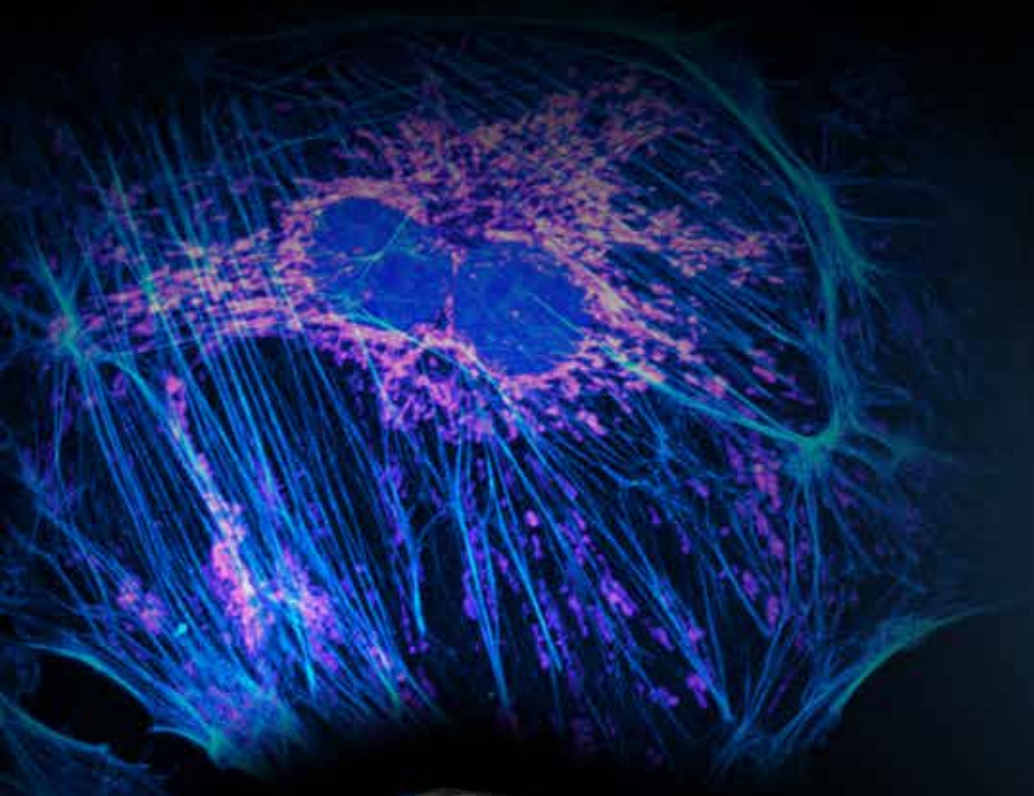
user manual

pco.lenscontrol



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) 866-662-6653 (+86) 0512-6763-4643
mail:	pco@excelitas.com
web:	www.excelitas.com/pco

pco.lenscontrol user manual 1.0.0

Released November 2025

Contents

1 General

This document describes the ***pco.lenscontrol*** library to use the lens control feature.
This feature is available in pco.dimax and pco.edge CLHS cameras (air-cooled only) and additionally requires a Birger Engineering EF-mount adapter.

For this library to communicate with the EF-mount adapter, the ***pco.sdk*** software development kit is needed to first establish a connection with the camera.

This library operates independently of the camera state, as long as the camera handle remains valid.

2 API Function Sections

2.1 PCO_LensCtrlOpen

Description Initializes a new lens control object and returns the handle.

Prototype

```
PCO_LENSCTRL_API int WINAPI PCO_LensCtrlOpen (
    HANDLE* hLens,                          //in,out
    HANDLE hCamera                          //in
);
```

Parameter

Name	Type	Description
hLens	HANDLE*	Pointer to a HANDLE: • On input the HANDLE must be set to NULL to open next available lens control object • On output a unique HANDLE is returned, if a valid connection was established
hCamera	HANDLE	Handle to a previously opened camera

Return value

Name	Type	Description
int	ErrorMessage	0 in case of success, Errorcode otherwise

2.2 PCO_LensCtrlClose

Description Closes and deletes a lens control object. The handle will be invalid afterwards.

Prototype

```
PCO_LENSCTRL_API int WINAPI PCO_LensCtrlClose (
    HANDLE hLens                            //in
);
```

Parameter

Name	Type	Description
hLens	HANDLE	Pointer to a previously created lens control object

Return value

Name	Type	Description
int	ErrorMessage	0 in case of success, Errorcode otherwise

2.3 PCO_LensCtrlGetFocus

Description Gets the current focus of the lens control device as value between 0...0x3FFF.

Prototype

```
PCO_LENSCTRL_API int WINAPI PCO_LensCtrlGetFocus (
    HANDLE hLens,                                   //in
    LONG* lFocusPos,                                //out
    DWORD* dwflags                                  //out
);
```

Parameter

Name	Type	Description
hLens	HANDLE	Pointer to a previously created lens control object
lFocusPos	LONG*	Pointer to receive the current focus position
dwflags	DWORD*	Pointer to receive status flags

Return value

Name	Type	Description
int	ErrorMessage	0 in case of success, Errorcode otherwise

2.4 PCO_LensCtrlSetFocus

Description Sets the focus of the lens control device as value between 0...0x3FFF.

Prototype

```
PCO_LENSCTRL_API int WINAPI PCO_LensCtrlSetFocus (
    HANDLE hLens,                                   //in
    LONG* lFocusPos,                                //in,out
    DWORD dwflagsin,                                //in
    DWORD* dwflagsout                               //out
);
```

Parameter

Name	Type	Description
hLens	HANDLE	Pointer to a previously created lens control object
lFocusPos	LONG*	Pointer to set the new and receive the current focus position
dwflagsin	DWORD	Variable to control the function. Set LENSCONTROL_IN_-LENSVALUE_RELATIVE to change the focus relative to the current position
dwflagsout	DWORD*	Pointer to receive status flags. LENSCONTROL_OUT_-LENSWASCHANGED indicates that the focus changed, LENSCON-TROL_OUT_LENSHITSTOP indicates that a stop was hit (either 0 or 0x3FFF)

Return value

Name	Type	Description
int	ErrorMessage	0 in case of success, Errorcode otherwise

2.5 PCO_LensCtrlGetAperture

Description Gets the current aperture position of the lens control device in steps. Position ranging from 0...max steps (dwFNumberNumStops).

Prototype

```
PCO_LENSCTRL_API int WINAPI PCO_LensCtrlGetAperture (
    HANDLE hLens,                                    //in
    WORD* wAperturePos,                              //out
    DWORD* dwflags                                   //out
);
```

Parameter

Name	Type	Description
hLens	HANDLE	Pointer to a previously created lens control object
wAperturePos	WORD*	Pointer to receive the current aperture position
dwflags	DWORD*	Pointer to receive status flags

Return value

Name	Type	Description
int	ErrorMessage	0 in case of success, Errorcode otherwise

2.6 PCO_LensCtrlSetAperture

Description Sets the current aperture position of the lens control device in steps. Position ranging from 0...max steps (dwFNumberNumStops).

Prototype

```
PCO_LENSCTRL_API int WINAPI PCO_LensCtrlSetAperture (
    HANDLE hLens,                                    //in
    WORD* wAperturePos,                              //in,out
    DWORD dwflagsin,                                 //in
    DWORD* dwflagsout                                //out
);
```

Parameter

Name	Type	Description
hLens	HANDLE	Pointer to a previously created lens control object
wAperturePos	WORD*	Pointer to set the new and receive the current aperture position
dwflagsin	DWORD	Variable to control the function. Set LENSCONTROL_IN_-LENSVALUE_RELATIVE to change the aperture relative to the current position
dwflagsout	DWORD*	Pointer to receive status flags. LENSCONTROL_OUT_-LENSWASCHANGED indicates that the aperture changed

Return value

Name	Type	Description
int	ErrorMessage	0 in case of success, Errorcode otherwise

2.7 PCO_LensCtrlGetApertureF

Description Gets the current aperture position of the lens control device in f/position * 10 (member of dwApertures). The dwApertures array is reinitialized in case the zoom changes and either PCO_GetApertureF or PCO_SetApertureF are called. Change in zoom will be shown in dwflags as LENSCONTROL_OUT_ZOOMHASCHANGED.

Prototype

```
PCO_LENSCTRL_API int WINAPI PCO_LensCtrlGetApertureF (
    HANDLE hLens,                                   //in
    DWORD* dwfAperturePos,                          //in
    WORD* wAperturePos,                             //out
    DWORD* dwflags                                  //out
);
```

Parameter

Name	Type	Description
hLens	HANDLE	Pointer to a previously created lens control object
dwfAperturePos	DWORD*	Pointer to receive the current aperture position in f/x * 10 (e.g. f/5.4 -> 54)
wAperturePos	WORD*	Pointer to receive the current aperture position; Can be NULL
dwflags	DWORD*	Pointer to receive status flags: LENSCONTROL_OUT_ZOOMHASCHANGED indicates that the dwApertures array was changed due to zoom change

Return value

Name	Type	Description
int	ErrorMessage	0 in case of success, Errorcode otherwise

2.8 PCO_LensCtrlSetApertureF

Description Sets the current aperture position of the lens control device in f/position * 10 (member of dwApertures). Please select a member of the current dwApertures array. The dwApertures array is reinitialized in case the zoom changes and either PCO_GetApertureF or PCO_SetApertureF are called. Change in zoom will be shown in dwflagsout as LENSCONTROL_OUT_ZOOMHASCHANGED.

Prototype

```
PCO_LENSCTRL_API int WINAPI PCO_LensCtrlSetApertureF (
    HANDLE hLens,                                   //in
    DWORD* dwfAperturePos,                          //in,out
    DWORD dwflagsin,                                //in
    DWORD* dwflagsout                               //out
);
```

Parameter

Name	Type	Description
hLens	HANDLE	Pointer to a previously created lens control object
dwfAperturePos	DWORD*	Pointer to set the new and receive the current aperture position in f/x * 10 (e.g. f/5.4 -> 54)
dwflagsin	DWORD	Value to set control flags

Continued on next page

Continued from previous page

Name	Type	Description
dwflagsout	DWORD*	Pointer to receive status flags: LENSCONTROL_OUT_-ZOOMHASCHANGED indicates that the dwApertures array was changed due to zoom change, LENSCONTROL_OUT_-LENSWASCHANGED indicates that the aperture changed

Return value

Name	Type	Description
int	ErrorMessage	0 in case of success, Errorcode otherwise

2.9 PCO_LensCtrlGetParameters

Description Gets parameters from a lens control object.

Prototype

```
PCO_LENSCTRL_API int WINAPI PCO_LensCtrlGetParameters (
    HANDLE hLense,                                  //in
    PCO_LensControlParameters* pstrLensParas        //out
);
```

Parameter

Name	Type	Description
hLense	HANDLE	Pointer to a previously created lens control object
pstrLensParas	PCO_LensControlParameters*	Pointer to get the current lens control parameters (see 2.10)

Return value

Name	Type	Description
int	ErrorMessage	0 in case of success, Errorcode otherwise

2.10 PCO_LensControlParameters

Name	Type	Description
wSize	WORD	Size of this structure
wHardwareVersion	WORD	Hardware version queried by *hv*
wBootloaderVersion	WORD	Bootloader version queried by *bv*
wSerialNumber	WORD	Serial number queried by *sn*
bLibraryIdentity[48]	BYTE	Full library identity string queried by *lv*
dwLENSType	DWORD	This identifies the type of the lens control (Birger=0x00B189E8)
dwStatusFlags	DWORD	LENSCONTROL_STATUS (see 2.11)
dwInitCounter	DWORD	Counts number of inits in order to reflect lens changes
		F number queried by *da*
dwFNumberMinimum	DWORD	Min aperture as f/ *10
dwFNumberNumStops	DWORD	Number of stops
dwFNumberMaximum	DWORD	Max aperture as f/ *10
		Zoom range queried by *dz*
dwZoomRangeMin	DWORD	Min zoom position
dwZoomRangeMax	DWORD	Max zoom position
dwZoomPos	DWORD	Not used, set to zero
dwLastZoomPos	DWORD	Last zoom position queried by *gs*
dwApertures[50]	DWORD	Possible aperture values in f/ * 10
dwFocalLength	DWORD	Last focal length got from lens by *lc*

Continued on next page

Continued from previous page

Name	Type	Description
lFocusMin	LONG	Focus range minimum; Usually 0
lFocusMax	LONG	Focus range maximum; Usually 16383
lFocusCurr	LONG	Focus position 0… 16383
lFocusLastCurr	LONG	Last current focus position
wAperturePos	WORD	Current aperture position
wLastAperturePos	WORD	Last current aperture position
dwfLastAperturePos	DWORD	Last aperture position as f/ * 10

2.11 Lens Control Status

The lens control statuses are defined with the following values:

Name	Value	Description
LENSCONTROL_STATUS_LA_CMD_DONE	0x00000001	Indicates command 'la' was sent to lens
LENSCONTROL_STATUS_LENSPRESENT	0x00000002	Indicates presence of a lens
LENSCONTROL_STATUS_NOAPERTURE	0x00000004	No aperture settings are possible
LENSCONTROL_STATUS_MANUALFOCUS	0x00000008	No focus settings are possible
LENSCONTROL_STATUS_WAITINGFORLENS	0x00000010	Birger is here, but no lens

2.12 PCO_GetVersion_LensCtrl

Description Get the version of the pco.lenscontrol library.

Prototype

```
PCO_LENSCTRL_API int WINAPI PCO_GetVersion_LensCtrl (
    int* piMajor,                                    //out
    int* piMinor,                                    //out
    int* piPatch,                                    //out
    int* piBuild                                     //out
);
```

Parameter

Name	Type	Description
piMajor	int*	Pointer to receive the major version
piMinor	int*	Pointer to receive the minor version
piPatch	int*	Pointer to receive the patch version
piBuild	int*	Pointer to receive the build version

Return value

Name	Type	Description
`int`	ErrorMessage	0 in case of success, Errorcode otherwise

3 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) 866-662-6653 (+86) 0512-6763-4643
mail:	pco@excelitas.com
web:	www.excelitas.com/pco

excelitas®

excelitas.com