Camera identification number.
*pCameraConfiguration [in] Ethernet IP configuration structure.

Return Values

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

Remarks

The Ethernet IP configuration structure is describe in section 7.2.7

5.3 LucamAddRgbPreviewCallback

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

Platform support

Windows	☒	Linux	☐	Mac	☒

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☐	MATLAB	☐

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 call back 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 call back context data
rgbPixelFormat	[in] pixel format of data

Return Values

If the function succeeds, the return value is the unique call back 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 call back 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

5.4 LucamAddSnapshotCallback

Allows the user to add a data filter call back function, which is called after each hardware or software triggered snapshot is returned from the camera but before it is processed.

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☐

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 call back 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 call back context data

Return Values

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

Remarks

The declaration for the snapshot call back 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

5.5 LucamAddStreamingCallback

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

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☐

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 call back 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 call back context data

Return Values

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

Remarks

The declaration for the streaming call back 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

5.6 LucamAdjustDisplayWindow

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

Platform support

Windows ☒	Linux ☐	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☒

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.

5.7 LucamAdjustWhiteBalanceFromSnapshot

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

Platform support

Windows ☒	Linux ☐	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☒

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.

5.8 LucamAutoFocusQueryProgress

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

Platform support

Windows ☒	Linux ☒	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☒

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.

5.9 LucamAutoFocusStart

Starts an auto focus calibration on cameras with a Canon EF lens model

Platform support

Windows	☒	Linux	☒	Mac	☐

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☒	MATLAB	☒

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 zero
putZeroThere2	[in] reserved value and should be set to zero
putZeroThere3	[in] reserved value and should be set to zero
ProgressCallback	[in] optional call back function pointer
contextForCallback	[in] context parameter for call back function

Return Values

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

Remarks

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

The declaration for the progress call back 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

5.10 LucamAutoFocusStop

Stops the auto focus calibration.

Platform support

Windows ☒	Linux ☒	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☒

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 pre-set 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.

5.11 LucamAutoFocusWait

Waits for the completion of the auto focus calibration.

Platform support

Windows ☒	Linux ☒	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☒

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. The timeout value is express in milliseconds.

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 either when 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.

In the case of an error, so returned values is FALSE. The error code can be retrieve by using LucamGetLastErrorForCamera() or LucamGetLastError() function. The error code should be 19 for Timeout error.

5.12 LucamAutoRoiGet

Returns the region of interest used for the auto functions.

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☐	MATLAB	☒

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.

5.13 LucamAutoRoiSet

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

Platform support

Windows ☒	Linux ☐	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☒

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.

5.14 LucamCameraClose

Closes a connection to a Lumenera camera.

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☒

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.

5.15 LucamCameraOpen

Opens a connection to a Lumenera camera.

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☒

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()*.

5.16 LucamCameraReset

Resets the camera to its power-on default state.

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☒

Usage

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

Parameters

hCamera [in] handle to the camera

Return Values

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

Remarks

None.

5.17 LucamCancelTakeFastFrame

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

Platform support

Windows ☒	Linux ☐	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☐

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`.

5.18 LucamCancelTakeVideo

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

Platform support

Windows ☒	Linux ☐	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☐	LabVIEW ☐	MATLAB ☐

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`.

5.19 LucamContinuousAutoExposureEnable

Enable auto exposure control with selected region of interest (ROI).

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☐	LabVIEW	☐	MATLAB ☐

Usage

```
void LucamContinuousAutoExposureEnable(m_hCamera,
                UCHAR autoExposureTarget,
                ULONG startX,
                ULONG startY,
                ULONG width,
                ULONG height,
                FLOAT lightingPeriod
                );
```

Parameters

m_hCamera	[in] Handle to the camera
autoExposureTarget	in] Auto exposure target, value from 0-255.
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
lightingPeriod	[in] Inverse of the lighting frequency in milliseconds (being either 1/60/2hz or 1/50/2hz)

Return Values

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

Remarks

If width or height are zero then API interprets each as full width, full height.
If the lighting period is 0.0, the API ignores this parameter.
This function is not supported currently by the MATLAB plug-in.

5.20 LucamContinuousAutoExposureDisable

Disable the auto exposure control that has been started with LucamContinuousAutoExposureEnable.

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☐	LabVIEW	☐	MATLAB	☐

Usage

```
BOOL LucamContinuousAutoExposureDisable(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.

5.21 LucamConvertBmp24ToRgb24

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

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☐	LabVIEW	☐	MATLAB ☐

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

None.

5.22 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.

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☐	MATLAB	☐

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 7.2.4. The pixel format should be LUCAM_PF_8. The output frame consists of 8-bit pixels of greyscale data.

5.23 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.

Platform support

Windows ☒	Linux ☐	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☐	LabVIEW ☐	MATLAB ☐

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] Memory address of the 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 7.2.6. 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 7.2.5. The output frame consists of 8 bit pixels of greyscale data.

5.24 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).

Platform support

Windows ☒	Linux ☐	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☐

Usage

```
BOOL LucamConvertFrameToGreyscale16(HANDLE hCamera,
                USHORT *pDest,
                const USHORT *pSrc,
                LUCAM_IMAGE_FORMAT *pImageFormat,
                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()
*pImageFormat	[in] Memory address of the 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

The LUCAM_CONVERSION structure is described in Section 7.2.4. The pixel format should be LUCAM_PF_16. The output frame consists of 16 bit pixels of greyscale data.

5.25 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).

Platform support

Windows ☒	Linux ☐	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☐	LabVIEW ☐	MATLAB ☐

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 7.2.6. 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 7.2.5. The output frame consists of 16 bit pixels of greyscale data.

5.26 LucamConvertFrameToRGB24

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

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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 7.2.4. The available pixel formats are listed in Section 7.1.2. The RGB24 data format has 24 bits per pixel. The three bytes of each pixel are the Blue, Green, Red values in that order.

5.27 LucamConvertFrameToRGB24Ex

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

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☐

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 7.2.6. 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 7.2.5. The RGB24 data format has 24 bits per pixel. The three bytes of each pixel are the Blue, Green, Red values in that order.

5.28 LucamConvertFrameToRGB32

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

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☐	LabVIEW ☒	MATLAB ☐

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 7.2.4. The available pixel formats are listed in Section 7.1.2. 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.

5.29 LucamConvertFrameToRGB32Ex

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

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☐	LabVIEW ☐	MATLAB ☐

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 7.2.6. 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 7.2.5. 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.

5.30 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).

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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 7.2.4. The pixel format must 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.

5.31 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).

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☐

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 7.2.6. 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 7.2.5. 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.

5.32 LucamConvertRawAVIToStdVideo

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

Platform support

Windows ☒	Linux ☐	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☐

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 from the output file name. Note that the output file could be 3 to 4 times larger than the original raw AVI file.

5.33 LucamCreateDisplayWindow

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

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☐	MATLAB	☐

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.

5.34 LucamDestroyDisplayWindow

Destroys the display window created with *LucamCreateDisplayWindow*.

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☐	MATLAB	☐

Usage

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

Parameters

hCamera	[in] handle to the camera

Return Values

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

Remarks

None.

5.35 LucamDataLsbAlign

By default 16 bit image pixels values are shift to the right to provide most significant bit alignment. This allows most imaging applications to display tiff with a good brightness. However, in 16 bit mode the true pixel depth of the camera can vary (10, 12 or 14 bit). In order to work with true camera data information it might be desirable to have data in less significant bit alignment.

Platform support

Windows ☒	Linux ☐	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☒

Usage

```
BOOL LucamDataLsbAlign(HANDLE hCamera,
                LUCAM_IMAGE_FORMAT *pLif,
                UCHAR *pData);
```

Parameters

hCamera	[in] handle to the camera
LUCAM_IMAGE_FORMAT*	[in] Image format definition of the raw data.
UCHAR*	[in] Image raw data values.

Return Values

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

Remarks

See Also

LucamGetTruePixelDepth, LucamGetStillImageFormat, LucamGetVideoImageFormat

5.36 LucamDigitalWhiteBalance

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

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☒

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.

5.37 LucamDigitalWhiteBalanceEx

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

Platform support

Windows ☒	Linux ☐	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☒

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.

5.38 LucamDisableFastFrames

Disables the fast snapshot capture mode.

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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.

5.39 LucamDisableSynchronousSnapshots

Disables the simultaneous snapshot capture mode.

Platform support

Windows ☒	Linux ☐	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☐

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.

5.40 LucamDisplayPropertyPage

Pops up a Direct Show dialog window with the camera properties.

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☐	MATLAB	☐

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.

5.41 LucamDisplayPutMessageDrain

Send the preview window mouse event to the application.

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☐	MATLAB ☐

Usage

```
BOOL LucamDisplayPutMessageDrain (int hCamera,
                int  hWindow);
```

Parameters

hCamera	[in] handle to the camera
hWindows	[in] handle to windows created by LucamStreamVideoControl().

Return Values

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

Remarks

None.

5.42 LucamDisplayVideoFormatPage

Pops up a Direct Show dialog window with the video properties.

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☐	MATLAB	☒

Usage

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

Parameters

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

Return Values

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

Remarks

None.

5.43 LucamEnableFastFrames

Enables the fast snapshot capture mode.

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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 7.2.1.

5.44 LucamEnableInterfacePowerSpecViolation

Enables the usage of Y cable to provide power to USB 3.0 products. This functionality is not supported by all camera models.

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☐	LabVIEW	☐	MATLAB	☐

Usage

```
BOOL LucamEnableInterfacePowerSpecViolation(HANDLE hCamera, BOOL enable);
```

Parameters

hCamera	[in] handle to the camera
enable	[in] true to enable or false to disable power spec violation feature.

Return Values

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

Remarks

Not all camera models support this features.

5.45 LucamEnableSynchronousSnapshots

Enables the simultaneous snapshot capture mode.

Platform support

Windows ☒	Linux ☐	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☐

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.

5.46 LucamEnableTimestamp

Enables or disable insertion of timestamp information in captured images.

Platform support

Windows	☒	Linux	☒	Mac	☐

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☐	MATLAB	☐

Usage

```
BOOL LucamEnableSynchronousSnapshots(
                HANDLE hCamera,
                BOOL enable);
```

Parameters

hCamera	[in] handles of the camera
enable	[in] Flag to enable or disable timestamp function.

Return Values

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

Remarks

- Time stamp function is not supported by all camera models.

See Also

LucamGetMetadata(), LucamGetTimestamp(), LucamGetTimestampFrequency(), LucamIsTimestampEnable(), LucamSetTimestamp();

5.47 LucamEnumAvailableFrameRates

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

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☒

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 zero.

Remarks

Frame rates are in frames per second. Call this function with entryCount equal to zero. 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.

5.48 LucamEnumCameras

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

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☒

Usage

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

Parameters

*pVersionsArray	[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.

5.49 LucamForceTakeFastFrame

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

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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.

5.50 LucamGetCameraId

Gets the camera model ID number.

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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
Lw290C, Lw295C	0x1CD
Lu330C, Lu335C	0x09B
Lw330C, Lw335C	0x19B

Camera Model	ID
Lu370C, Lu375C	0x08B
Lw570C, Lw575	0x1C5
Lw620M, Lw625M, Lw620C, Lw625C	0x186
Lw11050C, Lw11056C, Lw11057C, Lw11058C, Lw11059C	0x1C8
Lm11059	0x2C8
Lw16059	0x1C9
InfinityX-21	0x0A0
Infinity1-1, Infinity 1	0x0A1
Infinity1-3, Infinity 3	0x0A3
Infinity1-5	0x1AC
Infinity1-6	0x1A6
Infinity 2	0x0A2
Infinity 2-1	0x1A2
Infinity 2-2	0x1A7
Infinity 2-3	0x1A4
Infinity 3	0x0A3
Infinity 3-1	0x1A5
Infinity 3-1URB	0x711
Infinity 3-3	0x713
Infinity 3-6	0x716
Infinity 3-9	0x719
Infinity 3-12	0x71C
Infinity 4	0x0A4
Infinity 4-2	0x1AA
Infinity 4-4	0x1AB
Infinity 4-11	0x1A8
Infinity X32	0x1A9
Lg235	0x40080
Lg11059	0x400c8
Lg11059ii	0x400ca
Lt420	0x604
Lt220	0x602
Lt340	0x643
Lt365	0x613
Lt540	0x645
Lt665	0x616
Lt940	0x649
Lt965	0x619
Lt1240	0x64c
Lt1260	0x61c
Lt16059H	0x630
Lt29059H	0x631

5.51 LucamGetCurrentMatrix

Gets the current color correction matrix being applied for video preview or during color conversion. See LucamSetupCustomMatrix() for more information about uploading a Matrix to the camera.

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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 is influenced by digital values from hue, saturation etc.…

5.52 LucamGetHardwareRevision

Get the current hardware revision from the camera.

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☐	LabVIEW ☐	MATLAB ☐

Usage

```
BOOl LucamGetHardwareRevision(HANDLE hCamera, ULONG *pRevision);
```

Parameters

hCamera	[in] Handle to the camera.
*pRevision	[out] Pointer to camera revision.

Return Values

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

Remarks

None.

5.53 LucamGetFormat

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

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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

5.54 LucamGetImageIntensity

Returns the pixel intensity value of a given image.

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☐	MATLAB	☐

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 immediately 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 color intensities will return the same value as the global intensity, pIntensity.
- startX, startY, width and height input parameters need to multiples of two.

5.55 LucamGetLastError

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

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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.

5.56 LucamGetLastErrorForCamera

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

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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.

5.57 LucamGetMetadata

Extract metadata information embedded in image.

Platform support

Windows	☒	Linux	☒	Mac	☐

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☐	MATLAB	☐

Usage

```
BOOL LucamGetMetadata(
                HANDLE hCamera,
                BYTE* pImageBuffer,
                LUCAM_IMAGE_FORMAT* pFormat,
                ULONG metaDataIndex,
                ULONGLONG* pTimestamp);
```

Parameters

hCamera	[in] handles of the camera
pImageBuffer	[in] Address of image data to extract metadata information from.
pFormat	[in] Lumenera image format information.
metaDataIndex	[in] index of the desired metadata information. See remarks section.
pTimestamp	[out] location to put 64 bit metadata information.

Return Values

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

Remarks

- Time stamp function is not supported by all camera models.
- MetaDataIndex valid value:
    i. LUCAM_METADATA_FRAME_COUNTER
    ii. LUCAM_METADATA_TIMESTAMP

See Also

LucamEnableTimestamp(), LucamGetMetadata(), LucamGetTimestamp(), LucamGetTimestampFrequency(), LucamIsTimestampEnable(), LucamSetTimestamp();

5.58 LucamGetProperty

Gets the value of the specified camera property.

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☒

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 **Error! Reference source not found.**. Not all roperties are supported by all cameras. If a property is unsupported the function will return a failed condition (FALSE) and the value of *pValue will be undefined. The allowable capability flags are listed in Section 7.1.1.

5.59 LucamGetStillImageFormat

Returns the snapshot image format used to capture a snapshot.

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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 7.2.6.

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.

5.60 LucamGetTimestamp

Get current timestamp counter value from camera.

Platform support

Windows	☒	Linux	☒	Mac	☐

Language Support

C++	☒	C# / VB .NET	☐	LabVIEW	☐	MATLAB	☐

Usage

```
BOOL LucamGetTimestamp(
                HANDLE hCamera,
                ULONGLONG* pTimeStamp);
```

Parameters

hCamera	[in] handles of the camera
pTimeStamp	[out] location to put 64 bit timestamp counter value.

Return Values

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

Remarks

- The timestamp support depends of the camera models

See Also

LucamEnableTimestamp(), LucamGetMetadata(), LucamGetTimestamp(), LucamGetTimestampFrequency(), LucamIsTimestampEnable(), LucamSetTimestamp();

5.61 LucamGetTimestampFrequency

Get number of counts produce by camera in one second.

Platform support

Windows	☒	Linux	☒	Mac	☐

Language Support

C++	☒	C# / VB .NET	☐	LabVIEW	☐	MATLAB	☐

Usage

```
BOOL LucamGetTimestampFrequency(
                HANDLE hCamera,
                ULONGLONG* pTimestampTickFrequency);
```

Parameters

hCamera	[in] handles of the camera
pTimeStampTickFrequency	[out] location to put 64 bit timestamp frequency counter value.

Return Values

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

Remarks

- Time stamp function is not supported by all camera models.

See Also

LucamEnableTimestamp(), LucamGetMetadata(), LucamGetTimestamp(), LucamGetTimestampFrequency(), LucamIsTimestampEnable(), LucamSetTimestamp();

5.62 LucamGetTruePixelDepth

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

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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.

5.63 LucamGetVideoImageFormat

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

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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 7.2.6.

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 later using any of the LucamConvertFrame***Ex() based functions.

5.64 LucamGpioRead

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

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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.

5.65 LucamGpioWrite

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

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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.

5.66 LucamGpioConfigure

Configures the direction of a bi-directional GPIO pin.

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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 one 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 zero puts the GPIO pin into its default input mode.

5.67 LucamGpoSelect

Enables and disables the alternate GPO functionality.

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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 one 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 zero 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)

5.68 LucamInitAutoLens

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

Platform support

Windows ☒	Linux ☒	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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.
- Camera that has support for P-IRIS does not need to call LucamInitAutoLens but they do have to initialise the LUCAM_PROP_IRIS_STEPS_COUNT for the lens currently in use.

5.69 LucamIsInterfacePowerSpecViolationEnabled

Get status of the power specification violation (usage of Y cable to power camera).

Platform support

Windows ☒	Linux ☐	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☐	LabVIEW ☐	MATLAB ☐

Usage

```
BOOL LucamIsInterfacePowerSpecViolationEnabled (HANDLE hCamera, BOOL* pIsEnable);
```

Parameters

hCamera	[in] handle to the camera
pIsEnable	[in] Location where to save status.

Return Values

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

Remarks

Not all camera models support this feature.

5.70 LucamIsTimestampEnabled

Get status of timestamp function.

Platform support

Windows ☒	Linux ☒	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☐	LabVIEW ☐	MATLAB ☐

Usage

```
BOOL LucamIsTimestampEnabled(
                HANDLE hCamera,
                BOOL* pIsEnable);
```

Parameters

hCamera	[in] handles of the camera
pIsEnable	[out] location to put enable status flag.

Return Values

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

Remarks

- Time stamp function is not supported by all camera models.

See Also

LucamEnableTimestamp(), LucamGetMetadata(), LucamGetTimestamp(), LucamGetTimestampFrequency(), LucamIsTimestampEnable(), LucamSetTimestamp().

5.71 LucamLedSet

When camera support the LucamLedSet() function, it will allow to enable or disable camera status LED

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☐	LabVIEW	☐	MATLAB	☐

Usage

```
BOOL LucamLedSet(HANDLE hCamera, ULONG led);
```

Parameters

hCamera	Camera handle.
Led	Multi color status flag. Bit 0 – for Green LED. Bit 1 – for Yellow LED.

Return Values

The function will return TRUE if succeeds.

Remarks

None.

5.72 LucamNumCameras

Returns the number of Lumenera cameras attached to the computer.

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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.

5.73 LucamOneShotAutoExposure

Performs a single exposure adjustment to attempt reaching the auto exposure target.

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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 exposes
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.

5.74 LucamOneShotAutoExposureEx

Performs a single (one iteration) exposure adjustment in an attempt to reach the image intensity target for a given illumination level.

Platform support

Windows	☒	Linux	☐	Mac	☒

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☐	MATLAB	☐

Usage

```
BOOL LucamOneShotAutoExposureEx(HANDLE hCamera,
                UCHAR target,
                ULONG startX,
                ULONG startY,
                ULONG width,
                ULONG height,
                Float lightingPeriod);
```

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 exposes
width	[in] width of window to auto expose
height	[in] height of window to auto expose
lighthingPeriod	[in] light period to calculate with. Ex (1/60hz/2). This parameter is expected to be in milliseconds.

Return Values

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

Remarks

None.

5.75 LucamOneShotAutoGain

Performs a single gain adjustment to reach the requested image intensity.

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☐

Usage

```
BOOL LucamOneShotAutoGain(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 use for auto gain adjustment
height	[in] height of window to use for auto gain adjustment

Return Values

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

Remarks

None.

5.76 LucamOneShotAutoIris

Performs a single iris adjustment in an attempt to reach the auto exposure target.

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☒	MATLAB	☐

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

- Camera model that support P-IRIS can also use this function.

5.77 LucamOneShotAutoWhiteBalance

Performs a single on-chip analog gain adjustment on the video stream in an attempt to color balance the image.

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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.
- startX, startY, width and height input parameters need to multiples of two.

5.78 LucamOneShotAutoWhiteBalanceEx

Performs a single on-chip analog gain adjustment on the video stream in an attempt to color balance the image to a specific target color.

Platform support

Windows ☒	Linux ☐	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☐

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 both redOverGreen and blueOverGreen to 1.0.
- startX, startY, width and height input parameters need to multiples of 2.

5.79 LucamPerformDualTapCorrection

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

Platform support

Windows ☒	Linux ☒	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☐

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

This function is supported by any camera that has taps. Taps are areas of sensor that can be read at the same time and the conversion from light to digital value is performing by more than one converters. There will be one tap per converter.

5.80 LucamPerformMonoGridCorrection

Performs an additional correction on a monochrome captured image from cameras.

Platform support

Windows ☒	Linux ☐	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☐

Usage

```
BOOL LucamPerformMonoGridCorrection(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

None.

5.81 LucamPerformMultiTapCorrection

Perform additional correction to captured image. The captured image has to be read with more than one sensor readout tap.

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☐

Usage

```
BOOL LucamPerformMultiTapCorrection(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

This function is supported by any camera that has taps. Taps are areas of sensor that can be read at the same time and the conversion from light to digital value is performing by more than one converters. There will be one tap per converter.

5.82 LucamPermanentBufferRead

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

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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.

5.83 LucamPermanentBufferWrite

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

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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.

5.84 LucamPreviewAVIClose

Closes the controller to an AVI file.

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☐	MATLAB	☐

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

None.

5.85 LucamPreviewAVIControl

Controls the previewing of an AVI video.

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☐	MATLAB	☐

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.

5.86 LucamPreviewAVIGetDuration

Returns the length of an open AVI file.

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☐	MATLAB	☐

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

None.

5.87 LucamPreviewAVIGetFormat

Returns the AVI file information.

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☐	MATLAB	☐

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.

5.88 LucamPreviewAVIGetFrameCount

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

Platform support

Windows ☒	Linux ☐	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☐

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

None.

5.89 LucamPreviewAVIGetFrameRate

Returns the recorded frame rate of the AVI file.

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☐	MATLAB	☐

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

None.

5.90 LucamPreviewAVIGetPositionFrame

Returns the current frame based position within the AVI file.

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☐	MATLAB	☐

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

None.

5.91 LucamPreviewAVIGetPositionTime

Returns the current time based position within the AVI file.

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☐	MATLAB	☐

Usage

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

Parameters

hAVI	[in] handle of the AVI controller
*pPositionMinutes	[out] minute portion of current position
*pPositionSeconds	[out] second portion of current position
*pPositionMilliseconds	[out] millisecond portion of current position
*pPositionMicroSeconds	[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

None.

5.92 LucamPreviewAVISetPositionFrame

Sets the current frame based position within the AVI file.

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☐	MATLAB	☐

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

None.

5.93 LucamPreviewAVISetPositionTime

Sets the current time based position within the AVI file.

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☐	MATLAB	☐

Usage

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

Parameters

hAVI	[in] handle of the AVI controller
pPositionMinutes	[in] minute portion of current position
pPositionSeconds	[in] second portion of current position
pPositionMilliseconds	[in] millisecond portion of current position
pPositionMicroSeconds	[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

None.

5.94 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.

Platform support

Windows ☒	Linux ☐	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☐

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

None.

5.95 LucamPropertyRange

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

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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**Error! Reference source not found.**. Not all roperties are supported by all cameras. If a property is unsupported, the function will return a failed condition (FALSE). The allowable capability flags are listed in Section 7.1.1.

5.96 LucamQueryDisplayFrameRate

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

Platform support

Windows ☒	Linux ☐	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☐

Usage

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

Parameters

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

Return Values

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

Remarks

None.

5.97 LucamQueryExternInterface

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

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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 7.1.7.

5.98 LucamQueryRgbPreviewPixelFormat

Returns the pixel format for the preview window.

Platform support

Windows	☒	Linux	☐	Mac	☒

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☐	MATLAB	☐

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 7.1.2

5.99 LucamQueryVersion

Returns version information about the camera.

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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 7.2.3.

5.100 LucamReadRegister

Reads the internal camera registers.

Platform support

Windows ☒	Linux ☐	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☐

Usage

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

Parameters

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

Return Values

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

Remarks

None.

5.101 LucamRegisterCallbackNotification

Registers a callback notification to be called if a known event be raised.

Platform support

Windows	☐	Linux	☐	Mac	☒

Language Support

C++	☒	C# / VB .NET	☐	LabVIEW	☐	MATLAB	☐

Usage

```
LONG LucamRegisterCallbackNotification(HANDLE hCamera,
                DWORD eventId,
                int (*callback)(void *context, ULONG eventId),
                void *contextForCallback);
```

Parameters

hCamera	[in] handle to the camera
eventId	[in] type of event
callback	[void *()] address of callback function.
contextForCallback	[void *] callback input parameters.

Return Values

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

5.102 LucamRegisterEventNotification

Registers an event handle with the LuCam API.

Platform support

Windows ☒	Linux ☐	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☐	LabVIEW ☐	MATLAB ☐

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 7.1.14 for more information on the types of event notifications that are available.

5.103 LucamRemoveRgbPreviewCallback

Removes the specified video filter call back function registered using the function LucamAddRgbPreviewCallback().

Platform support

Windows	☒	Linux	☐	Mac	☒

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☐	MATLAB	☐

Usage

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

Parameters

hCamera	[in] handle to the camera
callbackId	[in] call back 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

None.

5.104 LucamRemoveSnapshotCallback

Removes the specified data filter call back function registered using the function LucamAddSnapshotCallback().

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☐

Usage

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

Parameters

hCamera	[in] handle to the camera
callbackId	[in] call back ID returned from LucamAddSnapshotCallback()

Return Values

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

Remarks

None.

5.105 LucamRemoveStreamingCallback

Removes the specified video filter call back function registered using the function LucamAddStreamingCallback().

Platform support

Windows	☒	Linux	☒	Mac	☒

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☐	MATLAB	☐

Usage

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

Parameters

hCamera	[in] handle to the camera
callbackId	[in] call back ID returned from LucamAddStreamingCallback()

Return Values

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

Remarks

None.

5.106 LucamSaveImage

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

Platform support

Windows ☒	Linux ☒	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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 Photographic Experts Group (.jpg), Tagged Image File Format (.tif) and Raw (.raw).
The available pixel formats are listed in Section 7.1.2.
If an unsupported file type (indicated by its extension) is provided, the function will fail.

5.107 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.

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☐

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 Photographic Experts Group (.jpg), Tagged Image File Format (.tif) and Raw (.raw).
The available pixel formats are listed in Section 7.1.2.
If an unsupported file type (indicated by its extension) is provided, the function will fail.

5.108 LucamSaveImageW

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

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☐	LabVIEW	☐	MATLAB	☐

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 Photographic Experts Group (.jpg), Tagged Image File Format (.tif) and Raw (.raw).
The available pixel formats are listed in Section 7.1.2.
If an unsupported file type (indicated by its extension) is provided, the function will fail.

5.109 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.

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☐	LabVIEW	☐	MATLAB	☐

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 Photographic Experts Group (.jpg), Tagged Image File Format (.tif) and Raw (.raw).
The available pixel formats are listed in Section 7.1.2.
If an unsupported file type (indicated by its extension) is provided, the function will fail.

5.110 LucamSelectExternInterface

Selects external interface to use to connect with the camera. Refer to section 7.1.7 for supported interface.

Platform support

Windows ☒	Linux ☐	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☒

Usage

```
BOOL LucamSelectExternInterface( ULONG externInterface);
```

Parameters

ExternInterface	[in] Extern Interface to connect with camera as described in section 7.1.7.

Return Values

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

Remarks

To inquire which interface is being used by the associated camera, us the LucamQueryExternInterface() function.

5.111 LucamSequencingCancelTakeSequence

Stop active LucamSequengingTakeSequence().

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☐	LabVIEW	☐	MATLAB	☐

Usage

```
BOOL LucamSequencingCancelTakeSequence(HANDLE hCamera)
```

Parameters

hCamera	Handle of the camera that is being use by trigger sequencing.

Return Values

Return TRUE if the execution of the function is successful. A returned value of FALSE indicates that the execution of the function failed and the error code can be queried by using LucamGetLastErrorForCamera() or LucamGetLastError().

See Also

LucamCancelTakeFastFrame(), LucamGetLastErrorForCamera(), LucamGetLastError()

Remarks

The camera models limit the support of trigger sequencing.

5.112 LucamSequencingGetIndexForFrame

Get the index of a frame in the trigger sequence.

Platform support

Windows	☒	Linux	☒	Mac	☐

Language Support

C++	☒	C# / VB .NET	☐	LabVIEW	☐	MATLAB	☐

Usage

```
LONG LucamSequencingGetIndexForFrmae( HANDLE hCamera
                , BYTE *pFrame);
```

Parameters

hCamera	Handle of the camera that is being use by trigger sequencing.
pFrames	Address of the frame to find the index.

Return Values

The function returns the index of the frame. The function returns a value of -1 if it was not able to find the index for the frame. The function LucamGetLastErrorForCamera() or LucamGetLastError() can be used to get the camera error code.

See Also

LucamGetLastErrorForCamera(), LucamGetLastError().

Remarks

The camera models limit the support of trigger sequencing.

5.113 LucamSequencingGetStatus

Query the camera for the trigger-sequencing status.

Platform support

Windows	☒	Linux	☒	Mac	☐

Language Support

C++	☒	C# / VB .NET	☐	LabVIEW	☐	MATLAB	☐

Usage

```
BOOL LucamSequencingGetStatus(HANDLE hCamera
        , LUCAM_SEQUENCING_STATUS *pStatus);
```

Parameters

hCamera	Handle of the camera that is being use by trigger sequencing.
pStatus	Address of the Lumenera sequencing status data structure to fill.

Return Values

Return TRUE if the execution of the function is successful. A returned value of FALSE indicates that the execution of the function failed and the error code can be queried by using LucamGetLastErrorForCamera() or LucamGetLastError().

See Also

LucamGetLastErrorForCamera(), LucamGetLastError()

Remarks

The camera models limit the support of trigger sequencing.

5.114 LucamSequencingTakeSequence

Execute the trigger-sequence to capture images.

Platform support

Windows ☒	Linux ☒	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☐	LabVIEW ☐	MATLAB ☐

Usage

```
BOOL LucamSequencingTakeSequence(HANDLE hCamera
        , ULONG frameCount
        , BYTE *pFrames[]
        , ULONG expectedInitialSequenceIndex=0);
```

Parameters

hCamera	Handle of the camera that is being use by trigger sequencing.
frameCount	Number of images in the array of image pointers.
pFrames[]	Address of the array of image pointers.
expectedInitialSequenceIndex	The index where the sequence should start.

Return Values

Return TRUE if the execution of the function is successful. A returned value of FALSE indicates that the execution of the function failed and the error code can be queried by using LucamGetLastErrorForCamera() or LucamGetLastError().

See Also

LucamGetLastErrorForCamera(), LucamGetLastError()

Remarks

The camera models limit the support of trigger sequencing.

5.115 LucamSequencingSetup

Initialise the trigger sequencing.

Platform support

Windows	☒	Linux	☒	Mac	☐

Language Support

C++	☒	C# / VB .NET	☐	LabVIEW	☐	MATLAB	☐

Usage

```
BOOL LucamSequencingSetup(HANDLE hCamera
        , ULONG framesPerSequence
        , ULONG settingsCount
        , LUCAM_SEQUENCE_SETTING pSettings[]
        , LUCAM_SEQUENCING_CONTROL code);
```

Parameters

hCamera	Handle of the camera that is being use by trigger sequencing.
framePerSequence	The number of images that the sequence will capture.
settingsCount	The number of sequence settings in the settings array.
code	The flags to control the sequence mode.

Return Values

Return TRUE if the execution of the function is successful. A returned value of FALSE indicates that the execution of the function failed and the error code can be queried by using LucamGetLastErrorForCamera() or LucamGetLastError().

See Also

LucamGetLastErrorForCamera(), LucamGetLastError()

Remarks

The camera models limit the support of trigger sequencing.

5.116 LucamSetFormat

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

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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 7.2.2.
Each dimension of the subwindow must be evenly divisible by eight.

5.117 LucamSetProperty

Sets the value of the specified camera property.

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☒

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 **Error! Reference source not found.**. Not all roperties are supported by all cameras. If a property is unsupported, the function will return a failed condition (FALSE). The allowable capability flags are listed in Section 7.1.1. If a capability flag is not supported by the property, it is silently ignored.

5.118 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. The timeout value is expressed in milliseconds and represents the maximum time to wait for an image prior to returning to the calling thread. A good starting point is to use 500ms plus exposure time. This value can then be tuned for a particular camera model and number of cameras used at the same time.

Platform support

Windows ☒	Linux ☒	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☐	LabVIEW ☐	MATLAB ☐

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.

5.119 LucamSetTimestamp

Assign value to the current timestamp on camera.

Platform support

Windows	☒	Linux	☒	Mac	☐

Language Support

C++	☒	C# / VB .NET	☐	LabVIEW	☐	MATLAB	☐

Usage

```
BOOL LucamSetTimestamp(
                HANDLE hCamera,
                ULONGLONG timestamp);
```

Parameters

hCamera	[in] handles of the camera
timestamp	[in] new timestamp value.

Return Values

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

Remarks

- Time stamp function is not supported by all camera models.
- The actual value can only be set to zero for now. This would allow to sync with host time.

See Also

LucamEnableTimestamp(), LucamGetMetadata(), LucamGetTimestamp(), LucamGetTimestampFrequency(), LucamIsTimestampEnable(), LucamSetTimestamp().

5.120 LucamSetTriggerMode

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

Platform support

Windows ☒	Linux ☐	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☐	LabVIEW ☐	MATLAB ☐

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.

5.121 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 one or many color channels by setting the appropriate parameters.

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☐	LabVIEW	☐	MATLAB ☐

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 zero to disable it or 256 to enable it.

5.122 LucamSetup8bitsLUT

Populates the 8-bit LUT inside the camera.

Platform support

Windows ☒	Linux ☐	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☐

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 zero to disable it or 256 to enable it.

5.123 LucamSetupCustomMatrix

Uploading the color correction matrix to use when converting raw data to color (RGB24, RGB32 and RGB48). In order to use the uploaded custom CCM, the LUCAM_PROP_CORRRECTION_MATRIX needs to be set to LUCAM_CM_CUSTOM. See also **LucamGetCurrentMatrix()**.

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

Usage

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

Parameters

hCamera	[in] handle to the camera
*pMatrix	[in] 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 array of coefficients contains nine float elements and represents a 3X3 matrix.
- When using a conversion function, the color correction matrix parameter is required. 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.

5.124 LucamStreamVideoControl

Controls the streaming video.

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☐

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.

5.125 LucamStreamVideoControlAVI

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

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☒	LabVIEW	☐	MATLAB	☐

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.

5.126 LucamTakeFastFrame

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

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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.

5.127 LucamTakeFastFrameNoTrigger

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

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☐

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 a new snapshot request and waiting for the next snapshot.

5.128 LucamTakeSnapshot

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

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☒

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 7.2.1. 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()*

5.129 LucamTakeSynchronousSnapshots

Simultaneously takes a single image from each of several cameras.

Platform support

Windows ☒	Linux ☐	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☐

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.

5.130 LucamTakeVideo

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

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☒	MATLAB ☒

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.

5.131 LucamTakeVideoEx

Capture a video frame from active video stream with connected camera.

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☐

Usage

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

Parameters

hCamera	[in] handle to the camera
*pData	[out] video data returned from the camera
*pLength	[out] number of bytes copied in 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

In case of failure, use LucamGetLastErrorForCamera(handle) to get camera error code.

5.132 LucamTriggerFastFrame

Initiates the request to take a snapshot.

Platform support

Windows ☒	Linux ☒	Mac ☒

Language Support

C++ ☒	C# / VB .NET ☒	LabVIEW ☐	MATLAB ☐

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.

5.133 LucamUnRegisterCallbackNotification

Unregister an event callback notification.

Platform support

Windows	☐	Linux	☐	Mac	☒

Language Support

C++	☒	C# / VB .NET	☐	LabVIEW	☐	MATLAB	☐

Usage

```
LONG LucamUnRegisterCallbackNotification(HANDLE hCamera,
                LONG callbackID);
```

Parameters

hCamera	[in] handle to the camera
callbackID	[LONG] callback identification number.

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.

5.134 LucamUnregisterEventNotification

Deregisters an event handle with the LuCam API.

Platform support

Windows ☒	Linux ☐	Mac ☐

Language Support

C++ ☒	C# / VB .NET ☐	LabVIEW ☐	MATLAB ☐

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.

5.135 LucamWriteRegister

Writes the internal camera registers.

Platform support

Windows	☒	Linux	☐	Mac	☐

Language Support

C++	☒	C# / VB .NET	☐	LabVIEW	☐	MATLAB	☐

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.

6 API Properties

This section covers the properties that may be supported by a Lumenera camera. Most properties can be read from or written to the camera. Some properties can be programmed to be continuously adjusted by the camera and this can be determined by using the property flags. It is possible to query the camera to detect whether a property is or is not supported. This is normally done by reading the desired property and checking if the function returns successfully or not. If unsuccessful then just check the error code to confirm that property is not currently supported.

6.1 LUCAM_PROP_ABS_FOCUS

This property is used to control the focus of a lens. This property is only available on cameras that have a Canon EF lens interface.

Platform support

Windows	☒	Linux	☐	Mac	☒

6.2 LUCAM_PROP_AUTO_EXP_MAXIMUM

Set or get the maximum exposure value to be used with the continuous auto exposure algorithm.

Platform support

Windows	☒	Linux	☒	Mac	☒

6.3 LUCAM_PROP_AUTO_EXP_TARGET

Set or read the intensity target [0:255] value for the continuous auto exposure to reach.

Platform support

Windows	☒	Linux	☒	Mac	☒

6.4 LUCAM_PROP_AUTO_GAIN_MINIMUM

Set or get the minimum possible gain adjustment for the continuous auto gain mode.

Platform support

Windows	☒	Linux	☒	Mac	☐

6.5 LUCAM_PROP_AUTO_GAIN_MAXIMUM

Set or get the maximum possible gain adjustment for the continuous auto gain mode.

Platform support

Windows	☒	Linux	☒	Mac	☐

6.6 LUCAM_PROP_AUTO_IRIS_MAX

Set or get the maximum possible iris value when in continuous auto iris mode. This mode requires a Canon EF or compatible lens.

Platform support

Windows	☒	Linux	☐	Mac	☒

6.7 LUCAM_PROP_BLACK_LEVEL

This property controls the level of brightness at the darkest part of an image. Lumenera optimizes this analog offset of their cameras in the manufacturing process. However, access to this property has been enabled on some camera models.

Platform support

Windows	☒	Linux	☐	Mac	☒

6.8 LUCAM_PROP_BRIGHTNESS

Property that digitally changes the intensity of an image. In most cases, the property has a range of -100 to 100.

Platform support

Windows	☒	Linux	☒	Mac	☒

6.9 LUCAM_PROP_CONTRAST

Contrast is the difference in luminance that makes an object distinguishable. In most cases, the property has a range of 0-100.

Platform support

Windows	☒	Linux	☒	Mac	☒

6.10 LUCAM_PROP_CORRECTION_MATRIX

The color correction matrix property is used to optimize the color balance for a particular type of light. Possible types of light supported by a camera are defined in section 7.1.5

Platform support

Windows ☒	Linux ☒	Mac ☒

6.11 LUCAM_PROP_COLOR_FORMAT

This read-only property will provide color or monochrome status of your camera. If a color camera, then it will also give information on the Bayer filter used by the camera sensor. The possible color format is defined in section 7.1.6. Using the properties flags, it is also possible to determine the 16 bit data alignment by making the LUCAM_PROP_FLAG_LITTLE_ENDIAN bit (see section 7.1.1).

Platform support

Windows ☒	Linux ☒	Mac ☒

Possible values

LUCAM_CF MONO	Monochrome camera.
LUCAM_CF_BAYER_RGGB	RGGB Bayer filter.
LUCAM_CF_BAYER_GRBG	GRBG Bayer filter.
LUCAM_CF_BAYER_GBRG	GBRG Bayer filter.
LUCAM_CF_BAYER_BGGR	BGGR Bayer filter.

6.12 LUCAM_PROP_DEMOSAICING_METHOD

Algorithm used to reconstruct a full color image from the video stream raw data. Possible methods are defined in section 7.1.4

Platform support

Windows ☒	Linux ☒	Mac ☒

6.13 LUCAM_PROP_DIGITAL_GAIN

This property is mainly used to fine-tune the color balance. Most of the time this property will have a valid range from 0 to 2.

Platform support

Windows ☒	Linux ☒	Mac ☒

6.14 LUCAM_PROP_DIGITAL_GAIN_BLUE

This property is mainly used to fine tune the color balance by adjusting the blue channel. Most of the time this property will have a valid range from 0 to 2.5.

Platform support

Windows ☒	Linux ☒	Mac ☒

6.15 LUCAM_PROP_DIGITAL_GAIN_GREEN

This property is mainly used to fine-tune the color balance by adjusting the green channel. Most of the time this property will have a valid range from 0 to 2.5.

Platform support

Windows ☒	Linux ☒	Mac ☒

6.16 LUCAM_PROP_DIGITAL_GAIN_RED

This property is mainly used to fine-tune the color balance by adjusting the red channel. Usually, this property will have a valid range from 0 to 2.5.

Platform support

Windows ☒	Linux ☒	Mac ☒

6.17 LUCAM_PROP_DIGITAL_WHITEBALANCE_U

Fine tuning color balance adjustment on the U chromatic channel. The values will usually be between -100 to 100.

Platform support

Windows ☒	Linux ☒	Mac ☒

Property Definition

LUCAM_PROP_DIGITAL_WHITEBALANCE_U

6.18 LUCAM_PROP_DIGITAL_WHITEBALANCE_V

Fine tuning color balance adjustment on the V chromatic channel. The values will usually be between -100 to 100.

Platform support

Windows ☒	Linux ☒	Mac ☒

6.19 LUCAM_PROP_EXPOSURE

The time used by the sensor to accumulate light. This is normally expressed in milliseconds (ms) and range will vary between camera models.

Platform support

Windows	☒	Linux	☒	Mac	☒

6.20 LUCAM_PROP_EXPOSURE_INTERVAL

Get the interval between exposures when camera is working in burst mode (multi frame capture).

Platform support

Windows	☒	Linux	☐	Mac	☐

6.21 LUCAM_PROP_FAN

Enable or disable cooling fan on camera.

Platform support

Windows	☒	Linux	☐	Mac	☐

6.22 LUCAM_PROP_FLIPPING

Most camera models allow flipping and/or mirror data as the raw frame is converted to RGB. This property is used to select the desired orientation of the video stream images.

Platform support

Windows	☒	Linux	☒	Mac	☒

Possible values

LUCAM_PROP_FLIPPING_NONE	no flipping and no mirroring.
LUCAM_PROP_FLIPPING_X	mirroring only.
LUCAM_PROP_FLIPPING_Y	flipping only.
LUCAM_PROP_FLIPPING_XY	flipping and mirroring.

6.23 LUCAM_PROP_CAMERA_FLIPPING

A selection of camera models allow flipping and/or mirror data onboard the camera. Using these property flags on a supported camera will directly impact the orientation of the Raw data.

Platform support

Windows	☒	Linux	☒	Mac	☒

Possible values

LUCAM_PROP_CAMERA_FLIPPING_NONE	no flipping and no mirroring.
LUCAM_PROP_ CAMERA_FLIPPING_X	mirroring only.
LUCAM_PROP_ CAMERA_FLIPPING_Y	flipping only.

LUCAM_PROP_ CAMERA_FLIPPING_XY	flipping and mirroring.

6.24 LUCAM_PROP_FOCAL_LENGTH

Get focal length of lens. This is only available on cameras that have Canon EF or compatible lenses.

Platform support

Windows	☒	Linux	☒	Mac	☐

6.25 LUCAM_PROP_FOCUS

Control the focus position of a lens. This property is only available on cameras with a Canon EF or compatible lens.

Platform support

Windows	☒	Linux	☒	Mac	☒

6.26 LUCAM_PROP_GAIN

Analog amplification applied before the analog to digital conversion.

Platform support

Windows	☒	Linux	☒	Mac	☒

6.27 LUCAM_PROP_GAIN_BLUE

The blue color channel analog amplification applied before the analog to digital conversion.

Platform support

Windows	☒	Linux	☒	Mac	☒

6.28 LUCAM_PROP_GAIN_GREEN1

The green1 color channel analog amplification applied before the analog to digital conversion.

Platform support

Windows	☒	Linux	☒	Mac	☒

6.29 LUCAM_PROP_GAIN_GREEN2

The green2 color channel analog amplification applied before the analog to digital conversion.

Platform support

Windows	☒	Linux	☒	Mac	☒

6.30 LUCAM_PROP_GAIN_RED

The red color channel analog amplification applied before the analog to digital conversion.

Platform support

Windows	☒	Linux	☒	Mac	☒

6.31 LUCAM_PROP_GAMMA

Gamma correction is a non-linear operation used to code and decode luminance in an image from the camera.

Platform support

Windows	☒	Linux	☒	Mac	☒

6.32 LUCAM_PROP_GEV_IPCONFIG_DHCP

Get or set DHCP configuration value for GIGE cameras.

Platform support

Windows	☒	Linux	☐	Mac	☐

6.33 LUCAM_PROP_GEV_IPCONFIG_LLA

Get local link address value for GIGE cameras.

Platform support

Windows	☒	Linux	☐	Mac	☐

6.34 LUCAM_PROP_GEV_IPCONFIG_PERSISTENT

Get persistent flag for GIGE cameras.

Platform support

Windows	☒	Linux	☐	Mac	☐

6.35 LUCAM_PROP_GEV_IPCONFIG_PERSISTENT_DEFAULTGATEWAY

Get or set gateway ip address for persistent addressing on GigE cameras.

Platform support

Windows	☒	Linux	☐	Mac	☐

6.36 LUCAM_PROP_GEV_IPCONFIG_PERSISTENT_IPADDRESS

Get or set persistent ip address for persistent addressing on GigE cameras.

Platform support

Windows	☒	Linux	☐	Mac	☐

6.37 LUCAM_PROP_GEV_IPCONFIG_PERSISTENT_SUBNETMASK

Get or set persistent subnet mask for GigE cameras.

Platform support

Windows	☒	Linux	☐	Mac	☐

6.38 LUCAM_PROP_GEV_SCPD

Get or set packet resent delay. This is used for packet synchronisation.

Platform support

Windows	☒	Linux	☐	Mac	☐

6.39 LUCAM_PROP_HOST_AUTO_EX_ALGORITHM

Get or set processing method for the auto exposure evaluation.

Platform support

Windows	☒	Linux	☐	Mac	☐

Valid values:

- AUTO_ALGORITHM_SIMPLE_AVERAGING
- AUTO_ALGORITHM_HISTOGRAM (default)
- AUTO_ALGORITHM_MACROBLOCKS

6.40 LUCAM_PROP_HOST_AUTO_WB_ALGORITHM

Get or set processing method for the white balance adjustments..

Platform support

Windows	☒	Linux	☐	Mac	☐

Valid values:

- AUTO_ALGORITHM_SIMPLE_AVERAGING
- AUTO_ALGORITHM_HISTOGRAM (default)
- AUTO_ALGORITHM_MACROBLOCKS

6.41 LUCAM_PROP_HUE

The attribute of a color by virtue of which it is discernible as red, green etc., and which is dependent on its dominant wavelength. Also referred to as color, shade, tint, tone. The expected property range is -180 to 180.
On Mac is also called PROP_DIGITAL_HUE, however it is recommended to use LUCAM_PROP_HUE

Platform support

Windows	☒	Linux	☒	Mac	☒

6.42 LUCAM_PROP_IRIS

Control the aperture of a lens. This property is only available on a camera with a Canon EF or equivalent lens and Camera models that support P-IRIS lens. When using P-IRIS lens it is user's responsibility to properly initialise the LUCAM_PROP_IRIS_STEPS_COUNT.

Platform support

Windows	☒	Linux	☒	Mac	☒

Remarks

- F-Stop relationship for Canon EF
  Iris = 100/F-Stop^2 or F-Stop=10/sqrt(Iris)
- F-Stop relationship for P-IRIS Lens { Lt345, Lt545, Lt945, Lt1245}.
  Iris = 10000/F-Stop^2 or F-Stop = 10/sqrt(Iris)
- P-IRIS also support auto iris functions.

6.43 LUCAM_PROP_IRIS_LATENCY

Get the latency of the iris in milliseconds. How long it takes the iris to reach the desired aperture opening for a snapshot.

Platform support

Windows	☒	Linux	☐	Mac	☐

6.44 LUCAM_PROP_IRIS_STEPS_COUNT

This property defines the number of steps of the P-IRIS lens. This value has been default to 73 , which seems a popular value for P-IRIS lens. Users will have to check the P-IRIS lens datasheet to confirm the number of steps for their lens and update this property. This property is only supported by cameras that has support for P-IRIS {Lt345, Lt545, Lt945, Lt1245}.

Platform support

Windows	☒	Linux	☒	Mac	☐

6.45 LUCAM_PROP_JPEG_QUALITY

Set or get the JPEG quality factor to use when saving to file. The higher the quality factor the better image quality and the larger the file storage size.

Platform support

Windows	☒	Linux	☐	Mac	☒

6.46 LUCAM_PROP_KNEE1_EXPOSURE

Control first knee point exposure value for high dynamic range described in section 7.1.18 for video mode.

Platform support

Windows	☒	Linux	☒	Mac	☒

6.47 LUCAM_PROP_KNEE2_EXPOSURE

Control second knee point exposure value for high dynamic range describe in section 7.1.18 for video mode.

Platform support

Windows	☒	Linux	☒	Mac	☒

6.48 LUCAM_PROP_KNEE1_LEVEL

Control first Knee point intensity level for high dynamic range describe in section 7.1.18 for video and snapshot mode.

Platform support

Windows	☒	Linux	☒	Mac	☒

6.49 LUCAM_PROP_KNEE2_LEVEL

Control second Knee point intensity level for high dynamic range describe in section 7.1.18 for video and snapshot mode.

Platform support

Windows	☒	Linux	☒	Mac	☒

6.50 LUCAM_PROP_LIGHT_FREQUENCY

Set or get value for light period compensation (ex: (1/60hz)/2). This is to align start of exposure to be constant on light frequency to avoid intensity flickering. This property is express in milliseconds.

Platform support

Windows	☒	Linux	☐	Mac	☐

6.51 LUCAM_PROP_LSC_X

Set or get value for the lens shading correction in the X-axis.

Platform support

Windows	☒	Linux	☒	Mac	☐

6.52 LUCAM_PROP_LSC_Y

Set or get value for the lens shading correction in the Y-axis.

Platform support

Windows	☒	Linux	☒	Mac	☐

6.53 LUCAM_PROP_MAX_FRAME_RATE

The frame rate response of the camera is proportionally inverted to the exposure. Therefore, if exposure is increasing, frame rate will reduce and vice versa if exposure is decreasing frame rate will increase. This property enables control on the frame rate of the camera when streaming in video mode. So if exposure is reduce in a way that frame rate would exceed the maximum frame rate then the camera would limit the frame rate. In the other hand, if the exposure is increasing then frame rate will reduce due to exposure centric camera. By default the property is set to infinity (-1) .

Platform support

Windows	☒	Linux	☐	Mac	☐

6.54 LUCAM_PROP_MAX_WIDTH

This read only property will provide the maximum width the camera sensor can output.

Platform support

Windows	☒	Linux	☒	Mac	☒

6.55 LUCAM_PROP_MAX_HEIGHT

This read only property will provide the maximum height the camera sensor can output.

Platform support

Windows	☒	Linux	☒	Mac	☒

6.56 LUCAM_PROP_MEMORY

Get number of frames in camera memory.

Platform support

Windows	☒	Linux	☐	Mac	☐

6.57 LUCAM_PROP_SATURATION

Saturation defines a range from pure color to gray at a constant lightness level. A pure color is fully saturated. In most cases, saturation property ranges if from 0 to 2.

On Mac is also called PROP_DIGITAL_SATURATION, however it is recommended to use LUCAM_PROP_ SATURATION

Platform support

Windows	☒	Linux	☒	Mac	☒

6.58 LUCAM_PROP_SHARPNESS

The amount of detail the camera can reproduce. Most cameras do not support this property.

Platform support

Windows	☒	Linux	☐	Mac	☐

6.59 LUCAM_PROP_SNAPSHOT_CLOCK_SPEED

Get or set the snapshot clock speed. A value of 0 represents the fastest clock. The slower the clock is , the less read noise there is on the image. Each camera will have different range and could be ready by using LucamPropertyRange describe in section 5.95.

Platform support

Windows ☒	Linux ☒	Mac ☒

6.60 LUCAM_PROP_SNAPSHOT_COUNT

Get or set number of frames to capture per trigger. (Burst mode).

Platform support

Windows ☒	Linux ☒	Mac ☒

6.61 LUCAM_PROP_STILL_EXPOSURE

Time that sensor is exposed to light in snapshot mode. This is normally expressed in milliseconds (ms) and range will vary between camera models.

Platform support

Windows ☒	Linux ☒	Mac ☒

6.62 LUCAM_PROP_STILL_EXPOSURE_DELAY

Set exposure delay in milliseconds for snapshot timing synchronisation.

Platform support

Windows ☒	Linux ☒	Mac ☒

6.63 LUCAM_PROP_STILL_GAIN

Analog amplification applied before the analog to digital conversion on snapshot image.

Platform support

Windows ☒	Linux ☒	Mac ☒

6.64 LUCAM_PROP_STILL_GAIN_BLUE

The blue color channel analog amplification applied before the analog to digital conversion.

Platform support

Windows ☒	Linux ☒	Mac ☒

6.65 LUCAM_PROP_STILL_GAIN_GREEN1

The green1 color channel analog amplification applied before the analog to digital conversion.

Platform support

Windows ☒	Linux ☒	Mac ☒

6.66 LUCAM_PROP_STILL_GAIN_GREEN2

The green2 color channel analog amplification applied before the analog to digital conversion.

Platform support

Windows ☒	Linux ☒	Mac ☒

6.67 LUCAM_PROP_STILL_GAIN_RED

The red color channel analog amplification applied before the analog to digital conversion.

Platform support

Windows ☒	Linux ☒	Mac ☒

6.68 LUCAM_PROP_STILL_KNEE1_EXPOSURE

Control first knee point exposure value for high dynamic range described in section 7.1.18 for snapshot mode.

Platform support

Windows ☒	Linux ☒	Mac ☒

6.69 LUCAM_PROP_STILL_KNEE2_EXPOSURE

Control second knee point exposure value for high dynamic range describe in section 7.1.184 for snapshot mode.

Platform support

Windows ☒	Linux ☒	Mac ☒

6.70 LUCAM_PROP_STILL_KNEE3_EXPOSURE

Control first knee point exposure value for high dynamic range describe in section 7.1.18 for snapshot mode.

Platform support

Windows ☒	Linux ☒	Mac ☒

6.71 LUCAM_PROP_STILL_STROBE_DELAY

Set the strobe delay for snapshot synchronisation.

Platform support

Windows ☒	Linux ☒	Mac ☒

6.72 LUCAM_PROP_STILL_STROBE_DURATION

Set or get the strobe pulse width in milliseconds.

Platform support

Windows ☒	Linux ☒	Mac ☒

6.73 LUCAM_PROP_STILL_TAP_CONFIGURATION

Get or set snapshot tap configuration. This property will be supported on cameras that have tap support.

Platform support

Windows ☒	Linux ☒	Mac ☒

Possible values

TAP_CONFIGURATION_SINGLE: single tap configuration.
TAP_CONFIGURATION_DUAL: Dual tap configuration.
TAP_CONFIGURATION_QUAD: Quad tap configuration.

6.74 LUCAM_PROP_STROBE_PIN

Get or set pin where the strobe output signal will be generated. This property is also use to set the polarity of the signal trough the LUCAM_PROP_FLAG_POLARITY. By default, an active high strobe pulse will be generated.

Platform support

Windows ☒	Linux ☒	Mac ☒

Example

```
LucamSetProperty(handle,LUCAM_PROP_STROBE_PIN, 3, 0); // configure GPIO3 as the strobe pin
```

```
                                                    // Configure strobe to be low when active.
LucamSetProperty(handle,LUCAM_PROP_STROBE_PIN, 3, LUCAM_PROP_FLAG_POLARITY);
```

6.75 LUCAM_PROP_SYNC_MODE

Set the synchronisation mode between cameras in snapshot mode.

Platform support

Windows	☒	Linux	☐	Mac	☐

6.76 LUCAM_PROP_TAP_CONFIGURATION

Get or set video tap configuration. This property will be supported on cameras that have tap support. It is also important that video stream should be stopped before changing tap configuration.

Platform support

Windows	☒	Linux	☒	Mac	☒

Possible values

TAP_CONFIGURATION_SINGLE: single tap configuration.
TAP_CONFIGURATION_DUAL: Dual tap configuration.
TAP_CONFIGURATION_QUAD: Quad tap configuration.

6.77 LUCAM_PROP_TEMPERATURE

Get internal temperature and set target temperature for camera that has cooling capability.

Platform support

Windows	☒	Linux	☐	Mac	☒

6.78 LUCAM_PROP_TEMPERATURE2

Get image sensor temperature (few camera model support this property).

Platform support

Windows	☒	Linux	☐	Mac	☐

6.79 LUCAM_PROP_THRESHOLD

Enable or disable thresholding on cameras that support this function.

Platform support

Windows	☒	Linux	☐	Mac	☒

6.80 LUCAM_PROP_THRESHOLD_HIGH

Get or set the intensity of the high threshold point.

Platform support

Windows	☒	Linux	☐	Mac	☐

6.81 LUCAM_PROP_THRESHOLD_LOW

Get or set the intensity of the low threshold point.

Platform support

Windows	☒	Linux	☐	Mac	☐

6.82 LUCAM_PROP_TIMESTAMPS

Enable or disable frame count. When enabled, a 16-bit counter will be placed in the first 2 bytes of the raw image.

Platform support

Windows	☒	Linux	☒	Mac	☒

6.83 LUCAM_PROP_TRIGGER_MODE

Select desired snapshot trigger mode.

Platform support

Windows	☒	Linux	☒	Mac	☐

Support values

TRIGGER_MODE_NORMAL: this is the default operating mode of the camera and trigger will be generated on the rising edge or falling edge depending on the polarity of the signal.

TRIGGER_MODE_BULB: This is the standard bulb mode of any dSLR consumer product where the camera will expose as long as trigger signal is active. Bulb exposure cannot be used while camera is in burst mode. Please note that bulb exposure is support by USB3 product only.

Remarks

There are some limitations while working with trigger mode set to TRIGGER_MODE_BULB:

- BULB exposure is not meant to be used with software triggers. If a bulb exposure snapshot is software triggered, the resulting frame will have a very low exposure.

- BULB exposure is not meant to be used with burst mode. If used with "busted" mode, extra frames will have nearly 0 exposures.
- Fast frames mode may hang and need to be disabled and re-enabled if bulb exposure mode is turned on/off while a trigger is active or a frame is exposing or reading out. It is safe to turn on or off bulb mode between frames, while the trigger line is inactive.

6.84 LUCAM_PROP_TRIGGER_PIN

Get or set pin where the trigger input signal will be taken from. This property is also used to set the polarity of the signal through the LUCAM_PROP_FLAG_POLARITY. By default, the trigger will be activated on the rising edge of the trigger signal.

Platform support

Windows ☒	Linux ☒	Mac ☒

Example

```
LucamSetProperty(handle,LUCAM_PROP_TRIGGER_PIN, 2, 0); // configure GPIO2 as the trigger pin
                                                       // Configure pin to trigger on falling edge.
LucamSetProperty(handle,LUCAM_PROP_TRIGGER_PIN, 2, LUCAM_PROP_FLAG_POLARITY);
```

6.85 LUCAM_PROP_UNIT_HEIGHT

This read only property defines the multiple of any values on the Y-axis. So any Y offset or a height value has to be to multiple of Unit Height to be valid.

Platform support

Windows ☒	Linux ☒	Mac ☒

6.86 LUCAM_PROP_UNIT_WIDTH

This read only property defines the multiple of any values on the X-axis. So any X offset or width values has to be to multiple of Unit Width to be valid.

Platform support

Windows ☒	Linux ☒	Mac ☒

6.87 LUCAM_PROP_VIDEO_CLOCK_SPEED

Get or set the video clock speed. A value of 0 represents the fastest clock. The slower the clock is , the less read noise there is on the image. Each camera will have different range and could be read by using LucamPropertyRange describe in section 5.95.

Platform support

Windows	☒	Linux	☒	Mac	☐

7 Constants and Structures Descriptions

7.1 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.

7.1.1 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_BUSY	
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 or LUCAM_PROP_STROBE that we are changing the signal 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 sub sampling.
LUCAM_PROP_GEV_IPCONFIG_LLA	Identifies the usage of Link Local Address flags.
LUCAM_PROP_GEV_IPCONFIG_DHCP	Identifies the usage of DHCP for the IP assignment.
LUCAM_PROP_GEV_IPCONFIG_PERSISTENT	Identifies the usage of persistent IP address.
LUCAM_FRAME_FORMAT_FLAGS_BINNING	Identifies the frame format is in binning mode or not and use with get or set frame format functions.
LUCAM_PROP_FLAG_RED	This flags can be used with GAMMA, BRIGHTNESS, CONTRAST (specific camera models only).
LUCAM_PROP_FLAG_GREEN1	This flags can be used with GAMMA, BRIGHTNESS, CONTRAST (specific camera models only).
LUCAM_PROP_FLAG_GREEN2	This flags can be used with GAMMA, BRIGHTNESS, CONTRAST (specific camera models only).

LUCAM_PROP_FLAG_BLUE	This flags can be used with GAMMA, BRIGHTNESS, CONTRAST (specific camera models only).
LUCAM_PROP_FLAG_SEQUENCABLE	Indicates that the property can be use by the trigger-sequencing feature.
LUCAM_PROP_FLAG_UNKNOWN_MAXIMUM	When lens is not initialized, so no focus maximum.
LUCAM_PROP_FLAG_UNKNOWN_MINIMUM	When lens is not initialized, so no focus maximum.
LUCAM_PROP_FLAG_MEMORY_READBACK	Use with LUCAM_PROP_MEMORY
LUCAM_PROP_FLAG_BACKLASH_COMPENSATION	Use with LUCAM_PROP_FOCUS and LUCAM_PROP_IRIS
LUCAM_PROP_FLAG_HW_ENABLE	Use with VIDEO_TRIGGER, also need the LUCAM_PROP_FLAG_USE
LUCAM_PROP_FLAG_SW_TRIGGER	Use with VIDEO_TRIGGER property, the LUCAM_PROP_FLAG_USE is also required when using this flag.
LUCAM_PROP_FLAG_MASTER	Use with LUCAM_PROP_SYNC_MODE property.

7.1.2 **Auto Algorithm**

The algorithm for white balance and for the auto exposure operations can be selected by writing the algorithm value to the LUCAM_PROP_HOST_AUTO_WB_ALGORIGHTM or LUCAM_PROP_HOST_AUTO_EX_ALGORITHM.

AUTO_ALGORITHM_SIMPLE_AVERAGING	Averaging pixel values
AUTO_ALGORITHM_HISTOGRAM (default)	Align histogram channel spike together
AUTO_ALGORITHM_MACROBLOCKS	Block matching algorithm on multiple frame.

7.1.3 **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.

7.1.4 **Demosaicing 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 demosaicing method is used.
LUCAM_DM_FAST:	Identifies to use the fast demosaicing method. This method can be completed quickly but does not provide high quality images.
LUCAM_DM_HIGH_QUALITY:	Identifies to use the high quality demosaicing method. This method provides high quality images at a medium rate of speed.

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

7.1.5 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_LED:	Identifies to use the correction matrix to correct for LED lighting.
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.

7.1.6 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.

7.1.7 External Interfaces

These values state which external 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.
LUCAM_EXTERN_INTERFACE_USB3:	identifies that the camera is connected to a USB 3.0 interface.
LUCAM_EXTERN_INTERFACE_GIGEVISION:	Identifies that the camera is connected to a GigE Interface.

7.1.8 Shutter Types

These flags state that 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.

7.1.9 Trigger Modes

These values state witch hardware trigger mode to use by writing to LUCAM_PROP_TRIGGER_MODE property.

TRIGGER_MODE_NORMAL: Select default trigger mode. Camera will trigger on rising or falling edge depending of polarity.

TRIGGER_MODE_BULB: Exposure length is control by the hardware trigger length.

There are some limitations while working with trigger mode set to TRIGGER_MODE_BULB:

- BULB exposure is not meant to be used with software triggers. If a bulb exposure snapshot is software triggered, the resulting frame will have a very low exposure.
- BULB exposure is not meant to be used with burst mode. If used with "busted" mode, extra frames will have nearly 0 exposures.
- Fast frames mode may hang and need to be disabled and re-enabled if bulb exposure mode is turned on/off while a trigger is active or a frame is exposing or reading out. It is safe to turn on or off bulb mode between frames, while the trigger line is inactive.

7.1.10 **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 effect until the raw image data is converted to color through the LucamConvertFrameToRgbXX() functions or to greyscale through the LucamConvertFrameToGreyscaleXX() 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.

Certain camera models support flipping the image data in the camera. For those cameras, use the following flags to control the flipping mode of the raw frames. The use of these property flags will not influence images being converted from raw to RGB.

LUCAM_PROP_CAMERA_ FLIPPING_NONE:	Sets the camera to not use flipping.
LUCAM_PROP_CAMERA _FLIPPING_X:	Controls the flipping in the X direction (mirror) only.
LUCAM_PROP_CAMERA _FLIPPING_Y:	Controls the flipping in the Y direction only.
LUCAM_PROP_CAMERA _FLIPPING_XY:	Controls the flipping in the X and Y directions.

7.1.11 **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:	Pause the video stream or video preview.
START_RGBSTREAM:	Start streaming RGB data.

7.1.12 **AVI video preview controls**

These flags control the state of the AVI playback.

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

7.1.13 **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_8:	Defines that the pixel format of the captured AVI file is in standard 8-bit monochrome only.
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.
AVI_XVID_24:	Defines that the pixel format of the captured AVI file is in Xvid 24 bit color pixel format.

7.1.14 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 activated on changes to GPI1.
LUCAM_EVENT_GPI2_CHANGED:	Identifies that the event should be activated on changes to GPI2.
LUCAM_EVENT_GPI3_CHANGED:	Identifies that the event should be activated on changes to GPI3.
LUCAM_EVENT_GPI4_CHANGED:	Identifies that the event should be activated 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 not supported by all cameras and users have to check if the feature is supported by camera model.

Notes:

The LUCAM_EVENT_GPIX_CHANGED events are supported by very few cameras.

7.1.15 Trigger sequencing control

This is the definition of controls that can be done when using camera in trigger sequencing mode. These control need to be apply of read from the LUCAM_SEQUENING_CONTROL type.

LUCAM_SEQUENCING_CONTROL_OFF	Disable of enable trigger sequencing mode.
LUCAM_SEQUENCING_CONTROL_ON_IN_SNAPSHOT_MODE sequencing mode	Disable or enable snapshot mode in trigger
LUCAM_SEQUENCING_CONTROL_CIRCULAR restarting sequence)	Disable or enable circular mode (continuously
LUCAM_SEQUENCING_SETTING_TYPE_PROPERTY	Use in the main header.

7.1.15.1 Remarks

Trigger sequencing can only be use in snapshot mode. This mean the LUCAM_SEQUENCING_CONTROL_ON_IN_SNAPSHOT_MODE should always be active in the LUCAM_SEQUENCING_CONTROL type;

LUCAM_SEQUENCING_SETTING_TYPE_PROPERTY is the only mode supported by cameras that support sequencing.

7.1.16 TAP configuration

To increase speed performance, some camera sensors allow the read out to be performed by sensor area in parallel (at the same time). On supported camera models, it is possible to select how pixel data is extracted from the detector by writing the desired mode to the LUCAM_PROP_TAP_CONFIGURATION or LUCAM_PROP_STILL_TAP_CONFIGURATION. When writing to these properties, the camera should not be in streaming mode or waiting for a trigger in snapshot mode. On camera models that support TAP configuration if should be possible to use one of the tap configuration modes listed below. If a particular mode is not supported by the sensor, and error will be returned by the property write operation.

TAP_CONFIGURATION_SINGLE	Only one section of the sensor is read at a time (max quality).
TAP_CONFIGURATION_DUAL	2 sections of the sensor are read-out at the same time.
TAP_CONFIGURATION_2x1	2 sections of the sensor are read-out at the same time.
TAP_CONFIGURATION_QUAD	4 sections of the sensor read-out at the same time (fastest).
TAP_CONFIGURATION_2X2	4 sections of the sensor are read-out at the same time(fastest).

7.1.17 **Metadata extraction**

With some camera model and specific mode, it may be possible that images have Meta data embedded in the pixels data. The API distributes the data over multiple pixels in order to minimise the effect on the image. This data can be extracted from the image using LucamGetMetaData(). Specify the timestamp type to retrieve with this function

i. LUCAM_METADATA_FRAME_COUNTER
   - To extract image frame counter.

ii. LUCAM_METADATA_TIMESTAMP
   - To extract timestamp information.

See Also

LucamEnableTimestamp(), **LucamGetMetadata(), LucamGetTimestamp(),**
LucamGetTimestampFrequency(), LucamIsTimestampEnable(), LucamSetTimestamp();

7.1.18 **High Dynamic Range**

Normally, as the exposure time is increased, there will be a linear increase in the pixel intensity level. Some of the Lumenera camera models support a high dynamic range functionality that enables the setting of ”knee” points to adjust the way that the sensor responds to a scene with both dark and bright regions.

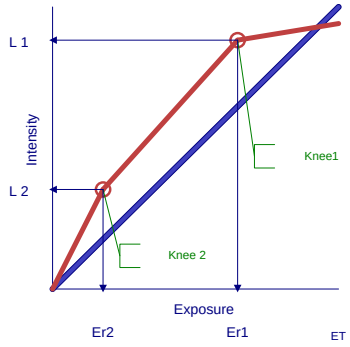

Figure 1 Intensity VS Exposure

For a camera model that has only one knee point:
When the camera only supports one knee point exposure, if the current exposure is higher than the knee exposure (Er1) and smaller than the total exposure, then the maximum pixel intensity that can be returned will be the LUCAM_PROP_KNEE1_LEVEL value.

For a camera model that supports 2 knee points:
When the camera supports 2 knee points, the first point becomes the lower range where if the current exposure is smaller than the first exposure point then the API will make sure that the minimum value returned is the LUCAM_PROP_KNEE2_LEVEL value. The second knee point will become the higher end of exposure.

7.2 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.

7.2.1 LUCAM_SNAPSHOT Structure

The LUCAM_SNAPSHOT data structure contains all settings to prepare sensor for an image acquisition in snapshot mode.

```
Struct
{
  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_EXPOSURE
  }
FLOAT strobeDelay;             // Delay in milliseconds to generate the strobe signal on reception
                               of the trigger BOOL useHwTrigger;  // wait for hardware trigger
                               flag
FLOAT timeout;                 // maximum time (ms) to wait for an image in snapshot mode prior to
                               returning to the main application. Suggested value to start with
                               USB 2.0 products is to use 500 ms + exposure time has timeout value
                               (See also LucamSetTimeout().
LUCAM_FRAME_FORMAT format;     // Definition of the sensor area to take image from,  see also
                               7.2.2.
ULONG shutterType; // Shutter mode of the camera for camera that support mode than 1 mode, see
                also 7.1.8
FLOAT exposureDelay; // Delay in milliseconds to start exposure from reception of trigger signal.
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)
}
```

7.2.2 LUCAM_FRAME_FORMAT Structure

The LUCAM_FRAME_FORMAT data structure contains all settings to setup camera sensor for an image acquisition. This is used by LucamGetFormat, LucamSetFormat and the LUCAM_SNAPSHOT data structure to capture images in snapshot mode.

```
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
```

7.2.3 LUCAM_VERSION Structure

This data structure contains different camera information populated by LucamEnumCameras and LucamQueryVersion.

```
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 cameraid; // Camera model identification number.
```

7.2.4 LUCAM_CONVERSION Structure

This data structure contains information to control parameter how the raw to color conversion

```
ULONG DemosaicMethod;         // Demosaic method to convert Bayer data to full color (see section
                              7.1.4)
ULONG CorrectionMatrix;       // Color correction matrix, see section 7.1.5 for possible values.
```

7.2.5 LUCAM_CONVERSION_PARAMS Structure

This data structure contains settings to create the color image from the raw using extended conversion function.

```
ULONG Size;                             // Size of this structure
ULONG DemosaicMethod;                   // Demosaic method to convert the Bayer data to full
                color see section 7.1.4
ULONG CorrectionMatrix;         // Correction matrix to use, see section 7.1.5
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, most likely
                color gains.
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
  };
};
```

7.2.6 LUCAM_IMAGE_FORMAT Structure

The LUCAM_IMAGE_FORMAT data structure hold information about image data. It will define the width, the height and the pixel format.

```
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
```

Please note that the fields Size need to be initialized to the size of the LUCAM_IMAGE_FORMAT size (sizeof(LUCAM_IMAGE_FORMAT)) before using the data structure with Lumenera API functions.

7.2.7 LGCAM_IP_CONFIGURATION Structure

The LG_CAM_IP_CONFIGURATION data structure contains Ethernet information of the GigE camera or the Network card that is connected to the GiGe camera.

```
ULONG IPAddress;        // Ethernet IP address.
ULONG SubnetMasK;               // Ethernet subnet.
ULONG DefaultGateway;           // Gateway Ethernet IP address.
```

7.2.8 LUCAM_SEQUENCE_SETTING_HEADER Structure

This is the main header use by trigger sequencing. Trigger sequencing is explained in detail in an application note.

```
typedef struct _LUCAM_SEQUENCE_SETTING_HEADER
{
   USHORT Type;     // see LUCAM_SEQUENCING_SETTING_TYPE_* above
   USHORT Size;     // set this to sizeof(LUCAM_SEQUENCE_SETTING)
   USHORT Frame;    // zero-based sequence number
   USHORT Reserved; // set this to 0
}LUCAM_SEQUENCE_SETTING_HEADER;
```

7.2.9 LUCAM_SEQUENCE_SETTING_PROPERTY Data Structure

This is the definition of a property manipulation to add to Trigger sequencing.

```
typedef struct _LUCAM_SEQUENCE_SETTING_PROPERTY
{
   LUCAM_SEQUENCE_SETTING_HEADER Header;
   ULONG Property; // LUCAM_PROP_*, for example LUCAM_PROP_STILL_EXPOSURE
   LONG Flags;     // LUCAM_PROP_FLAG_*
   FLOAT Value;
}LUCAM_SEQUENCE_SETTING_PROPERTY;
```

7.2.10 LUCAM_SEQUENCE_SETTING Data Structure

This is the data structure used with the High dynamic range and trigger sequencing functions.

```
typedef struct _LUCAM_SEQUENCE_SETTING
{
   union
   {
      LUCAM_SEQUENCE_SETTING_HEADER Hdr;
      LUCAM_SEQUENCE_SETTING_PROPERTY Property;
   };
}LUCAM_SEQUENCE_SETTING; // use with LucamSequencingSetup
```

7.2.11 LUCAM_SEQUENCING_STATUS

This is the data structure to use when querying trigger sequencing status information.

```
typedef struct _LUCAM_SEQUENCING_STATUS
{
   ULONG MaximumFrameCountInSequence;  // if 0 then sequencing is not supported
   ULONG CurrentFrameCountInSequence;  // corresponds to LucamSequencingSetup's framesPerSequence
```

```
   ULONG NextFrameCount;               // 'timestamp' of the next frame
   ULONG NextFrameIndexInSequence;     // zero-based sequence of the next frame
   LUCAM_SEQUENCING_CONTROL State;     // will automatically clear if
                LUCAM_SEQUENCING_CONTROL_CIRCULAR is not set
   ULONG Reserved;
}LUCAM_SEQUENCING_STATUS; // use with LucamSequencingGetStatus
```

Annex A. Starting Video stream flowchart