technical info

# pco.edge

# CameraLink Interface Description


pco.
imaging

This document describes the **CameraLink Interface** of the pco.edge scientific CMOS (sCMOS) camera.

For SDK implementation please refer to the PCO **Software Development Kit** for pco.cameras (MA_DCSDKWINE_xxx.pdf).

For a detailed description on the camera handling and operating please refer to the **user's manual** of the pco.edge camera.

This document replaces:

- pco.edge camera control commands
- pco.edge camera link packing modes

Target Audience: This camera is designed for use by technicians, engineers and scientists.

**In case of any questions or comments, please contact us at PCO.**



| | |
|---|---|
| telephone | +49 (0) 9441 2005 50 |
| fax | +49 (0) 9441 2005 20 |
| email | info@pco.de |
| postal address | PCOAG<br>Donaupark11<br>93309 Kelheim, Germany |


The cover photo shows an exemplary PCO camera system.
The lens is sold separately.



Updated March 2011 © PCO AG

# pco.edge CameraLink Interface

## Index

# 1 Introduction

This document describes the CameraLink interface of the pco.edge. The CameraLink specification includes higher-bandwidth configurations that provide additional video data paths over a second connector/cable. The "Medium" configuration doubles the video bandwidth, adding additional 24 bits of data and the same 4 framing/enable signals as present in the "Base" configuration. The "Full" configuration adds another 16 bits to the data path, resulting in a 64 bit wide video path.

The “10-tap” or also called “Basler Configuration” extends the width of the "Full" configuration by reassigning some of the redundant framing/enable signals to produce a data path width of 80 bits. The standard configuration of the pco.edge uses these 10 taps per clock cycle in order to transfer the data to the PC. Each CameraLink port (A and B) must be connected to the grabber ports (A and B). In all available modes the CameraLink frequency is set to 85 MHz.

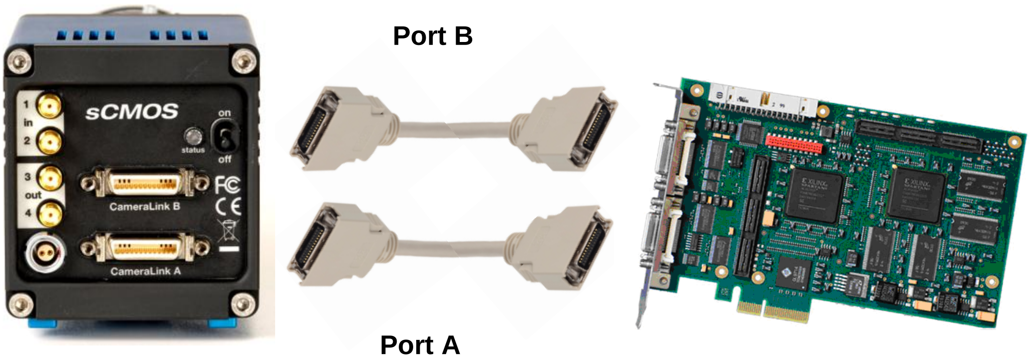



Figure 1.1: camera to grabber connection

# 2 Camera Data Processing

The pco.edge does not contain internal memory for storing images. All incoming data from the top and the bottom chip halves are sent alternatingly line by line through the CameraLink interface to the PC. The camera itself is controlled by low level commands, which are transported over the implemented UART channel of CameraLink port A. All settings are temporarilly stored in the camera and will be lost after a reboot or the next power cycle.

Figure 2.1 shows the internal data path of the pco.edge. The format of the output data is generated by consecutive modules. All settings are combined in the SDK command PCO_SetTransferParameter.

```
SC2_SDK_FUNC int WINAPI PCO_SetTransferParameter(HANDLE ph, void* buffer, int ilen)
```

The data rate of the image sensor must not be greater than the output data rate. In order to achieve this, a compression of the input data stream can be activated by sending the PCO_SetActiveLookupTable command through the UART communication port.

```
SC2_SDK_FUNC int WINAPI PCO_SetActiveLookupTable(HANDLE ph, WORD *wIdentifier, WORD *wParameter)
```

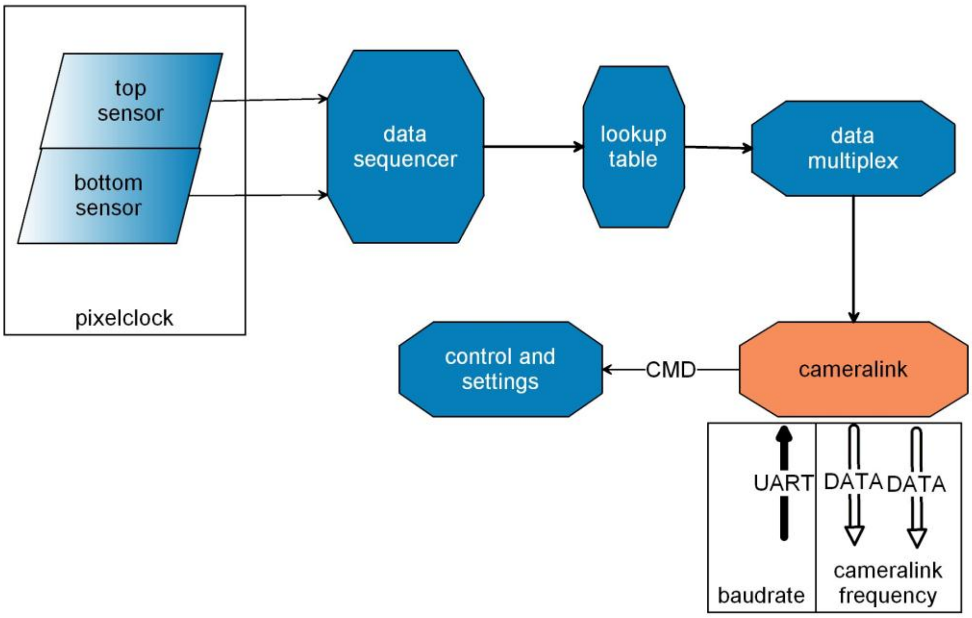



Figure 2.1: camera internal schematic

The camera has three clock domains, which the user can set through the pco.sdk. The communication speed is related to the baudrate of the UART channel. This baudrate can be set to the values 9600, 19200, 38400, 57600 and 115200. The default baudrate is 9600, but it is recommended to set it to 115200. The CameraLink frequency defines the clockrate of the cameralink interface. This frequency is at 85MHz by default and cannot be changed for the pco.edge. The user can select the pixelclock of the sensor between 286MHz and 95.3MHz. This affects the framerate and the produced data rate.

# 2.1 Readout Formats

There are five different readout formats implemented for rolling shutter. Each format can be set by the “Set Interface Output Format” command. The appropriate parameter for the destination interface is 0x0002 for SCMOS. The format parameters are defined below. For the best image quality and fastest frame rate Mode A is recommended.

```
SC2_SDK_FUNC int WINAPI PCO_SetInterfaceOutputFormat (HANDLE ph, WORD wDestInterface, WORD
wFormat, WORD wReserved1, WORD wReserved2)
```

Destination interface definition:

```
#define INTERFACE_CL_SCCMOS                                0x0002
```

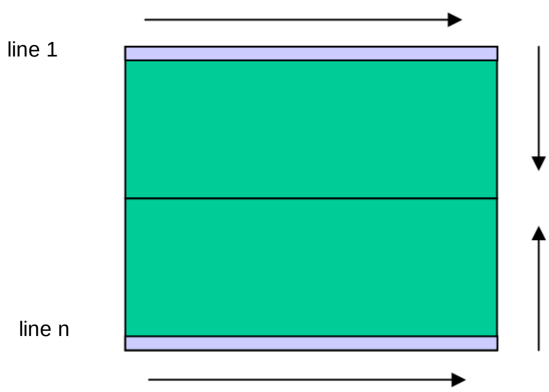
Readout Format definition:

```
#define SCCMOS_FORMAT_TOP_BOTTOM                           0x0000
//Mode E
#define SCCMOS_FORMAT_TOP_CENTER_BOTTOM_CENTER             0x0100
//Mode A
#define SCCMOS_FORMAT_CENTER_TOP_CENTER_BOTTOM             0x0200
//Mode B
#define SCCMOS_FORMAT_CENTER_TOP_BOTTOM_CENTER             0x0300
//Mode C
#define SCCMOS_FORMAT_TOP_CENTER_CENTER_BOTTOM             0x0400
//Mode D
```

Readout Format description:

**Mode A**

Readout Direction Top-Center/Bottom-Center

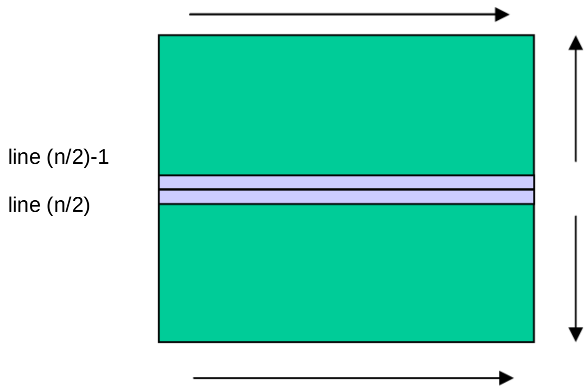



readout order:

| output line | image line |
|---|---|
| 1 | 1 |
| 2 | n |
| 3 | 2 |
| 4 | n-1 |
| 5 | 3 |
| 6 | n-2 |
| 7 | a.s.o |

**Mode B**

Readout Direction Center-Top/Center-Bottom

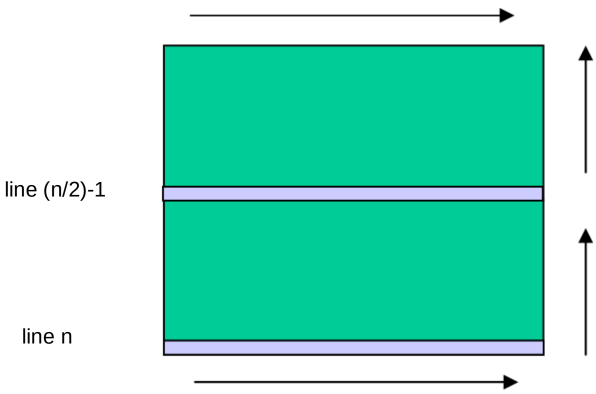



readout order:

| output line | image line |
|---|---|
| 1 | line (n/2) -1 |
| 2 | line (n/2) |
| 3 | line (n/2) - 2 |
| 4 | line (n/2) + 1 |
| 5 | line (n/2) - 3 |
| 6 | line (n/2) + 2 |
| 7 | a.s.o. |

**Mode C**

Readout Direction Center-Top/Bottom-Center

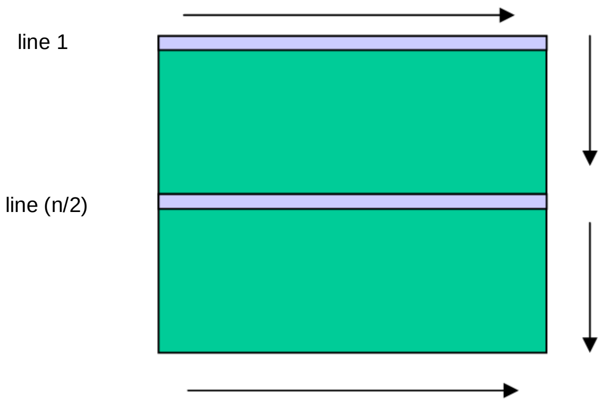



readout order:

| output line | image line |
|---|---|
| 1 | line (n/2) -1 |
| 2 | line n |
| 3 | line (n/2) - 2 |
| 4 | line n - 1 |
| 5 | line (n/2) - 3 |
| 6 | line n -2 |
| 7 | a.s.o. |

**Mode D**

Readout Direction Top-Center/Center-Bottom

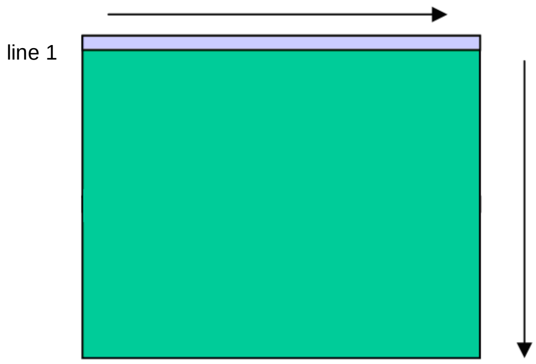



readout order:

| output line | image line |
|---|---|
| 1 | line 1 |
| 2 | line (n/2) |
| 3 | line 2 |
| 4 | line (n/2) + 1 |
| 5 | line 3 |
| 6 | line (n/2) + 2 |
| 7 | a.s.o. |

**Mode E**

Readout Direction Top- Bottom




readout order:

| output line | image line |
|---|---|
| 1 | line 1 |
| 2 | line 2 |
| 3 | line 3 |
| 4 | line 4 |
| 5 | line 5 |
| 6 | line 6 |
| 7 | a.s.o. |

Note: By using mode E the maximal frame rate is divided by 2.

## 2.2 Data Multiplex Overview

The camera works with a Big Endian pixel structure internally. When the single pixel value covers fewer bits as offered by the selected transfer mode, the most significant bits (MSB) are filled with zeros. In turn when the pixel value covers more bits as offered by the selected transfer mode, the MSB bits are cut off.

Examples:

a) Transfer Mode 16 Bit / Pixel Values 12 Bit

| | | |
|---|---|---|
| available bits: | 16 | |
| used bits: | 12 | |
| minimal value: | 0x0000 | (decimal: 0) |
| maximal value: | 0x0FFF | (decimal: 4095) |

b) Transfer Mode 12 Bit / Pixel Values 11 Bit

| | | |
|---|---|---|
| available bits: | 12 | |
| used bits: | 11 | |
| minimal value: | 0x000 | (decimal: 0) |
| maximal value: | 0x7FF | (decimal: 2047) |

c) Transfer Mode 8 Bit / Pixel Values 8 Bit

| | | |
|---|---|---|
| available bits: | 8 | |
| used bits: | 8 | |
| minimal value: | 0x00 | (decimal: 0) |
| maximal value: | 0xFF | (decimal: 255) |

d) Transfer Mode 8 Bit / Pixel Values 16 Bit

| | | |
|---|---|---|
| available bits: | 8 | |
| used bits: | 16 | |
| minimal value: | 0x00 | (decimal: 0) |
| maximal value: | 0xFFFF | (decimal: 65535) |
| maximal value: | 0xFF | (decimal: 255) |

## 2.3 CameraLink Output Tap Sorting

The Camera Link specification names each 8 bit port (also called tap) from A to J. Figure 2.2 shows the distribution of the 10 taps over the 80 bits. So the taps are counted from right to left.

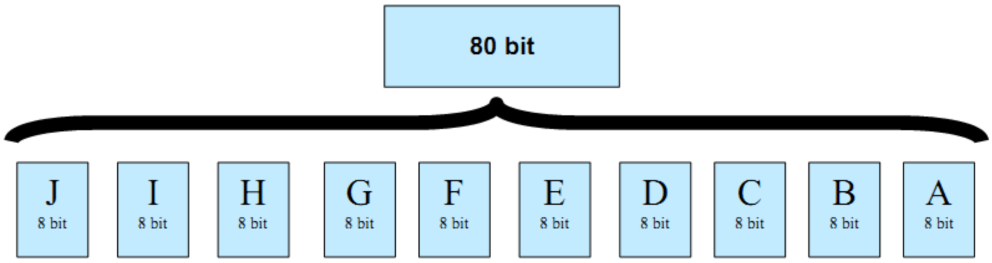



Figure 2.2: Distribution of the taps in 10-tap mode

The numbering of pixels in the camera always starts on the left side of the image to the right side. figure 2.3 shows the counting of the pixels in one line.

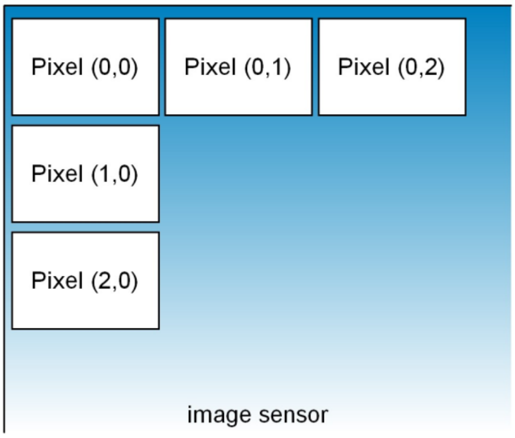



Figure 2.3: Counting direction of pixels on the image sensor

## 2.3.1 Packing of 16 bit

When using 16 bits per pixel, each pixel needs two taps for the transfer. In each clock cycle five pixels are buffered internally in one package. Counting from left to right the camera generates the data stream as shown in figure 2.4. After reordering of the pixels, they are sent from the camera to the grabber.

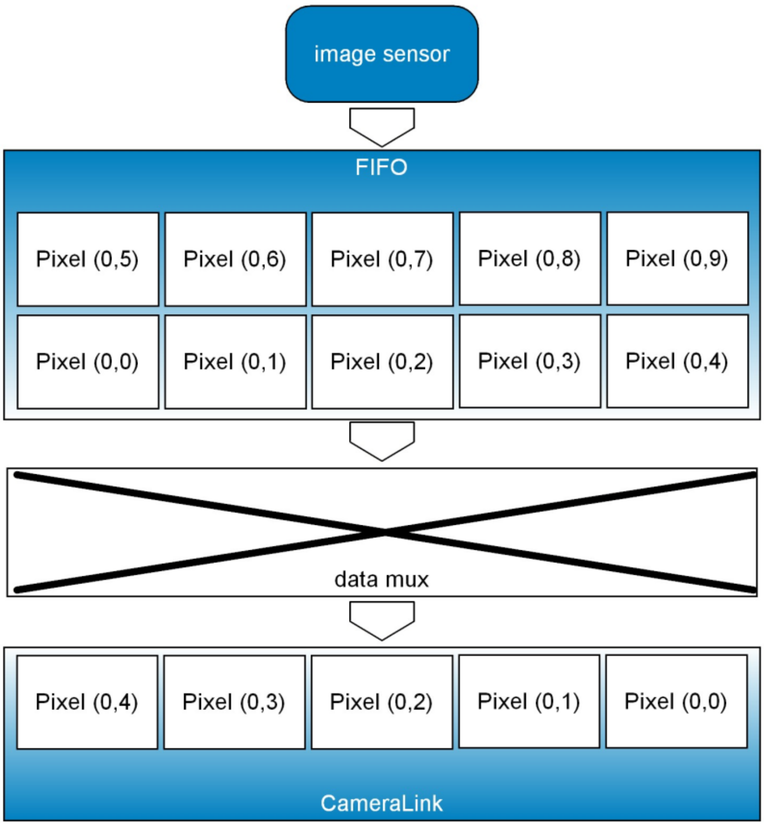



Figure 2.4: Internally generated data packages

Most grabber manufacturers (SiliconSoftware, Bitflow, etc.) interpret the pixels in the Camera Link data stream from right to left. For this reason the sequence of the five pixels in one 10-tap package is transferred in reverse order (figure 2.5). This is the most efficient way to transfer 16 bit pixels over Camera Link.

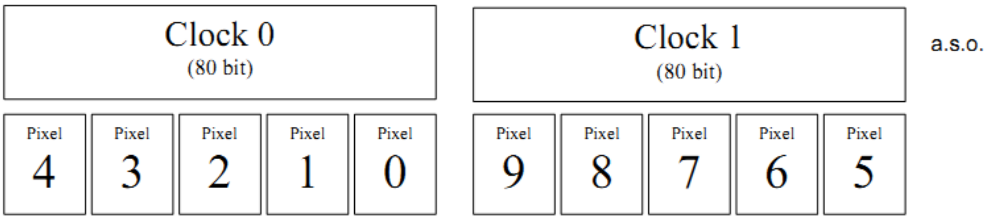



Figure 2.5: 80 bit packages sent over Camera Link

## 2.3.2 Packing of 12 bit

In case of transferring 12 bits per pixel, the camera needs three clock cycles to send 20 pixels to the grabber (figure 2.6). Each data package contains six full and one or two fractional pixels.

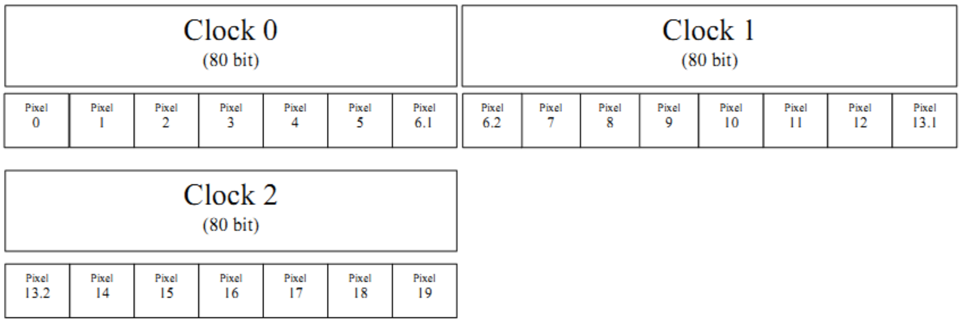



Figure 2.6: Internally generated packages to transfer 20 pixels with 12 bits

The generated data stream has to be sent by using the existing CameraLink structures in the camera. So the internally generated 80 bit data stream is matched into five 2 byte blocks (16 bit) and transferred in reverse order as described in figure 2.5.

## 2.3.3 Packing of 8 bit

When using 8 bits per pixel, each pixel needs one tap to be transferred. In this case each clock cycle contains 10 pixels. Counting internally from left to right the Camera Link specification is numbering the bits from the opposite side. So the bits are switched before transferred over Camera Link (figure 2.7).

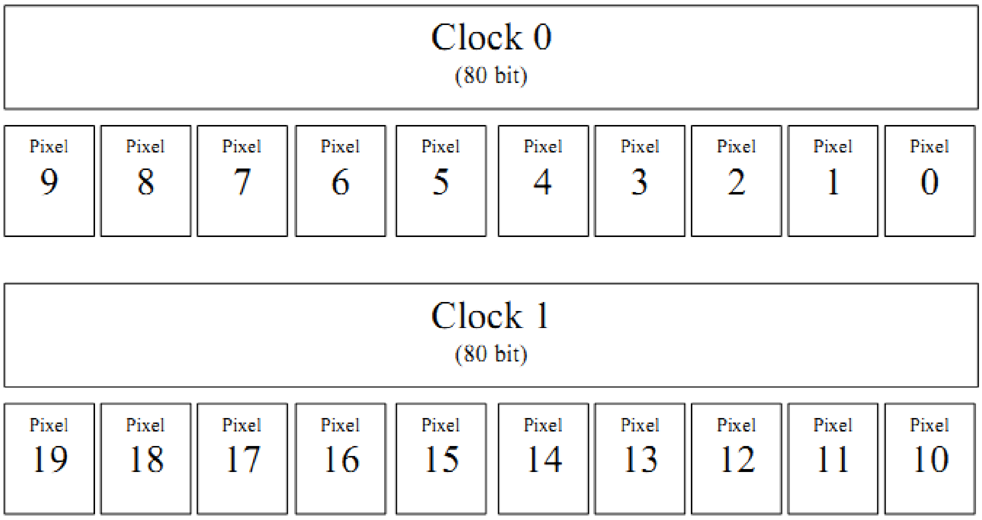



Figure 2.7: 80 bit packages sent over Camera Link

### 2.3.4 Tap Sorting Examples

Packing 16 bit (Pixel Value Counting 0 to 15)

a) Internal generation

| | |
|---|---|
| Clock Cycle 0 | **0000 0001 0002 0003 0004** |
| Clock Cycle 1 | **0005 0006 0007 0008 0009** |
| Clock Cycle 2 | **000A 000B 000C 000D 000E** |

b) Data stream after 16 bit switch

| | |
|---|---|
| Clock Cycle 0 | **0004 0003 0002 0001 0000** |
| Clock Cycle 1 | **0009 0008 0007 0006 0005** |
| Clock Cycle 2 | **000E 000D 000C 000B 000A** |

Packing 12 bit (pixel value counting 0 to 19)

a) Internal generation

| | |
|---|---|
| Clock Cycle 0 | 0000 0100 2003 0040 0500 |
| Clock Cycle 1 | 6007 0080 0900 A00B 00C0 |
| Clock Cycle 2 | 0D00 E00F 0100 1101 2013 |

b) Data stream after 16 bit switch

| | |
|---|---|
| Clock Cycle 0 | 0500 0040 2003 0100 0000 |
| Clock Cycle 1 | 00C0 A00B 0900 0080 6007 |
| Clock Cycle 2 | 2013 1101 0100 E00F 0D00 |

Packing 12 bit (all pixel values at 0x7FF)

a) Internal generation

| | |
|---|---|
| Clock Cycle 0 | 7FF7 FF7F F7FF 7FF7 FF7F |
| Clock Cycle 1 | F7FF 7FF7 FF7F F7FF 7FF7 |
| Clock Cycle 2 | FF7F F7FF 7FF7 FF7F F7FF |

b) Data stream after 16 bit switch

| | |
|---|---|
| Clock Cycle 0 | FF7F 7FF7 F7FF FF7F 7FF7 |
| Clock Cycle 1 | 7FF7 F7FF FF7F 7FF7 F7FF |
| Clock Cycle 2 | F7FF FF7F 7FF7 F7FF FF7F |

## 2.4 Lookup Table

### 2.4.1 Data Compression with square root

The compression with the square root reduces the range of values considering the photon noise.

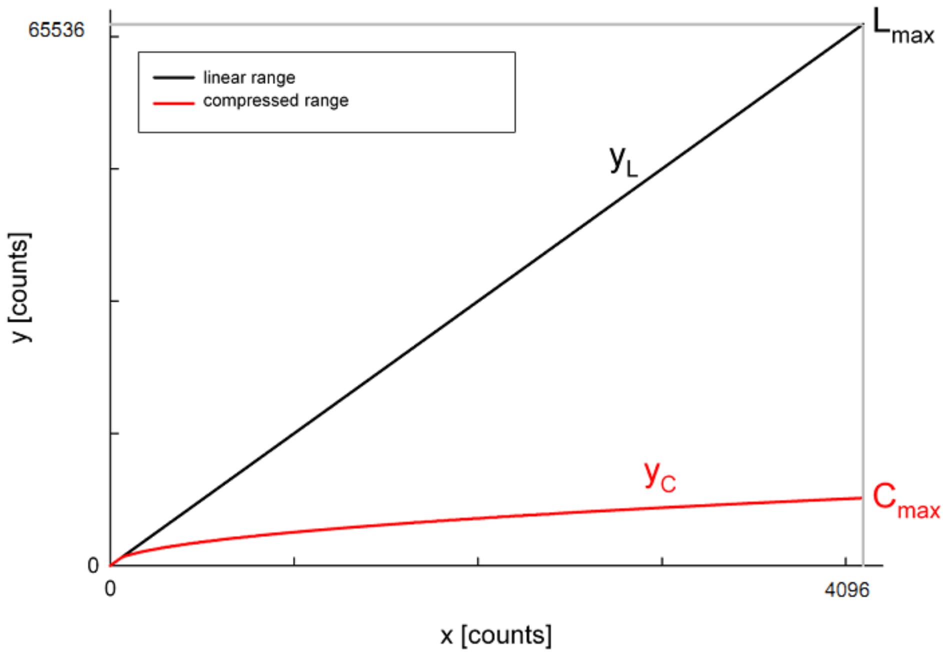



figure 2.8: data compression with square root function

Linear range of values:

$$L = \{0, 65535\} \text{ (16 Bit)} \qquad (2.1)$$

Compressed data range:

$$C = \{0, 4095\} \text{ (12 Bit)} \qquad (2.2)$$

The compress function is divided into a linear and a nonlinear part with intersection point $x_S$= 256:

$$\boxed{y_C(x) = \begin{cases} x, & x \leq 256 \\ \sqrt{256x}, & x > 256 \end{cases}, \quad x \in L, y_C \in C} \qquad (2.4)$$

### 2.4.2 Square root implementation

The formula given in 2.4 is implemented into the FPGA and could be enabled with the “set look up table” command. The output is linear from 0 to 256. The square root result is accumulated with 0.5 and rounded to an integer value. The formula is y = (int) (sqrt(256*x) + 0.5). The maximum output value is limited to 4095.

# 3 Computer Data Processing

At the PC side there are different instances involved in the image transfer process. The image data is sent through the two CameraLink cables to the grabber. The grabber is controlled by the driver through a specific SDK provided by its manufacturer. PCO provides a high level access to the grabber through the pco.sdk.

Figure 3.1 gives an overview of the internal image processing. The grabber transfers image data through DMA into previously allocated memory buffers.

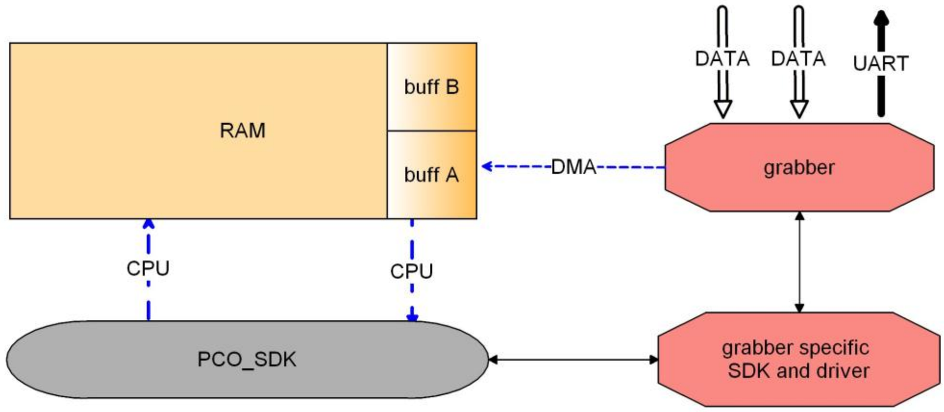



Figure 3.1: image transfer from grabber to RAM

Because camera and grabber are involved in the data processing, the setup of the grabber must match the setup of the camera and vice versa. Also the preparation of the PC-memory buffers, where the images should be stored must match the settings of grabber and camera. The pco.edge produces a high data throughput (data stream of up to 810MB/s), so it is important that all used components are able to support this high data rate in all circumstances.

As described above the camera holds the actual settings until it is powered off and starts with default settings. Therefore the initialization process of the camera application has always to start from a well-defined state. This can be accomplished by:

- reading the actual settings of the camera and setting up the grabber and memory buffers according to these settings
- reset the camera to its default settings, change the camera settings as requested by the current process and setting up grabber and memory according to these settings

Trying to use the camera, grabber and memory management with wrong setup might cause:

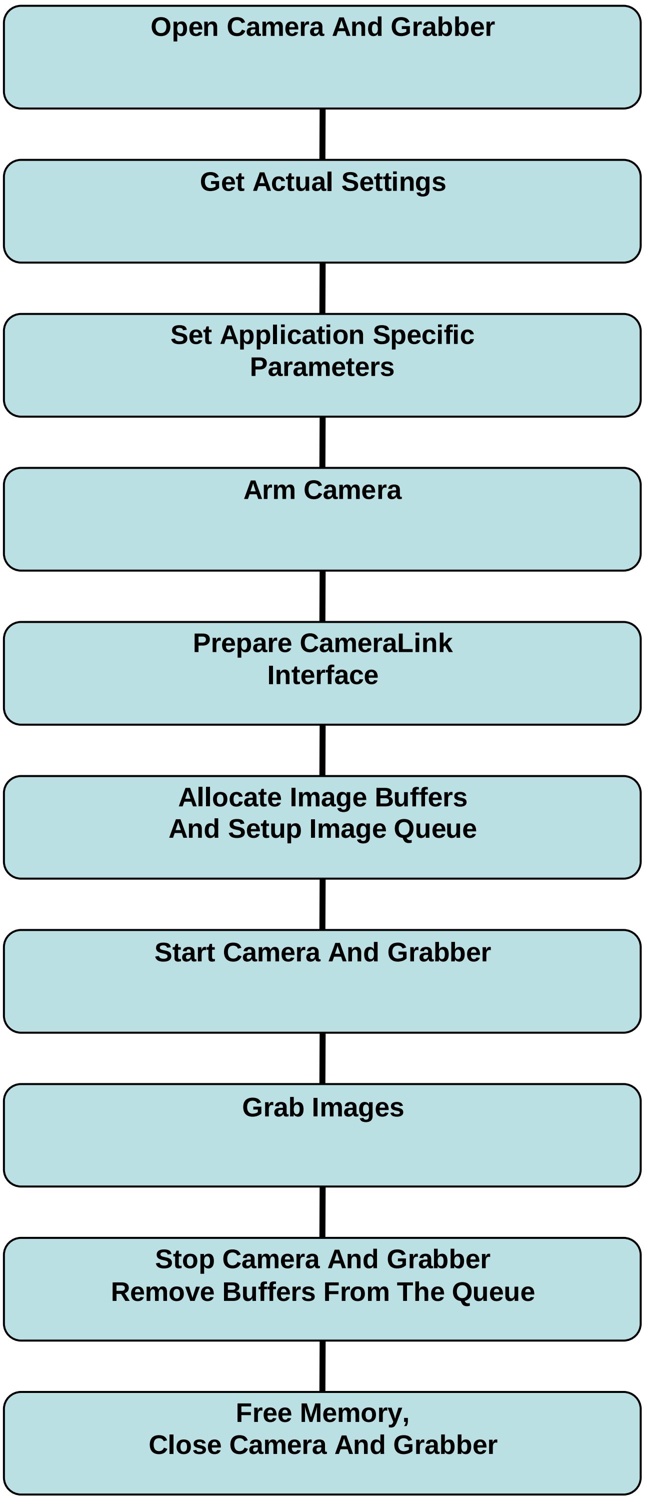
- system freezes
- errors in the camera, which then will stop further image transfer
- errors in the grabber SDK, which then will stop further image transfer
- misaligned and wrong image data in memory

When using the pco.sdk most of the grabber settings are processed internally in the layer of the attached grabber. The software engineer using the SDK just has to follow the description in the next chapter.

For grabbers and/or operating systems, which are not supported by the pco.sdk, the software engineer has to use the grabber specific SDK to initialize and setup the grabber and to send the camera commands. The software engineer should have excellent knowledge of the grabber SDK and should be familiar with the "PCO Camera Control Commands" in order to send the commands to the camera by using the grabber SDK.

Note: For some of these tasks example source code is available from PCO.

## 3.1 Application process chart



## 3.2 Description of process steps

Description of all necessary steps to set up the camera and grabber. For all necessary steps pco.sdk functions are included.

### 3.2.1 Open Camera and Grabber

Open a connection to the camera. Search for installed Grabber. Test if camera is connected to a Grabber. Scan with all baudrates, which are supported by the camera and the grabber, to find also cameras, which have been initialized before.
`PCO_OpenCamera()`

### 3.2.2 Get actual Settings

Get camera type to check the current shutter mode (rolling shutter or global shutter) of the camera.
`PCO_GetCameraType()`

Get the camera descriptor.
`PCO_GetCameraDescription()`

Get actual CameraLink transfer parameters
`PCO_GetTransferParameter()`

Get active lookup table
`PCO_GetActiveLookupTable()`

Check if camera is already in recording state=ON. If camera is recording and no camera settings should be changed, proceed with Step 3.2.5
`PCO_GetRecordingState()`

### 3.2.3 Set application specific Parameters

Stop camera.
`PCO_SetRecordingState(OFF)`

To get a well defined startup state issue a reset settings to default
`PCO_ResetSettingsToDefault()`

Change all camera parameters (e.g. exposuretime, ROI, pixelrate, etc.) to match the requirements of your application.
`PCO_SetDelayExposureTime(),PCO_SetROI(),PCO_SetPixelRate()`

### 3.2.4 Arm Camera

Arm Camera is verifying all settings and prepares the camera for recording. This could take several seconds.
`PCO_ArmCamera()`

### 3.2.5 Prepare CameraLink Interface

Get the actual resolution.
`PCO_GetSizes()`

Get actual setting of the pixelrate of the sensor.
`PCO_GetPixelRate()`

Get actual transfer parameters
`PCO_GetTransferParameter()`

If the pixelrate is 95MHz the transfer parameter dataformat should be set to `PCO_CL_DATAFORMAT_5x16` for all resolutions

If pixelrate is 286MHz and the horizontal resolution is equal or below 1920 pixel transfer parameters dataformat can be set to `PCO_CL_DATAFORMAT_5x16`

If pixelrate is 286MHz and the horizontal resolution is above 1920 pixel transfer parameters dataformat must be set to `PCO_CL_DATAFORMAT_5x12`, `PCO_CL_DATAFORMAT_5x12L`, `PCO_CL_DATAFORMAT_5x12R` or `PCO_CL_DATAFORMAT_10x8` and the according lookup table has to be set.

In addition to the CL_DATAFORMAT one of the SCCMOS _FORMAT's must be selected.

The transfer parameter 'transmit' must be set to 1 and the baudrate can be set to 115200. All other parameters must not be changed.
`PCO_SetTransferParameter()`

Set the lookup table according to the dataformat
`PCO_SetActiveLookupTable()`

If the transfer parameters or the active lookup table is changed, an additional 'arm' command must be sent to the camera.
`PCO_ArmCamera()`

Set grabber parameters according to the actual resolution and transfer parameter dataformat. `PCO_CamLinkSetImageParameters()`

### 3.2.6 Allocate Image Buffers and setup Image Queue

Allocate Image Buffers of appropriate size for the different data formats:

1) `PCO_CL_DATAFORMAT_5x16 or PCO_CL_DATAFORMAT_5x12L`
Horizontal resolution * vertical resolution * 2 bytes

2) `PCO_CL_DATAFORMAT_5x12 or PCO_CL_DATAFORMAT_5x12R`
(Horizontal resolution * vertical resolution * 12) / 16 * 2 bytes

3) `PCO_CL_DATAFORMAT_10x8`
Horizontal resolution * vertical resolution * 1 bytes

At least 2 better 4 buffers or best an image array should be allocated to be able to store all grabbed images
`PCO_AllocateBuffer()`

Add buffers to the image queue
`PCO_AddBufferEx()`

### 3.2.7 Start Camera

Start camera.
`PCO_SetRecordingState(ON)`

### 3.2.8 Grab Images

Images are grabbed from the camera and stored in the buffers of the queue. An event is fired to inform the application that an image has been transmitted. The status of the buffer must be checked for error conditions.

In case of the `PCO_CL_DATAFORMAT_5x12L` the image data from the camera is unpacked and recalculated to 16bit values.

In case of the `PCO_CL_DATAFORMAT_5x12` the image data from the camera is unpacked.
In case of the `PCO_CL_DATAFORMAT_5x12R` the image data from the camera is not changed.

After the application finishes its processing off a received buffer it can be added to the image queue again.
`PCO_GetBufferStatus(), PCO_AddBufferEx()`

### 3.2.9 Stop Camera

Stop camera.
`PCO_SetRecordingState()`

Clear image queue
`PCO_CancelImages()`

### 3.2.10 Free memory and Close Camera

Free allocated Image Buffers
`PCO_FreeBuffer()`
Close Camera
`PCO_CloseCamera()`

## 3.3 CL_DATAFORMAT Translation

**Input Pattern**

12bit packed

```
0FF1 0010 1102 1031 0410
```

**Output Pattern**

*PCO_CL_DATAFORMAT_5x12*

```
00FF 0100 0101 0102 0103
0104 .... .... .... ....
```

*PCO_CL_DATAFORMAT_5x12L*

```
00FF 0100 0102 0104 0106
010A .... .... .... ....
```

*PCO_CL_DATAFORMAT_5x12R*

```
0FF1 0010 1102 1031 0410
.... .... .... .... ....
```

# 4 Rolling Shutter and Global Shutter

The pco.edge offers two different modes. On one hand global shutter and on the other hand rolling shutter. These modes can be selected by the PCO_SetCameraSetup command.

```
SC2_SDK_FUNC int WINAPI PCO_GetCameraSetup(HANDLE ph, WORD wType, DWORD* dwSetup,
WORD wLen)
```

```
#define PCO_EDGE_SETUP_ROLLING_SHUTTER 0x00000001        //
rolling shutter
#define PCO_EDGE_SETUP_GLOBAL_SHUTTER  0x00000002        //
global shutter
```

After the camera is switched to another mode, the camera must be rebooted and a new connection should be executed.

## 4.1 Rolling Shutter

CameraLink 10-tap with 85 MHz clock frequency offers a data rate of about 810 MB / s. The following list shows the data rate of the sensor for different resolutions, different clock speeds and different bit resolutions. For the frame rate only the vertical size of the ROI is important. The horizontal size defines the data rate.

| X | Y | pixelrate | fps | bit per pixel | data rate | ok |
|---|---|---|---|---|---|---|
| 2560 | 2160 | 286 | 100 | 16 | 1055 | ✗ |
| 2560 | 1080 | 286 | 200 | 16 | 1055 | ✗ |
| 1920 | 2160 | 286 | 100 | 16 | 792 | ✓ |
| 1920 | 1080 | 286 | 200 | 16 | 790 | ✓ |
| 1600 | 1200 | 286 | 180 | 16 | 658 | ✓ |
| 2560 | 2160 | 286 | 100 | LUT[1] | 791 | ✓ |
| 1920 | 1080 | 286 | 200 | LUT[1] | 790 | ✓ |
| 1600 | 1200 | 286 | 180 | LUT[1] | 494 | ✓ |
| 2560 | 2160 | 95.3 | 33 | 16 | 352 | ✓ |

By operating at full speed and high horizontal resolution the CameraLink data rate is too small in order to transfer all image data to the grabber. Therefore the user has to select 16-to-12-bit-LUT compression for transferring only 12 bit per pixel.

```
SC2_SDK_FUNC int WINAPI PCO_SetActiveLookupTable(HANDLE ph, WORD *wIdentifier, WORD
*wParameter)
```

Also the CameraLink output format must be changed from 16 to 12 bit. Thus the PCO_SetTransferParameter command (see previous chapter) has to be set to the appropriate format.

[1] LUT: lookup table compression from 16 to 12 bit per pixel

## 4.2 Global Shutter

In global shutter mode two images are transferred from the camera to the grabber. These two 12bit images are added together in one frame of double height. The grabber has onboard routines to calculate the resulting 16 bit image. The following table shows the data rate of the CameraLink connection and the PCIe bus data rate:

CameraLink

| X | Y | pixelrate | fps | bit per pixel | data rate | ok |
|---|---|---|---|---|---|---|
| 2560 | 4320 | 286 | 50 | 12 | 791 | ✓ |
| 1920 | 2160 | 286 | 100 | 12 | 593 | ✓ |
| 2560 | 4320 | 95.3 | 17 | 12 | 264 | ✓ |
| 1920 | 2160 | 95.3 | 33 | 12 | 198 | ✓ |

PCIe

| X | Y | fps | bit per pixel | data rate |
|---|---|---|---|---|
| 2560 | 2160 | 50 | 16 | 527 |
| 1920 | 1800 | 100 | 16 | 395 |
| 2560 | 2160 | 17 | 16 | 176 |
| 1920 | 1800 | 33 | 16 | 132 |

# maximize the moment

PCO AG was founded in 1987. The company headquarters in Kelheim employs more than 50 specialists in the development and production of optimized, fast, sensitive camera systems for scientific applications. PCO's range of products includes digital camera systems featuring high dynamics, extremely high sensitivity, high resolution, high speed, and extremely low noise, which are sold in industrial and scientific markets all over the world.

## Cameras for every point of view.

The systems produced by PCO AG are cameras and scientific measuring instruments at the same time. Our high-tech systems are mostly the result of manual labor: over 50 highly specialized employees handle development and production at the Kelheim site. We deliver roughly 4.000 cameras a year to customers all over the world. As in every cutting edge technology, dialogue with the user is the main focus of PCO's approach. Worldwide representatives, in cooperation with the in-house marketing division and technical support team, ensure that PCO camera systems are developed in step with the individual requirements of our customers.

pco.
imaging

pco.
imaging