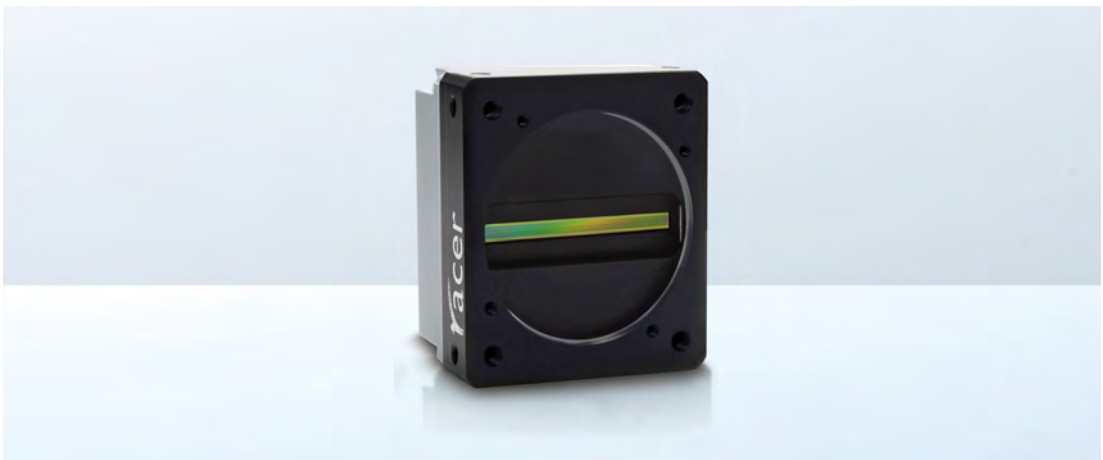
Basler racer



USER'S MANUAL FOR GigE VISION CAMERAS

Document Number: AW001183
Version: 06 Language: 000 (English)
Release Date: 08 September 2016

BASLER
the power of sight

For customers in the USA

This equipment has been tested and found to comply with the limits for a Class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communications. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

You are cautioned that any changes or modifications not expressly approved in this manual could void your authority to operate this equipment.

The shielded interface cable recommended in this manual must be used with this equipment in order to comply with the limits for a computing device pursuant to Subpart B of Part 15 of FCC Rules.

For customers in Canada

This apparatus complies with the Class A limits for radio noise emissions set out in Radio Interference Regulations.

Pour utilisateurs au Canada

Cet appareil est conforme aux normes Classe A pour bruits radioélectriques, spécifiées dans le Règlement sur le brouillage radioélectrique.

Life support applications

These products are not designed for use in life support appliances, devices, or systems where malfunction of these products can reasonably be expected to result in personal injury. Basler customers using or selling these products for use in such applications do so at their own risk and agree to fully indemnify Basler for any damages resulting from such improper use or sale.

Warranty note

Do not open the housing of the camera. The warranty becomes void if the housing is opened.

Contacting Basler Support Worldwide

Europe, Middle East, Africa

Basler AG
An der Strusbek 60–62
22926 Ahrensburg
Germany

Tel. +49 4102 463 515
Fax +49 4102 463 599

support.europe@baslerweb.com

The Americas

Basler, Inc.
855 Springdale Drive, Suite 203
Exton, PA 19341
USA

Tel. +1 610 280 0171
Fax +1 610 280 7608

support.usa@baslerweb.com

Asia-Pacific

Basler Asia Pte. Ltd.
35 Marsiling Industrial Estate Road 3
#05–06
Singapore 739257

Tel. +65 6367 1355
Fax +65 6367 1255

support.asia@baslerweb.com

www.baslerweb.com

Table of Contents

1 Specifications, Requirements, and Precautions

This chapter lists the camera models covered by the manual. It provides the general specifications for those models and the basic requirements for using them.

This chapter also includes specific precautions that you should keep in mind when using the cameras. We strongly recommend that you read and follow the precautions.

1.1 Models

The current Basler racer GigE Vision camera models are listed in the top row of the specification tables on the next pages of this manual. The camera models are differentiated by their resolution and their maximum line rate at full resolution.

Unless otherwise noted, the material in this manual applies to all of the camera models listed in the tables. Material that only applies to a particular camera model or to a subset of models will be so designated.

1.2 General Specifications

Specification	raL2048-48gm	raL4096-24gm
Resolution	2048 pixels	4096 pixels
Sensor Type	Awaiba DR-2k-7 Monochrome Linear CMOS	Awaiba DR-4k-7 Monochrome Linear CMOS
Pixel Size	7 µm x 7 µm	
Max Line Rate	51 kHz	26 kHz
Min Line Rate	No minimum when an external line trigger signal is used 100 Hz when an external line trigger signal is not used	
Mono/Color	Mono	
Data Output Type	Fast Ethernet (100 Mbit/s) or Gigabit Ethernet (1000 Mbit/s)	
Pixel Data Formats	Mono 8 Mono 12 Mono 12 Packed YUV 4:2:2 Packed YUV 4:2:2 (YUYV) Packed	
ADC Bit Depth	12 bit	
Synchronization	Via external trigger signal, via software or free run	
Exposure Control	Programmable via the camera API	
Camera Power Requirements	+12 VDC (- 10 %) to +24 VDC (+ 5 %), < 1 % ripple, supplied via the camera's 6-pin connector	
Max Power Consumption (at 12 VDC)	3 W	4 W
I/O Lines	3 input lines and 2 output lines	
Lens Adapter	Universal camera front, suitable for lens mount adapters with the following lens mounts: C-mount (2k cameras), F-mount, M42x0.75-mount, M42x1.0-mount, M42x1.0-mount (FBD 45.56 mm), M58x0.75-mount. See Section 1.5.4 on page 16 for information about selecting a suitable lens adapter for your camera.	
Size (L x W x H)	36.12 mm x 56 mm x 62 mm (without lens adapter or connectors) 50.83 mm x 56 mm x 62 mm (with C-mount lens adapter and connectors) 79.92 mm x 56 mm x 62 mm (with F-mount lens adapter and connectors) 49.42 mm x 56 mm x 62 mm (with M42-mount lens adapter and connectors) 78.98 mm x 56 mm x 62 mm (with M42-mount FBD 45.56 mm lens adapter and connectors) 53.42 mm x 56 mm x 62 mm (with M58-mount lens adapter and connectors)	

Table 1: General Specifications - 2k and 4k Mono Cameras

Specification	raL2048-48gm	raL4096-24gm
Weight	≈ 240 g (typical) without lens adapter ≈ 270 g (typical) with C-mount lens adapter and connectors ≈ 330 g (typical) with F-mount lens adapter and connectors ≈ 260 g (typical) with M42-mount lens adapter and connectors ≈ 310 g (typical) with M42-mount FBD 45.56 lens adapter and connectors ≈ 270 g (typical) with M58-mount lens adapter and connectors	
Conformity	CE, RoHS, FCC, UL, GenICam, GigE Vision, IP30 The CE Conformity Declaration is available on the Basler website: www.baslerweb.com	
Software	Basler pylon Camera Software Suite (version 4.0 or higher) Available for Windows (x86, x64) and Linux (x86, x64, ARM).	

Table 1: General Specifications - 2k and 4k Mono Cameras

Specification	raL6144-16gm	raL8192-12gm
Resolution	6144 pixels	8192 pixels
Sensor Type	Awaiba DR-6k-7 Monochrome Linear CMOS	Awaiba DR-8k-3.5 Monochrome Linear CMOS
Pixel Size	7 µm x 7 µm	3.5 µm x 3.5 µm
Max Line Rate	17 kHz	12 kHz
Min Line Rate	No minimum when an external line trigger signal is used 100 Hz when an external line trigger signal is not used	
Mono/Color	Mono	
Data Output Type	Fast Ethernet (100 Mbit/s) or Gigabit Ethernet (1000 Mbit/s)	
Pixel Data Formats	Mono 8 Mono 12 Mono 12 Packed YUV 4:2:2 Packed YUV 4:2:2 (YUYV) Packed	
ADC Bit Depth	12 bit	
Synchronization	Via external trigger signal, via software or free run	
Exposure Control	Programmable via the camera API	
Camera Power Requirements	+12 VDC (- 10 %) to +24 VDC (+ 5 %), < 1 % ripple, supplied via the camera's 6-pin connector	
Max Power Consumption (at 12 VDC)	< 4.5 W	< 5.5 W
I/O Lines	3 input lines and 2 output lines	
Lens Adapter	Universal camera front, suitable for lens mount adapters with the following lens mounts: F-mount, M42x0.75-mount, M42x1.0-mount (FBD 16 mm), M42x1.0-mount (FBD 45.56 mm), M58x0.75-mount. See Section 1.5.4 on page 16 for information about selecting a suitable lens adapter for your camera.	
Size (L x W x H)	36.12 mm x 56 mm x 62 mm (without lens adapter or connectors) 79.92 mm x 56 mm x 62 mm (with F-mount lens adapter and connectors) 49.42 mm x 56 mm x 62 mm (with M42-mount lens adapter and connectors) 78.98 mm x 56 mm x 62 mm (with M42-mount FBD 45.56 mm lens adapter and connectors) 53.42 mm x 56 mm x 62 mm (with M58-mount lens adapter and connectors)	
Weight	≈ 240 g (typical) without lens adapter ≈ 330 g (typical) with F-mount lens adapter and connectors ≈ 260 g (typical) with M42-mount lens adapter and connectors ≈ 310 g (typical) with M42-mount FBD 45.56 lens adapter and connectors ≈ 270 g (typical) with M58-mount lens adapter and connectors	

Table 2: General Specifications - 6k and 8k Mono Cameras

Specification	raL6144-16gm	raL8192-12gm
Conformity	CE, RoHS, FCC, UL, GenICam, GigE Vision, IP30 The CE Conformity Declaration is available on the Basler website: www.baslerweb.com	
Software	Basler pylon Camera Software Suite (version 4.0 or higher) Available for Windows (x86, x64) and Linux (x86, x64, ARM).	

Table 2: General Specifications - 6k and 8k Mono Cameras

Specification	raL12288-8gm
Resolution	12288 pixels
Sensor Type	Awaiba DR-12k-3.5 Monochrome Linear CMOS
Pixel Size	3.5 µm x 3.5 µm
Max Line Rate	8 kHz
Min Line Rate	No minimum when an external line trigger signal is used 100 Hz when an external line trigger signal is not used
Mono/Color	Mono
Data Output Type	Fast Ethernet (100 Mbit/s) or Gigabit Ethernet (1000 Mbit/s)
Pixel Data Formats	Mono 8 Mono 12 Mono 12 Packed YUV 4:2:2 Packed YUV 4:2:2 (YUYV) Packed
ADC Bit Depth	12 bit
Synchronization	Via external trigger signal, via software or free run
Exposure Control	Programmable via the camera API
Camera Power Requirements	+12 VDC (- 10 %) to +24 VDC (+ 5 %), < 1 % ripple, supplied via the camera's 6-pin connector
Max Power Consumption (at 12 VDC)	< 6.5 W
I/O Lines	3 input lines and 2 output lines
Lens Adapter	Universal camera front, suitable for lens mount adapters with the following lens mounts: F-mount, M42x0.75-mount, M42x1.0-mount (FBD 16 mm), M42x1.0-mount (FBD 45.56 mm), M58x0.75-mount. See Section 1.5.4 on page 16 for information about selecting a suitable lens adapter for your camera.
Size (L x W x H)	36.12 mm x 56 mm x 62 mm (without lens adapter or connectors) 79.92 mm x 56 mm x 62 mm (with F-mount lens adapter and connectors) 49.42 mm x 56 mm x 62 mm (with M42-mount lens adapter and connectors) 78.98 mm x 56 mm x 62 mm (with M42-mount FBD 45.56 mm lens adapter and connectors) 53.42 mm x 56 mm x 62 mm (with M58-mount lens adapter and connectors)
Weight	≈ 240 g (typical) without lens adapter ≈ 330 g (typical) with F-mount lens adapter and connectors ≈ 260 g (typical) with M42-mount lens adapter and connectors ≈ 310 g (typical) with M42-mount FBD 45.56 lens adapter and connectors ≈ 270 g (typical) with M58-mount lens adapter and connectors

Table 3: General Specifications - 12k Mono Cameras

Specification	raL12288-8gm
Conformity	CE, RoHS, FCC, UL, GenICam, GigE Vision, IP30 The CE Conformity Declaration is available on the Basler website: www.baslerweb.com
Software	Basler pylon Camera Software Suite (version 4.0 or higher) Available for Windows (x86, x64) and Linux (x86, x64, ARM).

Table 3: General Specifications - 12k Mono Cameras

1.3 Accessories

Fig. 1: Basler Accessories

Basler's cooperation with carefully selected suppliers means you get accessories you can trust which makes building a high-performance image processing system hassle-free.

Key Reasons for Choosing Lenses, Cables, and Other Accessories from Basler

- Perfect match for Basler cameras
- One-stop-shopping for your image processing system
- Stable performance through highest quality standards
- Easy integration into existing systems
- Expert advice during selection process

See the Basler website for information about Basler's extensive accessories portfolio (e.g. cables, lenses, host adapter cards, switches): www.baslerweb.com

1.4 Spectral Response

The following graph shows the quantum efficiency curve for monochrome cameras.

The quantum efficiency curve excludes lens characteristics and light source characteristics.

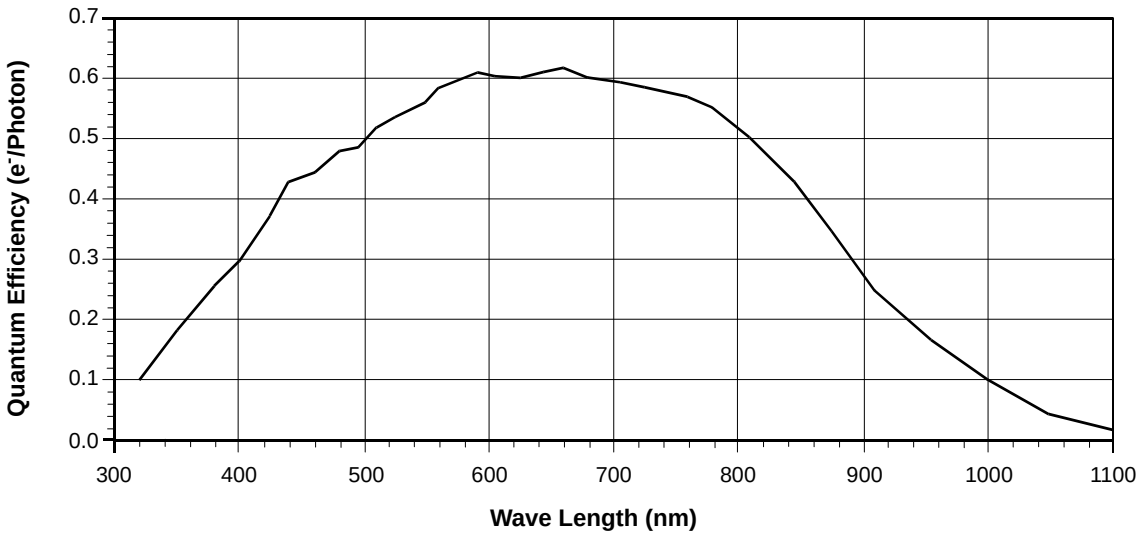


Fig. 2: Quantum Efficiency of the Monochrome Sensor in 12 Bit Depth Mode (Based on Sensor Vendor Information)

1.5 Mechanical Specifications

1.5.1 Camera Dimensions and Mounting Points

The cameras are manufactured with high precision. Planar, parallel, and angular sides ensures precise mounting with high repeatability.

The camera housings conform to the IP30 protection class provided the camera front is covered by the protective plastic seal that is shipped with the camera.

The camera’s dimensions in millimeters are as shown in the drawings below.

Camera housings are equipped with four mounting holes (4 x M4; 6.3 deep) on the front and two mounting holes (8 x M4; 6.3 deep) on each side as shown in the drawings. Four additional holes (4 x M2.5; 3.3 deep) are present on the camera front for mounting the lens mount adapter.

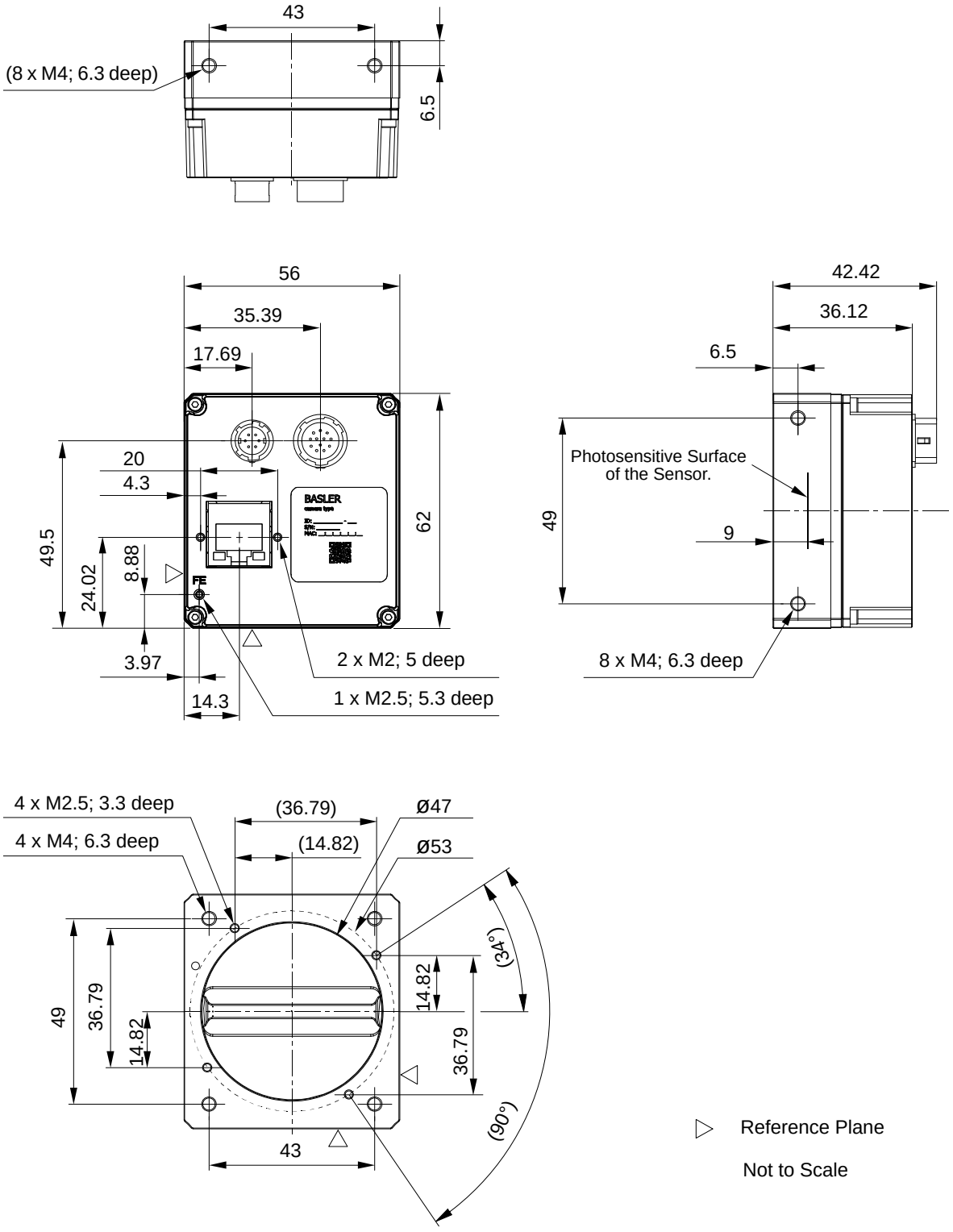


Fig. 3: Mechanical Dimensions (in mm)

1.5.2 Sensor Line Location

The location of the sensor line in the mono cameras is shown in Fig. 4. The sensor lines of different camera models vary in length, depending on maximum resolution and pixel size. As an example, a sensor line of maximum length, as applies to 6k and 12k cameras, is shown in Fig. 4.

A marker hole in the camera's front indicates the side of the camera where the pixel numbering for each sensor line starts. The first pixel is numbered one.

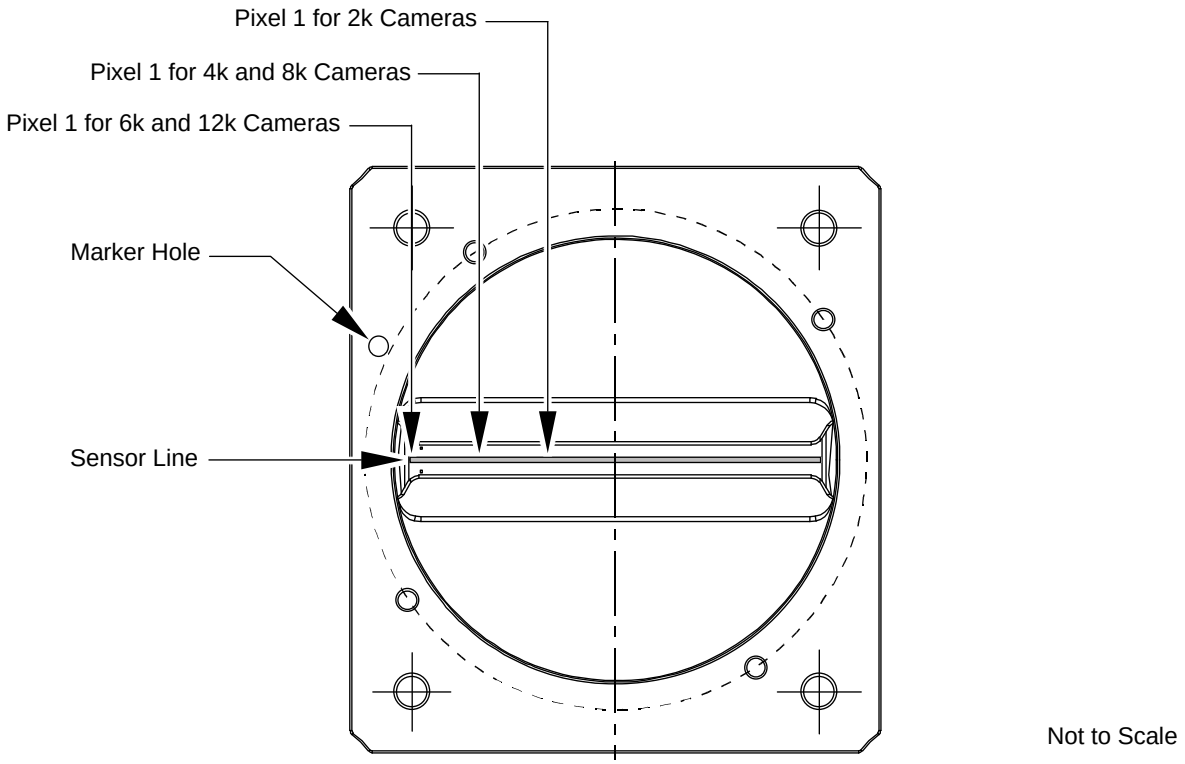


Fig. 4: Mono Sensor Line Location with Approximate Starting Points (Pixel 1) for Pixel Numbering

1.5.3 Lens Adapter Dimensions

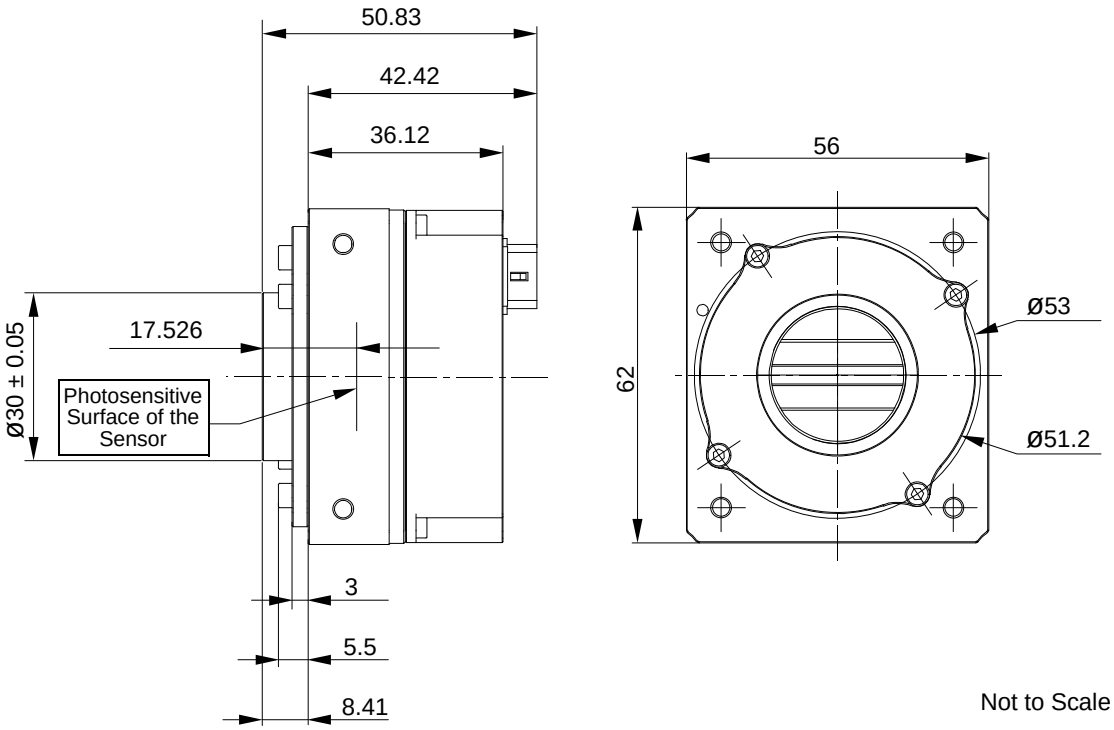


Fig. 5: C-mount Adapter on a racer GigE Camera; Dimensions in mm

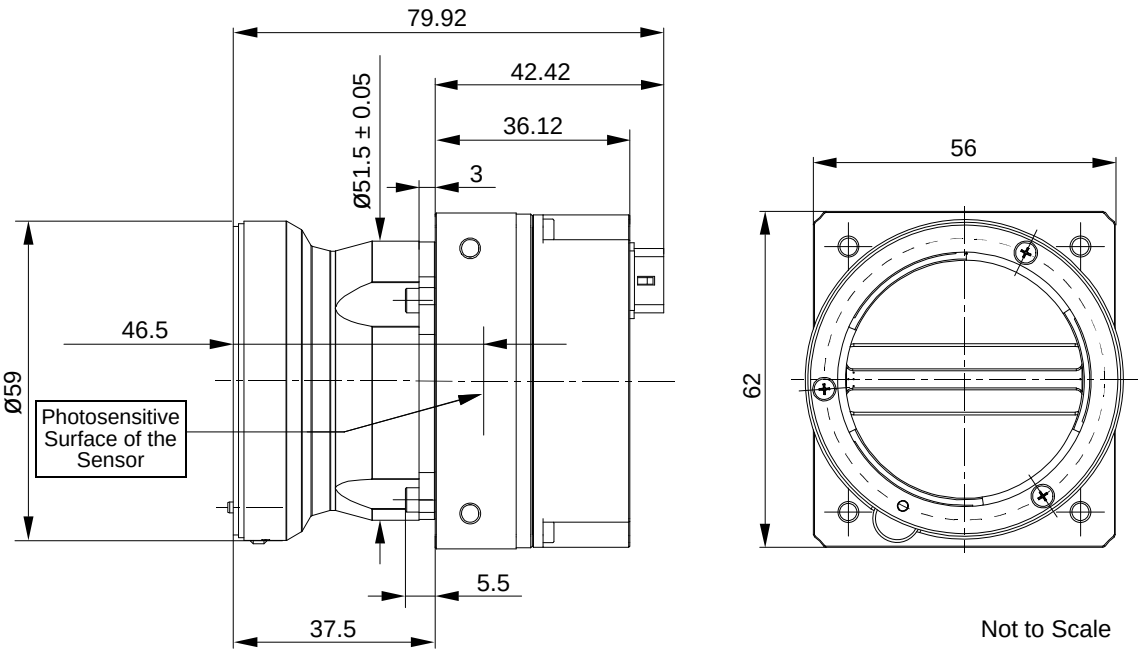


Fig. 6: F-Mount Adapter on a racer GigE Camera; Dimensions in mm

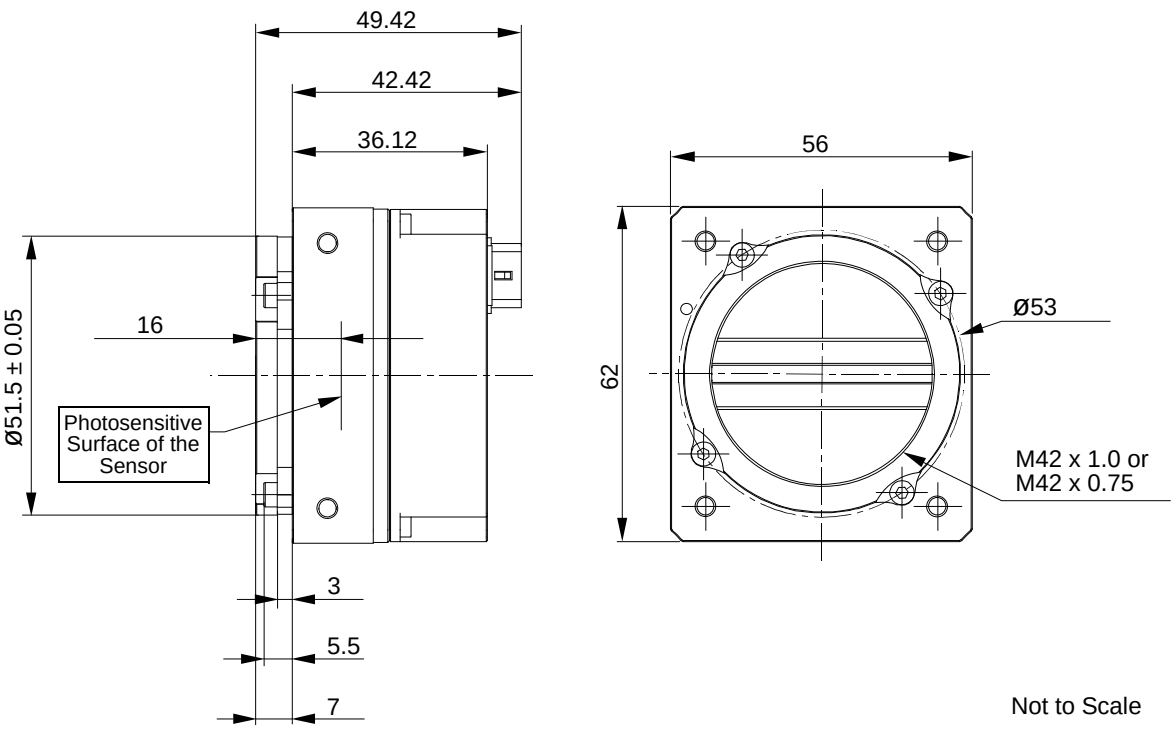


Fig. 7: M42 x 1.0 or M42 x 0.75 Mount Adapter on a racer GigE Camera; Dimensions in mm

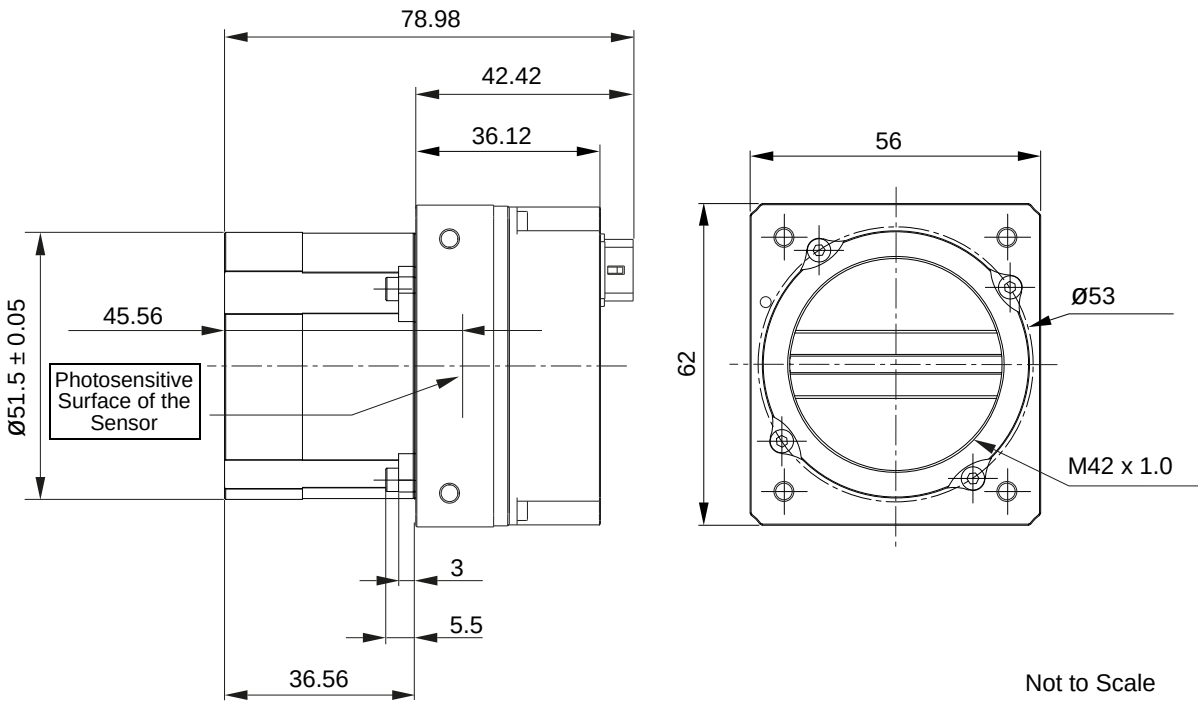


Fig. 8: M42 x 1.0 FBD 45.56 mm Mount Adapter on a racer GigE Camera; Dimensions in mm

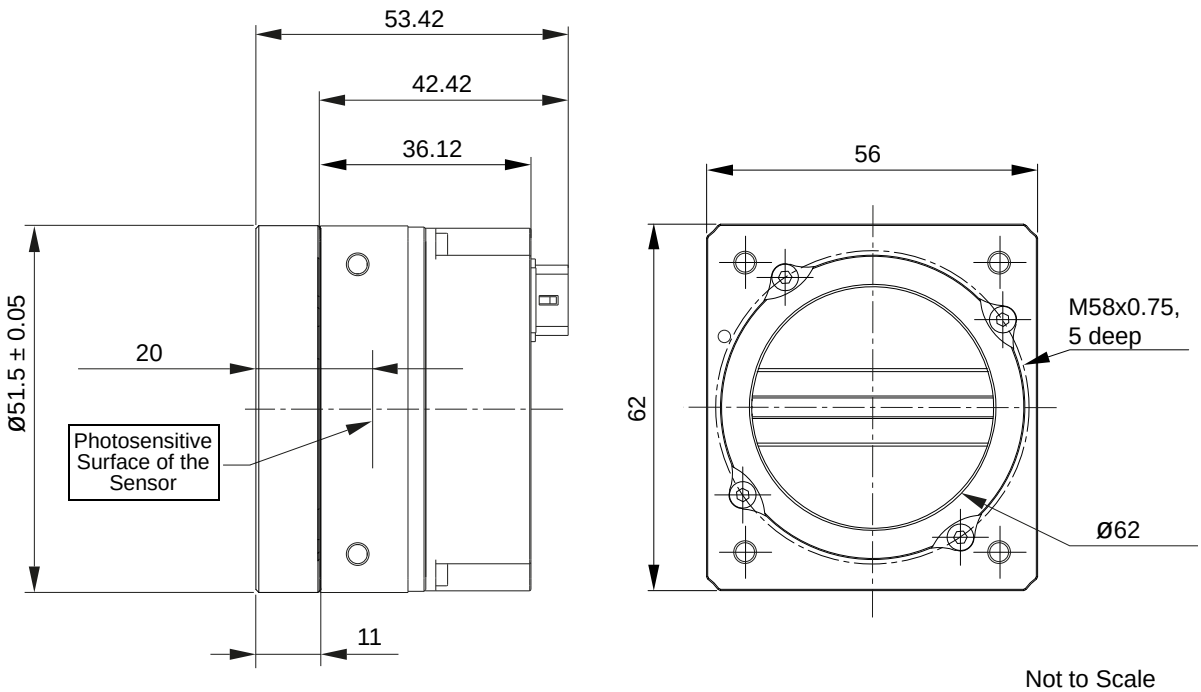


Fig. 9: M58 x 0.75 Mount Adapter on a racer GigE Camera; Dimensions in mm

1.5.4 Selecting the Optimum Lens Adapter

The camera's scope of delivery does not include a lens mount adapter. It is needed to attach a lens to a camera. You must order a lens adapter separately as an accessory.

The optimum choice of a lens mount adapter depends on the lens and on the resolution that will be used. The recommended combinations of lens mount adapters and camera models are indicated in the following table:

Lens Adapter	Camera Model				
	raL2048	raL4096	raL6144	raL8192	raL12288
C-mount	●	-	-	-	-
F-mount	●	●	●	●	●
M42 x 1.0	●	●	●1)	●	●1)
M42 x 0.75	●	●	●1)	●	●1)
M42 x 1.0 FBD 45.56	●	●	●1)	●	●1)
M58	●	●	●1)	●	●1)

Table 4: Recommended Lens Adapters Depending on Camera Model (● = recommended, - = not recommended. 1) To ensure coverage of the entire sensor, contact Basler technical support for assistance when choosing a lens.)

1.5.5 Attaching a Lens Adapter

Use the four M2.5 setscrews supplied with the lens adapter to lock the lens adapter to the camera. See Fig. 3 on page 11 for information where to place the M2.5 setscrews.

NOTICE

Screwing with excessive torque can damage the camera, lens adapter or setscrews.

When screwing in the supplied M2.5 setscrews, make sure to never exceed a torque of 0.4 Nm.

1.6 Software Licensing Information

1.6.1 LWIP TCP/IP Licensing

The software in the camera includes the LWIP TCP/IP implementation. The copyright information for this implementation is as follows:

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
3. The name of the author may not be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE AUTHOR "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED.

IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

1.6.2 LZ4 Licensing

The software in the camera includes the LZ4 implementation. The copyright information for this implementation is as follows:

LZ4 - Fast LZ compression algorithm

Copyright (C) 2011-2013, Yann Collet.

BSD 2-Clause License (www.opensource.org/licenses/bsd-license.php)

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

1.7 Avoiding EMI and ESD Problems

The cameras are frequently installed in industrial environments. These environments often include devices that generate electromagnetic interference (EMI) and they are prone to electrostatic discharge (ESD). Excessive EMI and ESD can cause problems with your camera such as false triggering or can cause the camera to suddenly stop capturing images. EMI and ESD can also have a negative impact on the quality of the image data transmitted by the camera.

To avoid problems with EMI and ESD, you should follow these general guidelines:

- Always use high quality shielded cables. The use of high quality cables is one of the best defenses against EMI and ESD.
- Try to use camera cables that are as short as possible and try to run the camera cables and power cables parallel to each other. Avoid coiling camera cables. If the cables are too long, use a meandering path rather then coiling the cables.
- Avoid placing camera cables parallel to wires carrying high-current, switching voltages such as wires supplying stepper motors or electrical devices that employ switching technology. **Placing camera cables near to these types of devices can cause problems with the camera.**
- Attempt to connect all grounds to a single point, e.g., use a single power outlet for the entire system and connect all grounds to the single outlet. This will help to avoid large ground loops. (Large ground loops can be a primary cause of EMI problems.)
- Use a line filter on the main power supply.
- Install the camera and camera cables as far as possible from devices generating sparks. If necessary, use additional shielding.
- Decrease the risk of electrostatic discharge by taking the following measures:
  - Use conductive materials at the point of installation (e.g., floor, workplace).
  - Use suitable clothing (cotton) and shoes.
  - Control the humidity in your environment. Low humidity can cause ESD problems.

A functional earth connection on the back of the camera (labelled "FE", see Fig. 16 on page 50) allows to establish an electrically conducting connection to the camera housing if required by your application.

The Basler application note called *Avoiding EMI and ESD in Basler Camera Installations* provides much more detail about avoiding EMI and ESD.
This application note can be obtained from the Downloads section of our website: www.baslerweb.com

1.8 Environmental Requirements

1.8.1 Temperature and Humidity

Housing temperature during operation: 0 °C ... +60 °C (+32 °F ... +140 °F)

Housing temperature according to UL 60950-1:	max. 70 °C (+158 °F)
Ambient temperature according to UL 60950-1:	max. 50 °C (+122 °F)

UL 60950-1 test conditions: no lens attached to the camera and without efficient heat dissipation; ambient temperature kept at 50 °C (+122 °F).

Humidity during operation:	20 % ... 80 %, relative, non-condensing
Storage temperature:	-20 °C ... +80 °C (-4 °F ... +176 °F)
Storage humidity:	20 % ... 80 %, relative, non-condensing

1.8.2 Heat Dissipation

You must provide sufficient heat dissipation to maintain the temperature of the camera housing at 60 °C or less. Since each installation is unique, Basler does not supply a strictly required technique for proper heat dissipation. Instead, we provide the following general guidelines:

- In all cases, you should monitor the temperature of the camera housing and make sure that the temperature does not exceed 60 °C. Keep in mind that the camera will gradually become warmer during the first hour of operation. After one hour, the housing temperature will have stabilized and will no longer increase.
- If your camera is mounted on a substantial metal component in your system, this may provide sufficient heat dissipation.
- The use of a fan to provide air flow over the camera is an extremely efficient method of heat dissipation. The use of a fan provides the best heat dissipation.

1.8.3 Imaging Sensor Over Temperature Condition

The camera has imaging sensor over temperature protection. If the temperature of the camera's imaging sensor rises above 75° C, an over temperature error condition will be reported (see also Section 10.15 on page 209) and the circuitry for the imaging sensor will switch off. In this situation, you will still be able to communicate with the camera but the camera will no longer acquire images.

Provide the necessary cooling when this situation arises. After the imaging sensor circuitry has sufficiently cooled bring the camera back to normal operation by either action:

- Carry out a camera restart by switching power off and on again or
- Carry out a camera reset as described in Section 12.1 on page 236.

1.9 Precautions

DANGER

Electric Shock Hazard

Non-approved power supplies may cause electric shock. Serious injury or death may occur.

You must use a camera power supply which meets the Safety Extra Low Voltage (SELV) and Limited Power Source (LPS) requirements.

WARNING

Fire Hazard

Non-approved power supplies may cause fire and burns.

You must use a camera power supply which meets the Limited Power Source (LPS) requirements.

NOTICE

Dust on the sensor can impair the camera's performance.

The camera is shipped with a protective plastic seal on the camera front. To avoid collecting dust on the camera's sensor, make sure that you always put the protective seal in place when there is no lens mounted on the camera.

Also, make sure to always point the camera downward when there is no protective seal on the camera front or no lens mounted.

NOTICE

Using a wrong pin assignment for the 12-pin receptacle can severely damage the camera.

Make sure the cable and plug you connect to the 12-pin receptacle follows the correct pin assignment. In particular, do **not** use a pin assignment that would be correct for Basler area scan cameras. The 12-pin receptacles of Basler line scan and area scan cameras are electrically incompatible.

NOTICE

Applying incorrect power can damage the camera.

You must supply camera power with the correct voltage: The camera‘s required operating voltage is +12 VDC (-10 %) to +24 VDC (+5 %), < 1 % ripple, effective on the camera‘s connector, with a nominal operating voltage of +12 VDC (± 10 %).

Applying power with the wrong polarity can severely damage the camera.

You must supply camera power with the correct polarity.

Insufficient camera power voltage can make the camera inoperable.

You must avoid a voltage drop. If you supply camera power via a long cable a voltage drop can occur. We recommend that you provide +12 VDC to +24 VDC separately through the wires connecting to pins 1 and 2 of the receptacle. We also recommend that you provide ground separately to the wires connecting to pins 5 and 6.

NOTICE

Incorrect plugs can damage the camera’s connectors.

The plug on the cable that you attach to the camera’s 6-pin connector must have 6 female pins. The plug on the cable that you attach to the camera’s 12-pin connector must have 12 female pins.

NOTICE

Inappropriate code may cause unexpected camera behavior.

- The code snippets provided in this manual are included as sample code only. Inappropriate code may cause your camera to function differently than expected and may compromise your application.
- To ensure that the snippets will work properly in your application, you must adjust them to meet your specific needs and must test them thoroughly prior to use.

Warranty Precautions

To ensure that your warranty remains in force:

Do not remove the camera's serial number label

If the label is removed and the serial number can't be read from the camera's registers, the warranty is void.

Do not open the camera housing

Do not open the housing. Touching internal components may damage them.

Keep foreign matter outside of the camera

Be careful not to allow liquid, flammable, or metallic material inside of the camera housing. If operated with any foreign matter inside, the camera may fail or cause a fire.

Avoid electromagnetic fields

Do not operate the camera in the vicinity of strong electromagnetic fields. Avoid electrostatic charging.

Transport properly

Transport the camera in its original packaging only. Do not discard the packaging.

Clean properly

Avoid cleaning the surface of the camera's sensor if possible. If you must clean it, use a soft, lint free cloth dampened with a small quantity of high quality window cleaner. Because electrostatic discharge can damage the sensor, you must use a cloth that will not generate static during cleaning (cotton is a good choice).

To clean the surface of the camera housing, use a soft, dry cloth. To remove severe stains, use a soft cloth dampened with a small quantity of neutral detergent, then wipe dry.

Do not use solvents or thinners to clean the housing; they can damage the surface finish.

Read the manual

Read the manual carefully before using the camera!

2 Software and Hardware Installation

The information you will need to install the camera is included in the *Installation and Setup Guide for Cameras Used with pylon for Windows* (AW000611).

You can download the document from the Basler website: www.baslerweb.com

The guide includes the information you will need to install both hardware and software and to begin capturing images. It also describes the recommended network adapters, describes the recommended architecture for the network to which your camera is attached, and deals with the IP configuration of your camera and network adapter.

After completing your camera installation, refer to Chapter 4 on page 28 and Chapter 5 on page 39 for information about improving your camera's performance in a network and about using multiple cameras.

3 Tools for Changing Camera Parameters

3.1 Basler pylon Camera Software Suite

The Basler pylon Camera Software Suite is available for Windows and Linux operating systems and is designed to operate all Basler cameras that have an IEEE 1394 interface, a GigE interface or a USB 3.0 interface. It will also operate some newer Basler camera models with a Camera Link interface. The pylon drivers offer reliable, real-time image data transport into the memory of your PC at a very low CPU load.

The options available with the Basler pylon Camera Software Suite let you

- change parameters and control the camera by using a standalone GUI known as the Basler pylon Viewer.
- change parameters and control the camera from within your software application using the Basler pylon SDKs.
- view and change the IP configuration of your GigE camera device by using the Basler pylon IP Configurator.

The remaining sections in this chapter provide an introduction to the tools.

You can obtain the Basler pylon Camera Software Suite from the Basler website by using this link: www.baslerweb.com

To help you install the software, you can also download the *Basler racer Installation and Setup Guide for Camera Link Cameras* (AW001186) from the Basler website.

3.1.1 pylon Viewer

The pylon Viewer is included in the Basler pylon Camera Software Suite. It is a standalone application that lets you view and change most of the camera's parameter settings via a GUI-based interface. Using the pylon Viewer is a very convenient way to get your camera up and running quickly during your initial camera evaluation or a camera design-in for a new project.

For more information about using the pylon Viewer, see the *Installation and Setup Guide for Cameras Used with Basler pylon for Windows* (AW000611).

3.1.2 pylon IP Configurator

The pylon IP Configurator is included in the Basler pylon Camera Software Suite. The pylon IP Configurator is a standalone application that lets you view and change the IP configuration of the camera via a GUI. The tool will detect all Basler GigE cameras attached to your network and let you make changes to a selected camera.

For more information about using the IP Configurator, see the *Installation and Setup Guide for Cameras Used with pylon for Windows* (AW000611).

3.1.3 pylon SDKs

Three pylon SDKs are part of the Basler pylon Camera Software Suite:

- pylon SDK for C++ (Windows and Linux)
- pylon SDK for C (Windows)
- pylon SDK for .NET / C# (Windows)

Each SDK includes an API, a set of sample programs, and documentation:

- You can access all of the camera's parameters and control the camera's full functionality from within your application software by using the matching pylon API (C++, C, or .NET).
- The sample programs illustrate how to use the pylon API to parameterize and operate the camera.
- For each environment (C++, C, and .NET), a *Programmer's Guide and Reference Documentation* is available. The documentation gives an introduction to the pylon API and provides information about all methods and objects of the API.

4 Basler Network Drivers and Parameters

This section describes the Basler network drivers available for your camera and provides detailed information about the parameters associated with the drivers.

Two network drivers are available for the network adapter used with your GigE cameras:

- The **Basler filter driver** is a basic GigE Vision network driver that is compatible with all network adapters. The advantage of this driver is its extensive compatibility.
- The **Basler performance driver** is a hardware specific GigE Vision network driver. The driver is only compatible with network adapters that use specific Intel chipsets. The advantage of the performance driver is that it significantly lowers the CPU load needed to service the network traffic between the PC and the camera(s). It also has a more robust packet resend mechanism.

During the installation process you should have installed either the filter driver or the performance driver.

For more information about compatible Intel chipsets and about installing the network drivers, see the *Installation and Setup Guide for Cameras Used with pylon for Windows* (AW000611).

4.1 The Basler Filter Driver

The Basler filter driver is a basic driver GigE Vision network driver. It is designed to be compatible with most network adapter cards.

The functionality of the filter driver is relatively simple. For each frame, the driver checks the order of the incoming packets. If the driver detects that a packet or a group of packets is missing, it will wait for a specified period of time to see if the missing packet or group of packets arrives. If the packet or group does not arrive within the specified period, the driver will send a resend request for the missing packet or group of packets.

The parameters associated with the filter driver are described below.

EnableResend - Enables or disables the packet resend mechanism.

If packet resend is disabled and the filter driver detects that a packet has been lost during transmission, the grab result for the returned buffer holding the image will indicate that the grab failed and the image will be incomplete.

If packet resend is enabled and the driver detects that a packet has been lost during transmission, the driver will send a resend request to the camera. If the camera still has the packet in its buffer, it will resend the packet. If there are several lost packets in a row, the resend requests will be combined.

PacketTimeout - The PacketTimeout parameter defines how long (in milliseconds) the filter driver will wait for the next expected packet before it initiates a resend request. Ensure the Packet Timeout parameter is set to a longer time interval than the time interval set for the inter-packet delay.

FrameRetention - The FrameRetention parameter sets the timeout (in milliseconds) for the frame retention timer. Whenever the filter driver detects the leader for a frame, the frame retention timer starts. The timer resets after each packet in the frame is received and will timeout after the last packet is received. If the timer times out at any time before the last packet is received, the buffer for the frame will be released and will be indicated as an unsuccessful grab.

You can set the filter driver parameter values from within your application software by using the Basler pylon API. The following code snippet illustrates using the API to read and write the parameter values:

```
// Enable Resend
Camera_t::StreamGrabber_t StreamGrabber(camera.GetStreamGrabber(0));
StreamGrabber.EnableResend.SetValue(false); // disable resends

// Packet Timeout / Frame Retention
Camera_t::StreamGrabber_t StreamGrabber(camera.GetStreamGrabber(0));
StreamGrabber.PacketTimeout.SetValue(40);
StreamGrabber.FrameRetention.SetValue(200);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer’s Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

4.2 The Basler Performance Driver

The Basler performance driver is a hardware specific GigE Vision network driver compatible with network adapters that use specific Intel chipsets. The main advantage of the performance driver is that it significantly lowers the CPU load needed to service the network traffic between the PC and the camera(s). It also has a more robust packet resend mechanism.

For more information about compatible Intel chipsets, see the installation and Setup Guide for Cameras Used with Basler's pylon API.

The performance driver uses two distinct "resend mechanisms" to trigger resend requests for missing packets:

- The threshold resend mechanism
- The timeout resend mechanism

The mechanisms are independent from each other and can be used separately. However, for maximum efficiency and for ensuring that resend requests will be sent for all missing packets, we recommend using both resend mechanisms in a specific, optimized combination, as provided by the parameter default values.

The performance driver's parameter values determine how the resend mechanisms act and how they relate to each other. You can set the parameter values by using the pylon Viewer or from within your application software by using the pylon API.

The parameter default values will provide for the following:

- The threshold resend mechanism precedes the timeout resend mechanism. This ensures that a resend request is sent for every missing packet, even at very high rates of arriving packets.
- The timeout resend mechanism will be effective for those missing packets that were not resent after the first resend request.

We strongly recommend using the default parameter settings. Only users with the necessary expertise should change the default parameter values.

The Basler performance driver uses a "receive window" to check the status of packets. The check for missing packets is made as packets enter the receive window. If a packet arrives from higher in the sequence of packets than expected, the preceding skipped packet or packets are detected as missing. For example, suppose packet (n-1) has entered the receive window and is immediately followed by packet (n+1). In this case, as soon as packet (n+1) enters the receive window, packet n will be detected as missing.

General Parameters

EnableResend - Enables the packet resend mechanisms.

If the EnableResend parameter is set to False, the resend mechanisms are disabled. The performance driver will not check for missing packets and will not send resend requests to the camera.

If the EnableResend parameter is set to True, the resend mechanisms are enabled. The performance driver will check for missing packets. Depending on the parameter settings and the resend response, the driver will send one or several resend requests to the camera.

ReceiveWindowSize - Sets the size of the receive window.

Threshold Resend Mechanism Parameters

The threshold resend request mechanism is illustrated in Fig. 10 where the following assumptions are made:

- Packets 997, 998, and 999 are missing from the stream of packets.
- Packet 1002 is missing from the stream of packets.

DIAGRAM IS NOT DRAWN TO SCALE

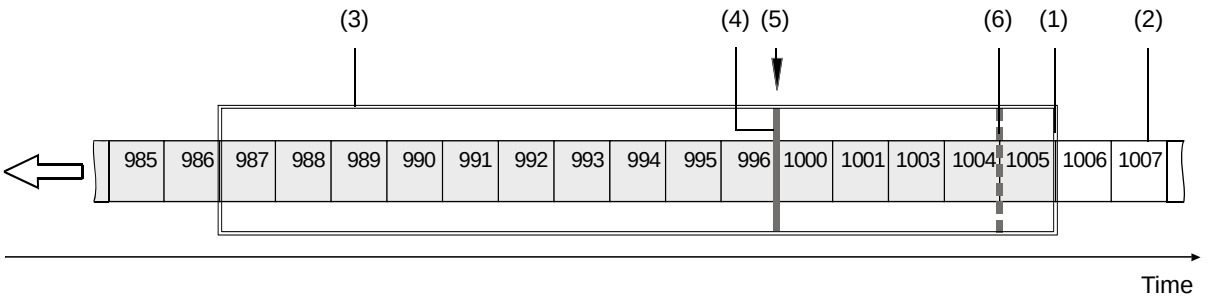


Fig. 10: Example of a Receive Window with Resend Request Threshold & Resend Request Batching Threshold

(1) Front end of the receive window. Missing packets are detected here.

(2) Stream of packets. Gray indicates that the status was checked as the packet entered the receive window. White indicates that the status has not yet been checked.

(3) Receive window of the performance driver.

(4) Threshold for sending resend requests (resend request threshold).

(5) A separate resend request is sent for each packets 997, 998, and 999.

(6) Threshold for batching resend requests for consecutive missing packets (resend request batching threshold). Only one resend request will be sent for the consecutive missing packets.

ResendRequestThreshold - This parameter determines the location of the resend request threshold within the receive window as shown in Fig. 10. The parameter value is in per cent of the width of the receive window. In Fig. 10 the resend request threshold is set at 33.33% of the width of the receive window.

A stream of packets advances packet by packet beyond the resend request threshold (i.e. to the left of the resend request threshold in Fig. 10). As soon as the position where a packet is missing advances beyond the resend request threshold, a resend request is sent for the missing packet.

In the example shown in Fig. 10, packets 987 to 1005 are within the receive window and packets 997 to 999 and 1002 were detected as missing. In the situation shown, a resend request is sent to the camera for each of the missing consecutive packets 997 to 999. The resend requests are sent after packet 996 - the last packet of the intact sequence of packets - has advanced beyond the resend request threshold and before packet 1000 - the next packet in the stream of packets - can advance beyond the resend request threshold. Similarly, a resend request will be sent for missing packet 1002 after packet 1001 has advanced beyond the resend request threshold and before packet 1003 can advance beyond the resend request threshold.

ResendRequestBatching - This parameter determines the location of the resend request batching threshold in the receive window (Fig. 10). The parameter value is in per cent of a span that starts with the resend request threshold and ends with the front end of the receive window. The maximum allowed parameter value is 100. In Fig. 10 the resend request batching threshold is set at 80% of the span.

The resend request batching threshold relates to consecutive missing packets, i.e., to a continuous sequence of missing packets. Resend request batching allows grouping of consecutive missing packets for a single resend request rather than sending a sequence of resend requests where each resend request relates to just one missing packet.

The location of the resend request batching threshold determines the maximum number of consecutive missing packets that can be grouped together for a single resend request. The maximum number corresponds to the number of packets that fit into the span between the resend request threshold and the resend request batching threshold plus one.

If the Resend Request Batching parameter is set to 0, no batching will occur and a resend request will be sent for each single missing packet. For other settings, consider an example: Suppose the Resend Request Batching parameter is set to 80 referring to a span between the resend request threshold and the front end of the receive window that can hold five packets (Fig. 10). In this case 4 packets (5 x 80%) will fit into the span between the resend request threshold and the resend request batching threshold. Accordingly, the maximum number of consecutive missing packets that can be batched is 5 (4 + 1).

Timeout Resend Mechanism Parameters

The timeout resend mechanism is illustrated in Fig. 11 where the following assumptions are made:

- The frame includes 3000 packets.
- Packet 1002 is missing within the stream of packets and has not been recovered.
- Packets 2999 and 3000 are missing at the end of the stream of packets (end of the frame).
- The MaximumNumberResendRequests parameter is set to 3.

DIAGRAM IS NOT DRAWN TO SCALE

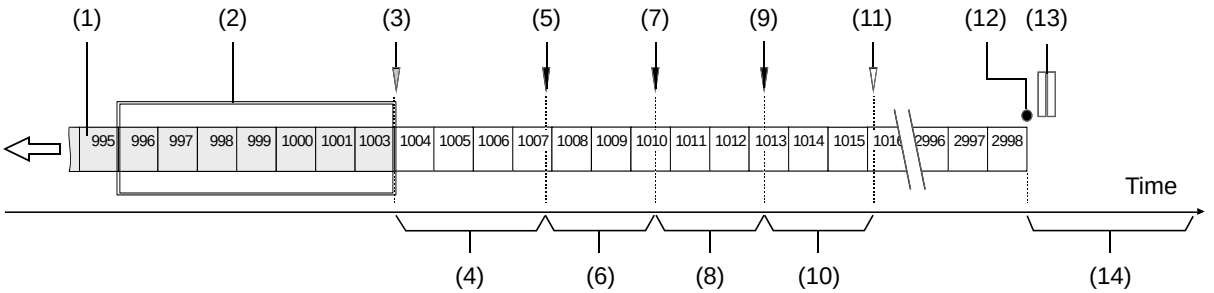


Fig. 11: Incomplete Stream of Packets and Part of the Resend Mechanism

(1) Stream of packets. Gray indicates that the status was checked as the packet entered the receive window. White indicates that the status has not yet been checked.

(2) Receive window of the performance driver.

(3) As packet 1003 enters the receive window, packet 1002 is detected as missing.

(4) Interval defined by the ResendTimeout parameter.

(5) The Resend Timeout interval expires and the first resend request for packet 1002 is sent to the camera. The camera does not respond with a resend.

(6) Interval defined by the ResendResponseTimeout parameter.

(7) The Resend Response Timeout interval expires and a second resend request for packet 1002 is sent to the camera. The camera does not respond with a resend.

(8) Interval defined by the ResendResponseTimeout parameter.

(9) The Resend Response Timeout interval expires and a third resend request for packet 1002 is sent to the camera. The camera still does not respond with a resend.

(10) Interval defined by the ResendResponseTimeout parameter.

(11) Because the maximum number of resend requests has been sent and the last Resend Response Timeout interval has expired, packet 1002 is now considered as lost.

(12) End of the frame.

(13) Missing packets at the end of the frame (2999 and 3000).

(14) Interval defined by the PacketTimeout parameter.

MaximumNumberResendRequests - The MaximumNumberResendRequests parameter sets the maximum number of resend requests the performance driver will send to the camera for each missing packet.

ResendTimeout - The ResendTimeout parameter defines how long (in milliseconds) the performance driver will wait after detecting that a packet is missing before sending a resend request to the camera. The parameter applies only once to each missing packet after the packet was detected as missing.

ResendRequestResponseTimeout - The ResendRequestResponseTimeout parameter defines how long (in milliseconds) the performance driver will wait after sending a resend request to the camera before considering the resend request as lost.

If a resend request for a missing packet is considered lost and if the maximum number of resend requests as set by the MaximumNumberResendRequests parameter has not yet been reached, another resend request will be sent. In this case, the parameter defines the time separation between consecutive resend requests for a missing packet.

PacketTimeout - The PacketTimeout parameter defines how long (in milliseconds) the performance driver will wait for the next expected packet before it sends a resend request to the camera. This parameter ensures that resend requests are sent for missing packets near to the end of a frame. In the event of a major interruption in the stream of packets, the parameter will also ensure that resend requests are sent for missing packets that were detected to be missing immediately before the interruption. Make sure the PacketTimeout parameter is set to a longer time interval than the time interval set for the inter-packet delay.

Threshold and Timeout Resend Mechanisms Combined

Fig. 12 illustrates the combined action of the threshold and the timeout resend mechanisms where the following assumptions are made:

- All parameters set to default.
- The frame includes 3000 packets.
- Packet 1002 is missing within the stream of packets and has not been recovered.
- Packets 2999 and 3000 are missing at the end of the stream of packets (end of the frame).

The default values for the performance driver parameters will cause the threshold resend mechanism to become operative before the timeout resend mechanism. This ensures maximum efficiency and that resend requests will be sent for all missing packets.

With the default parameter values, the resend request threshold is located very close to the front end of the receive window. Accordingly, there will be only a minimum delay between detecting a missing packet and sending a resend request for it. In this case, a delay according to the ResendTimeout parameter will not occur (see Fig. 12). In addition, resend request batching will not occur.

DIAGRAM IS NOT DRAWN TO SCALE

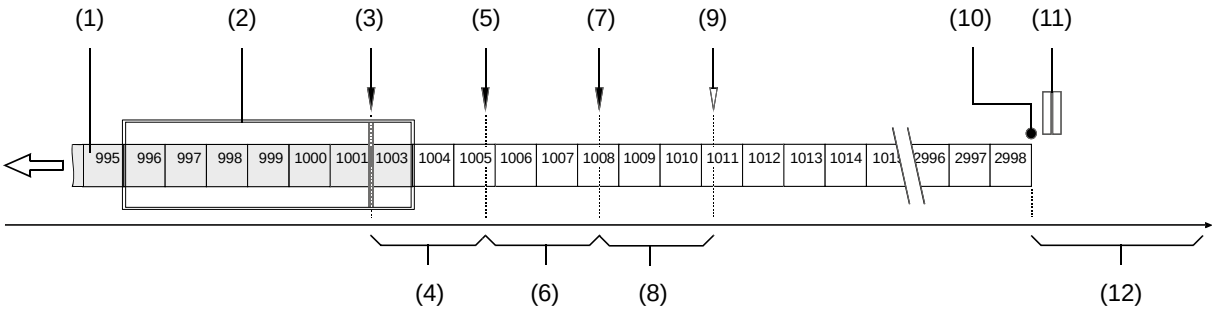


Fig. 12: Combination of Threshold Resend Mechanism and Timeout Resend Mechanism

(1) Stream of packets, Gray indicates that the status was checked as the packet entered the receive window. White indicates that the status has not yet been checked.

(2) Receive window of the performance driver.

(3) Threshold for sending resend requests (resend request threshold). The first resend request for packet 1002 is sent to the camera. The camera does not respond with a resend.

(4) Interval defined by the ResendResponseTimeout parameter.

(5) The Resend Timeout interval expires and the second resend request for packet 1002 is sent to the camera. The camera does not respond with a resend.

(6) Interval defined by the ResendResponseTimeout parameter

(7) The Resend Timeout interval expires and the third resend request for packet 1002 is sent to the camera. The camera does not respond with a resend.

(8) Interval defined by the ResendResponseTimeout parameter

(9) Because the maximum number of resend requests has been sent and the last Resend Response Timeout interval has expired, packet 1002 is now considered as lost.

(10) End of the frame.

(11) Missing packets at the end of the frame (2999 and 3000).

(12) Interval defined by the PacketTimeout parameter.

You can set the performance driver parameter values from within your application software by using the Basler pylon API. The following code snippet illustrates using the API to read and write the parameter values:

```
// Get the Stream Parameters object
Camera_t::StreamGrabber_t StreamGrabber(camera.GetStreamGrabber(0));

// Write the ReceiveWindowSize parameter
StreamGrabber.ReceiveWindowSize.SetValue(16);

// Disable packet resends
StreamGrabber.EnableResend.SetValue(false);

// Write the PacketTimeout parameter
StreamGrabber.PacketTimeout.SetValue(40);

// Write the ResendRequestThreshold parameter
StreamGrabber.ResendRequestThreshold.SetValue(5);

// Write the ResendRequestBatching parameter
StreamGrabber.ResendRequestBatching.SetValue(10);

// Write the ResendTimeout parameter
StreamGrabber.ResendTimeout.SetValue(2);

// Write the ResendRequestResponseTimeout parameter
StreamGrabber.ResendRequestResponseTimeout.SetValue(2);

// Write the MaximumNumberResendRequests parameter
StreamGrabber.MaximumNumberResendRequests.SetValue(25);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer’s Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.
(The performance driver parameters will only appear in the viewer if the performance driver is installed on the adapter to which your camera is connected.)

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

Adapter Properties

When the Basler Performance driver is installed, it adds a set of "advanced" properties to the network adapter. These properties include:

Max Packet Latency - A value in microseconds that defines how long the adapter will wait after it receives a packet before it generates a packet received interrupt.

Max Receive Inter-Packet Delay - A value in microseconds that defines the maximum amount of time allowed between incoming packets.

Maximum Interrupts per Second - Sets the maximum number of interrupts per second that the adapter will generate.

Network Address - allows the user to specify a MAC address that will override the default address provided by the adapter.

Packet Buffer Size - Sets the size in bytes of the buffers used by the receive descriptors and the transmit descriptors.

Receive Descriptors - Sets the number of descriptors to use in the adapter's receiving ring.

Transmit Descriptors - Sets the number of descriptors to use in the adapter's transmit ring.

To access the advanced properties for an adapter:

1. Open a **Network Connections** window and find the connection for your network adapter.
2. Right click on the name of the connection and select **Properties** from the drop down menu.
3. A **LAN Connection Properties** window will open. Click the **Configure** button.
4. An **Adapter Properties** window will open. Click the **Advanced** tab.

We strongly recommend using the default parameter settings. Changing the parameters can have a significant negative effect on the performance of the adapter and the driver.

4.3 Transport Layer Parameters

The transport layer parameters are part of the camera's basic GigE implementation. These parameters do not normally require adjustment.

ReadTimeout - If a register read request is sent to the camera via the transport layer, this parameter designates the time out (in milliseconds) within which a response must be received.

WriteTimeout - If a register write request is sent to the camera via the transport layer, this parameter designates the time out (in milliseconds) within which an acknowledge must be received.

HeartbeatTimeout - The GigE Vision standard requires implementation of a heartbeat routine to monitor the connection between the camera and the host PC. This parameter sets the heartbeat timeout (in milliseconds). If a timeout occurs, the camera releases the network connection and enters a state that allows reconnection.

Management of the heartbeat time is normally handled by Basler's basic GigE implementation. Changing this parameter is not required for normal camera operation. However, if you are debugging an application and you stop at a break point, you will have a problem with the heartbeat timer. The timer will time out when you stop at a break point and the connection to the camera will be lost. When debugging, you should increase the heartbeat timeout to a high value to avoid heartbeat timeouts at break points. When debugging is complete, you should return the timeout to its normal setting.

You can set the driver related transport layer parameter values from within your application software by using the Basler pylon API. The following code snippet illustrates using the API to read and write the parameter values:

```
 // Read/Write Timeout
Camera_t::TlParams_t TlParams(camera.GetTLNodeMap());
TlParams.ReadTimeout.SetValue(500);  // 500 milliseconds
TlParams.WriteTimeout.SetValue(500); // 500 milliseconds

// Heartbeat Timeout
Camera_t::TlParams_t TlParams(camera.GetTLNodeMap()(;
TlParams.HeartbeatTimeout.SetValue(5000);  // 5 seconds
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

5 Network Related Camera Parameters and Managing Bandwidth

This section describes the camera parameters that are related to the camera's performance on the network. It also describes how to use the parameters to manage the available network bandwidth when you are using multiple cameras.

5.1 Network Related Parameters in the Camera

The camera includes several parameters that determine how it will use its network connection to transmit data to the host PC. The list below describes each parameter and provides basic information about how the parameter is used. The following section describes how you can use the parameters to manage the bandwidth used by each camera on your network.

Payload Size (PayloadSize, read only)

Indicates the total size in bytes of the image data plus any chunk data (if chunks are enabled) in each frame that the camera will transmit. Packet headers are not included.

Packet Size (GevSCPSPacketSize, read/write)

As specified in the GigE Vision standard, each acquired frame will be fit into a data block. The block contains three elements: a *data leader* consisting of one packet used to signal the beginning of a data block, the *data payload* consisting of one or more packets containing the actual data for the current block, and a *data trailer* consisting of one packet used to signal the end of the data block.

The GevSCPSPacketSize parameter sets the size of the packets that the camera will use when it sends the data payload via the selected stream channel. The value is in bytes. The value does not affect the leader and trailer size using a total of 36 bytes, and the last data packet may be a smaller size. The payload size will be the packet size minus 36 bytes.

The GevSCPSPacketSize parameter should always be set to the maximum size that your network adapter and network switches (if used) can handle.

Inter-Packet Delay (GevSCPD, read/write)

Sets the delay in ticks between the packets sent by the camera. Applies to the selected stream channel. Increasing the inter-packet delay will decrease the camera's effective data transmission rate and will thus decrease the network bandwidth used by the camera.

In the current camera implementation, one tick = 8 ns. To check the tick frequency, you can read the GevTimestampTickFrequency parameter value. This value indicates the number of clock ticks per second.

When setting the time interval for the inter-packet delay, make sure that the time interval for the packet timeout is set to a higher value.

Frame Transmission Delay (GevSCFTD, read/write)

Sets a delay in ticks (one tick = 8 ns) between when a camera would normally begin transmitting an acquired frame and when it actually begins transmission. This parameter should be set to zero in most normal situations.

If you have many cameras in your network and you will be simultaneously triggering image acquisition on all of them, you may find that your network switch or network adapter is overwhelmed if all of the cameras simultaneously begin to transmit frame data at once. The GevSCFTD parameter can be used to stagger the start of frame data transmission from each camera.

Bandwidth Assigned (GevSCBWA, read only)

Indicates the bandwidth in bytes per second that will be used by the camera to transmit frame and chunk feature data and to handle resends and control data transmissions. The value of this parameter is a result of the packet size and the inter-packet delay parameter settings.

In essence, the bandwidth assigned is calculated this way:

$$\text{Bandwidth Assigned} = \frac{\frac{\text{X Packets}}{\text{Frame}} \times \frac{\text{Y Bytes}}{\text{Packet}}}{\left[\frac{\text{X Packets}}{\text{Frame}} \times \frac{\text{Y Bytes}}{\text{Packet}} \times \frac{8\ \text{ns}}{\text{Byte}}\right] + \left[\left(\frac{\text{X Packets}}{\text{Frame}} - 1\right) \times (\text{IPD} \times 8\ \text{ns})\right]}$$

Where: X = number of packets needed to transmit the frame

Y = number of bytes in each packet

IPD = Inter-packet delay setting in ticks (with a tick set to the 8 ns standard)

When considering this formula, you should know that on a Gigabit network it takes one tick to transmit one byte. Also, be aware that the formula has been simplified for easier understanding.

5.2 Managing Bandwidth When Multiple Cameras Share a Single Network Path

If you are using a single camera on a GigE network, the problem of managing bandwidth is simple. The network can easily handle the bandwidth needs of a single camera and no intervention is required. A more complicated situation arises if you have multiple cameras connected to a single network adapter as shown in Fig. 13.

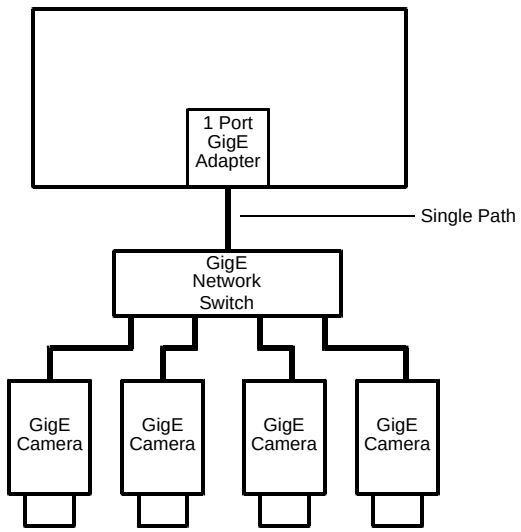


Fig. 13: Multiple Cameras on a Network

One way to manage the situation where multiple cameras are sharing a single network path is to make sure that only one of the cameras is acquiring and transmitting frames at any given time. The data output from a single camera is well within the bandwidth capacity of the single path and you should have no problem with bandwidth in this case.

If you want to acquire and transmit frames from several cameras simultaneously, however, you must determine the total data output rate for all the cameras that will be operating simultaneously and you must make sure that this total does not exceed the bandwidth of the single path (125 MByte/s).

An easy way to make a quick check of the total data output from the cameras that will operate simultaneously is to read the value of the Bandwidth Assigned (GevSCBWA) parameter for each camera. This parameter indicates the camera's gross data output rate in bytes per second with its current settings. If the sum of the bandwidth assigned values is less than 125 MByte/s, the cameras should be able to operate simultaneously without problems. If it is greater, you must lower the data output rate of one or more of the cameras.

You can lower the data output rate on a camera by using the Inter-Packet Delay (GevSCPD) parameter. This parameter adds a delay between the transmission of each packet from the camera and thus slows the data transmission rate of the camera. The higher the inter-packet delay parameter is set, the greater the delay between the transmission of each packet will be and the

lower the data transmission rate will be. After you have adjusted the Inter-Packet Delay (GevSCPD) parameter on each camera, you can check the sum of the Bandwidth Assigned parameter values and see if the sum is now less than 125 MByte/s.

5.2.1 A Procedure for Managing Bandwidth

In theory, managing bandwidth sharing among several cameras is as easy as adjusting the inter-packet delay. In practice, it is a bit more complicated because you must consider several factors when managing bandwidth. The procedure below outlines a structured approach to managing bandwidth for several cameras.

The objectives of the procedure are:

- To optimize network performance.
- To determine the bandwidth needed by each camera for frame data transmission.
- To determine the bandwidth actually assigned to each camera for frame data transmission.
- For each camera, to make sure that the actual bandwidth assigned for frame data transmission matches the bandwidth needed.
- To make sure that the total bandwidth assigned to all cameras does not exceed the network's bandwidth capacity.
- To make adjustments if the bandwidth capacity is exceeded.

Step 1 - Optimize the Network Performance.

If, as recommended, you are using the Basler performance driver with an Intel PRO network adapter or a compatible network adapter, the network parameters for the network adapter are automatically optimized and need not be changed. Go on to step two now.

If you are using the Basler filter driver and you have already set the network parameters for your adapter during the installation of the Basler pylon software, go on to step two now.

Otherwise, open the **Network Connection Properties** window for your network adapter and check the following network parameters:

- If you are using an Intel PRO network adapter, make sure the **Receive Descriptors** parameter is set to its maximum value and the **Interrupt Moderation Rate** parameter is set to **Extreme**.

  Also make sure the **Speed and Duplex Mode** parameter is set to **Auto Detect**.
- If you are using a different network adapter, see whether parameters are available that will allow you to set the number of receive descriptors and the number of CPU interrupts. The related parameter names may differ from the ones used for the Intel PRO adapters. Also, the way of setting the parameters may be different. You may, for example, need to use a parameter to set a low number for the interrupt moderation and then use a different parameter to enable the interrupt moderation.

  If possible, set the number of receive descriptors to a maximum value and set the number of CPU interrupts to a low value.

  If possible, also set the parameter for speed and duplex to auto detect.

Contact Basler technical support if you need further assistance.

Step 2 - Set the Packet Size (GevSCPSPacketSize) parameter on each camera as large as possible.

Using the largest possible packet size has two advantages, it increases the efficiency of network transmissions between the camera and the PC and it reduces the time required by the PC to process incoming packets. The largest packet size setting that you can use with your camera is determined by the largest packet size that can be handled by your network. The size of the packets that can be handled by the network depends on the capabilities and settings of the network adapter you are using and on capabilities of the network switch you are using.

Start by checking the documentation for your adapter to determine the maximum packet size (sometimes called “frame” size) that the adapter can handle. Many adapters can handle what is known as “jumbo packets” or "jumbo frames". These are packets with a 16 kB size. Once you have determined the maximum size packets the adapter can handle, make sure that the adapter is set to use the maximum packet size.

Next, check the documentation for your network switch and determine the maximum packet size that it can handle. If there are any settings available for the switch, make sure that the switch is set for the largest packet size possible.

Now that you have set the adapter and switch, you can determine the largest packet size the network can handle. The device with the smallest maximum packet size determines the maximum allowed packet size for the network. For example, if the adapter can handle 16 kB packets and the switch can handle 8 kB packets, then the maximum for the network is 8 kB packets.

Once you have determined the maximum packet size for your network, set the value of the Packet Size (GevSCPSPacketSize) parameter on each camera to this value.

The manufacturer’s documentation sometimes makes it difficult to determine the maximum packet size for a device, especially network switches. There is a "quick and dirty" way to check the maximum packet size for your network with its current configuration:

1. Open the pylon Viewer, select a camera, and set the Packet Size parameter (GevSCPSPacketSize) to a low value (1 kB for example).
2. Use the Continuous Shot mode to acquire several frames.
3. Gradually increase the value of the Packet Size (GevSCPSPacketSize) parameter and acquire a few frames after each size change.
4. When your Packet Size setting exceeds the packet size that the network can handle, the pylon Viewer will lose the ability to acquire frames. (When you use Continuous Shot, the Viewer’s status bar will indicate that it is acquiring frames, but the frame in the viewing area will appear to be frozen.)

Step 3 - Set the Bandwidth Reserve (GevSCBWR) parameter for each camera.

The Bandwidth Reserve (GevSCBWR) parameter setting for a camera determines how much of the bandwidth assigned to that camera will be reserved for lost packet resends and for asynchronous traffic such as commands sent to the camera. If you are operating the camera in a relatively EMI free environment, you may find that a bandwidth reserve of 2% or 3% is adequate. If you are operating in an extremely noisy environment, you may find that a reserve of 8% or 10% is more appropriate.

Step 4 - Calculate the "data bandwidth needed" by each camera.

The objective of this step is to determine how much bandwidth (in Byte/s) each camera needs to transmit the frame data that it generates. The amount of data bandwidth a camera needs is the product of several factors: the amount of image data included in each frame, the amount of chunk data being added to each frame, the "packet overhead" such as packet leaders and trailers, and the number of frames the camera is acquiring each second.

For each camera, you can use the two formulas below to calculate the data bandwidth needed. To use the formulas, you will need to know the current value of the PayloadSize parameter and the Packet Size (GevSCPSPacketSize) parameter for each camera. You will also need to know the frame rate (in frames/s) at which each camera will operate.

$$\text{Bytes/Frame} = \left\lfloor \left\lceil \frac{\text{Payload Size}}{\text{Packet Size}} \right\rceil_1 \times \text{Packet Overhead} \right\rfloor + \lceil \text{Payload Size} \rceil_4 + \text{Leader Size} + \text{Trailer Size}$$

$$\text{Data Bandwidth Needed} = \text{Bytes/Frame} \times \text{Frames/s}$$

Where:

Packet Overhead = 72 (for a GigE network)

78 (for a 100 MBit/s network)

Leader Size = Packet Overhead + 36 (if chunk mode is not active)

Packet Overhead + 12 (if chunk mode is active)

Trailer Size = Packet Overhead + 8

$\lceil x \rceil_1$ means round up x to the nearest integer

$\lceil x \rceil_4$ means round up x to the nearest multiple of 4

Step 5 - Calculate “data bandwidth assigned” to each camera.

For each camera, there is a parameter called Bandwidth Assigned (GevSCBWA). This read only parameter indicates the total bandwidth that has been assigned to the camera. The Bandwidth Assigned (GevSCBWA) parameter includes both the bandwidth that can be used for frame data transmission plus the bandwidth that is reserved for packet resends and camera control signals. To determine the “data bandwidth assigned,” you must subtract out the reserve.

You can use the formula below to determine the actual amount of assigned bandwidth that is available for data transmission. To use the formula, you will need to know the current value of the Bandwidth Assigned (GevSCBWA) parameter and the Bandwidth Reserve (GevSCBWR) parameter for each camera.

$$\text{Data Bandwidth Assigned} = \text{Bandwidth Assigned} \times \frac{100 - \text{Bandwidth Reserved}}{100}$$

Step 6 - For each camera, compare the data bandwidth needed with the data bandwidth assigned.

For each camera, you should now compare the data bandwidth assigned to the camera (as determined in step 4) with the bandwidth needed by the camera (as determined in step 3).

For bandwidth to be used most efficiently, the data bandwidth assigned to a camera should be equal to or just slightly greater than the data bandwidth needed by the camera. If you find that this is the situation for all of the cameras on the network, you can go on to step 6 now. If you find a camera that has much more data bandwidth assigned than it needs, you should make an adjustment.

To lower the amount of data bandwidth assigned, you must adjust a parameter called the Inter-Packet Delay (GevSCPD). If you increase the Inter-Packet Delay (GevSCPD) parameter value on a camera, the data bandwidth assigned to the camera will decrease. So for any camera where you find that the data bandwidth assigned is much greater then the data bandwidth needed, you should do this:

1. Raise the setting for the Inter-Packet Delay (GevSCPD) parameter for the camera.
2. Recalculate the data bandwidth assigned to the camera.
3. Compare the new data bandwidth assigned to the data bandwidth needed.
4. Repeat 1, 2, and 3 until the data bandwidth assigned is equal to or just greater than the data bandwidth needed.

If you increase the inter-packet delay to lower a camera’s data output rate there is something that you must keep in mind. When you lower the data output rate, you increase the amount of time that the camera needs to transmit an acquired frame. Increasing the frame transmission time can restrict the camera’s maximum allowed acquisition line rate.

Step 7 - Check that the total bandwidth assigned is less than the network capacity.

1. For each camera, determine the current value of the Bandwidth Assigned (GevSCBWA) parameter. The value is in Byte/s. (Make sure that you determine the value of the parameter after you have made any adjustments described in the earlier steps.)
2. Find the sum of the current Bandwidth Assigned (GevSCBWA) parameter values for all of the cameras.

If the sum of the Bandwidth Assigned values is less than 125 MByte/s for a GigE network or 12.5 M/Byte/s for a 100 Bit/s network, the bandwidth management is OK.

If the sum of the Bandwidth Assigned values is greater than 125 MByte/s for a GigE network or 12.5 M/Byte/s for a 100 Bit/s network, the cameras need more bandwidth than is available and you must make adjustments. In essence, you must lower the data bandwidth needed by one or more of the cameras and then adjust the data bandwidths assigned so that they reflect the lower bandwidth needs.

You can lower the data bandwidth needed by a camera either by lowering its line rate or by decreasing the size of the frame. Once you have adjusted the line rates and/or frame size on the cameras, you should repeat steps 2 through 6.

For more information about the camera's maximum allowed line rate, see Section 8.5 on page 131.

For more information about the frame size, see Section 8.1 on page 74.

6 Camera Functional Description

This chapter provides an overview of the camera's functionality from a system perspective. The overview will aid your understanding when you read the more detailed information included in the later chapters of the user's manual.

Each camera employs a single line CMOS sensor chip designed for monochrome imaging. For 2k cameras, the sensor includes 2048 pixels with a pixel size of 7 µm x 7 µm. For 4k/6k cameras, the sensor consists of two/three 2k sensor segments with a pixel size of 7 µm x 7 µm, resulting in a total of 4096/6144 pixels. For 8k/12k cameras, the sensor consists of 2/3 4k sensor segments with a pixel size of 3.5 µm x 3.5 µm, resulting in a total of 8192/12288 pixels. See Fig. 14 on page 48 for an overview of the different sensor architectures.

Acquisition start, frame start, and line start can be controlled via externally generated hardware trigger signals. These signals facilitate periodic or non-periodic frame/line start. Modes are available that allow the length of exposure time to be directly controlled by the external line start signal or to be set for a pre-programmed period of time.

Acquisition start, frame start, and exposure time can also be controlled by parameters transmitted to the camera via the Basler pylon API and the GigE interface.

Accumulated charges are read out of the sensor when exposure ends. At readout, accumulated charges are moved from the sensor's light-sensitive elements (pixels) into the analog processing section of the sensor (Fig. 14 on page 48). As the charges move from the pixels to the analog processing section, they are converted to voltages proportional to the size of each charge. The voltages from the analog processing section are next passed to a bank of 12 Bit Analog-to-Digital converters (ADCs).

Finally, the gray values pass through a section of the sensor where they receive additional digital processing and then they are moved out of the sensor. As each gray value leaves the sensor, it passes through an FPGA and into an image buffer (Fig. 15 on page 49). All shifting is clocked according to the camera's internal data rate. Shifting continues until all image data has been read out of the sensor.

The gray values leave the image buffer and pass back through the FPGA to an Ethernet controller where they are assembled into data packets. The packets are then transmitted via an Ethernet network to a network adapter in the host PC. The Ethernet controller also handles transmission and receipt of control data such as changes to the camera's parameters.

The image buffer between the sensor and the Ethernet controller allows data to be read out of the sensor at a rate that is independent of the data transmission rate between the camera and the host computer. This ensures that the data transmission rate has no influence on image quality.

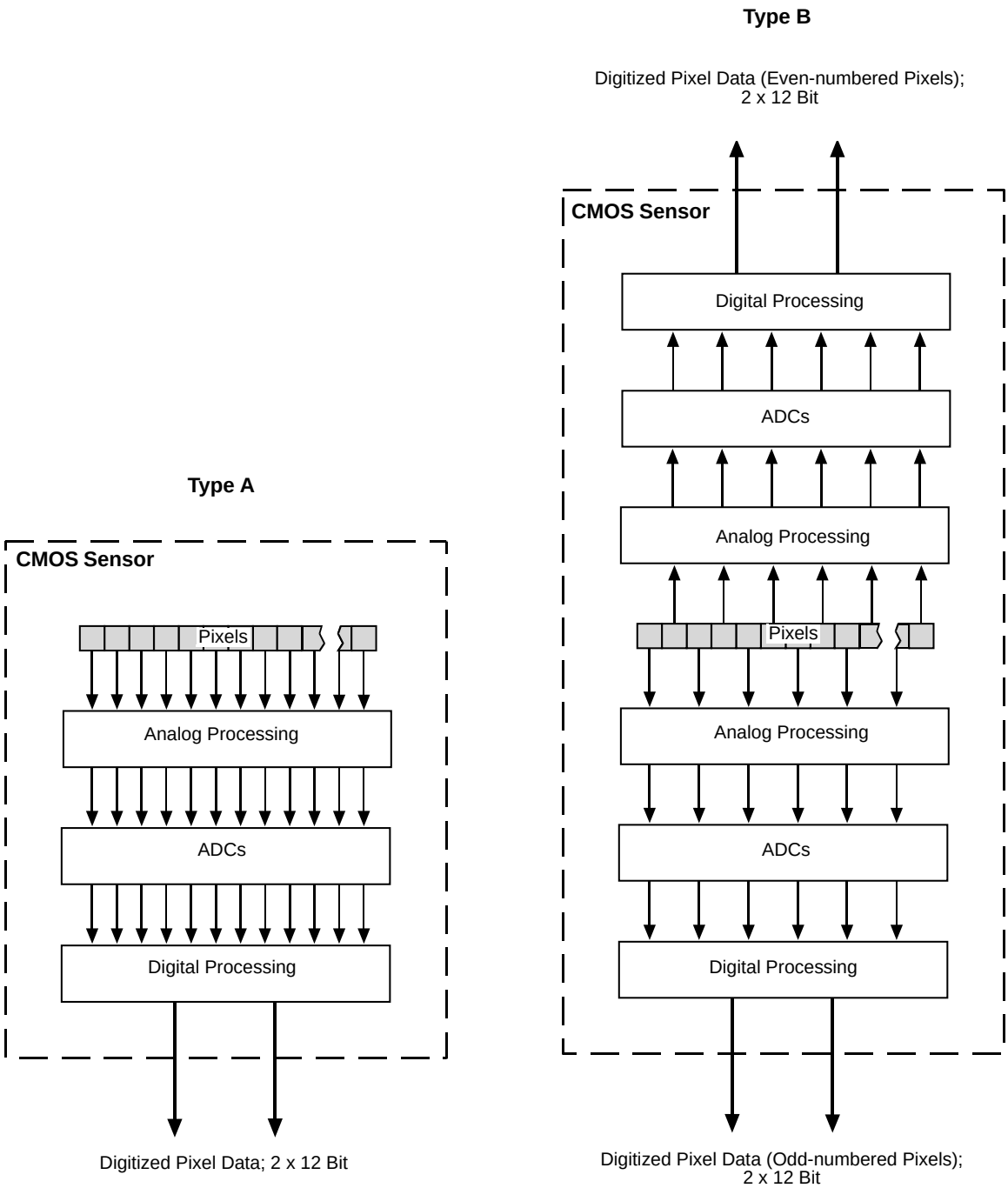


Fig. 14: CMOS Sensor Architecture. Type A is a 2k Sensor or 2k Sensor Segment with a Pixel Size of 7 µm x 7 µm and Type B is a 4k Sensor Segment with a Pixel Size of 3.5 µm x 3.5 µm.

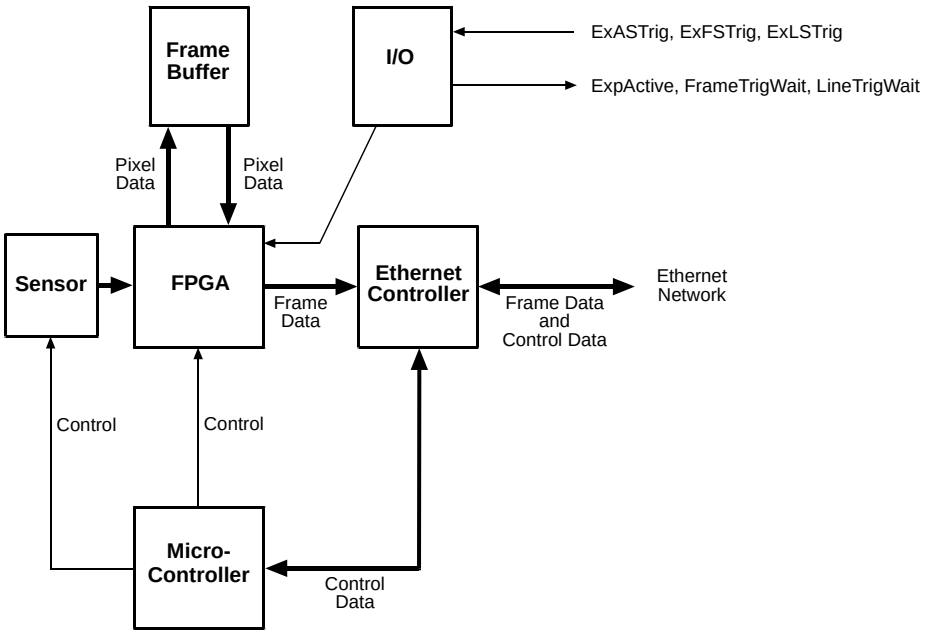


Fig. 15: Camera Block Diagram

7 Physical Interface

This chapter provides detailed information, such as pinouts and voltage requirements, for the physical interface on the camera. This information will be especially useful during your initial design-in process.

7.1 General Description of the Connections

The camera is interfaced to external circuity via connectors located on the back of the housing:

- A 6-pin receptacle used to provide power to the camera.
- A 12-pin receptacle used to provide access to the camera's I/O lines.
- An 8-pin RJ-45 jack used to provide a 100/1000 Mbit/s Ethernet connection to the camera. The jack includes a green LED and a yellow LED that indicate the state of the network connection.

The drawing below shows the location of the three connectors, the LEDs, and the functional earth connection.

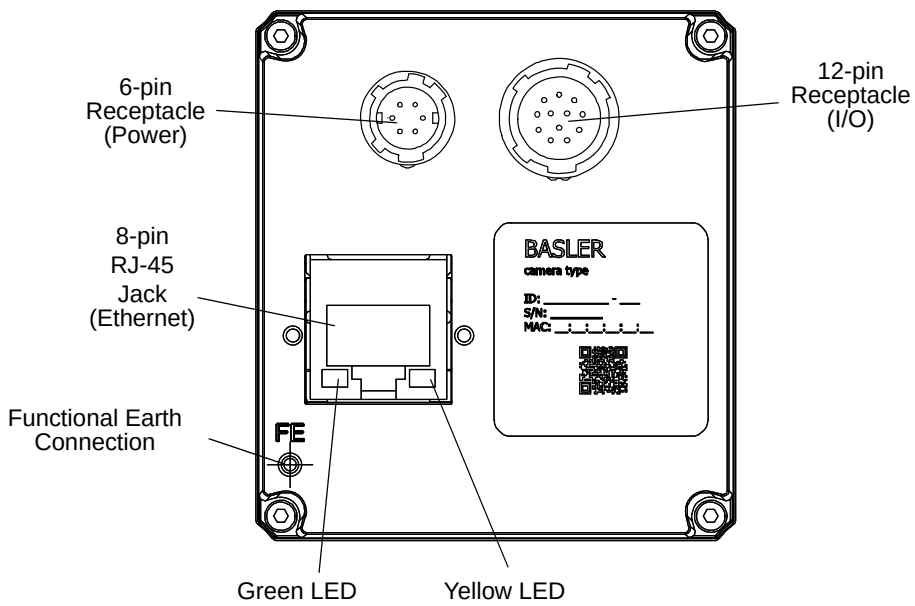


Fig. 16: Camera Connectors , LEDs, and Functional Earth Connection

7.1.1 Pin Numbering

Pin numbering for the camera's 6-pin and 12-pin receptacles is as shown in Fig. 17.

Pin numbering for the 8-pin RJ-45 jack adheres to the Ethernet standard.

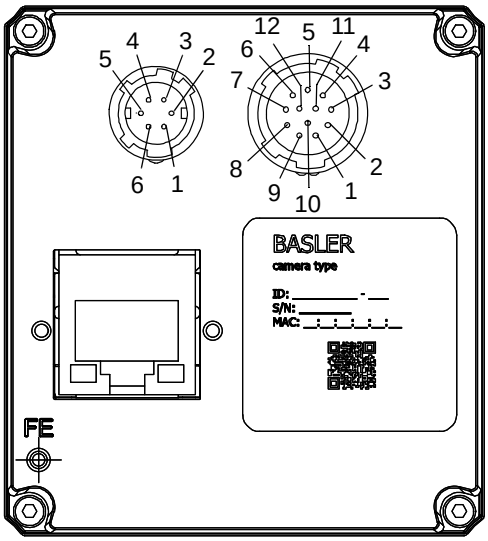


Fig. 17: Pin Numbering for the 6-pin and 12-pin Receptacles

7.2 Connector Pin Assignments

7.2.1 Pin Assignments for the 6-pin Connector

The 6 pin connector is used to supply power to the camera. The pin assignments for the connector are shown in Table 5.

Pin	Designation
1	+12 VDC (- 10 %) to +24 VDC (+ 5 %), < 1 % ripple, Camera Power (*)
2	
3	Not Connected
4	Not Connected
5	DC Ground (**)
6	
* Pins 1 and 2 are tied together inside the camera. ** Pins 5 and 6 are tied together inside the camera. To avoid a voltage drop when there are long wires between your power supply and the camera, we recommend that you provide +12 VDC through two separate wires between the power supply and pins 1 and 2 in the receptacle. We also recommend that you provide the ground through two separate wires between the power supply and pins 5 and 6.	

Table 5: Pin Assignments for the 6-pin Connector

NOTICE

Applying incorrect power can damage the camera.

The camera‘s nominal operating voltage is +12 VDC (- 10 %) to +24 VDC (+ 5 %), effective on the camera‘s connector.

Applying power with the wrong polarity can severely damage the camera.

Make sure that the polarity of the power applied to the camera is correct.

7.2.2 Pin Assignments for the 12-pin Connector

The 12-pin connector is used to access the three physical input lines and two physical output lines on the camera. The pin assignments for the connector are shown in Table 6.

Pin	Designation
1	I/O Input 1 -
2	I/O Input 1+
3	I/O Input 3 -
4	I/O Input 3 +
5	Gnd
6	I/O Output 1-
7	I/O Output 1 +
8	I/O Input 2 -
9	I/O Input 2 +
10	Not connected
11	I/O Output 2 -
12	I/O Output 2+

Table 6: Pin Assignments for the 12-pin Connector

NOTICE

Using a wrong pin assignment for the 12-pin connector can severely damage the camera.

Make sure the cable and plug you connect to the 12-pin connector follows the correct pin assignment. In particular, do **not** use a pin assignment that would be correct for Basler area scan cameras. The 12-pin connectors of Basler line scan and area scan cameras are electrically incompatible.

For the I/O lines to work correctly, pin 5 must be connected to ground.

7.2.3 Pin Assignments for the RJ-45 Jack

The 8-pin RJ-45 jack provides Ethernet access to the camera. Pin assignments adhere to the Ethernet standard.

7.3 Connector Types

7.3.1 6-pin Connector

The 6-pin connector on the camera is a Hirose micro receptacle (part number HR10A-7R-6PB) or the equivalent.

The recommended mating connector is the Hirose micro plug (part number HR10A-7P-6S) or the equivalent.

7.3.2 12-pin Connector

The 12-pin connector on the camera is a Hirose micro receptacle (part number HR10A-10R-12P) or the equivalent.

The recommended mating connector is the Hirose micro plug (part number HR10A-10P-12S) or the equivalent.

7.3.3 RJ-45 Jack

The 8-pin jack for the camera's Ethernet connection is a standard RJ-45 connector.
The recommended mating connector is any standard RJ-45 plug.

Cables terminated with screw-lock connectors are available from Basler. Contact your Basler sales representative to order cable assemblies. Suitable cable assemblies are also available from the Intercon 1 division of Nortech Systems, Inc.

To ensure that you order cables with the correct connectors, note the horizontal orientation of the screws before ordering.

Green and Yellow LEDs

This RJ-45 jack on the camera includes a green LED and a yellow LED.

- When the green LED is lit, it indicates that an active network connection is available.
- When the yellow LED is lit, it indicates that data is being transmitted via the network connection.

7.4 Cabling Requirements

7.4.1 Power Cable

A single power cable is used to supply power to the camera. DC ground and the camera housing (along with the shield contacts of all connectors) are connected within the camera (see Fig. 18).

The end of the power cable that connects to the camera's 6-pin connector must be terminated with a Hirose micro plug (part number HR10A-7P-6S) or the equivalent. The cable must be wired as shown in Fig. 18.

For proper EMI protection, the power cable terminated with the Hirose connector and attached to the camera must be a twin-cored, shielded cable. Also, the Hirose plug must be connected to the cable shield and the shield must be connected to earth ground at the power supply.

Close proximity to strong electromagnetic fields should be avoided.

NOTICE

An incorrect plug can damage the 6-pin connector.

The plug on the cable that you attach to the camera's 6-pin connector must have 6 female pins.

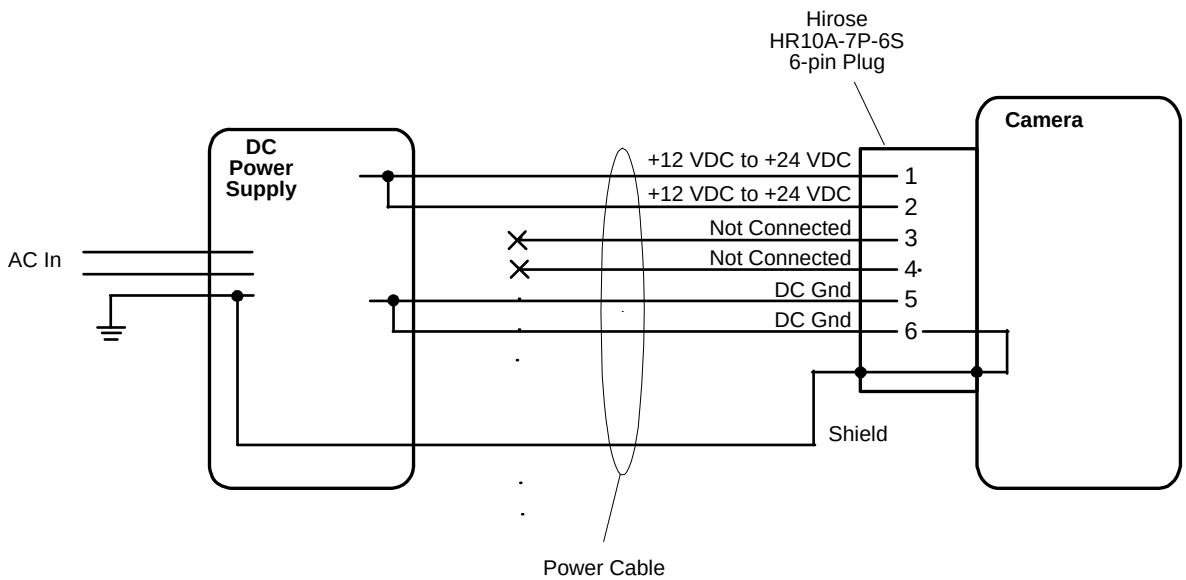


Fig. 18: Power Cable

7.4.2 I/O Cable

The end of the I/O cable that connects to the camera's 12-pin connector must be terminated with a Hirose micro plug (part number HR10A-10P-12S) or the equivalent. The cable must be wired as shown in Fig. 19.

The maximum length of the I/O cable is 10 meters, however, we strongly recommend keeping I/O cables as short as possible. The cable must be shielded and must be constructed with twisted pair wire. Use of twisted pair wire is essential to ensure that input signals are correctly received.

The required 12-pin Hirose plug is available from Basler. Basler also offers an I/O cable assembly that is terminated with a 12-pin Hirose plug on one end and unterminated on the other. Contact your Basler sales representative to order connectors or I/O cables.

Close proximity to strong electromagnetic fields should be avoided.

NOTICE

An incorrect plug can damage the 12-pin connector.

The plug on the cable that you attach to the camera's 12-pin connector must have 12 female pins.

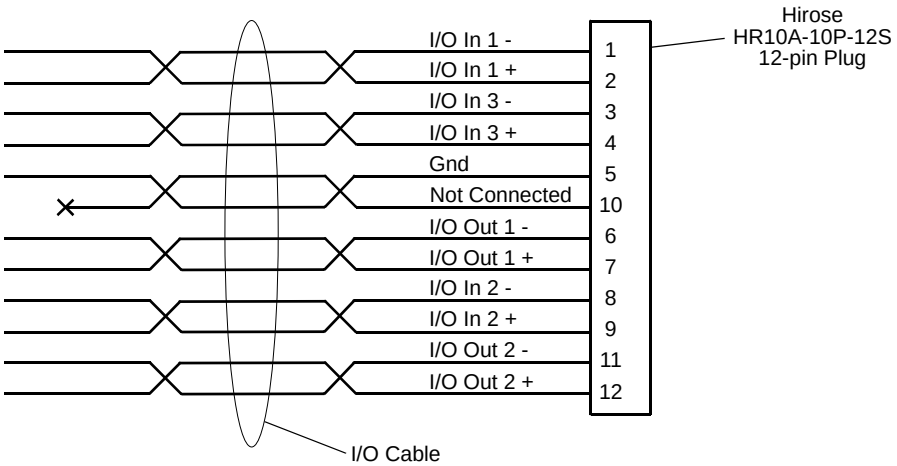


Fig. 19: I/O Cable

7.4.3 Ethernet Cables

Use high-quality Ethernet cables. To avoid EMI, the cables must be shielded. Use of category 6 or category 7 cables with S/STP shielding is strongly recommended. As a general rule, applications with longer cables or applications in harsh EMI conditions require higher category cables.

Either a straight-through (patch) or a cross-over Ethernet cable can be used to connect the camera directly to a GigE network adapter in a PC or to a GigE network switch.

Close proximity to strong electromagnetic fields should be avoided.

7.5 Camera Power

Camera power must be supplied to the 6-pin connector on the camera via a cable from your power supply. The nominal operating voltage of the camera is +12 VDC (± 10 %).The required operating voltage is +12 VDC (- 10 %) to +24 VDC (+ 5 %), effective on the camera‘s connector. Ripple must be less than one percent. Power consumption is as shown in the specification tables in Section 1 of this manual.

Close proximity to strong electromagnetic fields should be avoided.

NOTICE

Applying incorrect power can damage the camera.

The camera‘s required operating voltage is +12 VDC (- 10 %) to +24 VDC (+ 5 %), < 1 % ripple, effective on the camera‘s connector, with a nominal operating voltage of +12 VDC (± 10 %).

Applying power with the wrong polarity can severely damage the camera.

Make sure that the polarity of the power applied to the camera is correct. Applying power with the wrong polarity can severely damage the camera.

NOTICE

An incorrect plug can damage the 6-pin connector.

The plug on the cable that you attach to the camera’s 6-pin connector must have 6 female pins.

For more information about the 6-pin connector, see Section 7.2.1 on page 52 and Section 7.3.1 on page 54.

For more information about the power cable, see Section 7.4.1 on page 55.

7.6 Input and Output Lines

7.6.1 Input Lines

The camera is equipped with three physical input lines designated as Input Line 1, Input Line 2, and Input Line 3. The input lines are accessed via the 12-pin connector on the back of the camera. The inputs are designed to accept RS-422 differential signals, but they can also be used with RS-644 low voltage differential signals or low voltage TTL signals.

7.6.1.1 Electrical Characteristics

Using the Inputs with RS-422

As shown in Fig. 20 and in the I/O schematic at the beginning of this section, each input is designed to receive an RS-422 signal. For the camera's I/O circuitry to operate properly, you must supply a ground as shown in Fig. 20.

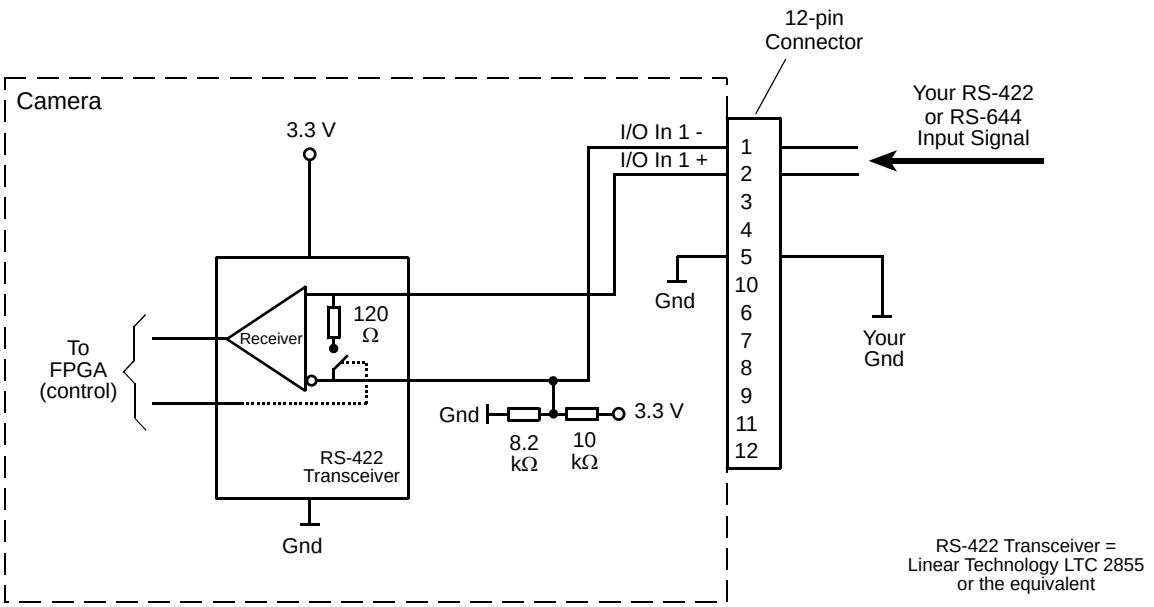


Fig. 20: Inputting RS-422 or RS-644 Signals

The RS-422 standard allows devices to be used with a bus structure to form an interface circuit. So, for example, input line 1 on several different cameras can be connected via an RS-422 bus as shown in Fig. 21.

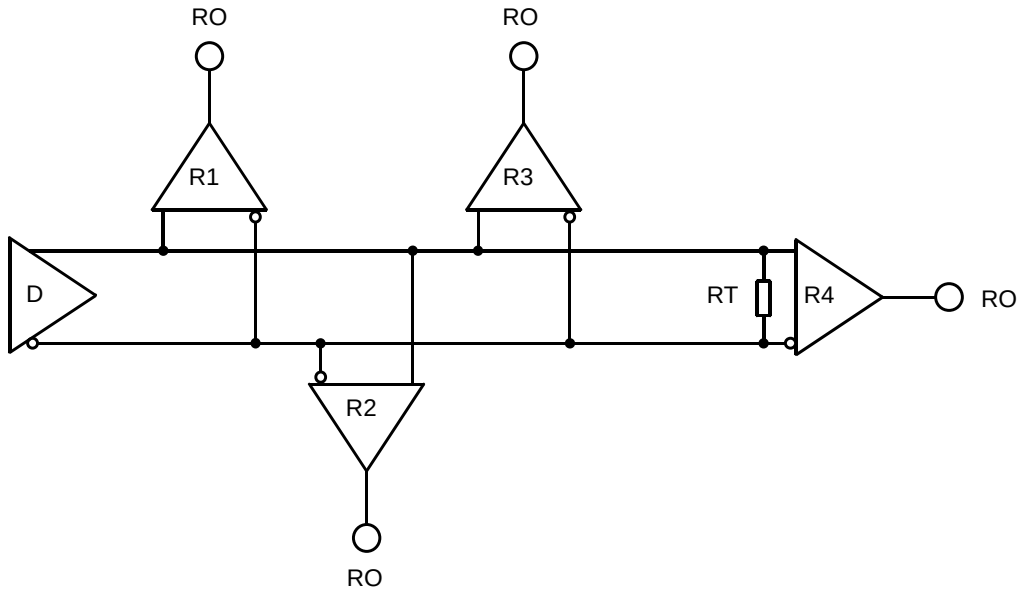


Fig. 21: RS-422 Interface Circuit Including Four Receivers as an Example

Connected to the bus would be one camera as the "master" transmitter (driver D; only one driver allowed) and up to ten cameras (receivers R), with the "master" transmitter sending signals to the "slave" inputs of the receivers. The inputs of the receivers would be connected in parallel to the driver via the bus.

The separations between receivers and bus should be as small as possible. The bus must be terminated by a 120 ohm termination resistor (RT). Each RS-422 input on the cameras includes a switchable 120 ohm termination resistor as shown in Fig. 20. When a camera input of the last receiver in the bus terminates the bus (as shown in Fig. 21.: R4), the termination resistor on that input should be enabled. You should not use multiple termination resistors on a single bus. Using multiple termination resistors will lower signalling reliability and has the potential for causing damage to the RS-422 devices.

Using the Inputs with RS-644 LVDS

The inputs on the camera can accept RS-644 low voltage differential signals (LVDS).

If you are supplying an RS-644 LVDS signal to an input on the camera, the 120 ohm termination resistor on that input **must** be enabled. The input will not reliably react to RS-644 signals if the resistor is disabled.

For the camera's I/O circuitry to operate properly, you must supply a ground as shown in Fig. 20.

Although the RS-644 standard allows several devices to be connected together in a "multidrop" configuration, we strongly recommend that you do not include any camera input in an RS-644 multidrop. Instead, we strongly recommend that you use a direct, point-to-point connection between your RS-644 transmitter and the camera input.

Using the Inputs with LVTTL

A camera input line can accept a Low Voltage TTL signal when the signal is input into the camera as shown in Fig. 22.

The following voltage requirements apply to the camera's I/O input (pin 2 of the 12-pin connector):

Voltage	Significance
+0 to + 5.0 VDC	Recommended operating voltage.
+0 to +0.8 VDC	The voltage indicates a logical 0.
> +0.8 to +2.0 VDC	Region where the transition threshold occurs; the logical state is not defined in this region.
> +2.0 VDC	The voltage indicates a logical 1.
+6.0 VDC	Absolute maximum; the camera can be damaged when the absolute maximum is exceeded.

Table 7: Voltage Requirements for the I/O Input When Using LVTTL

When LVTTL signals are applied to an input, the 120 ohm termination resistor on that input **must** be disabled. The input will not react to LVTTL signals if the resistor is enabled.

For the camera's I/O circuitry to operate properly, you must supply a ground as shown Fig. 22.

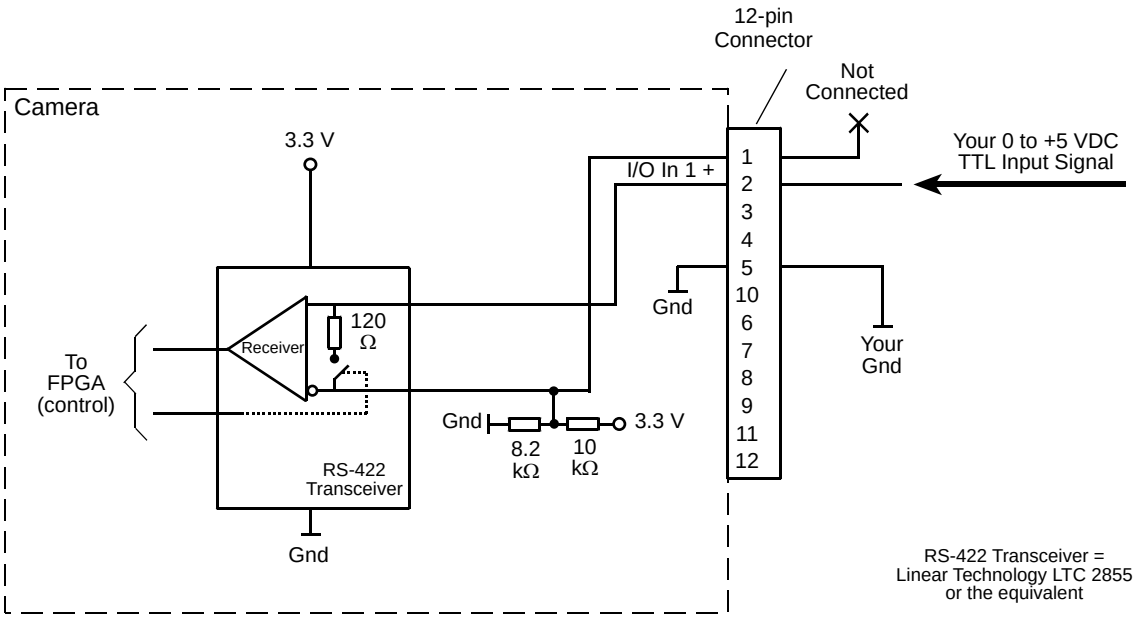


Fig. 22: Inputting Low Voltage TTL Signals

Enabling and Disabling the Termination Resistor

You can select an input line and enable or disable the termination resistor on the line from within your application software by using the pylon API. The following code snippet illustrates using the API to set the parameter values:

```
camera.LineSelector.SetValue(LineSelector_Line1);
camera.LineTermination.SetValue(true);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily enable or disable the resistors.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

7.6.1.2 Input Line Debouncers

Each individual input line is equipped with a debouncer. The debouncer aids in discriminating between valid and invalid input signals. The debouncer value specifies the minimum time that an input signal must remain high or remain low in order to be considered a valid input signal.

We recommend setting the debouncer value so that it is slightly greater than the longest expected duration of an invalid signal.

Setting the debouncer to a value that is too short will result in accepting invalid signals. Setting the debouncer to a value that is too long will result in rejecting valid signals.

The duration of a debouncer is determined by the value of the LineDebouncerTimeAbs parameter value. The parameter is set in microseconds and can be set in a range from 0 to approximately 1 s.

To set a debouncer:

1. Use the LineSelector parameter to select the camera input line for which you want to set the debouncer.
2. Set the value of the LineDebouncerTimeAbs parameter.

You can set the LineSelector and the value of the LineDebouncerAbs parameter from within your application software by using the pylon API. The following code snippet illustrates using the API to set the parameter values:

```
// Select input line 1 and set the debouncer value to 100 microseconds
camera.LineSelector.SetValue(LineSelector_Line1);
camera.LineDebouncerTimeAbs.SetValue(100);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

7.6.1.3 Input Line Inverters

You can set each individual input line to invert or not to invert the incoming electrical signal.

To set the invert function on an input line:

1. Use the LineSelector parameter to select an input line.
2. Set the value of the LineInverter parameter to true to enable inversion on the selected line and to false to disable inversion.

You can set the LineSelector and the LineInverter parameter values from within your application software by using the pylon API. The following code snippet illustrates using the API to set the parameter values:

```
// Enable the inverter on line 1
camera.LineSelector.SetValue(LineSelector_Line1);
camera.LineInverter.SetValue(true);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

7.6.1.4 Selecting an Input Line as a Source Signal for a Camera Function

You can select an input line as the source signal for the following camera functions:

- the Acquisition Start Trigger
- the Frame Start Trigger
- the Line Start Trigger
- the Phase A input for the shaft encoder module
- the Phase B input for the shaft encoder module

To use an input line as the source signal for a camera function, you must apply an electrical signal to the input line that is appropriately timed for the function.

For detailed information about selecting an input line as the source signal for the camera's Acquisition Start Trigger function, see Section 8.2.2.2 on page 79.

For detailed information about selecting an input line as the source signal for the camera's Frame Start Trigger function, see Section 8.2.3.3 on page 84.

For detailed information about selecting an input line as the source signal for the camera's Line Start Trigger function, see Section 8.2.4.2 on page 87 and Section 8.2.4.3 on page 90.

For detailed information about selecting an input line as the source signal for the shaft encoder model Phase A or Phase B input, see Section 8.6 on page 136.

Default Input Line Selections

By default:

- Input Line 1 is selected as the source signal for the camera's Line Start Trigger function.
- Input Line 1 is also selected as the source signal for shaft encoder module Phase A input.
- Input Line 2 is selected as the source signal for shaft encoder module Phase B input.
- Input Line 3 is selected as the source signal for the camera's Frame Start Trigger function.

7.6.2 Output Lines

The camera is equipped with two physical output lines designated as Output Line 1 and Output Line 2. The output lines are accessed via the 12-pin connector on the back of the camera. The outputs are designed to transmit RS-422 differential signals, but they can also be used with RS-644 low voltage differential signalling or low voltage TTL signalling.

7.6.2.1 Electrical Characteristics

Using the Outputs with RS-422

As shown in Fig. 23 and in the I/O schematic at the beginning of this section, each output is designed to transmit an RS-422 signal. For the camera's I/O circuitry to operate properly, you must supply a ground as shown in Fig. 23.

The RS-422 standard allows devices to be used with a bus structure to form an interface circuit. So, for example, output line 1 on a camera can be connected to an RS-422 bus in parallel with the inputs on several of your devices (receivers). The camera with output line 1 connected to the bus would serve as a "master" transmitter to the "slave" inputs of the other connected devices. For more information about an RS-422 interface circuit and a related figure, see the "Using the Inputs with RS-422" section.
Be aware that the last receiver in an RS-422 bus must have a 120 Ohm termination resistor.

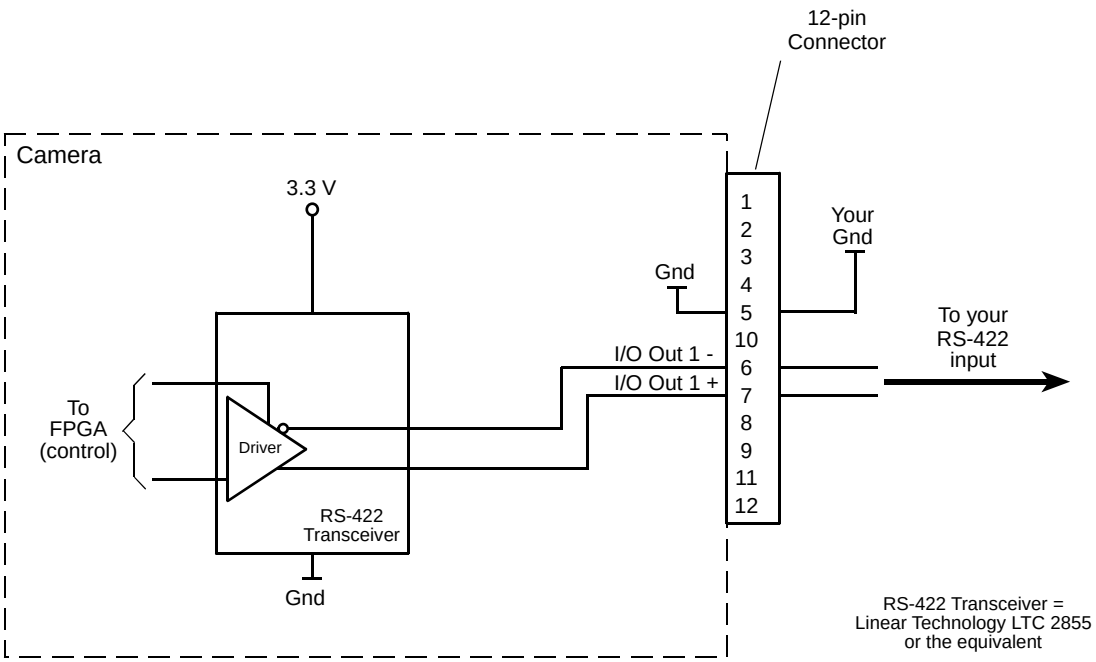


Fig. 23: RS-422 Output Signal

Using the Outputs with RS-644 LVDS

You cannot directly use the RS-422 signal from a camera output line as an input to an RS-644 low voltage differential signal (LVDS) receiver. However, if a resistor network is placed on the camera's output as shown in Fig. 24, you can use the signal from the camera's output line as an input to an RS-644 device.

For the camera's I/O circuitry to operate properly, you must supply a ground as shown in Fig. 24.

	Although the RS-644 standard allows several devices to be connected together in a "multidrop" configuration, we strongly recommend that you do not include any camera output in an RS-644 multidrop. Instead, we strongly recommend that you use a direct, point-to-point connection between the camera and your RS-644 LVDS receiver as shown Fig. 24.

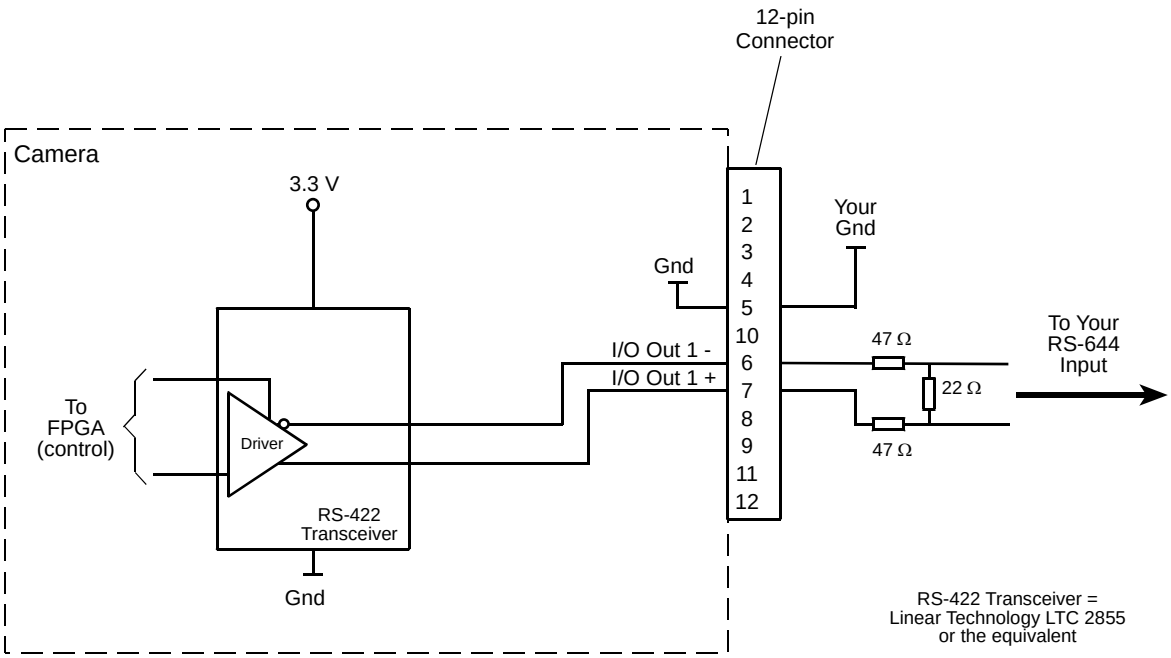


Fig. 24: RS-422 Output Signal Modified for Use with an RS-644 Input

Using the Outputs with LVTTL

You can use a camera output line as an input to a low voltage TTL receiver, but only if the camera's output signal is used as shown in Fig. 25. In this situation, a low will be indicated by a camera output voltage near zero, and a high will be indicated by a camera output voltage of approximately 3.3 VDC. These voltages are within the typically specified levels for low voltage TTL devices.

For the camera's I/O circuitry to operate properly, you must supply a ground as shown in Fig. 25.

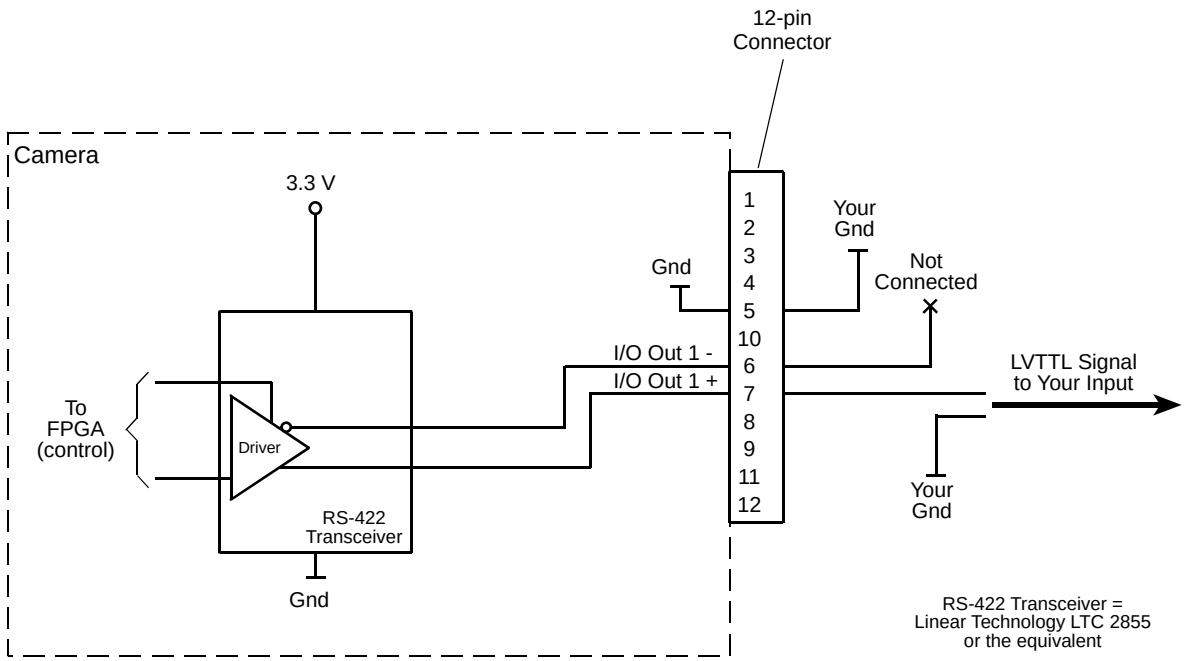


Fig. 25: Output Line Wired for Use with an LVTTL Input

7.6.2.2 Input Related Signals as Output Signals

The camera allows selecting the output signals of the shaft encoder module or of the frequency converter module and assigning them to one of the camera's digital output lines. In this fashion input signals can be passed through a camera to trigger additional cameras.

In this case, setting a minimum output pulse width may be necessary to ensure output signal detection.

For more information about selecting the source signal for an output line on the camera, see Section 7.6.2.5 on page 68.

For more information about the electrical characteristics of the camera's output lines, see Section 7.6.2.1 on page 64.

For more information about the minimum output pulse width feature, see Section 7.6.2.3 on page 67.

7.6.2.3 Minimum Output Pulse Width

You can use the minimum output pulse width feature to ensure that even very narrow camera output signals, e.g. signals originating from a shaft encoder, will reliably be detected by other devices. The MinOutPulseWidthAbs parameter sets output signals for the selected output line to a minimum width. The parameter is set in microseconds and can be set in a range from 0 to 100 µs.

You can set the LineSelector and the MinOutPulseWidthAbs parameter values from within your application software by using the pylon API. The following code snippet illustrates using the API to set the parameter values:

```
// Select the output line
camera.LineSelector.SetValue(LineSelector_Out1);

// Set the parameter value to 10.0 microseconds
camera.MinOutPulseWidthAbs.SetValue(10.0);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

For more information about selecting the source signal for an output line on the camera, see Section 7.6.2.5 on page 68.

For more information about the electrical characteristics of the camera's output lines, see Section 7.6.2.1 on page 64.

7.6.2.4 Output Line Inverters

You can set each individual output line to invert or not to invert the outgoing signal.

To set the invert function on an output line:

1. Use the LineSelector parameter to select an output line.
2. Set the value of the LineInverter parameter to true to enable inversion on the selected line and to false to disable inversion.

You can set the LineSelector and the LineInverter parameter values from within your application software by using the pylon API. The following code snippet illustrates using the API to set the parameter values:

```
// Enable the inverter on output line 1
camera.LineSelector.SetValue(LineSelector_Out1);
camera.LineInverter.SetValue(true);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

7.6.2.5 Selecting the Source Signal for an Output Line

To make a physical output line useful, you must select a source signal for the output line.

The camera has the following standard output signals available that can be selected as the source signal for an output line:

- the Exposure Active signal
- the Acquisition Trigger Wait signal
- the Frame Trigger Wait signal
- the Line Trigger Wait signal
- the Shaft Encoder Module Out signal
- the Frequency Converter signal

You can also select one of the following as the source signal for an output:

- the "User Output" signal (when you select "user output" as the source signal for an output line, you can use the camera's API to set the state of the line as you desire)
- Off (when Off is selected as the source signal, the output is disabled.)

To select one of the standard output signals as the source signal for an output line:

1. Use the LineSelector parameter to select an output line.
2. Set the value of the LineSource parameter to ExposureActive, AcquisitionTriggerWait, FrameTriggerWait, LineTriggerWait, UserOutput, or Off. This will select the source signal for the line.

You can set the LineSelector and the LineSource parameter values from within your application software by using the pylon API. The following code snippet illustrates using the API to set the parameter values:

```
// Disable output line 1
camera.LineSelector.SetValue(LineSelector_Out1);
camera.LineSource.SetValue(LineSource_Off);

// Select the exposure active signal for output line 1
camera.LineSelector.SetValue(LineSelector_Out1);
camera.LineSource.SetValue(LineSource_ExposureActive);

// Select the acquisition trigger wait for output line 2
camera.LineSelector.SetValue(LineSelector_Out2);
camera.LineSource.SetValue(LineSource_AcquisitionTriggerWait);

// Select the frame trigger wait for output line 2
camera.LineSelector.SetValue(LineSelector_Out2);
camera.LineSource.SetValue(LineSource_FrameTriggerWait);
```

```
// Select the line trigger wait signal for output line 2
camera.LineSelector.SetValue(LineSelector_Out2);
camera.LineSource.SetValue(LineSource_LineTriggerWait);OOOKKK

// Select the shaft encoder module out signal for output line 1
camera.LineSelector.SetValue(LineSelector_Out1);
camera.LineSource.SetValue(LineSource_ShaftEncoderModuleOut);

// Select the frequency converter signal for output line 2
camera.LineSelector.SetValue(LineSelector_Out2);
camera.LineSource.SetValue(LineSource_FrequencyConverter);

// Select output line 1 as a user output
camera.LineSelector.SetValue(LineSelector_Out1);
camera.LineSource.SetValue(LineSource_UserOutput);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about

- the pylon API and the pylon Viewer, see Section 3.1 on page 26.
- the Exposure Active signal, see Section 8.3.1 on page 120.
- the Acquisition Trigger Wait signal, see Section 8.3.3.1 on page 122.
- the Frame Trigger Wait signal, see Section 8.3.3.2 on page 124.
- the Line Trigger Wait signal, see Section 8.3.3.3 on page 126.
- working with outputs that have "user settable" as the signal source, see Section 7.6.2.6.

Default Output Line Source Signal Selections

By default, the camera's Exposure Active signal is selected as the source signal for Output Line 1, and the camera's Frame Trigger Wait signal is selected as the source signal for Output Line 2.

7.6.2.6 Setting the State of User Settable Output Lines

As mentioned in the previous section, you can select "user output" as the signal source for an output line. For an output line that has "user output" as the signal source, you can use camera parameters to set the state of the line.

Setting the State of a Single User Output Line

To set the state of a single user output line:

1. Use the User Output selector to select the output line you want to set. For example, if you have designated output line 2 as a user output, you would select output line 2.
2. Set the value of the UserOutputValue parameter to True (high) or False (low). This will set the state of the selected line.

You can set the OutputSelector and the UserOutputValue parameter from within your application software by using the pylon API. The following code snippet illustrates using the API to select "user settable" as the source signal for output line 2 and how to set the state of the output line:

```
// Select "user output" as output line 2 signal source
camera.LineSelector.SetValue(LineSelector_Out2);
camera.LineSource.SetValue(LineSource_UserOutput);

//Set the state of output line 2 and then read the state
camera.UserOutputSelector.SetValue(UserOutputSelector_UserOutput2);
camera.UserOutputValue.SetValue(true);
bool currentUserOutput2State = camera.UserOutputValue.GetValue();
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer’s Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

Setting the State of Multiple User Output Lines

If you have designated both of the cameras output lines as user outputs, you can use the UserOutputValueAll parameter to set the state of both outputs.

The UserOutputValueAll parameter is a 32 bit value. As shown in Fig. 26, the lowest two bits of the parameter value will set the state of the user outputs. If a bit is 0, it will set the state of the associated output to low. If a bit is high, it will set the state of the associated port to high.

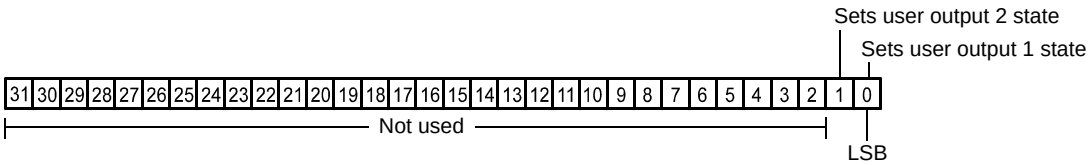


Fig. 26: User Output Value All Parameter Bits

To set the state of multiple user output lines:

- Use the UserOutputValueAll parameter to set the state of multiple user outputs.

You can set the UserOutputValueAll parameter from within your application software by using the pylon API. The following code snippet illustrates using the API to set the parameter:

```
// Set the state of both output lines to 1 and read the state
camera.UserOutputValueAll.SetValue(0x3);
int64_t currentOutputState = camera.UserOutputValueAll.GetValue();
```

If you have the invert function enabled on an output line that is designated as a user output, the user setting sets the state of the line before the inverter.

7.6.3 Checking the State of the I/O Lines

Checking the State of All I/O Lines

You can determine the current state of all input and output lines with a single operation.

To check the state of all lines:

- Read the value of the LineStatusAll parameter.

You can read the LineStatusAll parameter value from within your application software by using the pylon API. The following code snippet illustrates using the API to read the parameter value:

```
// Read the line status all value
int64_t lineState = camera.LineStatusAll.GetValue();
```

The LineStatusAll parameter is a 32 bit value. As shown in Fig. 27, certain bits in the value are associated with each line and the bits will indicate the state of the lines. If a bit is 0, it indicates that

the state of the associated line is currently low. If a bit is 1, it indicates that the state of the associated line is currently high.

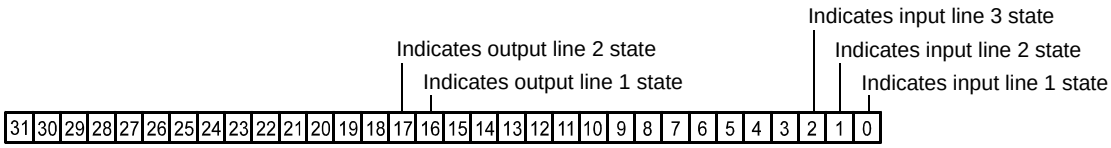


Fig. 27: LineStatusAll Parameter Bits

Checking the State of a Single Output Line

You can determine the current state of an individual output line.

To check the state of a line:

1. Use the LineSelector parameter to select an output line.
2. Read the value of the LineStatus parameter to determine the current state of the selected line. A value of true means the line's state is currently high and a value of false means the line's state is currently low.

You can set the LineSelector parameter value and read the LineStatus parameter value from within your application software by using the pylon API. The following code snippet illustrates using the API to set and read the parameter values:

```
// Select output line 2 and read the state
camera.LineSelector.SetValue(LineSelector_Out2);
bool outputLine2State = camera.LineStatus.GetValue();
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

7.6.4 Checking the Line Logic

To determine the type of line logic for each I/O line:

1. Use the LineSelector parameter to select a line.
2. Read the value of the LineLogic parameter to determine the type of line logic used by the line. The parameter will indicate whether the logic is positive or negative.

You can set the LineSelector parameter value and read the LineLogic parameter value from within your application software by using the Basler pylon API. The following code snippet illustrates using the API to set and read the parameter values:

```
// Select the I/O line and read the line logic type
camera.LineSelector.SetValue(LineSelector_Line1);
LineLogicEnums lineLogicLine1 = camera.LineLogic.GetValue();
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

7.6.5 I/O Line Response Times

In general, the response characteristics for the I/O lines on the camera are as follows:

- Propagation delay for an input receiver (input pins on the camera to the camera's FPGA) is less than 70 ns.
- Propagation delay for an output driver (camera FPGA to the output pins on the camera) is less than 20 ns.
- Signal rise time and signal fall time for the output driver is less than 12.5 ns.

As shown in the I/O schematic at the beginning of this section, the camera's I/O circuitry will incorporate Linear Technology LTC2855 transceivers or the equivalent. For more detailed information about response characteristics, refer to the LTC2855 data sheet.

The response times for the output lines on your camera will fall into the ranges specified above. The exact response time for your specific application will depend on your circuit design.

7.7 Ethernet GigE Device Information

The camera uses a standard Ethernet GigE transceiver. The transceiver is fully 100/1000 Base-T 802.3 compliant.

8 Acquisition Control

The sample code included in this section represents "low level" code that is actually used by the camera.

Many tasks, however, can be programmed more conveniently with fewer lines of code when employing the Instant Camera classes, provided by the Basler pylon C++ API.

For information about the Instant Camera classes, see the *C++ Programmer's Guide and Reference Documentation* delivered with the Basler pylon Camera Software Suite.

This section provides detailed information about controlling the acquisition of image information. You will find details about triggering frame and line acquisition, about setting the exposure time for acquired lines, about setting the camera's line acquisition rate, and about how the camera's maximum allowed line acquisition rate can vary depending on the current camera settings.

8.1 Defining a Frame

As with any other line scan camera, the sensor in a Gigabit Ethernet (GigE) camera is used to perform a series of line acquisitions as an object passes the camera. But unlike many other cameras, GigE line scan cameras do not transmit the pixel data from each individual line to a host PC immediately after the line acquisition is complete. Instead, GigE cameras accumulate acquired lines in a buffer and assemble them into a "frame". When enough line acquisitions have been accumulated to constitute a complete frame, the frame is transmitted via an Ethernet network to a host PC. An acquired frame, therefore, represents a single complete image acquired by the camera.

Three camera parameters, OffsetX, Width, and Height, are used to define what will constitute a frame.

The OffsetX and Width parameters determine the image area of interest (image AOI) on the sensor, i.e. which pixels in the sensor line will be transmitted out of the camera. The OffsetX determines the first pixel to be transmitted and the Width determines the number of pixels to be transmitted. The pixels in the sensor are numbered starting with 0.

Assume, for example, that you are working with a camera that has a 2048 pixel sensor line, that the OffsetX parameter is set to 0, and that the Width parameter is set to 2048. In this case, the full length of the sensor line would be used for each line acquisition.

As another example, assume that the OffsetX parameter is set to 10 and the Width parameter is set to 25. With these settings, pixels 10 through 34 would be used for each line acquisition as shown in Fig. 28.

Fig. 28: Pixels Used for Each Line Acquisition

The Height parameter determines the number of lines that will be included in each frame. For example, assume that the Height parameter is set to 100 and that the camera has just started to acquire lines. In this case, the camera will accumulate acquired line data in an internal buffer until 100 lines have been accumulated. Once pixel data for 100 lines has accumulated in the buffer, the camera will recognize this as a complete frame and it will begin to transmit the acquired frame to your host PC via the GigE network connection. The camera has multiple frame buffers, so it can begin to acquire lines for a new frame as it is transmitting data for the previously acquired frame.

The absolute maximum for the Height parameter value is 12288. Accordingly, a single frame may include 12288 lines at most.

This maximum number of lines can, however, not be obtained under all conditions: In the event of limitations due to the current camera parameter settings or due to the transport layer, the camera will automatically decrease the Height parameter to a suitable value. Each frame will then include fewer lines than originally set.

Given the current camera parameter settings, check the Height parameter to see whether the desired number of lines per frame can actually be obtained.

Guidelines When Setting the Frame Parameters

When setting the frame parameters, the following guidelines must be followed:

- The sum of the OffsetX parameter plus the Width parameter must be less than or equal to the total number of pixels in the camera's sensor line. For example, if you are working with a camera that has a line with 2048 pixels, the sum of the OffsetX setting plus the Width setting must be less than or equal to 2048.
- The Height parameter must be set to or below the maximum allowed value.

  The maximum allowed value for the Height parameter setting will be at least 512, but will vary depending on your camera model and on how the camera's parameters are set. To determine the maximum allowed Height value, see below.

Determining the Maximum Allowed Height Value

The following code snippet illustrates using the pylon API to get the maximum allowed Height value:

```
int64_t widthMax = camera.Width.GetMax();
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

Setting the Frame Parameters

You can set the OffsetX, Width, and Height parameter values from within your application software by using the pylon API. The following code snippets illustrate using the API to get the maximum allowed settings and the increments for the OffsetX, Width and Height parameters. They also illustrate setting the OffsetX, Width, and Height parameter values.

```
int64_t offsetxMax = camera.OffsetX.GetMax();
int64_t offsetxInc = camera.OffsetX.GetInc();
camera.OffsetX.SetValue(100);

int64_t widthMax = camera.Width.GetMax();
int64_t widthInc = camera.Width.GetInc();
camera.Width.SetValue(200);

int64_t heightMax = camera.Height.GetMax();
int64_t heightInc = camera.Height.GetInc();
camera.Height.SetValue(200);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

The Width and Height parameters cannot be changed while the camera is in the process of acquiring frames. If the camera receives commands to change the Width or Height parameter values while it is in the process of acquiring frames:

- If the camera is set for single frame mode, the parameters will not change until the current frame is complete or you issue an acquisition stop command.
- If the camera is set for continuous frame mode, the parameters will not change until you issue an acquisition stop command.

8.2 Controlling Acquisition

Five major elements are involved in controlling the acquisition of images:

- Acquisition start and acquisition stop commands
- The acquisition mode parameter
- Acquisition start triggering
- Frame start triggering
- Line start triggering

8.2.1 Acquisition Start and Stop Commands and the Acquisition Mode

The use of AcquisitionStart and AcquisitionStop commands and the camera's AcquisitionMode parameter setting are related.

Issuing an AcquisitionStart command to the camera prepares the camera to acquire frames. You must issue an AcquisitionStart command to the camera before you can begin acquiring frames.

Issuing an AcquisitionStop command to the camera terminates the camera's ability to acquire frames. When the camera receives an AcquisitionStop command:

- If the camera is not in the process of acquiring a frame, its ability to acquire frames will be terminated immediately.
- If the camera is in the process of acquiring a line for a frame, the line acquisition process will be allowed to finish. Frame acquisition will then be stopped, a partial frame will be transmitted, and the camera's ability to acquire frames will be terminated.

The camera's AcquisitionMode parameter has two settings: single frame and continuous.

If the camera's AcquisitionMode parameter is set to SingleFrame, after an AcquisitionStart command has been issued to the camera, a single frame can be acquired. When acquisition of one frame is complete, the camera will internally issue an AcquisitionStop command and can no longer acquire frames. To acquire another frame, you must issue a new AcquisitionStart command.

If the camera's AcquisitionMode parameter is set to ContinuousFrame, after an AcquisitionStart command has been issued to the camera, frame acquisition can be triggered as desired. Each time the proper frame and line triggers are applied, the camera will acquire and transmit a frame. The camera will retain the ability to acquire frames until an AcquisitionStop command has been issued to the camera. Once the AcquisitionStop command is received, the camera can no longer acquire frames.

To see graphical representations of the use of the AcquisitionStart and AcquisitionStop commands and the AcquisitionMode parameter, refer to the use case diagrams in Section 8.3 on page 119.

Setting the Acquisition Mode and Issuing Start/Stop Commands

You can set the AcquisitionMode parameter value and you can issue AcquisitionStart or AcquisitionStop commands from within your application software by using the pylon API. The code snippet below illustrates using the API to set the AcquisitionMode parameter value and to issue an AcquisitionStart command. The snippet also illustrates setting several parameters regarding frame and line triggering. These parameters are discussed later in this chapter.

```
camera.AcquisitionMode.SetValue(AcquisitionMode_SingleFrame);
camera.TriggerSelector.SetValue(TriggerSelector_FrameStart);
camera.TriggerMode.SetValue(TriggerMode_On);
camera.TriggerActivation.SetValue(TriggerActivation_RisingEdge);
camera.TriggerSelector.SetValue(TriggerSelector_LineStart);
camera.TriggerMode.SetValue(TriggerMode_On);
camera.TriggerActivation.SetValue(TriggerActivation_RisingEdge);
camera.ExposureMode.SetValue(ExposureMode_Timed);
camera.ExposureTimeAbs.SetValue(55);
camera.AcquisitionStart.Execute();
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer’s Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

When the camera's acquisition mode is set to single frame, the maximum possible acquisition frame rate for a given AOI cannot be achieved. This is true because the camera performs a complete internal setup cycle for each single frame.

8.2.2 Acquisition Start Triggering

The acquisition start trigger is used in conjunction with the frame start trigger to control the acquisition of frames. In essence, the acquisition start trigger is used as an enabler for the frame start trigger.

When the acquisition start trigger is enabled, the camera's initial acquisition status is "waiting for acquisition start trigger". When the camera is in this acquisition status, it will ignore any frame start trigger signals it receives. If an acquisition start trigger signal is applied to the camera, it will exit the "waiting for acquisition start trigger" acquisition status and enter the "waiting for frame start trigger" acquisition status. If a frame start trigger signal is applied to the camera, it will exit the "waiting for frame start trigger" acquisition status and enter the "waiting for line start trigger" acquisition status.

In this acquisition status, the camera can react to line start trigger signals and will begin to expose a line each time a proper line start trigger signal is applied.

A primary feature of the acquisition start trigger is that after an acquisition start trigger signal has been applied to the camera and the camera has entered the "waiting for frame start trigger" acquisition status, the camera will return to the "waiting for acquisition start trigger" acquisition status once a specified number of frame start triggers has been received. Before more frames can be acquired, a new acquisition start trigger signal must be applied to the camera to exit it from "waiting for acquisition start trigger". This feature is explained in greater detail in the following sections.

There are two main parameters associated with the acquisition start trigger: the TriggerMode parameter and the AcquisitionFrameCount parameter.

8.2.2.1 TriggerMode (Acquisition Start) = Off

When the TriggerMode parameter for the acquisition start trigger is set to Off, the camera will generate all required acquisition start trigger signals internally, and you do not need to apply acquisition start trigger signals to the camera.

8.2.2.2 TriggerMode (Acquisition Start) = On

When the TriggerMode parameter for the acquisition start trigger is set to On, you must apply an acquisition start trigger to the camera in order to make the camera's acquisition state valid. Once an acquisition start trigger has been applied to the camera and the acquisition state has become valid, the state will remain valid until the camera has acquired the number of frames specified by the AcquisitionFrameCount parameter. At that point, the acquisition state will become invalid, and you must apply a new acquisition start trigger to the camera before it can acquire any more frames.

When the TriggerMode parameter for the acquisition start trigger is set to On, you must select a source signal to serve as the acquisition start trigger. The TriggerSource parameter specifies the source signal. The available selections for the TriggerSource parameter are:

- Software - When the acquisition start trigger source is set to software, the user applies an acquisition start trigger to the camera by issuing an acquisition start TriggerSoftware command to the camera from the host PC.
- Line1, Line2 or Line3 - When the acquisition start trigger source is set to Line1, Line2 or Line3, the user applies an acquisition start trigger to the camera by injecting an externally generated acquisition start trigger signal (referred to as an ExASTrig signal) into physical input line 1, line 2 or line 3 on the camera.

If the TriggerSource parameter for the acquisition start trigger is set to Line1, Line2 or Line3, the user must also set the TriggerActivation parameter. The available settings for the TriggerActivation parameter are:

- RisingEdge - specifies that a rising edge of the hardware trigger signal will act as the acquisition start trigger.
- FallingEdge - specifies that a falling edge of the hardware trigger signal will act as the acquisition start trigger.

When the TriggerMode parameter for the acquisition start trigger is set to On, the camera's AcquisitionMode parameter must be set to continuous.

8.2.2.3 AcquisitionFrameCount

When the TriggerMode parameter for the acquisition start trigger is set to On, you must set the value of the camera's AcquisitionFrameCount parameter. The value of the AcquisitionFrameCount can range from 1 to 65535.

With acquisition start triggering on, the camera will initially be in a "waiting for acquisition start trigger" acquisition status. When in this acquisition status, the camera cannot react to frame start trigger signals. If an acquisition start trigger signal is applied to the camera, the camera will exit the "waiting for acquisition start trigger" acquisition status and will enter the "waiting for frame start trigger" acquisition status. It can then react to frame start trigger signals. When the camera has received a number of frame start trigger signals equal to the current AcquisitionFrameCount parameter setting, it will return to the "waiting for acquisition start trigger" acquisition status. At that point, you must apply a new acquisition start trigger signal to exit the camera from the "waiting for acquisition start trigger" acquisition status.

8.2.2.4 Setting The Acquisition Start Trigger Mode and Related Parameters

You can set the TriggerMode and TriggerSource parameter values for the acquisition start trigger and the AcquisitionFrameCount parameter value from within your application software by using the pylon API.

The following code snippet illustrates using the API to set the acquisition start Trigger Mode to on, the Trigger Source to software, and the AcquisitionFrameCount to 5:

```
// Select the acquisition start trigger
camera.TriggerSelector.SetValue(TriggerSelector_AcquisitionStart);
// Set the mode for the selected trigger
camera.TriggerMode.SetValue(TriggerMode_On);
// Set the source for the selected trigger
camera.TriggerSource.SetValue(TriggerSource_Software);
// Set the acquisition frame count
camera.AcquisitionFrameCount.SetValue(5);
```

The following code snippet illustrates using the API to set the Trigger Mode to on, the Trigger Source to line 1, the Trigger Activation to rising edge, and the AcquisitionFrameCount to 5:

```
// Select the acquisition start trigger
camera.TriggerSelector.SetValue(TriggerSelector_AcquisitionStart);
// Set the mode for the selected trigger
camera.TriggerMode.SetValue(TriggerMode_On);
// Set the source for the selected trigger
camera.TriggerSource.SetValue (TriggerSource_Line1);
// Set the activation mode for the selected trigger to rising edge
camera.TriggerActivation.SetValue(TriggerActivation_RisingEdge);
// Set the acquisition frame count
camera.AcquisitionFrameCount.SetValue(5);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer’s Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

8.2.3 Frame Start Triggering

The frame start trigger is used in conjunction with the line start trigger to control the acquisition of the lines that will be included in each frame. In essence, the frame start trigger is an enabler for the line start trigger, i.e., the camera will only react to line start triggers when the frame start trigger is valid. When the frame start trigger is not valid, line start triggers will be ignored by the camera and will not result in a line acquisition.

The first parameter associated with the frame start trigger is the TriggerMode parameter. The TriggerMode parameter has two available settings: off and on.

8.2.3.1 TriggerMode (Frame Start) = Off

When the Frame Start TriggerMode parameter is set to Off, selection of a source signal for the frame start trigger is not required. With the mode set to Off, the camera operates the frame start trigger automatically. How the camera will operate the frame start trigger depends on the setting of the camera's AcquisitionMode parameter:

- If the AcquisitionMode parameter is set to single frame, the camera will automatically make the frame start trigger valid when it receives an AcquisitionStart command. The trigger will remain valid until enough lines have been acquired to constitute a complete frame and then will become invalid.
- If the AcquisitionMode parameter is set to continuous frame:
  a. The camera will automatically make the frame start trigger valid when it receives an AcquisitionStart command.
  b. The frame start trigger will be held valid until enough lines have been acquired to constitute a complete frame and then will become invalid.
  c. As soon as acquisition of lines for a next frame can start, the frame start trigger will automatically be made valid, will be held valid until enough lines have been acquired to constitute a complete frame, and then will become invalid.
  d. The behavior in step c will repeat until the camera receives an AcquisitionStop command. When an AcquisitionStop command is received, the frame start trigger will become continuously invalid.

8.2.3.2 TriggerMode (Frame Start) = On

When the Frame Start TriggerMode parameter is set to On, you must select a source signal for the frame start trigger. The Frame Start Trigger Source parameter specifies the source of the signal. The available selections for the Frame Start Trigger Source parameter are:

- Software - When the frame start trigger source is set to software, the user triggers frame start by issuing a TriggerSoftware command to the camera from the host PC. Each time a TriggerSoftware command is received by the camera, the frame start trigger will become valid

and will remain valid until enough lines have been acquired to constitute a complete frame. The frame start trigger will then become invalid.

- Line 1 - When the frame start trigger source is set to line 1, the user triggers frame start by applying an external electrical signal (referred to as an ExFSTrig signal) to physical input line 1 on the camera.
- Line 2 - When the frame start trigger source is set to line 2, the user triggers frame start by applying an ExFSTrig signal to physical input line 2 on the camera.
- Line 3 - When the frame start trigger source is set to line 3, the user triggers frame start by applying an ExFSTrig signal to physical input line 3 on the camera.
- Shaft Encoder Module Out - When the frame start trigger source is set to shaft encoder module out, the output signal from the camera's shaft encoder software module will trigger frame start.

If the Frame Start Trigger Source parameter is set to Line 1, Line 2, Line 3, or Shaft Encoder Module Out, the user must also set the Frame Start Trigger Activation parameter. The available settings for the Frame Start Trigger Activation parameter are:

- RisingEdge - specifies that a rising edge of the source signal will make the frame start trigger valid. The frame start trigger will remain valid until enough lines have been acquired to constitute a complete frame and then will become invalid.
- FallingEdge - specifies that a falling edge of the source signal will make the frame start trigger valid. The frame start trigger will remain valid until enough lines have been acquired to constitute a complete frame and then will become invalid.
- LevelHigh - specifies that a rising edge of the source signal will make the frame start trigger valid. The frame start trigger will remain valid as long as the signal remains high. The frame start trigger will become invalid when the signal becomes low.
- LevelLow - specifies that a falling edge of the source signal will make the frame start trigger valid. The frame start trigger will remain valid as long as the signal remains low. The frame start trigger will become invalid when the signal becomes high.

If the Frame Start Trigger Activation parameter is set to LevelHigh or LevelLow, the user must also set the TriggerPartialClosingFrame parameter. The available settings for the TriggerPartialClosingFrame parameter are:

- True: When the frame start trigger signal transitions while a frame is being acquired frame acquisition will stop and only the portion of the frame acquired so far will be transmitted.
- False - When the frame start trigger signal transitions while a frame is being acquired the complete frame will be acquired and transmitted.

By default, Input Line 3 is selected as the source signal for the Frame Start Trigger.

If the Frame Start Trigger Source parameter is set to Shaft Encoder Module Out, the recommended setting for the Frame Start Trigger Activation parameter is RisingEdge.

If the Frame Start Trigger Source parameter is set to Line 1, Line 2, or Line 3, the electrical signal applied to the selected input line must be held high for at least 100 ns for the camera to detect a transition from low to high and must be held low for at least 100 ns for the camera to detect a transition from high to low.

To see graphical representations of frame start triggering, refer to the use case diagrams in Section 8.3 on page 119.

8.2.3.3 Setting the Frame Start Trigger Parameters

You can set the Trigger Mode, Trigger Source, and Trigger Activation parameter values for the frame start trigger from within your application software by using the pylon API. If your settings make it necessary, you can also issue a Trigger Software command. The following code snippet illustrates using the API to set the frame start trigger to mode = on, with rising edge triggering on input line 1:

```
// Select the trigger you want to work with
camera.TriggerSelector.SetValue(TriggerSelector_FrameStart);
// Set the mode for the selected trigger
camera.TriggerMode.SetValue(TriggerMode_On);
// Set the source for the selected trigger
camera.TriggerSource.SetValue (TriggerSource_Line1);
// Set the activation scheme for the selected trigger
camera.TriggerActivation.SetValue(TriggerActivation_RisingEdge);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

8.2.3.4 Frame Timeout

The Frame Timeout Abs parameter allows setting a maximum time (in microseconds) that may elapse for each frame acquisition, i.e. the maximum time for the acquisition of the lines for a frame.

When the frame timeout is enabled and a time is set a partial frame will be transmitted if the set time has elapsed before all lines specified for the frame are acquired. In addition, a frame timeout event will be generated if it was enabled.

The minimum value for the Frame Timeout Abs parameter is 0 and the maximum value is 10000000 (= 10 seconds).

You can enable and configure the frame timeout from within your application software by using the pylon API. The following code snippet illustrates using the API to enable and configure the frame timeout:

```
// enable FrameTimeout and set FrameTimeout value
camera.FrameTimeoutEnable.SetValue(true);
// Although FrameTimeoutAbs is measured in microseconds the current resolution
// is just milliseconds.
double FrameTimeout_us = 20000.0; // 20 ms
camera.FrameTimeoutAbs.SetValue(FrameTimeout_us);
```

You can enable the frame timeout event from within your application software by using the pylon API. The following code snippet illustrates using the API to enable the frame timeout event:

```
// enable FrameTimeout event
camera.EventSelector.SetValue(EventSelector_FrameTimeout);
camera.EventNotification.SetValue(EventNotification_GenICamEvent);

// In order to capture FrameTimeout events:
// Set up an event grabber and register a callback for
// the node 'FrameTimeoutEventPort'
```

For more information about event reporting and enabling an event, see Section 10.5 on page 175.

8.2.4 Line Start Triggering

The line start trigger is used to start a line acquisition. Keep in mind that the camera will only react to a line start trigger when the frame start trigger is valid. If the frame start trigger is invalid, line start triggers will be ignored.

The first parameter associated with the line start trigger is the TriggerMode parameter. The TriggerMode parameter has two available settings: off and on.

8.2.4.1 TriggerMode (Line Start) = Off

When the Line Start TriggerMode parameter is set to Off, selection of a source signal for the line start trigger is not required. With the mode set to Off, the camera operates the line start trigger automatically. How the camera will operate the line start trigger depends on the setting of the camera's AcquisitionMode parameter:

- If the AcquisitionMode parameter is set to **single frame**, the camera will automatically begin generating line start triggers when it receives an AcquisitionStart command. The camera will generate line start triggers until enough lines have been acquired to constitute a complete frame and then will stop generating line start triggers.
- If the AcquisitionMode parameter is set to **continuous frame**, the camera will automatically begin generating line start triggers when it receives an AcquisitionStart command. The camera will continue to generate line start triggers until it receives an AcquisitionStop command.

Setting the Acquisition Line Rate

The rate at which the line start triggers are generated will be determined by the camera's AcquisitionLineRateAbs parameter:

- If AcquisitionLineRateAbs is set to a value less than the maximum allowed line acquisition rate, the parameter limits the line rate to the given value. This is useful if you want to limit the amount of data to be transferred from the camera to the PC.
- If AcquisitionLineRateAbs is set to a value greater than the maximum allowed line acquisition rate, the camera will generate line start triggers at the maximum allowed line rate.

For more information about

- setting the parameter, see Section 8.2.4.3 on page 90.
- the maximum allowed line rate, see Section 8.5 on page 131.

Exposure Time Control with Line Start Trigger Mode Off

When the line start trigger mode is set to Off, the exposure time for each line acquisition is determined by the value of the camera's Exposure Time parameters.

For more information about the camera's exposure time parameters, see Section 8.2.5.2 on page 92.

8.2.4.2 TriggerMode (Line Start) = On

When the TriggerMode parameter for the line start trigger is set to On, you must select a source signal for the line start trigger. The TriggerSource parameter specifies the source signal. The available selections for the TriggerSource parameter are:

- Software - When the line start trigger source is set to software, the user triggers line start by issuing a TriggerSoftware command to the camera from the host PC. Each time a TriggerSoftware command is received by the camera, the line start trigger will become valid. It will become invalid during line acquisition and will become valid again when the next TriggerSoftware command is received and when the camera is ready again for a new line acquisition.
- Line1 - When the line start trigger source is set to Line1, the user triggers each line acquisition start by applying an external electrical signal (referred to as an ExLSTrig signal) to physical input line 1 on the camera.
- Line2 - When the line start trigger source is set to Line2, the user triggers each line acquisition start by applying an ExLSTrig signal to physical input line 2 on the camera.
- Line3 - When the line start trigger source is set to Line3, the user triggers each line acquisition start by applying an ExLSTrig signal to physical input line 3 on the camera.
- ShaftEncoderModuleOut - When the line start trigger source is set to ShaftEncoderModuleOut, the output signal from the camera's shaft encoder software module will trigger each line acquisition start.

If the TriggerSource parameter is set to Line 1, Line 2, Line 3, or ShaftEncoderModuleOut, the user must also set the TriggerActivation parameter for the line start trigger. The available settings for the TriggerActivation parameter are:

- RisingEdge - specifies that a rising edge of the source signal will start a line acquisition.
- FallingEdge - specifies that a falling edge of the source signal will start a line acquisition.

By default, input line 1 is selected as the source signal for the Line Start Trigger.

All line start trigger signals input into the camera when the frame start trigger signal is invalid will be ignored by the camera.

If the Trigger Source parameter is set to Shaft Encoder Module Out, the recommended setting for the Line Start Trigger Activation parameter is RisingEdge.

If the Line Start Trigger Source parameter is set to Line 1, Line 2, or Line 3, the electrical signal applied to the selected input line must be held high for at least 100 ns for the camera to detect a transition from low to high and must be held low for at least 100 ns for the camera to detect a transition from high to low.

If you are using a software trigger, do not trigger line acquisition at a rate that

- exceeds the maximum allowed for the current camera settings. If you apply line start trigger signals to the camera when it is not ready to receive them, the signals will be ignored. For more information about determining the maximum allowed line rate, see Section 8.5 on page 131.
- exceeds the host computer's capacity limits for data transfer or storage or both. If you try to transfer more image data than the host computer is able to process, lines may be dropped. For more information about bandwidth optimization, see the *Installation and Setup Guide for Cameras Used with Basler pylon for Windows* (AW000611).

Exposure Time Control with Line Start Trigger Mode On

When the Line Start TriggerMode parameter is set to On, there are three modes available to control the exposure time for each acquired line: trigger width control, timed control, and control off. You can set the camera's Exposure Mode parameter to select one of the exposure time control modes. The modes are explained in detail below.

If you have the Line Start Trigger Source parameter set to Line 1, Line 2, or Line 3, any one of the two exposure time control modes will work well. You should select the mode that is most appropriate for your application.

If you have the Line Start Trigger Source parameter set to Shaft Encoder Module out, we recommend that you select the timed control mode. The trigger width mode should not be used in this case.

In all cases, the exposure time for each line must be within the minimum and the maximum stated in Table 10 on page 92. This is true regardless of the method used to control exposure.

Trigger Width Exposure Time Control Mode

When the trigger width exposure time control mode is selected, the exposure time for each line acquisition will be directly controlled by the source signal for the line start trigger. If the camera is set for rising edge triggering, the exposure time begins when the signal rises and continues until the signal falls. If the camera is set for falling edge triggering, the exposure time begins when the signal falls and continues until the signal rises. Fig. 29 illustrates trigger width exposure with the camera set for rising edge line start triggering.

Trigger width exposure is especially useful if you intend to vary the length of the exposure time for each acquired line.

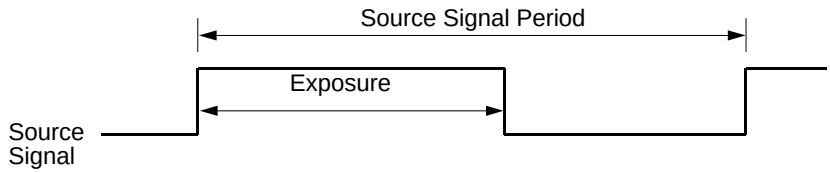


Fig. 29: Trigger Width Exposure with RisingEdge Line Start Triggering

Timed Exposure Control Mode

When the timed exposure control mode is selected, the exposure time for each line acquisition is determined by the value of the camera's Exposure Time parameters. If the camera is set for rising edge triggering, the exposure time starts when the source signal for the line start trigger rises. If the camera is set for falling edge triggering, the exposure time starts when the source signal falls. Fig. 30 illustrates timed exposure with the camera set for rising edge line start triggering.

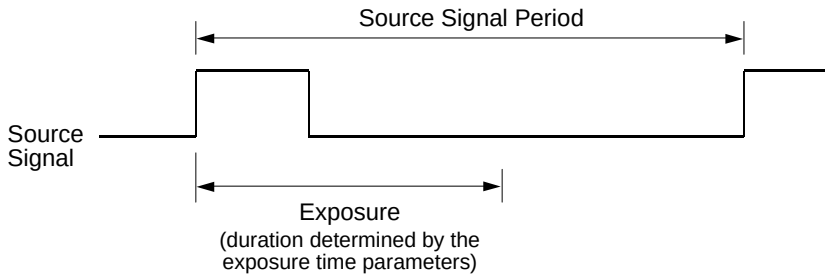


Fig. 30: Timed Exposure with RisingEdge Line Start Triggering

For more information about the camera's exposure time parameters, see Section 8.2.5.2 on page 92.

Exposure Start and Exposure End Delays

When the TriggerMode for the line start trigger is set to On and an input line is selected as the source signal, there is a delay between the transition of the line start signal and the actual start of exposure. For example, if you are using the timed exposure mode with rising edge triggering, there is a delay between the rise of the signal and the actual start of exposure.

There is also an exposure end delay, i.e., a delay between the point when exposure should end as explained in the diagrams on the previous page and when it actually does end.

The base exposure start and end delays are as shown in Table 8:

	raL2048-48gm	raL4096-24gm	raL6144-16gm	raL8192-12gm	raL12288-8 gm
Start Delay	1.5 µs	1.5 µs	1.5 µs	1.5 µs	1.5 µs

Table 8: Base Exposure Start and End Delays

	raL2048-48gm	raL4096-24gm	raL6144-16gm	raL8192-12gm	raL12288-8 gm
End Delay	1.2 µs	1.2 µs	1.2 µs	1.2 µs	1.2 µs

Table 8: Base Exposure Start and End Delays

When using the frequency converter, the delay values may slightly differ from those given in Table 8.

There is also a second component to the start and end delays. This second component is the debouncer setting for the input line. The debouncer setting for the input line must be added to the base start and end delays shown in Table 8 to determine the total start delay and end delay.
For example, assume that you are using an raL2048-48gm camera and that you have set the line start trigger mode to on. Also assume that you have selected input line 1 as the source signal for the line start trigger and that the debouncer parameter for line 1 is set to 5 µs. In this case:

Total Start Delay = Start Delay Value from Table 8 + Debouncer Setting

Total Start Delay = 1.5 µs + 5 µs

Total Start Delay = 6.5 µs

Total End Delay = End Delay Value from Table 8 + Debouncer Setting

Total End Delay = 1.2 µs + 5 µs

Total End Delay = 6.2 µs

8.2.4.3 Setting the Line Start Trigger Parameters

You can set the Trigger Mode, Trigger Source, and Trigger Activation parameter values for the line start trigger from within your application software by using the pylon API. If your settings make it necessary, you can also select an exposure mode and set the exposure time.

The following code snippet illustrates using the API to set the line start trigger to mode = off, the acquisition line rate to 20000, and the exposure time to 50 µs:

```
// Select the trigger you want to work with
camera.TriggerSelector.SetValue(TriggerSelector_LineStart);
// Set the mode for the selected trigger
camera.TriggerMode.SetValue(TriggerMode_Off);
// set a line rate
camera.AcquisitionLineRateAbs.SetValue(20000);
// set the exposure time to 50 µs
camera.ExposureTimeAbs.SetValue(50.0);
```

The following code snippet illustrates using the API to set the line start trigger to mode = on, to set rising edge triggering on input line 2, to set the exposure mode to timed, and to set the exposure time to 60 µs:

```
// Select the trigger you want to work with
camera.TriggerSelector.SetValue(TriggerSelector_LineStart);
// Set the mode for the selected trigger
camera.TriggerMode.SetValue(TriggerMode_On);
// Set the source for the selected trigger
camera.TriggerSource.SetValue (TriggerSource_Line2);
// Set the activation for the selected trigger
camera.TriggerActivation.SetValue(TriggerActivation_RisingEdge);

// set for the timed exposure mode and set exposure time to 60 µs
camera.ExposureMode.SetValue(ExposureMode_Timed);
camera.ExposureTimeAbs.SetValue(60.0);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

8.2.5 Exposure Time

As described in Section 8.2.4.1 on page 86, when you are operating the camera with the TriggerMode parameter for the line start trigger set to Off, the exposure time for each line acquisition will be determined by the camera's exposure time parameters.

As described in Section 8.2.4.2 on page 87, when you are operating the camera with the Line Start Trigger Mode set to On, the exposure time for each line acquisition may be controlled by an external signal or it may be determined by the exposure time parameters.

8.2.5.1 Minimum and Maximum Exposure Times

If you are operating the camera in either of these two ways:

- the TriggerMode for the line start trigger is set to Off
- the TriggerMode for the line start trigger is set to On and the Timed Exposure Time Control Mode is selected

the exposure time will be determined by the settings for the camera's exposure time parameters. The minimum and the maximum allowed exposure time for each acquired line are as shown in Table 9.

	raL2048-48gm	raL4096-24 gm	raL6144-16 gm	raL8192-12 gm	raL12288-8 gm
Min	2.0 µs	2.0 µs	2.0 µs	2.0 µs	2.0 µs
Max	10000 µs	10000 µs	10000 µs	10000 µs	10000 µs

Table 9: Minimum and Maximum Allowed Exposure Times

If you are operating the camera in this way:

- the Line Start Trigger Mode is set to On and the Trigger Width Exposure Time Control Mode is selected

The exposure time for each acquired line will be controlled by an external signal. The minimum allowed exposure time for each acquired line is as shown in Table 10 and there is no limit on the maximum exposure time. Keep in mind, however, that using a very long exposure time can lead to significant degradation of the image quality.

	raL2048-48gm	raL4096-24gm	raL6144-16 gm	raL8192-12 gm	raL12288-8 gm
Min	2.0 µs	2.0 µs	2.0 µs	2.0 µs	2.0 µs

Table 10: Minimum Allowed Exposure Times

8.2.5.2 Exposure Time Parameters

If you are operating the camera in either of the following ways, you must specify an exposure time by setting the camera's exposure time parameters:

- the Line Start Trigger Mode is set to Off
- the Line Start Trigger Mode is set to On and the Timed Exposure Time Control Mode is selected

There are two ways to specify the exposure time: by setting "raw" parameter values or by setting an "absolute" parameter value. The two methods are described below. You can use whichever method you prefer to set the exposure time.

Setting the Exposure Time Using "Raw" Settings

When exposure time is set using "raw" values, the exposure time will be determined by a combination of two elements. The first element is the value of the Exposure Time Raw parameter. The second element is the Exposure Time Base that is 100 ns on racer cameras.

Exposure Time = (Exposure Time Raw Parameter Value) x 100 ns

The Exposure Time Raw parameter value can be set in a range from 1 to 4095.

The exposure time, i.e. the product of the Exposure Time Raw parameter setting and the Exposure Time Base Abs parameter value (i.e., 100 ns) must be equal to or greater than the minimum exposure specified in the table on the previous page. It is possible to use the parameters to set the exposure time lower than what is shown in the table, but this is not allowed and the camera will not operate properly when set this way.

If you are using a GenICam compliant tool such as the Basler pylon Viewer and you attempt to set the exposure time to exactly the minimum allowed or to exactly the maximum allowed, you will see unusual error codes. This is an artifact of a rounding error in the GenICam interface architecture. As a work around, you could set the exposure time slightly above the minimum or below the maximum. Values between the minimum and the maximum are not affected by the problem.

You can set the Exposure Time Raw parameter value from within your application software by using the pylon API. The following code snippet illustrates using the API to set the parameter values:

```
camera.ExposureMode.SetValue(ExposureMode_Timed);
camera.ExposureTimeRaw.SetValue(2);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

Setting the Exposure Time Using "Absolute" Settings

You can also set the exposure time with an "absolute" parameter. This is accomplished by setting the camera's ExposureTimeAbs parameter. The unit for the ExposureTimeAbs parameter is µs.

The setting for the ExposureTimeAbs parameter must be between the minimum and the maximum allowed values (inclusive) shown in Table 8 on page 89.

The increment for the ExposureTimeAbs parameter is determined by the Exposure Time Base Abs parameter value, i.e. 100 ns. For example, you can set the ExposureTimeAbs parameter to 15.0 µs, 15.1 µs, 15.2 µs, etc.

If you set the ExposureTimeAbs parameter to a value that is not a multiple of the Exposure Time Base parameter value (i.e., 100 ns), the camera will automatically change the setting for the ExposureTimeAbs parameter to the nearest multiple of 100 ns.

You should also be aware that if you change the exposure time using the raw settings, the ExposureTimeAbs parameter will automatically be updated to reflect the new exposure time.

You can set the ExposureTimeAbs parameter value from within your application software by using the pylon API. The following code snippet illustrates using the API to set the parameter value:

```
camera.ExposureTimeAbs.SetValue(124.0);
double resultingExpTime = camera.ExposureTimeAbs.GetValue();
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

8.2.6 Use Case Descriptions and Diagrams

The following pages contain a series of use case descriptions and diagrams. The descriptions and diagrams are designed to illustrate how acquisition start triggering, frame start triggering and line start triggering will work with common combinations of parameter settings.

These use cases do not represent every possible combination of the parameters associated with acquisition start, frame start, and line start triggering. They are simply intended to aid you in developing an initial understanding of how triggers and parameters interact.

In each diagram, the black box in the upper left corner indicates how the parameters are set. The number of lines per frame parameter (Height), is set to three for each diagram. This is not realistic, but is used in the diagrams so that they will more conveniently fit onto a single page.

Use Case 1 - Acquisition Start, Frame Start, and Line Start Triggering Off (Free Run), SingleFrame Mode

Use case one is illustrated on page 95.

In this use case, the TriggerMode parameter for the acquisition start trigger, the frame start trigger, and the line start trigger is set to Off. The camera will internally manage acquisition start, frame start, and line start trigger signals. When the camera is set this way, it will acquire lines without any need for triggering by the user. This use case is commonly referred to as "free run".

The rate at which the camera will acquire lines will normally be determined by the camera's AcquisitionLineRateAbs parameter. If the AcquisitionLineRateAbs parameter is disabled, the camera will acquire lines at the maximum allowed line rate.

In this example, each frame is set to include three lines.

When the AcquisitionMode parameter is set to SingleFrame, an acquisition start command must be issued for the acquisition of each single frame.

Settings:
AcquisitionMode = SingleFrame

TriggerMode (Acquisition Start) = Off

TriggerMode (Frame Start) = Off
Height (Lines Per Frame) = 3

TriggerMode (Line Start) = Off

= trigger signal internally generated by the camera; trigger wait signal is not available

= line exposure and readout; white horizontal ruling: period used for exposure and exposure overhead

= frame transmitted

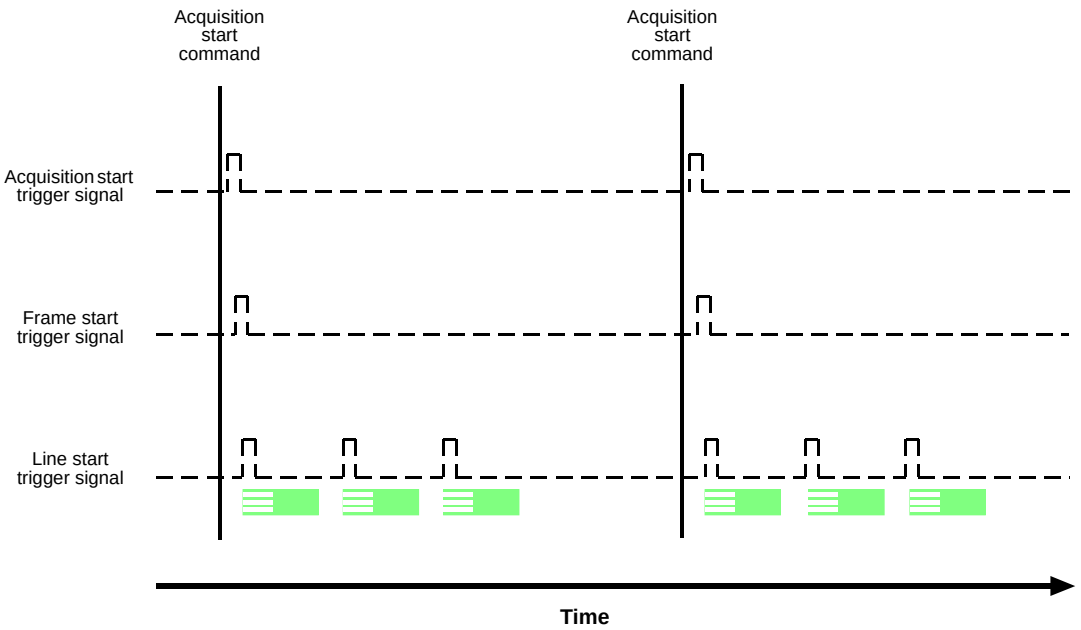


Fig. 31: Use Case 1 - Single Frame Mode with Acquisition Start, Frame Start, and Line Start Triggering Set to Off

Use Case 2 - Acquisition Start, Frame Start, and Line Start Triggering Off (Free Run), ContinuousFrame Mode

Use case two is illustrated on page 97.

This use case is equivalent to the preceding use case one, except for the fact that the acquisition mode is set to ContinuousFrame.

In this use case, the TriggerMode parameter for the acquisition start trigger, the frame start trigger, and the line start trigger is set to Off. The camera will internally manage acquisition start, frame start, and line start trigger signals. When the camera is set this way, it will constantly acquire lines without any need for triggering by the user. This use case is commonly referred to as "free run".

The rate at which the camera will acquire lines will normally be determined by the camera's AcquisitionLineRateAbs parameter. If the AcquisitionLineRateAbs parameter is disabled, the camera will acquire lines at the maximum allowed line rate.

In this example, each frame is set to include three lines.

When the AcquisitionMode parameter is set to ContinuousFrame, the camera will acquire frames until an acquisition stop command is issued.

If an acquisition stop command is issued when not all lines of the current frame are yet acquired, the partial frame will be transmitted.

Settings:
AcquisitionMode = ContinuousFrame

TriggerMode (Acquisition Start) = Off

TriggerMode (Frame Start) = Off
Height (Lines Per Frame) = 3

TriggerMode (Line Start) = Off

= trigger signal internally generated by the camera; trigger wait signal is not available

= line exposure and readout; white horizontal ruling: period used for exposure and exposure overhead

= complete frame transmitted

= partial frame transmitted

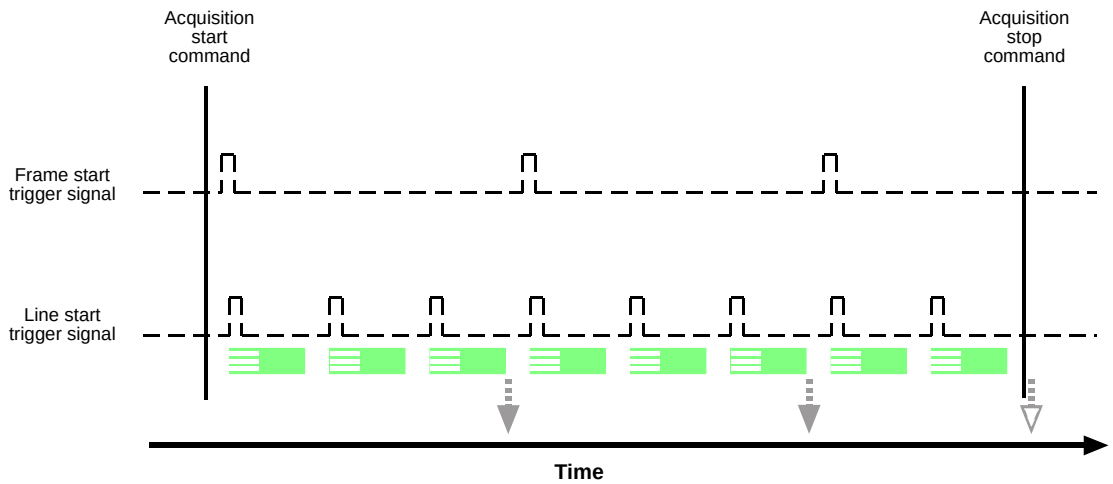


Fig. 32: Use Case 2 - Continuous Frame Mode with Acquisition Start, Frame Start and Line Start Triggering Set to Off

Use Case 3 - Acquisition Start and Line Start Triggering Off (Free Run), Frame Start Triggering On

Use case three is illustrated on page 99.

In this use case, the TriggerMode parameter for the acquisition start trigger and the line start trigger is set to Off. The camera will internally manage acquisition start and line start trigger signals without any need for triggering by the user ("free run").

The TriggerMode parameter for the frame start trigger is set to On, requiring that a frame start trigger signal is applied to the camera.

The rate at which the camera will acquire lines will normally be determined by the camera's AcquisitionLineRateAbs parameter. If the AcquisitionLineRateAbs parameter is disabled, the camera will acquire lines at the maximum allowed line rate. The overall line rate will also depend on the frame start trigger signal: Lines will only be acquired after a related preceding frame start trigger signal has transitioned.

In this example, each frame is set to include three lines.

When the AcquisitionMode parameter is set to ContinuousFrame, the camera will be set to acquire frames until an AcquisitionStop command is issued.

Settings:
AcquisitionMode = ContinuousFrame

TriggerMode (Acquisition Start) = Off

TriggerMode (Frame Start) = On
TriggerSource (Frame Start) = Line2
TriggerActivation (Frame Start) = RisingEdge
Height (Lines Per Frame) = 3

TriggerMode (Line Start) = Off

= trigger signal internally generated by the camera; trigger wait signal is not available

= trigger signal applied by the user

= camera is waiting for a frame start trigger signal

= line exposure and readout; white horizontal ruling: period used for exposure and exposure overhead

= frame transmitted

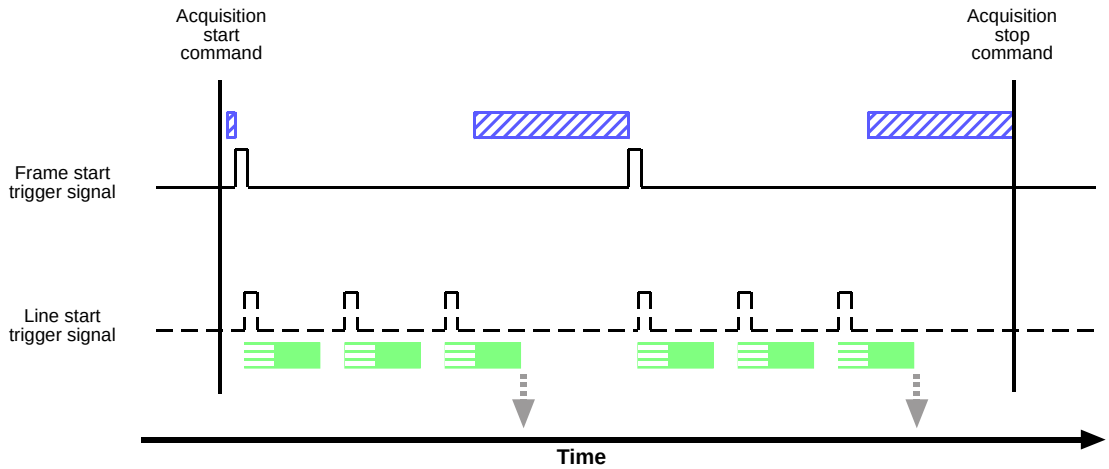


Fig. 33: Use Case 3 - Continuous Frame Mode with Acquisition Start and Line Start Triggering Set to Off and Frame Start Triggering Set to On

Use Case 4 - Acquisition Start Triggering Off (Free Run), Frame Start and Line Start Triggering On

Use case four is illustrated on page 101.

In this use case, the TriggerMode parameter for the acquisition start trigger is set to Off. The camera will internally manage acquisition start trigger signals without any need for triggering by the user ("free run").

The TriggerMode parameter for the frame start trigger and the line start trigger is to On, requiring that frame start and line start trigger signals are applied to the camera.

The rate at which the camera will acquire lines will be determined by the line start trigger signal and must be below the maximum allowed line rate determined by the current setting. The overall line rate will also depend on the frame start trigger signal: Lines will only be acquired after a related preceding transition of frame start trigger signal has occurred.

In this example, each frame is set to include three lines.

When the AcquisitionMode parameter is set to ContinuousFrame, the camera will be set to acquire frames until an AcquisitionStop command is issued.

Settings:
AcquisitionMode = ContinuousFrame

TriggerMode (Acquisition Start) = Off

TriggerMode (Frame Start) = On
TriggerSource (Frame Start) = Line2
TriggerActivation (Frame Start) = RisingEdge
Height (Lines Per Frame) = 3

TriggerMode (Line Start) = On
TriggerSource (Line Start) = Line3
TriggerActivation (Line Start) = RisingEdge

= trigger signal internally generated by the camera; trigger wait signal is not available

= trigger signal applied by the user

= camera is waiting for a frame start trigger signal

= camera is waiting for a line start trigger signal

= line exposure and readout; white horizontal ruling: period used for exposure and exposure overhead

= line start trigger signal is ignored because the camera is waiting for a frame start trigger signal

= frame transmitted

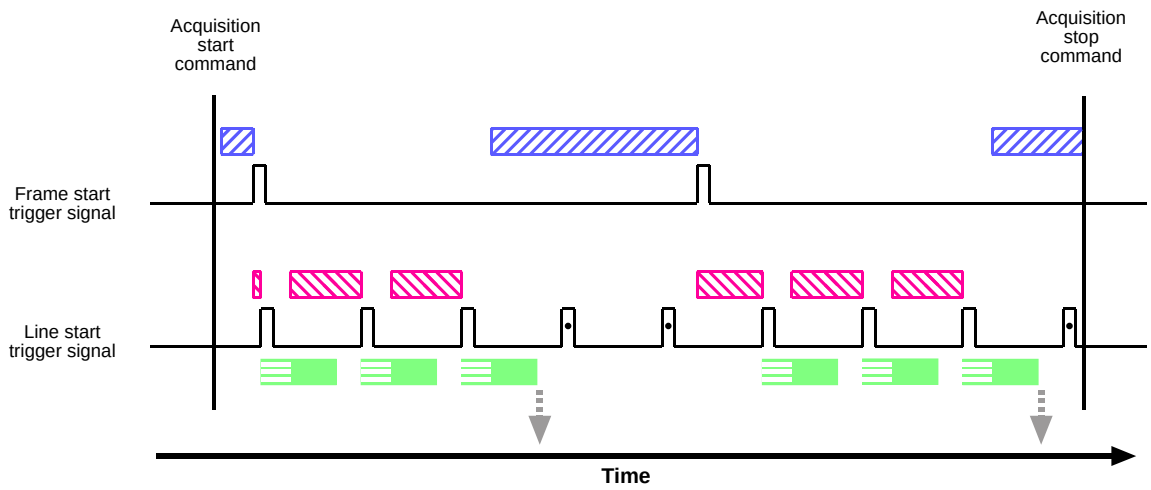


Fig. 34: Use Case 4 - Continuous Frame Mode with Acquisition Start Triggering Set to Off and Frame Start and Line Start Triggering Set to On

Use Case 5 - Acquisition Start Triggering Off (Free Run), Frame Start and Line Start Triggering On, Frame Start Trigger LevelHigh, TriggerPartialClosingFrame False

Use case five is illustrated on page 103.

In this use case, the TriggerMode parameter for the acquisition start trigger is set to Off. The camera will internally manage acquisition start trigger signals without any need for triggering by the user ("free run").

The TriggerMode parameter for the frame start trigger and the line start trigger is set to On, requiring that frame start and line start trigger signals are applied to the camera.

The TriggerActivation parameter for the frame trigger is set to LevelHigh. This means that a transition of the frame start trigger signal is always be present as long as the signal stays high. Accordingly, during this period frames can be acquired without interruption which otherwise will happen if a related preceding transition of the frame start trigger signal has not occurred (c.f. also use case four).

In this example, each frame is set to include three lines.

In this example, the frame start trigger signal goes low while a frame is being acquired (i.e. while line one of the closing frame of the sequence of frames is being acquired). However, with TriggerPartialClosingFrame set to false, the complete closing frame will be acquired and transmitted.

When the AcquisitionMode parameter is set to ContinuousFrame, the camera will be set to acquire frames until an acquisition stop command is issued.

Settings:
AcquisitionMode = ContinuousFrame

TriggerMode (Acquisition Start) = Off

TriggerMode (Frame Start) = On
TriggerSource (Frame Start) = Line2
TriggerActivation (Frame Start) = LevelHigh
TriggerPartialClosingFrame = False
Height (Lines Per Frame) = 3

TriggerMode (Line Start) = On
TriggerSource (Line Start) = Line3
TriggerActivation (Line Start) = RisingEdge

= trigger signal internally generated by the camera; trigger wait signal is not available

= trigger signal applied by the user

= camera is waiting for a frame start trigger signal

= camera is waiting for a line start trigger signal

= line exposure and readout; white horizontal ruling: period used for exposure and exposure overhead

= line start trigger signal is ignored because the camera is waiting for a frame start trigger signal

= frame transmitted

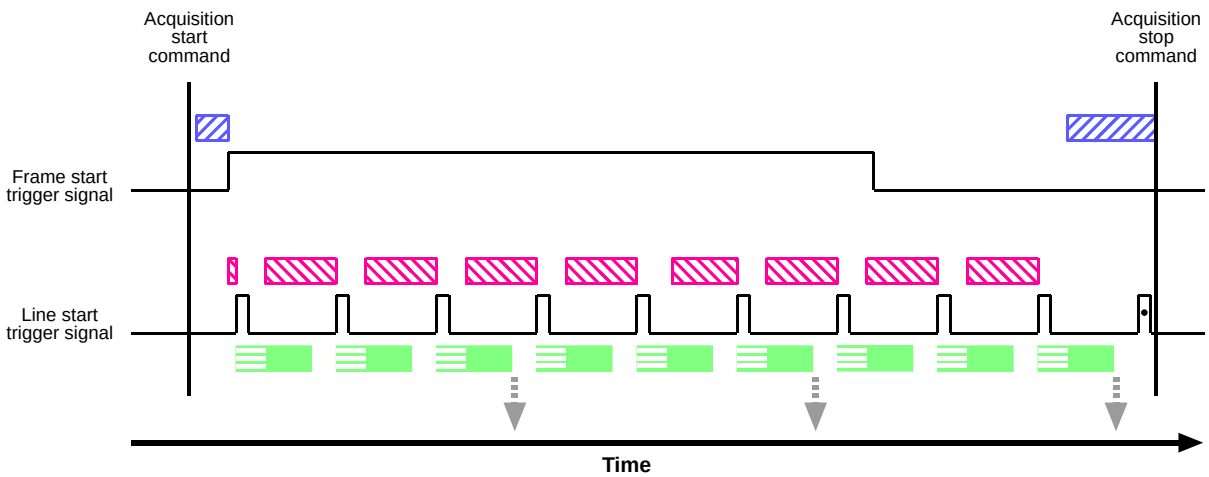


Fig. 35: Use Case 5 - Continuous Frame Mode with Acquisition Start Triggering Set to Off, Frame Start and Line Start Triggering Set to On, and TriggerPartialClosingFrame set to False

Use Case 6 - Acquisition Start Triggering Off (Free Run), Frame Start and Line Start Triggering On, Frame Start Trigger LevelHigh, TriggerPartialClosingFrame True

Use case six is illustrated on page 105.

This use case is equivalent to the preceding use case five, except for the fact that TriggerPartialClosingFrame is set to True.

In this use case, the TriggerMode parameter for the acquisition start trigger is set to Off. The camera will internally manage acquisition start trigger signals without any need for triggering by the user ("free run").

The TriggerMode parameter for the frame start trigger and for the line start trigger is set to On, requiring that frame start and line start trigger signals are applied to the camera.

The TriggerActivation parameter for the frame start trigger is set to LevelHigh. This means that a transition of the frame start trigger signal is always present as long as the signal stays high. Accordingly, during this period frames can be acquired without interruption which otherwise will happen if a related preceding transition of the frame start trigger signal has not occurred (c.f. also use case four).

In this example, each frame is set to include three lines.

In this example, the frame start trigger signal goes low while a frame is being acquired (i.e. while line one of the closing frame of the sequence of frames is being acquired). With TriggerPartialClosingFrame set to true, only the partial closing frame will be acquired and transmitted. In this example, the partial closing frame includes only one line.

When the AcquisitionMode parameter is set to ContinuousFrame, the camera will be set to acquire frames until an acquisition stop command is issued.

Settings:
AcquisitionMode = ContinuousFrame

TriggerMode (Acquisition Start) = Off

TriggerMode (Frame Start) = On
TriggerSource (Frame Start) = Line2
TriggerActivation (Frame Start) = LevelHigh
TriggerPartialClosingFrame = True
Height (Lines Per Frame) = 3

TriggerMode (Line Start) = On
TriggerSource (Line Start) = Line3
TriggerActivation (Line Start) = RisingEdge

= trigger signal internally generated by the camera; trigger wait signal is not available

= trigger signal applied by the user

= camera is waiting for a frame start trigger signal

= camera is waiting for a line start trigger signal

= line exposure and readout; white horizontal ruling: period used for exposure and exposure overhead

= line start trigger signal is ignored because the camera is waiting for a frame start trigger signal

= complete frame transmitted

= partial frame transmitted

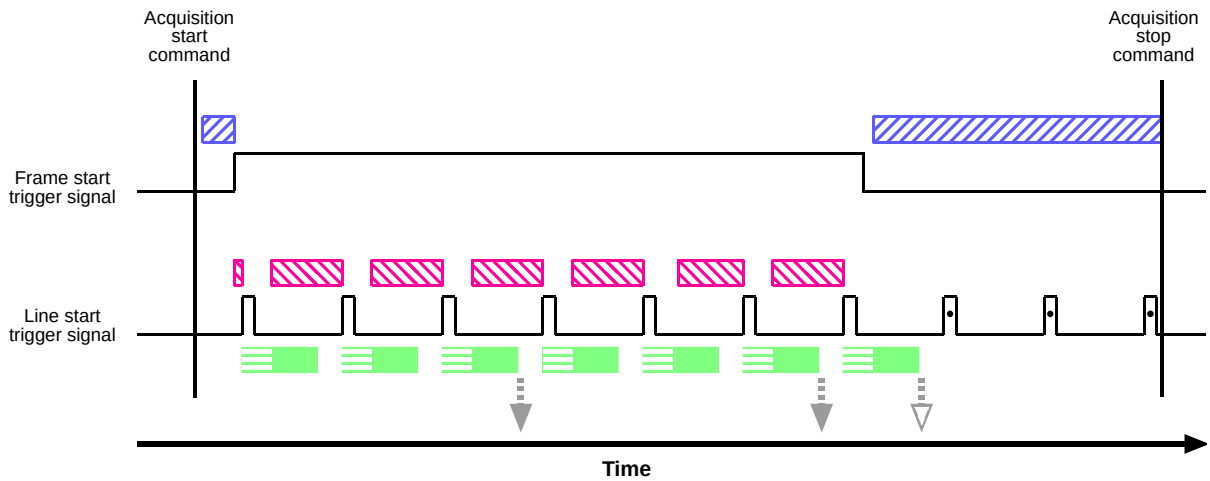


Fig. 36: Use Case 6 - Continuous Frame Mode with Acquisition Start Triggering Set to Off, Frame Start and Line Start Triggering Set to On, and TriggerPartialClosingFrame set to True

Use Case 7 - Acquisition Start and Frame Start Triggering Off (Free Run), Line Start Triggering On

Use case seven is illustrated on page 107.

This use case is equivalent to use case two, except for the fact that the Line Start TriggerMode parameter is set to On.

In this use case, the TriggerMode for the acquisition start trigger and for the frame start trigger is set to Off. The camera will internally manage acquisition start and frame start trigger signals without any need for triggering by the user ("free run").

The TriggerMode parameter for the line start trigger is set to On, requiring that a line start trigger signal is applied to the camera.

The rate at which the camera will acquire lines will be determined by the line start trigger signal and must be below the maximum allowed line rate determined by the current setting.

In this example, each frame is set to include three lines.

When the AcquisitionMode parameter is set to ContinuousFrame, the camera will be set to acquire frames until an acquisition stop command is issued.

If an AcquisitionStop command is issued when not all lines of the current frame are yet acquired, the partial frame will be transmitted. Here, two situations can be distinguished:

- if the AcquisitionStop command is issued during exposure of the second line acquisition, the second line will be dropped and the partial frame will only include its first line.
- if the AcquisitionStop command is issued during readout of the second line acquisition, the readout will be completed and the partial frame will include the related lines one and two.

Settings:
AcquisitionMode = ContinuousFrame

TriggerMode (Acquisition Start) = Off

TriggerMode (Frame Start) = Off
Height (Lines Per Frame) = 3

TriggerMode (Line Start) = On
TriggerSource (Line Start) = Line 3
TriggerActivation (Line Start) = RisingEdge

= trigger signal internally generated by the camera; trigger wait signal is not available

= trigger signal applied by the user

= camera is waiting for a line start trigger signal

= line exposure and readout; white horizontal ruling: period used for exposure and exposure overhead

= complete frame transmitted

= partial frame transmitted

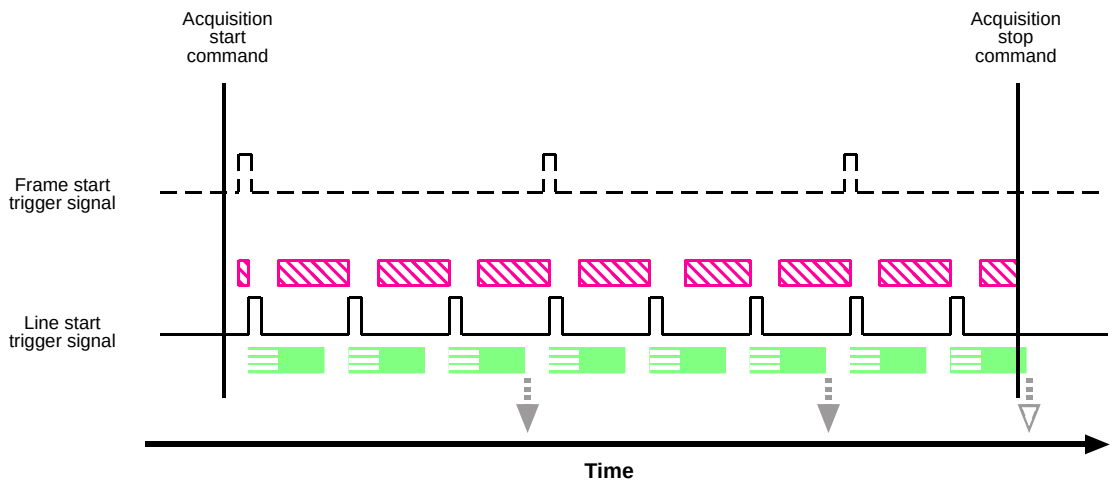


Fig. 37: Use Case 7 - Continuous Frame Mode with Acquisition Start and Frame Start Triggering Set to Off and Line Start Triggering Set to On

Use Case 8 - Acquisition Start Triggering On, Frame Start and Line Start Triggering Off (Free Run)

Use case eight is illustrated on page 109.

In this use case, the TriggerMode for the acquisition start trigger parameter is set to On, requiring that an acquisition start trigger signal is applied to the camera.

The TriggerMode parameter for the frame start trigger and for the line start trigger is set to Off. The camera will internally manage frame start and line start trigger signals without any need for triggering by the user ("free run").

In this example, AcquisitionFrameCount is set to two. Accordingly, two consecutive frames will be acquired for each transition of the acquisition start trigger signal.

The rate at which the camera will acquire lines will normally be determined by the camera's AcquisitionLineRateAbs parameter. If the AcquisitionLineRateAbs parameter is disabled, the camera will acquire lines at the maximum allowed line rate. The overall line rate will also depend on the acquisition start trigger signal: Lines will only be acquired after a related preceding transition of the acquisition start trigger signal has occurred.

In this example, each frame is set to include three lines.

When the AcquisitionMode parameter is set to ContinuousFrame, the camera will be set to acquire frames until an acquisition stop command is issued.

Settings:
AcquisitionMode = ContinuousFrame

TriggerMode (Acquisition Start) = On
TriggerSource (Acquisition Start) = 1
TriggerActivation (Acquisition Start) = RisingEdge
AcquisitionFrameCount = 2

TriggerMode (Frame Start) = Off
Height (Lines Per Frame) = 3

TriggerMode (Line Start) = Off

= trigger signal internally generated by the camera; trigger wait signal is not available

= trigger signal applied by the user

= camera is waiting for an acquisition start trigger signal

= line exposure and readout; white horizontal ruling: period used for exposure and exposure overhead

= frame transmitted

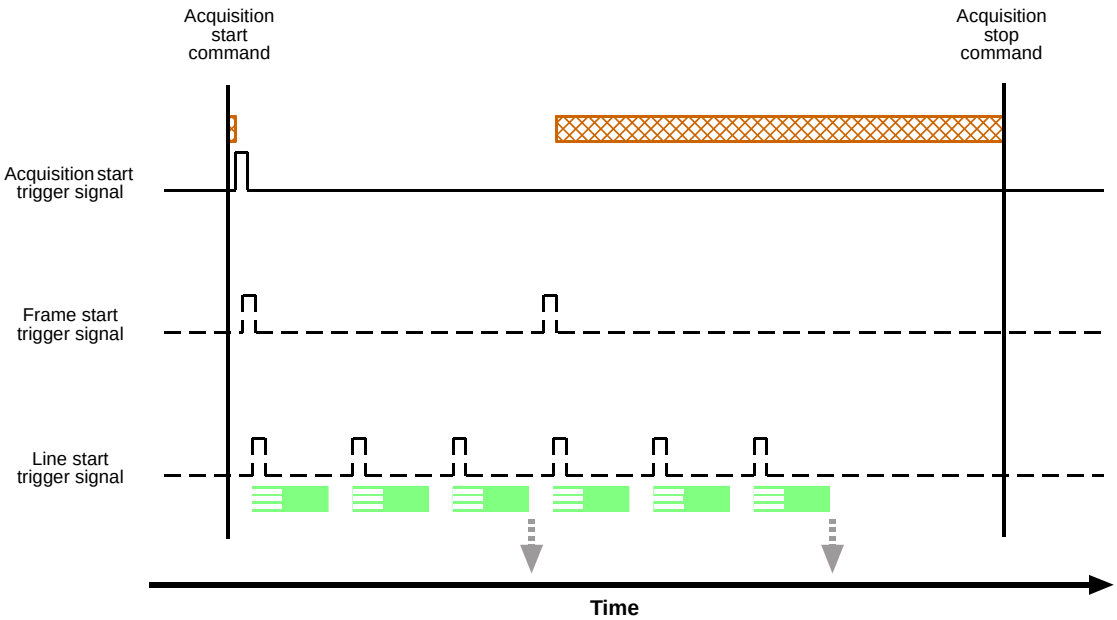


Fig. 38: Use Case 8 - Continuous Frame Mode with Acquisition Start Triggering Set to On and Frame Start and Line Start Triggering Set to Off

Use Case 9 - Acquisition Start and Line Start Triggering On, Frame Start Triggering Off (Free Run)

Use case nine is illustrated on page 111.

In this use case, the TriggerMode parameter for the acquisition start and the line start trigger is set to On, requiring that an acquisition start and a line start trigger signal are applied to the camera.

The TriggerMode parameter for the frame start trigger is set to Off. The camera will internally manage frame start signals without any need for triggering by the user ("free run").

In this example, AcquisitionFrameCount is set to 2. Accordingly, two consecutive frames will be acquired for each transition of the acquisition start trigger signal.

The rate at which the camera will acquire lines will be determined by the line start trigger signal and must be below the maximum allowed line rate determined by the current setting. The overall line rate will also depend on the acquisition start trigger signal: Lines will only be acquired after a related preceding transition of the acquisition start trigger signal has occurred.

In this example, each frame is set to include three lines.

When the AcquisitionMode parameter is set to ContinuousFrame, the camera will be set to acquire frames until an acquisition stop command is issued.

Settings:
AcquisitionMode = ContinuousFrame

TriggerMode (Acquisition Start) = On
TriggerSource (Acquisition Start) = 1
Trigger Activation (Acquisition Start) = RisingEdge
AcquisitionFrameCount = 2

TriggerMode (Frame Start) = Off
Height (Lines Per Frame) = 3

TriggerMode (Line Start) = On
TriggerSource (Line Start) = Line3
TriggerActivation (Line Start) = RisingEdge

= trigger signal internally generated by the camera; trigger wait signal is not available

= trigger signal applied by the user

= camera is waiting for an acquisition start trigger signal

= camera is waiting for a line start trigger signal

= line exposure and readout; white horizontal ruling: period used for exposure and exposure overhead

= line start trigger signal is ignored because the camera is waiting for an acquisition start trigger signal

= complete frame transmitted

= partial frame transmitted

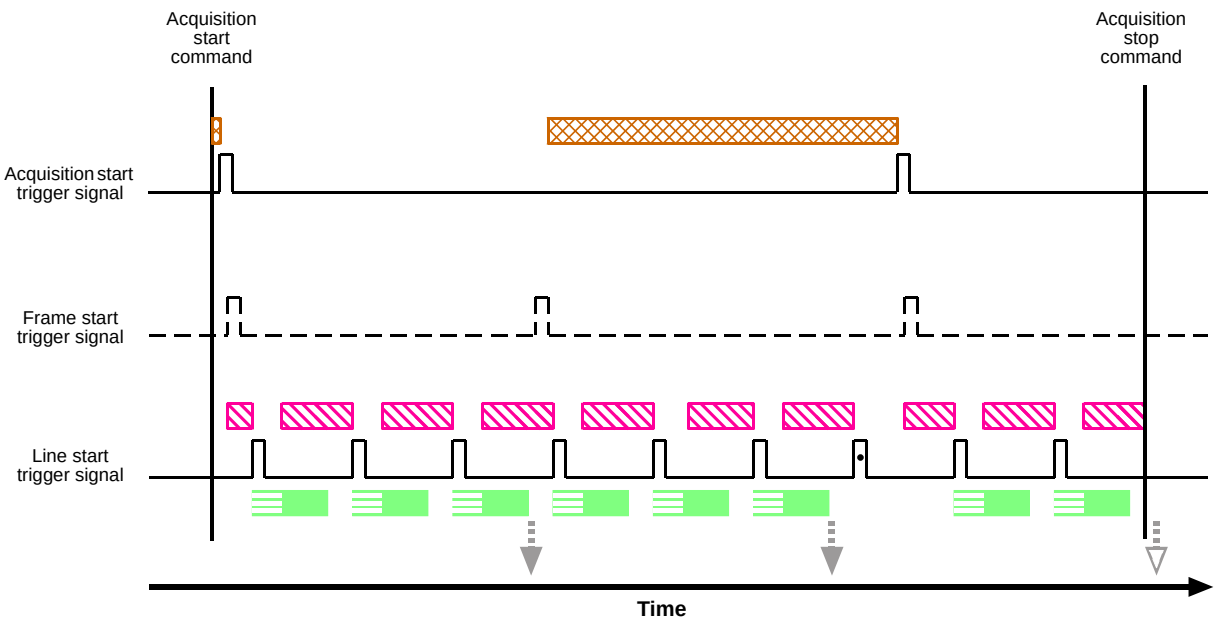


Fig. 39: Use Case 9 - Continuous Frame Mode with Acquisition Start and Line Start Triggering Set to On and Frame Start Triggering Set to Off

8.2.7 Overlapping Exposure with Sensor Readout

The line acquisition process on the camera includes two distinct parts. The first part is the exposure of the pixels in the imaging sensor. Once exposure is complete, the second part of the process – readout of the pixel values from the sensor – takes place. In regard to this line acquisition process, there are two common ways for the camera to operate: with “non-overlapped” exposure and with “overlapped” exposure.

In the non-overlapped mode of operation, each time a line is acquired the camera completes the entire exposure/readout process before acquisition of the next line is started. The exposure for a new line does not overlap the sensor readout for the previous line. This situation is illustrated in Fig. 40 with the camera set for the trigger width exposure mode.

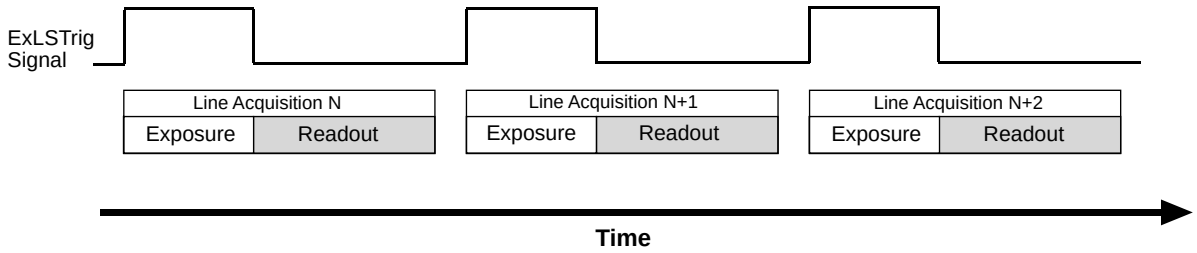


Fig. 40: Non-overlapped Exposure and Readout

In the overlapped mode of operation, the exposure of a new line begins while the camera is still reading out the sensor data for the previously acquired line. This situation is illustrated in Fig. 41 with the camera set for the trigger width exposure mode.

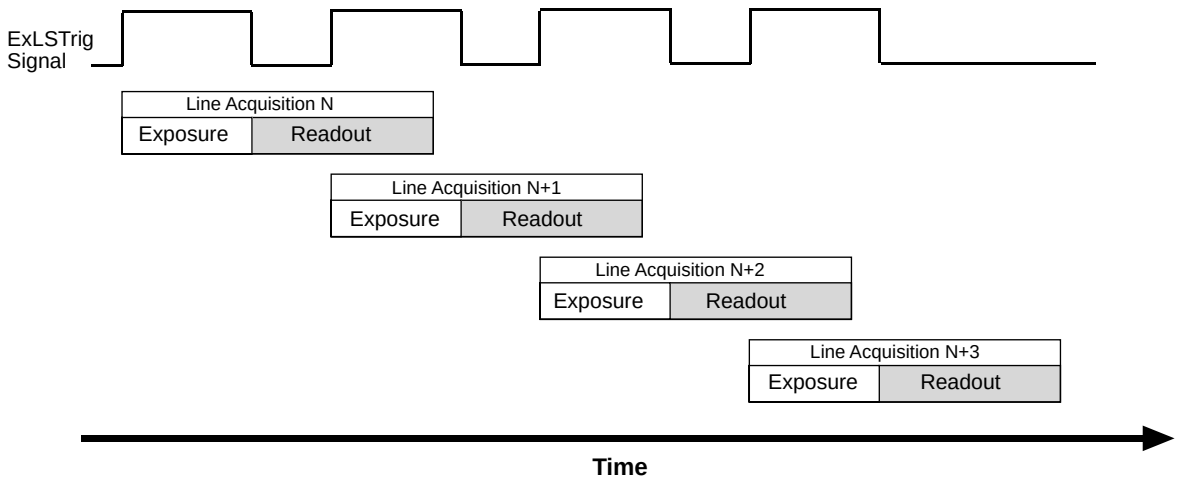


Fig. 41: Overlapped Exposure and Readout

Determining whether your camera is operating with overlapped or non-overlapped exposure and readout is not a matter of issuing a command or switching a setting on or off. Rather the way that you operate the camera will determine whether the exposures and readouts are overlapped or not. If we define the “line period” as the time from the start of exposure for one line acquisition to the start of exposure for the next line acquisition, then:

- Exposure will not overlap when: Line Period > Exposure Time + Readout Time
- Exposure will overlap when: Line Period ≤ Exposure Time + Readout Time

You can determine the readout time by reading the value of the ReadoutTimeAbs parameter. The parameter indicates what the readout time will be in microseconds given the camera’s current settings. You can read the ReadoutTimeAbs parameter value from within your application software by using the Basler pylon API. The following code snippet illustrates using the API to get the parameter value:

```
double ReadoutTime = camera.ReadoutTimeAbs.GetValue();
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer’s Guide and API Reference.

You can also use the Basler pylon Viewer application to easily get the parameter value.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

8.2.7.1 Guidelines for Overlapped Operation

To ensure smooth line acquisition and avoid overtriggering, you may only apply a line acquisition-related trigger when the camera is waiting for it. If the trigger is nonetheless applied, the trigger will be ignored and considered an overtrigger.

The risk of overtriggering exists particularly for overlapped operation where the sequence of line start triggers must be carefully coordinated both with the camera’s exposure time and the sensor readout time.

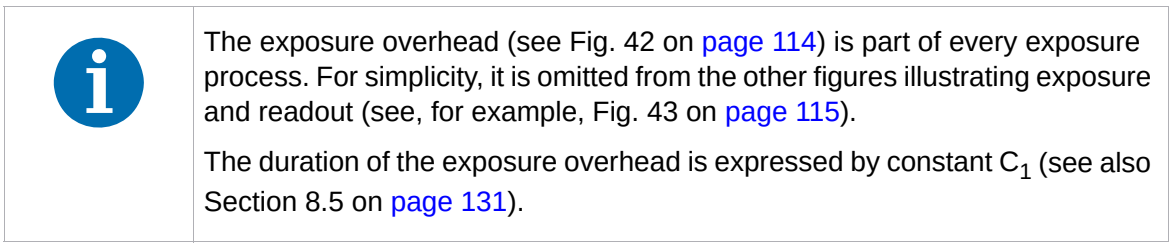
The following examples use a non-inverted, rising edge external line start trigger signal (ExLSTrig).

Illegal Line Acquisition by Overtriggering

Certain attempts of triggering overlapped line acquisition are illegal and do not result in line acquisitions: When a line start trigger signal attempts an illegal line acquisition the trigger signal will be ignored and, accordingly, no line acquisition will be performed. In addition, the trigger signal will be reported as an overtrigger (see also Section 10.15 on page 209). Illegal triggering and impossible overlaps are shown in Fig. 42 on page 114 and Fig. 43 on page 115.

Illegal triggering when overlapping line acquisitions:

- The line start trigger goes high to start the exposure for line acquisition N+1 before the exposure or the exposure overhead for line acquisition N has ended (see Fig. 42 on page 114). This would result in the illegal overlap of exposures or of exposure and exposure overhead.



The exposure overhead (see Fig. 42 on page 114) is part of every exposure process. For simplicity, it is omitted from the other figures illustrating exposure and readout (see, for example, Fig. 43 on page 115).

The duration of the exposure overhead is expressed by constant C_1 (see also Section 8.5 on page 131).

- The line start trigger goes low to end the exposure for line acquisition N+1 before readout for acquisition N has ended (premature exposure end; see Fig. 43 on page 115).
  This would result in the illegal overlap of two readouts (in trigger width exposure mode only).

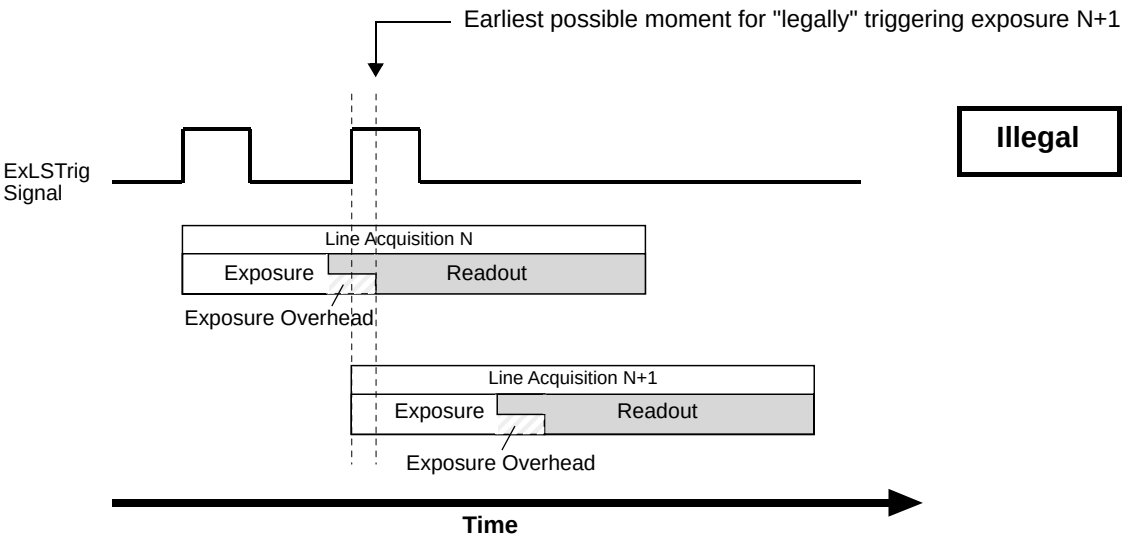


Fig. 42: Exposure N+1 Illegally Starts Before the Exposure Overhead for the Preceding Line Acquisition N Has Ended; the Shown Overlap of Readouts is Also Illegal; Timed Exposure Mode Used as an Example

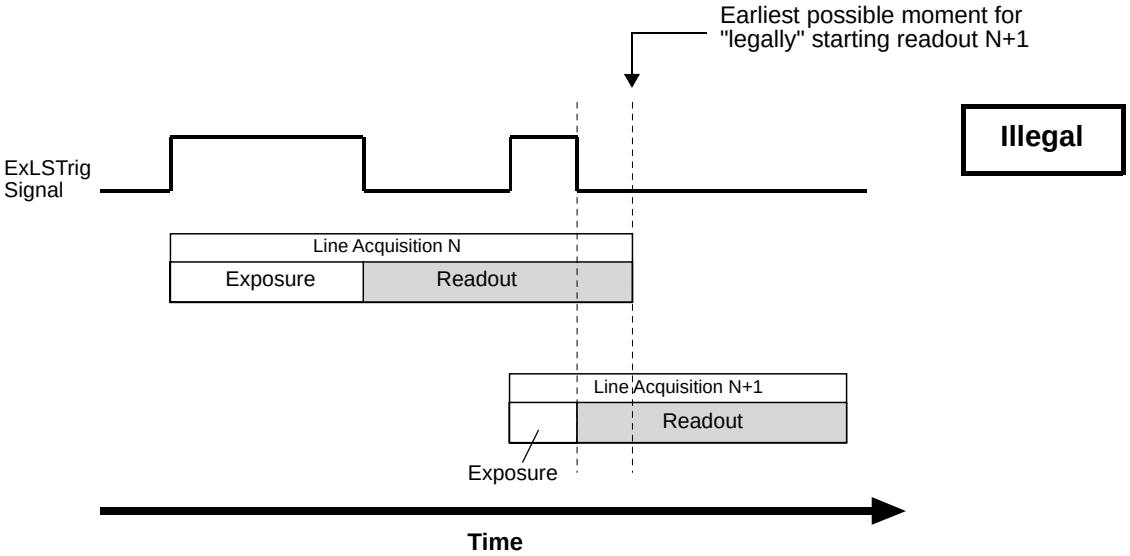


Fig. 43: Exposure N+1 Illegally Ends Before Readout of the Preceding Line Acquisition N Has Ended; Applies to Trigger Width Exposure Mode Only

When the line start trigger has illegally gone low to end the exposure for line acquisition N+1 before readout for acquisition N has ended (in trigger width exposure mode; see Fig. 43), the camera will behave as shown in Fig. 44: The camera will extend the exposure and end it when the next valid trigger for ending exposure occurs.

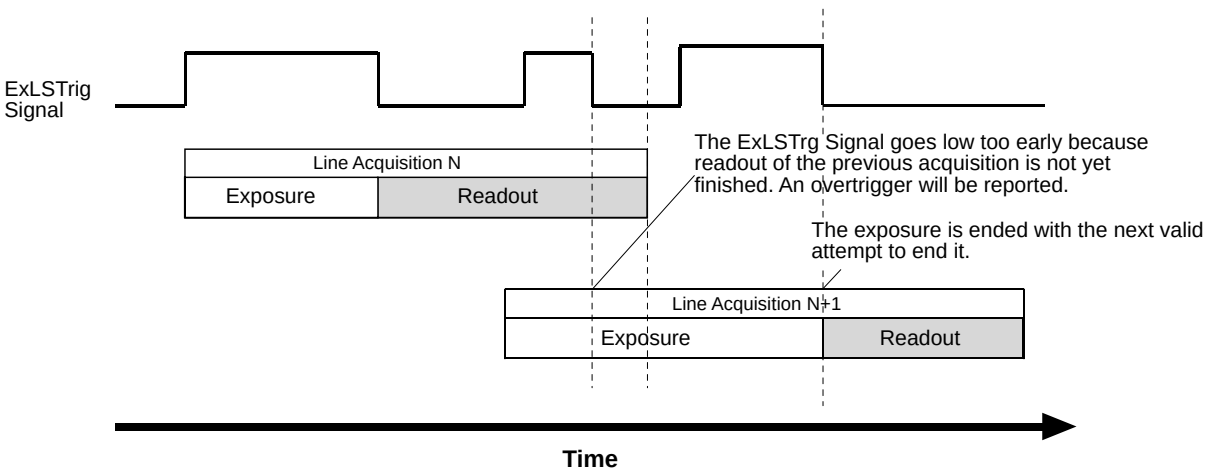


Fig. 44: Extension of Exposure N+1 After the Illegal Attempt of Ending It Too Early; Applies to Trigger Width Exposure Mode Only

Regular Line Acquisition Avoiding Overtriggering

As mentioned above, you can avoid overtriggering by applying an acquisition-related trigger **only** when the camera is waiting for it.

You can achieve this goal by

- making use of acquisition monitoring tools, i.e. monitoring the camera's acquisition status and triggering only when the camera indicates that it is in the "waiting status" for the trigger or by
- strictly obeying timing limits for exposure and triggering.

Using Acquisition Monitoring Tools

To get informed whether the camera is waiting for a trigger you can use the acquisition monitoring tools described in Section 8.3 on page 119.

By applying an ExLSTrig signal as soon as the camera indicates that it is waiting for an ExLSTrig signal you can operate the camera in "overlapped mode" wihout overtriggering.

As an example, and in the context of overlapped exposure, the use of the line trigger wait signal is described in Section 8.3.3.3 on page 126 for proper triggering with the line start trigger. Both timed and trigger width exposure mode are considered.

Line Acquisition While Obeying Timing Limits

When strictly obeying the following timing limits you can avoid overtriggering in "overlapped mode" and "non-overlapped mode" without having to monitor the camera's acquisition status (see also Fig. 45 and Fig. 46 below). You must ensure that the following four conditions are fulfilled **at the same time**:

- **Condition one:** The exposure time E is ≥ 2 µs.

  This is the minimum allowed exposure time also given in Section 8.2.5 on page 91.
- **Condition two:** Period F is ≥ C_1 (for C_1 values, see Section 8.5 on page 131); period F follows immediately after the exposure.

  The constant C_1 expresses the duration of exposure overhead. Exposure is not possible during this period. Its default value equals 5.4 µs. However, when the parameter limit is removed from the ExposureOverhead parameter (see Section 8.5.1 on page 134), C_1 equals 3.4 µs. Accordingly, F must be ≥ 5.4 µs in the general case, and ≥ 3.4 µs when the parameter limit is removed from the ExposureOverhead parameter.
- **Condition three:** ExLSTrig Signal Period ≥ Minimum allowed line period.
- **Condition four:** Make sure the ExposureOverlapTimeMaxAbs parameter value is set to the appropriate value:

  appropriate ExposureOverlapTimeMaxAbs parameter value =
  = minimum allowed line period - duration of exposure overhead (i.e. C_1,
  i.e. minimum F value).

  The maximum allowed exposure time E that is compatible with the maximum line rate is equal to the appropriate ExposureOverlapTimeMaxAbs parameter value.

It follows that

- if you increase the ExposureOverlapTimeMaxAbs parameter value above its appropriate value while maintaining the trigger signal period, the minimum value for the duration of the exposure overhead would be ignored and trigger signals can be considered overtriggers when in fact they are not.
- if you decrease the ExposureOverlapTimeMaxAbs parameter value below its appropriate value you would have to increase the duration of the exposure overhead and thereby increase the line rate.

	When you want to operate the camera at the maximum allowed line rate and have set the ExposureOverlapTimeMaxAbs parameter value to the appropriate value (see above) you can vary the exposure time E within a range of values between 2 µs and the applicable maximum allowed exposure time E (see condition four).

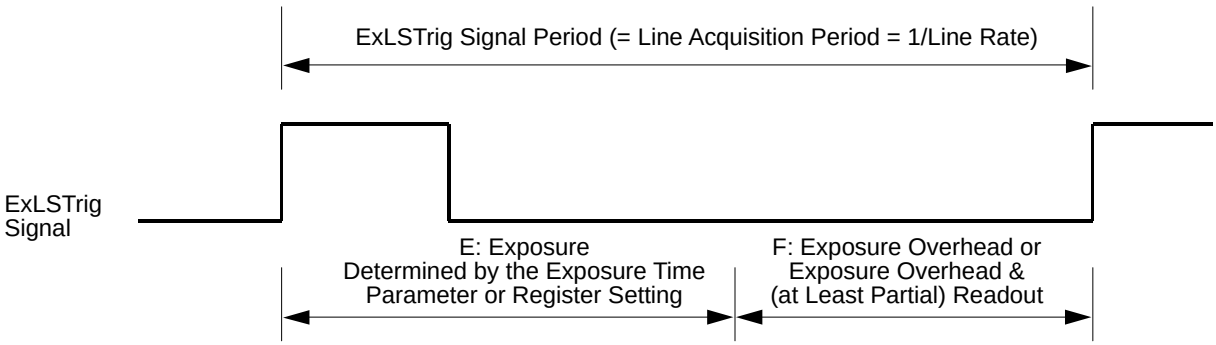


Fig. 45: Relation of the ExLSTrig Signal Period and Periods E and F for Regular Line Acquisition in Timed Exposure Mode

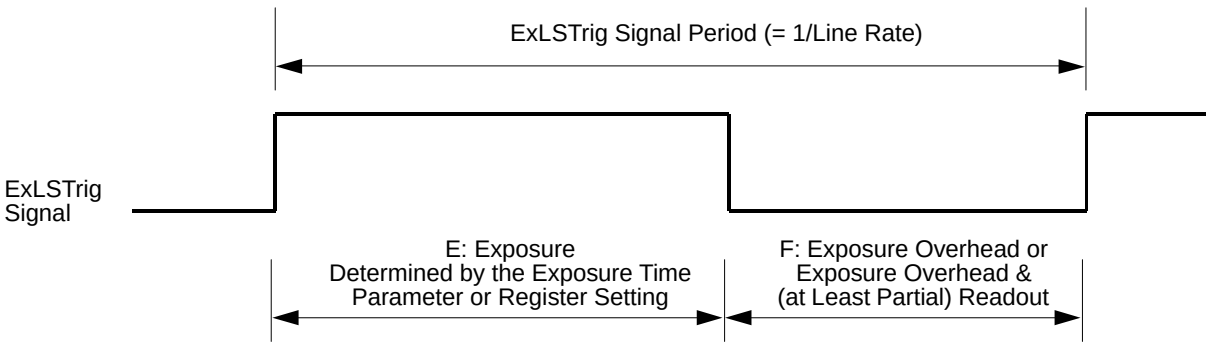


Fig. 46: Relation of the ExLSTrig Signal Period and Periods E and F for Regular Line Acquisition in Trigger Width Exposure Mode

From the above conditions, one can readily calculate the allowed values for E and F for regularly operating the camera at the maximum allowed line rate. This operation will also involve the maximum possible overlap between consecutive line acquisitions.

Example

Assume that you are using an raL2048-48gm camera at full resolution (2048 pixels), assume that you want to use the minimum allowed line acquisition period and the default value for C_1.

Also assume that the other relevant settings are in accord with operation at the minimum allowed line acquisition period.

At full resolution, the camera is capable of a minimum line acquisition period of 19.7 µs (corresponding to a maximum allowed line acquisition rate of approximately 51 kHz; see Section 1.2 on page 2).

Accordingly, the corresponding minimum allowed ExLSTrig signal period where overtriggering is avoided is 19.7 µs.

From condition number four follows the maximum possible exposure time that is compatible with (default) maximum overlap:

E = min. ExLSTrig Signal Period - C_1 = 19.7 µs - 5.4 µs = 14.3 µs

Therefore, when operating the camera at a line acquisition period of 19.7 µs (involving maximum overlap) and using the default value for C_1, the maximum possible exposure time is 14.3 µs. When also considering the above condition number one, it follows that the exposure time can range between 2 µs and 14.3 µs to remain in accord with camera operation at a line acquisition period of 19.7 µs (and a line rate of approximately 51 kHz).

If you increase an exposure time beyond its upper limits the related extent of overlap and the acquisition line rate will decrease. When extending exposure time even further, consecutive line acquisitions will eventually not overlap at all.

8.3 Acquisition Monitoring Tools

The camera includes the acquisition status feature and generates four output signals that you can use to monitor the progress of line and frame acquisition by the camera: the exposure active signal, the acquisition trigger wait signal, the frame trigger wait signal, and the line trigger wait signal.

These signals are designed to be used when you are triggering acquisition start, frame start or line start via a hardware trigger signal.

The camera will ignore a hardware trigger signal when it is not waiting for it:

- If a camera receives a hardware acquisition start trigger signal when it is not in a "waiting for acquisition start trigger" acquisition status, it will simply ignore the trigger signal and will generate an acquisition start overtrigger event.
- If a camera receives a hardware frame start trigger signal when it is not in a "waiting for frame start trigger" acquisition status, it will simply ignore the trigger signal and will generate a frame start overtrigger event.
- If a camera receives a hardware line start trigger signal when it is not in a "waiting for line start trigger" acquisition status, it will simply ignore the trigger signal and will generate a line start overtrigger event.

The camera’s trigger wait signals give you the ability to check whether the camera is waiting for the related hardware trigger signal:

- The acquisition trigger wait signal gives you the ability to check whether the camera is in a "waiting for acquisition start trigger" acquisition status. If you check the acquisition trigger wait signal before you apply each hardware acquisition start trigger signal, you can therefore avoid applying acquisition start trigger signals to the camera that will be ignored.
- The frame trigger wait signal gives you the ability to check whether the camera is in a "waiting for frame start trigger" acquisition status. If you check the frame trigger wait signal before you apply each hardware frame start trigger signal, you can avoid applying frame start trigger signals to the camera that will be ignored.
- The line trigger wait signal gives you the ability to check whether the camera is in a "waiting for line start trigger" acquisition status. If you check the line trigger wait signal before you apply each hardware line start trigger signal, you can avoid applying line start trigger signals to the camera that will be ignored.

For more information about the trigger wait signals, see Section 8.3.3 on page 122.

8.3.1 Exposure Active Signal

The camera's Exposure Active output signal will go high when the exposure time for each line acquisition begins and goes low when the exposure time ends. An example of the Exposure Active signal's behavior on a camera using a rising edge external line start trigger signal (ExLSTrig) and the timed exposure mode is shown in Fig. 47.

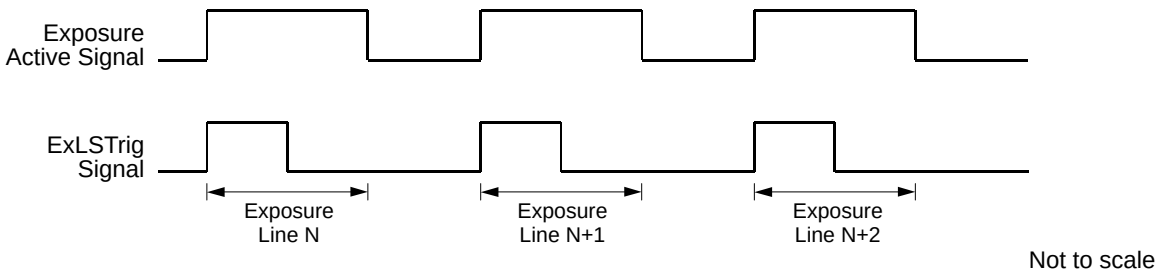


Fig. 47: Exposure Active Signal

By default, the Exposure Active signal is selected as the source signal for output line 1 on the camera. However, the selection of the source signal for a physical output line can be changed.

For more information about selecting the source signal for an output line on the camera, see Section 7.6.2.5 on page 68.

For more information about the electrical characteristics of the camera's output lines, see Section 7.6.2.1 on page 64.

8.3.2 Acquisition Status Indicator

If a camera receives a software acquisition start trigger signal when it is not in a "waiting for acquisition start trigger" acquisition status, it will simply ignore the trigger signal and will generate an acquisition start overtrigger event.

If a camera receives a software frame start trigger signal when it is not in a "waiting for frame start trigger" acquisition status, it will simply ignore the trigger signal and will generate a frame start overtrigger event.

If a camera receives a software line start trigger signal when it is not in a "waiting for line start trigger" acquisition status, it will simply ignore the trigger signal and will generate a line start overtrigger event.

The camera's acquisition status indicator gives you the ability to check whether the camera is in a "waiting for acquisition start trigger" acquisition status or in a "waiting for frame start trigger" acquisition status or in a "waiting for line start trigger" acquisition status. If you check the acquisition status before you apply each software acquisition start trigger signal, software frame start trigger signal and software line start trigger signal, you can avoid applying trigger signals to the camera that will be ignored.

The acquisition status indicator is designed for use when you are using host control of image acquisition, i.e., when you are using software acquisition start, frame start, and line start trigger signals.

To determine the acquisition status of the camera using the Basler pylon API:

1. Use the AcquisitionStatusSelector parameter to select the AcquisitionTriggerWait status or the FrameTriggerWait status or the LineTriggerWait status.
2. Read the value of the AcquisitionStatus parameter.

   If the value is set to False, the camera is not waiting for the trigger signal.

   If the value is set to True, the camera is waiting for the trigger signal.

You can check the acquisition status from within your application software by using the Basler pylon API. The following code snippet illustrates using the API to check the acquisition status:

```
// Check the acquisition start trigger acquisition status
// Set the acquisition status selector
camera.AcquisitionStatusSelector.SetValue
(AcquisitionStatusSelector_AcquisitionTriggerWait);
// Read the acquisition status
bool IsWaitingForAcquisitionTrigger = camera.AcquisitionStatus.GetValue();

// Check the frame start trigger acquisition status
// Set the acquisition status selector
camera.AcquisitionStatusSelector.SetValue
(AcquisitionStatusSelector_FrameTriggerWait);
// Read the acquisition status
bool IsWaitingForFrameTrigger = camera.AcquisitionStatus.GetValue();

// Check the line start trigger acquisition status
// Set the acquisition status selector
camera.AcquisitionStatusSelector.SetValue
(AcquisitionStatusSelector_LineTriggerWait);
// Read the acquisition status
bool IsWaitingForLineTrigger = camera.AcquisitionStatus.GetValue();
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer’s Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

8.3.3 Trigger Wait Signals

The trigger wait signals are designed to be used when you are triggering acquisition start, frame start or line start via a hardware trigger signal.

The camera will ignore a hardware trigger signal when it is not waiting for it:

- If a camera receives a hardware acquisition start trigger signal when it is not in a "waiting for acquisition start trigger" acquisition status, it will simply ignore the trigger signal and will generate an acquisition start overtrigger event.
- If a camera receives a hardware frame start trigger signal when it is not in a "waiting for frame start trigger" acquisition status, it will simply ignore the trigger signal and will generate a frame start overtrigger event.
- If a camera receives a hardware line start trigger signal when it is not in a "waiting for line start trigger" acquisition status, it will simply ignore the trigger signal and will generate a line start overtrigger event.

The camera’s trigger wait signals give you the ability to check whether the camera is waiting for the related hardware trigger signal:

- The acquisition trigger wait signal gives you the ability to check whether the camera is in a "waiting for acquisition start trigger" acquisition status. If you check the acquisition trigger wait signal before you apply each hardware acquisition start trigger signal, you can therefore avoid applying acquisition start trigger signals to the camera that will be ignored.
- The frame trigger wait signal gives you the ability to check whether the camera is in a "waiting for frame start trigger" acquisition status. If you check the frame trigger wait signal before you apply each hardware frame start trigger signal, you can avoid applying frame start trigger signals to the camera that will be ignored.
- The line trigger wait signal gives you the ability to check whether the camera is in a "waiting for line start trigger" acquisition status. If you check the line trigger wait signal before you apply each hardware line start trigger signal, you can avoid applying line start trigger signals to the camera that will be ignored.

8.3.3.1 Acquisition Trigger Wait Signal

As you are acquiring frames, the camera automatically monitors the acquisition start trigger status and supplies a signal that indicates the current status.

The Acquisition Trigger Wait signal will go high whenever the camera enters a "waiting for acquisition start trigger" status. The signal will go low when an external acquisition start trigger (ExASTrig) signal is applied to the camera and the camera exits the "waiting for acquisition start trigger status". The signal will go high again when the camera again enters a "waiting for acquisition trigger" status and it is safe to apply the next acquisition start trigger signal.

If you base your use of the ExASTrig signal on the state of the acquisition trigger wait signal, you can avoid "acquisition start overtriggering", i.e., applying an acquisition start trigger signal to the camera when it is not in a "waiting for acquisition start trigger" acquisition status. If you do apply an acquisition start trigger signal to the camera when it is not ready to receive the signal, it will be ignored and an acquisition start overtrigger event will be reported.

Fig. 48 illustrates the Acquisition Trigger Wait signal with the AcquisitionFrameCount parameter set to 2, with the lines per frame (Height) set to 3, and with exposure and readout overlapped. The figure assumes raising edge triggering and that the trigger mode for the frame start trigger and for the line start trigger is set to Off, so the camera is internally generating frame and line start trigger signals.

The acquisition trigger wait signal can be selected as the source signal for one of the output lines on the camera.

For more information about selecting the source signal for an output line on the camera, see Section 7.6.2.5 on page 68.

For more information about the electrical characteristics of the camera's output lines, see Section 7.6.2.1 on page 64.

For more information about event reporting, see Section 10.5 on page 175.

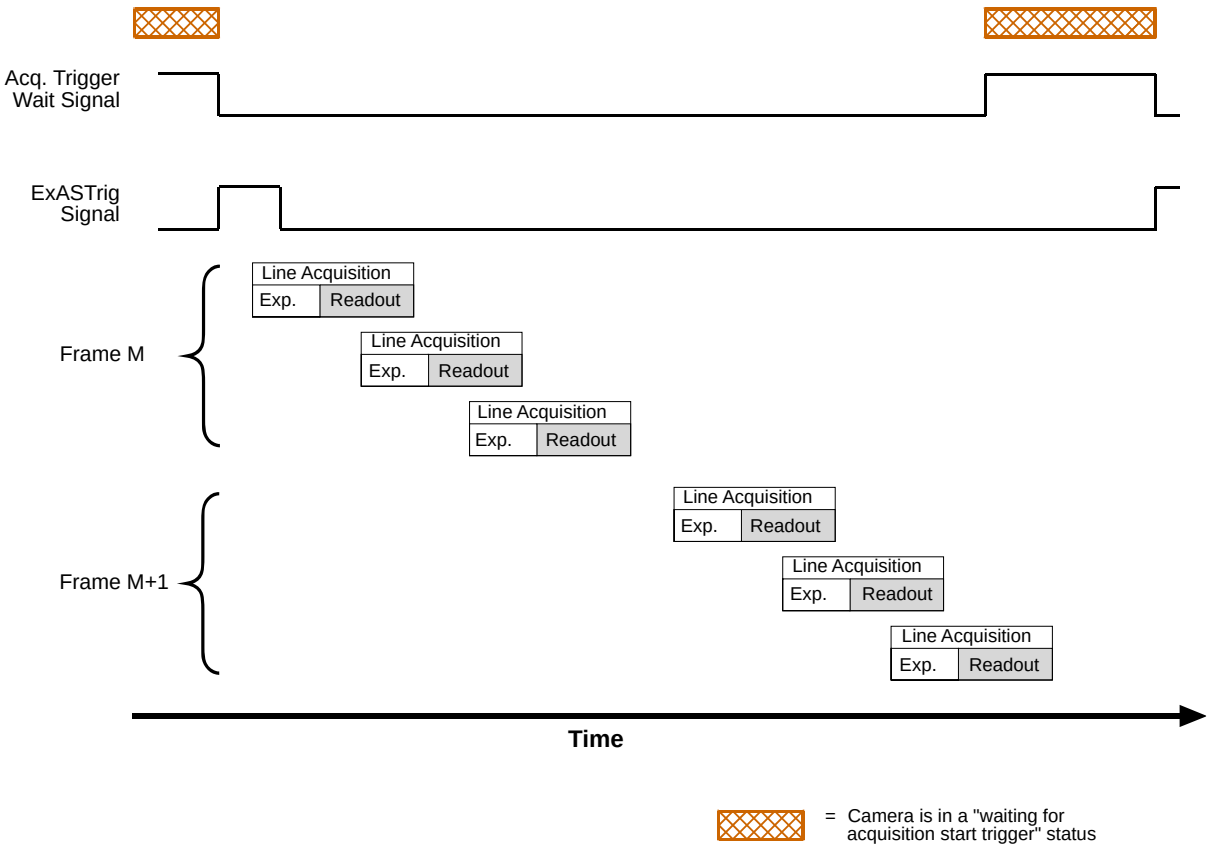


Fig. 48: Acquisition Trigger Wait Signal

The acquisition trigger wait signal will only be available when hardware acquisition start triggering is enabled.

Selecting the Acquisition Trigger Wait Signal as the Source Signal for the Output Line

The acquisition trigger wait signal can be selected to act as the source signal for e.g. camera output line 1. Selecting a source signal for the output line is a two step process:

- Use the LineSelector parameter to select output line 1.
- Set the value of the LineSource parameter to the acquisition trigger wait signal.

You can set the LineSelector and the LineSource parameter values from within your application software by using the Basler pylon API. The following code snippet illustrates using the API to set the parameter values:

```
camera.LineSelector.SetValue(LineSelector_Out1);
camera.LineSource.SetValue(LineSource_AcquisitionTriggerWait);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

For more information about selecting the source signal for an output line on the camera, see Section 7.6.2.5 on page 68.

For more information about the electrical characteristics of the camera's output line, see Section 7.6.2.1 on page 64.

8.3.3.2 Frame Trigger Wait Signal

As you are acquiring frames, the camera automatically monitors the frame start trigger status and supplies a signal that indicates the current status.

The Frame Trigger Wait signal will go high whenever the camera enters a "waiting for frame start trigger" status. The signal will go low when an external frame start trigger (ExFSTrig) signal is applied to the camera and the camera exits the "waiting for frame start trigger status". The signal will go high again when the camera again enters a "waiting for frame trigger" status and it is safe to apply the next frame start trigger signal.

If you base your use of the ExFSTrig signal on the state of the frame trigger wait signal, you can avoid "frame start overtriggering", i.e., applying a frame start trigger signal to the camera when it is not in a "waiting for frame start trigger" acquisition status. If you do apply a frame start trigger signal to the camera when it is not ready to receive the signal, it will be ignored and a frame start overtrigger event will be reported.

Fig. 49 illustrates the Frame Trigger Wait signal with the lines per frame (Height) set to 3, and with exposure and readout overlapped. The figure assumes raising edge triggering and that the trigger mode for the acquisition start trigger and for the line start trigger is set to Off, so the camera is internally generating acquisition and line start trigger signals.

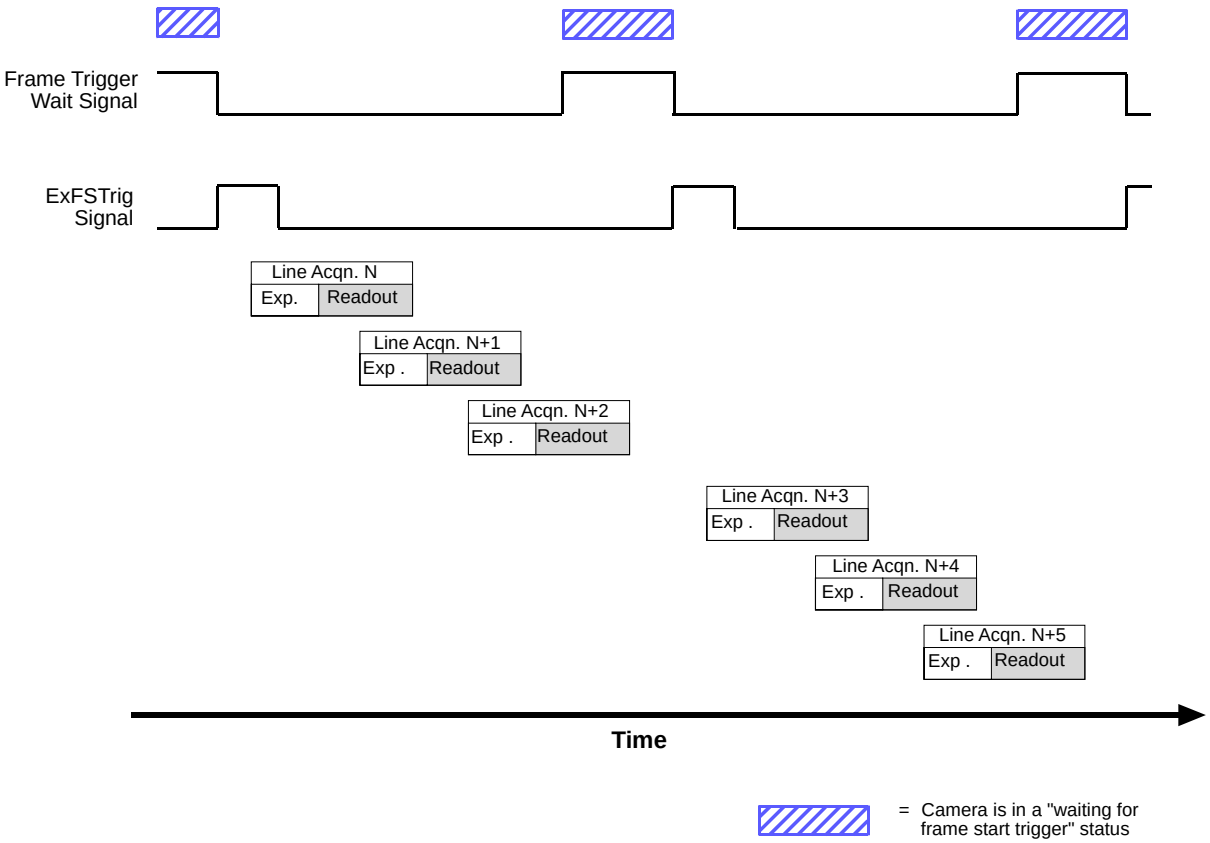


Fig. 49: Frame Trigger Wait Signal

The frame trigger wait signal will only be available when hardware frame start triggering is enabled.

By default, the frame trigger wait signal is selected as the source signal for output line 2 on the camera. However, the selection of the source signal for a physical output line can be changed.

Selecting the Frame Trigger Wait Signal as the Source Signal for the Output Line

The frame trigger wait signal can be selected to act as the source signal for e.g. camera output line 1. Selecting a source signal for the output line is a two step process:

- Use the LineSelector parameter to select output line 1.
- Set the value of the LineSource parameter to the frame trigger wait signal.

You can set the LineSelector and the LineSource parameter values from within your application software by using the Basler pylon API. The following code snippet illustrates using the API to set the parameter values:

```
camera.LineSelector.SetValue(LineSelector_Out1);
camera.LineSource.SetValue(LineSource_FrameTriggerWait);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

For more information about selecting the source signal for an output line on the camera, see Section 7.6.2.5 on page 68.

For more information about the electrical characteristics of the camera's output lines, see Section 7.6.2.1 on page 64.

For more information about event reporting, see Section 10.5 on page 175.

For more information about hardware triggering, see Section 8.2 on page 77.

8.3.3.3 Line Trigger Wait Signal

As you are acquiring lines, the camera automatically monitors the line start trigger status and supplies a signal that indicates the current status.

The Line Trigger Wait signall will go high whenever the camera enters a "waiting for line start trigger" status. The signal will go low when an external line start trigger (ExLSTrig) signal is applied to the camera and the camera exits the "waiting for line start trigger status". The signal will go high again when the camera again enters a "waiting for line trigger" status and it is safe to apply the next line start trigger signal.

If you base your use of the ExLSTrig signal on the state of the line trigger wait signal, you can avoid "line start overtriggering", i.e., applying a line start trigger signal to the camera when it is not in a "waiting for line start trigger" acquisition status. If you do apply a line start trigger signal to the camera when it is not ready to receive the signal, it will be ignored and a line start overtrigger event will be reported.

The line trigger wait signal can be selected as the source signal for one of the output lines on the camera.

For more information about selecting the source signal for an output line on the camera, see Section 7.6.2.5 on page 68.

For more information about the electrical characteristics of the camera's output lines, see Section 7.6.2.1 on page 64.

Fig. 49 and Fig. 51 illustrate the Frame Trigger Wait signal with exposure and readout overlapped. The figures assume raising edge triggering and that the trigger mode for the acquisition start trigger and for the frame start trigger is set to Off, so the camera is internally generating acquisition and frame start trigger signals.

Using the Line Trigger Wait Signal with the Timed Exposure Mode

When the camera is set for the timed exposure mode, the rise of the Line Trigger Wait signal is based on the current ExposureTimeAbs parameter setting and on when readout of the current line will end. This functionality is illustrated in Fig. 50.

If you are operating the camera in the timed exposure mode, you can avoid overtriggering by always making sure that the Line Trigger Wait signal is high before you trigger the start of line capture.

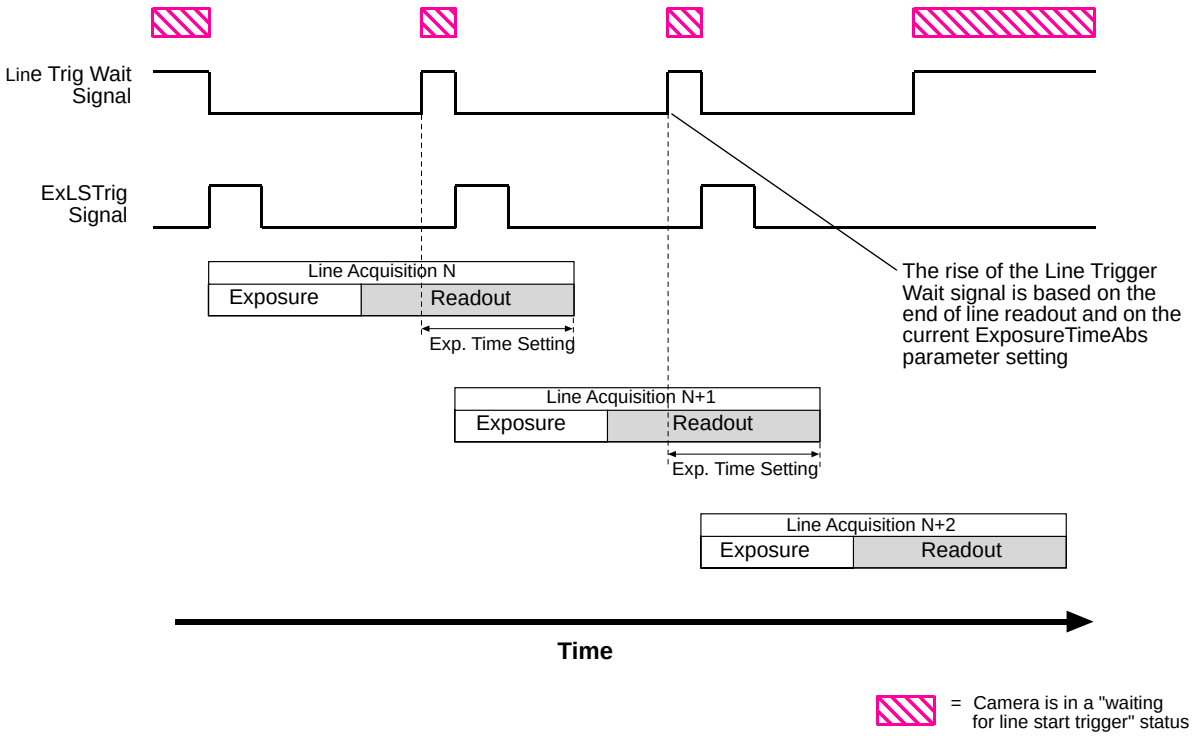


Fig. 50: Line Trigger Wait Signal with the Timed Exposure Mode

Using the Line Trigger Wait Signal with the Trigger Width Exposure Mode

When the camera is set for the trigger width exposure mode, the rise of the Line Trigger Wait signal is based on the ExposureOverlapTimeMaxAbs parameter setting and on when readout of the current line will end. This functionality is illustrated in Fig. 51.

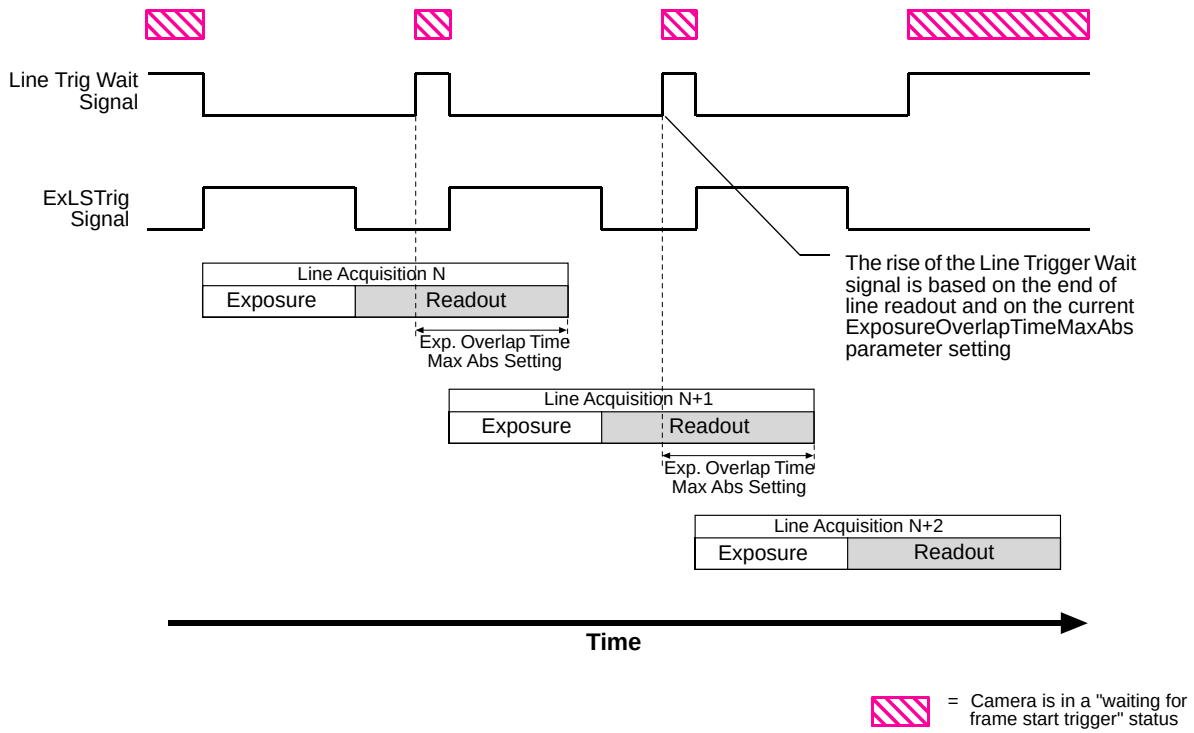


Fig. 51: Line Trigger Wait Signal with the Trigger Width Exposure Mode

If you are operating the camera in the trigger width exposure mode and monitor the Line Trigger Wait signal, you can avoid overtriggering the camera by always doing the following:

- Setting the camera's ExposureOverlapTimeMaxAbs parameter so that it represents the shortest exposure time you intend to use. If the shortest intended exposure time is larger than the maximum settable ExposureOverlapTimeMaxAbs parameter value, set the ExposureOverlapTimeMaxAbs parameter value to its maximum.
- Making sure that your exposure time is always equal to or greater than the setting for the ExposureOverlapTimeMaxAbs parameter.
- Only use the ExLSTrig signal to start exposure when the Line Trigger Wait signal is high.

Setting the ExposureOverlapTimeMaxAbs Parameter

You can use the Basler pylon API to set the ExposureOverlapTimeMaxAbs parameter value from within your application software. The following code snippet illustrates using the API to set the parameter value:

```
// Set the Exposure Overlap Time Max to 4 µs
camera.ExposureOverlapTimeMaxAbs.SetValue(4);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

Selecting the Line Trigger Wait Signal as the Source Signal for the Output Line

The line trigger wait signal can be selected to act as the source signal for e.g. camera output line 1. Selecting a source signal for the output line is a two step process:

- Use the LineSelector parameter to select output line 1.
- Set the value of the LineSource parameter to the line trigger wait signal.

You can set the LineSelector and the LineSource parameter values from within your application software by using the Basler pylon API. The following code snippet illustrates using the API to set the parameter values:

```
camera.LineSelector.SetValue(LineSelector_Out1);
camera.LineSource.SetValue(LineSource_LineTriggerWait);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

For more information about selecting the source signal for an output line on the camera, see Section 7.6.2.5 on page 68.

For more information about the electrical characteristics of the camera's output line, see Section 7.6.2 on page 64.

8.4 Frame Transmission Time

As mentioned in earlier sections of this chapter, each time that a complete frame has been accumulated in the camera's frame memory, the frame will be transmitted from the camera to your host PC via the camera's Ethernet network connection. The image data in the frame will be packetized and transmitted in compliance with the mechanisms described in the GigE Vision standard.

For more detailed information about receiving the frames as they arrive in your host PC, refer to the Basler pylon Programmer's Guide and API Reference. The sample programs included with the pylon software development kit (SDK) also provide more detailed information about handling incoming image data in your host PC.

For more information about managing the bandwidth of the Ethernet network connection between your camera(s) and your host PC, see Section 5 on page 39.

You can calculate the approximate time that it will take to transmit a frame from the camera to the host PC by using this formula:

$$\sim \text{Frame Transmission Time} = \frac{\text{Payload Size Parameter Value}}{\text{Device Current Throughput Parameter Value}}$$

This is an approximate frame transmission time. Due to the nature of the Ethernet network, the transmission time could vary.

Due to the nature of the Ethernet network, there can be a delay between the point where a complete frame is acquired and the point where transmission of the acquired frame begins. This start delay can vary from frame to frame. The start delay, however, is of very low significance when compared to the transmission time.

For more information about the Payload Size and Device Current Throughput parameters, see Section 5.1 on page 39.

8.5 Maximum Allowed Line Acquisition Rate

In general, the maximum allowed line acquisition rate can be limited by three factors:

- The amount of time it takes to read an acquired line out of the imaging sensor and into the camera's frame buffer. Since readout time is fixed, it establishes an absolute maximum for the line rate. The readout time stays the same regardless of the Width parameter setting for the frame.
- The exposure time for acquired lines. If you use longer exposure times, you can acquire fewer lines per second.
- The amount of time that it takes to transmit a completed frame from the camera to your host PC. The amount of time needed to transmit a frame depends on the bandwidth assigned to the camera.

When the camera's acquisition mode is set to single frame, the maximum possible acquisition frame rate can not be achieved. This is true because the camera performs a complete internal setup cycle for each single frame.

To determine the maximum allowed line acquisition rate with your current camera settings, you can use the ResultingLineRateAbs parameter. The ResultingLineRateAbs parameter indicates the camera's current maximum allowed line acquisition rate taking the readout time, exposure time, and bandwidth settings into account.

Increasing the Maximum Allowed Line Rate

You may find that you would like to acquire lines at a rate higher than the maximum allowed with the camera's current settings. In this case, you must first determine what factor is most restricting the maximum line rate. The descriptions of the three factors that appear below will let you determine which factor is restricting the rate.

Factor 1:

Factor 1 is the sensor readout time. The readout time for a particular sensor is a fixed value and thus the maximum line acquisition rate as determined by the readout time is also fixed. The table below shows the maximum line rate (in lines per second) based on sensor readout time for each camera model.

Max Lines/s (nominal; based on sensor readout)				
raL2048-48gm	**raL4096-24gm**	**raL6144-16 gm**	**raL8192-12 gm**	**raL12288-8 gm**
80000	80000	80000	80000	80000

Factor 2:

Factor 2 is the exposure time. You can use the formula below to calculate the maximum line rate based on the exposure time for each acquired line:

$$\text{Max Lines/s} = \frac{1}{\text{Exposure time in µs} + C_1}$$

Where the constant C_1 depends on the camera model and on whether the parameter limit is removed from the ExposureOverhead parameter, as shown in the table below:

	raL2048-48gm	**raL4096-24gm**	**raL6144-16 gm**	**raL8192-12 gm**	**raL12288-8 gm**
C_1 (Default Value)	5.4 µs	5.4 µs	5.4 µs	5.4 µs	5.4 µs
C_1 (Limit Removed From ExposureOverhead Parameter)	3.4 µs	3.4 µs	3.4 µs	3.4 µs	3.4 µs

For more information about setting the exposure time, see Section 8.2.5.2 on page 92. For more information about removing parameter limits and the implications, see Section 10.2 on page 161.

Factor 3:

Factor 3 is the frame transmission time. You can use the formula below to calculate the maximum line rate based on the frame transmission time:

$$\text{Max Lines/s} = \left(\frac{\text{Device Current Throughput Parameter Value}}{\text{Payload Size Parameter Value}}\right) \times \text{Frame Height}$$

Once you have determined which factor is most restrictive on the line rate, you can try to make that factor less restrictive if possible:

- If you find that the sensor readout time is most restrictive factor, you cannot make any adjustments that will result in a higher maximum line rate.
- If you are using long exposure times, it is quite possible to find that your exposure time is the most restrictive factor on the line rate. In this case, you should lower your exposure time. (You may need to compensate for a lower exposure time by using a brighter light source or increasing the opening of your lens aperture.) You can extend the exposure time to some degree after having removed the parameter limits from the ExposureOverhead parameter (for more information, see Section 8.5.1).
- The frame transmission time will not normally be a restricting factor. But if you are using multiple cameras and you have set a small packet size or a large inter-packet delay, you may find that the transmission time is restricting the maximum allowed line rate. In this case, you could increase the packet size or decrease the inter-packet delay. If you are using several cameras connected to the host PC via a network switch, you could also use a multiport network adapter in the PC instead of a switch. This would allow you to increase the Ethernet bandwidth assigned to the camera and thus decrease the transmission time.

For more information about the settings that determine the bandwidth assigned to the camera, see Section 5.2 on page 41.

Example

Assume that you are using an raL2048-48gm camera set for an exposure time of 190 µs and a frame height of 500 lines. Also assume that you use the default value for C_1, that you have checked the value of the DeviceCurrentThroughput parameter and the PayloadSize parameters and found them to be 110000000 and 5120000 respectively.

Factor 1 (sensor readout):

Max Lines/s = 80000

Factor 2 (exposure time):

$$\text{Max Lines/s} = \frac{1}{190\ \mu s + 5.4\ \mu s}$$

Max Lines/s = 5117 Lines/s

Factor 3 (frame transmission time):

$$\text{Max Lines/s} = \left(\frac{110000000}{5120000}\right) \times 500$$

Max Lines/s = 10742

Factor 2, the exposure time, is the most restrictive factor. In this case, the exposure time setting is limiting the maximum allowed line rate to 5117 lines per second. If you wanted to operate the camera at a higher line rate, you would need to lower the exposure time.

Because the exposure time is the most restrictive factor, you could also remove the limit from the ExposureOverhead parameter to increase the line rate. In this case C_1 = 3.4 µs would apply, and according to the formula for Factor 2 a maximum allowed line rate of 5170 lines per second would result.

For more information about removing the limit from the ExposureOverhead parameter and the resulting side effects, see Section 8.5.1.

8.5.1 Removing the Parameter Limits for the ExposureOverhead Parameter

By removing the parameter limits for the ExposureOverhead parameter you can improve the line acquisition with respect to either of the following two goals:

- You can extend the exposure time: When operating a camera at a maximum allowed line rate, the interval between two consecutive line start triggers will consist of the exposure time and the time needed to prepare the sensor for the next acquisition.

  In this situation, the time available for exposure will sometimes, despite bright illumination, be too short for obtaining a sufficiently bright image. To brighten of the image at a given high line rate, you can remove the parameter limits for the ExposureOverhead parameter. As a consequence, the exposure time can to some degree be extended at the expense of the sensor's "preparation time" that will be shortened accordingly. The image quality will somewhat decrease.
- You can increase the maximum allowed line rate: If you remove the parameter limits for the ExposureOverhead parameter, if you do not extend the exposure time accordingly, and if the exposure time is the determining factor for the maximum allowed line rate (Factor 2), you will be able to increase the maximum allowed line rate to some extent (see above). The image quality will somewhat decrease.

If you chose to extend the exposure time you can use the timed and trigger width exposure modes (see Section 8.2.4.2 on page 87, Section 8.3.1 on page 120 and Section 8.3.3.3 on page 126 for information about the timed exposure mode and Section 8.2.4.2 on page 87 and Section 8.3.3.3 on page 126 for information about the trigger width exposure mode):

- When the camera is set for the timed exposure mode, the upper limit of the ExposureTime parameters increases according to the decrease of the "preparation time".
- When the camera is set for the trigger width exposure mode, you can extend the ExLSTrig signal period according to the decrease of the exposure overhead time. To maintain the line rate, increase the ExposureOverlapTimeMaxAbs parameter value by the amount of time that was gained for exposure time.

To remove the limits for the ExposureOverhead parameter:

1. Use the ParameterSelector parameter to select the ExposureOverhead parameter.
2. Set the value of the RemoveLimits parameter.

You can set the ParameterSelector and the RemoveLimits parameter values from within your application software by using the Basler pylon API. The following code snippet illustrates using the API to set the parameter values:

```
// Select the parameter whose factory limits will be removed.
camera.ParameterSelector.SetValue(ParameterSelector_ExposureOverhead);
// Remove the limits for the selected parameter.
camera.RemoveLimits.SetValue(true);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

8.6 The Shaft Encoder Module

The camera is equipped with a shaft encoder software module. The module can accept input from a two channel shaft encoder (Phase A and Phase B). The module outputs a signal that can be used, for example, as a source signal for the line start trigger function or the frame start trigger function in the camera. Fig. 52 shows a typical implementation of the shaft encoder software module in the camera.

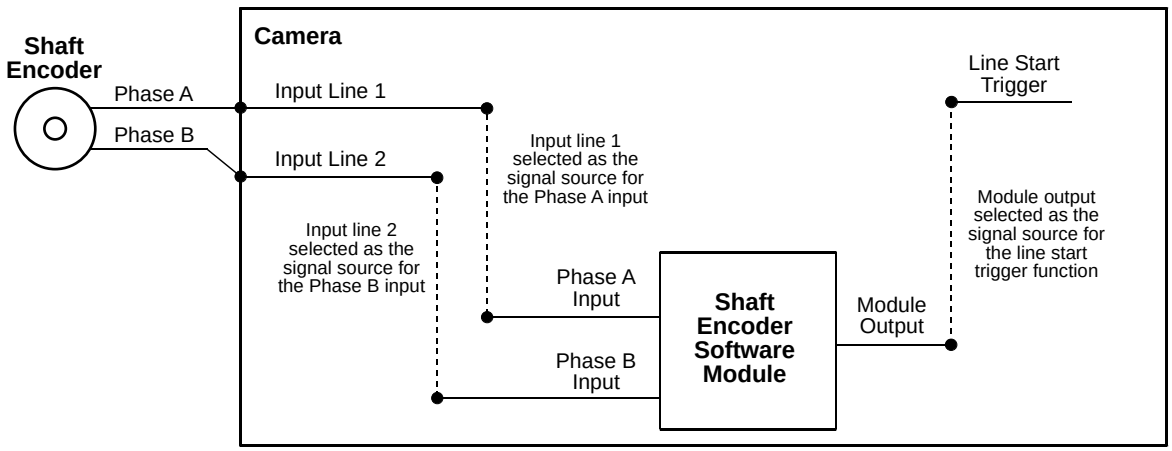


Fig. 52: Typical Shaft Encoder Module Implementation

To use the shaft encoder module, you must select a source signal for the Phase A input and for the Phase B input on the module. The allowed source signals for the Phase A and Phase B module inputs are camera input line 1, camera input line 2, and camera input line 3. So, for example, you could apply the Phase A signal from a shaft encoder to physical input line 1 of the camera and select input line 1 as the source signal for the Phase A input to the module. And you could apply the Phase B signal from a shaft encoder to physical input line 2 of the camera and select input line 2 as the source signal for the Phase B input to the module. More information about selecting a source signal for the module inputs appears in a code snippet later in this section.

Fig. 53 shows how the software module will interpret the input from the shaft encoder when the encoder is connected as illustrated in Fig. 52. The software module will sense forward ticks from the encoder when the input is as shown in the left part of Fig. 53. The software module will sense reverse ticks from the encoder when the input is as shown in the right part of the Fig. 53.

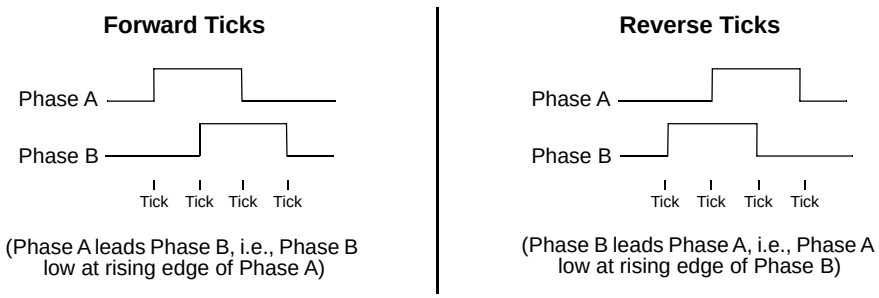


Fig. 53: Software Module Direction Sensing

If this interpretation of direction is not as you desire, you could change it by moving the Phase A output from the shaft encoder to input line 2 and the Phase B output to input line 1.

Shaft Encoder Module Parameters

There are several parameters and commands associated with the shaft encoder module. The list below describes the parameters and commands and explains how they influence the operation of the module.

- The **ShaftEncoderModuleCounterMode** parameter controls the tick counter on the shaft encoder module. The tick counter counts the number of ticks that have been received by the module from the shaft encoder. This parameter has two possible values: FollowDirection and IgnoreDirection.

  If the mode is set to FollowDirection, the counter will increment when the module receives forward ticks from the shaft encoder and will decrement when it receives reverse ticks.

  If the mode is set to IgnoreDirection, the counter will increment when it receives either forward ticks or reverse ticks.
- The **ShaftEncoderModuleCounter** parameter indicates the current value of the tick counter. This is a read only parameter.
- The **ShaftEncoderCounterModuleMax** parameter sets the maximum value for the tick counter. The minimum value for this parameter is 0 and the maximum is 32767.

  If the counter is incrementing and it reaches the max, it will roll over to 0. That is:
  Max + 1 = 0

  If the counter is decrementing and it reaches 0, it will roll back to the max. That is:
  0 - 1 = Max
- The **ShaftEncoderModuleCounterReset** command resets the tick counter count to 0.
- The **ShaftEncoderModuleMode** parameter controls the behavior of the "reverse counter" that is built into the module. This parameter has two possible values: AnyDirection and ForwardOnly. For more information about this parameter, see the detailed description of the reverse counter that appears later in this section.
- The **ShaftEncoderModuleReverseCounterMax** parameter sets a maximum value for the module's "reverse counter". The minimum value for this parameter is 0 and the maximum is 32767. For more information about this parameter, see the detailed description of the reverse counter that appears later in this section.
- The **ShaftEncoderModuleReverseCounterReset** command resets the reverse counter count to 0 and informs the software module that the current direction of conveyor movement is forward. For more information about this parameter, see the detailed description of the reverse counter that appears later in this section.

Setting the Shaft Encoder Module Parameters

To use the shaft encoder software module effectively:

1. Select a signal source for the Phase A and Phase B inputs on the module.

   (By default, input line 1 is selected as the signal source for the Phase A input and input line 2 is selected as the signal source for the Phase B input.)
2. Make sure that the output from the encoder module is selected as the signal source for a camera function. Currently, output from the encoder module can be selected as the signal source for the camera's Frame Start Trigger function or for the camera's Line Start Trigger function.
3. Set the ShaftEncoderModuleCounterMode and the ShaftEncoderModuleMode parameters as appropriate.

You can set the encoder module parameter values, issue commands to the encoder module, and select signal sources from within your application software by using the pylon API. The code snippet below illustrates using the API to set the parameter values and to issue commands to the encoder module.

```
// Select physical input line 1 as the source signal for the Phase A input on the
module and physical input line 2 as the source signal for the Phase B input
camera.ShaftEncoderModuleLineSelector.SetValue(ShaftEncoderModuleLineSelector_
PhaseA);
camera.ShaftEncoderModuleLineSource.SetValue(ShaftEncoderModuleLineSource_
Line1);
camera.ShaftEncoderModuleLineSelector.SetValue(ShaftEncoderModuleLineSelector_
PhaseB);
camera.ShaftEncoderModuleLineSource.SetValue(ShaftEncoderModuleLineSource_
Line2);


// Enable the camera's Line Start Trigger function and select the output from the
encoder module as the source signal for the Line Start Trigger
camera.TriggerSelector.SetValue(TriggerSelector_LineStart);
camera.TriggerMode.SetValue(TriggerMode_On);
camera.TriggerSource.SetValue (TriggerSource_ShaftEncoderModuleOut);
camera.TriggerActivation.SetValue(TriggerActivation_RisingEdge);


// Set the shaft encoder module counter mode
camera.ShaftEncoderModuleCounterMode.SetValue(ShaftEncoderModuleCounterMode_
FollowDirection);


// Set the shaft encoder module mode
camera.ShaftEncoderModuleMode.SetValue(ShaftEncoderModuleMode_AnyDirection);


// Set the shaft encoder module counter max and the shaft encoder module reverse
counter max
camera.ShaftEncoderModuleCounterMax.SetValue(32767);
camera.ShaftEncoderModuleReverseCounterMax.SetValue(15);
```

```
// Get the current value of the shaft encoder module counter
int64_t encodercounterSize = camera.ShaftEncoderModuleCounter.GetValue();

// Reset the shaft encoder module counter and the shaft encoder module reverse
counter
camera.ShaftEncoderModuleCounterReset.Execute();
camera.ShaftEncoderModuleReverseCounterReset.Execute();
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

For more information about the line start trigger, see Section 8.2.4 on page 86.

The Reverse Counter

The main purpose of the reverse counter is to compensate for mechanical "jitter" in the conveyor used to move objects past the camera. This jitter usually manifests itself as a momentary change in the direction of the conveyor.

The rules that govern the operation of the reverse counter are as follows:

- If the conveyor is running in the reverse direction and the current reverse counter count is less than the maximum (i.e., less than the current setting of the ReverseCounterMax parameter), the reverse counter will increment once for each shaft encoder reverse tick received.
- If the conveyor is running in the forward direction and the current reverse counter count is greater than zero, the reverse counter will decrement once for each shaft encoder forward tick received.
- When the Shaft Encoder Mode is set to Forward Only:

  If the reverse counter is not incrementing or decrementing, the software module will output a trigger signal for each forward tick received from the shaft encoder.

  If the reverse counter is incrementing or decrementing, trigger signal output will be suppressed.
- When the Shaft Encoder Mode is set to Any Direction:

  If the reverse counter is not incrementing or decrementing, the software module will output a trigger signal for each forward tick or reverse tick received from the shaft encoder.

  If the reverse counter is incrementing or decrementing, trigger signal output will be suppressed.

To understand how these rules affect the operation of the encoder software module, consider the following cases:

Case 1

This is the simplest case, i.e., the Shaft Encoder Reverse Counter Max is set to zero. In this situation, the reverse counter never increments or decrements and it will have no effect on the operation of the encoder software module.

When the Shaft Encoder Reverse Counter Max is set to zero:

- If the Shaft Encoder Module Mode is set to Forward Only, the software module will output a trigger signal whenever it receives a forward tick from the shaft encoder, but not when it receives a reverse tick.
- If the Shaft Encoder Module Mode is set to Any Direction, the software module will output a trigger signal whenever it receives either a forward tick or a reverse tick from the shaft encoder.

Case 2

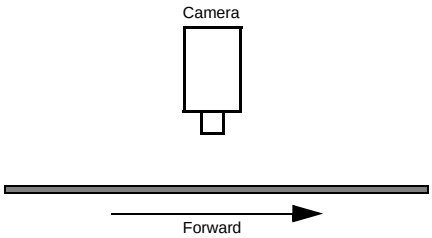
In this case, assume that:

- A shaft encoder is attached to a conveyor belt that normally moves continuously in the forward direction past a camera.
- The conveyor occasionally "jitters" and when it jitters, it moves in reverse for 4 or 5 ticks.

For this case, the ShaftEncoderModuleMode parameter should be set to ForwardOnly. The ShaftEncoderModuleReverseCounterMax should be set to a value that is higher than the jitter we expect to see. We decide to set the value to 10.

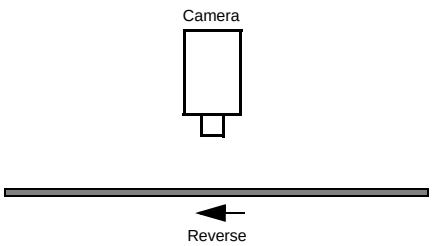
Given this situation and these settings, the series of diagrams below explains how the encoder software module will act:

1



The conveyor is moving forward and the encoder is generating forward ticks. Whenever the module receives a forward tick, it outputs a trigger signal. The reverse counter is at 0.

2

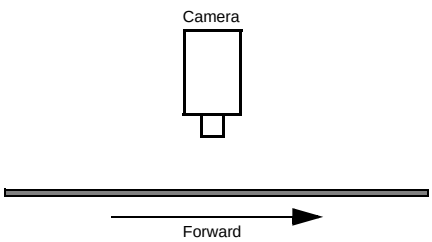


The conveyor jitters and moves briefly in reverse. During this reverse movement, the shaft encoder generates 5 reverse ticks.

The reverse counter will increment by 1 for each reverse tick and when the reverse motion stops, the reverse counter count will be 5.

While the reverse counter is incrementing, the output of trigger signals from the module is suppressed.

3



The conveyor resumes forward motion and the shaft encoder module begins generating forward ticks.

The reverse counter will decrement by 1 for each forward tick. While the reverse counter is decrementing, the output of trigger signals from the module is suppressed.

When the reverse counter decrements to 0, decrementing stops and suppression of the trigger signals stops. The module will begin outputting a trigger signal for each forward tick received.

By suppressing trigger signals when the conveyor was moving in reverse and then suppressing an equal number of trigger signals when forward motion is resumed, we ensure that the conveyor is in its "pre-jitter" position when the module begins generating trigger signals again.

Note in step two that if the conveyor runs in reverse for a long period and the reverse counter reaches the max setting, the counter simply stops incrementing. If the conveyor continues in reverse, no output triggers will be generated because the Shaft Encoder Mode is set to Forward only.

Case 3

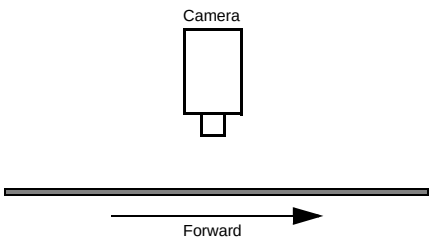
In this case, assume that:

- We are working with a small conveyor that moves back and forth in front of a camera.
- A shaft encoder is attached to the conveyor.
- The conveyor moves in the forward direction past the camera through its complete range of motion, stops, and then begins moving in reverse.
- The conveyor moves in the reverse direction past the camera through its complete range of motion, stops, and then begins moving forward.
- This back an forth motion repeats.
- The conveyor occasionally "jitters". When it jitters, it moves 4 or 5 ticks in a direction of travel opposite to the current normal direction.

For this case, the ShaftEncoderModuleMode parameter should be set to Any Direction. The ShaftEncoderModuleReverseCounterMax should be set to a value that is higher than the jitter we expect to see. We decide to set the value to 10.

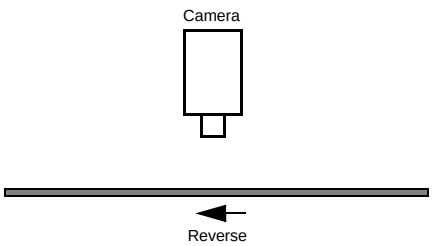
Given this situation and these settings, this series of diagrams explains how the encoder software module will act during conveyor travel:

1



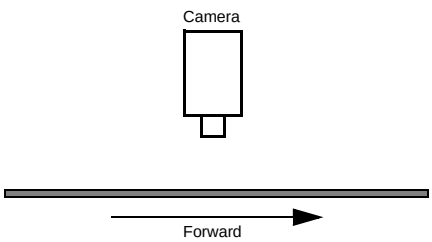
The conveyor is moving forward and the encoder is generating forward ticks. Whenever the module receives a forward tick, it outputs a trigger signal.
The reverse counter is at 0.

2



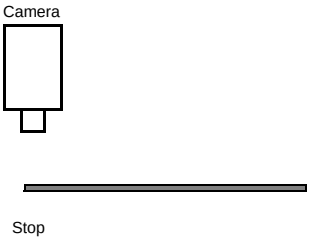
The conveyor jitters and moves briefly in reverse. During this reverse movement, the shaft encoder generates 5 reverse ticks.
The reverse counter will increment by 1 for each reverse tick and when the reverse motion stops, the reverse counter count will be 5.
While the reverse counter is incrementing, the output of trigger signals from the module is suppressed.

3



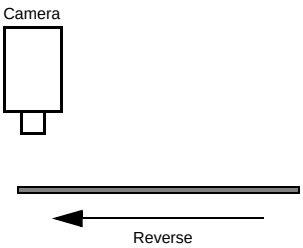
The conveyor resumes forward motion and the shaft encoder module begins generating forward ticks.
The reverse counter will decrement by 1 for each forward tick. While the reverse counter is decrementing, the output of trigger signals from the module is suppressed
When the reverse counter decrements to 0, decrementing stops and suppression of the trigger signals stops. The module will begin outputting a trigger signal for each forward tick received.

4



The conveyor reaches the end of its forward travel and it stops.

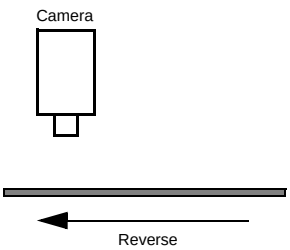
5



The conveyor begins moving in reverse and the shaft encoder starts generating reverse ticks.

The reverse counter will increment by 1 for each reverse tick. While the reverse counter is incrementing and the reverse count is below the max (10 in this case), the output of trigger signals from the module is suppressed.

6

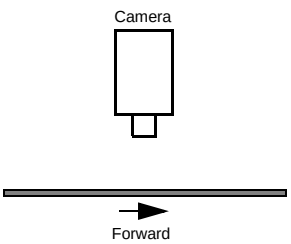


The reverse counter reaches the max (10 in this case) and stops incrementing.

Suppression of trigger signals is ended. Because the shaft encoder mode is set to any direction, the module begins generating one trigger signal for each reverse tick received.

The reverse counter remains at 10.

7

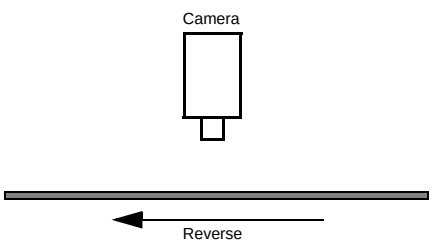


The conveyor jitters and moves forward briefly. During this forward movement, the shaft encoder generates 4 forward ticks.

The reverser counter will decrement by 1 for each forward tick. When the forward motion stops, the reverse counter count will be 6.

While the reverse counter is decrementing, the output of trigger signals from the module is suppressed.

8

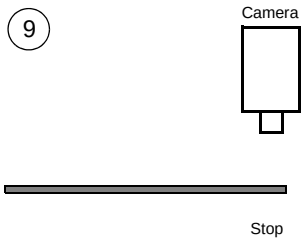


The conveyor resumes reverse motion and the shaft encoder module begins generating reverse ticks.

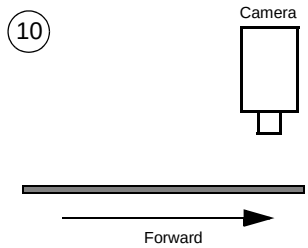
The reverse counter will increment by 1 for each reverse tick. While the reverse counter is incrementing, the output of trigger signals from the module is suppressed.

When the reverse counter reaches the max (10 in this case) it stops incrementing and suppression of the trigger signals stops. The module will resume outputting a trigger signal for each reverse tick received.

The reverse counter count is now 10.

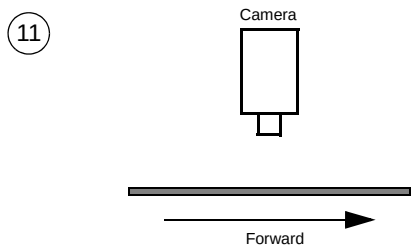


The conveyor reaches the end of its reverse travel and it stops.



The conveyor begins moving forward and the shaft encoder starts generating forward ticks.

The reverse counter is at 10 and will now begin decrementing by 1 for each forward tick. While the reverse counter is decrementing and the reverse count is greater than 0, the output of trigger signals from the module is suppressed.



The reverse counter reaches 0.

Suppression of trigger signals is ended. Because the shaft encoder mode is set to any direction, the module begins generating one trigger signal for each forward tick received.

The reverse counter remains at 0.

There are two main things to notice about this example. First, because the encoder mode is set to any direction, ticks from the shaft encoder will cause the module to output trigger signals regardless of the conveyor direction, as long as the reverse counter is not incrementing or decrementing. Second, the reverse counter will compensate for conveyor jitter regardless of the conveyor direction.

	It is important to reset the reverse counter before the first traverse in the forward direction. A reset sets the counter to 0 and synchronizes the counter software with the conveyor direction. (The software assumes that the conveyor will move in the forward direction after a counter reset).

8.7 Frequency Converter

The camera is equipped with a frequency converter module that allows triggering the camera at a frequency that differs from the frequency of the input signals received.

The module can accept input signals from one of the three input lines or signals (ticks) from the shaft encoder module.

The frequency converter module includes three sub-modules acting in sequence on the original signals:

- The pre-divider module receives the input signals. The module allows employing an integer factor, the pre-divider, to decrease the original frequencies and passes the signals on to the next module, the multiplier module.

  If for example a pre-divider of 2 is selected only every other input signal is passed out unchanged to the multiplier module and, accordingly, the frequency is halved. If a pre-divider of 1 is selected every input signal is passed out unchanged to the multiplier module.

  Employing the pre-divider may be advisable for decreasing periodic jitter of the input signals and will be required if the input signal frequency is higher than 100 kHz. The signal frequency of the signals passed on to the multiplier module must be within the range of 10 Hz to 100 kHz. Periodic jitter is likely to be present when input signals from the shaft encoder are accepted.

  We recommend to only use low values for the pre-divider. The original signal frequency should be changed as little as possible to facilitate frequency adjustment by the multiplier module.
- The multiplier module receives the signals from the pre-divider module. The signal frequency must be within the range of 10 Hz to 100 kHz. The multiplier module allows applying an integer factor, the multiplier, to generate signals at increased frequencies and passes the signals on to the next module, the post-divider module.

  If, for example, a multiplier of 2 is selected signals are generated at double the frequency of the signals received from the pre-divider module and are passed on to the divider module. If a multiplier of 1 is selected every signal received from the pre-divider module is passed unchanged on to the divider module.

  The Align parameter can be set to "rising edge" and "falling edge". If "rising edge" is selected there will be for the rising edge of each signal received from the pre-divider module a phase-locked, matching rising edge among the signals generated. If "falling edge" is selected there will be for the falling edge of each signal received from the pre-divider module a phase-locked, matching falling edge among the signals generated.

  Make sure to select a multiplier that will not too much increase the frequency such that the camera will be overtriggered. Temporarily, a too high frequency may occur during frequency adjustment causing overtriggering even if a relatively low multiplier was selected. The PreventOvertrigger parameter provides a safeguard against overtriggering the camera.
  We recommend setting the PreventOvertrigger parameter to True to prevent overtriggering.
- The post-divider module receives the signals from the multiplier module. The post-divider module allows employing an integer factor, the post-divider, to generate signals at decreased frequencies and provides these signals to be used as camera trigger signals, e.g. as line start triggers.

If for example a post-divider of 2 is selected only every other signal received from the multiplier module is passed out from the divider module and, accordingly, the frequency is halved. If a post-divider of 1 is selected every signal received from the multiplier module is passed out unchanged from the divider module.

You can use the frequency converter to multiply the original signal frequency by a fractional value. We recommend multiplying the frequency by the enumerator value using the multiplier module and dividing the resulting frequency by the denominator value using the post-divider module.

You can configure the frequency converter module from within your application by using a dynamic API. The following code snippet illustrates setting parameter values:

```
INodeMap &Control = *camera.GetNodeMap();

// possible values for FrequencyConverterInputSource:
// Line1
// Line2
// Line3
// ShaftEncoderModuleOut
CEnumerationPtr(Control.GetNode("FrequencyConverterInputSource"))-
>FromString("ShaftEncoderModuleOut");

// ranges for divider and multiplier:
// divider   : 1...128
// multiplier: 1...32
CIntegerPtr(Control.GetNode("FrequencyConverterPreDivider"))->SetValue(4);
CIntegerPtr(Control.GetNode("FrequencyConverterMultiplier"))->SetValue(17);
CIntegerPtr(Control.GetNode("FrequencyConverterPostDivider"))->SetValue(1);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

For more information about the shaft encoder module see, Section 8.6 on page 136.

9 Pixel Data Formats

By selecting a pixel data format, you determine the format (layout) of the image data transmitted by the camera. This section provides detailed information about the available pixel data formats.

9.1 Setting the Pixel Data Format

The setting for the camera's PixelFormat parameter determines the format of the pixel data that will be output from the camera. Table 11 lists the pixel formats available on each camera type.

Mono Camera Pixel Formats
Mono 8
Mono 12
Mono 12 Packed
YUV 4:2:2 Packed
YUV 4:2:2 (YUYV) Packed

Table 11: Available Pixel Formats

Details of the monochrome camera formats are described in Section 9.2 on page 148.

You can set the PixelFormat parameter value from within your application software by using the pylon API. The following code snippet illustrates using the API to set the parameter value:

```
camera.PixelFormat.SetValue(PixelFormat_Mono8);
camera.PixelFormat.SetValue(PixelFormat_Mono12);
camera.PixelFormat.SetValue(PixelFormat_Mono12Packed);
camera.PixelFormat.SetValue(PixelFormat_YUV422Packed);
camera.PixelFormat.SetValue(PixelFormat_YUV422_YUYV_Packed);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

9.2 Pixel Data Formats

9.2.1 Mono 8 Format

When a monochrome camera is set for the Mono 8 pixel data format, it outputs 8 bits of brightness data per pixel.

The table below describes how the pixel data for a received frame will be ordered in the image buffer in your PC when the camera is set for Mono 8 output.

The following standards are used in the table:

P_0 = the first pixel transmitted by the camera

P_n = the last pixel transmitted by the camera

B_0 = the first byte in the buffer

B_m = the last byte in the buffer

Byte	Pixel - Data Bits
B_0	P_0 7 ... 0
B_1	P_1 7 ... 0
B_2	P_2 7 ... 0
B_3	P_3 7 ... 0
B_4	P_4 7 ... 0
•	•
•	•

Byte	Pixel - Data bits
•	•
•	•
B_{m-4}	P_{n-4} 7 ... 0
B_{m-3}	P_{n-3} 7 ... 0
B_{m-2}	P_{n-2} 7 ... 0
B_{m-1}	P_{n-1} 7 ... 0
B_m	P_n 7 ... 0

With the camera set for Mono 8, the pixel data output is 8 bit data of the “unsigned char” type. The available range of data values and the corresponding indicated signal levels are as shown in the table below.

This Data Value (Hexadecimal)	Indicates This Signal Level (Decimal)
0xFF	255
0xFE	254
•	•
•	•
•	•
0x01	1
0x00	0

9.2.2 Mono 12 Format

When a monochrome camera is set for the Mono 12 pixel data format, it outputs 16 bits of brightness data per pixel with 12 bits effective. The 12 bits of effective pixel data fill from the least significant bit. The four unused most significant bits are filled with zeros.

The table below describes how the pixel data for a received frame will be ordered in the image buffer in your PC when the camera is set for Mono 12 output. The data is placed in the image buffer in **little endian format**.

The following standards are used in the table:

P_0 = the first pixel transmitted by the camera

P_n = the last pixel transmitted by the camera

B_0 = the first byte in the buffer

B_m = the last byte in the buffer

x = unused bit, zero filled

Byte	Pixel - Data Bits
B_0	P_0 7 ... 0
B_1	P_0 x x x x 11 ... 8
B_2	P_1 7 ... 0
B_3	P_1 x x x x 11 ... 8
B_4	P_2 7 ... 0
B_5	P_2 x x x x 11 ... 8
B_6	P_3 7 ... 0
B_7	P_3 x x x x 11 ... 8
B_8	P_4 7 ... 0
B_9	P_4 x x x x 11 ... 8
•	•
•	•
•	•
B_{m-7}	P_{n-3} 7 ... 0
B_{m-6}	P_{n-3} x x x x 11 ... 8
B_{m-5}	P_{n-2} 7 ... 0
B_{m-4}	P_{n-2} x x x x 11 ... 8
B_{m-3}	P_{n-1} 7 ... 0
B_{m-2}	P_{n-1} x x x x 11 ... 8
B_{m-1}	P_n 7 ... 0
B_m	P_n x x x x 11 ... 8

When the camera is set for Mono 12, the pixel data output is 16 bit data of the “unsigned short (little endian)” type. The available range of data values and the corresponding indicated signal levels are as shown in the table below. For 16 bit data, you might expect a value range from 0x0000 to 0xFFFF. However, with the camera set for Mono 12 only 12 bits of the 16 bits transmitted are effective. Therefore, the highest data value you will see is 0x0FFF indicating a signal level of 4095.

This Data Value (Hexadecimal)	Indicates This Signal Level (Decimal)
0x0FFF	4095
0x0FFE	4094
•	•
•	•
•	•
0x0001	1
0x0000	0

9.2.3 Mono 12 Packed Format

When a monochrome camera is set for the Mono 12 Packed pixel data format, it outputs 12 bits of brightness data per pixel. Every three bytes transmitted by the camera contain data for two pixels.

The table below describes how the pixel data for a received frame will be ordered in the image buffer in your PC when the camera is set for Mono 12 Packed output.

The following standards are used in the table:

P_0 = the first pixel transmitted by the camera

P_n = the last pixel transmitted by the camera

B_0 = the first byte in the buffer

B_m = the last byte in the buffer

Byte	Pixel - Data Bits	
B_0	P_0 11 ... 4	
B_1	P_1 3 ... 0	P_0 3 ... 0
B_2	P_1 11 ... 4	
B_3	P_2 11 ... 4	
B_4	P_3 3 ... 0	P_2 3 ... 0
B_5	P_3 11 ... 4	
B_6	P_4 11 ... 4	
B_7	P_5 3 ... 0	P_4 3 ... 0
B_8	P_5 11 ... 4	
B_9	P_6 11 ... 4	

B_{10}	P_7 3 ... 0	P_6 3 ... 0
B_{11}	P_7 1 ... 4	
•	•	
•	•	•
•	•	
B_{m-5}	P_{n-3} 11 ... 4	
B_{m-4}	P_{n-2} 3 ... 0	P_{n-3} 3 ... 0
B_{m-3}	P_{n-2} 11 ... 4	
B_{m-2}	P_{n-1} 11 ... 4	
B_{m-1}	P_n 3 ... 0	P_{n-1} 3 ... 0
B_m	P_n 11 ... 4	

When a monochrome camera is set for Mono 12 Packed, the pixel data output is 12 bit data of the “unsigned” type. The available range of data values and the corresponding indicated signal levels are as shown in the table below.

This Data Value (Hexadecimal)	**Indicates This Signal Level (Decimal)**
0x0FFF	4095
0x0FFE	4094
•	•
•	•
•	•
0x0001	1
0x0000	0

9.2.4 YUV 4:2:2 Packed Format

When a monochrome camera is set for the YUV 4:2:2 Packed pixel data format, the camera transmits Y, U, and V values in a fashion that mimics the output from a color camera set for YUV 4:2:2 Packed.

The Y value transmitted for each pixel is the actual 8 bit brightness value similar to the pixel data transmitted when a monochrome camera is set for Mono 8. The U and V values transmitted will always be zero. With this format, a Y value is transmitted for each pixel, but the U and V values are only transmitted for every second pixel.

The table below describes how the pixel data for a received frame will be ordered in the image buffer in your PC when the camera is set for YUV 4:2:2 Packed output.

The following standards are used in the table:

P_0 = the first pixel transmitted by the camera

P_n = the last pixel transmitted by the camera

B_0 = the first byte in the buffer

B_m = the last byte in the buffer

Byte	Pixel - Data Bits
B_0	U P_0 7 ... 0
B_1	Y P_0 7 ... 0
B_2	V P_0 7 ... 0
B_3	Y P_1 7 ... 0
B_4	U P_2 7 ... 0
B_5	Y P_2 7 ... 0
B_6	V P_2 7 ... 0
B_7	Y P_3 7 ... 0
B_8	U P_4 7 ... 0
B_9	Y P_4 7 ... 0
B_{10}	V P_4 7 ... 0
B_{11}	Y P_5 7 ... 0
•	•
•	•
•	•
B_{m-7}	U P_{n-3} 7 ... 0
B_{m-6}	Y P_{n-3} 7 ... 0
B_{m-5}	V P_{n-3} 7 ... 0
B_{m-4}	Y P_{n-2} 7 ... 0
B_{m-3}	U P_{n-1} 7 ... 0
B_{m-2}	Y P_{n-1} 7 ... 0
B_{m-1}	V P_{n-1} 7 ... 0
B_m	Y P_n 7 ... 0

When the camera is set for YUV 4:2:2 Packed output, the pixel data output for the Y component is 8 bit data of the “unsigned char” type. The range of data values for the Y component and the corresponding indicated signal levels are shown below.

This Data Value (Hexadecimal)	Indicates This Signal Level (Decimal)
0xFF	255
0xFE	254
•	•
•	•
•	•
0x01	1
0x00	0

The pixel data output for the U component or the V component is 8 bit data of the “straight binary” type and will always be zero.

This Data Value (Hexadecimal)	Indicates This Signal Level (Decimal)
0x00	0

9.2.5 YUV 4:2:2 (YUYV Packed) Format

When a monochrome camera is set for the YUV 4:2:2 (YUYV) Packed pixel data format, the camera transmits Y, U, and V values in a fashion that mimics the output from a color camera set for YUV 4:2:2 (YUYV) Packed.

The YUV 4:2:2 (YUYV) packed pixel data format is similar to the YUV 4:2:2 pixel format described in the previous section. The only difference is the order of the bytes transmitted to the host PC. With the YUV 4:2:2 format, the bytes are ordered as specified in the DCAM standard issued by the 1394 Trade Association. With the YUV 4:2:2 (YUYV) format, the bytes are ordered to emulate the ordering normally associated with analog frame grabbers and Windows® frame buffers.

The Y value transmitted for each pixel is the actual 8 bit brightness value similar to the pixel data transmitted when a monochrome camera is set for Mono 8. The U and V values transmitted will always be zero. With this format, a Y value is transmitted for each pixel, but the U and V values are only transmitted for every second pixel.

The table below describes how the pixel data for a received frame will be ordered in the image buffer in your PC when the camera is set for YUV 4:2:2 (YUYV) output.

With this format, the Y component is transmitted for each pixel, but the U and V components are only transmitted for every second pixel.

The following standards are used in the table:

P_0 = the first pixel transmitted by the camera

P_n = the last pixel transmitted by the camera

B_0 = the first byte in the buffer

B_m = the last byte in the buffer

Byte	Pixel - Data Bits
B_0	Y P_0 7 ... 0
B_1	U P_0 7 ... 0
B_2	Y P_1 7 ... 0
B_3	V P_0 7 ... 0
B_4	Y P_2 7 ... 0
B_5	U P_2 7 ... 0
B_6	Y P_3 7 ... 0
B_7	V P_2 7 ... 0
B_8	Y P_4 7 ... 0
B_9	U P_4 7 ... 0
B_{10}	Y P_5 7 ... 0
B_{11}	V P_4 7 ... 0
•	•
•	•
•	•
B_{m-7}	Y P_{n-3} 7 ... 0
B_{m-6}	U P_{n-3} 7 ... 0
B_{m-5}	Y P_{n-2} 7 ... 0
B_{m-4}	V P_{n-3} 7 ... 0
B_{m-3}	Y P_{n-1} 7 ... 0
B_{m-2}	U P_{n-1} 7 ... 0
B_{m-1}	Y P_n 7 ... 0
B_m	V P_{n-1} 7 ... 0

When the camera is set for YUV 4:2:2 (YUYV) output, the pixel data output for the Y component is 8 bit data of the “unsigned char” type. The range of data values for the Y component and the corresponding indicated signal levels are shown below.

This Data Value (Hexadecimal)	Indicates This Signal Level (Decimal)
0xFF	255
0xFE	254
•	•
•	•
•	•
0x01	1
0x00	0

The pixel data output for the U component or the V component is 8 bit data of the “straight binary” type and will always be zero.

This Data Value (Hexadecimal)	Indicates This Signal Level (Decimal)
0x00	0

9.3 Pixel Transmission Sequence

For each acquired frame, pixel data is transmitted from the camera in the following sequence:

$Row_0 Col_0$,	$Row_0 Col_1$,	$Row_0 Col_2$	.. ..	$Row_0 Col_{m-2}$,	$Row_0 Col_{m-1}$,	$Row_0 Col_m$
$Row_1 Col_0$,	$Row_1 Col_1$,	$Row_1 Col_2$	.. ..	$Row_1 Col_{m-2}$,	$Row_1 Col_{m-1}$,	$Row_1 Col_m$
$Row_2 Col_0$,	$Row_2 Col_1$,	$Row_2 Col_2$	.. ..	$Row_2 Col_{m-2}$,	$Row_2 Col_{m-1}$,	$Row_2 Col_m$
:	:	:		:	:	:
:	:	:		:	:	:
$Row_{n-2} Col_0$,	$Row_{n-2} Col_1$,	$Row_{n-2} Col_2$	.. ..	$Row_{n-2} Col_{m-2}$,	$Row_{n-2} Col_{m-1}$,	$Row_{n-2} Col_m$
$Row_{n-1} Col_0$,	$Row_{n-1} Col_1$,	$Row_{n-1} Col_2$	.. ..	$Row_{n-1} Col_{m-2}$,	$Row_{n-1} Col_{m-1}$,	$Row_{n-1} Col_m$
$Row_n Col_0$,	$Row_n Col_1$,	$Row_n Col_2$	.. ..	$Row_n Col_{m-2}$,	$Row_n Col_{m-1}$,	$Row_n Col_m$

Where:

- $Row_0 Col_0$ is the upper left corner of the frame.
- The columns are numbered 0 through m from the left side to the right side of the frame
- The rows are numbered 0 through n from the top to the bottom of the frame, corresponding to n+1 line acquisitions.

10 Standard Features

This chapter provides detailed information about the standard features available on each camera. It also includes an explanation of their operation and the parameters associated with each feature.

10.1 Gain and Black Level

10.1.1 Gain

The camera's gain is adjustable. As shown in Fig. 54, increasing the gain increases the slope of the response curve for the camera. This results in an increase in the gray values output from the camera for a given amount of output from the imaging sensor. Decreasing the gain decreases the slope of the response curve and results in lower gray values for a given amount of sensor output.

Increasing the gain is useful when at your brightest exposure, the highest gray values achieved are lower than 255 (for pixel data formats with 8 bit depth) or 4095 (for pixel data formats with 12 bit depth). For example, if you found that at your brightest exposure the gray values output by the camera were no higher than 127 (in an 8 bit format), you could increase the gain to 6 dB (an amplification factor of 2) and thus reach gray values of 254.

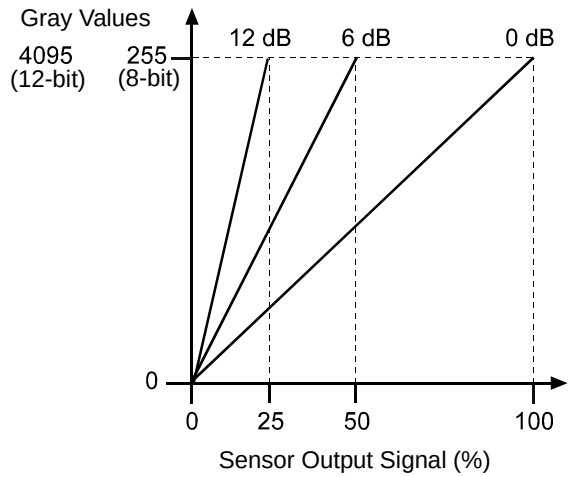


Fig. 54: Gain in dB

You can use the analog gain for coarsely setting gain and the digital gain for finer adjustment.

10.1.1.1 Analog Gain

The camera's analog gain is determined by the GainRaw parameter with the gain selector set to AnalogAll. All pixels in the sensor are affected by this setting.

The allowed parameter values are 1 and 4. A parameter value of 1 corresponds to 0 dB and gain will not be modified. A parameter value of 4 corresponds to 12 dB and an amplification factor of 4.

You must stop image acquisition by issuing an acquisition stop command before changing the analog gain settings.

For more information about the acquisition stop command, see Section 8.2.1 on page 77.

To set the analog gain:

1. Set the GainSelector parameter to AnalogAll.
2. Set the GainRaw parameter to 1 or 4, as desired.

You can set the GainSelector and the GainRaw parameter values from within your application software by using the pylon API. The following code snippet illustrates using the API to set the parameter values:

```
// Set analog gain
camera.GainSelector.SetValue(GainSelector_AnalogAll);
camera.GainRaw.SetValue(4);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

10.1.1.2 Digital Gain

Adjusting the camera's digital gain will digitally shift the group of bits that is output for the pixel values from each ADC in the camera.

Increasing the digital gain setting will result in an amplified gain and therefore in higher pixel values. Decreasing the digital gain setting will result in a decreased gain and therefore in lower pixel values.

The digital gain can be set on an integer scale ranging from 256 to 2047. This range of settings is linearly related to a range of amplification factors where a parameter value of 256 corresponds to 0 dB and gain will not be modified and a parameter value of 2047 corresponds to 18.058 dB and an amplification factor of approximately 7.996.

You can use the formula below to calculate the dB of gain that will result from the GainRaw parameter values:

$$Gain[dB] = 20 \times \log_{10}\left(\frac{\text{Gain Raw}}{256}\right)$$

$$Gain[dB] = 20 \times \log_{10}(\text{Gain Raw}) - 48.165$$

Due to the nature of digital gain, certain gray values will be absent in the image ("missing codes") if digital gain is set to a value larger than 256.

You can use the remove parameter limits feature to remove to lower limit for digital gain parameter values. When you use the remove parameter limits feature you can also set digital gain parameter values in the range from 0 to 255. This corresponds to a range of amplification factors from 0 to approximately 0.99.

If the digital gain parameter value is set below 256 using the remove parameter limits feature: In this case, regardless of the brightness of illumination, the camera will not be able to reach the maximum gray values that otherwise could be reached. For example, if the camera is set to a 12 bit pixel data format, the maximum gray value of 4095 can not be reached if the digital gain parameter value is set below 256.

For more information about the remove parameter limits feature, see Section 10.2 on page 161.

To set the digital gain:

1. Set the GainSelector parameter to DigitalAll.
2. Set the GainRaw parameter to your desired value.

You can set the GainSelector and the GainRaw parameter values from within your application software by using the pylon API. The following code snippet illustrates using the API to set the the parameter values:

```
// Set digital gain
camera.GainSelector.SetValue(GainSelector_DigitalAll);
camera.GainRaw.SetValue(264);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

10.1.1.3 Using Both Analog Gain and Digital Gain

You can use analog gain and digital gain at the same time. In this case, the amplification factors will multiply. For example, if you set analog gain to an amplification factor of 4 and use an amplification factor of 1.2 for digital gain, the total amplification factor will be 4.8. This corresponds to adding 12 dB and 1.6 dB to give a total gain of 13.6 dB.

For optimum image quality, we recommend to set the total amplification as low as possible. If you need an amplification factor larger than 4 we recommend to set analog gain to 4 and then digital gain to reach the desired total amplification.

If you use analog gain and digital gain at the same time and also use the remove parameter limits for digital gain with digital gain parameter values below 264, the amplification factor for total gain will be 0 if the digital gain setting is 0.

10.1.2 Black Level

Adjusting the camera's black level will result in an offset to the pixel values output from the camera.

The camera's black level is determined by the BlackLevelRaw parameter with the black level selector set to All. All pixels in the sensor are affected by this setting.

If the camera is set for a pixel data format with an 8 bit depth, an increase of 16 in a black level setting will result in a positive offset of 1 in the pixel values output from the camera. And a decrease of 16 in a black level setting will result in a negative offset of 1 in the pixel values output from the camera.

If the camera is set for a pixel data format with a 12 bit depth, an increase of 1 in a black level setting will result in a positive offset of 1 in the pixel values output from the camera. A decrease of 1 in a black level setting will result in a negative offset of 1 in the pixel values output from the camera.

To set the black level:

1. Set the BlackLevelSelector parameter to All.
2. Set the BlackLevelRaw parameter to your desired value.

You can set the BlackLevelSelector and the BlackLevelRaw parameter values from within your application software by using the pylon API. The following code snippet illustrates using the API to set the parameter values:

```
// Set black level
camera.BlackLevelSelector.SetValue(BlackLevelSelector_All);
camera.BlackLevelRaw.SetValue(64);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

10.2 Remove Parameter Limits

For each camera feature, the allowed range of any associated parameter values is normally limited. The factory limits are designed to ensure optimum camera operation and, in particular, good image quality. For special camera uses, however, it may be helpful to set parameter values outside of the factory limits.

The remove parameter limits feature lets you remove the factory limits for parameters associated with certain camera features. When the factory limits are removed, the parameter values can be set within extended limits. Typically, the range of the extended limits is dictated by the physical restrictions of the camera's electronic devices, such as the absolute limits of the camera's variable gain control.

The values for any extended limits can be determined by using the Basler pylon Viewer or from within your application via the pylon API.

Currently, the limits can only be removed from these features and parameters:

- **Gain**: Removing this parameter limit reduces the lower limit for the Gain parameter to 0. For more information, see Section 10.1.1 on page 157.
- **Target Gray Value**: Removing this parameter limit increases the lower and upper limits of the TargetGrayValue parameter used for auto functions. For more information, see Section 10.4 on page 164.
- **Exposure Overhead:** Removing this parameter limit allows you to improve line acquisition. Also, it slightly increases the upper limit of the ExposureOverlapTimeMaxAbs parameter. For more information, see Section 8.5.1 on page 134.
- **Exposure Overlap Time Max:** Removing this parameter limit greatly increases the upper limit of the ExposureOverlapTimeMaxAbs parameter. For more information, see Section 8.3.3.3 on page 126.

To remove the limits for a parameter:

1. Use the ParameterSelector parameter to select the parameter whose limits you want to remove.
2. Set the value of the RemoveLimits parameter.

You can set the ParameterSelector and the RemoveLimits parameter values from within your application software by using the Basler pylon API. The following code snippet illustrates using the API to set the parameter values:

```
// Select the feature whose factory limits will be removed.
camera.ParameterSelector.SetValue(ParameterSelector_Gain);
// Remove the limits for the selected feature.
camera.RemoveLimits.SetValue(true);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

10.3 Image Area of Interest

The image area of interest (image AOI) feature lets you specify a portion of the sensor line. During operation, only the pixel information from the specified portion of the line is transmitted out of the camera. The image AOI also specifies the width of a frame. For more information about a frame and how to specify it, see Section 8.1 on page 74.

One of the main advantages of the image AOI feature is that decreasing the size of the AOI can increase the camera's maximum allowed acquisition line rate. For more information about how changing the image AOI size affects the maximum allowed line rate, see Section 8.5 on page 131.

The area of interest is referenced to the left end of the sensor array. The outer left pixel of the sensor is designated as pixel 0 (see Fig. 55).

The location and size of the area of interest is defined by declaring an offset X and a width. For example, suppose that you specify the offset X as 10 and the width as 25. The area of the array that is defined by these settings is shown in Figure 55. With these settings, the camera will read out and transmit pixel values for pixels 15 through 34.

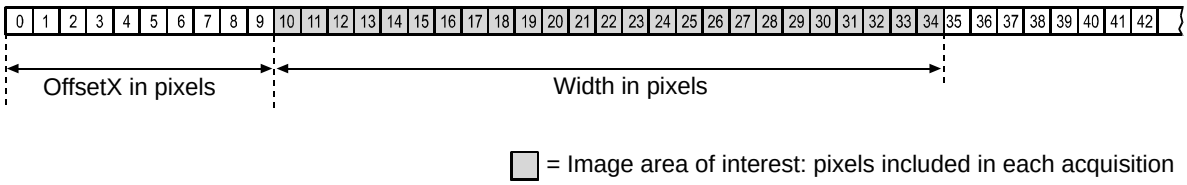


Fig. 55: Image Area of Interest

10.3.1 Setting the Image AOI

By default, the AOI is set to use the full resolution of the camera's sensor. You can change the size and the position of the AOI by changing the values of the camera's OffsetX and Width parameters. You can also enable automatic X centering using the CenterX parameter (see below).

When you are setting the camera's area of interest, you must follow this guideline: The sum of the OffsetX setting plus the Width setting must not exceed the width of the camera's sensor.

For example, on the raL2048-48gm, the sum of the OffsetX setting plus the Width setting must not exceed 2048.

Normally, the OffsetX and Width parameter settings refer to the physical line of the sensor. But if binning is enabled, these parameters are set in terms of a "virtual" line. For more information about binning, see Section 10.7 on page 181.

For information about the contribution of the image AOI feature to defining a frame and for code snippets illustrating how to set the OffsetX and Width parameters using the API, see Section 8.1 on page 74.

10.3.2 Automatic Image AOI X Centering

The image AOI feature includes Center X capabilities. When Center X is enabled, the camera will automatically center the image AOI on the sensor.

Setting Automatic X Centering

You can set the CenterX parameter value from within your application software by using the Basler pylon API. The following code snippet illustrates using the API to enable automatic image AOI centering.

```
// Enable automatic image AOI X centering.
camera.CenterX.SetValue(true);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

10.4 Auto Functions

10.4.1 Common Characteristics

Auto functions control image properties and are the "automatic" counterparts of certain features which normally require "manually" setting the related parameter values. Auto functions are particularly useful when an image property must be adjusted quickly to achieve a specific target value and when a specific target value must be kept constant during acquisition.

An auto function area of interest (auto function AOI) lets you designate a specific part of the image as the base for adjusting an image property. Each auto function uses the pixel data from the auto function AOI to automatically adjust a parameter value and, accordingly, to control the related image property.

On racer cameras, two auto functions are available: Gain Auto and Exposure Auto.

The auto functions automatically adjust a parameter value until the related image property reaches a target value. Note that the manual setting of the parameter value is not preserved. For example, when the gain auto function adjusts the gain parameter value, the manually set gain parameter value is not preserved.

For both auto functions, the target value can be set. Also, the limits between which the related parameter value will be automatically adjusted can be set. For example, the gain auto function lets you set an average gray value as a target value and also set a lower and an upper limit for the gain parameter value.

Both auto functions can operate at the same time. For more information, see the following sections describing the individual auto functions.

A target value for an image property can only be reached if it is in accord with all pertinent camera settings and with the general circumstances used for acquisition. Otherwise, the target value will only be approached.

For example, with a short exposure time, insufficient illumination, and a low setting for the upper limit of the gain parameter value, the Gain Auto function may not be able to achieve the current target average gray value setting for the image.

You can use an auto function when binning is enabled. An auto function uses the binned pixel data and controls the image property of the binned line.

For more information about binning, see Section 10.7 on page 181.

10.4.2 Auto Function Operating Modes

The following auto function modes of operation are available:

- "**Once**" mode of operation: When this mode of operation is selected, the parameter values are automatically adjusted until the related image property reaches the target value. After the automatic parameter value adjustment is complete, the auto function will automatically be set to Off and the new parameter value will be applied to the following frames.

  The parameter value can be changed by using the "once" mode of operation again, by using the "continuous" mode of operation, or by manual adjustment.

  If an auto function is set to the "once" operation mode and if the circumstances will not allow reaching a target value for an image property, the auto function will try to reach the target value for a maximum of 30 frames and will then be set to Off.

- "**Continuous**" mode of operation: When this mode of operation is selected, the parameter value is adjusted repeatedly while frames are acquired. Depending on the current frame rate, the automatic adjustments will usually be carried out for every or every other frame.

  The repeated automatic adjustment will proceed until the "once" mode of operation is used or until the auto function is set to Off, in which case the parameter value resulting from the latest automatic adjustment will operate, unless the parameter is manually adjusted.

- **Off**: When an auto function is set to Off, the parameter value resulting from the latest automatic adjustment will operate, unless the parameter is manually adjusted.

You can enable auto functions and change their settings while the camera is capturing frames ("on the fly").

If you have set an auto function to "once" or "continuous" operation mode while the camera was continuously capturing frames, the auto function will become effective with a short delay and the first few frames may not be affected by the auto function.

10.4.3 Auto Function AOI

The auto function AOI is used by the auto functions to automatically adjust a parameter value, and accordingly, to control the related image property (gain or exposure time).

To set the auto function AOI, you must specify

- a specific portion of the line using the AutoFunctionAOIWidth and AutoFunctionAOIOffsetX parameters. Setting these parameters is similar to setting the Width and OffsetX parameters of the image AOI (see Section 10.3 on page 162). The outer left pixel of the sensor line is designated as pixel 0.
- a specific portion of the frame height using the AutoFunctionAOIHeight parameter. You can set the auto function AOI to cover only a portion of the frame height, e.g. the first 10 lines of 100 lines.

Only the pixel data from this specified portion will be used for auto function control.

Example: You set the AutoFunctionAOIOffsetX parameter to 14, the AutoFunctionAOIWidth parameter to 5, and the AutoFunctionAOIHeight parameter to 6. The resulting Auto Function AOI is shown in Fig. 56.

For more information about setting the parameters, see Section 10.4.3.2 on page 169.

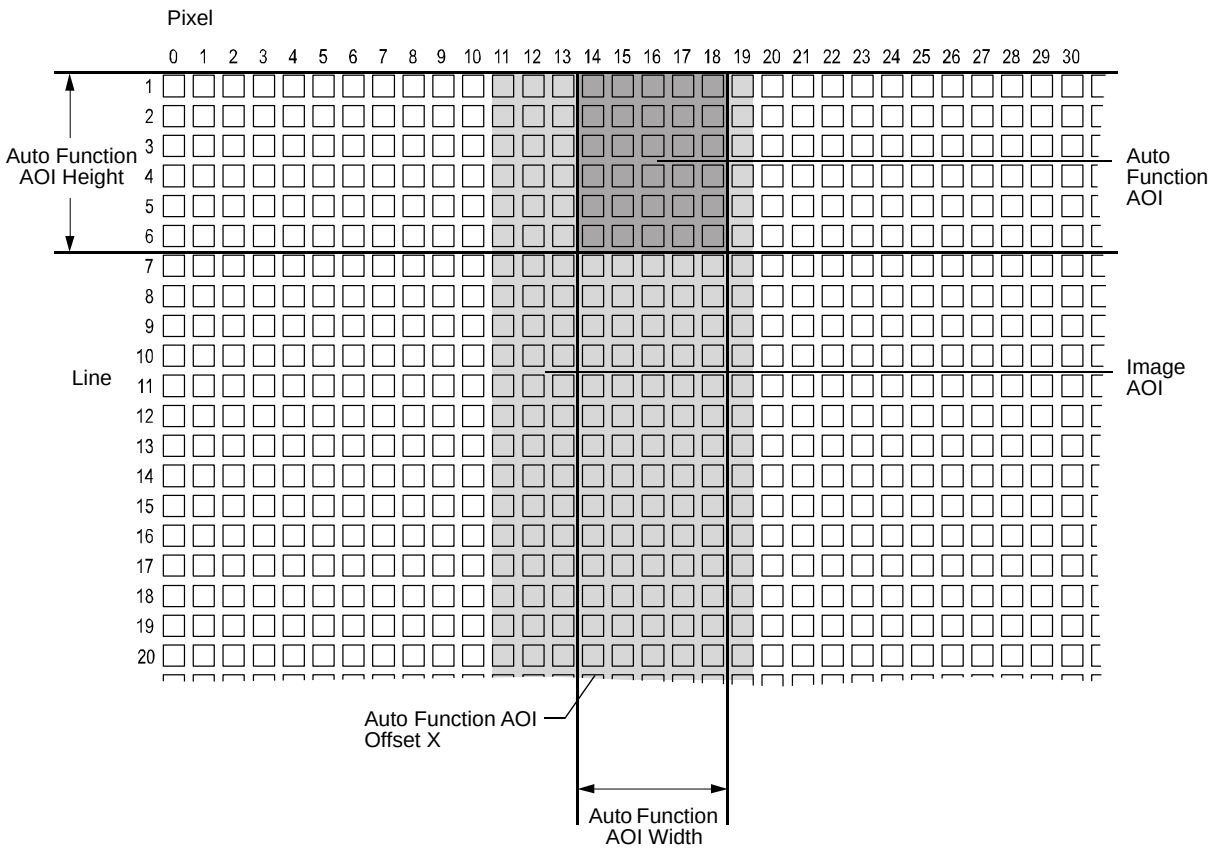


Fig. 56: Example Auto Function Area of Interest

10.4.3.1 Positioning of the Auto Function AOI Relative to the Image AOI

The size and position of the auto function AOI can be, but need not be, identical to the size and position of the image AOI.

The overlap between auto function AOI and image AOI determines whether and to what extent the auto function will control the related image property. Only the pixel data from the areas of overlap will be used by the auto function to control the image property of the entire frame.

Different degrees of overlap are illustrated in Fig. 57. The hatched areas in the figure indicate areas of overlap.

- If the auto function AOI is completely included in the image AOI (see (a) in Fig. 57), the pixel data from the auto function AOI will be used to control the image property.
- If the image AOI is completely included in the auto function AOI (see (b) in Fig. 57), only the pixel data from the image AOI will be used to control the image property.
- If the image AOI only partially overlaps the auto function AOI (see (c) in Fig. 57), only the pixel data from the area of partial overlap will be used to control the image property.
- If the auto function AOI does not overlap the image AOI (see (d) in Fig. 57), the Auto Function will **not** or only to a **limited** degree control the image property. For details, see the sections below, describing the individual auto functions.

We strongly recommend completely including the auto function AOI within the image AOI, or, depending on your needs, choosing identical positions and sizes for auto function AOI and image AOI.

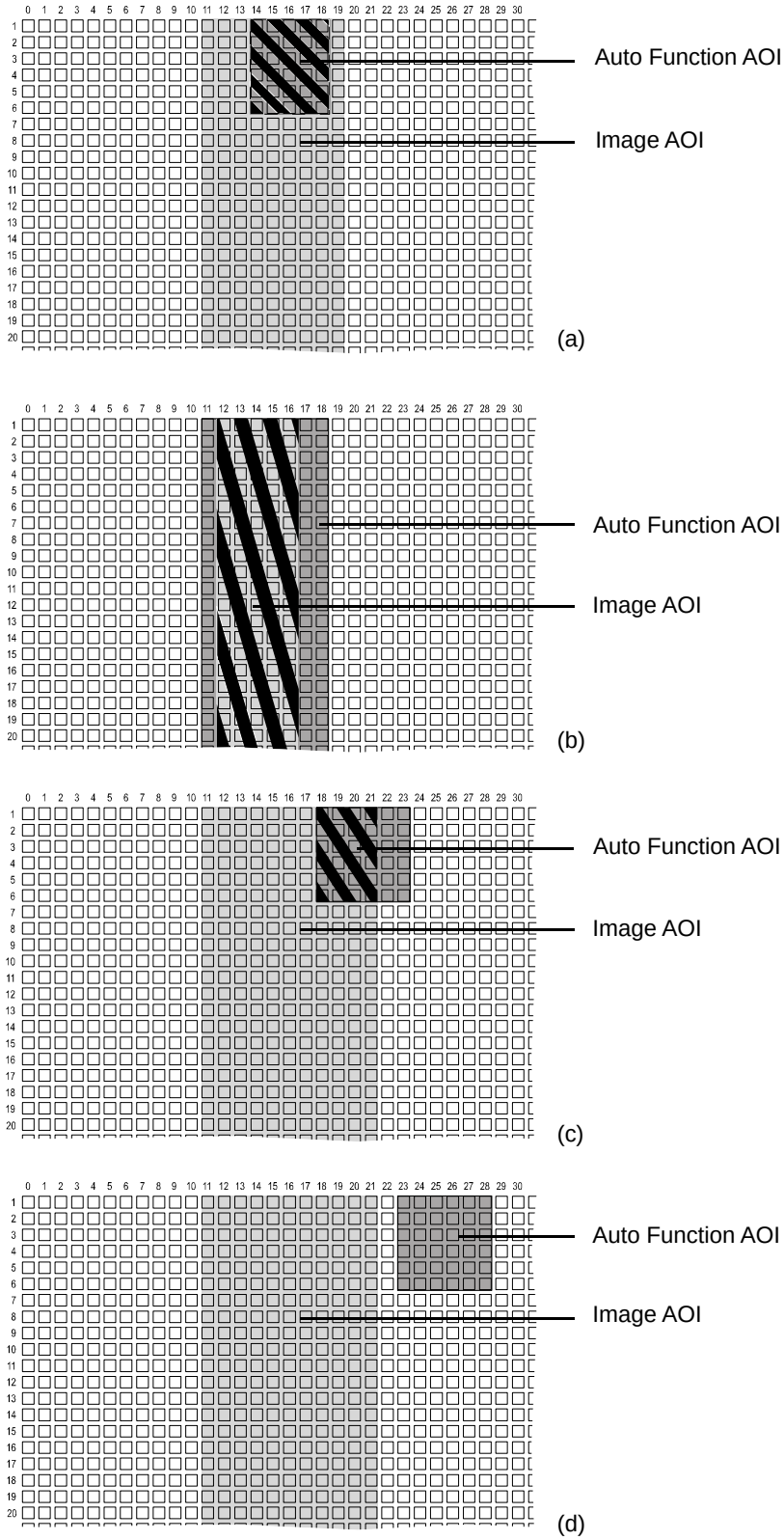


Fig. 57: Various Degrees of Overlap Between the Auto Function AOI and the Image AOI

10.4.3.2 Setting the Auto Function AOI

By default, the auto function AOI is set to the full width of the sensor line and to a height of 128 lines. You can change the size and the position of the auto function AOI by changing the value of the following parameters:

- AutoFunctionAOIOffsetX: determines the starting pixel (in horizontal direction) for the auto function AOI. The outer left pixel is designated as pixel 0.
- AutoFunctionAOIWidth: determines the width of the auto function AOI.
- AutoFunctionAOIHeight: determines the height of the auto function AOI.

All parameters can be set in increments of 1.

For general information about the parameters, see Section 10.4.1 on page 164.

When you are setting the auto function AOI, you must follow these guidelines:

Guideline	Notes
AutoFunctionAOIOffsetX + AutoFunctionAOIWidth < Width of camera sensor	**Example:** raL4096-24gm AutoFunctionAOIOffsetX + AutoFunctionAOIWidth < 4096.
AutoFunctionAOIHeight ≤ Height	The Height parameter defines the image AOI height. If AutoFunctionAOIHeight > Height, the effective auto function AOI height will be equal to Height. **Example:** If you set AutoFunctionAOIHeight to 400 and Height to 100, the effective auto function AOI height will be 100.

Normally, the AutoFunctionAOIOffsetX and the AutoFunctionAOIWidth parameter settings refer to the physical line of the sensor. But if binning is enabled, these parameters are set in terms of a "virtual" line. For more information about binning, see Section 10.7 on page 181.

Depending on your firmware version, you may be able to choose from multiple auto function AOIs (AOI 1, AOI 2, ...) using the AutoFunctionAOISelector parameter. However, leave the auto function AOI 1 selected at all times and do not use the other auto function AOIs.

You can set the parameter values for the auto function AOI from within your application software by using the Basler pylon API:

```
// Select Auto Function AOI 1
camera.AutoFunctionAOISelector.SetValue(AutoFunctionAOISelector_AOI1);
// Set the position and size of the auto function AOI (sample values)
camera.AutoFunctionAOIOffsetX.SetValue(20);
```

```
camera.AutoFunctionAOIWidth.SetValue(500);
camera.AutoFunctionAOIHeight.SetValue(300);
```

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

10.4.4 Gain Auto

Gain Auto is the "automatic" counterpart to manually setting the GainRaw parameter. When the gain auto function is operational, the camera will automatically adjust the GainRaw parameter value within set limits until a target average gray value for the pixel data from the auto function AOI is reached.

The gain auto function can be operated in the "once" and continuous" modes of operation.

If the auto function AOI does not overlap the image AOI (see Section 10.4.3.1 on page 167), the pixel data from the auto function AOI will not be used to control the gain. Instead, the current manual setting for the GainRaw parameter value will control the gain.

The gain auto function and the exposure auto function can be used at the same time. In this case, however, you must also set the auto function profile feature.

For more information about

- setting the gain "manually", see Section 10.1 on page 157.
- the auto function profile feature, see Section 10.4.7 on page 174.

The limits within which the camera will adjust the GainRaw parameter are defined by the AutoGainRawUpperLimit and the AutoGainRawLowerLimit parameters. The minimum and maximum allowed settings for the AutoGainRawUpperLimit and AutoGainRawLowerLimit parameters depend on the current pixel data format, on the current settings for binning, and on whether or not the parameter limits for manually setting the gain feature are disabled.

The AutoTargetValue parameter defines the target average gray value that the gain auto function will attempt to achieve when it is automatically adjusting the GainRaw value. The target average gray value can range from 50 (black) to 205 (white) when the camera is set for an 8-bit pixel format or from 800 (black) to 3280 (white) when the camera is set for a 12-bit pixel format.

To set the gain auto functionality:

1. Set the value of the AutoFunctionAOIOffsetX, AutoFunctionAOIWidth, and AutoFunctionAOIHeight parameters.
2. Set the GainSelector parameter to DigitalAll.
3. Set the value of the AutoGainRawLowerLimit and AutoGainRawUpperLimit parameters.
4. Set the value of the AutoTargetValue parameter.
5. Set the value of the GainAuto parameter for the "once" or the "continuous" mode of operation.

You can set the gain auto functionality from within your application software by using the pylon API. The following code snippets illustrate using the API to set the exposure auto functionality:

```
// Set the position and size of the auto function AOI (sample values)
camera.AutoFunctionAOIOffsetX.SetValue(20);
camera.AutoFunctionAOIWidth.SetValue(500);
camera.AutoFunctionAOIHeight.SetValue(300);

// Select gain all and set the upper and lower gain limits for the
// gain auto function
camera.GainSelector.SetValue(GainSelector_DigitalAll);
camera.AutoGainRawLowerLimit.SetValue(camera.GainRaw.GetMin());
camera.AutoGainRawUpperLimit.SetValue(camera.GainRaw.GetMax());

// Set the target gray value for the gain auto function
// (If exposure auto is enabled, this target is also used for
// exposure auto control.)
camera.AutoTargetValue.SetValue(128);

// Set the mode of operation for the gain auto function
camera.GainAuto.SetValue(GainAuto_Continuous);
```

You can also use the Basler pylon Viewer application to easily set the parameters.

For general information about auto functions, see Section 10.4.1 on page 164.

For more information about

- the pylon API and the pylon Viewer, see Section 3.1 on page 26.
- the auto function AOI and how to set it, see Section 10.4.3 on page 166.

10.4.5 Exposure Auto

The exposure auto function will not work if the camera's exposure mode is set to trigger width. For more information about the trigger width exposure mode, see Section 8.2.4.2 on page 87.

Exposure Auto is the "automatic" counterpart to manually setting the ExposureTimeAbs parameter. The exposure auto function automatically adjusts the ExposureTimeAbs parameter value within set limits until a target average gray value for the pixel data from the auto function AOI is reached.

The exposure auto function can be operated in the "once" and continuous" modes of operation.

If the auto function AOI does not overlap the image AOI (see Section 10.4.3.1 on page 167), the pixel data from the auto function AOI will not be used to control the exposure time. Instead, the current manual setting of the ExposureTimeAbs parameter value will control the exposure time.

The exposure auto function and the gain auto function can be used at the same time. In this case, however, you must also set the auto function profile feature.

For more information about

- setting the exposure time "manually", see Section 8.2.5 on page 91.
- the auto function profile feature, see Section 10.4.7 on page 174.

The limits within which the camera will adjust the ExposureTimeAbs parameter are defined by the AutoExposureTimeAbsUpperLimit and the AutoExposureTimeAbsLowerLimit parameters. The current minimum and the maximum allowed settings for the AutoExposureTimeAbsUpperLimit parameter and the AutoExposureTimeAbsLowerLimit parameters depend on the minimum allowed and maximum possible exposure time for your camera model.

The AutoTargetValue parameter defines the target average gray value that the exposure auto function will attempt to achieve when it is automatically adjusting the ExposureTimeAbs value. The target average gray value can range from 50 (black) to 205 (white) when the camera is set for an 8-bit pixel format or from 800 (black) to 3280 (white) when the camera is set for a 12-bit pixel format.

If the AutoExposureTimeAbsUpperLimit parameter is set to a sufficiently high value, the camera's frame rate may be decreased.

To set the exposure auto functionality:

1. Set the value of the AutoFunctionAOIOffsetX, AutoFunctionAOIWidth, and AutoFunctionAOIHeight parameters.
2. Set the value of the AutoExposureTimeAbsLowerLimit and AutoExposureTimeAbsUpperLimit parameters.
3. Set the value of the AutoTargetValue parameter.
4. Set the value of the ExposureAuto parameter for the "once" or the "continuous" mode of operation.

You can set the exposure auto functionality from within your application software by using the pylon API. The following code snippets illustrate using the API to set the parameters:

```
// Set the position and size of the auto function AOI (sample values)
camera.AutoFunctionAOIOffsetX.SetValue(20);
camera.AutoFunctionAOIWidth.SetValue(500);
camera.AutoFunctionAOIHeight.SetValue(300);

// Set the exposure time limits for exposure auto control
camera.AutoExposureTimeAbsLowerLimit.SetValue(1000);
camera.AutoExposureTimeAbsUpperLimit.SetValue(5000);

// Set the target gray value for the exposure auto function
// (If gain auto is enabled, this target is also used for
// gain auto control.)
camera.AutoTargetValue.SetValue(128);

// Set the mode of operation for the exposure auto function
camera.ExposureAuto.SetValue(ExposureAuto_Continuous);
```

You can also use the Basler pylon Viewer application to easily set the parameters.

For information about

- the pylon API and the pylon Viewer, see Section 3.1 on page 26
- the auto function AOI and how to set it, see Section 10.4.3 on page 166
- minimum allowed and maximum possible exposure time, see Section 8.2.5 on page 91.

For general information about auto functions, see Section 10.4.1 on page 164.

10.4.6 Gray Value Adjustment Damping

The gray value adjustment damping controls the rate by which pixel gray values are changed when Exposure Auto or Gain Auto or both are enabled.

If an adjustment damping factor is used, the gray value target value is not immediately reached, but after a certain delay. This can be useful, for example, when objects move into the camera's view area and where the light conditions are gradually changing due to the moving objects.

By default, the gray value adjustment damping is set to 0.75. This is a setting where the damping control is as stable and quick as possible.

Setting the Adjustment Damping

The adjustment damping is determined by the value of the GrayValueAdjustmentDampingAbs parameter.

The higher the value, the lower the adjustment damping is, i.e.

- the sooner the target value will be reached,
- the adaptation is realized over a smaller number of frames.

You can set the gray value adjustment damping from within your application software by using the pylon API. The following code snippets illustrate using the API to set the gray value adjustment damping:

```
camera.GrayValueAdjustmentDampingAbs.SetValue(0.5);
```

You can also use the Basler pylon Viewer application to easily set the parameters.

10.4.7 Auto Function Profile

If you want to use the gain auto function and the exposure auto function at the same time,

- the auto function profile feature also takes effect.
  The auto function profile specifies whether the gain or the exposure time will be kept as low as possible when the camera is making automatic adjustments to achieve a target average gray value for the pixel data. By default, the auto function profile feature minimizes exposure time.
- you should set both functions for the continuous mode of operation.

To use the gain auto function and the exposure auto function at the same time:

1. Set the value of the AutoFunctionProfile parameter to specify whether gain or exposure time will be minimized during automatic adjustments.
2. Set the value of the GainAuto parameter to the "continuous" mode of operation.
3. Set the value of the ExposureAuto parameter to the "continuous" mode of operation.

You can set the auto function profile from within your application software by using the pylon API. The following code snippet illustrates using the API to set the auto function profile. As an example, Gain is set to be minimized during adjustments:

```
// Use GainAuto and ExposureAuto simultaneously
camera.AutoFunctionProfile.SetValue(AutoFunctionProfile_GainMinimum);
camera.GainAuto.SetValue(GainAuto_Continuous);
camera.ExposureAuto.SetValue(ExposureAuto_Continuous);
```

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

10.5 Event Reporting

Event reporting is available on the camera. With event reporting, the camera can generate an "event" and after some intermediate steps transmit a related event message to the PC whenever a specific situation has occurred.

The camera can generate and transmit events for the following types of situations:

- Overtriggering of the acquisition start trigger has occurred (AcquisitionStartOvertriggerEventData).
  This happens if the camera receives an acquisition start trigger while it is currently not in the waiting for an acquisition start trigger status.
- Overtriggering of the frame start trigger has occurred (FrameStartOvertriggerEventData).
  This happens if the camera receives a frame start trigger while it is currently not in the waiting for a frame start trigger status.
- Overtriggering of the line start trigger has occurred (LineStartOvertriggerEventData).
  This happens if the camera receives a line start trigger while it is currently in the process of acquiring a line.
- A frame timeout has occurred (FrameTimeoutEventData).
  This happens if the acquisition of a frame is not completed within a set period, provided frame timeout is enabled and configured. For more information, see the Frame Timeout section.
- An event overrun has occurred (EventOverrunEventData).
  This situation is explained later in this section.

An Example of Event Reporting

An example related to the Frame Start Overtrigger event illustrates how event reporting works. The example assumes that your system is set for event reporting (see below) and that the camera has received a frame start trigger while it is currently in the process of acquiring a frame. In this case:

1. A Frame Start Overtrigger event is created. The event contains the event in the strict sense and supplementary information:

   An *Event Type Identifier*. In this case, the identifier would show that a frame start overtrigger type event has occurred.

   A *Stream Channel Identifier*. Currently this identifier is always 0.

   A *Timestamp*. This is a timestamp indicating when the event occurred. (The time stamp timer starts running at power off/on or at camera reset. The unit for the timer is "ticks" where one tick = 8 ns. The timestamp is a 64 bit value.)
2. The event is placed in an internal queue in the camera.
3. As soon as network transmission time is available, an event message will be sent to the PC. If only one event is in the queue, the message will contain the single event. If more than one event is in the queue, the message will contain multiple events.
   a. After the camera sends an event message, it waits for an acknowledgement. If no acknowledgement is received within a specified timeout, the camera will resend the event message. If an acknowledgement is still not received, the timeout and resend mechanism will repeat until a specified maximum number of retries is reached. If the maximum number of retries is reached and no acknowledge has been received, the message will be dropped.

During the time that the camera is waiting for an acknowledgement, no new event messages can be transmitted.

4. Event reporting involves some further software-related steps and settings to be made. For more information, see the "Camera Events" code sample included with the pylon software development kit.

The Event Queue

As mentioned in the example above, the camera has an event queue. The intention of the queue is to handle short term delays in the camera's ability to access the network and send event messages. When event reporting is working "smoothly", a single event will be placed in the queue and this event will be sent to the PC in an event message before the next event is placed in the queue. If there is an occasional short term delay in event message transmission, the queue can buffer several events and can send them within a single event message as soon as transmission time is available.

However, if you are operating the camera at high line rates, the camera may be able to generate and queue events faster than they can be transmitted and acknowledged. In this case:

1. The queue will fill and events will be dropped.
2. An event overrun will occur.
3. Assuming that you have event overrun reporting enabled, the camera will generate an "event overrun event" and place it in the queue.
4. As soon as transmission time is available, an event message containing the event overrun event will be transmitted to the PC.

The event overrun event is simply a warning that events are being dropped. The notification contains no specific information about how many or which events have been dropped.

Setting Your System for Event Reporting

Event reporting must be enabled in the camera and some additional software-related settings must be made. This is described in the "Camera Events" code sample included with the pylon software development kit.

Event reporting must be specifically set up for each type of event using the parameter name of the event and of the supplementary information. The following table lists the relevant parameter names:

Event	Event Parameter Name	Supplementary Information Parameter Name
Acquisition Start Overtrigger	AcquisitionStartOvertriggerEventData	AcquisitionStartOvertriggerEventStreamChannelIndex
		AcquisitionStartOvertriggerEventTimestamp
Frame Start Overtrigger	FrameStartOvertriggerEventData	FrameStartOvertriggerEventStreamChannelIndex
		FrameStartOvertriggerEventTimestamp
Line Start Overtrigger	LineStartOvertriggerEventData	LineStartOvertriggerEventChannelIndex
		LineStartOvertriggerEventTimestamp
Frame Timeout	FrameTimeoutEventData	FrameTimeoutEventStreamChannelIndex
		FrameTimeoutEventTimestamp
Event Overrun	EventOverrunEventData	EventOverrunEventStreamChannelIndex
		EventOverrunEventTimestamp

Table 12: Parameter Names of Events and Supplementary Information

You can enable event reporting and make the additional settings from within your application software by using the pylon API. The pylon software development kit includes a "Camera Events" code sample that illustrates the entire process.

For more detailed information about using the pylon API, refer to the Basler pylon Programmer’s Guide and API Reference.

10.6 Luminance Lookup Table

The type of electronics used on the camera allow the camera's sensor to acquire pixel values at a 12 bit depth. Normally, when a camera is set for a 12 bit pixel data format, the camera uses the actual 12 bit pixel values reported by the sensor.

The luminance lookup table feature lets you create a custom 12 bit to 12 bit lookup table that maps the actual 12 bit values output from the sensor to substitute 12 bit values of your choice. When the lookup table is enabled, the camera will replace the actual pixel values output from the sensor with the substitute values from the table.

The lookup table has 4096 indexed locations with a 12 bit value stored at each index. The values stored in the table are used like this:

- When the sensor reports that a pixel has an actual 12 bit value of 0, the substitute 12 bit value stored at index 0 will replace the actual pixel value.
- The numbers stored at indices 1 through 7 are not used.
- When the sensor reports that a pixel has an actual 12 bit value of 8, the substitute 12 bit value stored at index 8 will replace the actual pixel value.
- The numbers stored at indices 9 through 15 are not used.
- When the sensor reports that a pixel has an actual 12 bit value of 16, the substitute 12 bit value stored at index 16 will replace the actual pixel value.
- The numbers stored at indices 17 through 23 are not used.
- When the sensor reports that a pixel has an actual 12 bit value of 24, the substitute 12 bit value stored at index 24 will replace the actual pixel value.
- And so on.

As you can see, the table does not include a defined 12 bit substitute value for every actual pixel value that the sensor can report. If the sensor reports an actual pixel value that is between two values that have a defined substitute, the camera performs a straight line interpolation to determine the substitute value that it should use. For example, assume that the sensor reports an actual pixel value of 12. In this case, the camera would perform a straight line interpolation between the substitute values at index 8 and index 16 in the table. The result of the interpolation would be used by the camera as the substitute.

Another thing to keep in mind about the table is that index 4088 is the last index that will have a defined substitute value associated with it (the values at indices 4089 through 4095 are not used.) If the sensor reports an actual value greater than 4088, the camera will not be able to perform an interpolation. In cases where the sensor reports an actual value greater than 4088, the camera simply uses the 12 bit substitute value from index 4088 in the table.

The advantage of the luminance lookup table feature is that it lets a user customize the response curve of the camera. The graphs below represent the contents of two typical lookup tables. The first graph is for a lookup table where the values are arranged so that the output of the camera increases linearly as the actual sensor output increases. The second graph is for a lookup table where the values are arranged so that the camera output increases quickly as the actual sensor output moves from 0 through 2048 and increases gradually as the actual sensor output moves from 2049 through 4096.

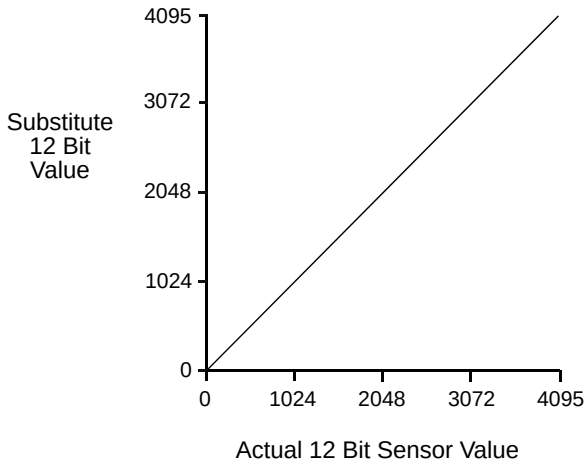


Fig. 58: Lookup Table with Values Mapped in a Linear Fashion

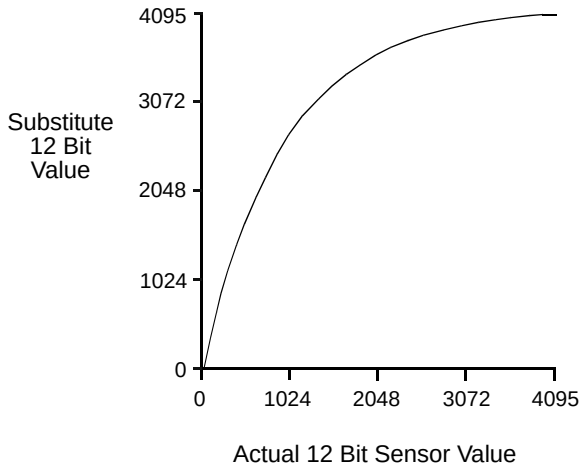


Fig. 59: Lookup Table with Values Mapped for Higher Camera Output at Low Sensor Readings

Using the Luminance Lookup Table to Get 8 Bit Output

As mentioned above, when the camera is set for a 12 bit pixel data format, the lookup table can be used to perform a 12 bit to 12 bit substitution. The lookup table can also be used in 12 bit to 8 bit fashion. To use the table in 12 bit to 8 bit fashion, you enter 12 bit substitution values into the table and enable the table as you normally would. But instead of setting the camera for a 12 bit pixel data format, you set the camera for an 8 bit format (such as Mono 8). In this situation, the camera will first use the values in the table to do a 12 bit to 12 bit substitution. It will then truncate the lowest 4 bits of the substitute value and will transmit the remaining 8 highest bits.

Changing the Values in the Luminance Lookup Table and Enabling the Table

You can change the values in the luminance lookup table (LUT) and enable the use of the lookup table by doing the following:

1. Use the LUTSelector parameter to select a lookup table. (Currently there is only one lookup table available, i.e., the "luminance" lookup table described above.)
2. Use the LUTIndex parameter to select an index number.
3. Use the LUTValue parameter to enter the substitute value that will be stored at the index number that you selected in step 2.
4. Repeat steps 2 and 3 to enter other substitute values into the table as desired.
5. Use the LUTEnable parameter to enable the table.

You can set the LUTSelector, the LUTIndex parameter, and the LUTValue parameter from within your application software by using the pylon API. The following code snippet illustrates using the API to set the parameter values:

```
// Select the lookup table
camera.LUTSelector.SetValue(LUTSelector_Luminance);

// Write a lookup table to the device.
// The following lookup table causes an inversion of the sensor values
// (bright -> dark, dark -> bright)
for (int i = 0; i < 4096; i += 8)
{
   camera.LUTIndex.SetValue(i);
   camera.LUTValue.SetValue(4095 - i);
}
// Enable the lookup table
camera.LUTEnable.SetValue(true);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer’s Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

10.7 Binning

Binning increases the camera's response to light by summing the charges from adjacent pixels into one pixel.

With horizontal binning, the charges of 2, 3, or a maximum of 4 adjacent pixels are summed and are reported out of the camera as a single pixel. Fig. 60 illustrates horizontal binning.

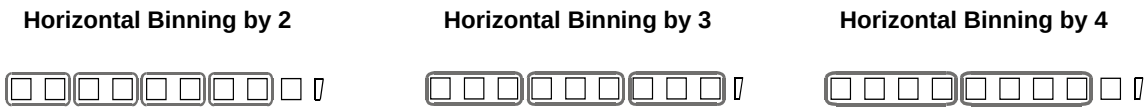


Fig. 60: Horizontal Binning

Setting Binning

You can enable horizontal binning by setting the BinningHorizontal parameter. Setting the parameter's value to 2, 3, or 4 enables horizontal binning by 2, horizontal binning by 3, or horizontal binning by 4 respectively. Setting the parameter's value to 1 disables horizontal binning.

You can set the BinningVertical or the BinningHorizontal parameter value from within your application software by using the Basler pylon API. The following code snippet illustrates using the API to set the parameter values:

```
// Enable horizontal binning by 4
camera.BinningHorizontal.SetValue(4);

// Disable horizontal binning
camera.BinningHorizontal.SetValue(1);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

10.8 Gamma Correction

The gamma correction feature lets you modify the brightness of the pixel values output by the camera's sensor to account for a non-linearity in the human perception of brightness. To accomplish the correction, a gamma correction factor (γ) is applied to the brightness value (Y) of each pixel according to the following formula:

$$Y_{corrected} = \left(\frac{Y_{uncorrected}}{Y_{max}}\right)^{\gamma} \times Y_{max}$$

The formula uses uncorrected and corrected pixel brightnesses that are normalized by the maximum pixel brightness. The maximum pixel brightness equals 255 for 8 bit output and 4095 for 12 bit output.

When the gamma correction factor is set to 1, the output pixel brightness will not be corrected.

A gamma correction factor between 0 and 1 will result in increased overall brightness, and a gamma correction factor greater than 1 will result in decreased overall brightness.

In all cases, black (output pixel brightness equals 0) and white (output pixel brightness equals 255 at 8 bit output and 4095 at 12 bit output) will not be corrected.

Enabling Gamma Correction and Setting the Gamma

You can enable or disable the gamma correction feature by setting the value of the GammaEnable parameter.

When gamma correction is enabled, the correction factor is determined by the value of the Gamma parameter. The Gamma parameter can be set in a range from 0 to 3.99902. So if the Gamma parameter is set to 1.2, for example, the gamma correction factor will be 1.2.

You can set the GammaEnable and Gamma parameter values from within your application software by using the Basler pylon API. The following code snippet illustrates using the API to set the parameter values:

```
// Enable the Gamma feature
camera.GammaEnable.SetValue(true);

// Set the Gamma value to 1.2
camera.Gamma.SetValue(1.2);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

10.9 Shading Correction

Two types of shading correction are available on the camera, offset shading correction and gain shading correction. You can set the camera to only perform offset shading correction, to only perform gain shading correction, or to perform both types of shading correction.

10.9.1 Offset Shading Correction

When a line scan camera with a digital sensor captures a line in complete darkness, all of the pixel values in the line should be near zero and they should be equal. In practice, slight variations in the performance of the pixels in the sensor will cause some variation in the pixel values output from the camera when the camera is capturing lines in darkness. Offset shading correction (also known as dark signal non-uniformity (DSNU) correction) corrects for this type of variation.

Offset shading correction works by adding an individual gray value to each pixel value in the acquired lines. The gray values used for correction are included in a "shading file", commonly also referred to as a "shading set". In order to use offset shading correction, the user must enable offset shading correction and activate the related defaultshading file or the previously created usershading file (see below).

10.9.2 Gain Shading Correction

When a line scan camera with a digital sensor captures a line with the camera viewing a uniform light-colored target in bright light, all of the pixel values in the line should be near their maximum gray value and they should be equal. In practice, slight variations in the performance of the pixels in the sensor, variations in the optics, and variations in the lighting will cause some variation in the pixel values output from the camera. Gain shading correction (also known as photo response non-uniformity (PRNU) correction) corrects for this type of variation.

Gain shading correction works by applying an individual multiplier to each pixel value in the acquired lines. The multipliers used for correction are included in a "shading file", commonly also referred to as a "shading set". In order to use gain shading correction, the user must enable gain shading correction and activate the related defaultshading file or the previously created usershading file (see below).

10.9.3 Default Shading Set File and User Shading Set File

For each type of shading correction, two types of shading set files are available in the camera's nonvolatile memory:

- The first type of shading set file is called the "defaultshading" file. One "defaultshading" file is available for offset shading correction and another one for gain shading correction.

  A "defaultshading" file contains a complete collection of the values needed to perform either offset shading or gain shading correction. The values in the files are generated during the camera's factory setup procedure and they essentially serve as default shading correction values. The values are optimized for performing shading correction with "standard" optics and lighting. Using the factory values will give you reasonable shading correction performance in most situations. One advantage of the factory values is that they serve as a good default. The "defaultshading" files are in a protected area of the camera's memory and can't be changed.

  "Defaultshading" files are always enabled. When "usershading" files are also enabled, they will supplement the default shading correction by modifying the default correction values.

- The second type of shading set file is called the "usershading" file. One "usershading" file is available for offset shading correction and another one for gain shading correction.

  Like a "defaultshading" file, a "usershading" file also holds a complete collection of the values needed to perform either offset shading or gain shading correction. The values stored in the files must, however, be generated by the camera user. When the values are generated the camera must operate under its real world conditions. The "usershading" files contain the shading correction values that will normally be used for day-to-day camera operation. A procedure describing how to generate the values in the files appears below.

"Defaultshading" files are always enabled. When "usershading" files are also enabled, they will supplement the default shading correction by modifying the default correction values.

10.9.3.1 Creating a "Usershading" File

To create a "usershading" file and enable it, you must take the steps listed below. We strongly recommend that you read through all of the steps and read all of the other information in this section before you attempt to do shading correction.

The steps below are intended to give you the basic knowledge needed to create a "usershading" file and to activate it. **A code sample that includes the complete details of how to create a usershading file and how to enable shading correction on a camera is included with the Basler pylon SDK.**

When you create a "usershading" file you must make sure to create correction values for all of the pixels in the sensor's line regardless of how you plan to use the camera during normal operation.

Creating a "Usershading" File for Offset Shading Correction

Creating a "usershading" file for offset shading correction will overwrite any "usershading" file for offset shading correction that is already in the camera's memory.

If you want to preserve the previous "usershading" file save it to your PC before creating the new "usershading" file.

For information about saving a "usershading" file to the PC, see Section 10.9.3.2 on page 187.

To create a "usershading" file for offset shading correction:

1. Adjust the lighting, optics, line rate, exposure time control mode, exposure time, gain, and camera temperature as you would for normal operation.
2. Prevent light from striking the camera's sensor: Cover the camera lens, close the iris in the camera lens, or darken the room so that the camera will be capturing lines in complete darkness.
3. Set the camera's OffsetX and Width parameters so that the entire width of the sensor will be used during frame acquisition.
   You can create a "usershading" file for an AOI that is narrower than the entire width of the sensor. In this case, however, the "usershading" file will only apply to the narrower AOI or to smaller included AOIs. We recommend using the entire width of the sensor.
4. Select Offset Shading in the Basler pylon Viewer or via the Basler pylon API.
5. Select "usershading" file in the Basler pylon Viewer or via the Basler pylon API.
6. Go to the **Create** enumeration in the Basler pylon Viewer and select **Once** or send a create command via the Basler pylon API.
7. Perform at least 128 line acquisitions.

   To ease acquisition of the required number of lines we recommend to set the line start trigger mode to off for automatic line start triggering and to set the Height parameter for the frame to at least 128.

   For more information about the line start trigger mode, see Section 8.2.4 on page 86.

   For more information about defining a frame, see Section 8.1 on page 74.

   After 128 line acquisitions are completed the camera creates the "usershading" file automatically. The "usershading" file is stored in the camera's non-volatile memory and is not lost if the camera power is switched off.

Any time you make a change to the line rate, exposure time control mode, exposure time, gain, or camera temperature, you must create a new "usershading" file for offset shading correction. Using an out of date "usershading" file can result in poor image quality.

Creating a "Usershading" File for Gain Shading Correction

Creating a "usershading" file for gain shading correction will overwrite any "usershading" file for gain shading correction that is already in the camera's memory.

If you want to preserve the previous "usershading" file save it to your PC before creating the new "usershading" file.

For information about saving a "usershading" file to the PC, see Section 10.9.3.2 on page 187.

To create a "usershading" file for gain shading correction:

1. Adjust the lighting, optics, line rate, exposure time control mode, exposure time, gain, and camera temperature as you would for normal operation.
2. Place a uniform white target in the field of view of the camera.
3. Set the camera's OffsetX and Width parameters so that the entire width of the sensor lines will be used during frame acquisition.
4. Perform several line acquisitions and examine the pixel values returned from the camera. The pixel values for the brightest pixels in each line should be about 90 to 95 % of maximum (i.e., if the camera is set for 8 bit output, the pixels should be from 90 to 95 % of 255).
   a. If the values for the brightest pixels are at 90 to 95 % of maximum, go on to step 5.
   b. If the values for the brightest pixels are not at 90 to 95 % of the maximum, adjust your lighting and/or lens aperture to achieve 90 to 95 %.
5. Perform several line acquisitions and examine the pixel values returned from the camera. In each line, the values for the darkest pixels must be greater than 1/4 of the values for the brightest pixels. (If the values for the darkest pixels are less than 1/4 of the values for the brightest, the camera will not be able to fully correct for shading variations.)
   a. If the values for the darkest pixels are greater than 1/4 of the values for the brightest, go on to step 6.
   b. If the values for the darkest pixels are less than 1/4 of the values for the brightest pixels, it usually indicates extreme variations in lighting or poor quality optics. Make corrections as required.
6. Select Gain Shading in the Basler pylon Viewer or via the Basler pylon API.
7. Select "usershading" file in the Basler pylon Viewer or via the Basler pylon API.
8. Go to the **Create** enumeration in the Basler pylon Viewer and select **Once** or send a create command via the Basler pylon API.
9. Perform at least 128 line acquisitions.

   To ease acquisition of the required number of lines we recommend to set the line start trigger mode to off for automatic line start triggering and to set the Height parameter for the frame to at least 128.

   For more information about the line start trigger mode, see Section 8.2.4 on page 86.

   For more information about defining a frame, see Section 8.1 on page 74.

After 128 line acquisitions are completed the camera creates the "usershading" file automatically. The "usershading" file is stored in the camera's non-volatile memory and is not lost if the camera power is switched off.

Any time you make a change to the optics or lighting or if you change the camera's gain settings or exposure mode, you must create a new "usershading" file. Using an out of date "usershading" file can result in poor image quality.

10.9.3.2 Working with Shading Sets

Once you have created shading set files, you can use the following pylon API functions to work with the shading sets:

Shading Selector - is used to select the type of shading correction to configure, i.e. offset shading correction or gain shading correction.

Shading Create - is used to create a "usershading" file. The enumeration allows selecting the settings **Off** and **Once**.

Shading Enable - is used to enable and disable the selected type of shading correction.

Shading Set Selector - is used to select the shading set to which the activate and the create enumeration commands will be applied.

Shading Set Activate - is used to activate the selected shading set. "Activate" means that the shading set will be copied from the camera's non-volatile memory into it's volatile memory. When the shading correction feature is enabled, the shading set in the volatile memory will be used to perform shading correction.

Shading Set Default Selector - is used to select the shading set that will be loaded into the camera's volatile memory during camera bootup.

Shading Status - is used to determine the error status of operations such as Shading Set Activate. The following error statuses may be indicated:

No error - the last operation performed was successful.

Startup Set error - there was a problem with the default shading set.

Activate error - the selected shading set could not be loaded into the volatile memory.

Create error - and error occurred during the attempt of creating a "usershading" file.

The use of the pylon API functions listed above is illustrated in the shading correction sample code included with the pylon SDK.

You can also use the Shading parameters group in the Basler pylon Viewer application to access these functions.

And you can use the File Access selection in the Camera menu of the Viewer to save a shading set file to a PC and to upload a shading set file from the PC to the camera.

10.10 Trigger Delay

The trigger delay feature lets you specify a delay that will be applied between the receipt of a hardware acquisition start trigger or frame start trigger and it becoming effective.

The trigger delay may be specified as a time interval in the range from 0 to 1000000 µs (equivalent to 1 s) or as a number of consecutive line start triggers where the maximum number depends on the camera model. When the delay is set to 0 µs or 0 line start triggers, no delay will be applied.

The trigger delay will not operate when the camera is triggered by your application software and when the camera operates in continuous frame mode (free run).

When setting the trigger delay you must specify the kind of trigger to be delayed (acquisition start or frame start trigger) and the extend of the delay expressed as a time interval or as a number of consecutive line start triggers.

You can set the trigger delay from within your application software by using the pylon API. As examples, the following code snippets illustrate using the API to set the delay for the acquisition start trigger to 1000 µs and to set the delay for the frame start trigger to 100 line start triggers:

```
// Trigger delay
camera.TriggerSource = AcquisitionStart;
double TriggerDelay_us = 1000.0   // 1000us == 1ms == 0.001s;
camera.TriggerDelaySource = TriggerDelay_us;
camera.TriggerDelayAbs.SetValue(TriggerDelay_us);

// Trigger delay
camera.TriggerSource = FrameStart;
int NumberLineTriggers = 100;
camera.TriggerDelaySource = LineTrigger;
camera.TriggerDelayLineTriggerCount.SetValue(NumberLineTriggers);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer’s Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

10.11 Precision Time Protocol (IEEE 1588)

Available for ...
raL2048-48gm, raL4096-24gm, raL6144-16gm

The Precision Time Protocol (PTP) provides a method to synchronize multiple GigE cameras operated in the same network. It achieves clock accuracy in the sub-microsecond range.

The protocol is defined in the IEEE 1588 standard. The Basler racer GigE cameras support the revised version of the standard (IEEE 1588-2008, also known as PTP Version 2).

PTP enables a camera to use the following features:

- Action commands
  This feature lets you trigger actions in multiple cameras synchronously.
  For more information, see Section 10.12 on page 195.
- Scheduled action commands
  This feature lets you trigger actions in a camera or in multiple cameras at a specific time in the future.
  For more information, see Section 10.13 on page 202.
- Synchronous free run
  This feature makes it possible to let multiple cameras capture their images synchronously.
  For more information, see Section 10.14 on page 204.

Clock Synchronization in a Network via PTP

Measurement and automation systems involving multiple devices (e.g. cameras) often require accurate timing in order to facilitate event synchronization and data correlation.

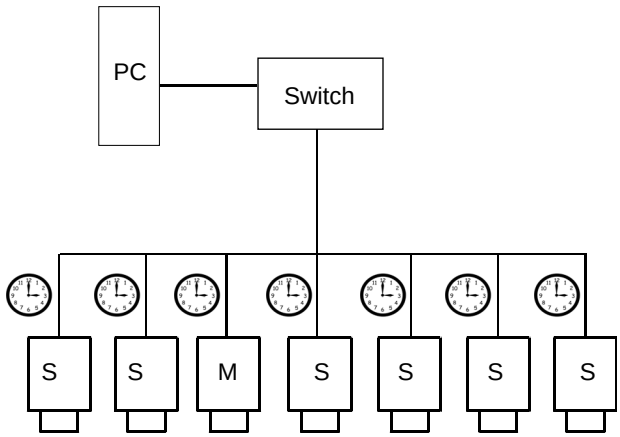
Through PTP, multiple devices (e.g. cameras) are automatically synchronized with the most accurate clock found in a network, the so-called master clock or best master clock.

The protocol enables systems within a network

- to **synchronize** a local clock with the master clock, i.e. to set the local clock as precisely as possible to the time of the master clock, and
- to **syntonize** a local clock with a master clock, i.e. to adjust the frequency of the local clock to the frequency of the master clock. The duration of a second is as identical as possible on both devices.

There are two different concepts of finding the master clock:

A. The synchronization between the different device clocks will determine a clock within one of the cameras to be the best master clock.
B. A clock outside of the cameras will be determined as the master clock; i.e. an external device (e.g. a GPS device) will be the best master clock.



Possibility A: One camera has the master clock (M); the other cameras have slave clocks (S).

Fig. 61: PTP-capable Cameras Using the Same System Time

How Does PTP Clock Synchronization Work?

The IEEE 1588 standard defines a Best Master Clock (BMC) algorithm in which each clock in a network identifies the most accurate clock and labels it "master clock". All other "slave clocks" synchronize and syntonize with this master.

The basic concept of IEEE 1588 is the exchange of timestamp messages. The protocol defines several periodic messages that trigger a clock to capture a timestamp and communicate timestamp information between the master and slave clocks. This method of using timestamps enables each slave clock in a network to analyze the master clock’s timestamp and the network propagation delay. This allows the slave clock to determine the delta from the master in its synchronization algorithm. For details about PTP messages, see the note box below.

IEEE 1588 defines 80-bit timestamps for storing and transporting time information. As GigE Vision uses 64-bit timestamps, the PTP timestamps are mapped to the 64-bit timestamps of GigE Vision.

An IEEE 1588 enabled device that operates in a network with no other enabled devices will not discipline its local clock. The drift and precision of the local clock is identical to a non-IEEE 1588 enabled device.

If no device in a network of IEEE 1588 enabled devices has a time traceable to the Universal Time Coordinated (UTC), the network will operate in the arbitrary timescale mode (ARB). In this mode, the epoch is arbitrary, as it is not bound to an absolute time. This timescale is relative, i.e. it is only valid in the network. The best master clock algorithm will select the clock which has the highest stability and precision as the master clock of the network.

Details about PTP Messages

In standard Ethernet frames, four IEEE 1588 messages are included (see Fig. 62):

Sync, Follow_up, Delay_Req, Delay_Resp

The Sync, Delay_Req, Follow_Up, and Delay_Resp messages are used to generate and communicate the timing information needed to synchronize the clocks using a delay request-response mechanism.

In order for a slave clock to synchronize with a master clock, the slave must know two pieces of information:

1. How much is the slave's clock off from the master clock?
   This is determined by the Sync and Follow_up message pair.
2. What is the network propagation delay?
   This is determined by the Delay_Req and Delay_Resp pair.

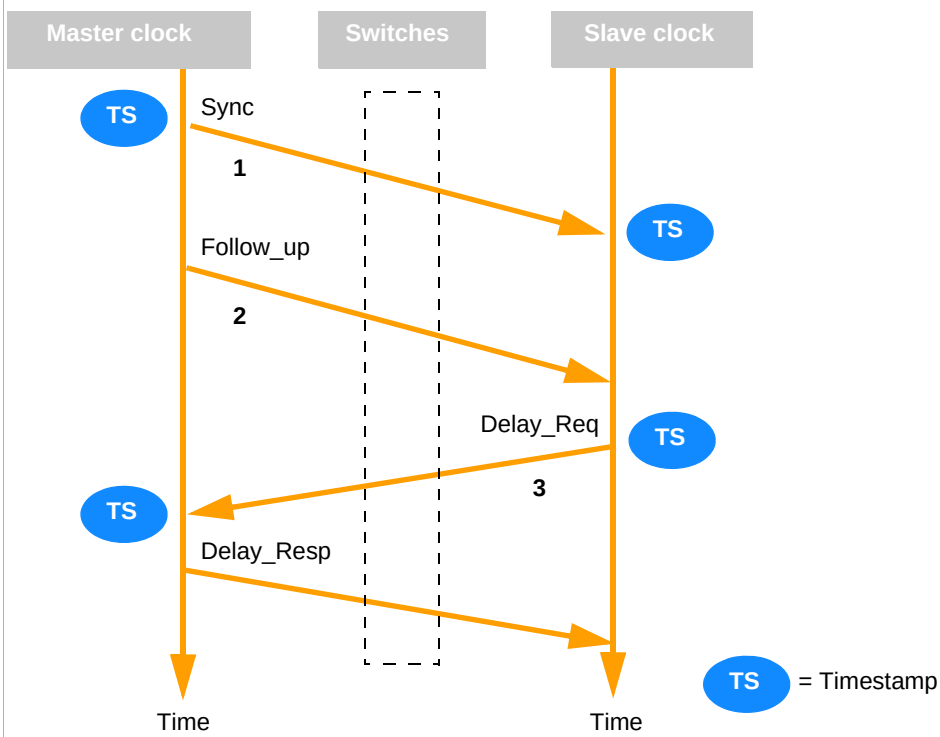


Fig. 62: PTP Clock Synchronization: Message Exchange Sequence

The "delay request" message is received and time stamped by the master clock, and the arrival timestamp is sent back to the slave clock in a "delay response" packet. The difference between these two timestamps is the network propagation delay.

By sending and receiving these synchronization packets, the slave clocks can accurately measure the offset between their local clock and the master clock. The slaves can then adjust their clocks by this offset to match the time of the master.

10.11.1 Enabling PTP Clock Synchronization

If you want to synchronize cameras using the Precision Time Protocol, you must enable PTP clock synchronization.

In the default factory setup, PTP clock synchronization is disabled.

To enable PTP clock synchronization:

1. If you want to use an external device as master clock (e.g. a GPS device or a software application on a computer synchronized by NTP - Network Time Protocol):
   Configure the external device as master clock. We recommend an ANNOUNCE interval of 2 seconds and a SYNC interval of 0.5 seconds.
2. Make sure that the following requirements are met:
   - All cameras you want to set up PTP for are installed and configured in the same network segment.
   - All cameras support PTP.

     You can check whether a camera supports PTP via the following command:

     ```
     if (GenApi::IsWritable(camera.GevIEEE1588))
     {
        // ...
     }
     ```
3. For **all cameras** that you want to synchronize, enable the PTP clock synchronization:

   ```
   camera.GevIEEE1588.SetValue(true);
   ```

What happens next depends on your setup:

- **An external device serves as the master clock** (e.g. a GPS device):
  As soon as PTP is enabled, the master clock starts sending and receiving synchronization packets so that the slave clocks can synchronize their time with the master clock. This will take a moment (approximately 30 seconds to 1 minute).
  From now on, the master and slave clocks are continuously synchronizing.
- **No external device serves as the master clock**
  One of the camera clocks will serve as the master clock.
  As soon as PTP is enabled for all camera clocks, the cameras start sending and receiving synchronization packets and determine the slave clocks and the best master clock. This will take a moment (approximately 30 seconds to one minute).
  The slaves can then adjust their clocks to match the time of the master.
  From now on, the master and slave clocks are continuously synchronizing.

If the PTP clock synchronization is enabled, and if the GigE Vision Timestamp Value bootstrap register is controlled by the IEEE 1588 protocol,

- the camera's GevTimestampTickFrequency parameter value is fixed to 1000000000 (1 GHz).
- the camera's GevTimestampControlReset feature is disabled.

10.11.2 Checking the Status of the PTP Clock Synchronization

After PTP clock synchronization has been enabled on all devices (see Section 10.11.1 on page 192), you can check the status of the synchronization.

Status Parameters

Four parameter values can be read from each device to determine the status of the PTP clock synchronization:

- GevIEEE1588OffsetFromMaster: A 32-bit number. Indicates the temporal offset between the master clock and the clock of the current IEEE 1588 device in nanoseconds.
- GevIEEE1588ClockId: A 64-bit number. Indicates the unique ID of the current IEEE 1588 device (the "clock ID").
- GevIEEE1588ParentClockId: A 64-bit number. Indicates the clock ID of the IEEE 1588 device that currently serves as the master clock (the "parent clock ID").
- GevIEEE1588StatusLatched: An enumeration. Indicates the state of the current IEEE 1588 device, e.g. whether it is a master or a slave clock. The returned values match the IEEE 1588 PTP port state enumeration (Initializing, Faulty, Disabled, Listening, Pre_Master, Master, Passive, Uncalibrated, and Slave). For more information, refer to the pylon API documentation and the IEEE 1588 specification.

The parameter values can be used to e.g.

- delay image acquisition until all cameras are properly synchronized, i.e. until the master and slave clocks have been determined and the temporal offset between the master clock and the slave clocks is low enough for your needs, or to
- optimize your network setup for high clock accuracy. For example, you can compare the temporal offsets of the IEEE 1588 devices while changing the network hardware, e.g. routers or switches.

Before the parameter values can be read, you must execute the GevIEEE1588DataSetLatch command to take a "snapshot" (also known as the "latched data set") of the camera's current PTP clock synchronization status. This ensures that all parameter values relate to exactly the same point in time.

The snapshot includes all four status parameter values: GevIEEE1588OffsetFromMaster, GevIEEE1588ClockId, GevIEEE1588ParentClockId, and GevIEEE1588StatusLatched. The values will not change until you execute the GevIEEE1588DataSetLatch command on this device again.

Instead of reading GevIEEE1588StatusLatched, you can read the equivalent GevIEEE1588Status parameter value from the device. This parameter value also provides the periodically updated IEEE 1588 device status, but does not require executing the GevIEEE1588DataSetLatch command beforehand.

Note, however, that if you read multiple IEEE 1588-related values from a device, the GevIEEE1588Status parameter value will not relate to the same point in time as the other values.

To check the status of the PTP clock synchronization:

1. Make sure that PTP clock synchronization has been enabled on all devices (see Section 10.11.1 on page 192).

For **all cameras** that you want to check the status for, perform the following steps:

2. Execute the GevIEEE1588DataSetLatch command to take a snapshot of the camera's current PTP clock synchronization status.
3. Read one or more of the following parameter values from the device:
   - GevIEEE1588OffsetFromMaster
   - GevIEEE1588ClockId
   - GevIEEE1588ParentClockId
   - GevIEEE1588StatusLatched

   All of these parameter values relate to exactly the same point in time, i.e. the point in time when the device received the GevIEEE1588Latch command.

Code Example

You can set the Precision Time Protocol parameters from within your application software by using the Basler pylon API.

The following code snippet illustrates using the API to take a snapshot of the synchronization status, read the clock ID of the current device and determine the temporal offset between the master clock and the clock of the current device.

```
camera.GevIEEE1588DataSetLatch.Execute();
int64_t clockId = camera.GevIEEE1588ClockId.GetValue();
int64_t offset = camera.GevIEEE1588OffsetFromMaster.GetValue();
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

10.12 Action Commands

Available for ...
raL2048-48gm, raL4096-24gm, raL6144-16gm

Action commands let you execute actions in multiple cameras at roughly the same time by using a single broadcast protocol message.

Each action protocol message contains an action device key, an action group key, and an action group mask. If the camera detects a match between this protocol information and one of the actions selected in the camera, the device executes the corresponding action.

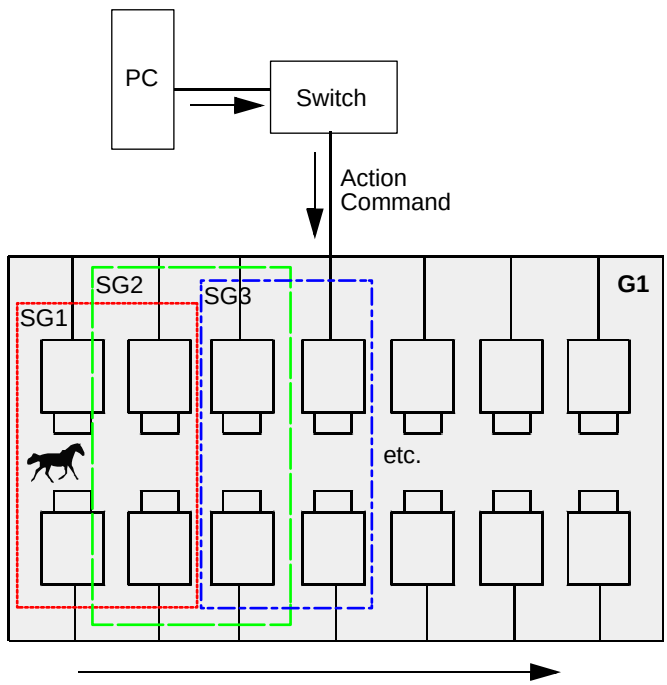
You can use action commands to synchronously

- capture images with multiple cameras (see Section 10.12.3.1 on page 198)
- reset the frame counter in multiple cameras (see Section 10.12.3.2 on page 200)

10.12.1 Action Command Example Setup

The following example setup will give you an idea of the basic concept of action commands.

To analyze the movement of a horse, multiple cameras are installed parallel to a race track. They form a group of cameras (G1, see Fig. 63).



Triggering sub-groups of cameras (SG1 - SGn) to capture images as the horse advances.

Fig. 63: Example Setup: Analyzing the Movements of a Horse

When the horse passes, four cameras positioned next to each other (sub-group SG1 in Fig. 63) synchronously execute an action (in this example: image acquisition).

As the horse advances, the next four cameras (sub-group SG2 in Fig. 63) synchronously capture images. One sub-group follows another in this fashion until the horse reaches the end of the race track. The resulting images can be combined and analyzed in a subsequent step.

In this sample use case, the following must be defined:

- A unique device key to authorize the execution of the synchronous image acquisition. The device key must be configured in each camera and it must be the same as the device key for the action command protocol message.
- The group of cameras in a network segment that is addressed by the action command. In Fig. 63, this group is G1.
- The sub-groups in the group of cameras that capture images synchronously. In Fig. 63, these sub-groups are SG1, SG2, SG3, and so on.

To define the device key, the group of cameras, and their sub-groups, the parameters action device key, action group key, and action group mask are used. For more information about these parameters, see Section 10.12.2.

10.12.2 Action Command Parameters

The main parameters associated with an action command are:

- **Action Device Key**

  An arbitrarily selectable 32-bit number used to authorize the execution of an action command in the camera. If the action device key in the camera and the action device key in the protocol message are identical, the camera will execute the corresponding action.
  The device key is write-only; it cannot be read out from the camera.

- **Action Group Key**

  An arbitrarily selectable 32-bit number used to define a group of devices on which an action should be executed. Each camera can be assigned to exactly one group.
  If the action group key in the camera and the action group key in the protocol message are identical, the camera will execute the corresponding action.

- **Action Group Mask**

  An arbitrarily selectable 32-bit number used for filtering out a sub-group of cameras belonging to a group of cameras. The cameras belonging to a sub-group execute an action at the same time.
  The filtering is done using a logical bitwise And operation against the group mask number of the action command and the group mask number of a camera. If both binary numbers have at least one common bit set to 1 (i.e. the result of the And operation is non-zero), the corresponding camera belongs to the sub-group.

Example Action Group Mask

A group of six cameras is installed on an assembly line.
For executing actions on specific sub-groups, the following group mask numbers have been assigned to the cameras (sample values):

Camera	Group Mask Number (Binary representation)	Group Mask Number (Hexadecimal representation)
1	000001	0x1
2	000010	0x2
3	000100	0x4
4	001000	0x8
5	010000	0x10
6	100000	0x20

To execute an action on cameras 1, 2, and 3 of these cameras, an action command with an action group mask of 000111 must be sent (hexadecimal representation: 0x7).
To execute an action on cameras 3, 4, and 6 of these cameras, an action command with an action group mask of 101100 must be sent (hexadecimal representation: 0x2C).

- **Number of Action Signals**

  An action signal is a device-internal signal that triggers an action (e.g. image acquisition). Each action command contains exactly one action signal. The number of action signals determines how many different action signals a device can handle (i.e. for how many different action commands a device can be configured).
  At the moment, the number of action signals is limited to 1 for all Basler cameras that support action commands. This means that if you previously set up a camera for an action command and you want to define a new action command, you have to replace the existing camera configuration.

- **Action Selector**

  A 32-bit number used to select the action command to configure. Because you cannot assign more than one action command to a Basler camera at a time, the Action Selector should always be set to 1 (see "Number of Action Signals").

- **Broadcast Address**

  A string variable used to define where the action command will be broadcast to. The broadcast address must be in dot notation, e.g. "255.255.255.255" (all adapters), "192.168.1.255" (all devices in a single subnet 192.168.1.xxx), or "192.168.1.38" (a single device). This parameter is optional. If omitted, "255.255.255.255" will be used.

These parameters can be accessed and modified by using the Basler pylon API or the Basler pylon Viewer application.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

For more information about the action command parameters, see

- the *Programmer's Guide and Reference Documentation* delivered with the pylon Camera Software Suite.
- the GigE Vision Specification, version 2.0, Section 14.3.

10.12.3 Using Action Commands

This chapter provides information about using action commands for different purposes.

10.12.3.1 Synchronous Image Acquisition

You can use action commands to synchronously capture images with multiple cameras (see example in Section 10.12.1 on page 195).

To use an action command to synchronously capture images:

1. Make sure that the following requirements are met:
   - All cameras you want to set up action commands for must be installed and configured in the same network segment.
   - The action commands feature is supported by the camera and the Basler pylon API you are using to configure and send action command(s).
   - If necessary, basic camera parameters are set (gain etc.).

For **all cameras** that you want to send an action command to, make the following settings:

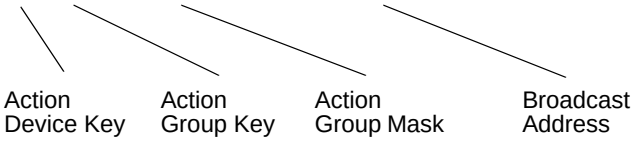
2. Open the camera connection.
3. Use the TriggerSelector parameter to select the trigger type.
   Available trigger types are FrameStart, AcquisitionStart, and LineStart.
4. Set the TriggerMode parameter to On.
5. Set the TriggerSource parameter to TriggerSource_Action1.
   At the moment, only this action command trigger source is available. This is because the number of action signals is limited to 1 (see "Number of Action Signals" in Section 10.12.2 on page 196).
6. Set the values of the following action command-specific parameters in the camera:

   ActionDeviceKey, ActionGroupKey, ActionGroupMask, and ActionSelector.

   The device key and group key values must match the corresponding values set in the protocol message (see "Action Device Key" and "Action Group Key" in Section 10.12.2 on page 196).
   The group mask value and the group mask value in the protocol message must have at least one common bit set to 1 (see "Action Group Mask" in Section 10.12.2 on page 196).
   The action selector value must always be 1 (see "Action Selector" in Section 10.12.2 on page 196).

7. Repeat steps 2 to 6 for all cameras.
8. To send the action command, call the IssueActionCommand method in your application. Example of an IssueActionCommand call (sample values):

```
IssueActionCommand(4711, 1, 0xFFFFFFFF, "255.255.255.255")
```



This will send an action command to all cameras with a device key of 4711 and a group key of 1, regardless of their group mask number or their network address.

Code Example

You can set the action command parameters from within your application software by using the Basler pylon API.

The following code snippet illustrates using the API to set up four cameras for synchronous image acquisition with a frame start trigger. For the ActionDeviceKey, the ActionGroupKey, and the ActionGroupMask parameters, sample values are used. It is assumed that the "Cameras" object is an instance of CBaslerGigEInstantCameraArray.

After the camera has been set up, an action command is sent to the cameras.

```
//--- Start of camera setup ---
for (size_t i = 0; i < Cameras.GetSize(); ++i)
   {
      Cameras[i].Open();
      //Set the trigger selector
      Cameras[i].TriggerSelector.SetValue(TriggerSelector_FrameStart);


      //Set the mode for the selected trigger
      Cameras[i].TriggerMode.SetValue(TriggerMode_On);


      //Set the source for the selected trigger
      Cameras[i].TriggerSource.SetValue(TriggerSource_Action1);


      //Set the action selector
      Cameras[i].ActionSelector.SetValue(1);


      //Set the action device key
      Cameras[i].ActionDeviceKey.SetValue(4711);


      //Set the action group key
      //In this sample, all cameras will be in the same group
      Cameras[i].ActionGroupKey.SetValue(1);


      //Set the action group mask
      //In this sample, all cameras will respond to any mask
```

```
        //other than 0
        Cameras[i].ActionGroupMask.SetValue(0xffffffff);
    }
//--- End of camera setup ---
//Send an action command to the cameras
GigeTL->IssueActionCommand(4711, 1, 0xffffffff, "255.255.255.255");
```

10.12.3.2 Synchronous Frame Counter Reset

You can use the Action Command feature to synchronously reset the frame counter in multiple cameras.

To use an action command to synchronously reset frame counters:

1. Make sure that the following requirements are met:
   - All cameras you want to set up action commands for must be installed and configured in the same network segment.
   - The action commands feature is supported by the camera and the Basler pylon API you are using to configure and send action command(s).

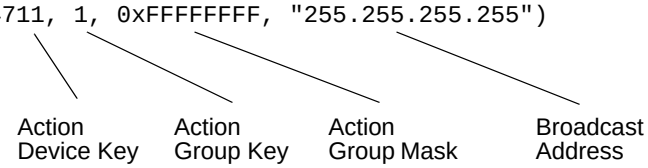
For **all cameras** that you want to send an action command to, make the following settings:

2. Open the camera connection.
3. Set the CounterResetSource parameter to CounterResetSource_Action1.
   At the moment, only this action command counter reset source is available. This is because the number of separate action signals is limited to 1 (see "Number of Action Signals" in Section 10.12.2 on page 196).
4. Set the values of the following action command-specific parameters in the camera:

   ActionDeviceKey, ActionGroupKey, ActionGroupMask, and ActionSelector.

   The device key and group key values must match the corresponding values set in the protocol message (see "Action Device Key" and "Action Group Key" in Section 10.12.2 on page 196). The group mask value and the group mask value in the protocol message must have at least one common bit set to 1 (see "Action Group Mask" in Section 10.12.2 on page 196).
   The action selector value must always be 1 (see "Action Selector" in Section 10.12.2 on page 196).
5. Repeat steps 2 to 4 for all cameras.
6. To send the action command, call the IssueActionCommand method in your application.
   Example of an IssueActionCommand call (sample values):

```
IssueActionCommand(4711, 1, 0xFFFFFFFF, "255.255.255.255")
```



   This will send an action command to all cameras with a device key of 4711 and a group key of 1, regardless of their group mask number or their network address.

Code Example

You can set the action command parameters from within your application software by using the Basler pylon API.

The following code snippet illustrates using the API to set up a specific camera to synchronously reset the frame counter. For the ActionDeviceKey, the ActionGroupKey, and the ActionGroupMask parameters, sample values are used. It is assumed that the object "Cameras" is an instance of CBaslerGigEInstantCameraArray.

After the camera has been set up, an action command is sent to the camera.

```
for (size_t i = 0; i < Cameras.GetSize(); ++i)
   {
      Cameras[i].Open();
      //Set the counter reset source
      Cameras[i].CounterResetSource.SetValue(CounterResetSource_Action1);

      //Set the action selector
      Cameras[i].ActionSelector.SetValue(1);

      //Set the action device key
      Cameras[i].ActionDeviceKey.SetValue(4711);

      //Set the action group key
      //In this sample, all cameras will be in the same group
      Cameras[i].ActionGroupKey.SetValue(1);

      //Set the action group mask
      //In this sample, all cameras will respond to any mask
      //other than 0
      Cameras[i].ActionGroupMask.SetValue(0xffffffff);
   }

//Send an action command to the cameras
GigeTL->IssueActionCommand(4711, 1, 0xffffffff, "255.255.255.255");
```

10.13 Scheduled Action Commands

Available for ...
raL2048-48gm, raL4096-24gm, raL6144-16gm

The scheduled action command feature lets you

- trigger actions in multiple devices (e.g. cameras) via a single broadcast message
- at exactly the same time and
- at a precise point of time.

If you want to use scheduled action commands, the cameras must support the Precision Time Protocol (IEEE 1588). For more information, see Section 10.11 on page 189.

10.13.1 Scheduled Action Command Parameters

The basic parameters of the scheduled action command feature are the same as for the action command feature. For information about the action command feature parameters, see Section 10.12.2 on page 196.

In addition to these parameters, the scheduled action command feature uses the following parameter:

- **Action Time**

  A 64-bit GigE Vision timestamp (in nanoseconds) used to define when the action should be executed.

  - If zero (0) is entered or if the action time is set to a point of time in the past, the action command will be executed at the next possible point of time.
  - If the action time is set to a point of time in the future, the action command will be executed at the given time.

  To check the current timestamp of the camera, execute the GevTimestampControlLatch command to take a "snapshot" of the camera's current time settings. After that, you can read the GevTimestampValue parameter to determine the timestamp value of the snapshot.

10.13.2 Using Scheduled Action Commands

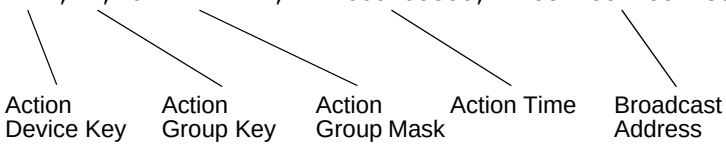
To set a scheduled action command:

1. Make sure that the following requirements are met before configuring the action command(s):
   - All cameras you want to set up action commands for must be installed and configured in the same network segment.
   - The action commands feature is supported by the camera and the Basler pylon API you are using to configure and send action command(s).

For **all cameras** that you want to send a scheduled action command to, make the following settings:

2. Open the camera connection.
3. If you want to use the scheduled action command
   - to synchronously capture images, set the TriggerSelector, TriggerMode, ActionDeviceKey, ActionGroupKey, ActionGroupMask, and ActionSelector parameters as described in Section 10.12.3.1 on page 198.
   - to synchronously reset the frame counter, set the CounterResetSource, ActionDeviceKey, ActionGroupKey, ActionGroupMask, and ActionSelector parameters as described in Section 10.12.3.2 on page 200.
4. Repeat steps 2 and 3 for all cameras.
5. To send the scheduled action command, call the IssueScheduledActionCommand method in your application.
   Example of an IssueScheduledActionCommand call (sample values):

```
IssueScheduledActionCommand(4711, 1, 0xFFFFFFFF, 171609799550, "255.255.255.255")
```



Code Example

Refer to the code examples in Section 10.12.3.1 on page 198 and Section 10.12.3.2 on page 200. These code examples can also be used to set up a scheduled action command. To do so, simply replace the IssueActionCommand call in the code examples by "IssueScheduledActionCommand" and add the Action Time parameter as described above.

10.14 Synchronous Free Run

Available for ...
raL2048-48gm, raL4096-24gm, raL6144-16gm

In a group of cameras that are not using the Precision Time Protocol (PTP), cameras that run in the free run mode may capture images at the same frame rate, but their image captures will be slightly asynchronous due to different reasons. See example A in Fig. 64.

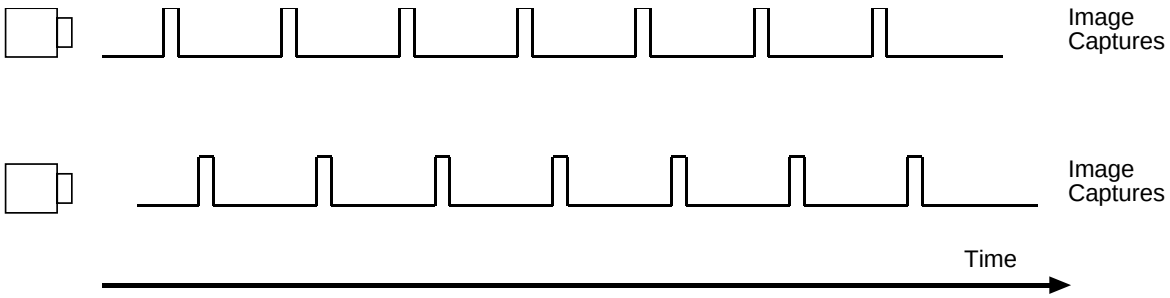


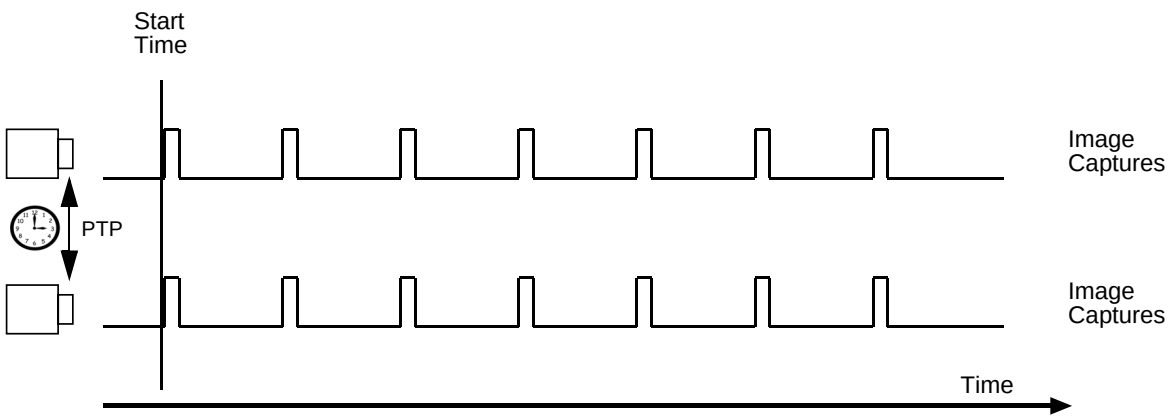
Fig. 64: **Example A: Without PTP**: Cameras Capturing at the Same Frame Rate, but Running Asynchronously

The Precision Time Protocol (PTP) in combination with the synchronous free run feature makes it possible to let multiple cameras in a network capture their images synchronously, i.e. at the **same time** and at the **same frame rate**. See example B in Fig. 65.

The frame rate is based on a tick frequency value that is the same for all cameras in the network.

It is also possible to start the image captures of multiple cameras at a precise start time.

For more information about the PTP feature, see Section 10.11 on page 189.



Example B With PTP: Same SyncFreeRunTriggerRateAbs Parameters

Fig. 65: Image Captures With PTP (Precision Time Protocol)

You can also use the synchronous free run feature in order to set a group of cameras as in example C (Fig. 66) and example D (Fig. 67):

- The cameras have exactly the **same exposure time** for their image capture but
- they capture their images in **precisely time-aligned intervals**, i.e. in a precise chronological sequence - for example: one camera starts capturing images immediately (start time = 0), the second camera 20 milliseconds after the start time, the third camera 30 milliseconds after the start time of the first camera and so on.

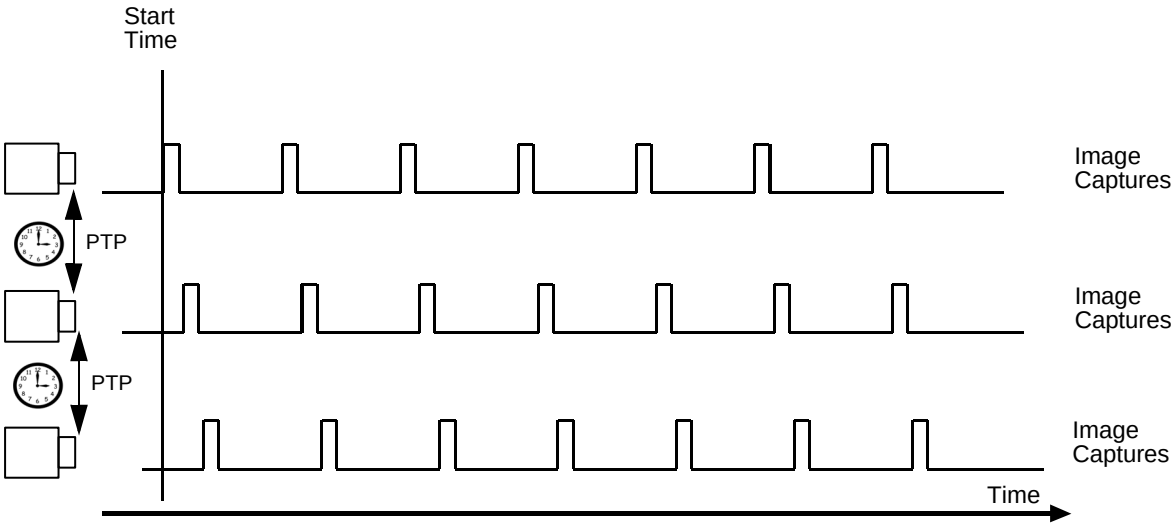


Fig. 66: **Example C**: Same SyncFreeRunTriggerRateAbs but in Chronological Sequence

The settings in **example D** (Fig. 67) are as follows:

- The cameras have the same start time (start time = 0)
- but they have **different exposure times** for their image capture

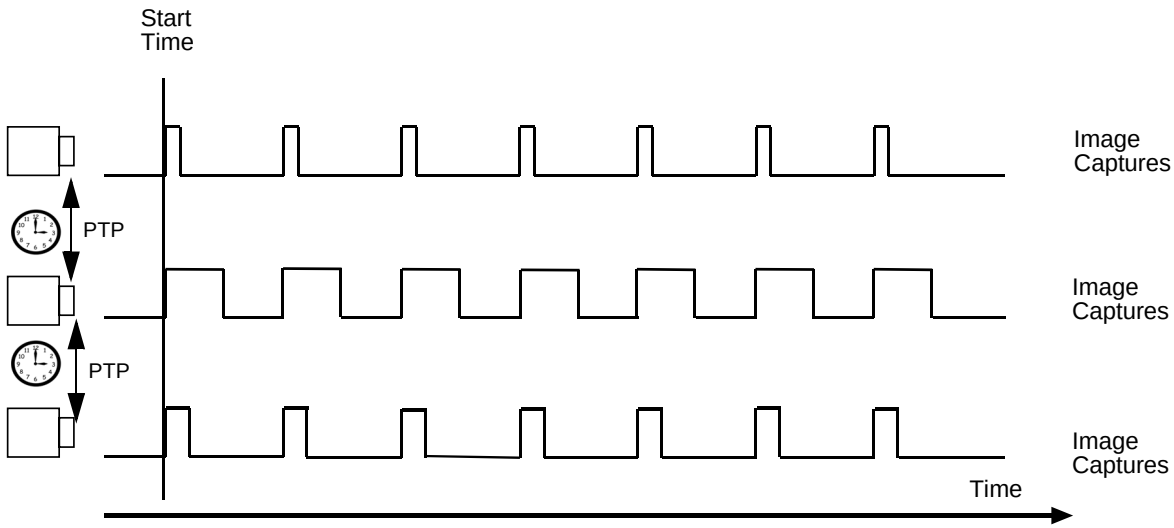


Fig. 67: **Example D**: Same Start Time and Same SyncFreeRunTriggerRateAbs but Different Exposure Times

10.14.1 Synchronous Free Run Parameters

The main parameters associated with the synchronous free run feature are:

- **SyncFreeRunTimerEnable**

  Enables or disables the synchronous free run feature.

- **SyncFreeRunTimerStartTimeLow and SyncFreeRunTimerStartTimeHigh**

  These two 32-bit values represent the lower and the higher part of a 64-bit GigE Vision timestamp (in nanoseconds).

  Combined, they form the start time for the synchronous free run feature.

  - If zero (0) is entered or if the start time is set to a point of time in the past, the free run starts at the next possible point of time.
  - If the start time is set to a point of time in the future, the free run starts at the given time.

  To check the current timestamp of the camera, execute the GevTimestampControlLatch command to take a "snapshot" of the camera's current time settings. After that, you can read the GevTimestampValue parameter to determine the timestamp value of the snapshot.

- **SyncFreeRunTimerTriggerRateAbs**

  Determines the rate at which the camera is triggering image acquisition using synchronous free run (in frames per second).

- **SyncFreeRunTimerUpdate**

  Each time you change one or more of the SyncFreeRunTimerStartTimeLow, SyncFreeRunTimerStartTimeHigh, or SyncFreeRunTimerTriggerRateAbs parameters, you must execute the SyncFreeRunTimerUpdate command to apply the changes. The update command ensures that all settings are applied at the same time.

10.14.2 Using Synchronous Free Run

To configure the synchronous free run for multiple Basler racer cameras:

1. Before configuring the synchronous free run of multiple cameras, make sure that the following requirements are met:
   - All cameras you want to trigger synchronously via the synchronous free run feature must be configured in the same network segment.
   - The Precision Time Protocol (PTP) is implemented and enabled for all cameras. All camera clocks run synchronously.

     For more information about enabling PTP, see Section 10.11.1 on page 192.

For **all cameras** that you want to run in the synchronized free run, make the following settings:

2. Make sure that the AcquisitionMode parameter is set to Continuous.
3. Set the TriggerMode parameter for the following trigger types to Off:
   - Acquisition start trigger
   - Frame start trigger
4. Set the parameters specific for the synchronous free run feature:
   a. Set the SyncFreeRunTimerStartTimeLow and SyncFreeRunTimerStartTimeHigh parameters to zero (0).
   b. Verify the maximum possible frame rate the camera can manage.
   c. Set the trigger rate for the synchronous free run (SyncFreeRunTimerTriggerRateAbs parameter) to the desired value. Example: If you want to acquire 10 frames per second, set the SyncFreeRunTimerTriggerRateAbs parameter value to 10.

      Make sure that you do not overtrigger the camera. If you overtrigger the camera, frame triggers may be ignored.
   d. Send the SyncFreeRunTimerUpdate command so that the complete start time (i.e. the low and high portion) and the frame rate are adopted by the camera.
   e. Set the SyncFreeRunTimerEnable parameter to True.
5. Set the parameters for all cameras you want to execute a synchronous free run for.

   As soon as the start time for the synchronous free run is reached, the camera starts acquiring images continuously.

Code Example

You can set the parameter values associated with synchronous free run feature from within your application software by using the Basler pylon API.

The following code snippets illustrate using the API to set the synchronous free run for a number of cameras so that they capture synchronously images, without a specific point of time in the future. The cameras will start as soon as the feature is enabled. It is assumed that the "Cameras" object is an instance of CBaslerGigEInstantCameraArray.

```
for (size_t i = 0; i < Cameras.GetSize(); ++i)
{
   Cameras[i].Open();

   // Enable PTP
   Cameras[i].GevIEEE1588.SetValue(true);

   // Make sure the frame trigger is set to Off to enable free run
   Cameras[i].TriggerSelector.SetValue(TriggerSelector_FrameStart);
   Cameras[i].TriggerMode.SetValue(TriggerMode_Off);

   // Let the free run start immediately without a specific start time
   camera.SyncFreeRunTimerStartTimeLow.SetValue(0);
   camera.SyncFreeRunTimerStartTimeHigh.SetValue(0);

   // Set the trigger rate to 30 frames per second
   camera.SyncFreeRunTimerTriggerRateAbs.SetValue(30.0);

   // Apply the changes
   camera.SyncFreeRunTimerUpdate.Execute();

   // Start the synchronous free run
   camera.SyncFreeRunTimerEnable.SetValue(true);
}
```

10.15 Error Codes

The camera can detect several user correctable errors. If one of these errors is present, the camera will set an error code and will flash both the yellow and green LEDs in the LED indicator:

Code	Condition	Meaning
0	No Error	The camera has not detected any errors since the last time that the error memory was cleared.
1	Overtrigger	An overtrigger has occurred. The user has applied an acquisition start trigger to the camera when the camera was not in a waiting for acquisition start condition. Or, the user has applied a frame start trigger to the camera when the camera was not in a waiting for frame start condition.
2	User Set Load	An error occurred when attempting to load a user set. Typically, this means that the user set contains an invalid value. Try loading a different user set.
3	Invalid Parameter	A parameter is set out of range or in an otherwise invalid manner.
4	Over Temperature	The camera has stopped image acquisition due to overheating. Provide adequate cooling to he camera.

Table 13: Error Codes

When the camera detects a user correctable error, it sets the appropriate error code in an error memory. If two or three different detectable errors have occurred, the camera will store the code for each type of error that it has detected (it will store one occurrence of the each code no matter how many times it has detected the corresponding error).

You can use the following procedure to check the error codes:

- Read the value of the LastError parameter. The LastError parameter will indicate the last error code stored in the memory.
- Execute the ClearLastErrorCommand to clear the last error code from the memory.
- Continue reading and clearing the last error until the parameter indicates a No Error code.

Reading and Clearing the Error Codes Using Basler Pylon

You can use the pylon API to read the value of the Last Error parameter and to execute a ClearLastError command from within your application software. The following code snippets illustrate using the API to read the parameter value and execute the command:

```
// Read the value of the last error code in the memory
LastErrorEnums lasterror = camera.LastError.GetValue();

// Clear the value of the last error code in the memory
camera.ClearLastError.Execute();
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

10.16 Test Images

All cameras include the ability to generate test images. Test images are used to check the camera's basic functionality and its ability to transmit an image to the host PC. Test images can be used for service purposes and for failure diagnostics.

When the camera is in test image mode, the optics, imaging sensor, and the ADCs are not used. The lines that make up each test image are generated internally by the camera's logic and the generated lines are collected in frame memory. When each test image frame is complete, it will be transmitted to the host PC in the same manner as with normal camera operation. The size of each test image frame will be determined by the frame parameter settings as with normal operation.

If the camera is set to use an electrical signal applied to input line 1, line 2, or line 3 as the source signal for the frame trigger and/or the line trigger, these signals must be provided to the camera in order to generate test images.

The Effect of Camera Settings on Test Images

When any test image is active, the camera's analog features such as analog gain, black level, and exposure time have no effect on the images transmitted by the camera. For test images 1, 2, and 3, the camera's digital features, will also have no effect on the transmitted images. But for test images 4 and 5, the cameras digital features will affect the images transmitted by the camera.

Enabling a Test Image

The TestImageSelector parameter is used to set the camera to output a test image. You can set the value of the TestImageSelector parameter to one of the test images or to "test image off".

You can set the TestImageSelector parameter from within your application software by using the pylon API. The following code snippets illustrate using the API to set the selector:

```
// set for no test image
camera.TestImageSelector.SetValue(TestImageSelector_Off);

// set for the first test image
camera.TestImageSelector.SetValue(TestImageSelector_Testimage1);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

Test Image 1 - Fixed Diagonal Gray Gradient (8 bit)

The 8 bit fixed diagonal gray gradient test image is best suited for use when the camera is set for monochrome 8 bit output. The test image consists of fixed diagonal gray gradients ranging from 0 to 255.

If the camera is set for 8 bit output, test image one will look similar to Fig. 68.

The mathematical expression for this test image is:

Gray Value = [column number + row number] MOD 256

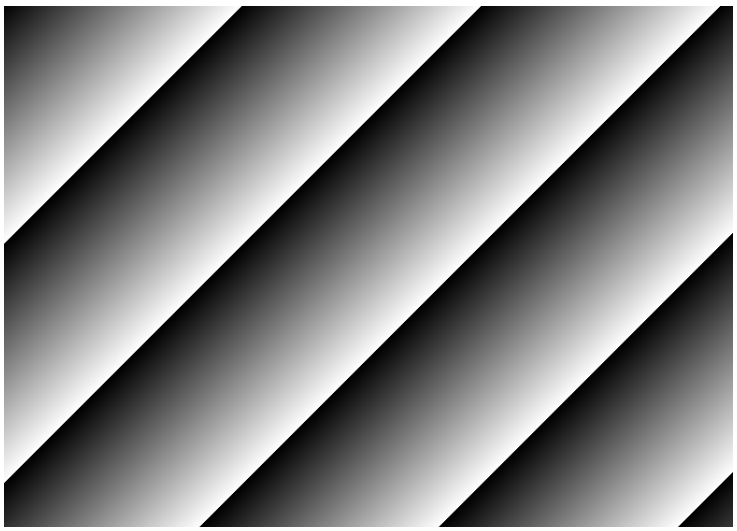


Fig. 68: Test Image One

Test Image 2 - Moving Diagonal Gray Gradient (8 bit)

The 8 bit moving diagonal gray gradient test image is similar to test image 1, but it is not stationary. The image moves by one pixel from right to left whenever a new frame acquisition is initiated. The test pattern uses a counter that increments by one for each new frame acquisition.

The mathematical expression for this test image is:

Gray Value = [column number + row number + counter] MOD 256

Test Image 3 - Moving Diagonal Gray Gradient (12 bit)

The 12 bit moving diagonal gray gradient test image is similar to test image 2, but it is a 12 bit pattern. The image moves by one pixel from right to left whenever a new frame acquisition is initiated. The test pattern uses a counter that increments by one for each new frame acquisition.

The mathematical expression for this test image is:

Gray Value = [column number + row number + counter] MOD 4096

Test Image 4 - Moving Diagonal Gray Gradient Feature Test (8 bit)

The basic appearance of test image 4 is similar to test image 2 (the 8 bit moving diagonal gray gradient image). The difference between test image 4 and test image 2 is this: if a camera feature that involves digital processing is enabled, test image 4 **will** show the effects of the feature while test image 2 **will not**. This makes test image 4 useful for checking the effects of digital features such as spatial correction.

Test Image 5 - Moving Diagonal Gray Gradient Feature Test (12 bit)

The basic appearance of test image 5 is similar to test image 3 (the 12 bit moving diagonal gray gradient image). The difference between test image 5 and test image 3 is this: if a camera feature that involves digital processing is enabled, test image 5 **will** show the effects of the feature while test image 3 **will not**. This makes test image 5 useful for checking the effects of digital features.

10.17 Device Information Parameters

Each camera includes a set of "device information" parameters. These parameters provide some basic information about the camera. The device information parameters include:

- DeviceVendorName (read only) - indicates the name of the camera’s vendor. This string will always indicate Basler as the vendor.
- DeviceModelName (read only) - indicates the model name of the camera, for example, raL2048-48gm.
- DeviceManufacturerInfo (read only) - can indicate some information about the camera manufacturer. This string usually indicates "none".
- DeviceVersion (read only) - indicates the device version number for the camera.
- FirmwareVersion (read only) - indicates the version of the firmware in the camera.
- DeviceID (read only) - indicates the serial number of the camera.
- DeviceUserID (read / write) - is used to assign a user defined name to a device. This name will be displayed in the Basler pylon Viewer and the Basler pylon IP Configuration Tool. The name will also be visible in the "friendly name" field of the device information objects returned by pylon’s device enumeration procedure.
- DeviceScanType (read only) - indicates the scan type of the camera, for example, line scan.
- SensorWidth (read only) - indicates the physical width of the sensor in pixels. The parameter value is identical to the camera’s maximum possible resolution.
- SensorHeight (read only) - indicates the physical height of the sensor in pixels.
- MaxWidth (read only) - indicates the camera’s maximum possible width setting for an image AOI.
- TemperatureAbs (read only) - Indicates the current temperature of the camera’s sensor board in degrees centigrade.

You can read the values for all of the device information parameters or set the value of the DeviceUserID parameter from within your application software by using the pylon API. The following code snippets illustrate using the API to read the parameters or write the DeviceUserID:

```
// Read the Vendor Name parameter
Pylon::String_t vendorName = camera.DeviceVendorName.GetValue();

// Read the Model Name parameter
Pylon::String_t modelName = camera.DeviceModelName.GetValue();

// Read the Manufacturer Info parameter
Pylon::String_t manufacturerInfo = camera.DeviceManufacturerInfo.GetValue();

// Read the Device Version parameter
Pylon::String_t deviceVersion = camera.DeviceVersion.GetValue();
```

```
// Read the Firmware Version parameter
Pylon::String_t firmwareVersion = camera.DeviceFirmwareVersion.GetValue();

// Read the Device ID parameter
Pylon::String_t deviceID = camera.DeviceID.GetValue();

// Write and read the Device User ID
camera.DeviceUserID = "custom name";
Pylon::String_t deviceUserID = camera.DeviceUserID.GetValue();

// Read the Sensor Width parameter
int64_t sensorWidth = camera.SensorWidth.GetValue();

// Read the Sensor Height parameter
int64_t sensorHeight = camera.SensorHeight.GetValue();

// Read the Max Width parameter
int64_t maxWidth = camera.WidthMax.GetValue();

// Read the Temperature Abs parameter
camera.TemperatureSelector.SetValue(TemperatureSelector_Sensorboard);
double temperature = camera.TemperatureAbs.GetValue();
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

For detailed information about using the pylon API and the pylon IP Configuration Tool, refer to the Basler pylon Programmer's Guide and API Reference.

10.18 User Defined Values

The camera can store two "user defined values". These two values are 32 bit signed integer values that you can set and read as desired. They simply serve as convenient storage locations for the camera user and have no impact on the operation of the camera.

The two values are designated as Value1 and Value2.

Setting User Defined Values

Setting a user defined value using Basler pylon is a two step process:

- Set the UserDefinedValueSelector parameter to Value1 or Value2.
- Set the UserDefinedValue parameter to the desired value for the selected value.

You can use the pylon API to set the UserDefinedValueSelector and the UserDefinedValue parameter values from within your application software. The following code snippet illustrates using the API to set the parameter values:

```
// Set user defined value 1
camera.UserDefinedValueSelector.SetValue(UserDefinedValueSelector_Value1);
camera.UserDefinedValue.SetValue(1000);

// Set user defined value 2
camera.UserDefinedValueSelector.SetValue(UserDefinedValueSelector_Value2);
camera.UserDefinedValue.SetValue(2000);

// Get the value of user defined value 1
camera.UserDefinedValueSelector.SetValue(UserDefinedValueSelector_Value1);
int64_t UserValue1 = camera.UserDefinedValue.GetValue();
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer’s Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

10.19 Configuration Sets

A configuration set is a group of values that contains all of the parameter settings needed to control the camera. There are three basic types of configuration sets: the active configuration set, the default configuration set, and user configuration sets.

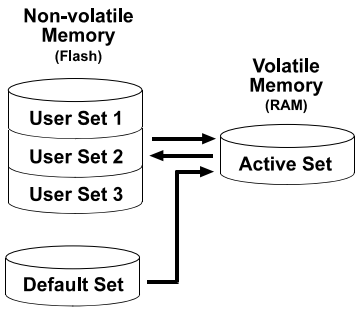


Fig. 69: Configuration Sets

Active Configuration Set

The active configuration set contains the camera's current parameter settings and thus determines the camera's performance, that is, what your image currently looks like. When you change parameter settings using the pylon API or the pylon Viewer, you are making changes to the active configuration set. The active configuration set is located in the camera's volatile memory and the settings are lost if the camera is reset or if power is switched off. The active configuration set is usually called the "active set" for short.

Default Configuration Set

When a camera is manufactured, a test setup is performed on the camera and an optimized configuration is determined. The default configuration set contains the camera's factory optimized configuration. The default configuration set is saved in a permanent file in the camera's non-volatile memory. It is not lost when the camera is reset or switched off and it cannot be changed. The default configuration set is usually just called the "default set" for short.

User Configuration Sets

As mentioned above, the active configuration set is stored in the camera's volatile memory and the settings are lost if the camera is reset or if power is switched off. The camera can save most of the settings from the current active set to a reserved area in the camera's non-volatile memory. A configuration set saved in the non-volatile memory is not lost when the camera is reset or switched off. There are three reserved areas in the camera's non-volatile memory available for saving configuration sets. A configuration set saved in a reserved area is commonly referred to as a "user configuration set" or "user set" for short.

The three available user sets are called User Set 1, User Set 2, and User Set 3.

	The settings for frame transmission delay, inter packet delay, and the luminance lookup table are not saved in the user sets and are lost when the camera is reset or switched off. If used, these settings must be set again after each camera reset or restart.

Default Startup Set

You can select the default configuration set or one of the user configuration sets stored in the camera's non-volatile memory to be the "default startup set." The configuration set that you designate as the default startup set will be loaded into the active set whenever the camera starts up at power on or after a reset. Instructions for selecting the default startup set appear on the next page.

10.19.1 Saving Configuration Sets

Saving the current active set into a user set in the camera's non-volatile memory is a three step process:

- Make changes to the camera's settings until the camera is operating in a manner that you would like to save.
- Set the UserSetSelector parameter to UserSet1, UserSet2, or UserSet3.
- Execute a UserSetSave command to save the active set to the selected user set.

Saving an active set to a user set in the camera's non-volatile memory will overwrite any parameters that were previously saved in that user set.

You can set the UserSetSelector parameter and execute the UserSetSave command from within your application software by using the pylon API. The following code snippet illustrates using the API to set the parameter and execute the command:

```
camera.UserSetSelector.SetValue(UserSetSelector_UserSet1);
camera.UserSetSave.Execute();
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

10.19.2 Loading a Saved Set or the Default Set into the Active Set

If you have saved a configuration set into the camera's non-volatile memory, you can load the saved set from the camera's non-volatile memory into the camera's active set. When you do this, the loaded set overwrites the parameters in the active set. Since the settings in the active set control the current operation of the camera, the settings from the loaded set will now be controlling the camera.

You can also load the default set into the camera's active set.

To load a saved configuration set or the default set into the active set:

1. Set the UserSetSelector parameter to UserSet1, UserSet2, UserSet3, or Default.
2. Execute a UserSetLoad command to load the selected set into the active set.

You can set the UserSetSelector parameter and execute the UserSetLoad command from within your application software by using the pylon API. The following code snippet illustrates using the API to set the parameter and execute the command:

```
camera.UserSetSelector.SetValue(UserSetSelector_UserSet2);
camera.UserSetLoad.Execute();
```

Loading a user set or the default set into the active set is only allowed when the camera is idle, i.e. when it is not acquiring lines.

Loading the default set into the active set is a good course of action if you have grossly misadjusted the settings in the camera and you are not sure how to recover. The default settings are optimized for use in typical situations and will provide good camera performance in most cases.

10.19.3 Selecting the Default Startup Set

You can select the default configuration set or one of the user configuration sets stored in the camera's non-volatile memory to be the "default startup set". The configuration set that you designate as the default startup set will be loaded into the active set whenever the camera starts up at power on or after a reset.

The UserSetDefaultSelector parameter is used to select the default startup set:

- Set the UserSetDefaultSelector parameter to UserSet1, UserSet2, UserSet3, or Default.

You can set the UserSetDefaultSelector parameter from within your application software by using the pylon API. The following code snippet illustrates using the API to set the parameter:

```
camera.UserSetDefaultSelector.SetValue(UserSetDefaultSelector_Default);
```

11 Chunk Features

11.1 What are Chunk Features?

In most cases, enabling a camera feature will simply change the behavior of the camera. The Test Image feature is a good example of this type of camera feature. When the Test Image feature is enabled, the camera outputs a test image rather than an acquired image. This type of feature is referred to as a "standard" feature.

When certain camera features are enabled, the camera actually develops some sort of information about each frame that it acquires. In these cases, the information is added to each frame as a trailing data "chunk" when the image is transferred to the host PC. Examples of this type of camera feature are the Frame Counter feature and the Time Stamp feature. When the Frame Counter feature is enabled, for example, after a frame is acquired, the camera checks a counter that tracks the number of frames acquired and develops a frame counter stamp for the frame. And if the Time Stamp feature is enabled, the camera creates a time stamp indicating when the frame was acquired. The frame counter stamp and the time stamp would be added as "chunks" of trailing data to each frame as the frame is transmitted from the camera. The features that add chunks to the acquired frames are referred to as “chunk” features.

Before you can use any of the features that add chunks to the frames, you must make the chunk mode active. Making the chunk mode active is described in the next section.

11.2 Making the "Chunk Mode" Active and Enabling the Extended Data Stamp

Before you can use any of the camera's "chunk" features, the "chunk mode" must be made active. Making the chunk mode active does two things:

- It automatically enables the Extended Frame Data chunk feature.
- It makes the camera's other chunk features available to be enabled.

To make the chunk mode active, set the ChunkModeActive parameter to true.

You can set the ChunkModeActive parameter value from within your application software by using the pylon API. The following code snippet illustrates using the API to set the parameter value:

```
camera.ChunkModeActive.SetValue(true);
```

Making the chunk mode inactive switches all chunk features off.

When you enable ChunkModeActive, the PayloadType for the camera changes from "Pylon::PayloadType_Image" to "Pylon::PayloadType_ChunkData".

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

When the chunk mode made is active, the Extended Frame Data feature will automatically be enabled, and the camera will add an "extended frame data" chunk to each acquired image. The extended frame data chunk appended to each acquired image contains some basic information about the frame. The information contained in the chunk includes:

- The OffsetX, Width, and Height settings for the frame
- The pixel format of the image data in the frame
- The minimum dynamic range and the maximum dynamic range

To retrieve data from the extended frame data chunk appended to a frame that has been received by your PC, you must first run the frame and its appended chunks through the chunk parser included in the pylon API. Once the chunk parser has been used, you can retrieve the extended frame data by doing the following:

- Read the value of the ChunkOffsetX parameter.
- Read the value of the ChunkWidth parameter.
- Read the value of the ChunkHeight parameter.
- Read the value of the ChunkPixelFormat parameter.
- Read the value of the ChunkDynamicRangeMin parameter.
- Read the value of the ChunkDynamicRangeMax parameter.

The following code snippet illustrates using the pylon API to run the parser and retrieve the extended image data:

```
// retrieve data from the extended frame data chunk
IChunkParser &ChunkParser = *camera.CreateChunkParser();
GrabResult Result;
StreamGrabber.RetrieveResult(Result);
ChunkParser.AttachBuffer((unsigned char*) Result.Buffer(),
Result.GetPayloadSize());
int64_t offsetX = camera.ChunkOffsetX.GetValue();
int64_t width = camera.ChunkWidth.GetValue();
int64_t height = camera.ChunkHeight.GetValue();
int64_t dynamicRangeMin = camera.ChunkDynamicRangeMin.GetValue();
int64_t dynamicRangeMax = camera.ChunkDynamicRangeMax.GetValue();
ChunkPixelFormatEnums pixelFormat = camera.ChunkPixelFormat.GetValue();
```

For more information about using the chunk parser, see the sample code that is included with the Basler pylon Software Development Kit (SDK).

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

11.3 Frame Counter

The Frame Counter feature numbers frames sequentially as they are acquired. When the feature is enabled, a chunk is added to each completed frame containing the value of the counter.

The frame counter is a 32 bit value. The counter starts at 0 and wraps back to 0 after it reaches its maximum. The counter increments by 1 for each acquired frame. Whenever the camera is powered off, the counter will reset to 0.

Be aware that if the camera is acquiring frames continuously and continuous acquisition is stopped, several numbers in the counting sequence may be skipped. This happens due to the internal buffering scheme used in the camera.

The chunk mode must be made active before you can enable the frame counter feature or any of the other chunk features. Making the chunk mode inactive disables all chunk features.

Enabling the Frame Counter and Retrieving Chunk Data

To enable the frame counter chunk:

1. Use the ChunkSelector parameter to select the frame counter chunk.
2. Set the ChunkEnable parameter to True.

Once the frame counter chunk is enabled, the camera will add a frame counter chunk to each acquired frame.

To retrieve data from a chunk appended a frame that has been received by your PC, you must first run the frame and its appended chunks through the chunk parser included in the pylon API. Once the chunk parser has been used, you can retrieve the frame counter information by reading the value of the ChunkFrameCounter parameter.

You can set the ChunkSelector and the ChunkEnable parameter values from within your application software by using the pylon API. You can also run the parser and retrieve the chunk data. The following code snippets illustrate using the API to activate the chunk mode, enable the frame counter chunk, run the parser, and retrieve the frame counter chunk data:

```
// make chunk mode active and enable frame counter chunk
camera.ChunkModeActive.SetValue(true);
camera.ChunkSelector.SetValue(ChunkSelector_Framecounter);
camera.ChunkEnable.SetValue(true);
// retrieve date from the chunk
IChunkParser &ChunkParser = *camera.CreateChunkParser();
GrabResult Result;
StreamGrabber.RetrieveResult(Result);
ChunkParser.AttachBuffer((unsigned char*) Result.Buffer(),
   Result.GetPayloadSize());
int64_t frameCounter = camera.ChunkFramecounter.GetValue();
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

Frame Counter Reset

Whenever the camera is powered off, the frame counter will reset to 0. During operation, you can reset the frame counter via I/O input 1, I/O input 2, I/O input 3 or software, and you can disable the reset. By default, the frame counter reset is disabled.

To use the frame counter reset:

1. Configure the frame counter reset by setting the CounterSelector parameter to Counter2 and setting the CounterEventSource parameter to FrameStart.
2. Set the CounterResetSource parameter to Line1, Line2, Line3, Software, or to Off.
3. If you set CounterResetSource to Software, execute the CounterReset command.
4. If you set CounterResetSource to Line1, Line2, or Line 3, apply a hardware trigger signal to the corresponding I/O input line.

You can set the frame counter reset parameter values from within your application software by using the pylon API. The following code snippets illustrate using the API to configure and set the frame counter reset and to execute a reset via software.

```
// configure reset of frame counter
camera.CounterSelector.SetValue(CounterSelector_Counter2);
camera.CounterEventSource.SetValue(CounterEventSource_FrameStart);

// select reset by signal on input line 1
camera.CounterResetSource.SetValue(CounterResetSource_Line1);
// select reset by signal on input line 2
camera.CounterResetSource.SetValue(CounterResetSource_Line2);
// select reset by signal on input line 3
camera.CounterResetSource.SetValue(CounterResetSource_Line3);

// select reset by software
camera.CounterResetSource.SetValue(CounterResetSource_Software);
// execute reset by software
camera.CounterReset.Execute();

// disable reset
camera.CounterResetSource.SetValue(CounterResetSource_Off);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

11.4 Time Stamp

The Time Stamp feature adds a chunk to each acquired frame. The chunk contains a time stamp that was generated when the frame start trigger for the frame became valid.

When the camera is set for continuous acquisition mode with the frame start trigger set to Off, the user is not required to apply frame start trigger signals to the camera. In this case, the camera will internally generate a signal that will be used for the stamp.

The time stamp is a 64 bit value. The time stamp is based on a counter that counts the number of "time stamp clock ticks" generated by the camera. The unit for each tick is 8 ns (as specified by the GevTimestampTickFrequency parameter). The counter starts at camera reset or at power off/on.

The chunk mode must be made active before you can enable the time stamp feature or any of the other chunk features. Making the chunk mode inactive disables all chunk features.

To enable the Time Stamp chunk and retrieve chunk data:

1. Use the ChunkSelector parameter to select the time stamp chunk.
2. Set the ChunkEnable parameter to True.

Once the time stamp chunk is enabled, the camera will add a time stamp chunk to each acquired frame.

To retrieve data from a chunk appended to a frame that has been received by your PC, you must first run the frame and its appended chunks through the chunk parser that is included in the pylon API.

Once the chunk parser has been used, you can retrieve the time stamp information by reading the value of the ChunkTimestamp parameter.

You can set the ChunkSelector and the ChunkEnable parameter values from within your application software by using the pylon API. You can also run the parser and retrieve the chunk data. The following code snippets illustrate using the API to activate the chunk mode, enable the time stamp chunk, run the parser, and retrieve the frame counter chunk data:

```
// make chunk mode active and enable Time Stamp chunk
camera.ChunkModeActive.SetValue(true);
camera.ChunkSelector.SetValue(ChunkSelector_Timestamp);
camera.ChunkEnable.SetValue(true);

// retrieve data from the chunk
IChunkParser &ChunkParser = *camera.CreateChunkParser();
GrabResult Result;
StreamGrabber.RetrieveResult(Result);
ChunkParser.AttachBuffer((unsigned char*) Result.Buffer(),
   Result.GetPayloadSize());
int64_t timeStamp = camera.ChunkTimestamp.GetValue();
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

11.5 Trigger Counters

The racer cameras have the following "trigger counters" available that can help you determine if you are triggering the camera correctly:

- Line Trigger Ignored Counter
- Frame Trigger Ignored Counter
- Line Trigger End To End Counter
- Frame Trigger Counter
- Frames Per Trigger Counter.
- Line Trigger Counter (only available on raL2048-48gm, raL4096-24gm, and raL6144-16gm camera models)

When a counter is enabled, a chunk is added to each completed frame containing the value of the counter. So if you have three counters enabled, for example, three chunks will be added to each frame.

The chunk mode must be made active before you can enable the time stamp feature or any of the other chunk features. Making the chunk mode inactive disables all chunk features.

The Line Trigger Ignored, Frame Trigger Ignored, Line Trigger End To End, and Line Trigger counters are each 32-bit counters. The Frame Trigger and Frames Per Trigger counters are each 16-bit counters.

Line Trigger Counter

Available for ...
raL2048-48gm, raL4096-24gm, raL6144-16gm

The Line Trigger Counter numbers external line acquisition triggers sequentially as they are received. When this counter is enabled, a chunk is added to each completed frame containing the value of the counter.

Be aware that the Line Trigger Counter counts **all** incoming line trigger signals, whether they are successful (acted on) or ignored (not acted on, e.g. due to overtriggering).

Example: You set the height of the frame to 100 lines, enable the line trigger counter, and send 120 line trigger signals at a rate that is higher than allowed. Due to overtriggering, 20 signals are ignored. Nevertheless, a line trigger counter value of 120 will be added to the first completed frame.

The line trigger counter is the only trigger counter that can be reset. For more information about resetting the line trigger counter, see Section 11.5.2 on page 230.

For more information about setting the height of a frame, see Section 8.1 on page 74.

Line Trigger Ignored Counter

The Line Trigger Ignored Counter counts the number of line triggers that were received during the acquisition of the current frame but were ignored (not acted on). A line trigger will be ignored if the camera is already in the process of acquiring a line when the trigger is received. Typically, this will happen if you overtrigger the camera, i.e., try to acquire lines at a rate that is higher than allowed. The magnitude of this counter will give you an idea of how badly the camera is being overtriggered. The higher the counter, the worse the overtriggering.

Frame Trigger Ignored Counter

The Frame Trigger Ignored Counter counts the number of frame triggers that were not acted upon during the acquisition of the frame because the camera was not ready to accept the trigger. Typically, this will happen if you attempt to trigger the start of a new frame while the camera is currently in the process of acquiring a frame.

Line Trigger End To End Counter

The Line Trigger End to End Counter counts the number of line triggers received by the camera from the end of the previous frame acquisition to the end of the current frame acquisition. If you subtract the number of lines actually included in the current frame from the number of lines shown by this counter, it will tell you the number of line triggers that were received but not acted on during the frame end to frame end period.

Frame Trigger Counter and Frames Per Trigger Counter

The Frame Trigger Counter and the Frames Per Trigger Counter are designed to be used together. They are available when the frame start trigger activation is set to either LevelHigh or LevelLow. The Frame Trigger Counter counts the number of frame trigger valid periods, and it increments each time the frame trigger becomes valid. The Frames Per Trigger counter counts the number of frames acquired during each frame valid period. The counter increments for each acquired frame (also for partial frames) and resets to zero for each new frame valid period. The way that the counters work is illustrated below.

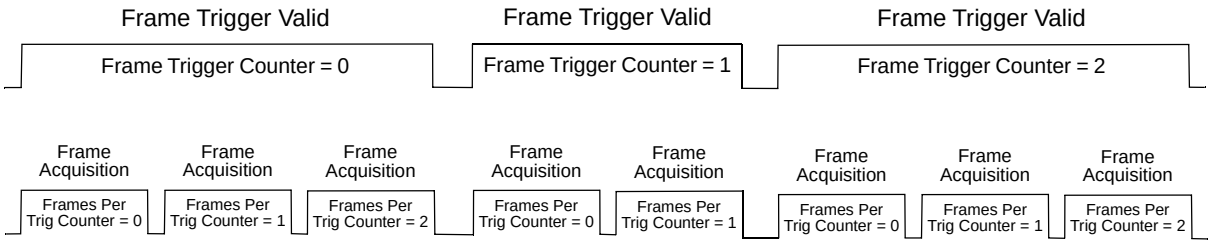


Fig. 70: Frame Trigger Counter and Frames Per Trigger Counter

These two counters can be used to determine which frames were acquired during a particular frame trigger valid period. This information will be especially useful in a situation where several frames must be stitched together to form an image of a single large object.

11.5.1 Enabling the Trigger Counters and Retrieving Chunk Data

To enable one of the trigger counter chunks:

1. Use the ChunkSelector parameter to select the trigger counter chunk.
2. Set the ChunkEnable parameter to True.

Once a trigger counter chunk has been enabled, the camera will add the counter chunk to each acquired frame.

You can set the ChunkSelector and ChunkEnable parameter values from within your application software by using the pylon API. The following code snippet illustrates using the API to activate the chunk mode and enable the trigger counter chunks:

```
// make chunk mode active
camera.ChunkModeActive.SetValue(true);

// enable the trigger counter chunks
camera.ChunkSelector.SetValue(ChunkSelector_LineTriggerCounter);
camera.ChunkEnable.SetValue(true);
camera.ChunkSelector.SetValue(ChunkSelector_LineTriggerIgnoredCounter);
camera.ChunkEnable.SetValue(true);
camera.ChunkSelector.SetValue(ChunkSelector_FrameTriggerIgnoredCounter);
camera.ChunkEnable.SetValue(true);
camera.ChunkSelector.SetValue(ChunkSelector_LineTriggerEndToEndCounter);
camera.ChunkEnable.SetValue(true);
camera.ChunkSelector.SetValue(ChunkSelector_FrameTriggerCounter);
camera.ChunkEnable.SetValue(true);
camera.ChunkSelector.SetValue(ChunkSelector_FramesPerTriggerCounter);
camera.ChunkEnable.SetValue(true);
```

To retrieve data from a chunk appended a frame that has been received by your PC, you must first run the frame and its appended chunks through the chunk parser included in the pylon API.

Once the chunk parser has been used, you can retrieve the counter values from the chunks by reading one or more of the following parameters:

- Chunk Line Trigger Counter
- Chunk Line Trigger Ignored Counter
- Chunk Frame Trigger Ignored Counter
- Chunk Line Trigger End To End Counter
- Chunk Frame Trigger Counter
- Chunk Frames Per Trigger Counter.

You can run the chunk parser and retrieve the counter values from within your application software by using the pylon API. The following code snippet illustrates using the API to run the parser and retrieve the frame counter chunk data:

```
// run the chunk parser
IChunkParser &ChunkParser = *camera.CreateChunkParser();
GrabResult Result;
StreamGrabber.RetrieveResult(Result);
ChunkParser.AttachBuffer((unsigned char*) Result.Buffer(),
   Result.GetPayloadSize());

// retrieve data from the chunks
int64_t LTCounter = camera.ChunkLineTriggerCounter.GetValue();
int64_t LTIgnoredCounter = camera.ChunkLineTriggerIgnoredCounter.GetValue();
int64_t FTIgnoredCounter = camera.ChunkFrameTriggerIgnoredCounter.GetValue();
int64_t LTEECounter = camera.ChunkLineTriggerEndToEndCounter.GetValue();
int64_t FTCounter = camera.ChunkFrameTriggerCounter.GetValue();
int64_t FPTCounter = camera.ChunkFramesPerTriggerCounter.GetValue();
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer’s Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

11.5.2 Resetting the Trigger Counters

Whenever the camera is powered off, all trigger counters will be reset to 0.

During operation, only the line trigger counter can be reset (available for raL2048-48gm, raL4096-24gm, and raL6144-16gm camera models only). Resetting the line trigger counter also works during frame acquisition. For example, if you set the height of the frame to 100 lines, acquire 60 lines, reset the counter, and then acquire the remaining 40 lines, the chunk added to the frame will contain a line trigger counter value of 40.

You can reset the line trigger counter via I/O input 1, I/O input 2, I/O input 3, or software, and you can disable the reset. By default, the line trigger counter reset is disabled.

To reset the line trigger counter:

1. Set the CounterSelector parameter to Counter3.
2. Set the CounterEventSource parameter to LineTrigger.
3. Set the CounterResetSource parameter to Line1, Line2, Line3, or Software.
4. If you set CounterResetSource to Software, execute the CounterReset command.
5. If you set CounterResetSource to Line1, Line2, or Line 3, apply a hardware trigger signal to the corresponding I/O input line.

You can set the line trigger counter reset parameter values from within your application software by using the pylon API. The following code snippets illustrate using the API to configure and set the line trigger counter reset and to execute a reset via software.

```
// configure the counter reset
camera.CounterSelector.SetValue(CounterSelector_Counter3);
camera.CounterEventSource.SetValue(CounterEventSource_LineTrigger);

// select reset by signal on input line 1
camera.CounterResetSource.SetValue(CounterResetSource_Line1);

// select reset by signal on input line 2
camera.CounterResetSource.SetValue(CounterResetSource_Line2);

// select reset by software
camera.CounterResetSource.SetValue(CounterResetSource_Software);

// execute reset by software
camera.CounterReset.Execute();

// disable reset
camera.CounterResetSource.SetValue(CounterResetSource_Off);
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference. You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

11.6 Encoder Counter

The encoder counter chunk indicates the value of the ShaftEncoderModuleCounter parameter at the time of the occurrence of a frame trigger. When the encoder counter chunk is enabled, a chunk is added to each frame containing the value of the ShaftEncoderModuleCounter parameter. The encoder counter chunk is a 16 bit value. The minimum value is 0 and the maximum is 32767.

The ShaftEncoderModuleCounter is part of the shaft encoder module. For more information about the shaft encoder module, about the ShaftEncoderModuleCounter, and about its possible modes of incrementing, see Section 8.6 on page 136.

The chunk mode must be made active before you can enable the time stamp feature or any of the other chunk features. Making the chunk mode inactive disables all chunk features.

To enable the encoder counter chunk:

1. Use the ChunkSelector parameter to select the encoder counter chunk.
2. Set the ChunkEnable parameter to True.

Once the encoder counter chunk is enabled, the camera will add an encoder counter chunk to each acquired frame.

To retrieve data from a chunk appended to an frame that has been received by your PC, you must first run the frame and its appended chunks through the chunk parser included in the pylon API. Once the chunk parser has been used, you can retrieve the encoder counter information by doing the following:

- Read the value of the ChunkEncoderCounter parameter.

You can set the ChunkSelector and ChunkEnable parameter values from within your application software by using the pylon API. You can also run the parser and retrieve the chunk data. The following code snippets illustrate using the API to activate the chunk mode, enable the encoder counter chunk, run the parser, and retrieve the encoder counter chunk data:

```
// make chunk mode active and enable Encoder Counter chunk
camera.ChunkModeActive.SetValue(true);
camera.ChunkSelector.SetValue(ChunkSelector_ChunkShaftEncoderCounter);
camera.ChunkEnable.SetValue(true);

// retrieve date from the chunk
IChunkParser &ChunkParser = *camera.CreateChunkParser();
GrabResult Result;
StreamGrabber.RetrieveResult(Result);
ChunkParser.AttachBuffer((unsigned char*) Result.Buffer(),
   Result.GetPayloadSize());
int64_t EncoderCounter = camera.ChunkShaftEncoderCounter.GetValue();
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

11.7 Input Line Status At Line Trigger

The Input Status At Line Trigger feature samples the status of all of the camera's input lines each time a line acquisition is triggered. It collects the input line status data for each acquired line in a chunk and adds the chunk to the frame that includes the acquired line.

The input status at line trigger information is a 4 bit value. As shown in Fig. 71, certain bits in the value are associated with each line and the bits will indicate the state of the lines. If a bit is 0, it indicates that the state of the associated line was low at the time of triggering. If a bit is 1, it indicates that the state of the associated line was high at the time of triggering.

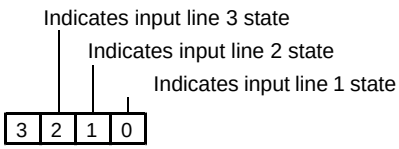


Fig. 71: Input Status At Line Trigger Parameter Bits

The chunk mode must be active before you can enable the input status at line trigger chunk or any of the other chunk feature. Making the chunk mode inactive disables all chunk features.

The maximum for the Height parameter value is 1024 if the input status at line trigger chunk is enabled. Other conditions may further decrease the maximum parameter value. For more information, see Section 8.1 on page 74.

To enable the input status at line trigger chunk:

1. Use the ChunkSelector parameter to select the input status at line trigger chunk.
2. Set the ChunkEnable parameter to True.

Once the input status at line trigger chunk is enabled, the camera will add an input status at line trigger chunk to each acquired frame.

To retrieve data from a chunk appended to an image that has been received by your PC, you must first run the image and its appended chunks through the chunk parser included in the pylon API. Once the chunk parser has been used, you can retrieve the input line status at line trigger information that was extant when acquisition of line i was triggered by doing the following:

- Read the value of the ChunkInputStatusAtLineTrigger parameter.

You can set the ChunkSelector and ChunkEnable parameter values from within your application software by using the pylon API. You can also run the parser and retrieve the chunk data. The following code snippets illustrate using the API to activate the chunk mode, enable the input status at line trigger chunk, run the parser, and retrieve the input status at line trigger chunk data for the acquired line i:

```
camera.ChunkModeActive.SetValue(true);
camera.ChunkSelector.SetValue(ChunkSelector_InputStatusAtLineTrigger);
camera.ChunkEnable.SetValue(true);

// grab image and feed it to the chunk parser ...

int MaxIdx = int(camera.ChunkInputStatusAtLineTriggerIndex.GetMax());
for (int i = 0; i <= MaxIdx; i++)
{
    camera.ChunkInputStatusAtLineTriggerIndex.SetValue(i);
    int value = int(camera.ChunkInputStatusAtLineTriggerValue.GetValue());
    printf("State of inputs at line %d: %X\n", i, value);
}
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer’s Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

11.8 CRC Checksum

The CRC (Cyclic Redundancy Check) Checksum feature adds a chunk to each acquired frame containing a CRC checksum calculated using the X-modem method. As shown in Fig. 72 on page 234, the checksum is calculated using all of the image data in the frame and all of the appended chunks except for the checksum itself. The CRC chunk is always the last chunk appended to the frame.

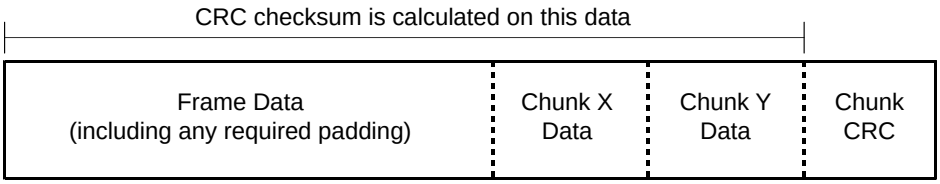


Fig. 72: CRC Checksum

The chunk mode must be made active before you can enable the time stamp feature or any of the other chunk features. Making the chunk mode inactive disables all chunk features.

To enable the CRC checksum chunk:

1. Use the ChunkSelector parameter to select the CRC chunk.
2. Set the ChunkEnable parameter to True.

Once the CRC chunk is enabled, the camera will add a CRC chunk to each acquired frame.

To retrieve CRC information from a chunk appended to a frame that has been received by your PC, you must first run the frame and its appended chunks through the chunk parser included in the pylon API. Once the chunk parser has been used, you can retrieve the CRC information. The CRC information provided by the chunk parser is not the CRC checksum itself. Rather it is a true/false result. When the frame and the appended chunks pass through the parser, the parser calculates a CRC checksum based on the received frame and chunk information. It then compares the calculated CRC checksum with the CRC checksum contained in the CRC checksum chunk. If the two match, the result will indicate that the frame data is OK. If the two do not match, the result will indicate that the frame is corrupted.

You can set the ChunkSelector and ChunkEnable parameter values from within your application software by using the pylon API. You can also run the parser and retrieve the chunk data. The following code snippets illustrate using the API to activate the chunk mode, enable the CRC checksum chunk, run the parser, and retrieve the frame counter chunk data:

```
// Make chunk mode active and enable CRC chunk
camera.ChunkModeActive.SetValue(true);
camera.ChunkSelector.SetValue(ChunkSelector_PayloadCRC16);
camera.ChunkEnable.SetValue(true);

// Check the CRC checksum of an acquired frame
IChunkParser &ChunkParser =
   *camera.CreateChunkParser();
GrabResult Result;
StreamGrabber.RetrieveResult(Result);
ChunkParser.AttachBuffer((unsigned char*) Result.Buffer(),
   Result.GetPayloadSize());
if (ChunkParser.HasCRC() && ! ChunkParser.CheckCRC())
   cerr << "Image corrupted!" << endl;
```

For detailed information about using the pylon API, refer to the Basler pylon Programmer's Guide and API Reference.

You can also use the Basler pylon Viewer application to easily set the parameters.

For more information about the pylon API and the pylon Viewer, see Section 3.1 on page 26.

12 Troubleshooting and Support

This chapter explains camera reset and outlines the resources available to you if you need help working with your camera.

12.1 Camera Reset

Some situations may require a camera reset. It is executed using the DeviceReset command parameter via the pylon API and is therefore also referred to as a "software reset".

During camera reset, camera power stays on. This is in contrast to camera restart where camera power is switched off and on again.

From the perspective of the Basler pylon software and the operating system, camera reset and camera restart both appear as a "surprise device removal" and must be handled accordingly.

When camera reset is carried out, all settings stored in the camera’s volatile memory are lost.

If you want to preserve settings stored in the camera’s volatile memory, safe them as a user set before carrying out camera reset. Some settings can not be saved in a user set, for example the settings for the luminance lookup table.

Resetting the camera

After having issued the camera reset command perform the subsequently necessary steps, e.g. some cleanup on the PC, in accord with the DeviceRemovalHandling sample code that is included in the pylon SDK documentation.

After camera reset was carried out, allow some time to elapse until the camera is detected again.

For more information about the pylon API and SDK documentation, see Section 3.1 on page 26.

12.2 Tech Support Resources

If you need advice about your camera or if you need assistance troubleshooting a problem with your camera, you can contact the Basler technical support team for your area. Basler technical support contact information is located in the front pages of this manual.

You will also find helpful information such as frequently asked questions, downloads, and application notes in the Downloads and the Support sections of our website: www.baslerweb.com

If you do decide to contact Basler technical support, please take a look at the form that appears on the last two pages of this section before you call. Filling out this form will help make sure that you have all of the information the Basler technical support team needs to help you with your problem.

12.3 Obtaining an RMA Number

Whenever you want to return material to Basler, you must request a Return Material Authorization (RMA) number before sending it back. The RMA number **must** be stated in your delivery documents when you ship your material to us! Please be aware that if you return material without an RMA number, we reserve the right to reject the material.

You can find detailed information about how to obtain an RMA number in the Support section of our website: www.baslerweb.com

12.4 Before Contacting Basler Technical Support

To help you as quickly and efficiently as possible when you have a problem with a Basler camera, it is important that you collect several pieces of information before you contact Basler technical support.

Copy the form that appears on the next two pages, fill it out, and fax the pages to your local dealer or to your nearest Basler support center. Or, you can send an e-mail listing the requested pieces of information and with the requested files attached. Basler technical support contact information is shown in the title section of this manual.

1 The camera's product ID: _____

2 The camera's serial number: _____

3 Network adapter that you use with the camera: _____

_____

4 Describe the problem in as much detail as possible: _____

_____

(If you need more space, use an extra sheet of paper.) _____

5 If known, what's the cause of the problem? _____

_____

_____

6 When did the problem occur?

☐ After start. ☐ While running.

☐ After a certain action (e.g., a change of parameters):

_____

_____

_____

_____

7 How often did/does the problem occur?

☐ Once. ☐ Every time.

☐ Regularly when:

_____

☐ Occasionally when:

_____

_____

8 How severe is the problem?

- ☐ Camera can still be used.
- ☐ Camera can be used after I take this action:

  _____

  _____

- ☐ Camera can no longer be used.

9 Did your application ever run without problems?

- ☐ Yes
- ☐ No

10 Parameter set

It is very important for Basler technical support to get a copy of the exact camera parameters that you were using when the problem occurred.

To make note of the parameters, use Basler's pylon Viewer.

If you cannot access the camera, please try to state the following parameter settings:

- ☐ Frame Size: _____
- ☐ Pixel Format: _____
- ☐ Packet Size: _____
- ☐ Exposure Time: _____
- ☐ Line rate: _____

11 Live image/test image

If you are having an image problem, try to generate and save live images that show the problem. Also generate and save test images. Please save the images in BMP format, zip them, and send them to Basler technical support.

Revision History

Doc. ID Number	Date	Changes
AW001183**01**000	20 Jun 2012	Preliminary release of this document. Applies to prototype cameras only.
AW001183**02**000	09 Apr 2013	Initial release for series cameras.
AW001183**03**000	13 Sep 2013	Updated the contact information for Asia. Updated some max. line rates in Section 1.2 on page 2. Added bit depth to Fig. 1 in Section 1.3 on page 8. Added a dimension value to Fig. 2 in Section 1.4.1 on page 9. Modified dimension values in Fig. 4 on page 12 and Fig. 5 on page 13. Updated the LZ4 licensing text in Section 1.5.2 on page 17. Updated the title of document AW000611 in Section 2 on page 24 through Section 4 on page 27. Rearranged Section 3.1.3 on page 26. Corrected the absolute maximum value for the Height parameter in Section 8.1 on page 73. Modified the Line Acquisition While Obeying Timing Limits section in Section 8.2.7 on page 111.
AW001183**04**000	04 Aug 2015	Minor corrections and modifications throughout the manual. Updated the contact information on the back of the front page. Corrected subpart J to subpart B in the FCC section at the back of the front page. Added references to the CE Conformity Declaration in Section 1.2 on page 2. Updated the dimensions of the F-mount lens adapter in Section 1.2 on page 2 and Section 1.4.3 on page 12. Added precautions related to SELV and LPS in Section 1.8 on page 21. Updated Chapter 3 on page 25 to reflect software update to Basler pylon Camera Software Suite. Corrected camera power requirements in Section 7.2.1 on page 51. Updated the power cable drawing in Section 7.4.1 on page 54. Added information about "low level" code and the Instant Camera classes in Chapter 8 on page 73. Added information about the AcquisitionLineRateAbs parameter in Section 8.2.4 on page 85. Added minimum and maximum values for the Frame Timeout Abs parameter in Section 8.2.3.4 on page 84. Renamed GainSelector_All to GainSelector_DigitalAll in Section 10.1.1.2 on page 157. Added the Auto Functions feature in Section 10.4 on page 163. Added the Precision Time Protocol feature in Section 10.11 on page 188. Added the Action Command feature in Section 10.12 on page 194.

Doc. ID Number	Date	Changes
		Added the Scheduled Action Command feature in Section 10.13 on page 201. Added the Synchronous Free Run feature in Section 10.14 on page 203. Re-organized information in Section 11.5 on page 225. Added the line trigger counter in Section 11.5 on page 225. Added information about resetting the trigger counters in Section 11.5.2 on page 229.
AW001183**05**000	01 Oct 2015	Added environmental requirements according to UL 60950-1 in Section 1.7.1 on page 19. Changed maximum housing temperature during operation from 50 °C to 60 °C in Section 1.7.1 on page 19 and Section 1.7.2 on page 19. Restructured and corrected Table 5 on page 51 ("Pin Assignments for the 6-pin Connector").
AW001183**06**000	08 Sep 2016	Added the M42-mount FBD 45.46 and M58-mount adapters to the specification tables in Section 1.2 on page 2, to the lens adapter drawings in Section 1.5.3 on page 13, and to the table in Section 1.5.4 on page 16. Added Section 1.3 on page 8 ("Accessories"). Added the ExposureOverlapTimeMaxAbs and the TargetGrayValue parameters to the list of parameters whose limits can be removed in Section 10.2 on page 161.

Index

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W

Y