user manual

pco.labview

excelitas.com

excelitas®

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.

address:	Excelitas PCO GmbH Donaupark 11 93309 Kelheim, Germany
phone:	(+49) 9441-2005-0 (+1) 86-662-6653 (+86) 0512-6763-4643
mail:	pco@excelitas.com
web:	www.excelitas.com/pco

pco.labview user manual 5.5.0

Released May 2025

Contents

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 SP1 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

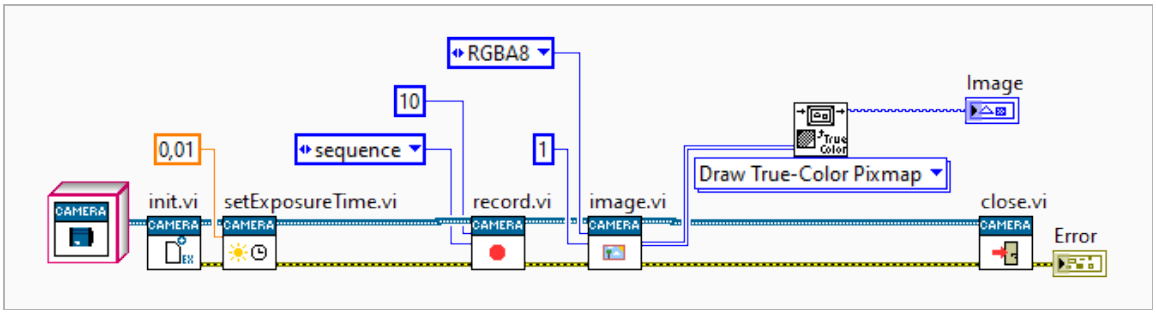
Download the windows installer, unzip and execute it. Simply follow the steps in the installer.

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 DLL's from the correct path

1.2 Basic Usage

This snippet shows the basic usage.



[1] The 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, other save images directly to file(s) on the disk and some are recording and reading directly into and from camera internal memory. However, for all modes, the recorded images can be accessed in the same way, just as they would be in memory.

Mode	Storage	Blocking	Description
`sequence`	Memory	yes	Record a sequence of images
`sequence_non_blocking`	Memory	no	Record a sequence of images, do not wait until record is finished
`ring_buffer`	Memory	no	Continuously record images in a ringbuffer, once the buffer is full, old images are overwritten
`fifo`	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
`sequence_dpcore`	Memory	yes	Same as `sequence`, but with DotPhoton preparation enabled
`sequence_non_blocking_dpcore`	Memory	no	Same as `sequence_non_blocking`, but with DotPhoton preparation enabled
`ring_buffer_dpcore`	Memory	no	Same as `ring_buffer`, but with DotPhoton preparation enabled

Continued on next page

Continued from previous page

Mode	Storage	Blocking	Description
`fifo_dpcore`	Memory	no	Same as `fifo`, but with DotPhoton preparation enabled
`tif`	File	no	Record images directly as tif files
`multitif`	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)
`camram_segement`	Camera RAM	no	Record images to camera memory. Stops when segment is full
`camram_ring`	Camera RAM	no	Record images to camera memory. Ram segment is used as ring buffer

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 2.1.33, 2.1.34, 2.1.35) of the `Camera` class. Depending on the selected format only one of the possible image outputs will be vaild, 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 `RGBA8` but with flipped color channels (provided only for completeness, in LabVIEW typically `RGBA8` is used)
`RGBA16`	Get image as 64 bit color data, use **image(U64)** output

Continued on next page

Continued from previous page

Format	Description
BGRA16	Same as RGBA16 but with flipped color channels (provided only for completeness, in LabVIEW typically RGBA16 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 2.1.21) or loadlut (see 2.1.22) from the Camera class.

1.5 Creating Executables

If you have the LabVIEW Application Builder toolbox, you can also create standalone executables. This is described on several NI websites and is unrelated to this SDK.
An executable file can be created from all examples as well as all custom applications that were created using the pco.labview package without restrictions.

Note The paths to PCO DLL's are designed so that an application can be created without having to worry about PCO dependencies. After completion of the creation, the PCO DLL files from runtime/bin64 just have to be copied to the folder in which your created **.exe* file is located.

1.6 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

The following list 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
- **configureHWIO_*_***()** Configure the HWIO channels (1-4)
- **autoExposureOn(), autoExposureOff()** Switch auto exposure on/off
- **configureAutoExposure()** Set the parameters for auto exposure calculations
- **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)
- **hasRam()** Check if camera has internal memory for recording with camram
- **switchToCamRam()** Set the camram segment where the images should be written to/read from
- **getCamRamSegment()** Get segment number of the active segment
- **getCamRamMaxImages()** Get number of images that can be stored in the active segment
- **getCamRamNumImages()** Get number of images that are available in the active segment
- **setCamRamAllocation()** Set allocation distribution of camram segments

2.1.1 init

Description

Open communication and initialize a camera. With the ”All interfaces” option, this VI will scan through all possible interfaces until it finds a connected camera.

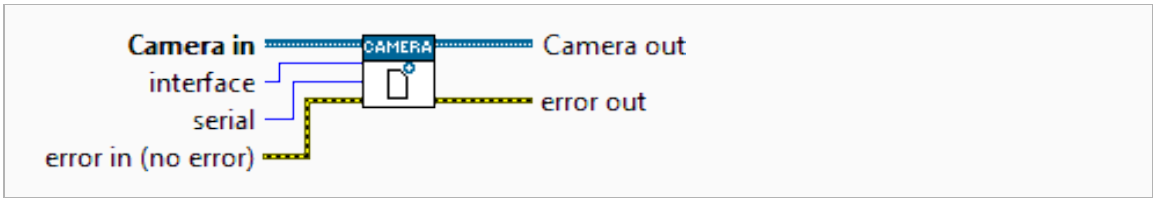
Optionally you can specify either which interface you want to look at or the serial number of the camera you want to open or both.

An error is generated if no camera is found. The camera is set to a defult configuration by this VI.

Note for windows

If you specify a serial number to be opened, we recommend to also specify the interface as this reduces the time for the function call.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initial Camera object, a constant can be used here
cU16	interface	Specific interface to search for cameras. If not connected, search on all interfaces.
cU32	serial	Search for the camera with this specific serial number. If not connected, search for any camera.
cErrClst	error in	Previous error state (no error if not connected)

Parameter OUT

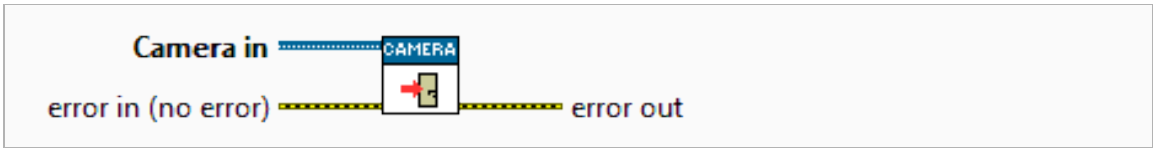
Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.2 close

Description

Close the activated camera and release 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)

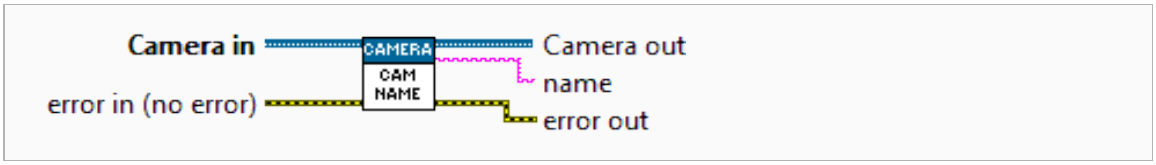
Parameter OUT

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 connected 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)

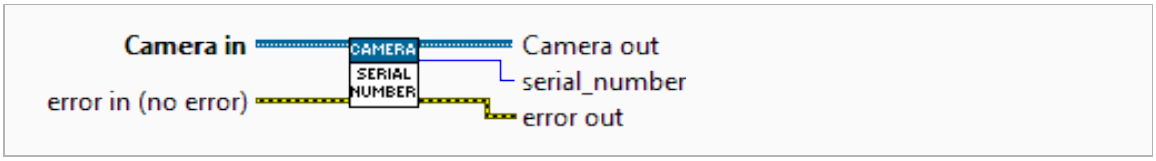
Parameter OUT

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)

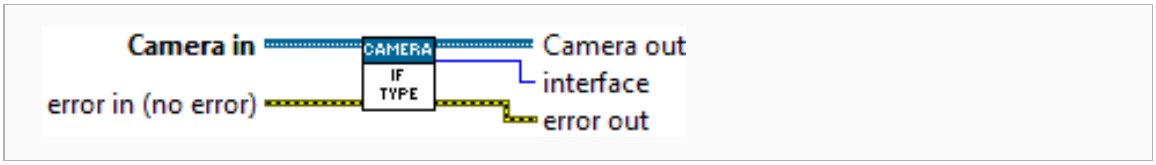
Parameter OUT

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 getInterface

Description Get the interface of the connected 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)

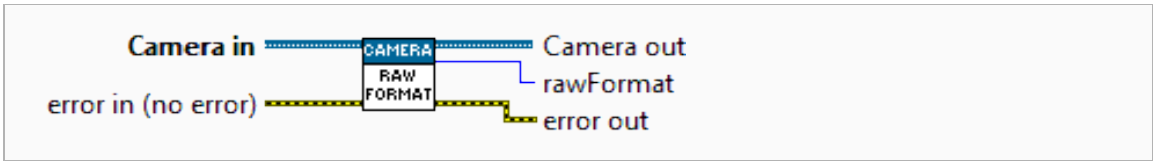
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iU16	interface	Interface of the connected 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)

2.1.6 getRawFormat

Description Get the current raw pixel format

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)

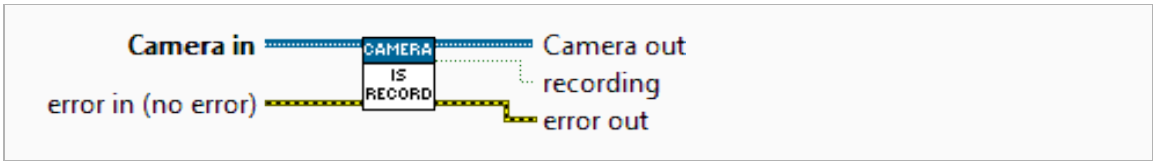
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iEnum	rawFormat	Current raw pixel format (can be Word or Byte)
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 isRecording

Description Determine if the camera is recording.

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)

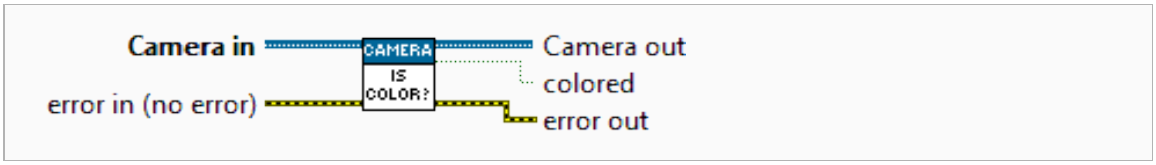
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iBool	recording	Inticates the recording state: TRUE - camera is recording FALSE - Camera is not recording
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 isColored

Description Determine if the camera has a color or monochrome sensor.

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)

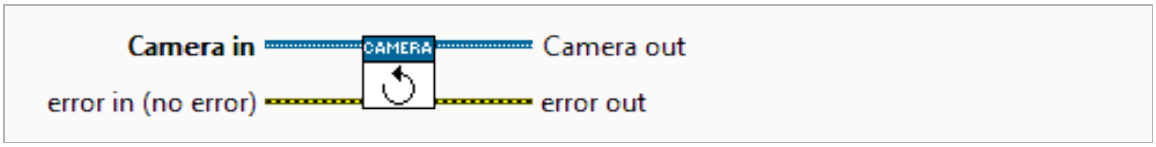
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iBool	colored	TRUE if camera has a color sensor
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 defaultConfiguration

Description Set the camera to it's default configuration. This function is also called during initialization inside init.vi and init_ex.vi.

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)

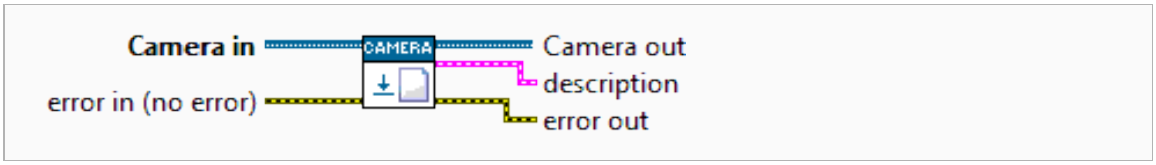
Parameter OUT

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)

2.1.10 getDescription

Description Get the camera description from 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)

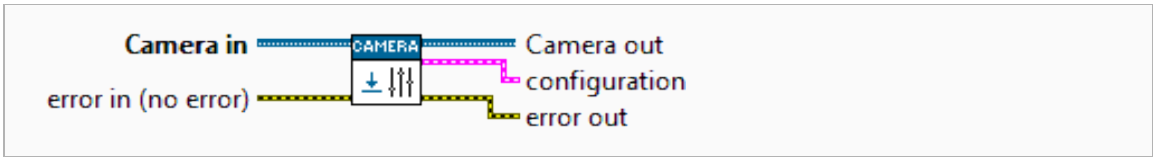
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iClst	description	Cluster containing data on the camera properties and capabilities (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)

2.1.11 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)

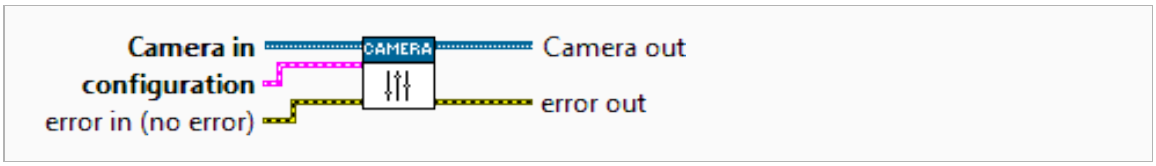
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iClst	configuration	Cluster containing the most important camera parameters, all combined in one configuration (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)

2.1.12 setConfiguration

Description Set a new coniguration to the camera.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cClst	configuration	Cluster containing the most important camera parameters, all combined in one configuration (see 2.2.4)
cErrClst	error in	Previous error state (no error if not connected)

Parameter OUT

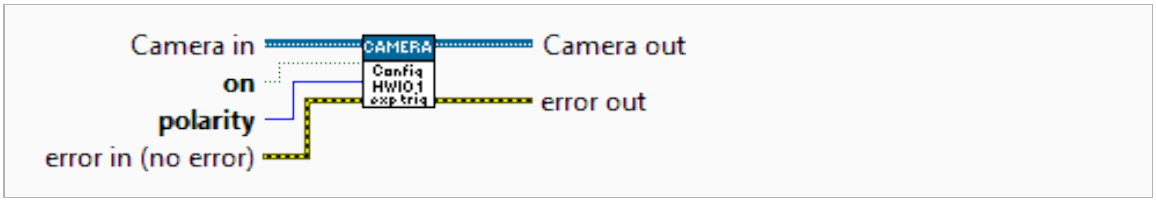
Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.13 configureHWIO_1_exposureTrigger

Description Configure the HWIO connector 1.

This connector is used for the exposure trigger signal input

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cBool	on	Flag if the HWIO connector should be enabled or disabled
cU16	polarity	Polarity the connector should react on (HWIO_EdgePolarity, see 2.1.16.1)
cErrClst	error in	Previous error state (no error if not connected)

Parameter OUT

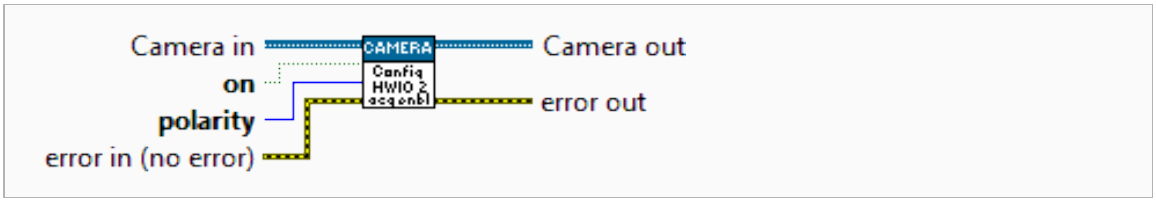
Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.14 configureHWIO_2_acquireEnable

Description Configure the HWIO connector 2.

This connector is used for the acquire enable signal input.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cBool	on	Flag if the HWIO connector should be enabled or disabled
cU16	polarity	Polarity the connector should have (HWIO_Polarity, see 2.1.16.1)
cErrClst	error in	Previous error state (no error if not connected)

Parameter OUT

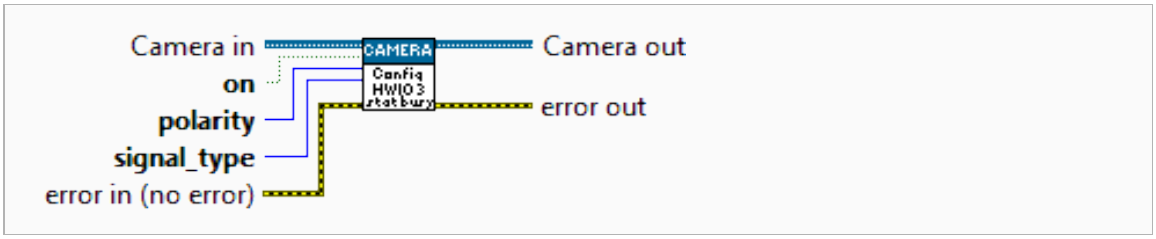
Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.15 configureHWIO_3_statusBusy

Description Configure the HWIO connector 3.

This connector is typically used for the status busy output of the camera. Depending on the camera it can also be configured to output different kind of signals, which can be selected by the signal_type parameter.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cBool	on	Flag if the HWIO connector should be enabled or disabled
cU16	polarity	Polarity the connector should have (HWIO_Polarity, see 2.1.16.1)
cU32	signal_type	Type of the signal the connector should have (HWIO_3_SignalType, see 2.1.16.1)
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)

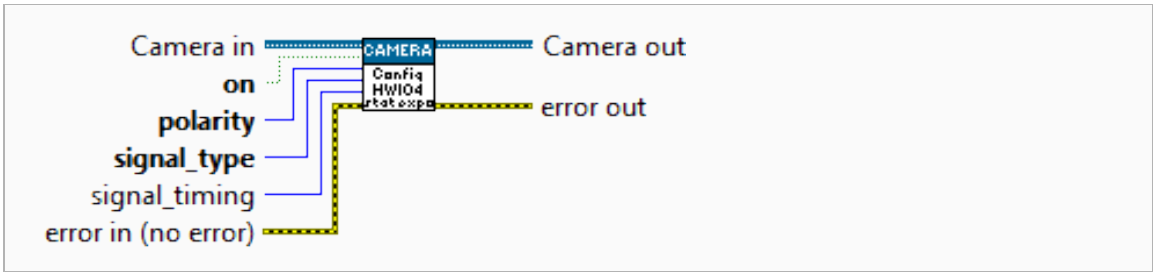
Note Even if you select a signal_type that is not valid, i.e. error out shows an error, the on and polarity parameters are set anyway.

2.1.16 configureHWIO_4_statusExpos

Description Configure the HWIO connector 4.

This connector is typically used for the status exposure output of the camera. Depending on the camera it can also be configured to output different kind of signals, selected by the `signal_type` parameter. In some cases, different timing modes for the exposure output signal can be selected by the `signal_timing` parameter.

Prototype



Parameter IN

Datatype	Name	Description
`cCamera`	`Camera in`	Initialized `Camera` object controlling an opened camera
`cBool`	`on`	Flag if the HWIO connector should be enabled or disabled
`cU16`	`polarity`	Polarity the connector should have (`HWIO_Polarity`, see 2.1.16.1)
`cU32`	`signal_type`	Type of the signal the connector should have (`HWIO_4_SignalType`, see 2.1.16.1)
`cU32`	`signal_timing`	Timing of exposure output signal (see 2.1.16.1). Only valid for Rolling Shutter cameras and `signal_type status_expos` (default is `undefined`, i.e. will not 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)

Note Even if you select a `signal_type` that is not valid, i.e. `error out` shows an error, the `on` and `polarity` parameters are set anyway.

2.1.16.1 HWIO Types

For the **configureHWIO_****** functions we have the following ring enum definitions:

HWIO_Polarity This is a ring enum with the following possible items:

Item	Value	Description
high level	1	I/O signal line can be sensed for high level
low level	2	I/O signal line can be sensed for low level

HWIO_-EdgePolarity

This is a ring enum with the following possible items:

Item	Value	Description
rising edge	4	I/O signal line can be sensed for rising edges
falling edge	8	I/O signal line can be sensed for falling edges

HWIO_3_-SignalType

This is a ring enum with the following possible items:

Item	Value	Description
status busy	0	Signal is output for camera busy state
status line	2	Signal is output for the internal periodical line time of the (rolling shutter) sensor
status armed	3	Signal is output for camera arm state

HWIO_4_-SignalType

This is a ring enum with the following possible items:

Item	Value	Description
status expos	1	Signal is output for camera exposing state
status line	2	Signal is output for the internal periodical line time of the (rolling shutter) sensor
status armed	3	Signal is output for camera arm state

HWIO_-StatusExpos_-Timing

This is a ring enum with the following possible items:

Item	Value	Description
undefined	0	Ignore this parameter
first line	1	Exposure time of the first rolling shutter line ($t_{firstline}$)
global	2	Core time while all lines are exposing (t_{global})
last line	3	Exposure time of the last rolling shutter line ($t_{lastline}$)
all lines	4	Complete exposure time from the start of first until the end of the last rolling shutter line ($t_{alllines}$)

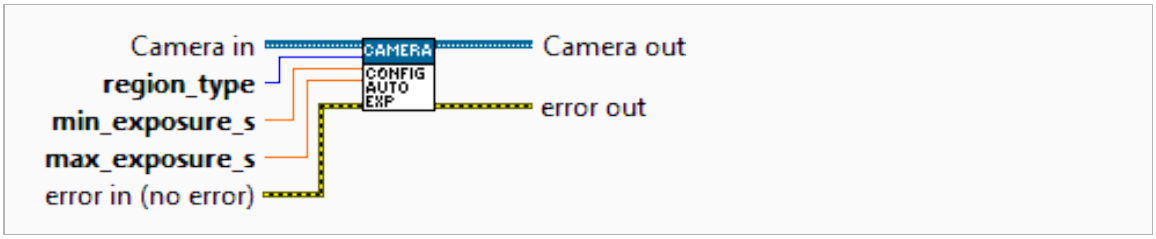
2.1.17 configureAutoExposure

Description Set the auto exposure parameters.

This does not activate or deactivate the auto exposure functionality.
For this please use `autoExposureOn.vi` and `autoExposureOff.vi`.

Note While `autoExposureOn.vi` and `autoExposureOff.vi` can be called also during record, this function can only be called when recording is off.

Prototype



Parameter IN

Datatype	Name	Description
`cCamera`	`Camera in`	Initialized `Camera` object controlling an opened camera
`cU16`	`region`	Image region type that should be used for auto exposure computation (see 2.2.1).
`cDbl`	`min_exposure_s`	Minimum exposure value that can be used for auto exposure
`cDbl`	`max_exposure_s`	Maximum exposure value that can be used for auto exposure
`cErrClst`	`error in`	Previous error state (no error if not connected)

Parameter OUT

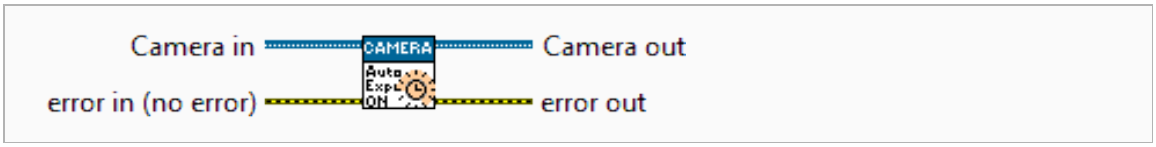
Datatype	Name	Description
`iCamera`	`Camera out`	Initialized `Camera` object controlling an opened camera
`iErrClst`	`error out`	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.18 autoExposureOn

Description Activate the auto exposure feature.

This will use the currently set configuration for auto exposure.
To set the auto exposure mode parameters please use `configureAutoExposure.vi`.

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)

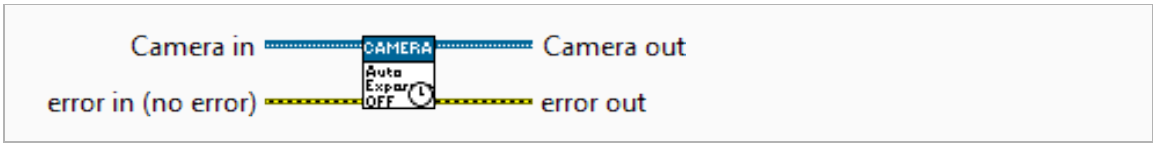
Parameter OUT

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)

2.1.19 autoExposureOff

Description Deactivate the auto exposure feature.

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)

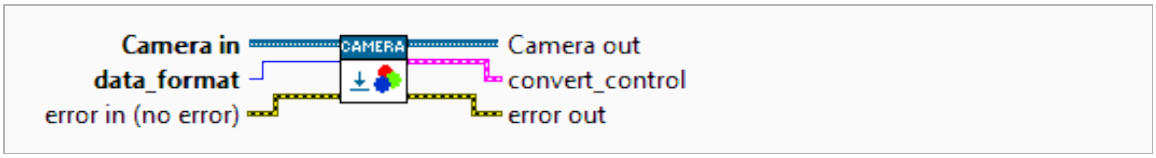
Parameter OUT

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)

2.1.20 getConvertControl

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

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cEnum	data_format	Image format for conversion of raw image
cErrClst	error in	Previous error state (no error if not connected)

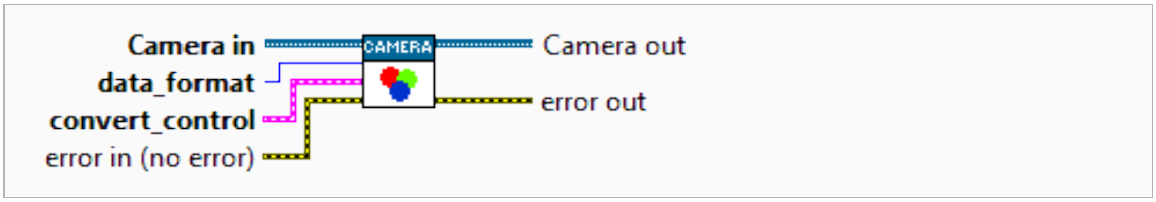
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iClst	convert_control	Cluster containing the current convert control settings (see 2.2.6)
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 setConvertControl

Description Set the paramerters for converting raw image data to other image formats.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cEnum	data_format	Image format of the converted image.
cClst	convert_control	Cluster of controls to set the conversion parameters.
cErrClst	error in	Previous error state (no error if not connected)

Parameter OUT

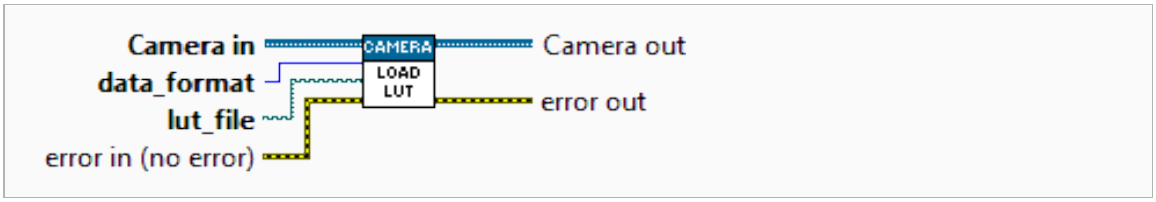
Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.22 loadLut

Description

Load a Lookup Table (LUT) from a flie. LUT's are used to apply pseudocolor to monochrome images. Note: the lut file could also be set using `setConvertControl` (see: 2.1.21).

Prototype



Parameter IN

Datatype	Name	Description
`cCamera`	`Camera in`	Initialized `Camera` object controlling an opened camera
`cEnum`	`data_format`	Format of the converted image.
`cPath`	`lut_file`	Path of the LUT file to load.
`cErrClst`	`error in`	Previous error state (no error if not connected)

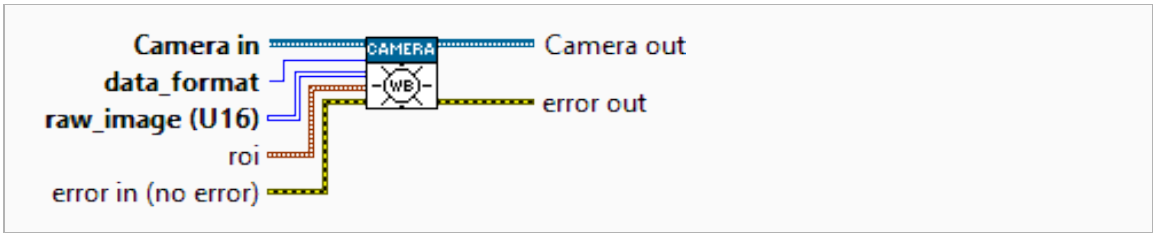
Parameter OUT

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 adaptWhiteBalance

Description Adjust the white balance of an image, based on the whole image or a specified region.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cEnum	data_format	Data format for the output image
cU16_2d	raw_image(U16)	Copied 16 bit image as 2D array of pixel values
cClstN	roi	Cluster containing a region of interest. Only this region of the image is used for white balance. x0 and y0 must be >= 1 x1 and y1 must be <= width and <= height of the image, respectively
cErrClst	error in	Previous error state (no error if not connected)

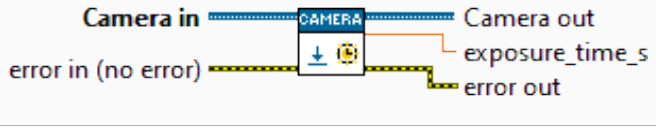
Parameter OUT

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 getExposureTime

Description Get the current exposure time of the camera in seconds.

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)

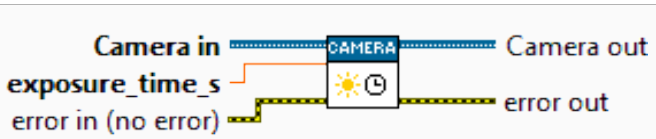
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iDbl	exposure_time_s	Current exposure time of the camera in seconds
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 setExposureTime

Description Set the exposure time of the camera, in seconds.

Prototype



Parameter IN

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

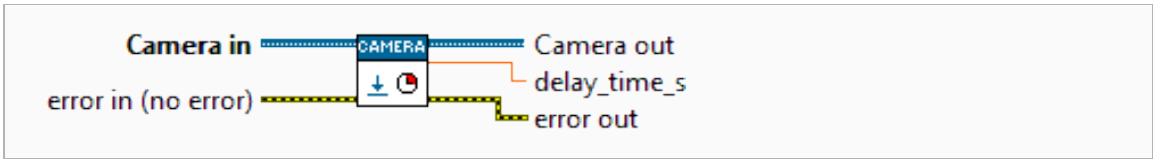
Parameter OUT

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)

2.1.26 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)

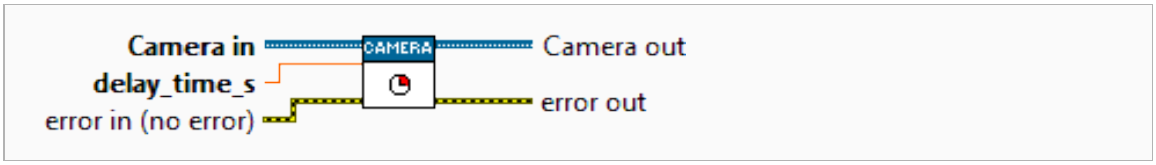
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iDbl	delay_time_s	Current delay time of the camera in seconds
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)

2.1.27 setDelayTime

Description Set the delay time of the camera, in seconds.

Prototype



Parameter IN

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

Parameter OUT

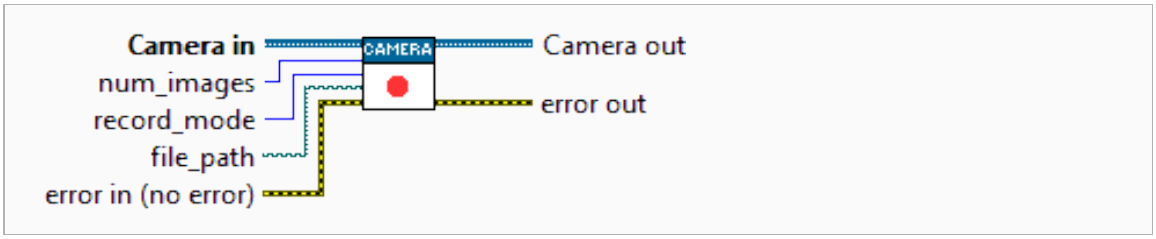
Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.28 record

Description

Create, configure, and start a new recorder instance. The entire camera configuration must be set before calling record.vi. The set_exposure_time.vi is the only exception. This function has no effect on the recorder object and 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 in the driver. The RAM, disk (of the PC) or camera RAM (depending on the mode) limits the maximum value.
cEnum	record_mode	Defines the mode for this recording (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)

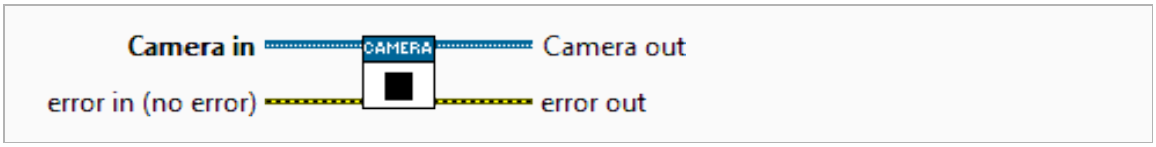
Parameter OUT

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)

2.1.29 stop

Description Stop the current recording. In 'ring buffer' and 'fifo' mode, this function must be called by the user. In 'sequence' and 'sequence non blocking' mode, this function is automatically called when the number_of_images is reached (see 1.3).

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)

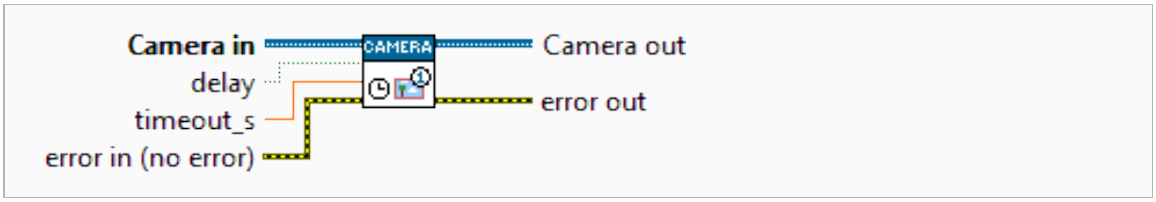
Parameter OUT

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)

2.1.30 waitForFirstImage

Description Wait for the first available image in the recorder memory. In recorder mode 'sequence non blocking', 'ring buffer'. and 'fifo', the function record.vi returns immediately. Therefore, this function can be used to wait for images from the camera before calling image.vi, images.vi, or image_average.vi.

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	Abort waiting loop if no image is 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

Continued on next page

Continued from previous page

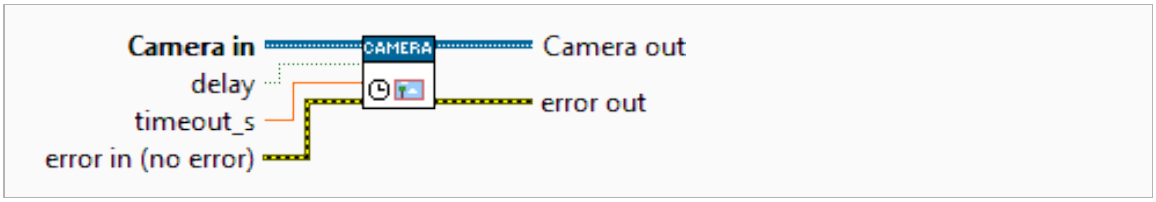
Datatype	Name	Description
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.31 waitForNewImage

Description

Wait for a new image to appear in the recorder memory (i.e. an image that has not been read yet). In recorder mode 'sequence non blocking', 'ring buffer'. and 'fifo', the function record.vi returns immediately. Therefore, this function can be used to wait for the next image from the camera before calling image.vi, images.vi, or image_average.vi.

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	Abort waiting loop if no image is recorded during timeout_s seconds.
cErrClst	error in	Previous error state (no error if not connected)

Parameter OUT

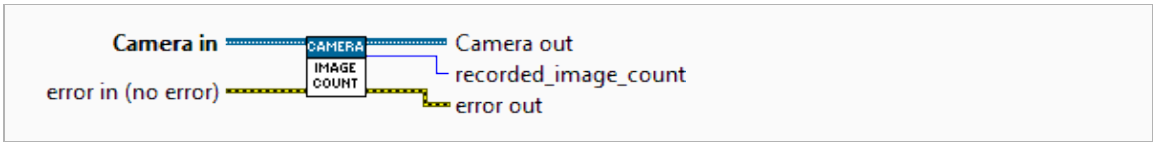
Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.32 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

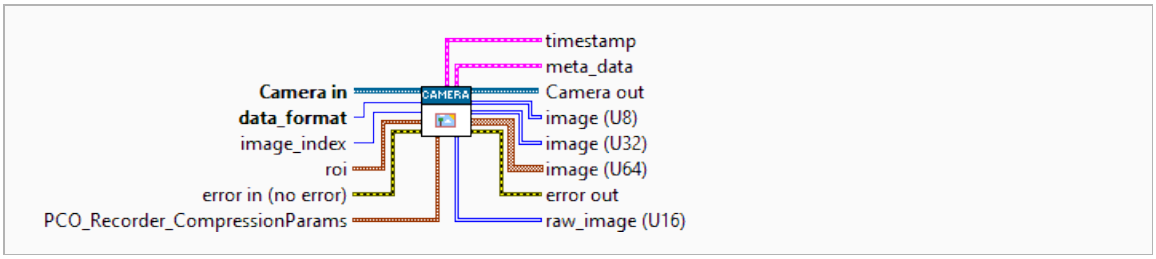
Datatype	Name	Description
`iCamera`	`Camera out`	Initialized `Camera` object controlling an opened camera
`iU32`	`recorded_image_count`	Number of currently recorded images
`iErrClst`	`error out`	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter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

2.1.33 image

Description

Get an image from the recorder. The type of the image is 2D array of pixel values. This array is shaped depending on the resolution and ROI of the image. The data format determines which output is valid, but 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 of the converted image (see 1.4)
`cU32`	`image_index`	Specifies the index of the image to be read. In 'sequence' or 'sequence non blocking' mode, the recorder index matches the image_index. If image_index is set to `0xFFFFFFFF`, the last recorded image is copied. This enables a live preview while recording. For modes `fifo`/`fifo_dpcore` always use 0 (see 1.3))
`cClstN`	`roi`	Cluster containing a region of interest. Only this region of the image is copied. (see 2.2.3 for the `Roi` structure)
`cErrClst`	`error in`	Previous error state (no error if not connected)
`cClstN`	`PCO_Recorder_CompressionParams`	Struct containing the necessary noise parameters for the compression / equilibration of the specific camera, not implemented yet

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 `Mono8`, see 1.4)

Continued on next page

Continued from previous page

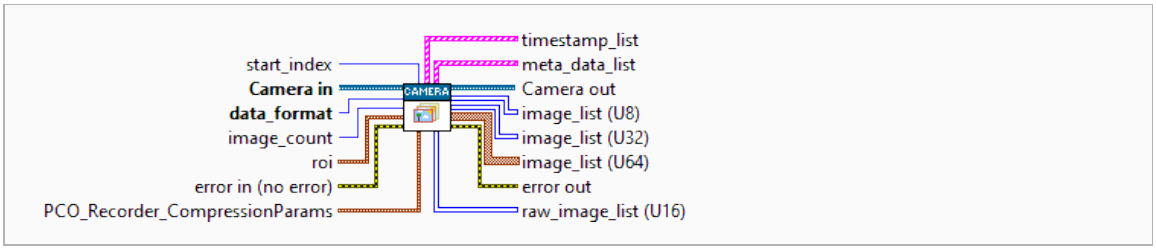
Datatype	Name	Description
iU32Arr_2d	image(U32)	Image data as 2d 32 bit array (use this for formats RGBA8, BGRA8, see 1.4)
iClstArrN_2d	image(U64)	Image data as 2d 64 bit (cluster of 4 times 116 bit) array (use this for formats RGBA16, BGRA16, see 1.4)
iErrClst	error out	Error state after VI: Error state 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)	Image data as 2d 16 bit array (use this for getting the raw image data, in format Mono16 this is the only valid image output, see 1.4)

2.1.34 images

Description

Get recorded images from the recorder as a 3D array, as specified by a start_index and an image_count. This array is shaped depending on the resolution and ROI of the image and the number of recorded images. The data is stored in the order of image, row, column. The data format determines which output is valid, but the raw_image(16) output is always valid.

Prototype



Parameter IN

Datatype	Name	Description
cU32	start_index	Index of the first image to copy
cCamera	Camera in	Initialized Camera object controlling an opened camera
cEnum	data_format	Data format of the output image array (see 1.4)
cU32	image_count	Number of images to copy
cClstN	roi	Cluster containing a region of interest. Only this region of the image is copied (see 2.2.3 for the Roi structure)
cErrClst	error in	Previous error state (no error if not connected)
cClstN	PCO_Recorder_CompressionParams	Struct containing the necessary noise parameters for the compression / equilibration of the specific camera (not implemented yet)

Parameter OUT

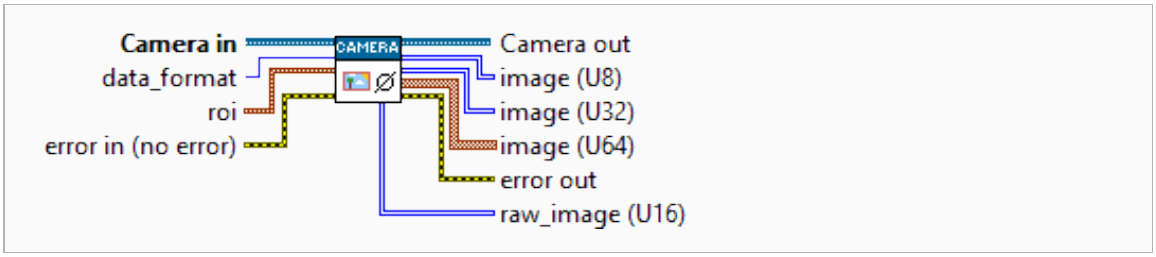
Datatype	Name	Description
iClstArr_1d	timestamp_list	Array of timestamps for the images
iClstArr_1d	meta_data_list	Array of metadata for the images
iCamera	Camera out	Initialized Camera object controlling an opened camera
iU8Arr_2d	image_list(U8)	List of images as 3d 8 bit array (use this for format Mono8, see 1.4)
iU32Arr_2d	image_list(U32)	List of images as 3d 32 bit array (use this for formats RGBA8, BGRA8, 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 RGBA16, BGRA16, 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_list(U16)	List of images as 3d 16 bit array (use this for getting the raw images, in format Mono16 this is the only valid image output, see 1.4)

2.1.35 imageAverage

Description Get the averaged image as a 2D array of pixel values. The average is calculated from all recorded images in the buffer. This array is shaped depending on the resolution and ROI of the image. The data format determines which output is valid, but 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 of the averaged image (see 1.4)
cClstN	roi	Cluster containing a region of interest. Only this region of the image is copied. (see 2.2.3 for the Roi structure)
cErrClst	error in	Previous error state (no error if not connected)

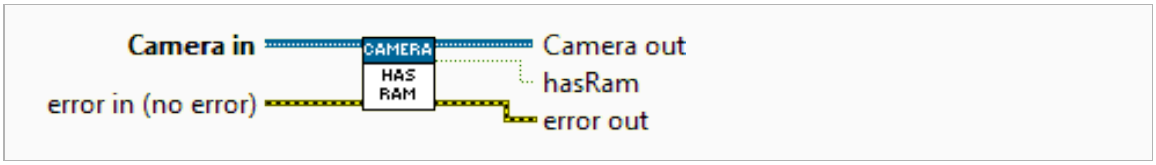
Parameter OUT

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 Mono8, see 1.4)
iU32Arr_2d	image(U32)	Averaged image as 2d 32 bit array (use this for formats RGBA8, BGRA8, see 1.4)
iClstArrN_2d	image(U64)	Averaged image as 2d 64 bit (cluster of 4 times 116 bit) array (use this for formats RGBA16, BGRA16, 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 Mono16 this is the only valid image output, see 1.4)

2.1.36 hasRam

Description Determine if the camera has internal memory (camera RAM)

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)

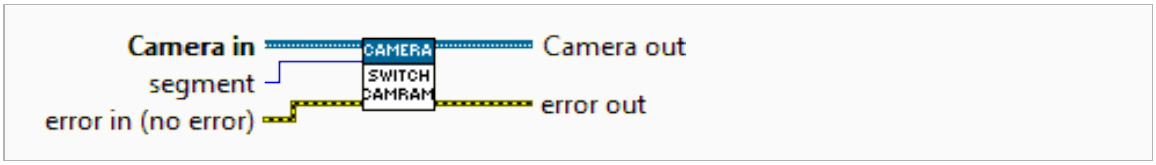
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iBool	hasRam	Boolean indicating whether cam ram is available. Set if camera has internal memory
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)

2.1.37 switchToCamRam

Description Select the active RAM segment in cameras with internal memory (CAMRAM).

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cU16	segment	RAM segment to use for recording and image readout.
cErrClst	error in	Previous error state (no error if not connected)

Parameter OUT

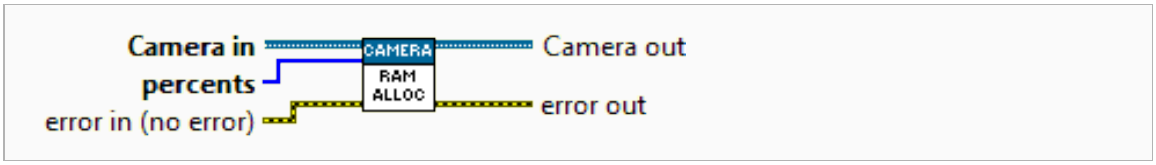
Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.38 setCamRamAllocation

Description

For cameras with internal memory (CAMRAM) this VI can be used to set the number of images allocated to each of the four segments. Allocation is specified as a percentage of the total, so the sum of the allocation elements cannot exceed 100. The total can be less than 100, i.e. not all memory is allocated.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cU32Arr_1d	percents	Array of four elements corresponding to the four memory segments, specifying the percentage of the total RAM in each segment. Note: Segments are index starting at 1, so element 0 here is segment 1, element 1 is segment 2, etc
cErrClst	error in	Previous error state (no error if not connected)

Parameter OUT

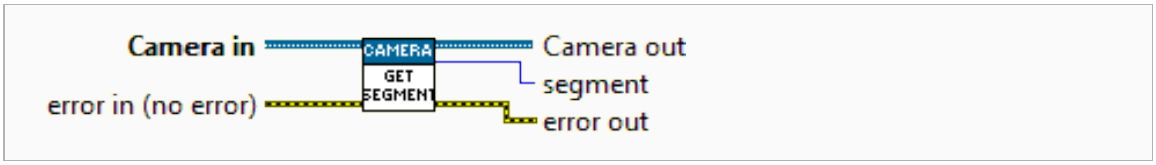
Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.1.39 getCamRamSegment

Description

Get the number of the current RAM segment. Return an error if camera does not have internal memory.

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
iU16	segment	Number of active camram segment

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)

2.1.40 setCamRamSegment

Description Set the number of the current CAMRAM segment. Subsequent recordings will use this segment, and images will be read from this segment. For cameras with internal memory only.

Prototype

Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
iU16	segment	Specifies the segement to use for recording or reading images.
cErrClst	error in	Previous error state (no error if not connected)

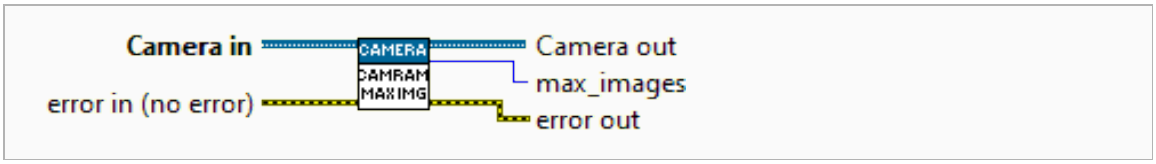
Parameter OUT

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)

2.1.41 getCamRamMaxImages

Description Determine the maximum number of images that can be stored in the camera RAM. For cameras with internal memory only. Return an error if camera does not have internal memory.

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)

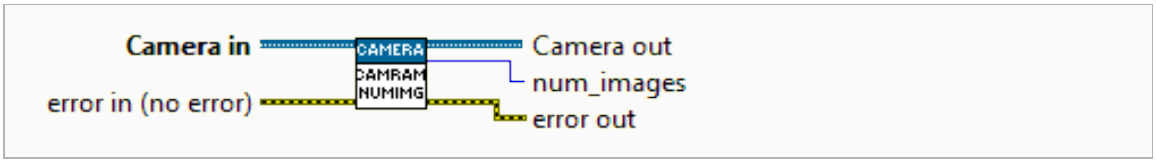
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iU32	max_images	Maximum number of images that can be stored in the camera RAM. For cameras with internal memory only
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)

2.1.42 getCamRamNumImages

Description Return the number of images stored in the current camera RAM segment. For cameras with internal memory only. Return an error if camera does not have internal memory

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)

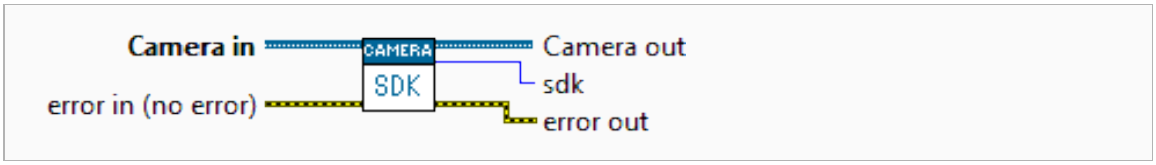
Parameter OUT

Datatype	Name	Description
iCamera	Camera out	Initialized Camera object controlling an opened camera
iU32	num_images	Number of images stored in the current camera RAM segment
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)

2.1.43 sdk

Description Retrieve the sdk handle. This handle allows direct access to all underlying functions of the pco.sdk.

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)

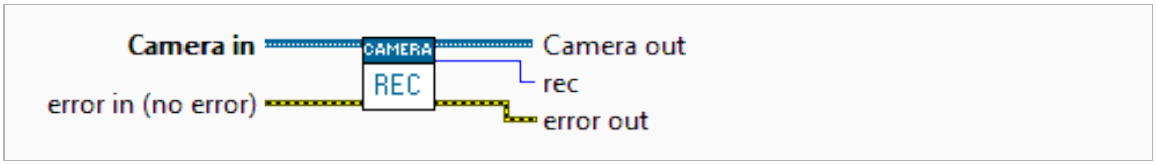
Parameter OUT

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)

2.1.44 rec

Description Retrieve the recorder handle. This handle allows direct access to all underlying functions of the pco.recorder

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)

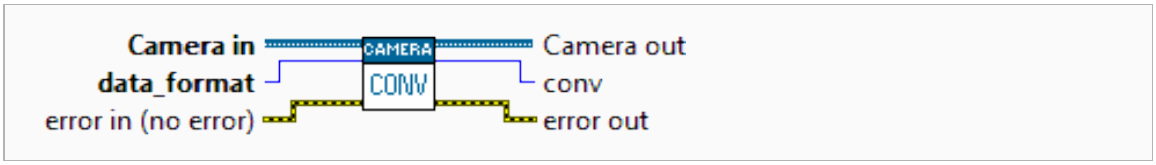
Parameter OUT

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)

2.1.45 conv

Description Retrieve the conv handle. This handle allows direct access to all underlying functions of the color conversion library pco.convert.

Prototype



Parameter IN

Datatype	Name	Description
cCamera	Camera in	Initialized Camera object controlling an opened camera
cEnum	data_format	Image data format for the convert functions
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 AutoExposure

Description Structure holding the auto exposure information.

Datatype	Name	Description
`U16`	`region`	Region type that should be used for auto exposure calculation (see below for explanation)
`Dbl`	`min_exposure_s`	Minimum exposure value that can be used for auto exposure
`Dbl`	`max_exposure_s`	Maximum exposure value that can be used for auto exposure

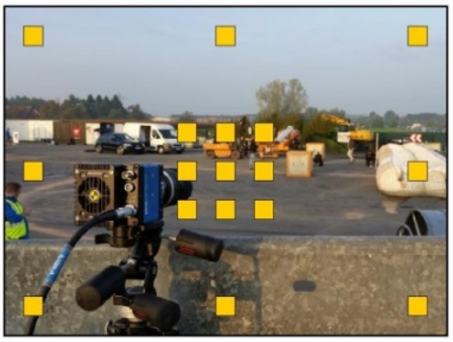
Note `region` is a ring enum with the following possible items:

Item	Value
balanced	0
center_based	1
corner_based	2
full	3

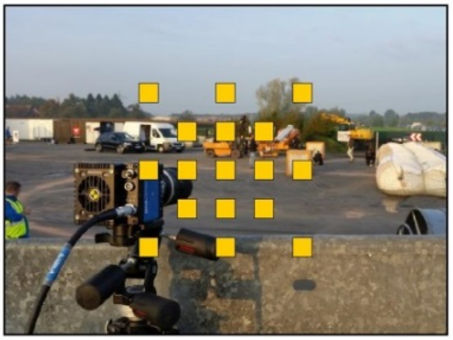
The size of the pixel clusters is fixed, but depends on the overall image size and is treated seperately for width and height:

- For width/height >= 1300 the cluster size is 100
- For 1300 > width/height >= 650 the cluster size is 50
- For 650 > width/height >= 325 the cluster size is 25
- For width/height < 325 the cluster size equal to width/height

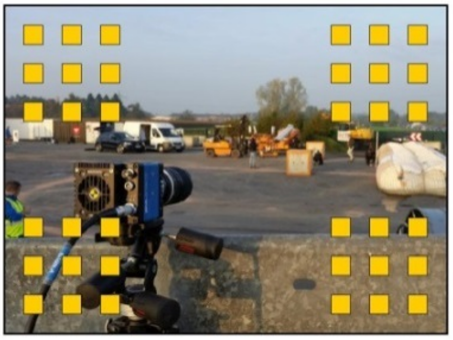
balanced Measurement fields positioned centrally and in all corners



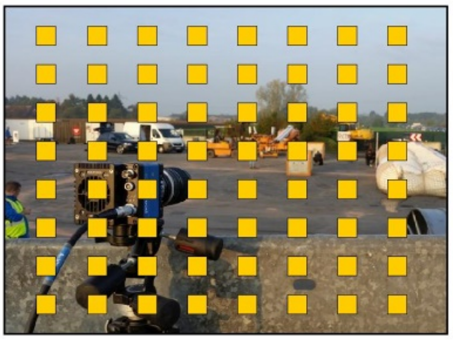
center_based Measurement fields positioned centrally.



corner_based Measurement fields positioned in all four corners.



full Measurement fields across the image.



2.2.2 Binning

Description Structure holding the binning information.

Datatype	Name	Description
U16	vert	Vertical binning
U16	horz	Horizontal binning
U16	mode	Binning mode (default is 0 which means sum)

Note mode is a ring enum with the following possible items:

Item	Value	Description
sum	0	Sum up the pixel
average	1	Average over the pixel

2.2.3 Roi

Description The roi.ctl element contains the region of interest parameters.

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)

2.2.4 Configuration

Description Cluster containing the most important camera parameters, all combined in one configuration.

Datatype	Name	Description
Dbl	exposure_time_s	Exposure time of the camera in seconds
Dbl	delay_time_s	Delay time of the camera in seconds
ClstN	roi	Cluster containing the region of interest of the camera (see 2.2.3)
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	Cluster of three U16 elements containing the binning setting of the camera(see 2.2.2)
ClstN	auto_exposure	Auto Exposure structure (see 2.2.1)

2.2.5 Description

Description Cluster of indicators containing the camera description. The camera description lists important camera properties, such as resolution, exposure time range, clock rates, etc.

Datatype	Name	Description
U32	serial	Serial number of the camera
U16	type	Sensor type used in the camera
U16	sub_type	Sub-type of the camera sensor
U16	interface_type	Interface used to connect the camera to the PC
U16	bit_resolution	Number of bits used in the conversion to digital
Dbl	min_exposure_time_s	Minimum exposure time in seconds
Dbl	max_exposure_time_s	Maximum exposure time in seconds
Dbl	min_exposure_step_s	Smallest increment of exposure time in seconds
Dbl	min_delay_time_s	Minimum delay time in seconds
Dbl	max_delay_time_s	Maximum delay time in seconds
Dbl	min_delay_step_s	Minimum increment for setting delay time, in seconds
U64	min_width	Minimum image width in pixels (hardware ROI)
U64	min_height	Minimum image height in pixels (hardware ROI)
U64	max_width	Maximum image width in pixels (hardware ROI)
U64	max_height	Maximum image height in pixels (hardware ROI)
U64	roi_step_horz	Minimum increment for setting horizontal ROI (hardware ROI)
U64	roi_step_vert	Minimum increment for setting vertical ROI (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	Array containing all possible values of the pixel clock, in Hz. Index 0 is default value (index 0 is default)

Continued on next page

Continued from previous page

Datatype	Name	Description
Bool	has_trigger_mode_extexpctrl	Flag if camera supports trigger mode external exposure control
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
Bool	has_ram	Flag if camera has internal memory (CAMRAM)
U16Arr_1d	binning_horz_vec	Array containing all possible horizontal binning values
U16Arr_1d	binning_vert_vec	Array containing all possible vertical binning values
Bool	has_average_binning	Flag if camera supports average binning

2.2.6 ConvertControl

Description Cluster of controls for the conversion of raw images to other image formats.

Datatype	Name	Description
Bool	sharpen	Set if the image should be sharpened
Bool	adaptive_sharpen	Set to enable adaptive sharpening
Bool	flip_vertical	Set to flip output image vertically with respect to input image
Bool	auto_minmax	Set to enable auto scaling
U16	add_conv_flags	Variable to set additional flags for image/color conversion (default is 0)
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	Set to enable PCO debayering
I32	color_temperature	Color temperature for image conversion
I32	color_saturation	Color saturation for image conversion
I32	color_vibrance	Color vibrance for image conversion
I32	color_tint	Color tint for image conversion
Path	lut_file	Path to the lut file for converting mono images

2.3 XCite

The following sections describe the XCite class to control X-Cite® devices within LabbVIEW.

With this class you can easily create and control a system with PCO cameras and Excelitas X-Cite® light sources.

2.3.1 init

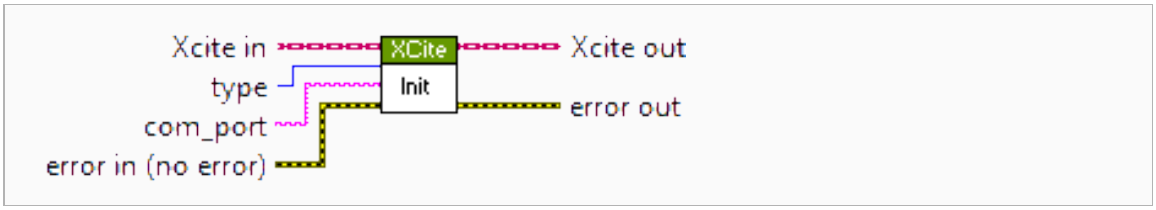
Description

Initialize the connection to a X-Cite® light source. With the ”All interfaces” option, this VI will scan through all possible interfaces until it finds a connected X-Cite® light source.

Optionally one can specify either the device or the com port.

An error is generated if no X-Cite® is found.

Prototype



Parameter IN

Datatype	Name	Description
cXcite	Xcite in	Initialized XCite object controlling an opened X-Cite® light source
cI32	type	X-Cite® device type (see 2.3.13.3)
cStr	com_port	Com port string
cErrClst	error in	Previous error state (no error if not connected)

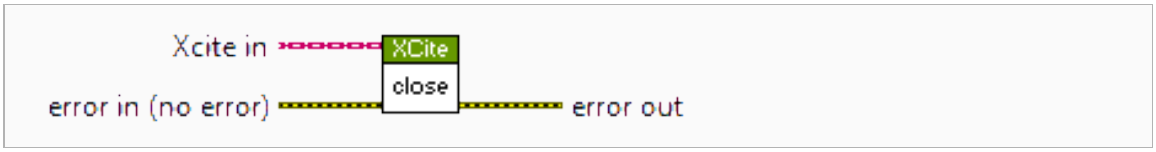
Parameter OUT

Datatype	Name	Description
iXCite	Xcite out	Initialized XCite object controlling an opened X-Cite® light source
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)

2.3.2 close

Description Close the activated connection and release the blocked resources.

Prototype



Parameter IN

Datatype	Name	Description
cXcite	Xcite in	Initialized XCite object controlling an opened X-Cite® light source
cErrClst	error in	Previous error state (no error if not connected)

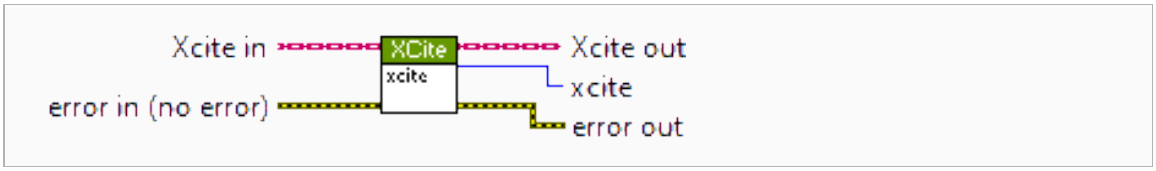
Parameter OUT

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)

2.3.3 xcite

Description Return the handler for the xcite connection for the xcite library

Prototype



Parameter IN

Datatype	Name	Description
cXcite	Xcite in	Initialized XCite object controlling an opened X-Cite® light source
cErrClst	error in	Previous error state (no error if not connected)

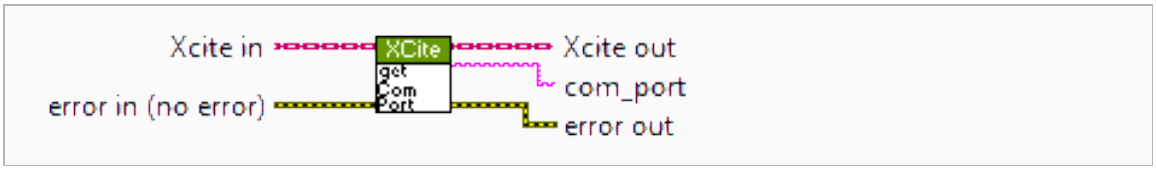
Parameter OUT

Datatype	Name	Description
iXCite	Xcite out	Initialized XCite object controlling an opened X-Cite® light source
iU64	xcite	HANDLE for the current xcite connection
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)

2.3.4 getComPort

Description Return the Com Port of the current connection

Prototype



Parameter IN

Datatype	Name	Description
cXcite	Xcite in	Initialized XCite object controlling an opened X-Cite® light source
cErrClst	error in	Previous error state (no error if not connected)

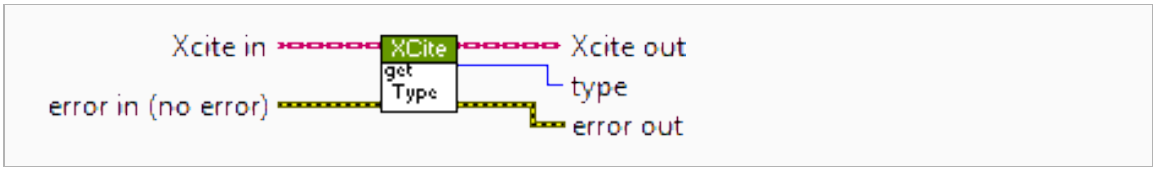
Parameter OUT

Datatype	Name	Description
iXCite	Xcite out	Initialized XCite object controlling an opened X-Cite® light source
iStr	com_port	Com port of the opened light source
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)

2.3.5 getType

Description Return the X-Cite® device type

Prototype



Parameter IN

Datatype	Name	Description
cXcite	Xcite in	Initialized XCite object controlling an opened X-Cite® light source
cErrClst	error in	Previous error state (no error if not connected)

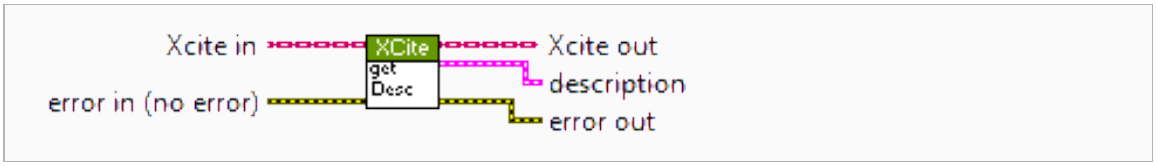
Parameter OUT

Datatype	Name	Description
iXCite	Xcite out	Initialized XCite object controlling an opened X-Cite® light source
iI32	type	X-Cite® device type (see 2.3.13.3)
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)

2.3.6 getDescription

Description Return the description parameters of the X-Cite® device

Prototype



Parameter IN

Datatype	Name	Description
cXcite	Xcite in	Initialized XCite object controlling an opened X-Cite® light source
cErrClst	error in	Previous error state (no error if not connected)

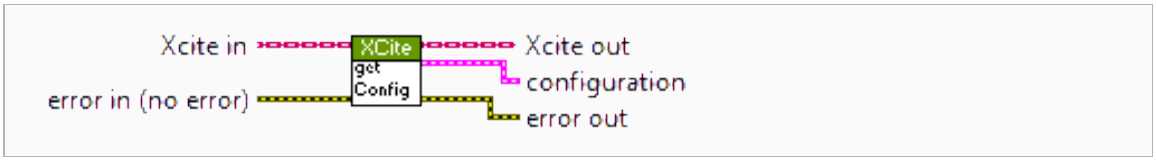
Parameter OUT

Datatype	Name	Description
iXCite	Xcite out	Initialized XCite object controlling an opened X-Cite® light source
iClst	description	Description structure of the X-Cite® device (see 2.3.13.2)
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)

2.3.7 getConfiguration

Description Return the configuration parameters of the X-Cite® device

Prototype



Parameter IN

Datatype	Name	Description
cXcite	Xcite in	Initialized XCite object controlling an opened X-Cite® light source
cErrClst	error in	Previous error state (no error if not connected)

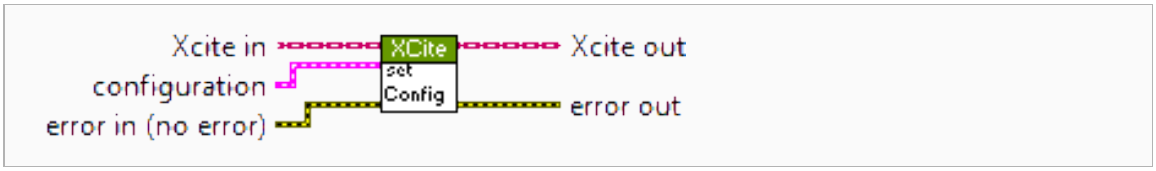
Parameter OUT

Datatype	Name	Description
iXCite	Xcite out	Initialized XCite object controlling an opened X-Cite® light source
iClst	configuration	Configuration structure of the X-Cite® device (see 2.3.13.1)
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)

2.3.8 setConfiguration

Description Write a configuration to the X-Cite® device

Prototype



Parameter IN

Datatype	Name	Description
cXcite	Xcite in	Initialized XCite object controlling an opened X-Cite® light source
cClst	configuration	Configuration structure for the X-Cite® device (see 2.3.13.1)
cErrClst	error in	Previous error state (no error if not connected)

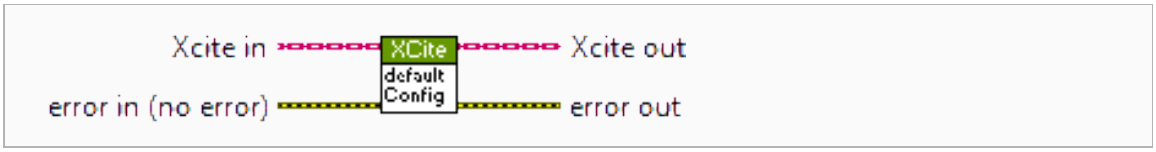
Parameter OUT

Datatype	Name	Description
iXCite	Xcite out	Initialized XCite object controlling an opened X-Cite® light source
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)

2.3.9 defaultConfiguration

Description Reset the configuration of the X-Cite® device to the default values, turns all lights off.

Prototype



Parameter IN

Datatype	Name	Description
cXcite	Xcite in	Initialized XCite object controlling an opened X-Cite® light source
cErrClst	error in	Previous error state (no error if not connected)

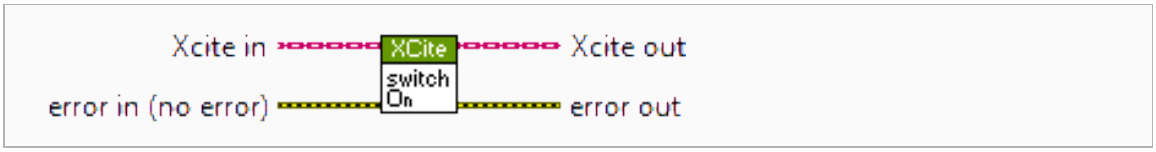
Parameter OUT

Datatype	Name	Description
iXCite	Xcite out	Initialized XCite object controlling an opened X-Cite® light source
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)

2.3.10 switchOn

Description Switch the configured lights on

Prototype



Parameter IN

Datatype	Name	Description
cXcite	Xcite in	Initialized XCite object controlling an opened X-Cite® light source
cErrClst	error in	Previous error state (no error if not connected)

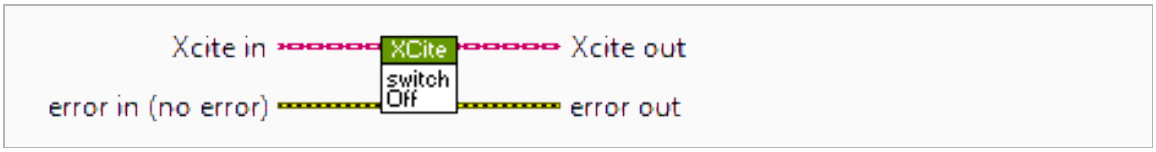
Parameter OUT

Datatype	Name	Description
iXCite	Xcite out	Initialized XCite object controlling an opened X-Cite® light source
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)

2.3.11 switchOff

Description Switch all lights off

Prototype



Parameter IN

Datatype	Name	Description
cXcite	Xcite in	Initialized XCite object controlling an opened X-Cite® light source
cErrClst	error in	Previous error state (no error if not connected)

Parameter OUT

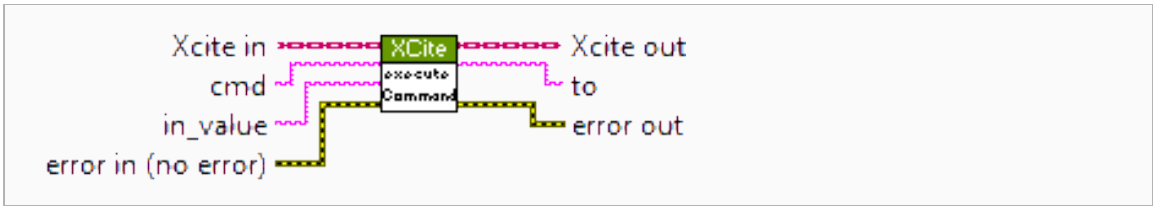
Datatype	Name	Description
iXCite	Xcite out	Initialized XCite object controlling an opened X-Cite® light source
iErrClst	error out	Error state after VI: Error state of the function (if no error was passed in) Error state of error in parameter (if error was passed in)

2.3.12 executeCommand

Description

The command list for X-Cite® is available by request. To obtain the latest update, please contact Excelitas Technologies support.

Prototype



Parameter IN

Datatype	Name	Description
cXcite	Xcite in	Initialized XCite object controlling an opened X-Cite® light source
cStr	cmd	Command string to be sent to the X-Cite® device
cStr	in_value	Parameter string if necessary for the command
cErrClst	error in	Previous error state (no error if not connected)

Parameter OUT

Datatype	Name	Description
iXCite	Xcite out	Initialized XCite object controlling an opened X-Cite® light source
iStr	to	Response string
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)

2.3.13 Controls

In the following sections you will find all structures and enums used in the `XCite` class.

2.3.13.1 XCITE_Configuration

Description Structure holding a X-Cite® configuration

Datatype	Name	Description
`U32Arr_1d`	`intensities`	Array of available intensities
`U8Arr_1d`	`on_states`	Array of which lights are on

2.3.13.2 XCITE_Description

Description Structure holding the X-Cite® description information

Datatype	Name	Description
`U32`	`serial`	Serial number
`U16`	`type`	XCite type (see 2.3.13.3)
`Str`	`name`	Name of the X-Cite® device
`U32Arr_1d`	`wavelengths_vec`	Array of available wavelengths
`U32Arr_1d`	`exclusivity_vec`	Array of values indicating which wavelengths can be set exclusively (matching wheel number). Wheel number 0: independent activation possible
`U32Arr_1d`	`intensity_max_vec`	Array of available maximum intensities
`U32Arr_1d`	`intensity_min_vec`	Array of available minimum intensities

2.3.13.3 XCiteType

Description This is a ring enum with the following possible items:

Item	Value	Light Source Name
XC_120PC	0	120PC
XC_exacte	1	exacte
XC_120LED	2	120LED
XC_110LED	3	110LED
XC_mini	4	mini
XC_XYLIS	5	XYLIS
XC_XR210	6	XR210
XC_XLED1	7	XLED1
XC_XT600	8	XT600
XC_XT900	9	XT900
Any	65535	

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
cXcite		`XCite` class object control
iXcite		`XCite` 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
cDbl		Double precision variable control
iDbl		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
iDblArr_1d		one-dimensional array of double precision values indicator
cU8Arr_1d		one-dimensional array of unsigned 8 bit values control

Continued on next page

Continued from previous page

Name	Type	Description
iU16Arr_1d		one-dimensional array of unsigned 16 bit values indicator
cU32Arr_1d		one-dimensional array of unsigned 32 bit values control
iU32Arr_1d		one-dimensional array of unsigned 32 bit values indicator
iClstArrN_2d		two-dimensional array of clusters (numeric data types) indicator
iU8Arr_2d		two-dimensional array of unsigned 8 bit values indicator
cU16_2d		two-dimensional array of unsigned 16 bit values control
iU16Arr_2d		two-dimensional array of unsigned 16 bit values indicator
iU32Arr_2d		two-dimensional array of unsigned 32 bit values indicator
iClstArrN_3d		three-dimensional array of clusters (numeric data types) indicator
iU8Arr_3d		three-dimensional array of unsigned 8 bit values indicator
iU16Arr_3d		three-dimensional array of unsigned 16 bit values indicator
iU32Arr_3d		three-dimensional array of unsigned 32 bit values indicator

4 About Excelitas PCO

Pioneering in Cameras and Optoelectronics (PCO) has been our shared philosophy since our establishment in 1987. Starting with image-intensified cameras, followed by the co-invention of the groundbreaking sCMOS sensor technology, PCO greatly surpassed the imaging performance standards of the day. Acquired by Excelitas in 2021, our PCO camera portfolio continues to forge ahead as a leader in digital imaging innovation across diverse applications such as scientific and industrial research, automotive testing, quality control, and metrology.

With sophisticated mechanical design, extensive software support, and a broad range of accessories, we deliver adaptable solutions for all demands. This adaptability extends to tailor-made firmware and custom image sensors, which allow us to develop highly specialized solutions for all our customers. PCO represents a world-renowned brand of high-performance camera systems that complement Excelitas' expansive range of illumination, optical, and sensor technologies and extend the bounds of our end-to-end photonic solutions capabilities.

Our comprehensive camera portfolio covers the entire spectrum - from deep ultraviolet (DUV) to shortwave infrared (SWIR), from long exposure to high-speed, from line scan to high-resolution area scan. Our camera systems are controlled and processed through an intuitive and powerful software suite addressing an extensive range of platforms and architectures.

pco.®

address:	Excelitas PCO GmbH Donaupark 11 93309 Kelheim, Germany
phone:	(+49) 9441-2005-0 (+1) 86-662-6653 (+86) 0512-6763-4643
mail:	pco@excelitas.com
web:	www.excelitas.com/pco

excelitas®

excelitas.com