user manual

pco.matlab

vid = videoinput('pcocameraadaptor_r2022b',0);
src = getselectedsource(vid);
src.TMTimestampMode = 'BinaryAndAscii';
triggerconfig(vid, 'immediate');
vid.FramesPerTrigger = 100;
start(vid);
images = getdata(vid);
implay(images);
pco.dicam C1
pco.edge
pco.pixelfly
pco.panda

EXCELITAS TECHNOLOGIES®

pco.
An Excelitas Technologies Brand

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

An Excelitas Technologies Brand

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

pco.matlab user manual 1.19.1

Released February 2024

Contents

1 Introduction

There are three main components in the pco.matlab package for PCO cameras:

The pco.matlab adaptor (chapter **2**) integrates your PCO camera(s) into the MATLAB Image Acquisition Toolbox. This makes it very simple for you to conveniently control your camera system and acquire images in MATLAB.

With the pco.matlab flim package (chapter **3**) you get a basic set of MATLAB functions. It helps you to compute phase and modulation depth images as well as lifetime images out of the camera's raw data. Furthermore, the pco.flim white paper supports you with detailed explanations and mathematical descriptions.

The pco.matlab scripts (chapter **4**) provide you a collection of example m-files that allow you to call functions from pco.sdk and pco.recorder directly from the MATLAB scripting language. More features of PCO cameras are accessible through the SDK than through the Image Acquisition toolbox. The examples show you how to initialize the camera, change camera settings, and grab images from an operating camera or the camera's internal memory. Grabbed images are displayed directly in a MATLAB figure window or collected in an image stack. Use MATLAB algorithms to further process the images from the image stack.

1.1 Installation

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

In your install directory (default: C:\Program Files\PCO Digital Camera Toolbox\pco.matlab) you will find:

- An **adaptor** directory containing everything releated to the image acquisition toolbox (see chapter refsec:adapterformatlab)
- A **flim** directory containing the example scripts to work with images recorded by a pco.flim camera (see chapter 3)
- An **scripts** directory containing example scripts for using pco.sdk pco.recorder SDK's directly in matlab (see chapter 4)
- A **runtime** directory with the needed binaries, libraries and header files

Linux Download the *.deb package file. Install it by using dpkg, e.g. in the command line with this command: [1]

```
    $ sudo dpkg -i pco.matlab_*.*.*_amd64.deb
```

This will install **pco.matlab** into **/opt/pco/pco.matlab**.

In your install directory (default: /opt/pco/pco.matlab) you will find:

- An **adaptor** directory containing everything related to the image acquisition toolbox (see chapter 2)
- A **flim** directory containing the example scripts to work with images recorded by a pco.flim camera (see chapter 3)
- An **scripts** directory containing example scripts for using pco.sdk pco.recorder SDK's directly in matlab (see chapter 4)
- A **runtime** directory with the needed binaries, libraries and header files

Note: On linux, currently the **pco.kaya-runtime** package causes problems when using the pco.matlab adaptor for Matlab Image Acquisition Toolbox. So in this case go to ***home/.config/pco***, open ***pco_devicemgr_param.ini*** and make sure that the KAYA interface is disabled.

```
[INTERFACES]
...
CLHS_KAYA = disable
CLHS_KAYA_SCAN = disable
...
```

[1] The dpkg package needs to be installed for this, this can be done by *sudo apt-get install dpkg*

2 Adapter for Matlab Image Acquisition Toolbox

This chapter describes the use and the features of the pco.matlab adaptor for the MATLAB Image Acquisition Toolbox.

The ***PCOCameraAdaptor_<version>.dll/so*** represents the actual adaptor which is used, where ***<version>*** represents the Matlab version, e.g ***R2022b***.
The section 2.1 (adapter setup) shows how to set up and register the PCO adapter. Additionally a readme.txt is provided giving the same installation instructions. Several example script files show how to set up the camera and acquire images for different use cases, using the adaptor functions. The ***pco_imaqregister.m*** function provides the registration of the adaptor.

After having set up and registered the adaptor it is possible to acquire images from pco cameras in two ways:

- You can completely work with commands either in the command window or by creating m-files for the image acquisition toolbox.
- The second possibility is to use the image acquisition GUI provided by MATLAB with graphical elements for properties and acquisition commands.

The adaptor supports all camera types.
Due to the toolbox structure only streaming is supported, it isn't possible to read out camera RAM.
If reading camera RAM is required, script files for MATLAB have to be used (see chapter 4).

This adaptor is supported for ***MATLAB R2016a*** and later versions.

2.1 Adapter Setup

To use the adaptor for the image acquisition toolbox in matlab, follow these steps:

1. Open Matlab and make sure you have the image acquisition toolbox installed (by typing `ver` in the Matlab command window)
2. Choose ***<installation folder>/adaptor*** as current folder in matlab
3. Register the adaptor in the toolbox by calling `pco_imaqregister` or `pco_imaqregister('register')` in the matlab command line.
   This will copy the suitable ***PCOCameraAdaptor_<version>.dll/so*** and necessary sdk binaries into the current folder and the adaptor will be registered from this location.
   (If an older version of the adaptor has already been registered before, it will automatically be unregistered before)

Note By calling `pco_imaqregister('unregister')` you can unregister the PCOCameraAdaptor (location doesn't matter)

To check if the register was successful, type `imaqhwinfo` in the command window.
Here `pcocameraadaptor_<version>` has to appear in the line of `InstalledAdaptors`

After completing these steps you will be able to access the pco cameras from the image acquisition toolbox.

2.2 Matlab Image Acquisition GUI

The Image Acquisition GUI provided by MATLAB is an easy way to acquire images. The available cameras and their attributes are clearly arranged. The previewing and recording can be controlled by several buttons.

Beginning with R2022a, MATLAB changed the style of the GUI completely and they call it **Image Acquisition Explorer**. The descriptions in the following chapters are based on this new GUI layout. If you need the description for the old Image Acquisition GUI, please contact us at pco@excelitas.com.

2.3 Related manuals

To get more information on working with the image acquisition toolbox, have a look at the MATLAB Image Acquisition Toolbox Documentation. For further information on the camera properties, read the latest pco.sdk manual.

2.4 Explorer Toolbar

On the top of the MATLAB Image Acquisition Explorer GUI there is a **EXPLORER** toolbar which contains the most important settings, grouped by functionality.

2.4.1 CONFIGURE FORMAT

The image/video configuration is done in the **CONFIGURE FORMAT** section of the toolbox

Video Format This property defines the interface(video) format of the camera. Since PCO cameras always have a fixed interface this value cannot be changed

Color Space This property sets the color of the image. The values that can be set here depend on the image type delivered by the camera. For monochrome cameras only *grayscale* is useful, since it is not possible to get color information out of a grayscale image. For color cameras valid values are *grayscale* and *rgb* representation.

Sensor Alignment This property would define the alignment of the color channels for correct bayer pattern demosaicing. Since this is done automatically inside the adaptor, there is nothing to select here.

2.4.2 LOGGING

The settings for saving acquired images can be selected in the **LOGGING** section of the explorer toolbox

- ***File:*** Log the data directly to disk
- ***Workspace Variable:*** Store the images in a workspace variable which can then be used e.g. for further processing

For both modes there are textboxes to specify the name for the image and the video.

2.4.3 SNAPSHOT

The **SNAPSHOT** section of the explorer toolbox has a *Capture* button to snap one image into a workspace variable.

2.4.4 RECORD

The recording settings are done in the **RECORD** section of the explorer toolbox.

Here you can switch between three options:

- ***Finite:*** Record the defined number of frames / Record for the defined number of seconds
- ***Continuous:*** Continuously record images until stop is pressed
- ***Hardware Trigger:*** Use hardware trigger to control the image acquisition (see section 2.6)

Pressing the **Record** button will start the recording.

2.4.5 Additional Sections

Additionally, the Explorer Toolbox has three more sections which provide additional functionality.

- ***VISUALIZE AND ANALYZE:*** This section lists tools which can be used to visualize and analyze the recorded images.
- ***EXPORT:*** Here you can open script editors to create scripts for operating the imaq toolbox.
- ***CLOSE:*** The **Close Session** button is a convenient way to cleanup everything and close the camera and the toolbox.

2.5 Device Properties

Most of the camera settings available in pco.camware are also available in the adaptor. They can be found in the ***Device Properties*** section of the GUI.

Due to the structure of adaptor properties they are arranged in a different way. The device properties don't belong to a video input object but to a video source object. Besides delay time, exposure time and frame rate, acquisition or preview is automatically stopped when changing a property, afterwards it will be restarted.

2.5.1 Overview

The following table shows an overview of all described device properties in alphabetical order and indicates if they are read-only and available for all cameras. Properties where the availability is specific are only visible if the camera supports them.

Property	Description	Read-only	Availability
AMAcquireMode	Acquire mode of the camera auto, extern or sequence triggered.	No	Specific

Continued on next page

Continued from previous page

Property	Description	Read-only	Availability
AMImageNumber	Images to acquire for one acquisition pulse. Will be ignored if acquire mode is auto or extern.	No	Specific
B1BinningHorizontal	Horizontal binning of the camera: The binning will reduce the resolution of the camera by the binning factor. This also effects the hardware ROI. Binning may change the pattern of color cameras.	No	Always
B2BinningVertical	Vertical binning of the camera: The binning will reduce the resolution of the camera by the binning factor. This also effects the hardware ROI. Binning may change the pattern of color cameras.	No	Always
CFConerversionFactor _e_count	Conversion factor of the camera.	No	Always
DelayTime	Delay time of the camera [s]]: Can only be set for some cameras. Will be ignored if FMFpsBased is set to on.	No, if selectable	Always
DPCorePreparation	Switches DPCore preparation on/off. Will appear if Jetraw is installed or if the DPCore binaries reside in the adaptor folder. The property is active if the camera is deposited in the DPCore, otherwise grayed out.	No	Specific
ExposureTime	Exposure time of the camera [s]	No	Always
FlimIPAsymCorrection	Switch asymmetry correction on/off.	No	Specific
FlimMPMasterFrequency_Hz	Master modulation frequency in Hz.	No	Specific
FlimMPModulationSource	Select the modulation source intern/extern.	No	Specific
FlimMPOutputWaveform	Select the output waveform: none/sine/square.	No	Specific
FlimMPRelativePhase_mdeg	Relative phase in millidegree.	No	Specific
FlimPSAddPhaseSampling	Switch additional phase sampling (phase symmetry) yes/no.	No	Specific
FlimPSPhaseNumber	Phase number.	No	Specific
FlimPSPhaseOrder	Phase order (ascending/opposite).	No	Specific
FlimPSTapSelection	Selected tap(s): Tap A/Tap B/ Tap A+B.	No	Specific

Continued on next page

Continued from previous page

Property	Description	Read-only	Availability
FMFpsBased	Select if exposure time is set together with delay time (off) or frame rate (on). Can only be set to -on- if trigger mode is not hardware triggered.	No	Specific
FRFrameRate	Frame rate of the camera in mHZ. Will be ignored if FMFpsBased is off.	No	Specific
H1HardwareROI_X_Offset	Horizontal offset for HW ROI. Can only be set for some cameras. Changing hardware ROI may change the pattern of color cameras.	No, if selectable	Always
H2HardwareROI_Width	Width for HW ROI. Can only be set for some cameras. Changing Hardware ROI may change the pattern of color cameras. If horizontal ROI has to be symmetric, changes in width will be reset and horizontal ROI can only be changed by the x offset.	No, if selectable	Always
H3HardwareROIHor_Sym	Indicates if horizontal hardware ROI has to be symmetric.	Yes	Always
H4HardwareROI_Y_Offset	Vertical offset for HW ROI. Can only be set for some cameras. Changing Hardware ROI may change the pattern of color cameras.	No, if selectable	Always
H5HardwareROI_Height	Height for HW ROI. Can only be set for some cameras. Changing Hardware ROI may change the pattern of color cameras. If vertical ROI has to be symmetric, changes in height will be reset and vertical ROI can only be changed by the y offset.	No, if selectable	Always
H6HardwareROI_Vert_Sym	Indicates if vertical hardware ROI has to be symmetric.	Yes	Always
IO_x_SignalEnableDisable	Enable/disable IO signal at port x.	No	Specific
IO_x_SignalName	Select name of signal that should be connected with IO port x.	No	Specific
IO_x_SignalPolarity	Polarity of IO signal at port x.	No	Specific
IO_x_SignalType	Type of IO Signal at port x.	No	Specific
IRMode	Switch IR sensitivity on/off.	No	Specific
NFNoiseFilter	Switch noise filter on/off.	No	Specific

Continued on next page

Continued from previous page

Property	Description	Read-only	Availability
PCPixelclock_Hz	Pixel rate. Will set the FMFpsBase to off if available.	No	Always
RDIDoubleImageMode	Switch double image mode on/off. See Annotations for property description.	No	Specific
SCMOSReadoutMode	Readout mode setting for pco.edge cameras. This parameter is only available for pco.edge cameras in rolling shutter.	No	Specific
SFSensorFormat	Sensor format of the camera. In the format (standard), only affective pixels are read out from the sensor. The readout in the format (extended) is camera dependent. This also affects the hardware ROI. The sensor format may change the pattern of color cameras.	No	Always
SMShutterMode	Shutter mode. **If shutter mode is changed, the adaptor has to be refreshed/reset.**	No	Specific
TMTimestampMode	Timestamp mode. No stamp, binary, binary-and-ASCII, ASCII.	No	Specific

2.5.2 Annotations

AMAcquireMode If acquire mode is set to extern or sequence triggered, the acquisition is controlled by an external source. If high or low state is effective depends on the settings of the IO signal properties for the acquire enable port. AMImageNumber property will only be created, if sequence trigger mode is available.

FMFpsBased If frame rate and exposure time are set, it can occur that the values are trimmed by the camera. If the selected exposure time is too high for the current frame rate, the frame rate will be trimmed and updated. If the selected frame rate is too high, the exposure time will be trimmed and updated.

IRMODE Since there are different minimal and maximal exposure times for IR sensitivity on and off, changing IR mode will change min and max exposure time and perhaps (if current exposure time exceeds the new range) also the current exposure time.

RDIDouble ImageMode If double image mode is set to ”on”, the camera will record two images directly after each other instead of just one image. In the adaptor the two images are treated as one with doubled height (first image on top, second on bottom). Setting some hardware ROI (if possible for the specific camera) will affect each of the two images individually (as expected). Setting the soft ROI of the pco.matlab adaptor will affect the two images as one, i.e. setting a y-offset of 10 means cutting off the first 10 lines of the upper image, while the second image will not be affected. If the images should be saved in separated files, the separation has to be done programmatically.

2.6 Triggering

For triggering you have to select ***Hardware Trigger*** in the **RECORD** section on top of the GUI. Then the **Hardware Trigger** section appears where you can configure your trigger parameters

Number of Triggers This parameter sets the number of triggers you want to acquire. Using as a command you have to set the **TriggerRepeat** property, which is always one less than the number of triggers.

Frames per Trigger This sets the frames that should be acquired with one trigger signal.

Since PCO cameras always record one image per trigger, setting *Frames per Trigger* > 1 results in the same behavior as adding this value to *Number of Triggers*. [1]

Trigger Source Depending on the camera type one (or both) of the following trigger sources are available

- ***ExternExposureStart:*** An image is taken when an external signal rises or falls (depends on the IO settings)
- ***ExternExposureCtrl:*** An image is taken when an external signal rises or falls (depends on the IO settings). The exposure time is controlled by the length of the external pulse.

[1] This means that the MATLAB expects *Frames per Trigger* * *Number of Triggers* images and the camera will always record 1 image per trigger pulse

2.7 Region of interest

In addition to the hardware ROI property provided by the camera, MATLAB is also able to perform ***software ROI***.

This can be done in the **ROI Position** section of the GUI. You can either select the roi by adjusting the rectangular frame that appears when pressing ***Select ROI*** or by manually adjusting ***X-Offset***,***Y-Offset***, ***Width*** and ***Height***

The toolbox uses these settings to crop the image to its final size. The range of the four values is limited by the camera resolution, the selected hardware ROI and binning.

2.8 Troubleshooting

The camera adaptor also supports a troubleshooting. If there are problems, you can force the adaptor to write the workflow into a log file by creating a file called ***sc2_imaq_adaptor.log*** in the following directory:

Windows	<systemdisc>:\ProgramData\pco\
Linux	/.pco/pco_logging

3 Flim package

The pco.matlab flim package contains a complete set of MATLAB functions to cover a pco.flim workflow.

3.1 Flim introduction

A function to rearrange the raw data recorded with the pco.flim in a general linear phase sequence is also provided. Please refer to the pco.flim white paper for detailed explanations and mathematical descriptions.

The functions in section **3.2.1** were designed to be independent from the used MATLAB version. Nevertheless, the MATLAB version 2017a or higher in conjunction with the **Image Processing Toolbox** and the **Image Acquisition Toolbox** is recommended to benefit from the correct display of the color bars in the lifetime figures when using the example functions in sections **3.2.2** and **3.2.3**.

3.2 File organization

Each MATLAB function is encapsulated into one file whose filename equals the function name. The directory ***flim_functions*** provides general FLIM functions. The directories ***flim_example_file*** and ***flim_example_memory*** provide examples incorporating these general functions.

3.2.1 Flim functions

The following functions were designed to be used as independent modules which represent the separate steps in the computation of FLIM images. Such steps are the rearrangement of phase images recorded with the pco.flim in a linear phase image stack, the computation of modulation depth and phase images, the referencing procedure to cancel influences induced by the measurement setup, and the computation of lifetime images.

If applicable, all functions offer either distinct modulation depth and phase images or their combination in the form of complex phasors as intermediate or final results.

3.2.1.1 flim_rearrange_stack

Description Takes a stack of raw images acquired by the pco.flim and generates a sequence of linearly ascending phase images. The output stack of linear ascending phase images can then be processed by flim_-numeric_harmonic_analysis.m to compute specified harmonic for each pixel.

Supported camera type(s) pco.flim

Prototype

```
function imgStack = flim_rearrange_stack (
    imgStackRaw,                              //in
    phaseNumber,                              //in
    addPhaseSampling,                         //in
    phaseOrder,                               //in
    tapSelection,                             //in
    asymCorrection                            //in
);
```

Parameter

Name	Description
`imgStackRaw`	3D image stack recorded with the pco.flim camera.
`phaseNumber`	’shiftablePair’, ’02’, ’04’, ’08’, ’16’.
`addPhaseSampling`	’no’, ’yes’.
`phaseOrder`	’ascending’, ’opposite’.
`tapSelection`	’Tap A’, ’Tap B’, ’Tap A+B’.
`asymCorrection`	’on’, ’off’.

Return value

Name	Description
`imgStack`	3D image stack of linearly ascending phases.

3.2.1.2 flim_numeric_harmonic_analysis

Description With a given sequence of linearly ascending phase images, the following function can be used to compute the parameters of a specified harmonic for each pixel, such as the modulation index (i.e. modulation depth), the phase and the normalized intensity (i.e. constant value). In most cases the fundamental, i.e. the first harmonic, is computed.

A more detailed description of these parameters can be found in the pco.flim manual (subsection FLIM setup in chapter pco.camware 4 software).

Supported camera type(s) pco.flim

Prototype

```
function [modIndex phase normInt] = flim_numeric_harmonic_analysis (
    imgStack,                                   //in
    nBits,                                      //in
    nPhases,                                    //in
    harmonic                                    //in
);
```

Parameter

Name	Description
`imgStack`	3D image stack containing phase information along 3rd dimension.
`nBits`	Bit resolution of imgStack raw intensity values.
`nPhases`	Number of phases.
`harmonic`	Harmonic component to be computed (1 for fundamental).

Return value

Name	Description
`modIndex`	2D modulation index image.
`phase`	2D phase image (in radians).
`normInt`	2D normalized intensity image.

3.2.1.3 flim_numeric_harmonic_analysis_phasor

Description This function equals the function in section **3.2.1.2** except that the computed modulation index image and the phase image are combined into an image of complex values called ’phasors’. The mathematical relationship is given by

$$p_{xy} = m_{xy} e^{i \o xy} \tag{3.1}$$

with the phasor p_{xy}, the modulation index m_{xy} and the phase $\o_{xy}$ for each pixel with the coordinates (x, y).

The advantage of complex notation is the simplification of the algebra needed for the description of transfer functions, since MATLAB is capable of handling complex numbers.

Supported camera type(s) pco.flim

Prototype

```
function [phasor normInt] = flim_numeric_harmonic_analysis_phasor (
    imgStack,                                 //in
    nBits,                                    //in
    nPhases,                                  //in
    harmonic                                  //in
);
```

Parameter

Name	Description
`imgStack`	3D image stack containing phase information along 3rd dimension.
`nBits`	Bit resolution of imgStack raw intensity values.
`nPhases`	Number of phases.
`harmonic`	Harmonic component to be computed (1 for fundamental).

Return value

Name	Description
`phasor`	2D complex phasor image.
`normInt`	2D normalized intensity image.

3.2.1.4 flim_referencing

Description Since each single FLIM measurement is influenced by the response of the overall setup (pco.flim camera, light source, optical pathways, cables) a reference measurement is needed to correct for this. In most cases a photoluminescent sample with a known luminescence lifetime is used as a reference.

The following function computes referenced modulation index and phase images using the single results of the reference and sample measurements.

Supported camera type(s) pco.flim

Prototype

```
function [modIndex phase] = flim_referencing (
    modIndexSample,                              //in
    phaseSample,                                 //in
    modIndexRef,                                 //in
    phaseRef,                                    //in
    tauRef,                                      //in
    f                                            //in
);
```

Parameter

Name	Description
`modIndexSample`	2D modulation index image of sample measurement.
`phaseSample`	2D phase image of sample measurement.
`modIndexRef`	2D modulation index image of reference measurement.
`phaseRef`	2D phase image of reference measurement.
`tauRef`	Time constant of reference system (in seconds).
`f`	Modulation frequency (in Hertz).

Return value

Name	Description
`modIndex`	Referenced 2D modulation index in image.
`phase`	Referenced 2D phase image.

3.2.1.5 flim_referencing_phasor

Description This function equals the function in section **3.2.1.4** except that phasor images are used as input parameters and return value.

Supported camera type(s) pco.flim

Prototype

```
function phasor = flim_referencing_phasor (
    phasorSample,                                  //in
    phasorRef,                                     //in
    tauRef,                                        //in
    f                                              //in
);
```

Parameter

Name	Description
`phasorSample`	2D complex phasor image of sample measurement.
`phasorRef`	2D complex phasor image of reference measurement.
`tauRef`	Time constant of reference system (in seconds).
`f`	Modulation frequency (in Hertz).

Return value

Name	Description
`phasor`	Referenced 2D complex phasor image.

3.2.1.6 flim_get_lifetimes

Description

Assuming that the measured samples can be characterized by first-order low-pass systems (monoexponential behavior) their single time constants (lifetimes) can be computed by means of the (referenced) modulation indices and phases using the following function.

Supported camera type(s)

pco.flim

Prototype

```
function [mLifetime pLifetime] = flim_get_lifetimes (
    modIndex,                                   //in
    phase,                                      //in
    f                                           //in
);
```

Parameter

Name	Description
`modIndex`	2D modulation index image.
`phase`	2D phase image (in radians).
`f`	Modulation frequency (in Hertz).

Return value

Name	Description
`mLifetime`	2D lifetime image based on the modulation index (in seconds).
`pLifetime`	2D lifetime image bsaed on the phase (in seconds).

3.2.1.7 flim_get_lifetimes_phasor

Description This function equals the function in section **3.2.1.6** except that a phasor image is used as input parameter.

Supported camera type(s) pco.flim

Prototype

```
function [mLifetime pLifetime] = flim_get_lifetimes_phasor (
    phasor,                                     //in
    f                                           //in
);
```

Parameter

Name	Description
phasor	2D complex phasor image.
f	Modulation frequency (in Hertz).

Return value

Name	Description
mLifetime	2D lifetime image based on the modulation index (in seconds).
pLifetime	2D lifetime image bsaed on the phase (in seconds).

3.2.2 Example using multi-tiff files

Description The following function incorporates the functions mentioned in section **3.2.1** to demonstrate a complete FLIM workflow using multi-TIFF files as raw input. In addition to the input parameters described in that section there are parameters to specify the reference and sample files containing the raw data as well as displaying options for the results. After the completion of the FLIM computation a normalized intensity image and two lifetime images based on the modulation index and phase are displayed.

Supported camera type(s) pco.flim

Prototype

```
function flim_example_file (
    refFileName,                            //in
    sampleFileName,                         //in
    f,                                      //in
    tauRef,                                 //in
    nPhases,                                //in
    nBits,                                  //in
    bitAlignDiv,                            //in
    phaseNumber,                            //in
    addPhaseSampling,                       //in
    phaseOrder,                             //in
    tapSelection,                           //in
    asymCorrection,                         //in
    dispMinLifetime,                        //in
    dispMaxLifetime                         //in
);
```

Parameter (excerpt)

Name	Description
`refFileName`	Filename of reference image stack.
`sampleFileName`	Filename of sample image stack.
`dispMinLifetime`	Minimum lifetime for display (in seconds).
`dispMaxLifetime`	Maximum lifetime for display (in seconds).

A script which calls the above function with predefined parameters and input filenames is provided by the following file:
`flim_example_file_script.m`

3.2.3 Example using matlab memory data

Description

In case the raw image stacks containing the phase images are already available in the MATLAB memory, e.g. recorded by means of the ***Image Acquisition Toolbox*** in conjunction with the ***pco.matlab adaptor***, the following function can be used to compute and display the FLIM results in a simple way. The computation workflow is the same as in section **3.2.2**.

Supported camera type(s)

pco.flim

Prototype

```
function flim_example_memory (
    refRawStack,                                   //in
    sampleRawStack,                                //in
    f,                                             //in
    tauRef,                                        //in
    nPhases,                                       //in
    nBits,                                         //in
    phaseNumber,                                   //in
    addPhaseSampling,                              //in
    phaseOrder,                                    //in
    tapSelection,                                  //in
    asymCorrection,                                //in
    dispMinLifetime,                               //in
    dispMaxLifetime                                //in
);
```

Parameter (excerpt)

Name	Description
`refRawStack`	Reference raw image stack.
`sampleRawStack`	Sample raw image stack.

A typical call in the MATLAB console using the data recorded with the ***Image Acquisition Toolbox*** could look like this:

```
flim_example_memory(permute(refStack(:, :, 1, :), [1 2 4 3]), permute↩
    (...
sampleStack(:, :, 1, :), [1 2 4 3]), 1E6, 4E-9, 8, 14, '08', 'yes',
...'ascending', 'Tap A+B', 'off', 0, 10E-9)
```

As all input stacks must be three-dimensional the four-dimensional access is based on how the toolbox stores gray value images in a sequence.

4 SDK example scripts

The pco.matlab scripts provide a collection of example m-files.
The necessary binary-files and additional header files are included.

The script files contain examples for camera setup, grabbing images to a MATLAB image stack and displaying them in a figure window. There are also examples, which show how to work with the pco.recorder from script file.

Only a subset of all possible camera settings is covered by the examples. For further setup see the pco.sdk description.

When working on a 64 bit Windows system a C-compiler must be installed to enable MATLAB to use external libraries. See also http://de.mathworks.com/support/compilers/current_release/.

To run the example code open MATLAB and select the install directory.

Call the script ***setup_files*** to copy the following files (could also be done manually):

- Copy the binary files from the runtime\bin64 (64 bit) directory to the install directory.
- Copy the header files from the runtime\include directory to the install directory.
- Depending on the installed MATLAB version copy the pco_uint.m and pco_uinterr.m from the ver_7 or ver_8 directory.
- Call pco_camera_create_deffile.m, which will create a pco_camera_def.txt file with header definitions.

All m-files whose names start with pco_camera have subfunctions included, which can be used in other files or also standalone.

All m-files whose names start with pco_sdk_example are examples with different functionality.
All files include helptext which can be shown with the MATLAB command help.
i.e. `help pco_sdk_example_stack`

All scripts output some text to the command window using `MATLAB disp()` function to show processing steps and give some information. This helps to evaluate the correct behavior of camera and script code.

A short description of the example files follows.

4.1 General

All m-files use a common structure glvar.
By setting the variables of this structure different behaviour of loading/unloading SDK library and open/close of the camera can be accomplished.

All example code was tested with different MATLAB versions in 64 bit.

When writing your own m-files it might happen that MATLAB stops due to syntax or other error. Therefore all examples are built with MATLAB exception handling, which on error does close the camera, unloads the sc2_cam library and the pco_recorder library, and shows the error source.

Use pco_reset_camlib and/or pco_reset_recorder to initialize the libraries again.

4.2 SDK examples

The pco_sdk examples show how to setup and use the pco.sdk API within MATLAB.

For a simple test call pco_camera_info.m m-file:

```
pco_camera_info();
```

This should output some messages about camera type and camera revisions.

4.2.1 PCO_sdk_example_single

Description Grab and display a single image with selectable trigger mode and exposure time. `pco_sdk_example_single()` uses subfunctions from pco_camera_() and draw_image().

Supported camera type(s) All cameras

Prototype

```
function [errorCode, ima] = PCO_sdk_example_single (
    exposure_time,                                //in
    triggermode                                   //in
);
```

Parameter

Name	Description
`exposure_time`	Camera exposure time in ms (default=10 ms).
`triggermode`	Camera trigger mode (default=AUTO).

Return value

Name	Description
`errorCode`	0 in case of success, error code otherwise.
`ima`	grabbed image.

4.2.2 PCO_sdk_example_swtrig

Description Grab and display a single software triggered image with selectable exposure time. `pco_sdk_example_swtrig()` uses direct SDK calls and imshow().

Supported camera type(s) All cameras

Prototype

```
function [errorCode] = PCO_sdk_example_swtrig (
    exposure_time                                 //in
);
```

Parameter

Name	Description
`exposure_time`	Camera exposure time in ms (default=10 ms).

Return value

Name	Description
`errorCode`	0 in case of success, error code otherwise.

4.2.3 PCO_sdk_example_live_getima

Description Start camera, grab and display images in a loop. `pco_sdk_example_live_getima` does use subfunctions from `pco_camera_()` and `draw_image()` and grab with SDK-function **PCO_GetImageEx** with a single buffer.

Supported camera type(s) All cameras

Prototype

```
function [errorCode] = PCO_sdk_example_live_getima (
    looptime,                                  //in
    exposure_time,                             //in
    triggermode                                //in
);
```

Parameter

Name	Description
`looptime`	Time the loop is running in seconds (default=10 seconds).
`exposure_time`	Camera exposure time in ms (default=10 ms).
`triggermode`	Camera trigger mode (default=AUTO).

Return value

Name	Description
`errorCode`	0 in case of success, error code otherwise.

4.2.4 PCO_sdk_example_live_add

Description

Start camera, grab and display images in a loop.
`pco_sdk_example_live_add` uses subfunctions from `pco_camera_()` and `draw_image ()` and grab images with SDK-function **PCO_AddBuffer** and **PCO_WaitforBuffer** with four buffers.

Supported camera type(s)

All cameras

Prototype

```
function [errorCode] = PCO_sdk_example_live_add (
    looptime,                                    //in
    exposure_time,                               //in
    triggermode                                  //in
);
```

Parameter

Name	Description
`looptime`	Time the loop is running in seconds (default=10 seconds).
`exposure_time`	Camera exposure time in ms (default=10 ms).
`triggermode`	Camera trigger mode (default=AUTO).

Return value

Name	Description
`errorCode`	0 in case of success, error code otherwise.

4.2.5 PCO_sdk_example_stack

Description Grab images from a streaming camera directly to an image stack.
The `image_stack` is transposed before returned to workspace.
The returned `image_stack` can be displayed with one of the `draw_images` functions.

Supported camera type(s) All cameras

Prototype

```
function [errorCode, ima_stack, metastructs] = PCO_sdk_example_stack ↩
    (
    imacount,                                         //in
    exposure_time,                                    //in
    triggermode                                       //in
);
```

Parameter

Name	Description
`imacount`	Number of images to grab.
`exposure_time`	Camera exposure time in ms (default=10 ms).
`triggermode`	Camera trigger mode (default=AUTO).

Return value

Name	Description
`errorCode`	0 in case of success, error code otherwise.
`ima_stack`	Stack with ’imacount’ images.
`metastructs`	Stack with ’imacount’ structures of type PCO_-METADATA_STRUCT, if metadata are available and enabled.

4.2.6 PCO_sdk_example_read

Description Grab images into camera internal memory and readout afterwards. The `pco_sdk_example_read ()` uses subfunctions from `pco_camera_()`. Read is done with SDK-functions **PCO_AddBuffer** and **PCO_WaitforBuffer** with four buffers.

Supported camera type(s) All cameras with internal memory

Prototype

```
function [errorCode, ima_stack, metastructs] = PCO_sdk_example_read (
    imacount,                                        //in
    segment,                                         //in
    exposure_time,                                   //in
    triggermode                                      //in
);
```

Parameter

Name	Description
`imacount`	Number of images to grab.
`segment`	Segment to use for grab and readout (default=1).
`exposure_time`	Camera exposure time in ms (default=10 ms).
`triggermode`	Camera trigger mode (default=AUTO).

Return value

Name	Description
`errorCode`	0 in case of success, error code otherwise.
`ima_stack`	Stack with ’imacount’ images.
`metastructs`	Stack with ’imacount’ structures of type PCO_-METADATA_STRUCT, if metadata are available and enabled.

4.3 Recorder examples

The pco.recorder is built on top of the SDK and forms an API with reduced amount of functions to simplify acquiring and retrieving images compared to the raw pco_sdk functions. The pco_sdk_-recorder examples show how to setup and use the pco.recorder API within MATLAB.

4.3.1 PCO_sdk_example_recorder

Description Set variables and grab images with the pco.recorder from a single pco.camera. The camera is opened within pco.sdk before the recorder functions are called. When recording has been done, imagedata is copied from recorder memory to a MATLAB array.

Supported camera type(s) All cameras

Prototype

```
function [errorCode, ima_stack] = PCO_sdk_example_recorder (
    imacount,                                        //in
    exposure_time,                                   //in
    triggermode                                      //in
);
```

Parameter

Name	Description
imacount	Number of images to grab.
exposure_time	Camera exposure time in ms (default=10 ms).
triggermode	Camera trigger mode (default=AUTO).

Return value

Name	Description
errorCode	0 in case of success, error code otherwise.
ima_stack	Stack with ’imacount’ images.

5 About Excelitas PCO

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

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

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

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

pco.

An Excelitas Technologies Brand

An Excelitas Technologies Brand

postal address:	Excelitas PCO GmbH Donaupark 11 93309 Kelheim, Germany
telephone:	+49 (0) 9441 2005 0
e-mail:	pco@excelitas.com
web:	www.excelitas.com/pco