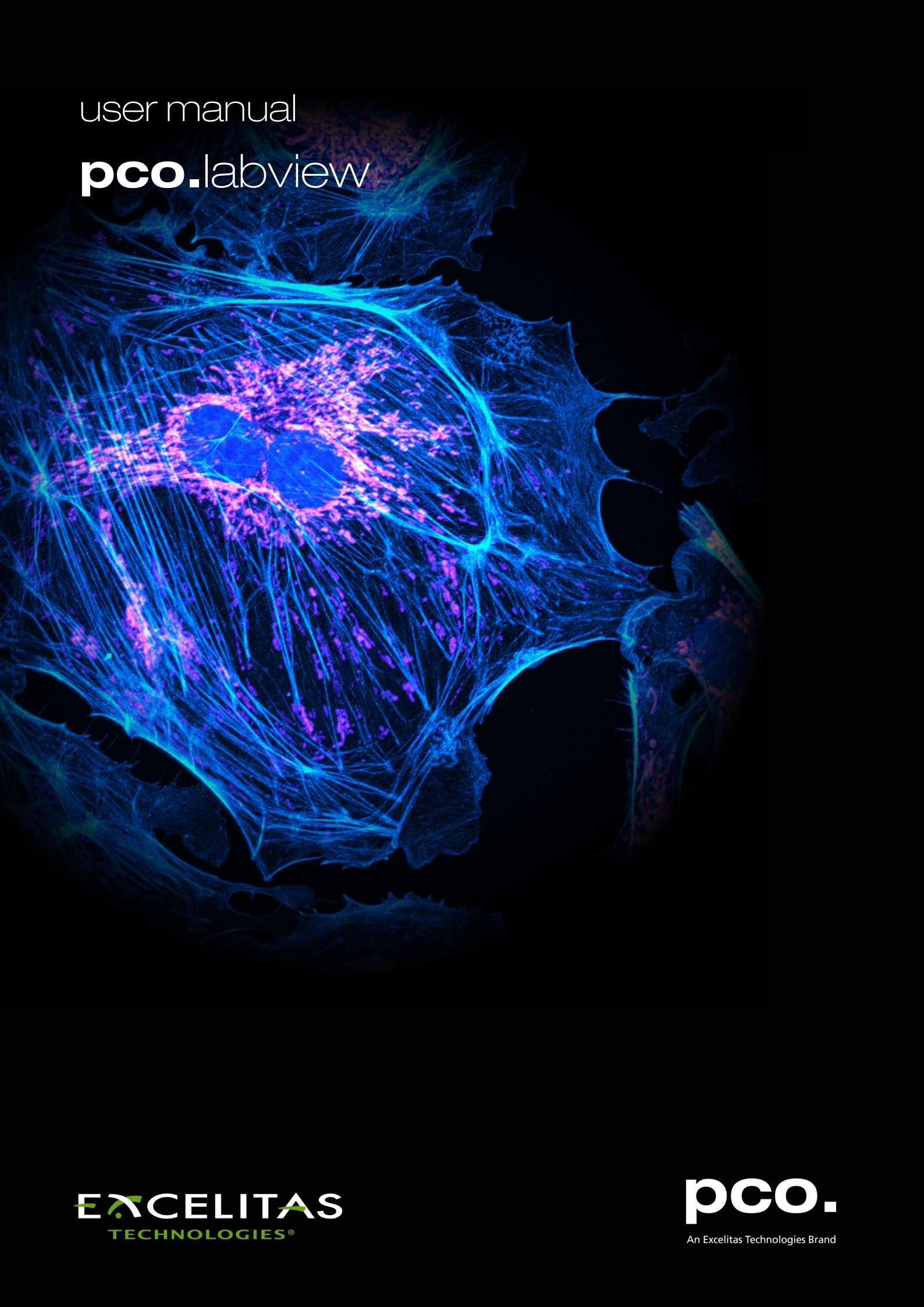


user manual

pco.labview



Excelitas PCO GmbH asks you to carefully read and follow the instructions in this document. For any questions or comments, please feel free to contact us at any time.



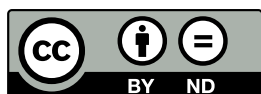
An Excelitas Technologies Brand

telephone:	+49 (0) 9441 2005 50
fax:	+49 (0) 9441 2005 20
postal address:	Excelitas PCO GmbH Donaupark 11 93309 Kelheim, Germany
email:	pco@excelitas.com
web:	www.pco.de

pco.labview user manual 5.1.0

Released August 2023

©Copyright Excelitas PCO GmbH



This work is licensed under the Creative Commons Attribution-NoDerivatives 4.0 International License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nd/4.0/> or send a letter to Creative Commons, PO Box 1866, Mountain View, CA 94042, USA.

Contents

1	General	4
1.1	Installation	4
1.2	Basic Usage	4
1.3	Recorder Modes	5
1.4	Image Formats	6
1.5	Error Handling	7
2	API Documentation	8
2.1	Camera Functions	8
2.1.1	init	9
2.1.2	close	10
2.1.3	getName	11
2.1.4	getSerial	12
2.1.5	isRecording	13
2.1.6	isColored	14
2.1.7	defaultConfiguration	15
2.1.8	getDescription	16
2.1.9	getConfiguration	17
2.1.10	setConfiguration	18
2.1.11	getConvertControl	19
2.1.12	setConvertControl	20
2.1.13	loadLut	21
2.1.14	adaptWhiteBalance	22
2.1.15	getSensorInfo	23
2.1.16	getExposureTime	24
2.1.17	setExposureTime	25
2.1.18	getDelayTime	26
2.1.19	setDelayTime	27
2.1.20	record	28
2.1.21	stop	29
2.1.22	waitForFirstImage	30
2.1.23	waitForNewImage	31
2.1.24	getRecordedImageCount	32
2.1.25	image	33
2.1.26	images	35
2.1.27	imageAverage	37
2.1.28	sdk	38
2.1.29	rec	39
2.1.30	conv	40
2.2	Controls	41
2.2.1	Binning	41
2.2.2	Roi	41
2.2.3	Configuration	41
2.2.4	Description	42
2.2.5	ConvertControl	43
3	Type Definitions	44
4	About Excelitas PCO	46

1 General

The **pco.labview** package is a powerful and easy to use high-level Software Development Kit (SDK) for working with PCO cameras under LabVIEW. It contains everything needed for camera setup, image acquisition, readout and color conversion.

The high-level class architecture makes it very easy to integrate PCO cameras into your own LabVIEW software, while still having access to the underlying **pco.sdk** and **pco.recorder** functions for a detailed control of all possible functionalities.

This version of pco.labview is suitable for LabVIEW 2019 and newer. If you need to use an older LabVIEW version please contact us for getting suitable pco.labview packages for your version.

Note This document describes only the functions and usage of PCO's *Camera* class. All functions of the **pco.sdk** and **pco.recorder** SDK are wrapped into LabVIEW VI's inside the corresponding folders. If you need a full description of those functions, please check out the manuals of **pco.sdk** and **pco.recorder**. Although these are C libraries, the functions are nearly identical to their wrapped LabVIEW counterparts.

1.1 Installation

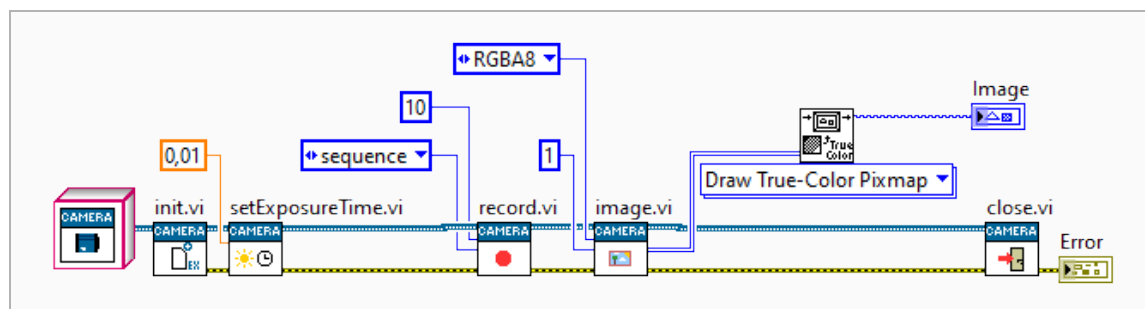
Download the windows installer, unzip and execute it. Simply follow the steps in the installer.

In your install directory you will find:

- A LabVIEW project containing all (sub)VI's, arranged similar to the folder structure
- An **Examples** folder containing all example VI's showing different use cases ¹
- The **pco.camera** folder containing the actual sources of this *Camera* class
- The **pco.sdk**, **pco.recorder**, **pco.convert** folders with the wrapped resources of the corresponding SDK's
- A **runtime** folder containing the required runtime DLL's
- In **errorHandling** you will find a VI to map PCO's error codes into LabVIEW error clusters (this is used in all wrapped Sub-VIs)
- **GetDLLPaths.vi** is needed to automatically load the correct runtime DLL's for your LabVIEW version, i.e. 64 or 32 bit

1.2 Basic Usage

This snippet shows the basic usage.



¹The deprecated subfolder contains old examples showing everything without using the *Camera* class

By calling **init.vi**, a camera is searched, opened and initialized. Several functions are provide to adjust the camera settings, and here we set the exposure time to 10 ms. Calling **record.vi** will start the recording. Depending on the mode it either waits until record is finished (like for sequence mode which is selected here) or directly returns (see 1.3 for the full list of available modes).

The **image.vi** returns different image data types. Regardless of the image format, the function always outputs the raw image as two-dimensional array of 16 bit values. Depending on the selected format (see 1.4 for available formats) you can also access the data either as 2d array of 8 Bit values, 2d array of 32 Bit values (RGB encoded) or 2d array of clusters (each cluster contains for values for R, G, B and A).

Here we want to have the image with **index** 1 in the *RGBA8* format, so we use the U32 image output.

1.3 Recorder Modes

Depending on your workflow you can choose between different recording modes.

Some modes are blocking, i.e. the `record` function waits until recording is finished, some are non-blocking.

Some modes store images in memory, the others directly as file(s) on the disk. However, for the recorder modes that store the images as files, accessing the recorded images is identical to accessing images in memory.

Mode	Storage	Blocking	Description
<code>sequence</code>	Memory	yes	Record a sequence of images
<code>sequence_non_blocking</code>	Memory	no	Record a sequence of images, do not wait until record is finished
<code>ring_buffer</code>	Memory	no	Continuously record images in a ringbuffer, once the buffer is full, old images are overwritten
<code>fifo</code>	Memory	no	Record images in fifo mode, i.e. you will always read images sequentially and once the buffer is full, recording will pause until older images have been read
<code>sequence_dpcore</code>	Memory	yes	Same as <code>sequence</code> , but with DotPhoton preparation enabled
<code>sequence_non_blocking_dpcore</code>	Memory	no	Same as <code>sequence_non_blocking</code> , but with DotPhoton preparation enabled
<code>ring_buffer_dpcore</code>	Memory	no	Same as <code>ring_buffer</code> , but with DotPhoton preparation enabled
<code>fifo_dpcore</code>	Memory	no	Same as <code>fifo</code> , but with DotPhoton preparation enabled

Continued on next page

Continued from previous page

Mode	Storage	Blocking	Description
tif	File	no	Record images directly as tif files
multitiff	File	no	Record images directly as one or more multitiff file(s)
pcoraw	File	no	Record images directly as one pcoraw file
dicom	File	no	Record images directly as dicom files
multidicom	File	no	Record images directly as one or more multi-dicom file(s)

In the code the mode is represented as an enum type.

Note For more information on the DotPhoton preparation and image compression, please visit [DotPhoton](#) or feel free to contact us.

1.4 Image Formats

Besides the standard 16 bit raw image data you also have the possibility to get your images in different formats, shown in the table below.

The data types have been selected so they can easily be converted to corresponding **IMAQ images** used by the NI-IMAQ module, as shown in all *_IMAQ.vi examples. This makes it very easy for you to integrate PCO cameras inside your NI-IMAQ applications.

The format is selected when calling the `image / images / imageAverage` functions (see 2.1.25, 2.1.26, 2.1.27) of the `Camera` class. Depending on the selected format only one of the possible image outputs will be valid, but in all cases the functions provide the raw image data as 2d 16 bit array.

Format	Description
Mono8	Get image as 8 bit grayscale data, use image(U8) output
Mono16	Get the raw image as 16 bit grayscale/raw data, use raw_image(U16) output
RGBA8	Get image as 32 bit color data, use image(U32) output
BGRA8	Same as <code>RGBA8</code> but with flipped color channels (provided only for completeness, in LabVIEW typically <code>RGBA8</code> is used)
RGBA16	Get image as 64 bit color data, use image(U64) output
BGRA16	Same as <code>RGBA16</code> but with flipped color channels (provided only for completeness, in LabVIEW typically <code>RGBA16</code> is used)

In the code the image format is represented as an enum type.

Note For monochrome cameras, the `RGBA16/BGRA16` format is not available and the colors in the `RGBA8/BGRA8` depend on the selected lut, which is a standard grayscale mapping by default. For selecting different lut files you can use the functions `setConvertControl` (see 2.1.12) or `loadlut` (see 2.1.13) from the `Camera` class.

1.5 Error Handling

Each VI has an `error in` and an `error out` connector for a standard error cluster variable. All VI's check the incoming error before they execute their content. With this mechanism you can simply connect the VI's error connectors and be sure that possible errors are forwarded correctly. Either at the end or anywhere in between you can view the current error status, as you can see it in the example in 1.2.

We recommend to always connect the error inputs and outputs.

Additionally you can also enable the logging of the underlying SDK's. For more information on that please visit our [pco.logging page](#).

Note In the `close` function the content will also be executed in case of an incoming error. This ensures that every opened camera or resource gets closed properly, even if an error occurs. Incoming errors are merged with other errors that might occur during the function execution.

2 API Documentation

The following section describes the functionality of the `Camera` class. The class can be divided into functions and controls.

2.1 Camera Functions

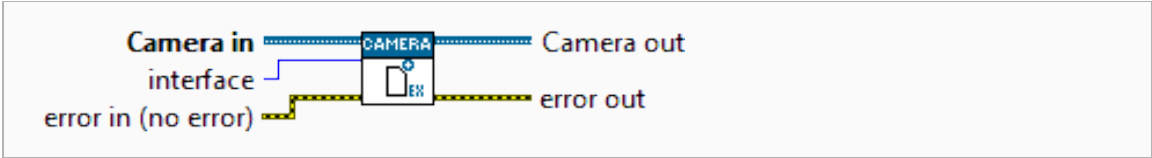
This section describes the functions of the `Camera` class. The following table provides a short overview of the most important functions:

- **init** Open and initialize a camera with its default configuration
- **close** Close the camera and clean up everything
- **defaultConfiguration()** Set default configuration to the camera
- **getConfiguration()** Get current camera configuration
- **setConfiguration()** Set a new configuration to the camera
- **getExposureTime()** Get current exposure time
- **setExposureTime()** Set new exposure time to the camera
- **record()** Initialize and start the recording of images
- **stop()** Stop the current recording
- **waitForFirstImage()** Wait until the first image has been recorded
- **waitForNewImage()** Wait until a new image has been recorded
- **getConvertControl()** Get current color convert settings
- **setConvertControl()** Set new color convert settings
- **image()** Read a recorded image
- **images()** Read a series of recorded images
- **imageAverage()** Read an averaged image (averaged over all recorded images)

2.1.1 init

Description Initializes the camera.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initial Camera object, a constant can be used here
cU8	interface	Specific interface to search for cameras. If not connected, search on all interfaces.
cErrClst	error in	Previous error state (no error if not connected)

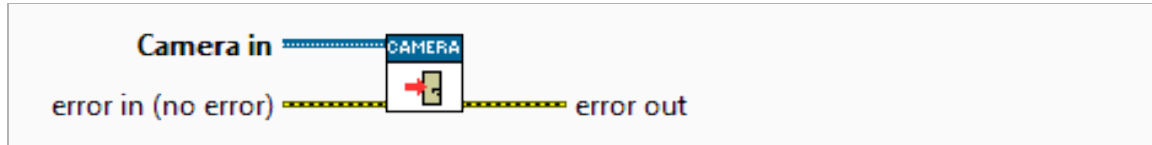
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.2 close

Description Closes the activated camera and releases the blocked resources.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cErrClst	error in	Previous error state (no error if not connected)

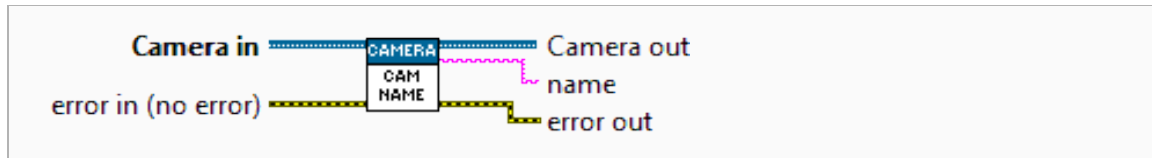
Parameter OUT

Datatype	Name	Description
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.3 getName

Description Get the name of the camera.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cErrClst	error in	Previous error state (no error if not connected)

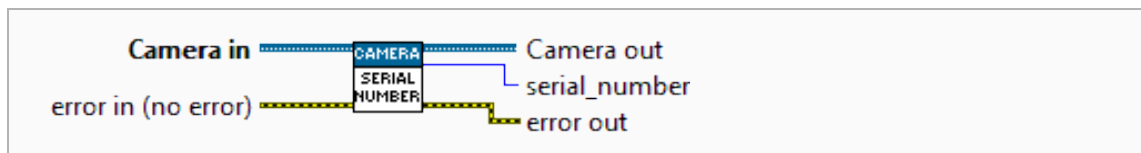
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iStr	name	Camera name
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.4 getSerial

Description Get the serial number of the camera.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cErrClst	error in	Previous error state (no error if not connected)

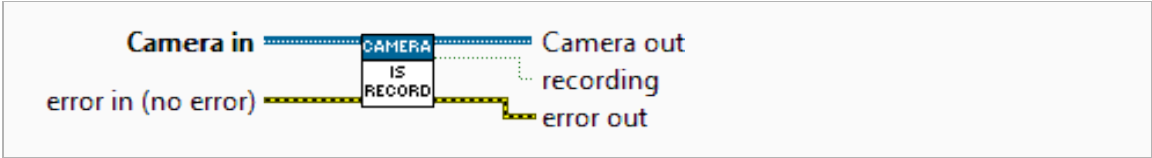
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iU32	serial_number	Camera serial number
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.5 isRecording

Description Return the flag if a recording is currently active.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cErrClst	error in	Previous error state (no error if not connected)

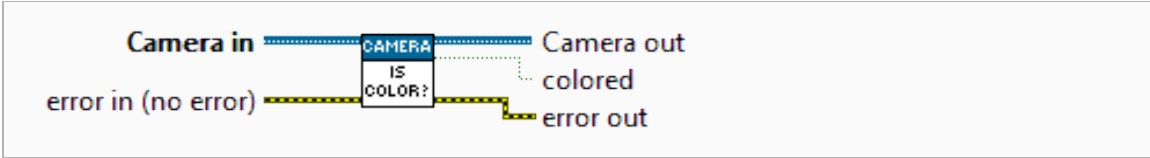
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iBool	recording	Flag if a recording is currently active
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.6 isColored

Description Return the flag if camera is a color camera.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cErrClst	error in	Previous error state (no error if not connected)

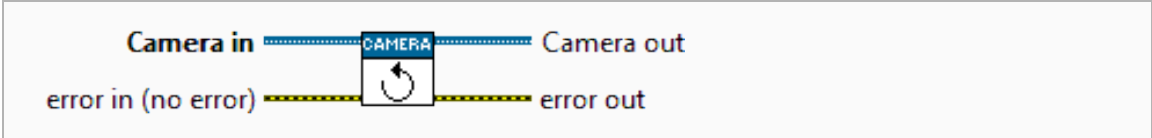
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iBool	colored	Flag if camera is colored
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.7 defaultConfiguration

Description (Re)set the camera to its default configuration.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cErrClst	error in	Previous error state (no error if not connected)

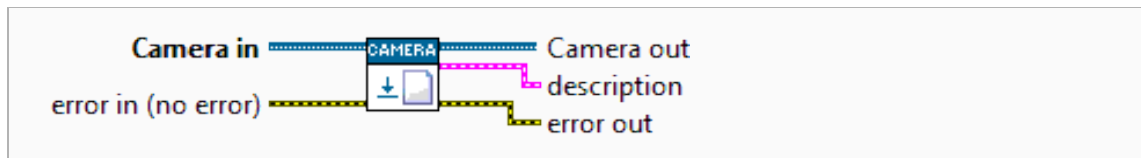
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.8 getDescription

Description Get the description parameters of the camera.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cErrClst	error in	Previous error state (no error if not connected)

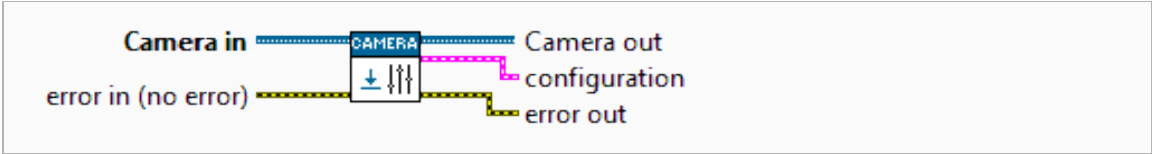
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iClst	description	Structure containing the description of the camera (see 2.2.4)
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.9 getConfiguration

Description Get the current camera configuration.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cErrClst	error in	Previous error state (no error if not connected)

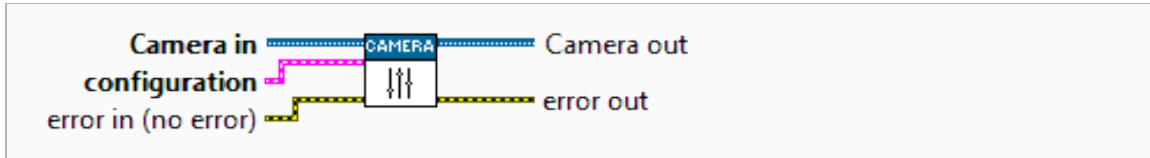
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iClst	configuration	Structure containing the current configuration of the camera (see 2.2.3)
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.10 setConfiguration

Description Set a configuration to the camera.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cClst	configuration	Configuration that should be set (see 2.2.3)
cErrClst	error in	Previous error state (no error if not connected)

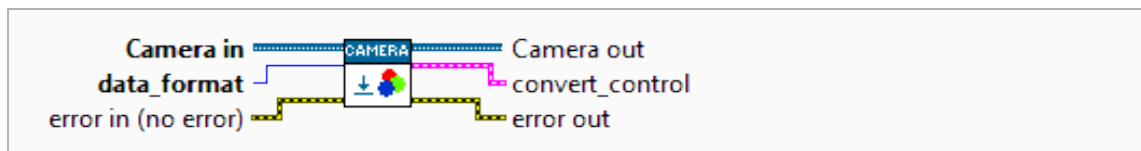
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.11 getConvertControl

Description Get the current convert control settings for the specified data format.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cEnum	data_format	Data format for which the convert settings should be queried
cErrClst	error in	Previous error state (no error if not connected)

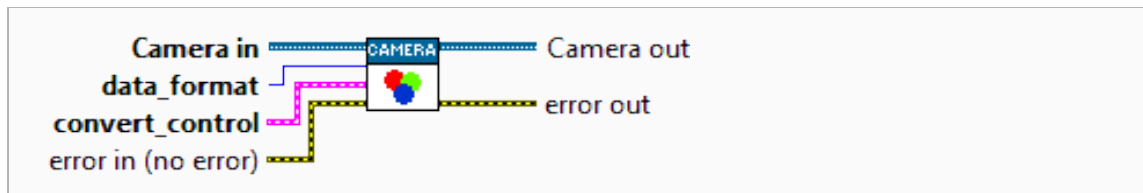
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iClst	convert_control	Structure containing the current convert settings for the specified data format (see 2.2.5)
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.12 setConvertControl

Description Set convert control settings for the specified data format.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cEnum	data_format	Data format for the convert settings to be set
cClst	convert_control	Convert control settings that should be set.
cErrClst	error in	Previous error state (no error if not connected)

Parameter OUT

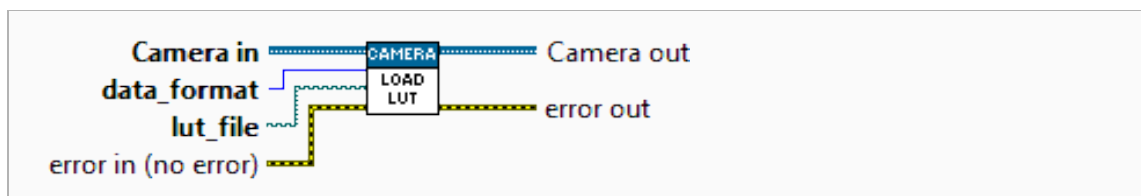
Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.13 loadLut

Description Set the lut file for the convert control settings.

This is just a convenience function, the lut file could also be set using `setConvertControl` (see: 2.1.12).

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cEnum	data_format	Data format for which the lut file should be set.
cPath	lut_file	Actual lut file path to be set.
cErrClst	error in	Previous error state (no error if not connected)

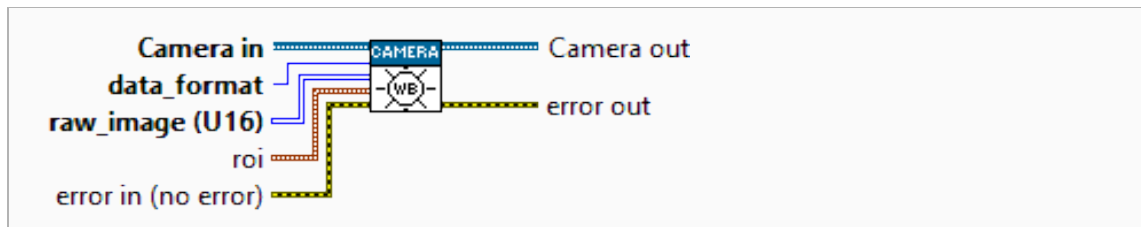
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.14 adaptWhiteBalance

Description Do a white-balance according to a transferred image.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cEnum	data_format	Data format for which the white balance should be adapted.
cU16_2d	raw_image (U16)	Image that should be used for white-balance computation
cClstN	roi	Use only the specified ROI for white-balance computation
cErrClst	error in	Previous error state (no error if not connected)

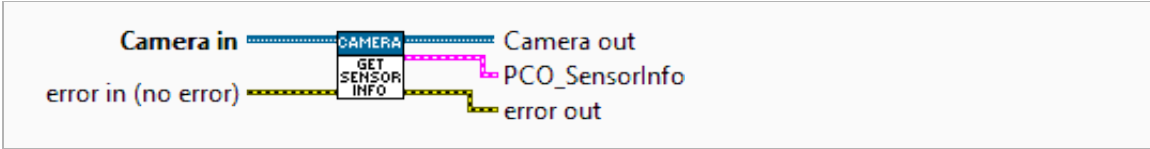
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.15 getSensorInfo

Description Get the sensor information of the camera.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cErrClst	error in	Previous error state (no error if not connected)

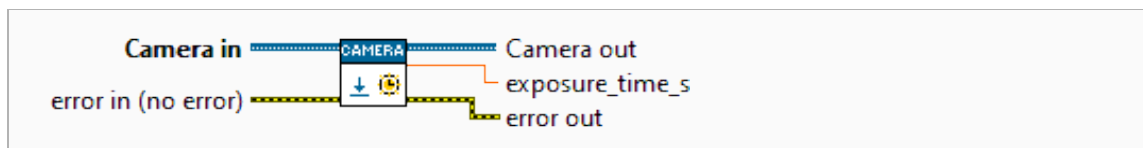
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iClst	PCO_SensorInfo	Sensor information of the camera (please see pco.sdk manual for a detailed structure information)
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.16 getExposureTime

Description Get the current exposure time of the camera.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cErrClst	error in	Previous error state (no error if not connected)

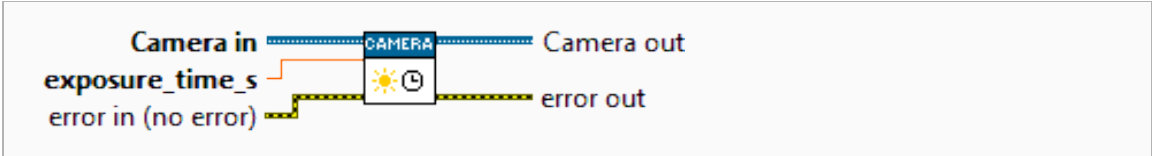
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iDbl	exposure_time_s	Exposure time of the camera [s]
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.17 setExposureTime

Description Set a new exposure time to the camera.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cDbl	exposure_time_s	Exposure time [s] that should be set
cErrClst	error in	Previous error state (no error if not connected)

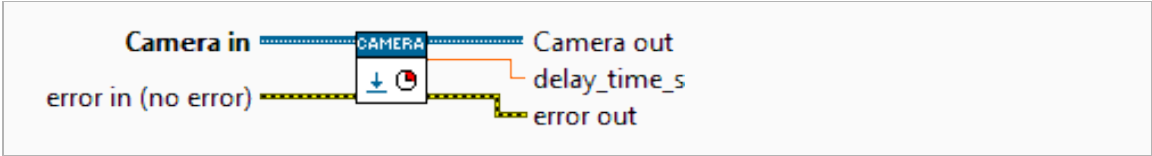
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.18 getDelayTime

Description Get the current delay time of the camera.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cErrClst	error in	Previous error state (no error if not connected)

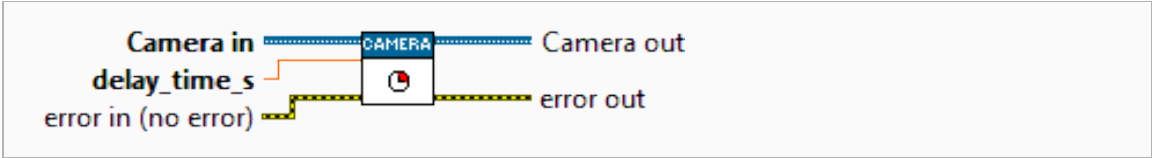
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iDb1	delay_time_s	Delay time of the camera [s]
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.19 setDelayTime

Description Set a new delay time to the camera.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cDbl	delay_time_s	Delay time [s] that should be set
cErrClst	error in	Previous error state (no error if not connected)

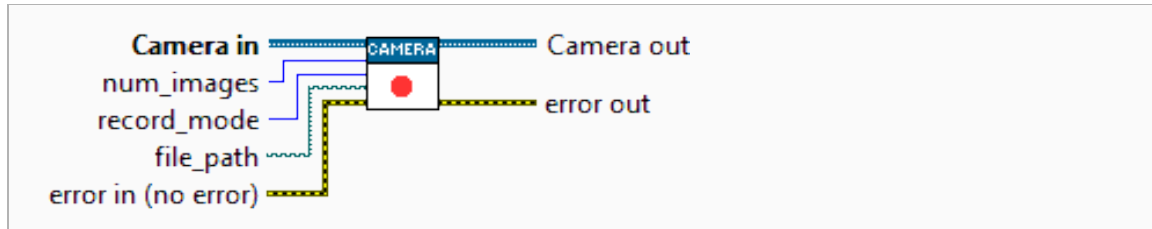
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.20 record

Description Creates, configures, and starts a new recorder instance. The entire camera configuration must be set before calling `record`. The commands for getting and setting delay/exposure time are the only exception. They can be called during the recording.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cU32	num_images	Sets the number of images allocated. The RAM or disk (depending on the mode) of the PC limits the maximum value
cEnum	record_mode	Defines the recording mode for this record (see 1.3)
cPath	file_path	Path where the image file(s) should be stored (only for modes who directly save to file, see 1.3)
cErrClst	error in	Previous error state (no error if not connected)

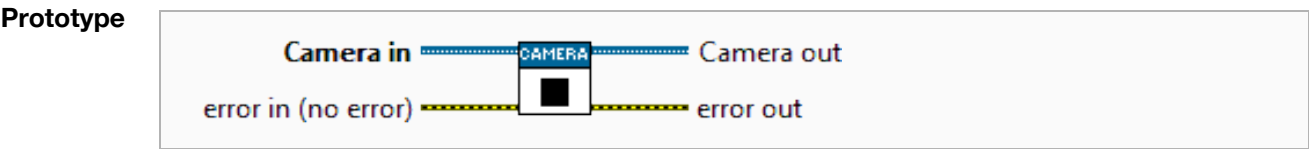
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.21 stop

Description Stops the current recording.

For blocking recorder modes (see 1.3), the recording is automatically stopped when the required number of images is reached. In this case `stop` is not needed.



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cErrClst	error in	Previous error state (no error if not connected)

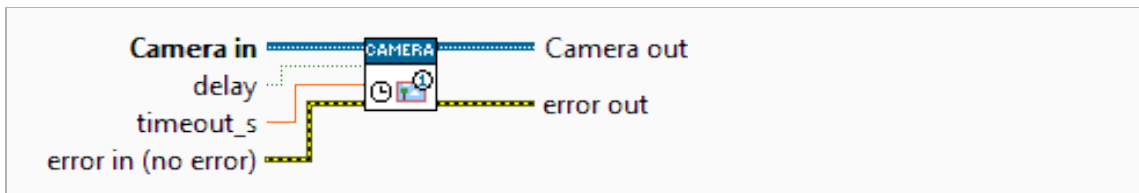
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.22 waitForFirstImage

Description Waits until the first image has been recorded and is available.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cBool	delay	Flag if a small delay should be used in the waiting loop (typically recommended to reduce cpu load)
cDbl	timeout_s	If connected, the waiting loop will be aborted if no image was recorded during timeout_s seconds.
cErrClst	error in	Previous error state (no error if not connected)

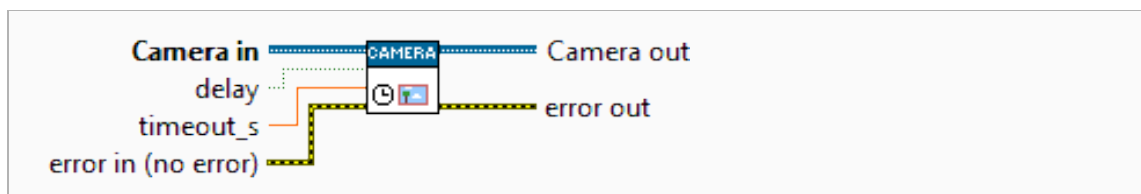
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.23 waitForNewImage

Description Waits until a new image has been recorded and is available (i.e. an image that has not been read yet).

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cBool	delay	Flag if a small delay should be used in the waiting loop (typically recommended to reduce cpu load)
cDbl	timeout_s	If connected, the waiting loop will be aborted if no image was recorded during timeout_s seconds.
cErrClst	error in	Previous error state (no error if not connected)

Parameter OUT

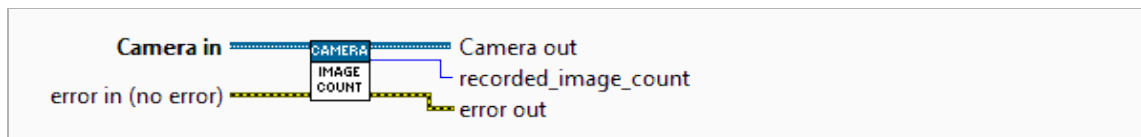
Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.24 getRecordedImageCount

Description Get the number of currently recorded images.

Note For recorder modes *fifo* and *fifo_dpcore* (see 1.3) this represents the current fill level of the fifo buffer, not the overall number of recorded images.
So in FIFO mode check for **getRecordedImageCount() > 0** to see if a new image is available.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cErrClst	error in	Previous error state (no error if not connected)

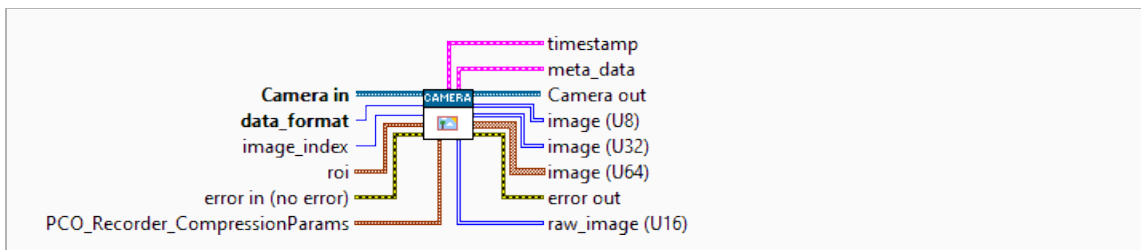
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iU32	recorded_image_count	Number of currently recorded images
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.25 image

Description Get a recorded image in the given format. Depending on the transferred format you have to use one of the image outputs. The `raw_image(16)` output is always valid.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cEnum	data_format	Data format the image should have (see 1.4)
cU32	image_index	Index of the image that should be queried, <code>0xFFFFFFFF</code> for latest image (for modes <code>fifo</code> / <code>fifo_dpcore</code> always use 0 (see 1.3))
cClstN	roi	Soft ROI to be applied, i.e. get only the ROI portion of the image (see 2.2.2 for the <code>Roi</code> structure)
cErrClst	error in	Previous error state (no error if not connected)
cClstN	PCO_Recorder_CompressionParams	Compression parameter struct, not implemented jet

Parameter OUT

Datatype	Name	Description
iClst	timestamp	Timestamp of the image
iClst	meta_data	Metadata of the image
iCamera	Camera out	Initialized Camera object controlling an opened camera
iU8Arr_2d	image (U8)	Image data as 2d 8 bit array (use this for format <code>MonO8</code> , see 1.4)
iU32Arr_2d	image (U32)	Image data as 2d 32 bit array (use this for formats <code>RGBA8</code> , <code>BGRA8</code> , see 1.4)
iClstArrN_2d	image (U64)	Image data as 2d 64 bit (cluster of 4 times 16 bit) array (use this for formats <code>RGBA16</code> , <code>BGRA16</code> , see 1.4)
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

Continued on next page

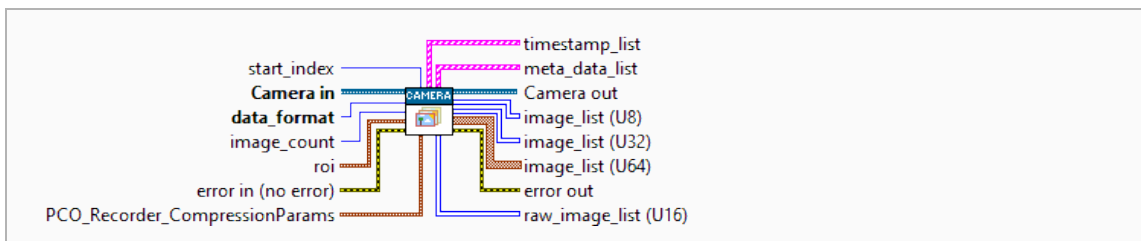
Continued from previous page

Datatype	Name	Description
iU16Arr_2d	raw_image (U16)	Image data as 2d 16 bit array (use this for getting the raw image data, in format <code>MonO16</code> this is the only valid image output, see 1.4)

2.1.26 images

Description Get a series of images in the given format as 3d image data. The positions of the images to query are defined by a `start_index` and an `image_count`. Depending on the transferred format you have to use one of the image outputs. The `raw_image (16)` output is always valid.

Prototype



Parameter IN

Datatype	Name	Description
cU32	start_index	Index of the first image that should be queried
cCamera	Camera in	Initialized <code>Camera</code> object controlling an opened camera
cEnum	data_format	Data format the image should have (see 1.4)
cU32	image_count	Number of images that should be queried
cClstN	roi	Soft ROI to be applied, i.e. get only the ROI portion of the image (see 2.2.2 for the <code>Roi</code> structure)
cErrClst	error in	Previous error state (no error if not connected)
cClstN	PCO_Recorder_CompressionParams	Compression parameter struct, not implemented jet

Parameter OUT

Datatype	Name	Description
iClstArr_1d	timestamp_list	Array of timestamps for the images
iClstArr_1d	meta_data_list	Array of metadata for the images
iCamera	Camera out	Initialized <code>Camera</code> object controlling an opened camera
iU8Arr_2d	image_list (U8)	List of images as 3d 8 bit array (use this for format <code>Mono8</code> , see 1.4)
iU32Arr_2d	image_list (U32)	List of images as 3d 32 bit array (use this for formats <code>RGBA8</code> , <code>BGRA8</code> , see 1.4)
iClstArrN_2d	image_list (U64)	List of images as 3d 64 bit (cluster of 4 times 16 bit) array (use this for formats <code>RGBA16</code> , <code>BGRA16</code> , see 1.4)

Continued on next page

Continued from previous page

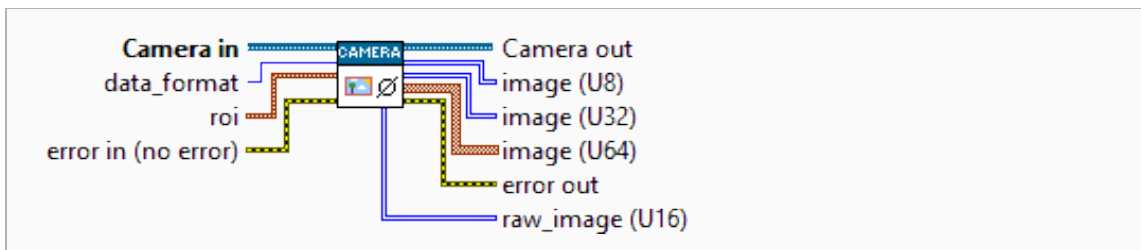
Datatype	Name	Description
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)
iU16Arr_2d	raw_image_list(U16)	List of images as 3d 16 bit array (use this for getting the raw images, in format <code>MonO16</code> this is the only valid image output, see 1.4)

2.1.27 imageAverage

Description Get an averaged image, averaged over all recorded images in the given format. Depending on the transferred format you have to use one of the image outputs. The `raw_image(16)` output is always valid.

Note We recommend that you do not use this function while recording is active, as it may give unexpected results (especially in `ring_buffer` mode, see 1.3). Typically you would record the number of images you want to average as sequence, then compute the average and after all images have been recorded.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cEnum	data_format	Data format the image should have (see 1.4)
cClstN	roi	Soft ROI to be applied, i.e. get only the ROI portion of the image (see 2.2.2 for the <code>Roi</code> structure)
cErrClst	error in	Previous error state (no error if not connected)

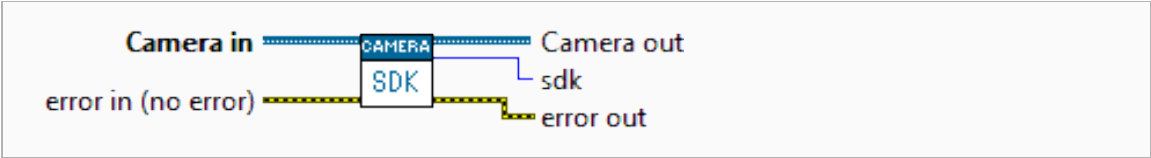
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iU8Arr_2d	image (U8)	Averaged image as 2d 8 bit array (use this for format <code>Mono8</code> , see 1.4)
iU32Arr_2d	image (U32)	Averaged image as 2d 32 bit array (use this for formats <code>RGBA8</code> , <code>BGRA8</code> , see 1.4)
iClstArrN_2d	image (U64)	Averaged image as 2d 64 bit (cluster of 4 times 16 bit) array (use this for formats <code>RGBA16</code> , <code>BGRA16</code> , see 1.4)
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)
iU16Arr_2d	raw_image (U16)	Averaged image as 2d 16 bit array (use this for getting the raw image data, in format <code>Mono16</code> this is the only valid image output, see 1.4)

2.1.28 sdk

Description Get the internal handle to the pco.sdk API. This is needed whenever you need to call special pco.sdk functions directly.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cErrClst	error in	Previous error state (no error if not connected)

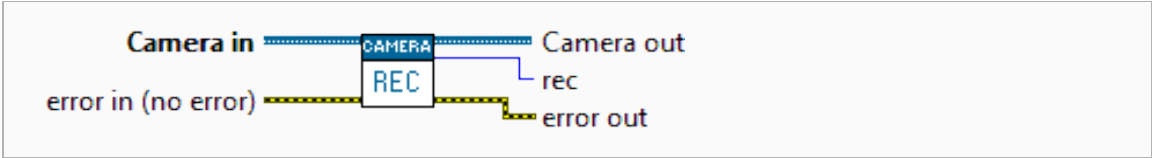
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iU64	sdk	Handle to the pco.sdk library functions
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.29 rec

Description Get the internal handle to the pco.recorder API. This is needed whenever you need to call special pco.recorder functions directly.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cErrClst	error in	Previous error state (no error if not connected)

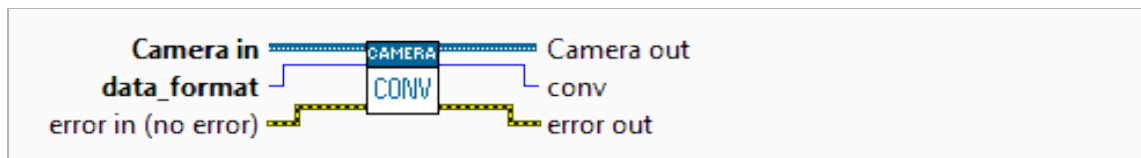
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iU64	rec	Handle to the pco.recorder library functions
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.30 conv

Description Get the internal handle to the pco.convert API. This is needed whenever you need to call special pco.convert functions directly.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cEnum	data_format	Data format for which the convert handle should be queried.
cErrClst	error in	Previous error state (no error if not connected)

Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iU64	conv	Handle to the pco.convert library functions
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.2 Controls

In the following sections you will find all controls used in the `Camera` class.

Here we omit the the "c" and "i" in the datatype description because control elements can be used as controls and indicators.

2.2.1 Binning

Description Structure holding the binning information.

Datatype	Name	Description
U16	vert	Vertical binning
U16	horz	Horizontal binning

2.2.2 Roi

Description Structure holding the roi information.

Datatype	Name	Description
U64	x0	Left position of ROI (starting from 1)
U64	y0	Top position of ROI (starting from 1)
U64	x1	Right position of ROI (up to full width)
U64	y1	Bottom position of ROI (up to full height)

2.2.3 Configuration

Description Structure holding a camera configuration.

Datatype	Name	Description
Db1	exposure_time_s	Exposure time [s]
Db1	delay_time_s	Delay time [s]
ClstN	roi	Hardware ROI structure (see 2.2.2)
U16	timestamp_mode	Timestamp mode
U32	pixelrate	Pixelrate
U16	trigger_mode	Trigger mode
U16	acquire_mode	Acquire mode
U16	metadata_mode	Metadata mode
U16	noisefilter_mode	Noise filter mode
ClstN	binning	Binning structure (see 2.2.1)

2.2.4 Description

Description Structure holding the camera description information.

Datatype	Name	Description
U32	serial	Serial number of the camera
U16	type	Sensor type
U16	sub_type	Sensor sub type
U16	interface_type	Interface type
Db1	min_exposure_time_s	Minimal possible exposure time
Db1	max_exposure_time_s	Maximal possible exposure time
Db1	min_exposure_step_s	Minimal possible exposure step
Db1	min_delay_time_s	Minimal possible delay time
Db1	max_delay_time_s	Maximal possible delay time
Db1	min_delay_step_s	Minimal possible delay step
Db1	min_delay_step_s	Minimal possible delay step
U64	min_width	Minimal possible image width (hardware ROI)
U64	min_height	Minimal possible image height (hardware ROI)
U64	max_width	Maximal possible image width (hardware ROI)
U64	max_height	Maximal possible image height (hardware ROI)
U64	roi_step_horz	Horizontal ROI stepping (hardware ROI)
U64	roi_step_vert	Vertical ROI stepping (hardware ROI)
Bool	roi_symmetric_horz	Flag if hardware ROI has to be horizontally symmetric (i.e. if x0 is increased, x1 has to be decreased by the same value)
Bool	roi_symmetric_vert	Flag if hardware ROI has to be vertically symmetric (i.e. if y0 is increased, y1 has to be decreased by the same value)
Bool	has_timestamp_mode	Flag if camera supports the timestamp setting
Bool	has_timestamp_mode_ascii_only	Flag if camera supports setting the timestamp to ascii-only
U32Arr_1d	pixelrate_vec	Vector containing all possible pixelrate frequencies (index 0 is default)
Bool	has_acquire_mode	Flag if camera supports the acquire mode setting
Bool	has_ext_acquire_mode	Flag if camera supports the external acquire setting
Bool	has_metadata_mode	Flag if metadata can be activated for the camera

Continued on next page

Continued from previous page

Datatype	Name	Description
Bool	has_ram	Flag if camera has internal memory
U16Arr_1d	binning_horz_vec	Vector containing all possible horizontal binning values
U16Arr_1d	binning_vert_vec	Vector containing all possible vertical binning values

2.2.5 ConvertControl

Description Structure containing (color) convert information.

Datatype	Name	Description
Bool	sharpen	Flag if the image should be sharpened
Bool	adaptive_sharpen	Flag if adaptive sharpening should be enabled
Bool	flip_vertical	Flag if the image should be vertically flipped
Bool	auto_minmax	Flag if auto scale should be enabled
U16	min_limit	Minimum scaling value (will be ignored if auto scale is enabled)
U16	max_limit	Maximum scaling value (will be ignored if auto scale is enabled)
Dbl	gamma	Gamma of the image (default is 1.0)
I32	contrast	Contrast of the image (default is 0)
Bool	pco_debayer_algorithm	Flag if PCO debayering should be used
I32	color_temperature	Color temperature of the image
I32	color_saturation	Color saturation of the image
I32	color_vibrance	Color vibrance of the image
I32	color_tint	Color tint of the image
Path	lut_file	Path of the lut file that should be used

3 Type Definitions

The following table shows the correlation of data type names used in this manual and the corresponding **LabVIEW** data type symbols.

All controls start with **c**, all indicators start with **i**.

Name	Type	Description
cCamera		Camera class object control
iCamera		Camera class object indicator
cBool		Boolean variable control
iBool		Boolean variable indicator
cClst		Cluster variable (mixed data types) control
iClst		Cluster variable (mixed data types) indicator
cDb1		Double precision variable control
iDb1		Double precision variable indicator
cEnum		Enum variable control
iEnum		Enum variable indicator
cErrClst		LV error code cluster control
iErrClst		LV error code cluster indicator
iI16		Signed 16 bit variable indicator
cI32		Signed 32 bit variable control
iI32		Signed 32 bit variable indicator
cClstN		Cluster variable (numeric data types) control
iClstN		Cluster variable (numeric data types) indicator
cPath		(File) Path variable control
iPath		(File) Path variable indicator
cStr		String variable control
iStr		String variable indicator
cU8		Unsigned 8 bit variable control
iU8		Unsigned 8 bit variable indicator
cU16		Unsigned 16 bit variable control
iU16		Unsigned 16 bit variable indicator
cU32		Unsigned 32 bit variable control
iU32		Unsigned 32 bit variable indicator
cU64		Unsigned 64 bit variable control
iU64		Unsigned 64 bit variable indicator
iTS		LV timestamp variable indicator
iClstArr_1d		one-dimensional array of clusters (mixed data types) indicator
iDb1Arr_1d		one-dimensional array of double precision values indicator
iU16Arr_1d		one-dimensional array of unsigned 16 bit values indicator
iU32Arr_1d		one-dimensional array of unsigned 32 bit values indicator
iClstArrN_2d		two-dimensional array of clusters (numeric data types) indicator

Continued on next page

Continued from previous page

Name	Type	Description
iU8Arr_2d	[U8]	two-dimensional array of unsigned 8 bit values indicator
cU16_2d	[U16]	two-dimensional array of unsigned 16 bit values control
iU16Arr_2d	[U16]	two-dimensional array of unsigned 16 bit values indicator
iU32Arr_2d	[U32]	two-dimensional array of unsigned 32 bit values indicator
iClstArrN_3d	[U8]	three-dimensional array of clusters (numeric data types) indicator
iU8Arr_3d	[U8]	three-dimensional array of unsigned 8 bit values indicator
iU16Arr_3d	[U16]	three-dimensional array of unsigned 16 bit values indicator
iU32Arr_3d	[U32]	three-dimensional array of unsigned 32 bit values indicator

4 About Excelitas PCO

PCO, an Excelitas Technologies® Corp. brand, is a leading specialist and Pioneer in Cameras and Optoelectronics with more than 30 years of expert knowledge and experience of developing and manufacturing high-end imaging systems. The company's cutting edge sCMOS and high-speed cameras are used in scientific and industrial research, automotive testing, quality control, metrology and a large variety of other applications all over the world.

The PCO® advanced imaging concept was conceived in the early 1980s by imaging pioneer, Dr. Emil Ott, who was conducting research at the Technical University of Munich for the Chair of Technical Electrophysics. His work there led to the establishment of PCO AG in 1987 with the introduction of the first image-intensified camera followed by the development of its proprietary Advanced Core technologies which greatly surpassed the imaging performance standards of the day.

Today, PCO continues to innovate, offering a wide range of high-performance camera technologies covering scientific, high-speed, intensified and FLIM imaging applications across the scientific research, industrial and automotive sectors.

Acquired by Excelitas Technologies in 2021, PCO represents a world renowned brand of high-performance scientific CMOS, sCMOS, CCD and high-speed cameras that complement Excelitas' expansive range of illumination, optical and sensor technologies and extend the bounds of our end-to-end photonic solutions capabilities.

pco.

An Excelitas Technologies Brand

pco.

An Excelitas Technologies Brand

telephone:

+ 49 (0) 9441 2005 50

fax:

+ 49 (0) 9441 2005 20

postal address:

Excelitas PCO GmbH

Donaupark 11

93309 Kelheim, Germany

email:

pco@excelitas.com

web:

www.pco.de

www.excelitas.com

