Alkeria
IMAGINACTION

NECTA series

USB3 line-scan camera

USER GUIDE

Save the planet: do not print this manual unless you really need to.

Release 2.4 - March, 2022

WELCOME TO NECTA LINE-SCAN CAMERAS

Before starting using your camera, make sure you follow these basic rules and steps for a safe and effective use of NECTA .

- Make sure you are using a correct hardware (Section 1.1);
- Check safety and precautions chapter (Section 2.8);
- Access your User Area on www.alkeria.com and download camera manual and install MaestroUSB3 SDK (Section 1.2);
- Install your camera in your setup (Section 1.3).

Warning

Please do not connect I/O interface yet. Use the I/O connectors only after completing the first start of the camera and after reading Chapter 4 of this manual.

- Connect and start the camera for the first time (Section 1.4);
- Perform a camera calibration (Section 8.2).

Tip

Check the videos in your User Area about camera calibration.

IMPORTANT NOTES

- Make sure you check the information about NECTA Player and its functions in Chapter 9;
- Deeply check our manual and code samples: most of common operations and options available are described there;
- Remember that you can always reach our support team through your User Area on www.alkeria.com or emailing support@alkeria.com

Warning

NEVER try to power the camera through the I/O interface. Our cameras are self powered through USB3.

Summary

1 Getting started

1.1 REQUIREMENTS

For getting started with NECTA you will need the material listed below:

- A NECTA camera equipped with a lens adapter (Figure 1.1);
- A lens compatible with the chosen lens adapter;
- An USB 3.2 Gen 1x1 cable (Figure 1.2);
- A computer with an USB 3.2 Gen 1x1 controller;

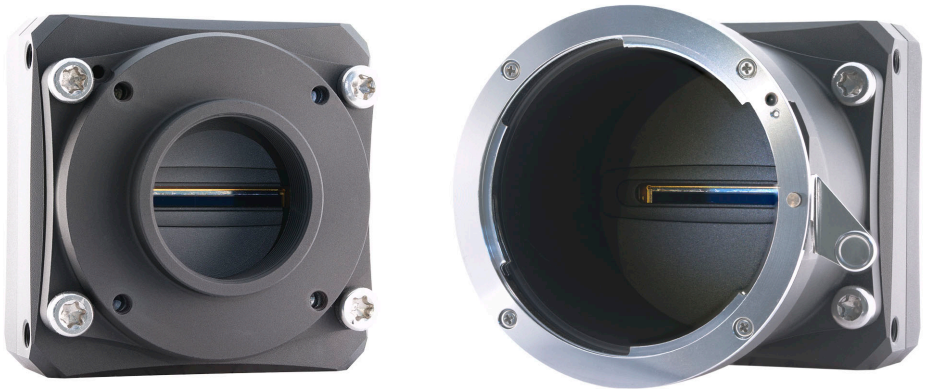


Figure 1.1: *NECTA camera*

(a) *(b)*

Figure 1.2: *Required USB cable is type-A on the PC side (a) and micro-B on the camera side (b).*

1.2 SOFTWARE DOWNLOAD

After camera purchasing, our Sales Department will provide you with access to our website's User Area. In there you can find everything you need to set and use the camera: manuals, guides, tutorials, software and firmware updates. Users Area accounts are personal, and can only be created by Alkeria's Sales Dept.

Once your account has been activated, you will receive an email to reset your account password: click on the link to set your password. Please store your login credentials in a secure place.

To download the software, follow these steps:

Step 1: Go to our website www.alkeria.com (Figure 1.3)

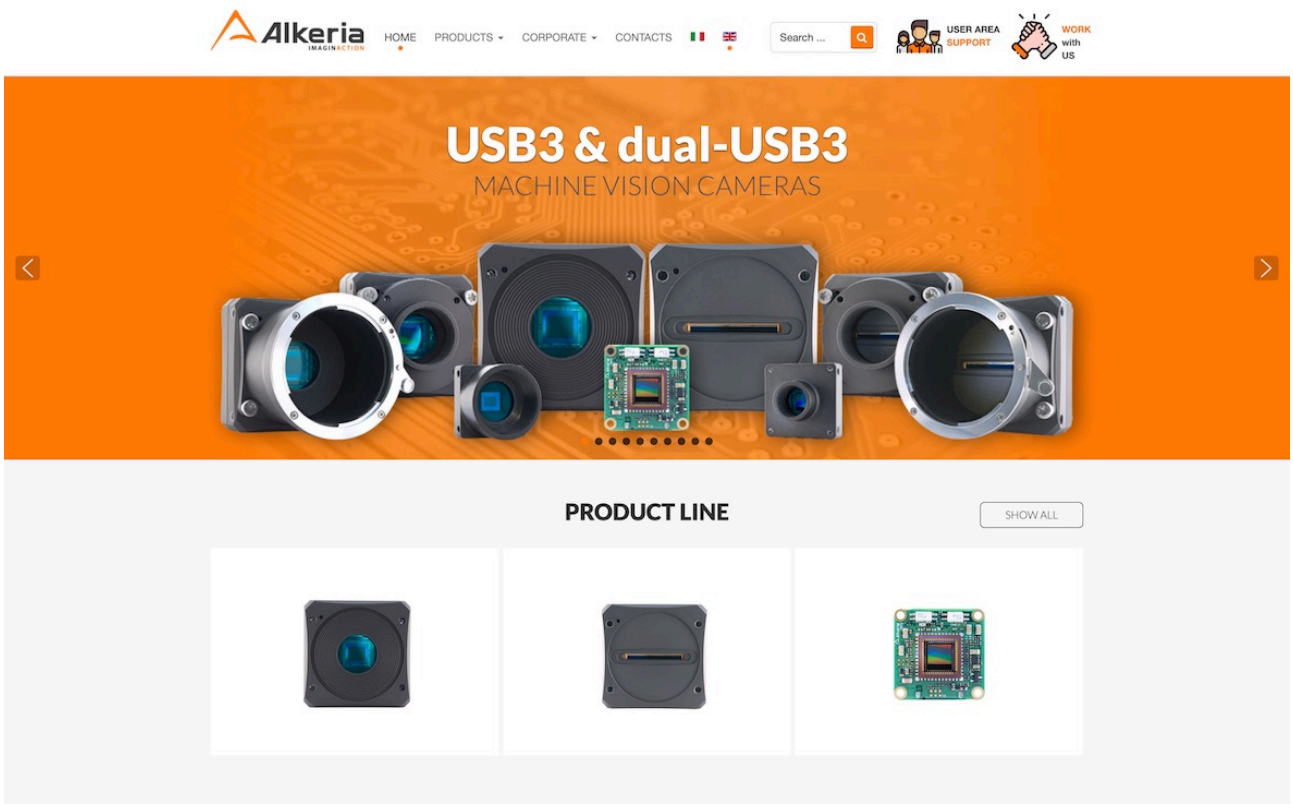


Figure 1.3: *Alkeria website*

Step 2: Click on "User Area - Support" top-right button (Figure 1.4)

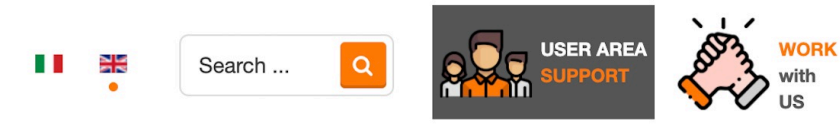


Figure 1.4: *User Area button*

Step 3: Once redirected to login form, insert your username and password and click "Log in" (Figure 1.5)

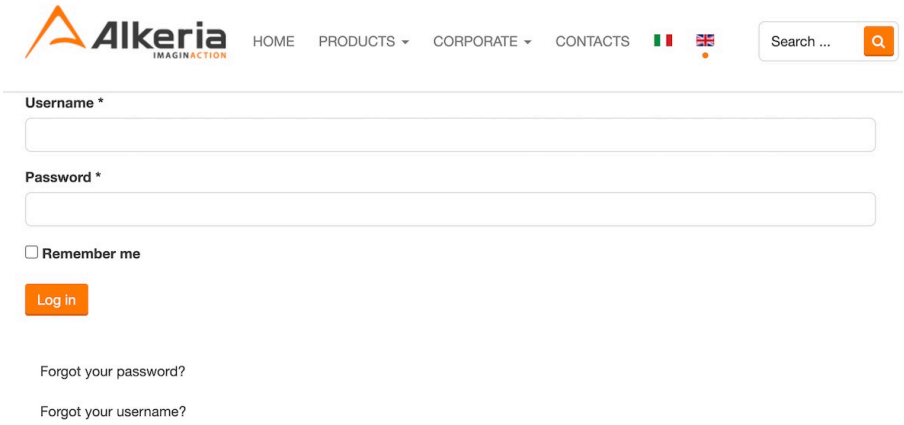


Figure 1.5: *User Area log in form*

Step 4: On "Downloads" root folder, you can find many file categories, each containing various files (Figure 1.6)

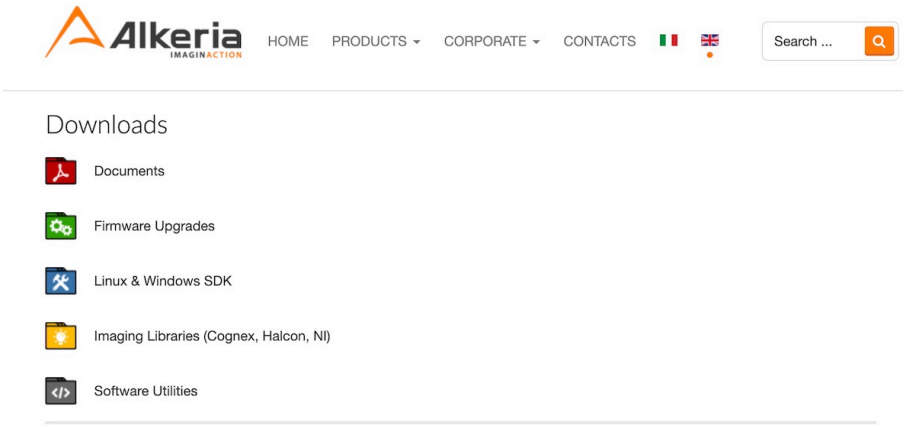


Figure 1.6: *User Area root folder*

Step 5: Click on "Download" to download a file (fig. 1.7)

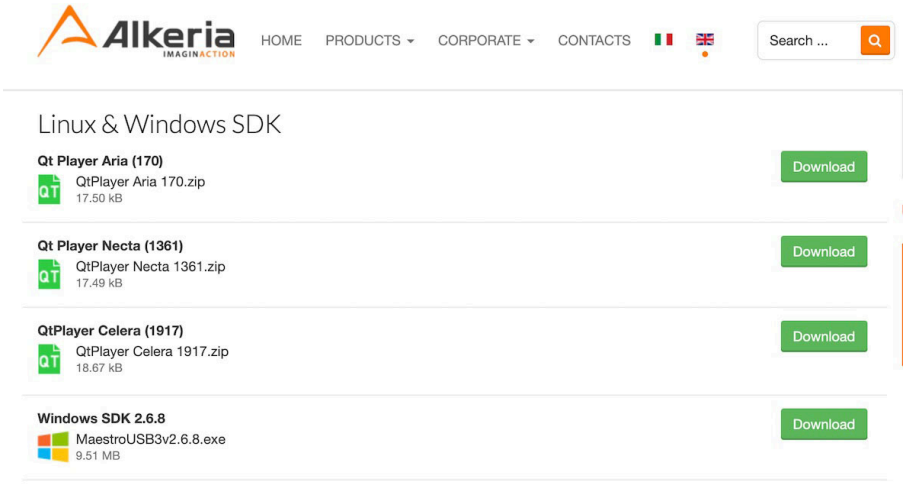


Figure 1.7: *Example of file listing inside Linux & Windows SDK category*

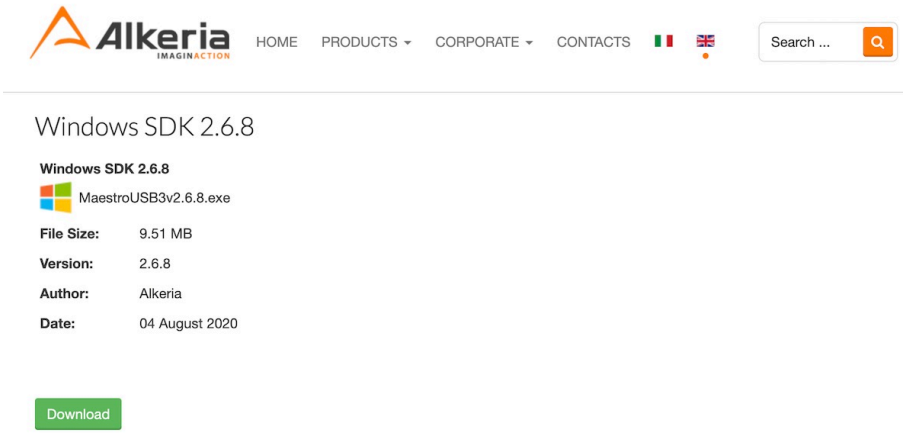


Figure 1.8: *Example of single file view*

Step 6: Click on the "Log out" orange button to end your session (Figure 1.8), then install the downloaded executable as administrator.

1.3 CONNECTIONS

Before starting, connect the camera to the PC following these steps:

Step 1: Mount the camera on a stable support;

Step 2: Mount the lens on the adapter;

Step 3: Connect the USB micro-B plug (Figure 1.2b) to the camera;

Step 4: Connect the USB type A plug (Figure 1.2a) to a SuperSpeed USB port on the PC.

1.4 FIRST START

1.4.1 Status LED

NECTA features a status LED on its back, showing the operating conditions of the camera. When a USB connector is plugged in, LED remains red until the computer detects and configures the camera. At that point the LED turns green: the camera is now ready to acquire. During acquisition, LED turns orange.

Different behaviours of the status LED may indicate that camera is malfunctioning. To a complete reference on the status LED codes, refer to Table 5.1.

1.4.2 Run Alkeria player

MaestroUSB3 comes with a user-friendly software application, allowing to explore the main features of NECTA cameras and check the correct camera and software installation. Once installed MaestroUSB3 SDK, you can find the application at the following path:

Program Files\Alkeria\USB3\MaestroUSB3\Players

Alternatively, you can easily start the Alkeria player by accessing the Windows Start button and look for *Alkeria player* from the MaestroUSB3 Program folder: Start->All programs->Alkeria->USB3->MaestroUSB3.

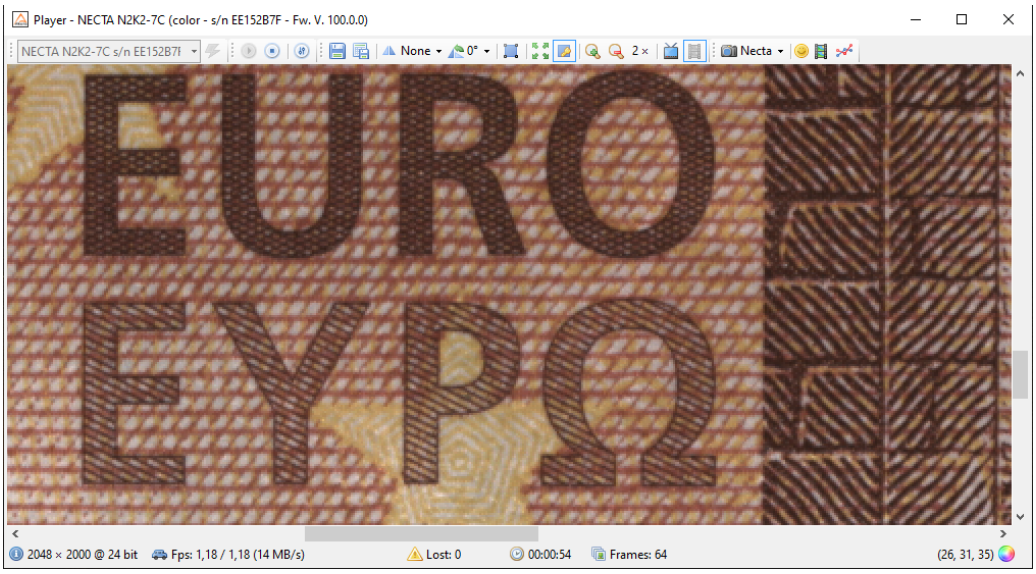


Figure 1.9: *Alkeria player main window*

A subset of the toolbar controls is shown in Table 1.1.

	Init	Initialize the camera to default settings (camera power-up status).
	Play	Start image playback.
	Stop	Stop image playback.
	Settings	Open the settings panel.

Table 1.1: *Relevant Alkeria player buttons*

Press “Play” button to start the acquisition of images from the camera. At the bottom of the player window a status bar displays the following data:

- The image resolution and the pixel depth;
- The frame rate;
- A frame lost counter, which indicates whether some frames have been corrupted during transfer and have been discarded;
- A timer counter indicating the acquisition time;
- A frame counter;

Pressing the “Settings” button, you will have access to the main camera acquisition parameters such as Shutter and Gain. Move them while NECTA is acquiring images to familiarize with the controls.

To get more information about player functionalities, please refer to Chapter 9.

1.4.3 How to get a good image

The camera is not yet able to provide a perfect image. A step of calibration is needed to cancel out all the differences between the individual pixels on the sensor, such as black level and individual pixel's gain.

To calibrate the camera press NECTA on the toolbar and then click on Calibration to open the wizard.

If you want to learn more about calibration, check out tutorials and guides on your website's User Area.

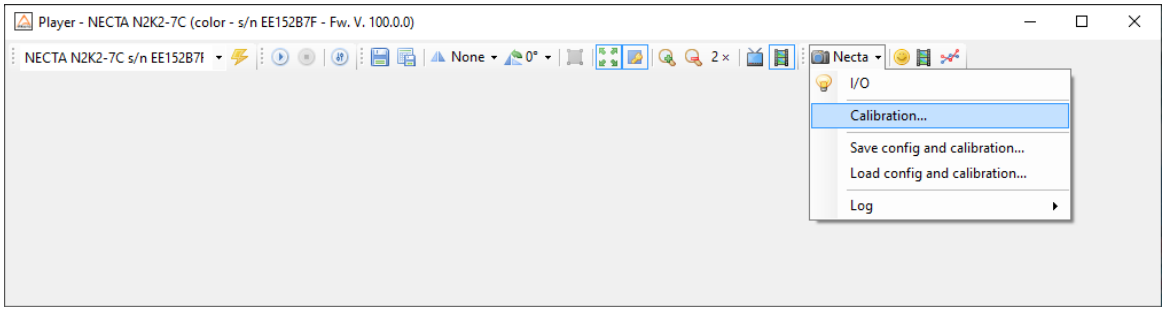


Figure 1.10: *Alkeria player Calibration Wizard*

2

Introducing NECTA Line Scan Cameras

NECTA is Alkeria's answer to the demand of its customers in industrial and medical field for high speed line scan cameras.

Featuring USB 3.2 Gen 1x1 interface, complying with the modern advance in machine vision, NECTA takes performances to a new level, being specifically designed to get the best from AMS (formerly AWAIBA) CMOS sensors, sharing the same compact, reliable and versatile form factor of CELERA, its area-scan counterpart. With advanced firmware features, uncompromised quality and a large FPGA for on-demand customization, NECTA is the flagship in Alkeria line-scan camera range.

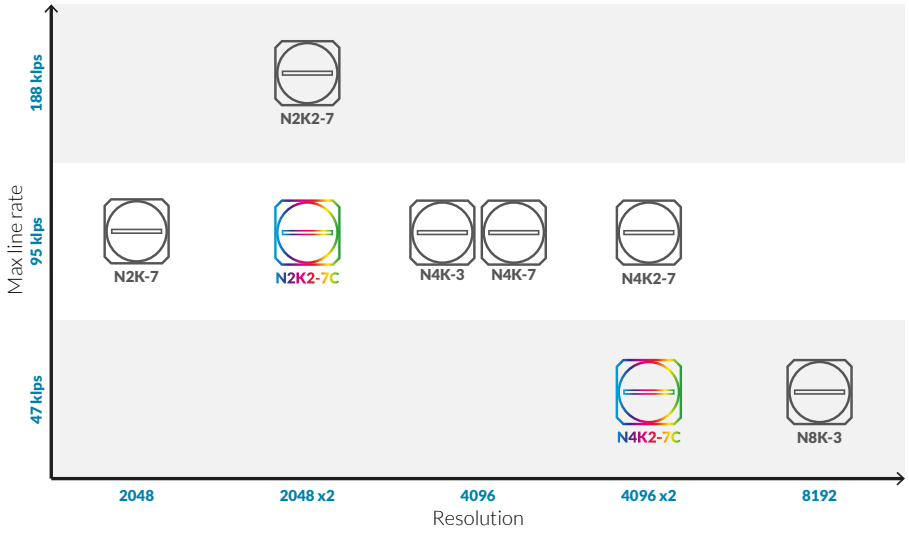


Figure 2.1: *NECTA cameras overview*

2.1 NECTA

NECTA is a family of ultra-compact line scan cameras with USB 3.2 Gen 1x1 interface. The adoption of new technology Complementary MOS (CMOS) linear sensors and USB 3.2 Gen 1x1 connectivity allows NECTA to reach very high acquisition speed while still using consumer-class interface.

NECTA is directly powered by the USB 3.2 Gen 1x1 bus eliminating the need for external power adapters. Reduced size and low power consumption, solid construction, low installation cost, ease of programming and versatility make NECTA devices suited for the most demanding industrial applications: in-line inspection, web inspection, document scanning, metrology.

An easy-to-use set of software API is available, which allows developers to quickly produce fast and clean code both on Windows and Linux, supporting all major programming languages, such as Python, C#, VB, C++ in Visual Studio or QT environment. Our Software Development Kit provided with NECTA cameras, is fully compatible with Cognex VisionPro, MVTech Halcon, NI LabView and other major machine vision software for industrial application.

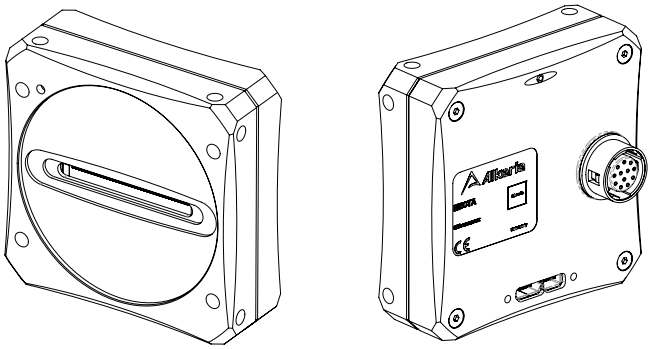


Figure 2.2: *NECTA camera*

2.2 NECTA FAMILY

NECTA cameras use high performance linear CMOS sensors. The models in the family differ in size, layout and number of pixels of the sensor. Table 2.1 shows all models and their characteristics. Some cameras use sensors featuring two lines and they are available both in BW and RGB version. All cameras are directly powered through the USB 3.2 Gen 1x1 connection. Except where explicitly specified, all statements herein refer to the entire NECTA family.

Model Name	Sensor	Resolution (H x W)	Chroma
NECTA N2K-7	AMS DR2K7	1 x 2048	Mono
NECTA N2K2-7	AMS DR2X2K7	2 x 2048	Mono
NECTA N2K2-7C	AMS DR2X2K7	2 x 2048	Color
NECTA N4K-3	AMS DR4K3.5	1 x 4096	Mono
NECTA N4K-7	AMS DR4K7	1 x 4096	Mono
NECTA N4K2-7	AMS DR2X4K7	2 x 4096	Mono
NECTA N4K2-7C	AMS DR2X4K7	2 x 4096	Color
NECTA N8K-3	AMS DR8K3.5	1 x 8192	Mono

Table 2.1: *NECTA cameras family overview*

2.2.1 Detailed Specifications

Specification	NECTA N2K-7
Sensor model	AMS DR2K7
Sensor type	Mono
Sensor resolution	1(H) x 2048(W)
Sensor technology	CMOS, fill factor 100%
Pixel size	7 µm × 7 µm
Max. line rate (ADC Res. 12 bit)	23 641 lps (MONO8, MONO16)
Max. line rate (ADC Res. 9 bit)	94 340 lps (MONO8, MONO16)
A/D Conversion	9 to 12 bits
Synchronization	External trigger, software trigger
Controls	Gain, shutter, frame rate, brightness, contrast, LUT, gamma, trigger
Shutter control	2 µs ÷ 100 ms (100 ns steps)
Binning factor	2x horizontal
Readout	Normal, ROI, binning, frame combiner, subsampling
Power supply	2 W max, powered by USB 3.2 Gen 1x1
Inputs/outputs	2 in, 2 out, 1 I/O (RS-422, RS-644, LV-CMOS, LV-TTL)
Lens adapter	C-mount, F-mount, M42
Interface	USB 3.2 Gen 1x1
Pixel formats	Mode 0: MONO8, MONO16; Mode 1: RAW8, RAW16
Weight	129 g (with C-mount adapter)
Size (without adapters)	56 mm x 56 mm x 26.7 mm
Size (with C-mount)	56 mm x 56 mm x 38.3 mm
Size (with F-mount)	56 mm x 56 mm x 67.3 mm
Conformity	CE, RoHS, FCC, IC
Operating temperature	0 ÷ 50 °C (referred to housing)
Software	MaestroUSB3
Link	`www.alkeria.com/products/NECTA-series`
Warranty	24 months

Table 2.2: *NECTA N2K-7 Specifications*

Specification	NECTA N2K2-7	NECTA N2K2-7C
Sensor model	AMS DR2X2K7	AMS DR2X2K7-RGB
Sensor type	Mono	Color
Sensor resolution	2(H) x 2048(W)	
Sensor technology	CMOS, fill factor 100%	
Pixel size	7 µm × 7 µm	
Max. line rate (ADC Res. 12 bit)	47 281 lps (all video modes)	23 641 lps (all video modes)
Max. line rate (ADC Res. 9 bit)	188 679 lps (MONO8) 95 238 lps (MONO16)	94 340 lps (MONO16, YUV422) 63 291 lps (RGB24)
A/D conversion	9 to 12 bits	
Synchronization	External trigger, software trigger	
Controls	Gain, shutter, frame rate, brightness, contrast, LUT, gamma, trigger	
Shutter control	2 µs ÷ 100 ms (100 ns steps)	
Binning factor	2x horizontal, 2x vertical	None
Readout	Normal, ROI, binning, frame combiner, subsampling	
Power supply	2.5 W max, powered by USB 3.2 Gen 1x1	
Inputs/outputs	2 in, 2 out, 1 I/O (RS-422, RS-644, LV-CMOS, LV-TTL)	
Lens adapter	C-mount, F-mount, M42	
Interface	USB 3.2 Gen 1x1	
Pixel formats	RAW8, RAW16, MONO8, MONO16	RAW16, MONO16, YUV422, RGB24
Weight	129 g (with C-mount adapter)	
Size (without adapters)	56 mm x 56 mm x 26.7 mm	
Size (with C-mount)	56 mm x 56 mm x 38.3 mm	
Size (with F-mount)	56 mm x 56 mm x 67.3 mm	
Conformity	CE, RoHS, FCC, IC	
Operating temperature	0 ÷ 50 °C (referred to housing)	
Software	MaestroUSB3	
Link	`www.alkeria.com/products/NECTA-series`	
Warranty	24 months	

Table 2.3: *NECTA N2K2-7 / N2K2-7C Specifications*

Specification	NECTA N4K-3
Sensor model	AMS DR4K3.5
Sensor type	Mono
Sensor resolution	1(H) x 4096(W)
Sensor technology	CMOS, fill factor 100%
Pixel size	3.5 µm × 3.5 µm
Max. line rate (ADC Res. 12 bit)	23 641 lps (MONO8, MONO16)
Max. line rate (ADC Res. 9 bit)	94 340 lps (MONO8) 47 619 lps (MONO16)
A/D conversion	9 to 12 bits
Synchronization	External trigger, software trigger
Controls	Gain, shutter, frame rate, brightness, contrast, LUT, gamma, trigger
Shutter control	2 µs ÷ 100 ms (100 ns steps)
Binning factor	2x horizontal
Readout	Normal, ROI, binning, frame combiner, subsampling
Power supply	2.5 W max, powered by USB 3.2 Gen 1x1
Inputs/outputs	2 in, 2 out, 1 I/O (RS-422, RS-644, LV-CMOS, LV-TTL)
Lens adapter	C-mount, F-mount, M42
Interface	USB 3.2 Gen 1x1
Pixel formats	Mode 0: MONO8, MONO16; Mode 1: RAW8, RAW16
Weight	129 g (with C-mount adapter)
Size (without adapters)	56 mm x 56 mm x 26.7 mm
Size (with C-mount)	56 mm x 56 mm x 38.3 mm
Size (with F-mount)	56 mm x 56 mm x 67.3 mm
Conformity	CE, RoHS, FCC, IC
Operating temperature	0 ÷ 50 °C (referred to housing)
Software	MaestroUSB3
Link	`www.alkeria.com/products/NECTA-series`
Warranty	24 months

Table 2.4: *NECTA N4K-3 Specifications*

Specification	NECTA N4K-7
Sensor model	AMS DR4K7
Sensor type	Mono
Sensor resolution	1(H) x 4096(W)
Sensor technology	CMOS, fill factor 100%
Pixel size	7 µm × 7 µm
Max. line rate (ADC Res. 12 bit)	23 641 lps (MONO8, MONO16)
Max. line rate (ADC Res. 9 bit)	94 340 lps (MONO8) 47 619 lps (MONO16)
A/D conversion	9 to 12 bits
Synchronization	External trigger, software trigger
Controls	Gain, shutter, frame rate, brightness, contrast, LUT, gamma, trigger
Shutter control	2 µs ÷ 100 ms (100 ns steps)
Binning factor	2x horizontal
Readout	Normal, ROI, binning, frame combiner, subsampling
Power supply	2.5 W max, powered by USB 3.2 Gen 1x1
Inputs/outputs	2 in, 2 out, 1 I/O (RS-422, RS-644, LV-CMOS, LV-TTL)
Lens adapter	C-mount, F-mount, M42
Interface	USB 3.2 Gen 1x1
Pixel formats	Mode 0: MONO8, MONO16; Mode 1: RAW8, RAW16
Weight	129 g (with C-mount adapter)
Size (without adapters)	56 mm x 56 mm x 26.7 mm
Size (with C-mount)	56 mm x 56 mm x 38.3 mm
Size (with F-mount)	56 mm x 56 mm x 67.3 mm
Conformity	CE, RoHS, FCC, IC
Operating temperature	0 ÷ 50 °C (referred to housing)
Software	MaestroUSB3
Link	`www.alkeria.com/products/NECTA-series`
Warranty	24 months

Table 2.5: *NECTA N4K-7 Specifications*

Specification	NECTA N4K2-7	NECTA N4K2-7C
Sensor model	AMS DR4X2K7	AMS DR4X2K7-RGB
Sensor type	Mono	Color
Sensor resolution	2(H) x 4096(W)	
Sensor technology	CMOS, fill factor 100%	
Pixel size	7 µm × 7 µm	
Max. line rate (ADC Res. 12 bit)	47 281 lps (all video modes)	23 641 lps (all video modes)
Max. line rate (ADC Res. 9 bit)	95 238 lps (MONO8) 47 733 lps (MONO16)	47 619 lps (MONO16, YUV422) 31 746 lps (RGB24)
A/D conversion	9 to 12 bits	
Synchronization	External trigger, software trigger	
Controls	Gain, shutter, frame rate, brightness, contrast, LUT, gamma, trigger	
Shutter control	2 µs ÷ 100 ms (100 ns steps)	
Binning factor	2x horizontal, 2x vertical	None
Readout	Normal, ROI, binning, frame combiner, subsampling	
Power supply	2.5 W max, powered by USB 3.2 Gen 1x1	
Inputs/outputs	2 in, 2 out, 1 I/O (RS-422, RS-644, LV-CMOS, LV-TTL)	
Lens adapter	C-mount, F-mount, M42	
Interface	USB 3.2 Gen 1x1	
Pixel formats	RAW8, RAW16, MONO8, MONO16	RAW16, MONO16, YUV422, RGB24
Weight	129 g (with C-mount adapter)	
Size (without adapters)	56 mm x 56 mm x 26.7 mm	
Size (with C-mount)	56 mm x 56 mm x 38.3 mm	
Size (with F-mount)	56 mm x 56 mm x 67.3 mm	
Conformity	CE, RoHS, FCC, IC	
Operating temperature	0 ÷ 50 °C (referred to housing)	
Software	MaestroUSB3	
Link	`www.alkeria.com/products/NECTA-series`	
Warranty	24 months	

Table 2.6: *NECTA N4K2-7 / N4K2-7C Specifications*

Specification	NECTA N8K-3
Sensor model	AMS DR8K3.5
Sensor type	Mono
Sensor resolution	1(H) x 8192(W)
Sensor technology	CMOS, fill factor 100%
Pixel size	3.5 µm × 3.5 µm
Max. line rate (ADC Res. 12 bit)	23 641 lps (MONO8, MONO16)
Max. line rate (ADC Res. 9 bit)	47 619 lps (MONO8) 23 866 lps (MONO16)
A/D conversion	9 to 12 bits
Synchronization	External trigger, software trigger
Controls	Gain, shutter, frame rate, brightness, contrast, LUT, gamma, trigger
Shutter control	2 µs ÷ 100 ms (100 ns steps)
Binning factor	2x horizontal
Readout	Normal, ROI, binning, frame combiner, subsampling
Power supply	2.5 W max, powered by USB 3.2 Gen 1x1
Inputs/outputs	2 in, 2 out, 1 I/O (RS-422, RS-644, LV-CMOS, LV-TTL)
Lens adapter	C-mount, F-mount, M42
Interface	USB 3.2 Gen 1x1
Pixel formats	Mode 0: MONO8, MONO16; Mode 1: RAW8, RAW16
Weight	129 g (with C-mount adapter)
Size (without adapters)	56 mm x 56 mm x 26.7 mm
Size (with C-mount)	56 mm x 56 mm x 38.3 mm
Size (with F-mount)	56 mm x 56 mm x 67.3 mm
Conformity	CE, RoHS, FCC, IC
Operating temperature	0 ÷ 50 °C (referred to housing)
Software	MaestroUSB3
Link	`www.alkeria.com/products/NECTA-series`
Warranty	24 months

Table 2.7: *NECTA N8K-3 Specifications*

2.3 OPTIONAL ACCESSORIES

NECTA cameras can be completed with the following accessories:

- NAD-C adapter to C-mount lens (see Section 3.4.2)
- NAD-F adapter to Nikon F-mount lens (see Section 3.4.3)
- NIO-x cable for I/O interconnection – 3, 5, 10 m long available (see Section 4.1).

2.4 ORDERING INFORMATIONS

Alkeria has various camera series in its product line: each series has several models, with different resolutions, color modes and accessories.
To identify a specific model version, Alkeria uses an internal codename system that allows to describe every camera specification with a unique code.
Understanding how this code works can be useful to our customers both to identify correctly a camera and to request information or ordering to our sales department.
Every camera code is composed by the following parts:

Camera model

This is the first and main part: it includes camera series and information about sensor like resolution and number of lines.

Color option

The second part, separated by a hyphen, defines the sensor color option for each camera model. In our product line we use different sensor families, most of them available in color, monochrome or Near-InfraRed (NIR) mode.

Lens adapter

Last part of the code, indicates which lens adapter the camera is equipped with. It's the only part that can be omitted, in the case you're choosing a camera without any additional lens adapters.

Figure 2.3: *NECTA Ordering information*

2.5 USB 3.2 GEN 1X1 SUPERSPEED INTERFACE

NECTA cameras exploit the USB 3.2 Gen 1x1 standard interface to fulfil the bandwidth requirements of fast image sensors and communicate with PC. In order to maximize usable bandwidth of installed USB 3.2 Gen 1x1 controller, user can choose between *isochronous endpoint* and *bulk endpoint* for transmitting video stream. The command stream (controlling camera features) is supported by a bidirectional interrupt endpoint providing quick response and low latency.

2.5.1 Isochronous endpoint

The super-speed *isochronous endpoint* used by NECTA allows up to 3 burst transactions for each 125 µs service interval (micro-frame); each burst transaction supports up to 16 packets made by 1024 B, reaching up to 3x16x1024 B as payload for a single micro-frame. Total allowed bandwidth is 375 MiB/s for NECTA USB 3.2 Gen 1x1 connection. The *isochronous endpoint* allows user to reserve a given amount of bandwidth for each device, regardless of the connected device number to the host.

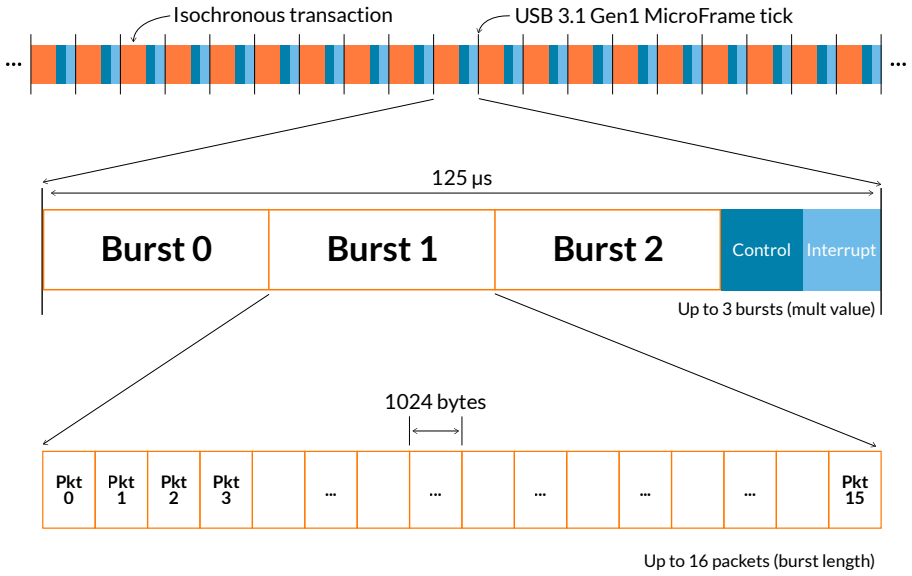


Figure 2.4: *USB 3.2 Gen 1x1 Isochronous transactions structure*

In order to exploit the maximum throughput, the computer must be able to support that load, therefore particular attention must be paid in choosing cables and USB 3.2 Gen 1x1 host controllers (see Section 5.1 for details).

2.5.2 Bulk endpoint

The super-speed *bulk endpoint* used by NECTA transfers data using bursts of 1 to 16 packets, 1024 B long. The theoretical maximum bandwidth is 500 MiB/s for a single host, even if the camera will only use up to 375 MiB/s of the available bandwidth. Unlike an *isochronous endpoint*, a *bulk endpoint* can not reserve the bandwidth for itself: all the available bandwidth is shared with all the other devices present on the same host, even if connected to other ports. For this reason, when a NECTA is connected to one USB3 port, it is not recommended to connect other USB devices, such as USB drives or Webcams, to the same host. When a NECTA camera is connected as *bulk endpoint*, frame loss phenomena will occur when other devices will be operated.

2.5.3 USB 3.2 Gen 1x1 controller cards

Since each host controller performs better or with *isochronous* or *bulk endpoints*, it is recommended to make extensive testings to determine the best controller-endpoint configuration. This is especially true if you want to reach the maximum speed. To exploit the maximum achievable performance of the USB 3.2 Gen 1x1 controller, start with a low bandwidth limit (Section 9.3.4), e.g. 16 KiB, and start the acquisition. If no error is raised, stop the acquisition, increase the limit and repeat. If the controller is not able to support the selected bandwidth, a pop-up may appear or the application will freeze. To recover from this condition, unplug the USB cable from the camera and reconnect it. The maximum achievable bandwidth is the higher selected which raised no errors.

2.6 INSTALLATION REQUIREMENTS

For proper NECTA performance, the controlling PC must meet the minimum requirements listed in Table 2.8 for single camera applications and in Table 2.9 for multi camera applications.

CPU	Intel Core i5 or AMD Ryzen 5
RAM	8 GB
Operating System	Windows 7/8/10, Ubuntu 20.04
USB 3.2 Gen 1x1 Controller	Based on Fresco Logic FL1100EX host or similar, plugged in a PCI Express 2.0 slot

Table 2.8: *Minimum system requirements for single camera applications*

CPU	Intel Core i7 or AMD Ryzen 7
RAM	16 GB
Operating System	Windows 7/8/10, Ubuntu 20.04
USB 3.2 Gen 1x1 Controller	Based on Fresco Logic FL1100EX host or similar, plugged in a PCI Express 2.0 slot

Table 2.9: *Minimum system requirements for multi-camera applications*

For further information on recommended USB host controllers, refer to Section 5.1.2.

In some operating conditions, the USB 3.2 Gen 1x1 data originated by NECTA may reach 3.2 Gbit/s. PCI slots hosting USB 3.2 Gen 1x1 controllers must be able to support PCI Express 2.0 or higher in order to sustain the required throughput; insufficient PCI bandwidth may limit camera performance (see also Section 5.1). The camera case allows using USB 3.2 Gen 1x1 cables with screw-lock connector; it is highly recommended to adopt them in the presence of shocks and vibrations that may affect the USB connection to the computer.

2.7 MAESTROUSB3 SDK

The NECTA family is supported by MaestroUSB3, a user-friendly software development system allowing quick code development for the MS Visual BASIC .NET/C++/C#. C++ development under Linux-based operating systems is supported through the Linux SDK. MaestroUSB3 is provided, for free use,

upon purchase of NECTA cameras. Refer to the software manual available along with MaestroUSB3 SDK for a full description of the development system, the installation procedures, software functions and source code examples.

Note

All code examples in this document are written in C#.

2.7.1 Software License

NECTA customers are granted free use of MaestroUSB3. For details about usage conditions, refer to the license agreement contained within the SDK and submitted for approval during installation.

2.8 SAFETY AND PRECAUTIONS

Electric shock hazard

Unapproved power supplies may cause electric shock. Serious injury or even death may occur. Computer equipment connected to the camera must meet the Safety Extra Low Voltage (SELV) and Limited Power Source (LPS) requirements.

Powered HUBs connected between the PC and the camera must meet the Safety Extra Low Voltage (SELV) and Limited Power Source (LPS) requirements as well.

All other equipment connected to the triggers or other ports must also have double insulation/reinforced isolation protection from the mains supply and its power supply must meet the Safety Extra Low Voltage (SELV) and Limited Power Source (LPS) requirements.

Remember that the camera case is made of conductive aluminum. Always check your mounting and avoid it to get in contact with wires or surfaces at dangerous electric potential.

Do NOT touch the camera with wet hands. Doing so may cause electric shock.

Fire hazard

Unapproved power supplies may cause fire and burns. Serious injury or even death may occur. Computer equipment connected to the camera must meet the Limited Power Source (LPS) requirements.

Powered HUBs connected between the PC and the camera must meet the Safety Extra Low Voltage (SELV) and Limited Power Source (LPS) requirements as well.

Foreign matter e.g. liquid, flammable or metallic materials may cause fire if inserted into camera during operations.

The camera housing can become very hot during operation. Do not operate the camera without lens or out of the recommended ambient temperature range. Always ensure a good airflow to cool down the camera housing. Read carefully Section 3.6.

Cautions

Do not expose the camera to X-rays. Exposition to X-rays may permanently damage the camera sensor. This kind of damage is not covered by warranty.

Never expose the sensor to laser sources or high-energy light. A constant exposure to light levels exceeding sensor saturation capability may lead to sensor irreversible failure.

Do not remove the camera's label. If the serial number can't be retrieved neither from camera registers nor from the label, the warranty is void.

Do not open the camera housing. There are no user serviceable parts inside. Internal components may be damaged by touching them. The warranty is void if seals are broken.

Use only the recommended 12-pin Hirose plug for I/O interface. Camera connector can be damaged by plugs with different number of pins. Preassembled cables of different lengths are available as an option and can ease camera connections (see Section 2.3).

An incorrect electric connection or pin assignment may severely damage the camera.

NECTA is designed to fit only USB 3.2 Gen 1x1 Micro-B connectors. Any other type of plugs may compromise the port connectivity.

Camera operating voltage is 5 VDC (directly powered through the USB connection). Camera must be supplied only with USB 3.2 Gen 1x1 compliant sources.

Disassemble, drop, try to repair or alter your NECTA Camera in any way may damage it and possibly cause electric shocks. Keep unsupervised children far from the device.

Should any liquid (like water, beverages or chemical substances) flow into the camera, STOP using it IMMEDIATELY and ask your distributor or Alkeria SRL for technical support.

Store your NECTA Camera with the lens opening covered to avoid dust contamination.

During operations, the temperature of the camera case should be kept in the 0-50 °C range.

3 Hardware Description

3.1 SENSORS

NECTA line scan cameras employ last-generation linear CMOS sensors manufactured by AMS, featuring an ADC module for each pixel which allows very high-speed acquisition. The tables in Section 2.2.1 help identifying the sensor used in each camera model, indicating the resolution and size of the pixel cell; sensors whose size is indicated by "2x..." are made by two sensitive pixel lines, parallel and contiguous.

3.1.1 Monochrome Quantum Efficiency

Figure 3.2 shows the quantum efficiency of the sensors used by NECTA cameras. The graph refers to the sensors used in monochrome variants; to derive the quantum efficiency of NECTA color cameras it is necessary to take into account also the transmittance of the Bayer filter, as reported in Section 3.1.2.

3.1.2 Color Quantum Efficiency

AMS sensors discriminate colors using the Bayer color filter array shown in Figure 3.1, that makes each pixel sensitive to only one of the primary colors. To recover the quantum efficiency of a color sensor camera, it is sufficient to combine the monochrome quantum efficiency with the transmittance of the Bayer filter array (Figure 3.3) for every wavelength. Solid lines in Figure 3.3 refer to the maximum achievable transmission, dashed lines to minimum guaranteed transmission index.

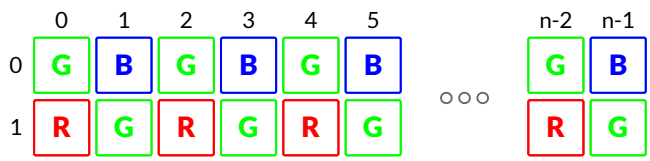


Figure 3.1: *Bayer pattern of color filters*

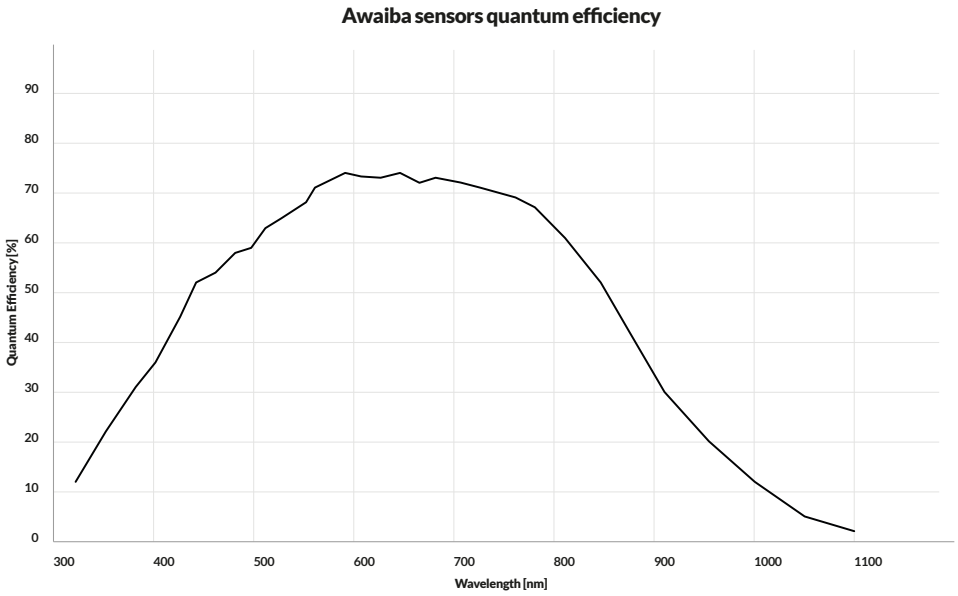


Figure 3.2: *Quantum efficiency of NECTA monochrome sensors - Source: AMS*

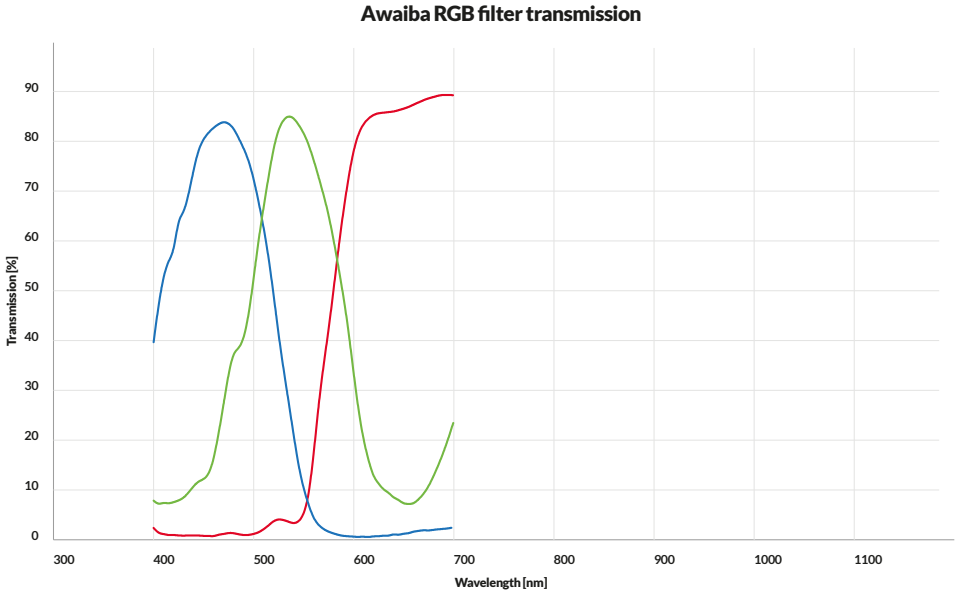
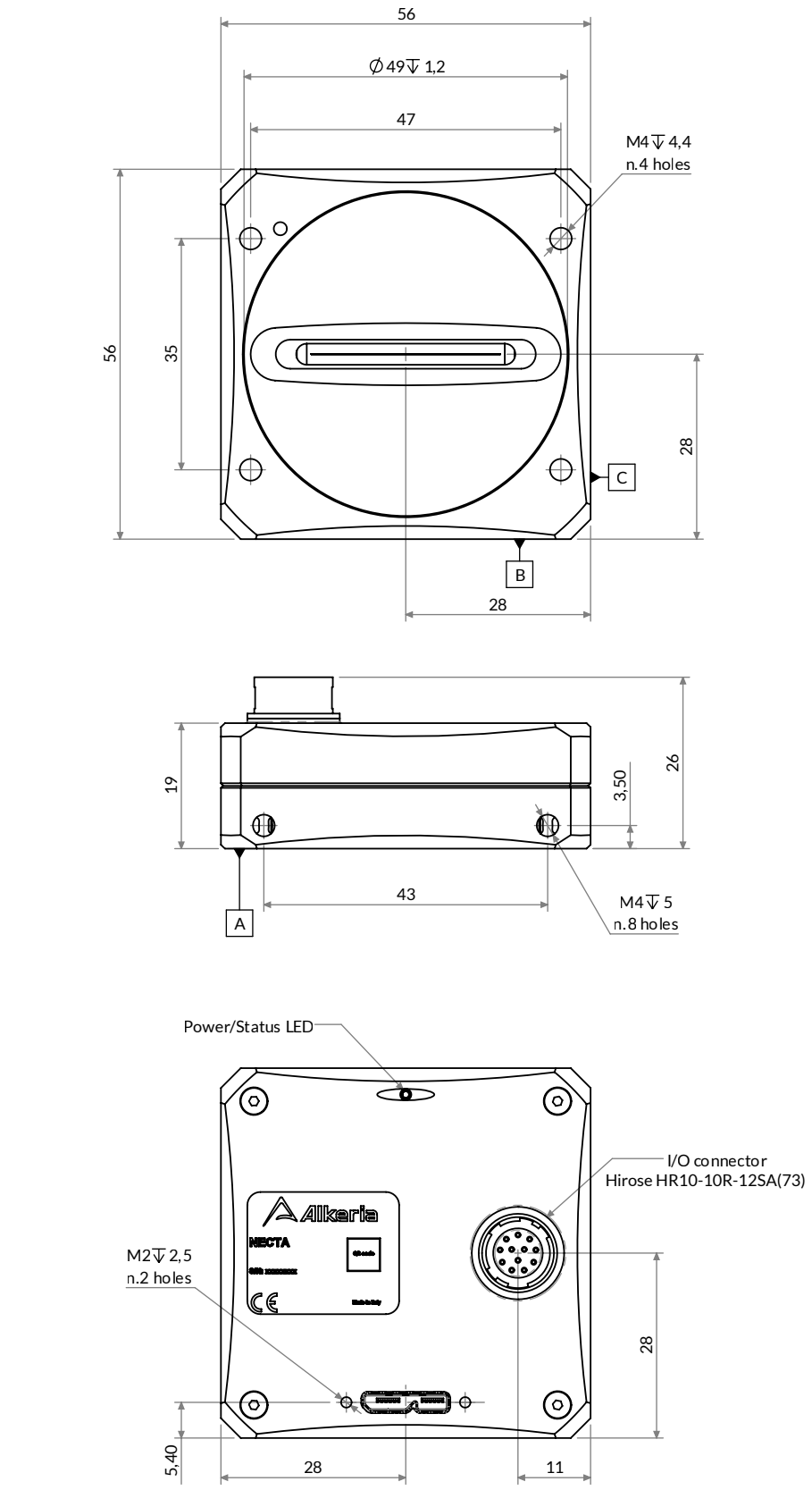


Figure 3.3: *Bayer filter transmission of NECTA color sensors - Source: AMS*

3.2 MECHANICAL DESCRIPTION

The housing of NECTA cameras is made of black anodized aluminum alloy, using high precision machining. It is provided with four fixing front holes (M4), intended for custom lens or lens adapter connection, and two M4 holes at each side of the camera, for connecting to the mount, as shown by the drawings in Figure 3.4. When designing the bracket that will support your NECTA , refer to the reference planes (marked as A, B and C); using these reference planes will allow to achieve a precise positioning of the devices with significant repeatability. When the lens is mounted, the camera housing conforms to protection class IP30.



Dimensions in mm
A - B - C Reference Datum

Figure 3.4: *NECTA camera dimensions (without lens adapter)*

3.3 SENSOR POSITION

Figure 3.5 shows the position and orientation of the sensitive area of the NECTA sensor referred to the main reference plane. The marking in the upper left in the figure (marked as *Sensor reference*) helps identifying the direction of the sensor and the position of the first pixel of its sensitive area.

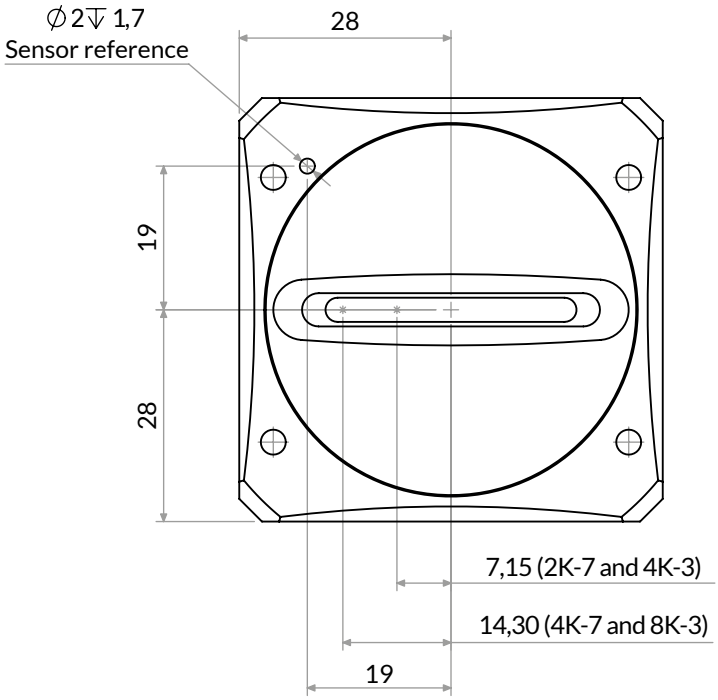


Figure 3.5: *NECTA sensors position and orientation*

3.4 ADAPTERS FOR STANDARD LENS

NECTA family has some adapters available (described in Table 3.1) which allow using standard lens. The housing of NECTA cameras also allows to easily build custom adapters for optical lens, to achieve the performance, economy and precision level required even by the most demanding application.

3.4.1 Choosing the right adapter

Due to sensor width, not all NECTA camera can be equipped with C-Mount lens mount. Table 3.1 shows a compatibility diagram between NECTA models and available lens adapters.

Adapter	N2K-7	N2K2-7	N2K2-7C	N4K-3	N4K-7	N4K2-7	N4K2-7C	N8K-3
C-Mount	✓	✓	✓	✓	N.A	N.A	N.A	N.A
F-Mount	✓	✓	✓	✓	✓	✓	✓	✓

Table 3.1: *Adapter compatibility with NECTA cameras*

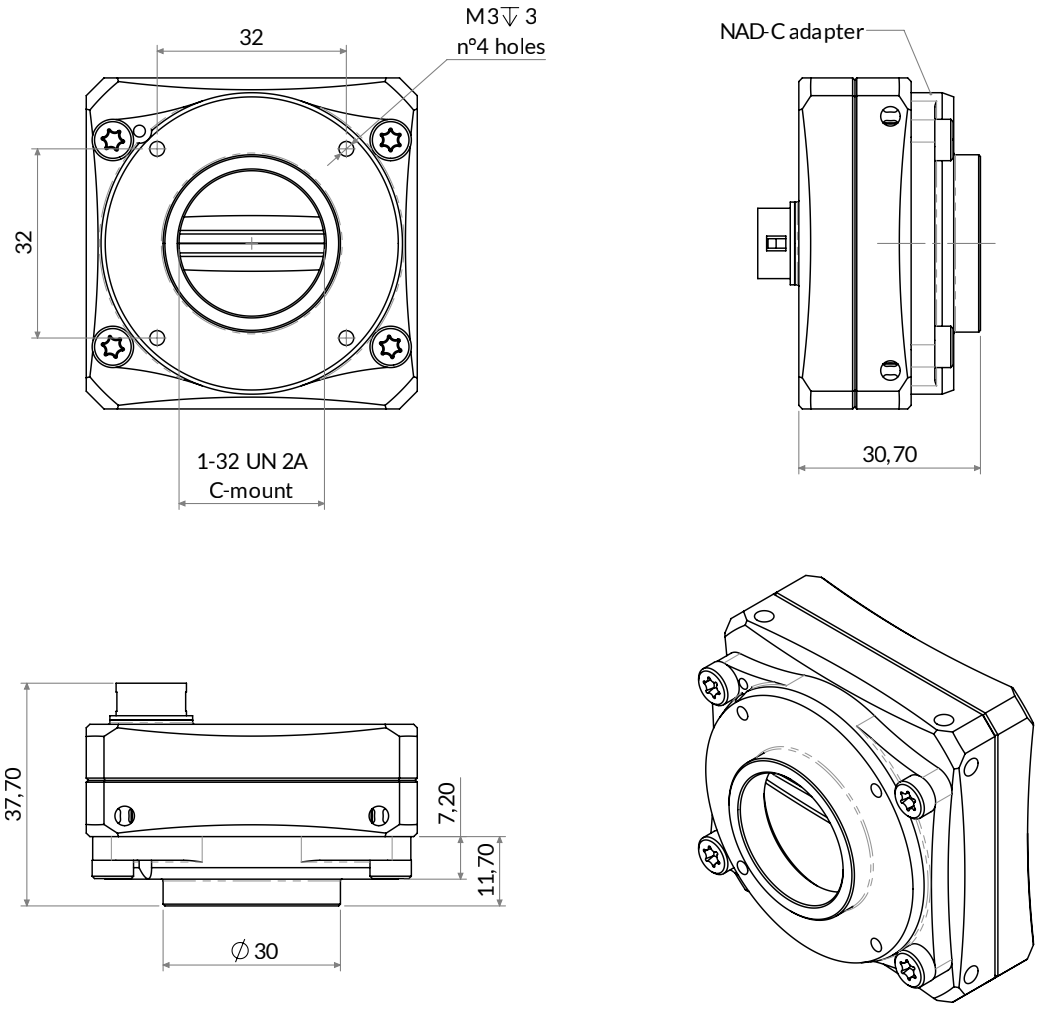


Figure 3.6: *NAD-C C-Mount mechanical drawings*

3.4.2 C-Mount adapter

The NAD-C adapter allows interfacing some NECTA camera models (N2K-7, N2K2-7, N2K2-7C and N4K-3) with standard C-mount lenses. To connect the adapter, simply align it to the holes in front of the camera and tighten the four M4 hex screws supplied with the adapter. If you need to disassemble it, just unscrew the four M4 hex front screws. This operation should be performed in a clean and dust-free environment.

Warning

The maximum tightening torque for screws is 2.0 N m. Exceeding the maximum tightening torque may damage the camera.

Caution

Do not contaminate the sensitive area of the camera. Avoid exposing optical parts to dust and dirt during handling operations; always operate in a clean working area, as dry as possible and free from dust. For best results, keep the camera facing downwards during operations.

3.4.2.1 Maximum lens protrusion

Maximum protrusion for a lens is the distance from the front surface of the sensor to the mounting surface of lens flange. NAD-C C-Mount adapter allows to install only lens with maximum protrusion of 12.70 mm.

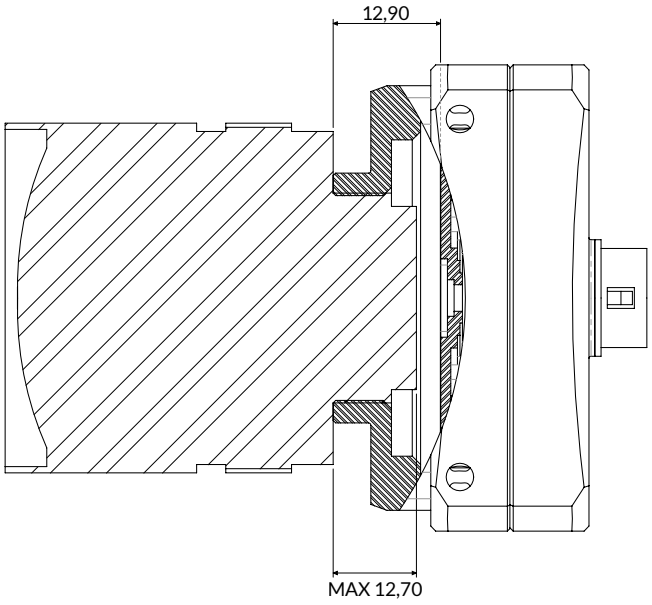


Figure 3.7: *Maximum protrusion for C-Mount type lens*

Warning

Mounting a lens with a protrusion longer that the maximum admitted value may seriously damage your NECTA camera and your lens.

3.4.3 F-Mount adapter

The NAD-F adapter allows interfacing any NECTA camera models with standard F-mount lenses. To connect the adapter, simply align the NAD-F adapter to the front holes of the camera and tighten the four M4 hex screws supplied with it.

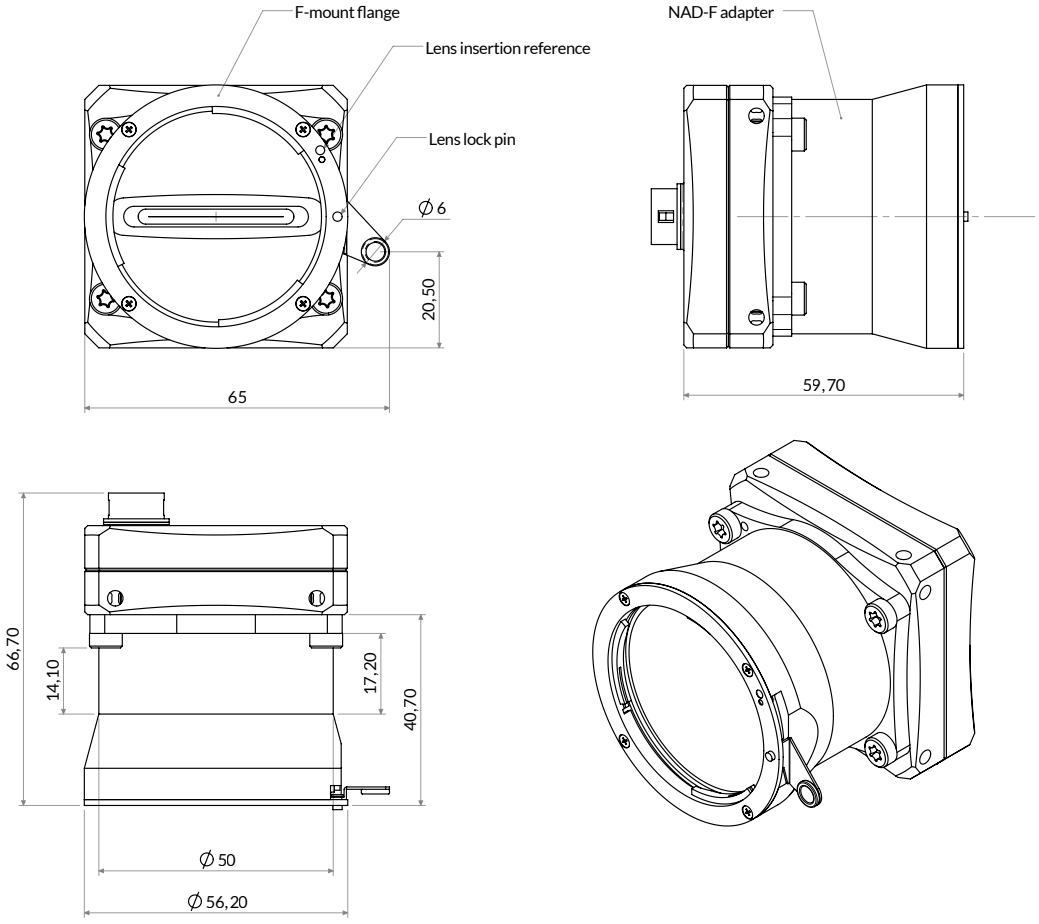


Figure 3.8: *NAD-F F-Mount mechanical drawings*

Warning

The maximum tightening torque for screws is 2.0 N m. Exceeding the maximum tightening torque may damage the camera.

Caution

Do not contaminate the sensitive area of the camera! Avoid exposing optical parts to dust and dirt during handling operations; always operate in a clean working area, as dry as possible and free from dust. For best results, keep the camera facing downwards during operations.

3.5 FLANGE FOCAL DISTANCE

The distance from the mounting flange to the sensor surface is called *flange focal distance*.

NECTA camera equipped with C-Mount adapter has a nominal flange focal distance FFD_C of 17.526 mm.

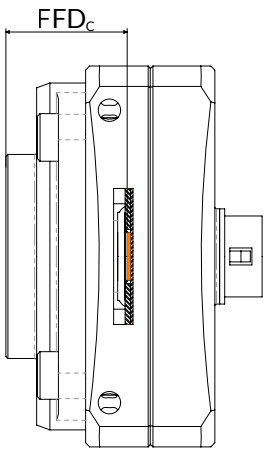


Figure 3.9: *NECTA camera equipped with C-Mount*

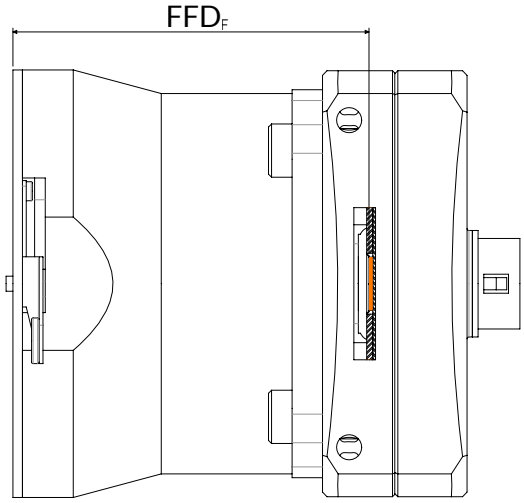


Figure 3.10: *NECTA camera equipped with F-Mount*

NECTA camera equipped with F-Mount adapter has a nominal flange focal distance FFD_F of 46.5 mm.

3.6 HANDLING PRECAUTIONS

The optical parts of NECTA cameras are assembled in a clean environment and the camera lens mount is factory sealed using a protective film, preserving optical parts from being contaminated by dust. Remove the seal only just before connecting the lens. Never leave the mount hole uncapped. Perform the required operations quickly and without hesitation, keeping the camera facing downwards. It is recommended to carry out all the operations that may contaminate the optical parts (like removing the protective seal) in an appropriate environment, free from dust and humidity.

3.6.1 Temperature and humidity

During operations, the case of the camera must remain below 50 °C. Otherwise sensitivity, linearity, dynamics and signal/noise ratio may be degraded.

	Min	Max	Notes
Housing temperature during operation	0 °C	50 °C	
Ambiental humidity during operation	20 %	80 %	Relative, non-condensing
Storage temperature	−15 °C	80 °C	
Storage humidity	20 %	80 %	Relative, non-condensing

Table 3.2: *Ambiental requirements*

3.6.2 Warranty limitations

Removing the label, opening or mishandling the camera will void its warranty.

3.6.3 Mechanical stress

NECTA cameras are made using high-precision machining, taking care of sensitive parts positioning. To maintain such accuracy is mandatory to avoid shock and stress that can deform reference surfaces. NECTA connector can host cables with screws or locking systems; it is highly recommended to use these cables when NECTA is exposed to shocks, vibrations or harsh environments.

3.6.4 Heat dissipation

It is very important to provide good heat dissipation in order to maintain the camera housing temperature as low as possible. Operating the camera at high temperature may impact negatively on image quality and may shorten camera life.

Warning

The camera housing may overheat during acquisition. Without proper heat dissipation image quality may be degraded and, in extreme environmental conditions, damage to the camera may occur. Never exceed the specified temperature limit during operation.

Since each installation is different, the following recommendations must be regarded to as a starting point for a correct thermal management, keeping in mind that these general guidelines must be customized by an experienced technician.

- Make sure a lens is always mounted on the camera. The lens acts as a heatsink and contributes to keep the system temperature low. The bigger the lens, the best it will work in keeping your camera cool. If your lens is not directly connected to the camera, always mount the camera on an aluminum or copper block in order to enlarge the radiating surface.
- Always monitor the camera housing temperature during the design phase of your project and keep an adequate safety margin in order to be sure that the case temperature will never exceed the maximum specified, even in worst environmental conditions.

- Provide adequate heat dissipation by mounting the camera on a wide thermally conductive surface that can act as heatsink and use a fan in order to provide forced air flow over the camera.
- Grant enough air circulation between camera and other system components, especially hot ones.
- Do not cover the camera case with thermally insulant materials like plastics films or cases.

3.6.5 Monitoring internal temperature

During continuous acquisition, NECTA temperature keeps rising until a steady temperature state is reached. NECTA provides internal temperature reading through an integrated sensor. Depending on the airflow, the internal temperature may be 10 °C to 20 °C higher than the housing. Continuous monitoring of the internal temperature is recommended to guarantee that camera is performing correctly. Actions must be taken if the internal temperature rises above 70 °C. Refer to Section 9.3.1 to know more about reading main board temperature using the NECTA viewer. Moreover, you can implement temperature monitoring by yourself using the following code:

Example Code 3.1 | *Temperature Monitoring*

```
// Read the current temperature
float temp = device.Temperature;
// Retrieve the temperature measurement unit
UnitInfo unit = device.GetFeatureUnitInfo(Feature.Temperature);
MessageBox.Show(String.Format("Temperature: {0} {1}", temp, unit.Unit.ToString()));
```

3.6.6 Automatic internal temperature safety mechanism

An automatic temperature monitoring mechanism, shown in Figure 3.11, has been implemented in NECTA cameras. The current temperature status can be retrieved reading the `OverTemperatureStatus` property. Three statuses are defined:

- **Normal**: Temperature is below 70 °C. The camera is working within the safe operation area and standard functionality is guaranteed.
- **Warning**: Temperature exceeded 70 °C. The camera is still able to acquire images but the temperature is very high and near to the upper operating limit. Working in this limit condition is definitely not recommended. To return in *Normal* status the camera temperature must fall below 65 °C.
- **Fault**: Temperature exceeded the 80 °C operative limit and the image acquisition was suspended to reduce the temperature below the fault threshold. The acquisition can be resumed only when the temperature is less than 75 °C.

Software temperature monitoring thread may be activated using `OverTemperatureMonitorEnable` property. When the temperature monitoring thread is active, the `OverTemperatureStatusChanged` event is fired every time the `OverTemperatureStatus` changes. The following code shows how to register on the event and activate the thread:

Figure 3.11: *Temperature status transition*

Example Code 3.2 | *Temperature Change Event*

```
device.OverTemperatureStatusChanged += delegate(object o, EventArgs a)
{
String msg = String.Format("Status: {0}", device.OverTemperatureStatus);
MessageBox.Show(msg);
};
device.OverTemperatureMonitorEnable = true;
```

3.6.7 EMI/ESD precautions: noise and electrostatic discharge

NECTA cameras are typically used in industrial environments, where many devices may generate electromagnetic interference and electrostatic discharge. Although NECTA cameras are designed to be extremely resistant to perturbations from the external environment, strong EMI and ESD transients may cause unwanted behavior, such as false triggering and defects in an acquired image. Those issues can be fixed by adopting good practice in the harness and cable routing. To reduce EMI and ESD, it is therefore essential to take some good general precautions when connecting the equipments:

- Install the camera as far as possible from high current loads such as motors or solenoids, especially if powered by a switching power supply.
- Use cables with double shielding and keep them as short as possible. Keep the contact area always clean and protected from dust.
- Separate power cables from signal cables. Avoid power cables running parallel to signal ones. This recommendation applies both to camera cables and to any other cable used in the equipment.
- Pay attention to the ground connections: they must be implemented through a single reference node, avoiding ground loops. Keeping your system powered from a single outlet will greatly reduce ground loop related problems.
- Install the camera far away from devices generating sparks or strong electromagnetic fields, such as, for example, large transformers, welding machines and electric motors.

3.7 CLEANING NECTA CAMERA

Each NECTA camera is carefully cleaned and checked before shipment. Nevertheless, using the camera in unclean environments may let dust enter the optical path. Since cleaning procedures require time and care and can be performed by trained personnel only, the best way of keeping your camera clean is avoiding extraneous matters enter the camera. Improper cleaning procedures may damage camera components. Contact Alkeria SRL if you are not familiar with the procedures described below. Any damage to sensor, filter, protection glass and camera housing occurred during cleaning procedures is not covered by Alkeria SRL warranty.

3.7.1 Before starting

Risk of electric shock

Disconnect USB cable from the camera.

Disconnect the I/O cable, if present.

Wear ESD-safe clothes and operate in an ESD-safe environment to avoid damaging the camera (never use synthetic clothes or insulating chairs).

Discharge yourself by touching a conductive grounded surface before handling the camera.

3.7.2 Cleaning NECTA camera housing

- Keep lens mount hole sealed (using a lens or a proper cover) when cleaning camera housing.
- Use only a soft, dry cloth.
- In case of persistent spots, use a soft cloth with few drops of optical grade cleanser or neutral detergent, then dry quickly.
- Never use cleansers such as gasoline, spirits or solvents. Aggressive products may damage the camera housing surface.

3.7.3 Optical path cleaning

- Perform the cleaning procedure of any optical component in a dust-free clean environment.
- Avoid directly touching optical parts with bare fingers. Wear clean-room grade latex or nitrile powder-free gloves.
- Avoid touching any optical surface with rough materials.
- Use only optical grade wipes, not to leave dirt residues and to scratch the glass. Never use commercial grade cotton swabs.
- Use only low residue optical grade cleansers, this avoids haloes over the glass.

3.7.4 Impurities

Before cleaning any parts, it is crucial to understand where impurities are located. You can distinguish particles on the lensor on the sensor by looking at live images on your player. To help identifying any impurity in the optical path, it is recommended to set the smallest aperture by rotating the lens dial (larger values correspond to small apertures). Point the camera to a homogeneous bright target to highlight possible impurities artifacts. Impurities on the sensor appear as dark well-focused spots that remain still while you move the camera respect to the subject . Dust on the lens looks blurred and rotates together with the lens.

3.7.5 Sensor cleaning

Installing the camera according to the recommended procedures makes unlikely sensor contamination.

Warning

The sensor protection glass cannot be accessed anyhow without opening the camera housing. Doing it by yourself will void the camera warranty.

If you think your camera sensor needs to be cleaned, contact rma@alkeria.com to arrange the camera return procedure (refer to Section 10.1).

3.7.6 Using compressed air

Alkeria SRL does not recommend cleaning methods involving compressed air. This kind of procedure may lead to various damage.

- High pressure air flow may push dust into the camera instead of removing it. Optical components may be permanently damaged and the sensor surface may be contaminated.
- Non-optical grade compressing systems may release oil vapor or moisture into the airflow. This contamination may damage the sensor or the camera and it could be very difficult to remove even for a trained operator.

If compressed air is used (never exceed 4 bar) you should choose optic-approved air compressor with anti-static ionizer and adopt filters to remove moisture and oil from the airflow.

3.7.7 After cleaning procedure

Risk of electric shock

Before reconnecting the plugs, be sure that cleaning materials have evaporated.

If any liquid has penetrated the camera housing, contact Alkeria SRL before plugging any connector (see Section 10.1).

4

Interfacing To The World

NECTA features a robust circular I/O connector (Hirose p/n HR10-10R-12SA (73)) interfacing to external signals. The I/O connector provides two input lines, two output lines and a programmable I/O line; all lines support both differential and single-ended signaling according to the RS-422, RS-644, LV-TTL/LV-CMOS and 12 V-24 V standards (where available. Check sec 4.0.1 for further information).
The I/O connector is located on the back of the camera as indicated in the following figure:

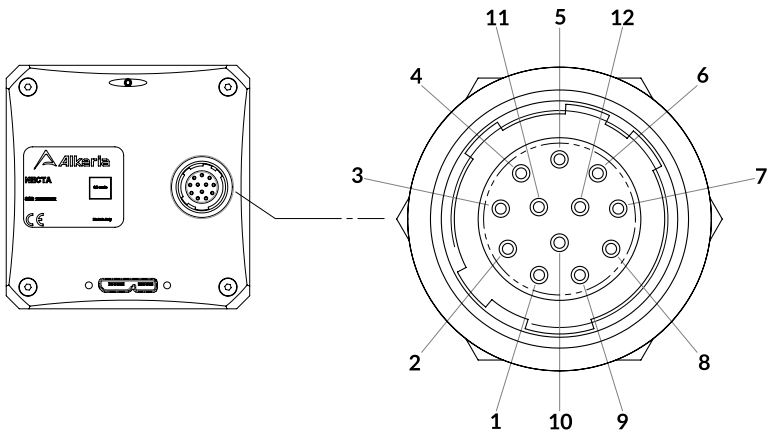


Figure 4.1: *NECTA I/O Connector*

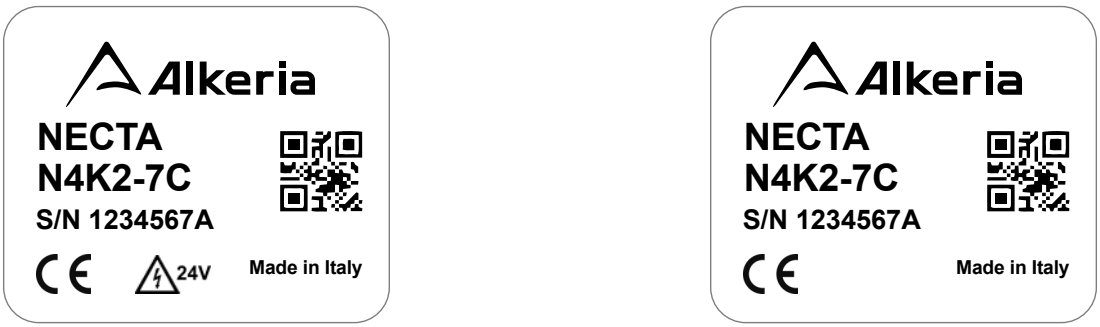
Pin	Name	Function
1	P0-	Bidirectional port, negative input
2	P0+	Bidirectional port, positive input
3	P1-	Input port, negative input
4	P1+	Input port, positive input
5	GND	Ground
6	DC Out	DC Output - 300 mA max
7	P3-	Output port, negative output
8	P3+	Output port, positive output
9	P2-	Input port, negative input
10	P2+	Input port, positive input
11	P4-	Output port, negative output
12	P4+	Output port, positive output

Table 4.1: *I/O Connector pinout*

The I/O ports are disabled at power-on, to disconnect any external load and user termination whose power consumption may exceed the USB 3.2 Gen 1x1 standard requirements during the bus recognition and enumeration phase. The I/O ports are automatically enabled at the end of the USB enumeration phase.

4.0.1 Detecting I/O power

NECTA features a 24 V tolerant I/O interface. However, previous NECTA hardware releases tolerates only 5 V signaling: for this reason, on each camera this feature is specified on the label, in order to quickly identify which model is compatible with 24 V signals.



(a) *NECTA with 24 V tolerant I/O*

(b) *NECTA with 5 V tolerant I/O*

Figure 4.2: *Differences between NECTA labels*

Caution

Always check your camera I/O power compatibility before interfacing the camera to other devices.
Connecting the camera to incompatible input signals may cause **permanent damage** to the camera itself.

4.1 I/O CABLE

The NECTA I/O connector can host a cable to interface the camera to the world. Standard interface cables can be provided as optional accessories (refer to Section 2.3); they are made of five signal pairs, a power pair and an additional global braid shielding. The cable shielding is connected to the NECTA aluminum case and to the protective earth of the controlling computer (through the USB cable). One end of the I/O cable features a Hirose connector (mod. HR10A-10P-12P) for connecting to NECTA , the other end is left open for connecting to external devices. The wire insulating material in the cable is either solid color or striped lengthwise in two colors, according to the DIN47100 standard. Refer to the table below for a detailed description:

Pin	Name	DIN47100 color	Pin	Name	DIN47100 color
1	P0-	White	7	P3-	Blue
2	P0+	Brown	8	P3+	Red
3	P1-	Green	9	P2-	Black
4	P1+	Yellow	10	P2+	Purple
5	GND	Grey	11	P4-	Grey/Pink
6	DC Out	Pink	12	P4+	Red/Blue

Table 4.2: *DIN47100 I/O cable color coding*

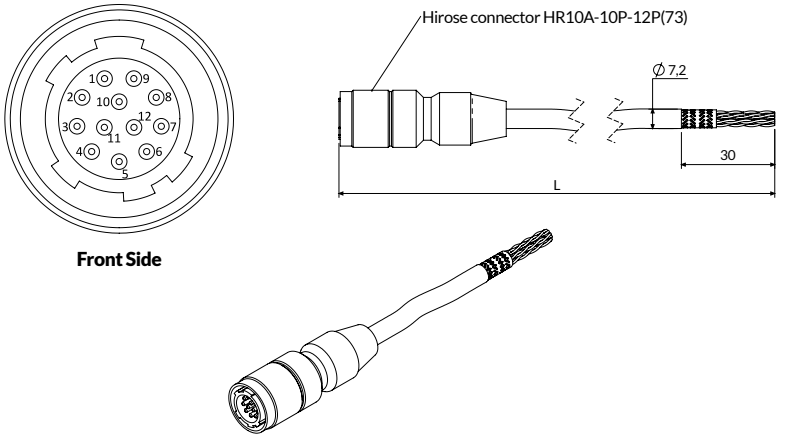


Figure 4.3: *NIO-L I/O cable internal structure*

If you need to build your own I/O cables, please use a good cable featuring a braid whose shielding coefficient is greater than 85 %, and connect the external braid shielding to the connector body. The maximum allowed length for the I/O cable is 10 m; it is recommended to use the shortest cable allowed by your application, see Section 2.3 for the available lengths.

Caution

Using cables not properly assembled may damage both the camera and connected devices.

4.2 POWER SELECTION

NECTA can supply to an external device up to 300 mA at 3.3 Vdc or 5 Vdc. User can select between those two power level using the following code example:

Example Code 4.1 | *IO Power Supply selection*

```
if(device.PIOHighVoltageOutAvailable)
    device.PIOHighVoltageOut = true;
```

To select 5 V power voltage, set the `PIOHighVoltageOut` property to True.

Caution

While booting, NECTA does not supply any power through the I/O. Once booted, the `PIOHighVoltageOut` is set to False (default value). Be sure that behavior will not damage the external hardware.

4.3 I/O MODULE

The signals available through the I/O connector natively support the RS-422 differential signaling and can also be used in RS-644, LV-TTL/LV-CMOS and 12 V - 24 V single ended signaling (where available). The NECTA family uses differential transceivers (p/n LTC2855-LTC2865, manufactured by Linear Technology); to access any data not shown below, refer to the manufacturer's datasheet.

4.3.1 Input module structure

Figure 4.4 - 4.5 shows the internal structure of a NECTA input module. An input internal termination can be enabled by software, replacing external end-of-line termination resistors; the input also features a voltage divider polarizing the "-" input to support single ended signaling (see Section 4.3.1.5).

Warning

To ensure proper operation of the input circuit, the reference ground related to input signals must be connected to the camera as well.

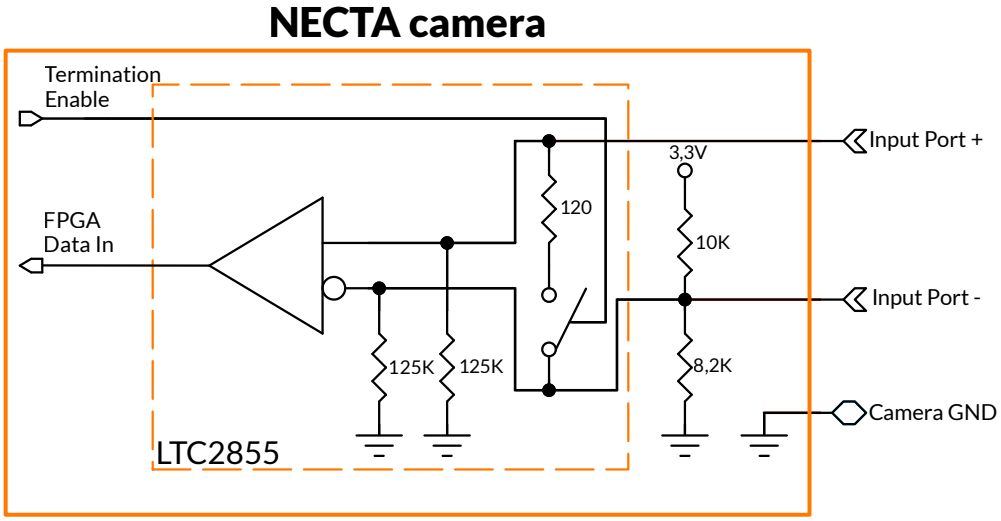


Figure 4.4: *NECTA with 5 V tolerant input port internal structure*

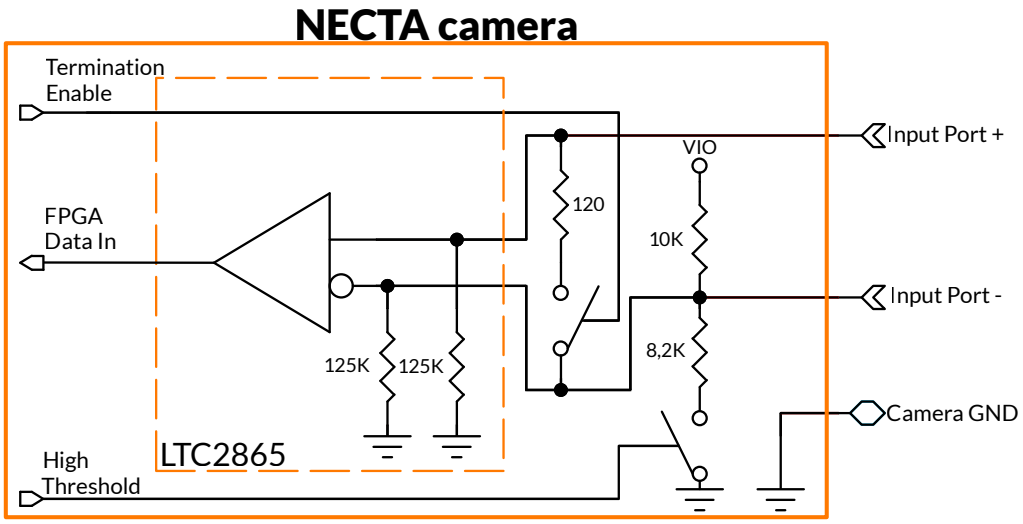


Figure 4.5: *NECTA 24 V tolerant input port internal structure*

4.3.1.1 Debouncing module

Each NECTA input line can feed a debouncing module to filter the input signal: enabling the debouncing module allows capturing stable signals with no spurious pulses when input signal suffers from electrical noise (e.g. signals generated by mechanical encoders or mechanical switches). The debouncing module waits for the input signal remaining stable for at least the user-programmed time interval (orange area in Figure 4.6); when this time is elapsed, the signal is considered valid and transferred to downstream

modules.

Warning

The debouncing filter adds a delay to the signal chain, starting from the instant when it has actually become stable, as long as the selected sampling time (see Figure 4.6). The debounce time can be set between 0 (debouncing disabled) and 65 535 µs, with a 1 µs granularity.

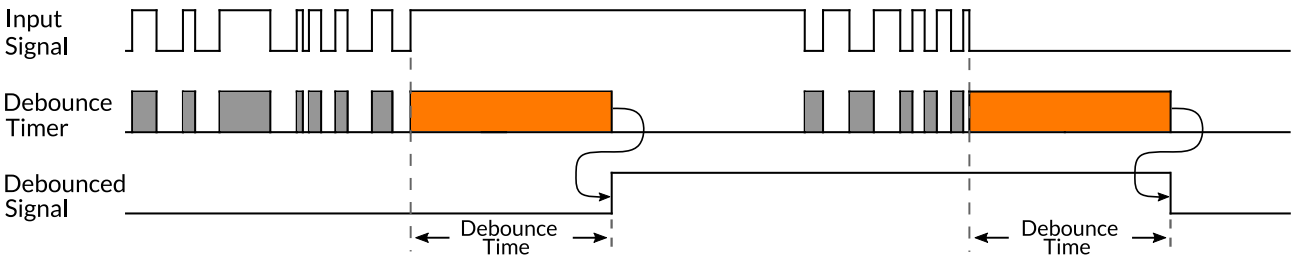


Figure 4.6: *Time diagram of the debouncing filter*

The debouncing module is independently available for all input lines and can be programmed with a different delay for each one of them.

Warning

When using the debouncing module, it is important to set a debounce time just longer than the maximum expected instability time of the input signal: setting shorter times might prevent filtering unwanted spurious transitions, while using longer times may cause missed detection of valid transitions.

4.3.1.2 Input events

NECTA input modules detect and log input events (raising and falling edges) occurring to any input port; the event logging allows easy detection of changes in input signals, avoiding continuous input polling. Rising and Falling events can be individually read; reading an event flag immediately resets its status. This allows you to check if any input status transition has occurred since the last event flag reading.

The following code source reads the event flags bound to input port 1:

Example Code 4.2 | *Input Change Event*

```
bool riseEvent = device.PIOPorts[1].RisingEvent;
bool fallEvent = device.PIOPorts[1].FallingEvent;
```

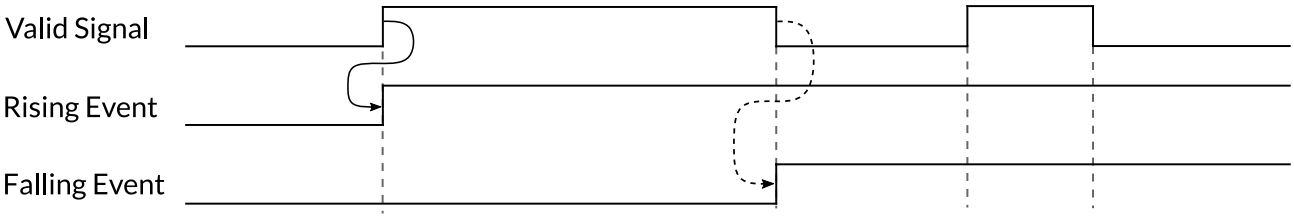


Figure 4.7: *Input event generation and readout*

Warning

Event flags are set at each input event; when consecutive input events of the same type (rising or falling) happen between two event flag readouts (event flag overrun), the input event latches the first event only and the following events will be lost.

4.3.1.3 RS-422

NECTA input ports natively support differential signaling according to the RS-422 standard.

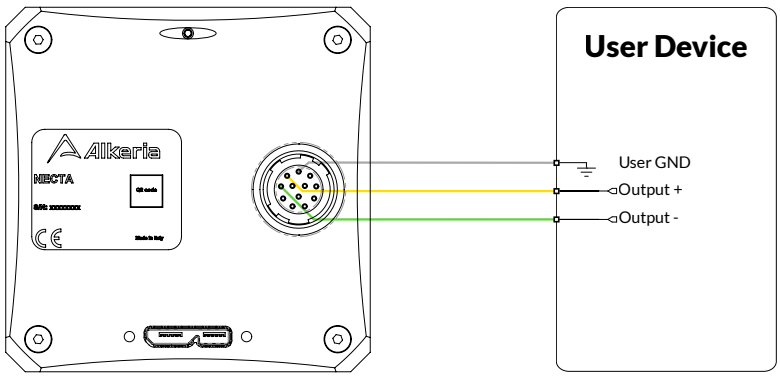


Figure 4.8: *Connecting a RS-422 input signal*

If your application requires connecting multiple devices to the same RS-422 line, please use a bus topology. For example, if you want to connect a trigger signal to multiple cameras, please implement the connections as shown in Figure 4.9:

Caution

AVOID other topologies (e.g. star) when connecting devices.

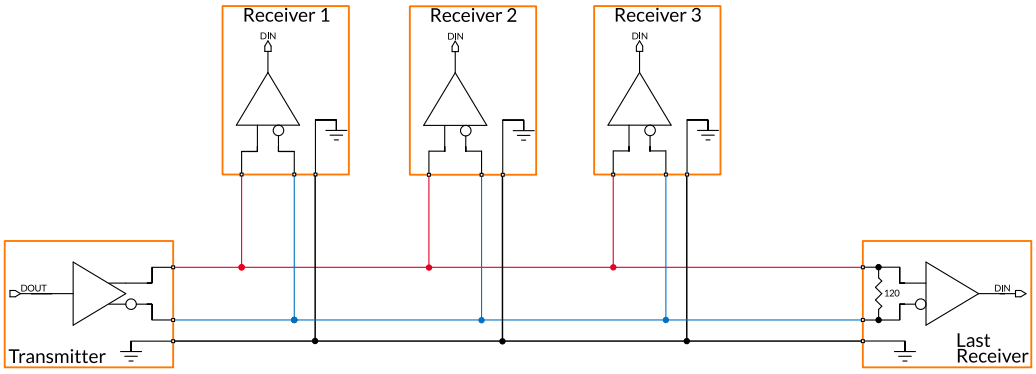


Figure 4.9: *Connecting an output port to four input ports, terminating the last input port*

Warning

To ensure the electrical signal integrity on the bus, you must connect only the termination resistor (120 Ω) to the last device (on the right in the figure above). If the last device is a NECTA camera, you can simply enable its internal termination via software (see Code 4.16).

As a general guideline, the master device (driving the bus) should be at one end of the line, the cables connecting each device to the bus should be as short as possible and the termination should be present (or enabled) only for the latest slave device.

4.3.1.4 RS-644

Input modules can also support RS-644 signals.

Warning

To achieve signaling compatibility, you must **always** keep the internal input termination enabled, even if this is not the last device of the bus.

Moreover, although the RS-644 standard (like RS-422) allows a bus connection, we recommend using point-to-point connections only avoiding connecting devices other than the RS-644 transmitter and NECTA .

4.3.1.5 LV-TTL / LV-CMOS / 12 V - 24 V

NECTA input ports natively support differential signals according to the RS-422 standard. Anyway, input modules can also support single-ended LV-CMOS, LV-TTL and 12 V - 24 V signals (where available): as shown in Figure 4.10, an internal bias network keeps the ”-” input line at a fixed voltage level, that can be configured according to the maximum voltage of the external signal. A single-ended signal connected

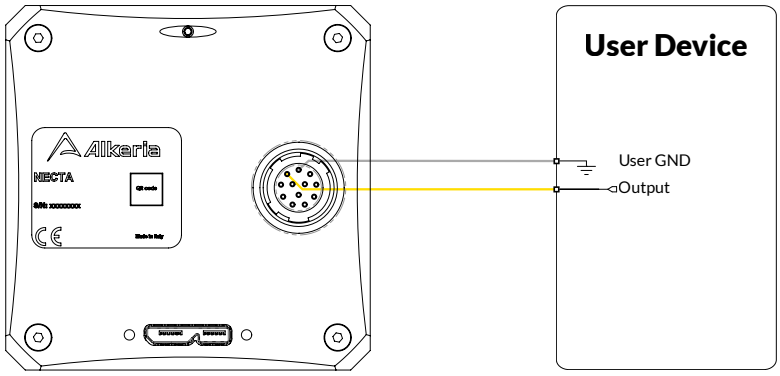


Figure 4.10: *Connecting a LV-TTL signal to input port P1+*

to the "+" is therefore detected as logical "1" when its level is at least 0.5 V higher than "-" reference signal. Following table shows how to configure the threshold level in NECTA with 24 V tolerant I/O, using the properties `PIOHighVoltageOut` and `PIOHighThreshold`.

Input maximum Voltage	`PIOHighVoltageOut`	`PIOHighThreshold`	**Reference Voltage**
3.3 V	False	False	1.49 V
5 V	True	False	2.25 V
5 V to 24 V	True	True	5 V

Table 4.3: *Single ended "-" input voltage value*

Note

Threshold level settings are common for all input ports.

Warning

When an input port is connected to a LV-CMOS/TTL signal its input termination must be kept disabled; otherwise, the input signal cannot be correctly detected.

Caution

Input signals **must never exceed +30 V** on NECTA with 24 V tolerant I/O.
Input signals **must never exceed +6 V** on NECTA with 5 V tolerant I/O.

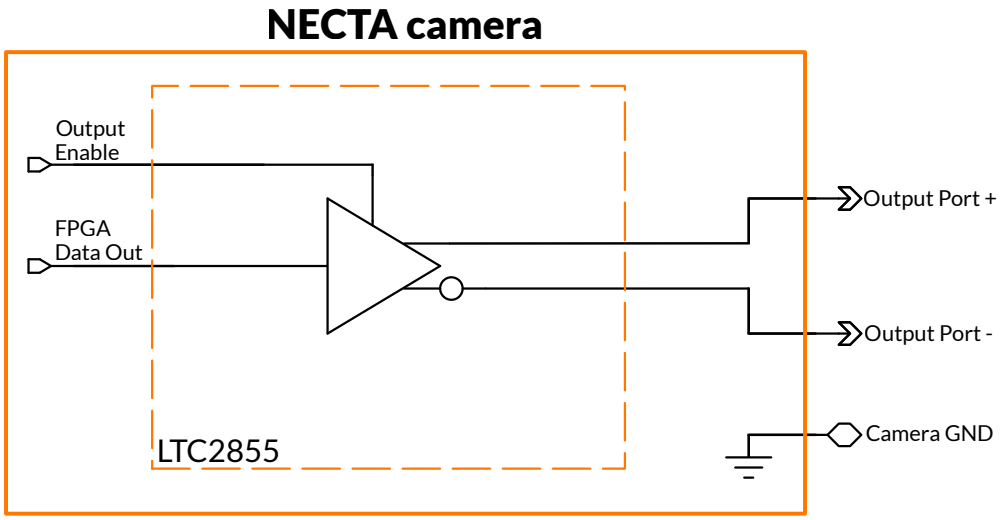


Figure 4.11: *NECTA with 5 V tolerant output port internal structure*

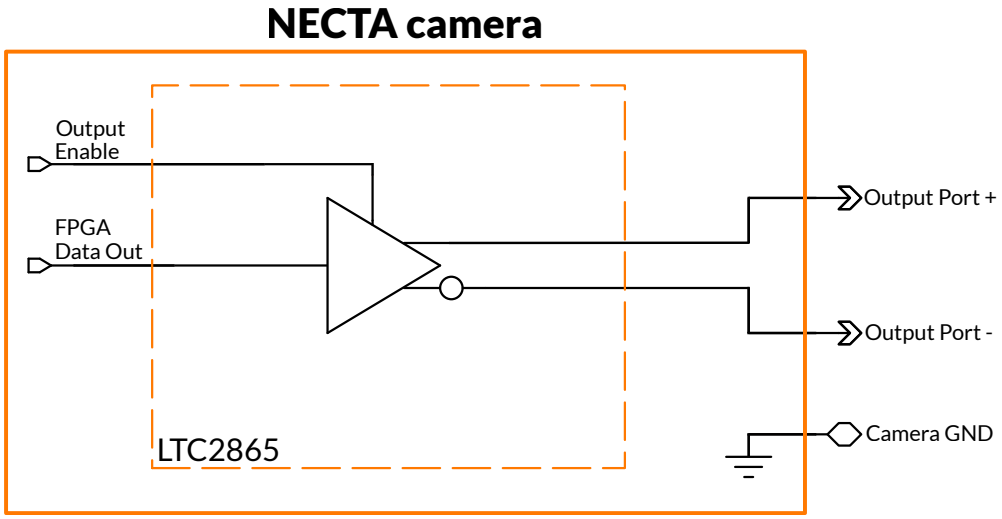


Figure 4.12: *NECTA with 24 V tolerant output port internal structure*

4.3.2 Output module structure

NECTA output ports natively support differential signaling according to the RS-422 standard and may be used even in RS-644 and LV-TTL mode. Figure 4.11 - 4.12 shows the internal structure of an output port. The NECTA family employs differential transceivers (p/n LTC2855-LTC2865, manufactured by Linear Technology); to access any data not shown below, refer to the manufacturer's datasheet.

Warning

To ensure proper operation of the output circuit, the camera reference ground must be connected to the slave device as well.

4.3.2.1 RS-422

NECTA output ports natively support differential signaling according to the RS-422 standard.

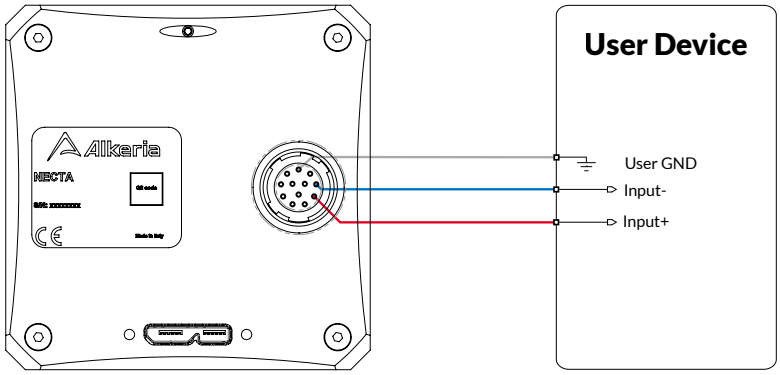


Figure 4.13: *Connecting output port P3 to a RS-422 input*

Warning

To ensure signal integrity on the bus it is highly recommended to use the RS-422 bus topology and keep all the bus connections to slave devices as short as possible. The master device (the NECTA camera) must be located at one end of the bus; a terminating resistor (120 Ω) must be connected only to the farthest slave at the other end of the bus (see Figure 4.9).

4.3.2.2 RS-644

NECTA outputs (supporting the RS-422 standard) may also be adapted to drive devices whose inputs follow the RS-644 standard. To make signaling compatible it is necessary to use a resistive divider, as shown in Figure 4.14, adapting the output signal level of the camera output to the RS-644 device input levels.

Warning

To ensure proper operation of the output circuit, the camera reference ground must be connected to the slave device as well.

Although the RS-644 standard allows bus topology (like RS-422), we strongly recommend using point to point connections only, avoiding connecting devices other than the NECTA camera and the RS-644 receiver.

Warning

Connecting NECTA camera outputs to an external RS-644 device inputs WITHOUT using the recommended resistive divider may seriously damage your device.

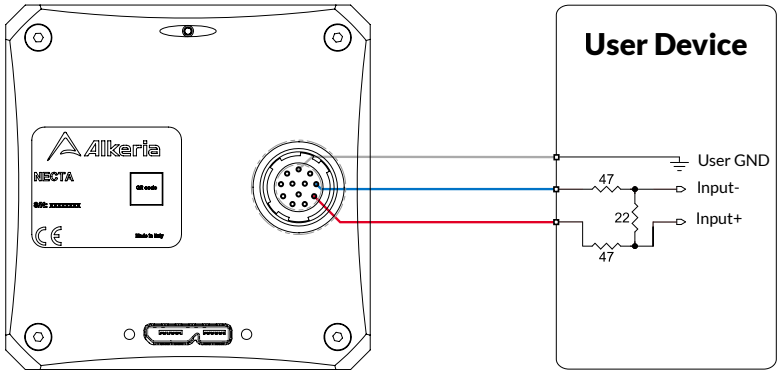


Figure 4.14: *Connecting output port P3 to a RS-644 input*

4.3.2.3 LV-TTL / LV-CMOS

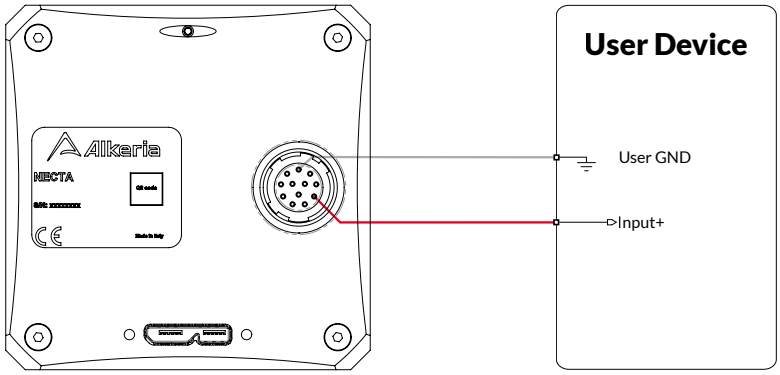


Figure 4.15: *Connecting output port P3+ to a LV-TTL / LV-CMOS input*

NECTA outputs can be interfaced also to LV-TTL/LV-CMOS inputs using the connections shown in Figure 4.15: the '0' logic level generated by the output driver is lower than 0.5 V, while the '1' level is about 3.3 V; both levels therefore satisfy the input requirements for both LV-TTL and LV-CMOS signaling. Please connect the P+ output for direct signaling, or P- output for inverted logic signaling.

Warning

To ensure proper operation of the output circuit, the camera reference ground must be connected to the slave device as well.

4.3.2.4 5 to 24 V Output

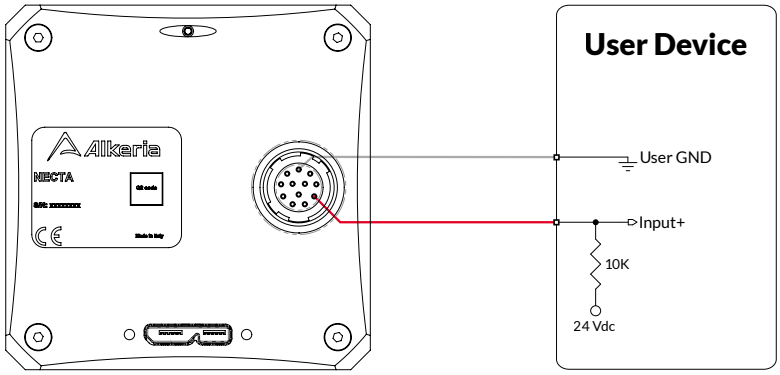


Figure 4.16: *Connecting output port P3+ to a 24 V input*

NECTA with 24 V tolerant I/O interface can drive also 5 to 24 V inputs using the connections shown in Figure 4.16: the '0' logic level is generate by the output driver "shorting" the external pullup resistor to ground, while the '1' level is imposed by the external pullup voltage. This configuration needs to be enabled using the procedure in Section 4.5.2.

External pullup resistor value must be calculate in order to maintain the 0 V output current under 30 mA, as shown below:

$$R_{Pullup}[\text{k}\Omega] \geq \frac{V_{Pullup}[\text{V}]}{30\,\text{mA}} \tag{4.1}$$

Note that the lowest pullup value, the slowest commutation time, the highest '0' logic voltage value.

4.3.3 Bidirectional module structure (port 0)

NECTA port 0 can be software-configured either as an input or as an output. When the camera is powered on, the port is initially configured as an input (termination disabled); it can be set as an output by software. All the statements above about input and output ports, including methods to interface to signals with logic levels different from RS-422, apply also to port 0. For more information on how to configure and use the input/output port refer to the code samples in Section 4.5 and to the examples found in the MaestroUSB3 SDK.

4.3.4 I/O switching time

As reported in the I/O schematics shown in Figure 4.4-4.5 and below, NECTA cameras use RS-422 differential transceivers mod. LTC2855-LTC2865, made by Linear Technology. They allow very short switching time: the input propagation delay, due to the input receiver, is less than 70 ns, the output delay is less than 50 ns[1]; both rise and fall time are less than 12.5 ns.

[1]Valid only for push-pull output configurations. When the output is open-collector, delay depends on the external pull-up.

Warning

Input signals are usually processed by the debouncing module (see Section 4.3.1.1), which avoids false acquisitions when the input signals may contain glitches. However, the debouncing module obviously limits the input signal bandwidth: when an input signal is changing very fast, properly setting the related debouncing module, or even disabling it, may be necessary to avoid losing events.

4.4 USING INPUT SIGNALS

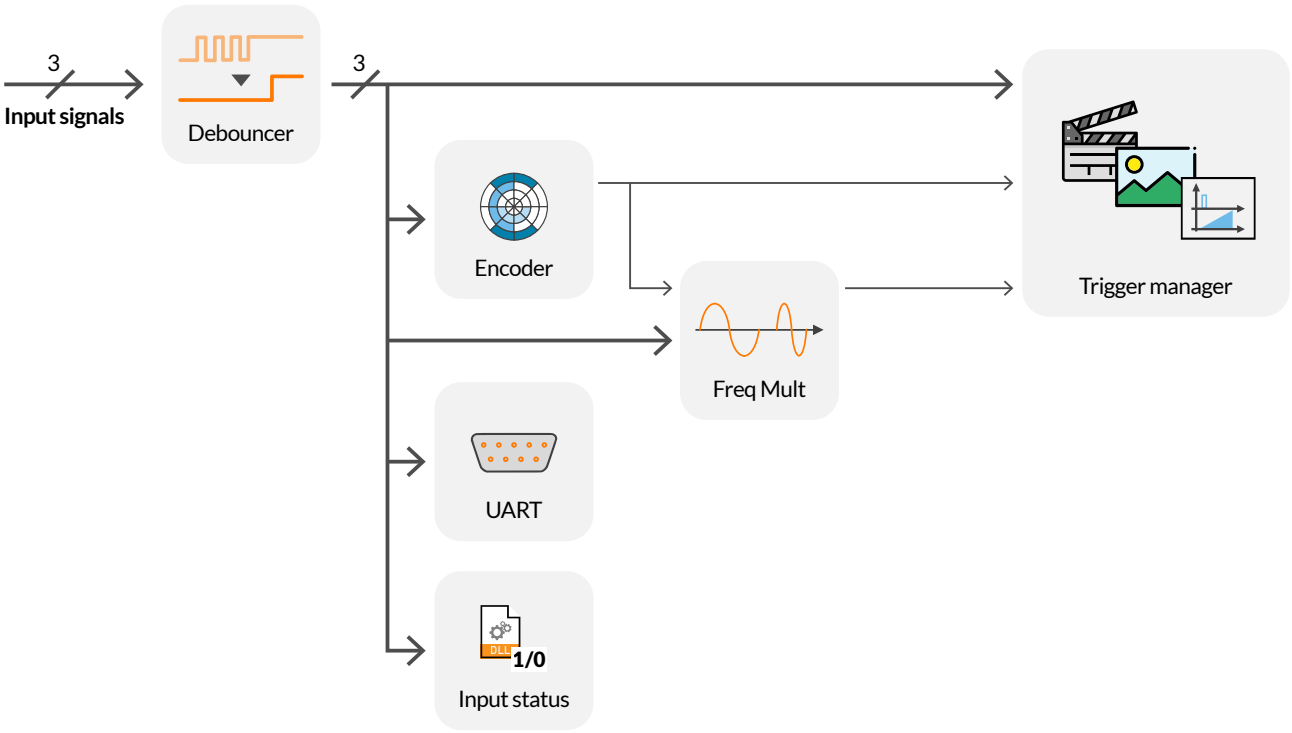


Figure 4.17: *Input port internal routing*

As shown in Figure 4.17, signals acquired through input ports can be routed, along with other internally generated signals, to the internal camera processing modules; some outputs of the processing modules can be routed to the trigger manager module, which controls camera timing and acquisition mode.

4.4.1 Encoder module

The encoder module allows interfacing the camera to an external encoder; it generates a pulse (tick) for each encoder transition and updates a resettable counter tracking the current position. Phase and quadrature signals (marked as A and B in Figure 4.18 and following) can be routed to any of the three camera inputs and are fed to the encoder module after being filtered by the debouncing module (see Section 4.3.1.1).

4.4.1.1 Encoder hysteresis

The encoder module uses a threshold mechanism to ignore temporary direction reversals due to vibrations and mechanical plays. The figures above show the operation of the encoder module, respectively in the cases of constant rotational speed (Figure 4.18) and input jitter due to mechanical vibrations (Figure 4.19).

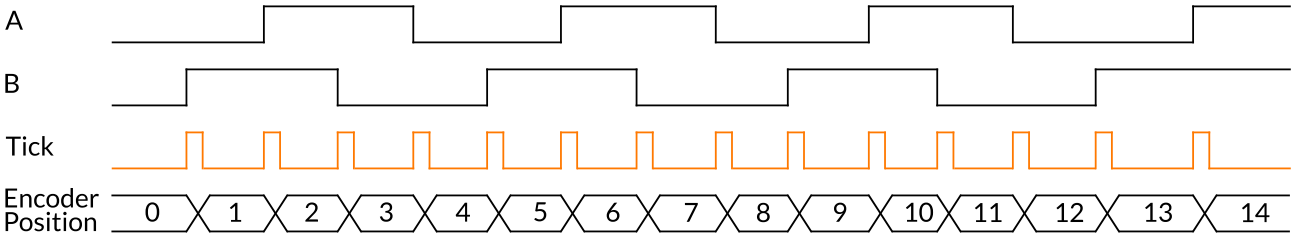


Figure 4.18: *Ticks generated in normal operations*

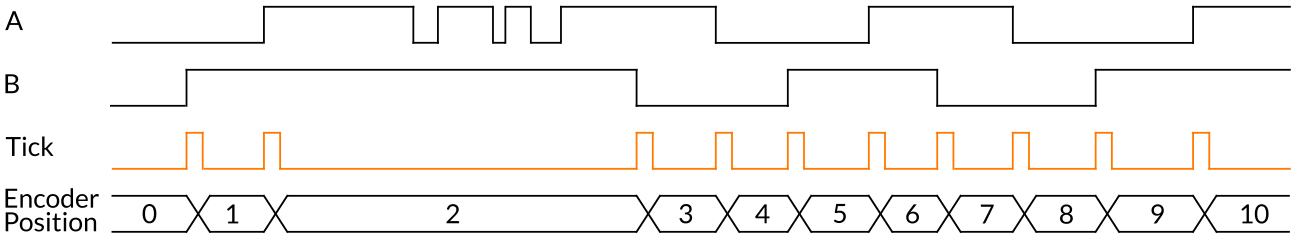


Figure 4.19: *Ticks generated in the presence of jitter*

4.4.1.2 Direction reversal

The IgnoreDirection property controls the behavior of the encoder module when an actual (permanent) direction reversal occurs: when it is set to true, the module continues generating ticks regardless of the direction of rotation (see Figure 4.20 and Figure 4.21).

As you can see, on T1 the backwards step is ignored due to the hysteresis effect. Therefore, no ticks are generated, even if the encoder continues to rotate backwards (T2) and position counter is decreased. When the encoder starts turning forward again (T3), the internal position counter will be increased again. However, ticks will be generated again only at T4, i.e. when the counter reaches the value it had just before the encoder direction inversion.

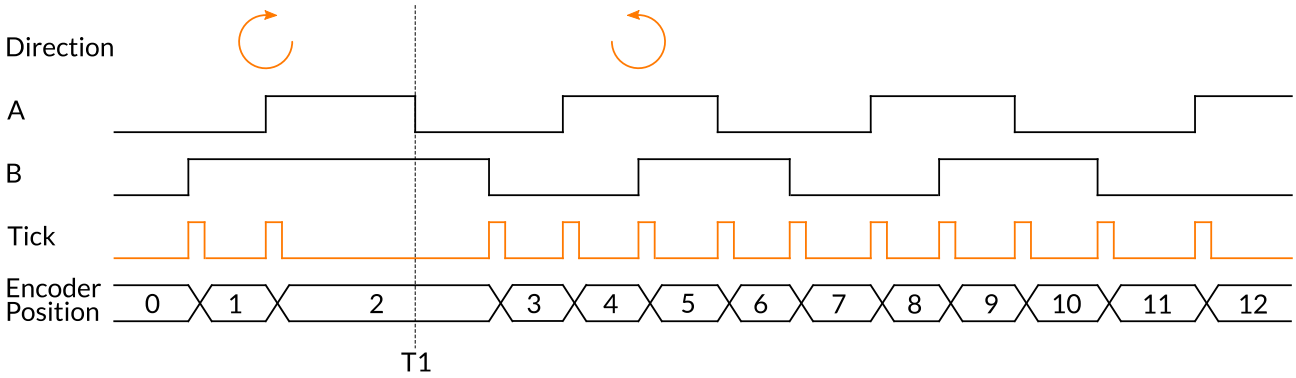


Figure 4.20: *Ticks generated when ignore direction is enabled*

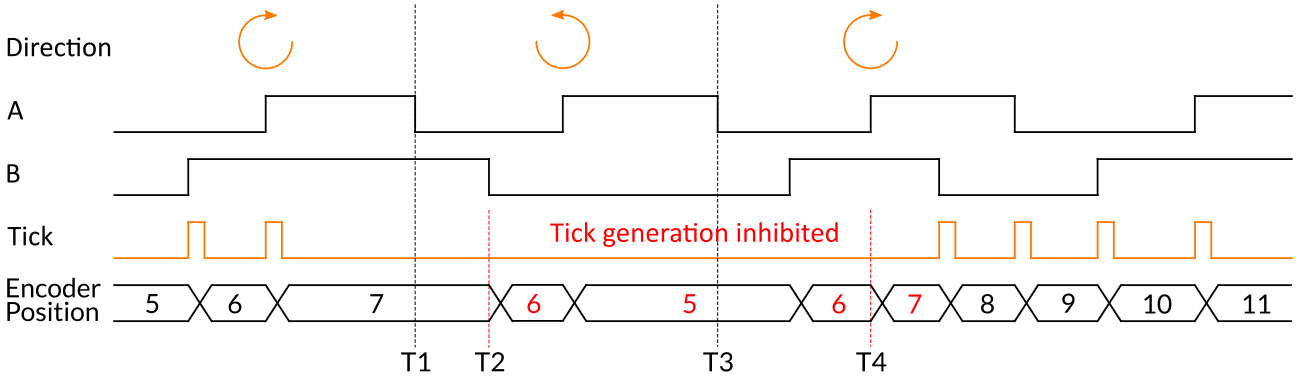


Figure 4.21: *Ticks generated when ignore direction is disabled*

4.4.1.3 Configuring the encoder module

The following code sets the input ports 1 and 2 as the phase (A) and quadrature (B) encoder inputs. The *encoder module* is programmed to generate a trigger only when rotating in the positive direction.

Example Code 4.3 | *Encoder ports configuration*

```
device.Encoder.InputA = 1;
device.Encoder.InputB = 2;
device.Encoder.IgnoreDirection = false;
```

4.4.1.4 Reading and resetting encoder position

The current encoder position can be read at any time as a 32-bit unsigned integer. The returned value is the last value of the *encoder module* position counter at the time of the request.

The *encoder module* position can be read and reset to 0 via software or with an external signal.

The following example shows how to reset encoder via software:

Example Code 4.4 | *Encoder position read*

```
uint position = device.Encoder.CurrentPosition; // Read counter current value
device.Encoder.Reset();                         // Reset value to 0
position = device.Encoder.CurrentPosition;      // Now position is 0
```

Note

When the encoder position is reset, the targets of all trigger modules are automatically reset too and restart counting from 0 (see Chapter 6).

The code above configures rising edge on port 2 as encoder reset source:

Example Code 4.5 | *Encoder reset with external source*

```
if (device.Encoder.ResetAvailable)
{
    EncoderResetSource[] resetSources = device.Encoder.GetAvailableResetSources();
    if (Array.Exists(resetSources, t => t == EncoderResetSource.External))
    {
  // Configure Input port 2 as Reset Input on the rising edge.
        device.Encoder.ResetExternalInput = 2;
  // Configure Reset as edge-sensitive.
        device.Encoder.DetectResetExternalInputEdge = true;
  // Reset will be performed on rising edge of input.
        device.Encoder.InvertResetExternalInput = false;
  // Enable external encoder reset.
        device.Encoder.ResetSource = EncoderResetSource.External;
    }
}
```

4.4.2 Frequency Multiplier (Phase Locked Loop (PLL))

When you want to capture events faster or slower than the related trigger signal frequency or longer/shorter than the encoder step interval, you can use the NECTA camera internal frequency multiplier: signals coming from both I/O connector and encoder module can be processed to change the acquisition frequency. The multiplier detects the rising edges of the signal selected as a source and generates an output pulse train at the resulting frequency. The output frequency f_o is generated from the input frequency f_i according to the following formula:

$$f_o = f_i \cdot \frac{M}{D} \tag{4.2}$$

where M and D are the Multiplier and Divisor factor, respectively. The allowed frequency range for the input signal goes from 400 Hz to 4 MHz.

The maximum allowed input jitter, expressed as the maximum absolute variation of the input period, is:

$$\frac{1}{f_i \cdot M} \tag{4.3}$$

while the maximum absolute jitter of the trigger output is not greater than 21 ns.

> **Note**
>
> Jitter in frequency input signals or encoder rotational affects the output trigger, resulting in erratical acquisition cadence. Minimize input jitter in your application for optimum performance.

The following code selects the source for the *frequency multiplier* and the multiplying factor:

Example Code 4.6 | *PLL configuration*

```
device.PLL.Source = PLLSource.External;            // Select I/O port 0 as PLL source
device.PLL.ExternalInput = 0;
device.PLL.Multiplier = 10;                        // Set multiplier factor to 10 / 3
device.PLL.Divisor = 3;
device.FrameStartTrigger.Source = TriggerSource.PLL;  // Set PLL as trigger source
device.FrameStartTrigger.Enable = true;            // Enable Frame Start Trigger
```

The maximum and minimum allowed values for M and D can be retrieved by reading the following properties:

Example Code 4.7 | *PLL boundaries readout*

```
uint maxM = device.PLL.MaxMultiplier;
uint minM = device.PLL.MinMultiplier;
uint maxD = device.PLL.MaxDivisor;
uint minD = device.PLL.MinDivisor;
```

4.4.3 Trigger manager

The signal incoming from an input port can be also simply routed to the *trigger manager*. Every signal can one the events listed below:

- Acquisition-start;
- Frame-start;
- Line-start;
- End-of-exposure.

For further information refer to Chapter 6.

4.5 CONTROLLING OUTPUT PORTS

NECTA outputs can be directly controlled by software or reflect the state of internally generated signals; the latter allows to synchronize lighting systems and/or external capture devices with NECTA acquisition.

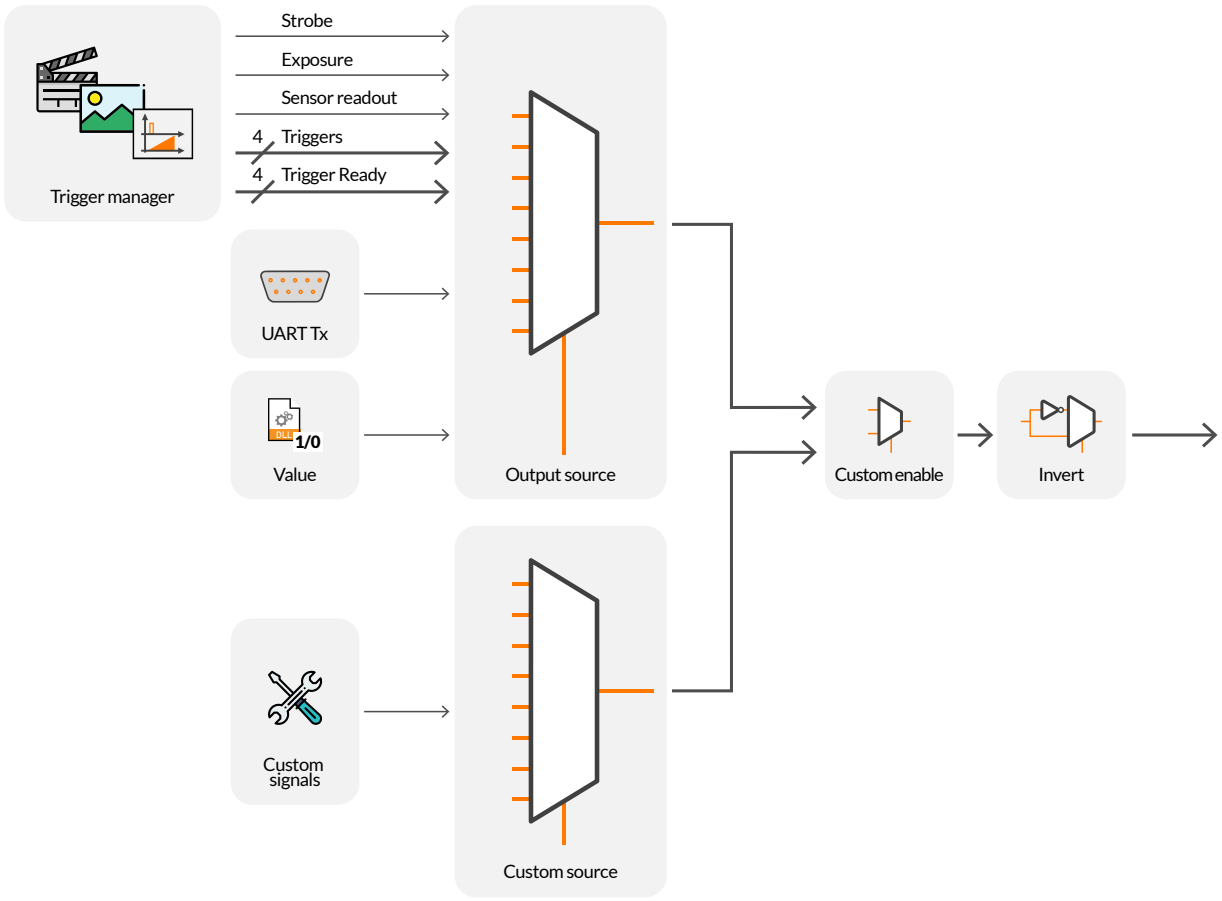


Figure 4.22: *Output port internal routing*

4.5.1 Polarity

The polarity of each output port can be reversed through the `Invert` property. This property works regardless of the output source. The following example reverses the polarity for output port 3:

Example Code 4.8 | *Output inversion*

```
device.PIOPorts[3].Invert = true;
```

Note

The `Invert` property affects output signals only. It has no effect on input signal decoding.

4.5.2 Open collector

The high voltage output mode can be directly set by your application. The following code sets port 4 as open-collector:

Example Code 4.9 | *Drive 24 V input signals*

```
if (device.PIOPorts[4].OpenCollectorModeAvailable)
    device.PIOPorts[4].OpenCollectorMode = true;
```

4.5.3 Software output

The logic level of any output port can be directly set by your application. The following code sets port 3 as a software-controlled output and sets the output level as logic '1':

Example Code 4.10 | *Output controlled via software*

```
device.PIOPorts[3].Invert = false;                  // Logic '1' == electric High
device.PIOPorts[3].Value = true;                    // Set port 3 output high
device.PIOPorts[3].Source = OutputSource.Manual;    // Port 3 output now manually driven
```

Note

The electrical level generated at the output port is related to the state of the `Invert` and the `OpenCollectorMode` properties (see Sections 4.5.1 and 4.5.2).

4.5.4 Trigger Output

Output signals generated by the *trigger manager* module can be connected to output ports. This allows synchronizing ancillary devices to the camera acquisition phases: for example, a lighting device can be synchronized to the camera exposure using the *Strobe* signal output. The next paragraphs illustrate how these signals can be routed to output ports and how to configure them.

4.5.4.1 Exposure

The *exposure* signal is active during the exposure phase of each line capture (the time the shutter is open). The following code sets the *exposure* signal as the source for output port 3.

Example Code 4.11 | *Exposure as P3 source*

```
device.PIOPorts[3].Source = OutputSource.Exposure;
```

The *exposure* signal can be used to synchronize multiple NECTA devices, so that the acquisition in progress signal of the master camera can be routed as a capture trigger for slave cameras. This method is simple and effective, but it it still affected by an intrinsic delay due to I/O switching time, debouncing filters etc; if your application requires precise synchronizazion of multiple cameras, or when external non-camera devices (e.g. lighting systems) must be perfectly synchronized to the acquisition, consider using the *strobe* signal instead and the related delay mechanism, as described in 4.5.4.2.

4.5.4.2 Strobe

The strobe pulse can be generated by setting two programmable delays from the beginning of exposure, for rise and falling time. These two delays can be properly set with 1 µs resolution, according to your needs.

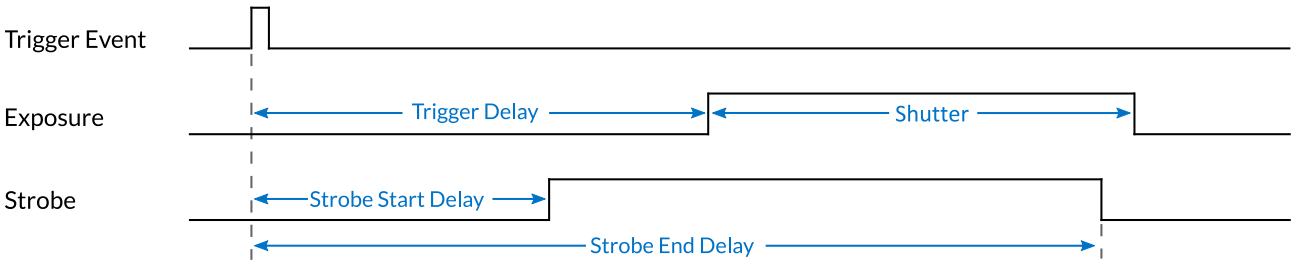


Figure 4.23: *Strobe pulse*

The following code configures the strobe signal and routes it to output port 3:

Example Code 4.12 | *Strobe as P3 source*

```
UnitInfo unit = device.Strobe.DelayUnitInfo;        // Retrieve Delay unit info
double startDelay = 1e-6;                           // Set strobe start delay to 1 us
double endDelay = 10e-6;                            // Set strobe end delay to 10 us
device.Strobe.StartDelay = (uint)(startDelay / unit.Factor);
device.Strobe.EndDelay = (uint)(endDelay / unit.Factor);

device.PIOPorts[3].Source = OutputSource.Strobe;
```

Note

The *strobe* signal is not a generalization of the exposure signal described in 4.5.4.1, but actually a different signal with its own use: both signals originate from the *line start* event trigger, so they can have different delays with respect to it. E.g., starting from a trigger event (e.g. crossing of an encoder target position), NECTA can start an exposure with a given delay (using the *trigger manager*) and generate the strobe signal with a shorter delay to pre-trigger the lighting system.

4.5.4.3 Trigger

Trigger signals generated from input ports and/or internal processing modules (encoder and frequency multiplier) can be routed to output ports. Each trigger signal rises on the corresponding event trigger signal (acquisition, frame, line, end-of-exposure – see Chapter 6) and remains active for about 5 µs. Like the *exposure* signal, trigger outputs allow synchronizing external devices to the camera trigger events.

Example Code 4.13 | *Frame start trigger as P2 source*

```
device.PIOPorts[2].Source = OutputSource.FrameStartTrigger;
```

4.5.4.4 Trigger ready

The *trigger manager* module also generates a ready signal for each trigger module (acquisition, frame, line, end-of-exposure – see Chapter 6), indicating that corresponding trigger module are ready to accept a trigger signal. *Trigger ready* signals are normally high (active); they fall on the corresponding trigger event and rise again when a new trigger can be accepted. When a *trigger ready* signal is low, all the related trigger events are ignored.

Example Code 4.14 | *Line start trigger ready as P2 source*

```
device.PIOPorts[2].Source = OutputSource.LineStartTriggerReady;
```

4.5.5 Custom sources

Additional application-specific output sources can be available on some NECTA models with customized firmware.

Example Code 4.15 | *Custom signal 10 as P0 source*

```
device.PIOPorts[0].CustomOutputSource = 10;
device.PIOPorts[0].CustomOutputSourceEnabled = true;
```

4.5.6 Input and output ports usage examples

The following code configures NECTA port 0 (bidirectional) as an input, sets 10 µs as debounce time and enables the internal differential termination:

Example Code 4.16 | *P0 configuration as input*

```
device.PIOPorts[0].Direction = IODirection.Input;   // Configure port 0 direction
device.PIOPorts[0].DebounceTime = 10;                   // Set debounce time to 10us
device.PIOPorts[0].Termination = true;                  // Enable termination
bool status = device.PIOPorts[0].Value;                 // Read port 0 value
bool rising = device.PIOPorts[0].RisingEvent;           // Read port 0 rising event
```

The following code configures port 0 (bidirectional) as an output and sets level as high:

Example Code 4.17 | *P0 configuration as output*

```
device.PIOPorts[0].Direction = IODirection.Output;  // Configure port 0 direction
device.PIOPorts[0].Value = true;                    // Set output high
```

4.6 SERIAL INTERFACE MODULE

NECTA cameras are equipped with a serial interface module to communicate with external devices. This module is able to send and receive data through dedicated FIFO buffers. Incoming data are stored in an input FIFO and can be retrieved via simple read commands issued to the camera. Output data sent through write commands are temporarily stored in an output buffer and then sent through the serial interface.

Each FIFO is 64-byte deep. If the receiving FIFO gets full before the received data are retrieved by your application, an error bit is set. This condition can be checked via the Overrun property.

MaestroUSB3 provides methods to set the requested baud rates, check the input buffer level and size and assign I/O pins to serial Tx and Rx. Hardware handshake is not supported.

Note

If your application needs serial input or serial output only, you can enable just the required pin (Tx or Rx), avoiding wasting an I/O pin for the unneeded function (see below).

Example Code 4.18 | *Serial interface output configuration*

```
device.PIOPorts[0].Source = OutputSource.UARTTx;    // Uart Tx on Port 0
device.PIOPorts[0].Direction = IODirection.Output;  // Port 0 output direction
device.UART.BaudRate = 38400;                       // 38400 bps
byte[] b = new byte[16];                            // Data to send
device.UART.Send(b);                                // Send byte to Tx pin
```

In case you want to send a message whose size in bytes is greater than the output FIFO size, you should mind that the Send method is blocking, i.e. it waits until the last byte of the message has found its place in the output FIFO. To be sure that the Send method will not pause your application flow, you should always send a number of bytes lower than UART . TxFIFOSize and wait until the output FIFO is empty before calling the Send method.

Example Code 4.19 | *Big buffer send over serial interface*

```
int bytesToSend = bigBuffer.Length;             // Remaining bytes to send
int srcIndex = 0;                               // Index to the next byte to send
uint txFifoSize = device.UART.TxFIFOSize;       // This is the size of output FIFO
while (bytesToSend > 0)                         // While we have bytes to send
{
    // Allocate a temp buffer.
    byte[] buffer = new byte[Math.Min(bytesToSend, txFifoSize)];
    // Copy the data to send from the big buffer.
    Array.Copy(bigBuffer, srcIndex, buffer, 0, buffer.Length);
    // Wait for the output fifo to be empty.
    while (!device.UART.TxFIFOEmpty)
        Thread.Sleep(500); // Leave CPU time to other user threads
    // Send the buffer
    device.UART.Send(buffer);
    // Update indexes and counters
    srcIndex += buffer.Length;
    bytesToSend -= buffer.Length;
}
```

Example Code 4.20 | *Serial interface input configuration*

```
device.UART.Reset();                                // Input FIFO Reset
device.UART.InputPort = 1;                          // Uart RX on Port 1

uint rxFIFOLevel = device.UART.RxFIFOLevel;         // How many bytes have been received
if (rxFIFOLevel > 0)
{
    // Check if there has been an overrun error
    if (device.UART.Overrun)
       Debug.WriteLine("Input FIFO overrun detected");
    // Extract the received bytes
    byte[] data = device.UART.Read(rxFIFOLevel);
}
```

5 System Setup

5.1 USB SETUP

NECTA is controlled and powered through the USB connector. When the camera is connected to the PC through one USB 3.2 Gen 1x1 hub, you must be sure that the hub directly connected to the camera uses a suitable power supply: using poor quality power supplies may degrade the performance of the camera or damage it.

For maximum performance, the controlling computer must be equipped with appropriate USB 3.2 Gen 1x1 interface (see section 5.1.2).

NECTA data throughput can reach 3.2 Gbps; it is therefore recommended to reserve a USB 3.2 Gen 1x1 host controller for the camera.

If any other device must coexist with NECTA , it must not transfer data while camera is operating. NECTA can also operate with USB interfaces prior to USB 3.2 Gen 1x1 . Depending on the setup, some functionality may not be available or suffer significant performance degradation.

Keep USB cable far from potential EMC interference sources (e.g. power cords and cables carrying strong impulsive currents) during installation.

Refer to Section 3.6.7 for further information.

5.1.1 How to choose a suitable USB cable

Cables must match the USB 3.2 Gen 1x1 standard in order to properly work with NECTA camera. Refer to “Universal Serial Bus 3.2 Specification” available at `www.usb.org`.

One of the USB greatest advantages is that it provides a 5 V power supply. Connected USB devices are usually powered through the USB cable, eliminating the need for an external power supply.
Since USB is also designed for quick and easy connections/disconnections, standard connectors do not

keep cables firmly enough to withstand the heavy vibration encountered in industrial applications.

With time, standard connectors can work themselves loose and cause malfunctioning. Loose USB cables can lead to data loss, software hang-ups and blue screens. In the wrong environment they can even lead to fire or explosion. To avoid these problems, the NECTA USB 3.2 Gen 1x1 connector can accommodate cables with screw-lock connectors. Using these cables is highly recommended especially when movements and vibrations may affect the stability of the connection between the camera and the PC.

If the cable is subjected to repeated stress, it is also recommended to adopt flexible chain-specific cables. The maximum recommended length for the USB cable is 3 m. Longer distance (up to 8 m) can be covered by selected high-quality cables or using USB 3.2 Gen 1x1 hubs or active extension cords, commercially available.

It is important to evaluate the electrical resistance of the cable power section. The power and ground wires shall comply with the aforementioned electrical requirements. Thin cables usually cause large voltage drop, limiting the required power being provided to the device. Furthermore, a good shielded cable will improve EMI/RFI performance.

5.1.2 Recommended USB interface

To get maximum performances out of Alkeria cameras, it is essential to choose the right USB3 host controller, focusing on it's maximum throughput, since the controller is responsible for the actual available bandwidth.

Not all the USB3 host controllers are capable to manage maximum data transfer rate. While choosing your USB3 host controller, please pay attention to brand, model and number of controllers available on it. For example, a 2 ports USB3 host board with single 5 Gb/s controller will NOT be suitable for running two cameras at once at maximum speed; a 4 ports USB3 host board with dual 10 Gb/s controller will be capable of running up to 4 single-USB3 cameras at maximum speed.

Recommended adapters should feature one of the following host controllers:

- Texas Instrument TUSB7340/TUSB7320, isochronous mode only (5 Gbit/s max);
- Fresco Logic FL1100EX (5 Gbit/s max);
- Integrated Intel USB 3.2 Gen 1x1 Host (10 Gbit/s max on some models).
- Asmedia - ASM3142 (10 Gbit/s max).

Only Texas Instruments controllers currently achieve maximum throughput for isochronous endpoint (48 KiB per micro-frame, see Figure 2.4) and thus maximum frame rate. Fresco Logic FL100EX and Intel integrated controllers achieve maximum throughput of 375 MB/s for Bulk endpoints. Other controllers usually achieve 42 KiB to 45 KiB per micro-frame only, involving a 10 % loss in maximum speed.

Note

Trying to reach a bandwidth higher than the maximum supported by the controller usually makes the camera stop communicating correctly and may freeze the controller. When this happens, unplug and re-plug the camera to re-establish a correct communication.

Furthermore, we recommend to disable any technology that may slow down the CPU frequency and responsivity (SpeedStep and all Power Management States/C-States). These settings can be usually accessed through the BIOS/UEFI configuration utility.

Low-cost USB 3.2 Gen 1x1 to PCI Express multiport adapters, as well as USB 3.2 Gen 1x1 ports built-in in some PCs, use a single controller and/or internal USB 3.2 Gen 1x1 hubs. Devices connected to adjacent ports therefore often share the same USB 3.2 Gen 1x1 controller and may interfere with proper NECTA operation.

5.2 STATUS LED

NECTA features a status LED on its back, showing the operating conditions of the camera. When a USB connector is plugged in, LED remains red until the computer detects and configures the camera. At that point the LED turns green. When the camera starts acquiring, LED turns orange.

Table 5.1 details the meaning of the status LED indications.

Status LED	Status
OFF	Device not powered
RED	Camera powered, waiting for driver enumeration
GREEN	USB plugged and enumerated
ORANGE	Camera acquiring
BLINKING	Generic error (contact support@alkeria.com)
-	Generic error (contact support@alkeria.com)

Table 5.1: *Status LED behaviour*

Different behaviours of the status LED may indicate that camera is malfunctioning.

5.3 WHEN ONE NECTA IS NOT ENOUGH...

You can even connect multiple NECTA cameras at the same PC and use them simultaneously: Application Programming Interface (API) makes it possible to balance each video data stream.

When designing the application, you can choose how the connected NECTA cameras will share the available USB 3.2 Gen 1x1 bandwidth, freely modulating acquisition rate, pixel format and ROI of each NECTA based on the specific application needs. Refer to Section 8.9.6 for further details.

6 Trigger

NECTA captures a predetermined number of lines specified by the `ImageSizeY` parameter of the ROI (see Section 8.3) under the control of configurable events called "triggers". The lines acquired are grouped together in a frame and sent to the PC via the USB 3.2 Gen 1x1 bus.

The modules generating trigger events are examined below. They are:

- Acquisition-start trigger;
- Frame-start trigger;
- Line-start trigger;
- End-of-exposure trigger.

6.1 TRIGGER SOURCES

Each trigger is supported by a special module, which can be enabled and processes one or more of the trigger sources available within the camera. Trigger sources available for NECTA are listed in Table 6.1

The `GetAvailableSources` method, when applied to a trigger module, returns the trigger sources that can be selected for that module.

Example Code 6.1 | *Get all selectable trigger sources*

```
TriggerSource[] sources = trigger.GetAvailableSources();
if (Array.Exists(sources, t => t == TriggerSource.Encoder))
    trigger.Source = TriggerSource.Encoder;
```

External	Trigger is generated on the detection of an input event. The trigger module can be activated either by the rising edge or by the high level; the input logic can also be reversed in order to detect the falling edge or the low level.
Software	Trigger is generated through the SoftwareTrigger software method.
PLL	Trigger is generated by the tick of the frequency multiplier module (see Section 4.4.2).
Encoder	Trigger is generated whenever the encoder position counter increases by a programmable number of steps (set through the `EncoderInterval` property).
EncoderOnce	Trigger is generated only the first time the encoder position counter reaches the value of `EncoderInterval` property.

Table 6.1: *Trigger sources*

6.2 ACQUISITION START TRIGGER

The *acquisition-start* trigger (`AcquisitionStartTrigger`) event enables the acquisition of a given number of frames, called a *burst*. When the module is enabled[1], the camera waits for a start acquisition trigger before starting capturing a burst.

The number of frames in a burst is set by the `AcquisitionBurstLength` property; when the burst is complete, the camera waits for a new start acquisition trigger, that will trigger the acquisition of the next burst.

The sources available for this module are External, Software, Encoder and EncoderOnce. When this module is disabled[2], the acquisition is always active.

6.3 FRAME START TRIGGER

The *frame-start* trigger (`FrameStartTrigger`) event enables the acquisition of a frame, i.e. the exposure of a given number of lines. When the module is enabled[3], the camera waits for a frame start trigger before capturing lines.

The number of lines acquired after the frame start trigger is set by the `ImageSizeY` property, as described in Section 8.3; when the required the number of lines has been acquired, the camera waits for a new frame start trigger, that triggers the acquisition of the next frame.

The sources available for this module are *External*, *Software*, *PLL* and *Encoder*. When the module is disabled[4], the frame starts on the first line's trigger.

6.4 LINE START TRIGGER

The *line-start* trigger (`LineStartTrigger`) is the event that triggers the exposure of a line. The sources available for this module are *External*, *Software*, *PLL* and *Encoder*.

[1] `AcquisitionStartTrigger.Enabled = true`
[2] `AcquisitionStartTrigger.Enabled = false`
[3] `FrameStartTrigger.Enabled = true`
[4] `FrameStartTrigger.Enabled = false`

When the module is disabled[5], lines are continuously acquired and the time between lines is controlled by the LinePeriod property.

6.5 END-OF-EXPOSURE TRIGGER

The *end-of-exposure* trigger (`ExposureEndTrigger`) event terminates the exposure of a line. The only available trigger source for this module is *External*. When the module is disabled[6], the duration of the exposure is controlled by the Shutter control value (see Section 8.10.8).

6.6 TRIGGER DELAY

Each module can delay the signaling of the trigger event for a period of time, set through the module's `TriggerDelay` property.

When an external signal is used as a trigger source, the delay set through `TriggerDelay` combines with the delay due to the debounce time set on the related input port (see Section 4.3.1.1).

> **Note**
>
> When encoder is not used, it is mandatory to set `EncoderDelay` to 0. If not, no trigger event will be raised.

For *line-start* trigger, user can also set `TriggerDelay` when in free run. If controlling an external lightning system with Strobe, `TriggerDelay` can be used to compensate delay introduced by controlling-chain and time for light to reach steady-state. Following the example of Figure 4.23 a representation of the statement above is depicted in Figure 6.1

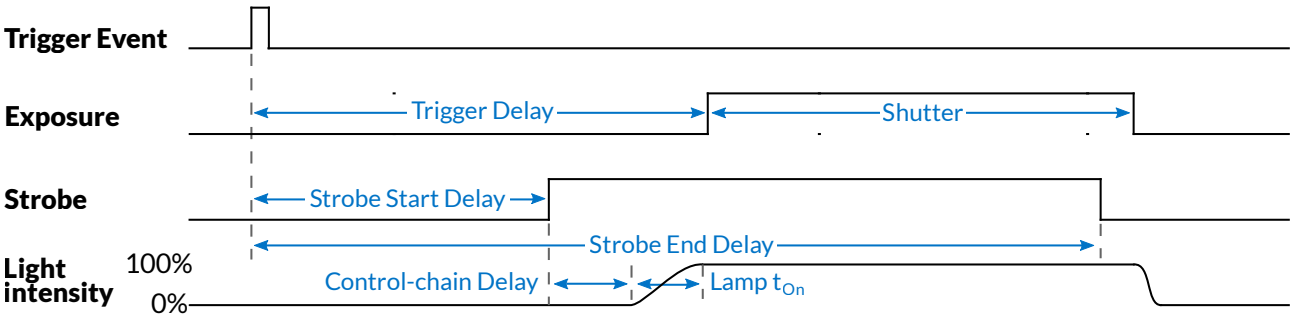


Figure 6.1: *Trigger delay with external light*

[5] `LineStartTrigger.Enabled = false`
[6] `ExposureEndTrigger.Enabled = false`

Note

The delay set through `TriggerDelay` forces `LinePeriod` to increment. Please check if resulting `LinePeriod` is still suitable for the application.

6.6.1 Encoder delay

Acquisition-start trigger and *frame-start* trigger can also delay the signaling of the trigger event for an amount of encoder steps, set through the module's `EncoderDelay` property.

The granularity of `EncoderDelay` is 1 encoder step.

When using `EncoderDelay`, `TriggerDelay` should be set to 0. If not, trigger event will be delayed by an additional `TriggerDelay` time after encoder required displacement.

6.6.2 Trigger overrun

Each trigger module generates its own ready signal indicating that it is able to receive a new trigger:

- `AcquisitionStartTriggerReady`
- `FrameStartTriggerReady`
- `LineStartTriggerReady`
- `ExposureEndTriggerReady`

These signals are named after the generating trigger module and can be sent to the output ports. You can also identify when the exposure is in progress by checking the exposure signal status.

A trigger module accepts an input signal only when its trigger-ready signal is active; trigger signals are otherwise ignored and increment an overrun counter. You can learn how many triggers have not been accepted by a trigger module through the `TriggerErrors` (read-only) property related to that trigger module.

Example Code 6.2 | *Usage of the* `TriggerErrors` *property*

```
if (device.FrameStartTrigger.TriggerErrors > 0)
    Debug.WriteLine("Invalid triggers detected.");
```

Figure 6.2 shows a possible timing of exposure and line-start-trigger-related signals.

The exposure begins after the specified trigger delay and a new trigger signal can be accepted only when `LineStartTriggerReady` is asserted. After receiving a valid trigger, the trigger-ready signal goes low, indicating that the requested exposure is being processed.

The second trigger in the picture is ignored as the `LineStartTriggerReady` signal is still low; the error condition is detected and increments the trigger overrun counter.

The third trigger signal is properly received and accepted as the previous exposure is already complete.

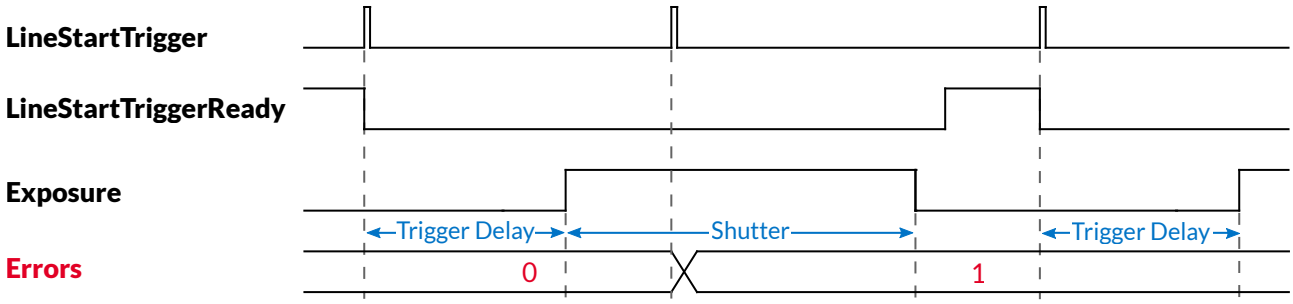


Figure 6.2: *Exposure timings*

6.7 TRIGGER CONFIGURATION EXAMPLE

6.7.1 External trigger frame acquisition

The following example assumes that you want to acquire a frame composed by 5 lines every time a rising edge on input port 1 is detected. Frame start trigger ready signal is routed to output port 3.

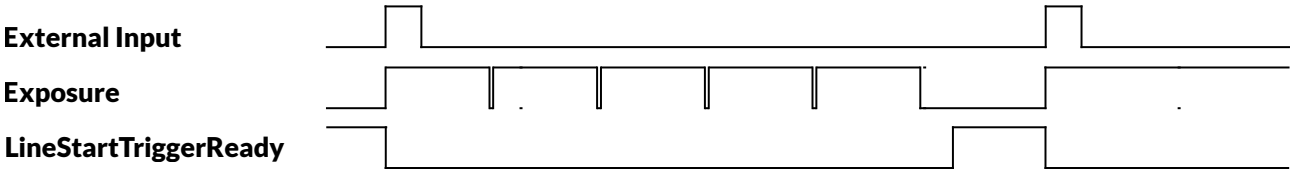


Figure 6.3: *External trigger frame acquisition timings*

As shown in Figure 6.3, the frame-start-trigger-ready signal has to go high before the rising edge of the signal on port 1. If this constraint is not respected, the trigger will not be accepted.

Example Code 6.3 | *External trigger line acquisition*

```
// Set 5 lines per frame
device.ImageSizeY = 5;
// Set shutter time to 1 ms
device.Shutter = (uint)(1e-3/device.GetFeatureUnitInfo(Feature.Shutter).Factor);
// Set debounce time to 1 us
UnitInfo units = device.PIOPorts[1].DebounceUnitInfo;
device.PIOPorts[1].DebounceTime = (ushort)(1e-6 / units.Factor);
device.FrameStartTrigger.Source = TriggerSource.External;
// Select input port 1
device.FrameStartTrigger.ExternalInput = 1;
// Rising edge
device.FrameStartTrigger.DetectExternalInputEdge = true;
device.FrameStartTrigger.InvertExternalInput = false;
// Enable frame trigger
device.FrameStartTrigger.Enabled = true;
// Export Trigger ready signal on port 3
device.PIOPorts[3].Source = OutputSource.FrameStartTriggerReady;
```

6.7.2 Signal-driven exposure time

The following example shows how to acquire a line every time a rising edge on input port 1 is detected. The exposure will end when a falling edge is detected on the same port. Line start trigger ready signal is routed to output port 3, while `ExposureEndTriggerReady` signal is routed to output port 4.

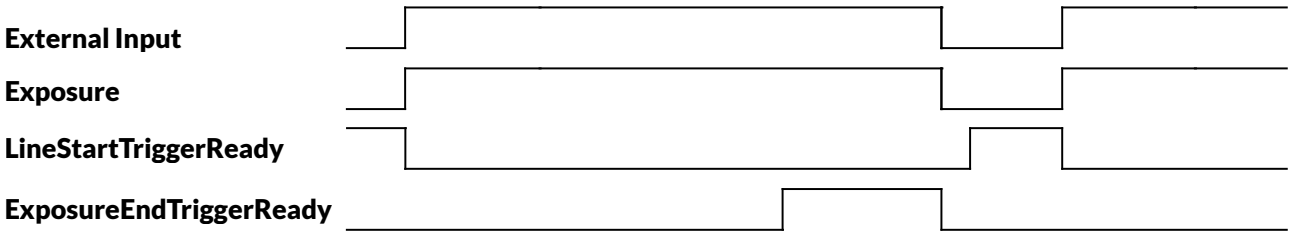


Figure 6.4: *Signal-driven exposure timings*

As shown in Figure 6.4, the `LineStartTriggerReady` signal has to go high before the rising edge of the signal on port 1. Similarly, the `ExposureEndTriggerReady` signal has to go high before the falling edge of the signal on port 1. If these constraints are not respected, the triggers will not be accepted.

Example Code 6.4 | *Signal-driven exposure time*

```
// Set debounce time to 1 us
UnitInfo units = device.PIOPorts[1].DebounceUnitInfo;
device.PIOPorts[1].DebounceTime = (ushort)(1e-6 / units.Factor);
// Configure LineStartTrigger on risingEdge event of Port 1
device.LineStartTrigger.Source = TriggerSource.External;
// Select input port 1
device.LineStartTrigger.ExternalInput = 1;
device.LineStartTrigger.DetectExternalInputEdge = true;
// Rising edge
device.LineStartTrigger.InvertExternalInput = false;
// Enable Line trigger
device.LineStartTrigger.Enabled = true;
// Configure ExposureEndTrigger on fallingEdge event of Port 1
device.ExposureEndTrigger.Source = TriggerSource.External;
// Select input port 1
device.ExposureEndTrigger.ExternalInput = 1;
device.ExposureEndTrigger.DetectExternalInputEdge = true;
// Falling edge
device.ExposureEndTrigger.InvertExternalInput = true;
// Enable exposure end trigger
device.ExposureEndTrigger.Enabled = true;
// Trigger ready routed on output 3
device.PIOPorts[3].Source = OutputSource.LineStartTriggerReady;
// Exposure End Trigger ready  routed on output 4
device.PIOPorts[4].Source = OutputSource.ExposureEndTriggerReady;
```

6.7.3 Encoder synchronization

In the following example the camera acquires images of some items carried by a conveyor. The camera receives an incoming trigger signal on port 0, whose falling edge enables the acquisition of 4 frames. Each frame consists of 512 lines and must be acquired every 1000 steps of an encoder, whose signals

are connected to ports 1 and 2. Since the tape speed is not constant, to prevent warping of the acquired image, the line exposure start must be synchronized to the encoder. Moreover, the application requests acquiring a line every 4/3 step.

Example Code 6.5 | *Encoder synchronization*

```
// Setup acquisition trigger:
// Set debounce time to 100 us
UnitInfo units = device.PIOPorts[0].DebounceTimeUnitInfo;
device.PIOPorts[0].DebounceTime = (ushort)(100e-6 / units.Factor);
// Set port 0 direction
device.PIOPorts[0].Direction = IODirection.Input;
// Enable acquisition start trigger
device.AcquisitionStartTrigger.Enabled = true;
// Set trigger source
device.AcquisitionStartTrigger.Source = TriggerSource.External;
device.AcquisitionStartTrigger.ExternalInput = 0;
// Detect falling edge
device.AcquisitionStartTrigger.InvertExternalInput = true;
device.AcquisitionStartTrigger.DetectExternalInputEdge = true;
// Set the number of frames to be acquired
device.AcquisitionBurstLength = 4;

// Setup frame trigger:
// Port 1 and 2 are input only, therefore there is no need to set port directions.
// Set debounce time to 1 us
device.PIOPorts[1].DebounceTime = (ushort)(1e-6 / units.Factor);
device.PIOPorts[2].DebounceTime = (ushort)(1e-6 / units.Factor);
// Set the encoder quadrature inputs
device.Encoder.InputA = 1;
device.Encoder.InputB = 2;
// Detect only forward movements
device.Encoder.IgnoreDirection = false;
// Enable frame trigger
device.FrameStartTrigger.Enabled = true;
// Set trigger source
device.FrameStartTrigger.Source = TriggerSource.Encoder;
// Acquire a frame every 1000 steps
device.FrameStartTrigger.EncoderInterval = 1000;

// Start acquisition
device.Acquire = true;
```

7 The processing chain

The image processing is performed on-camera, both by the sensor and the embedded Field-Programmable Gate Array (FPGA), saving computational power on your PC and leaving it available for the application. Please refer to Section 8.10 for more information about camera controls such as white balance, digital gain, etc. Figures 7.1 and 7.2 show the FPGA image processing chain in monochrome and color NECTA cameras. The sensor performs some analog operations such as black level calibration, CDS gain and analog gain.

The pre-processing operations involve calibration, ROI adjustments, binning and LUMA evaluation. At the end of these steps, when RAW output is selected, the frame is directly sent through the USB channel with no additional adjustment. Otherwise, the frame is further elaborated with the Image Processing block, which applies pixel intensity adjustments such as white balance, gain, demosaicing, Look-Up Table (LUT)s.

7.1 LIGHT METER

NECTA evaluates the compensation parameters (white balance, luma) over an image area called Light Meter ROI (LMR). It is contained into the current ROI and usually coincides with it, but it can be limited to a part of the ROI only to meet particular requirements (for example, to exclude a saturated region of the image).

The data analyzed by the light meter are sampled before any transformation by image processing controls such as digital gain, white balance, brightness, contrast and user LUT. Specifying the LMR requires setting its position and size through the `startX`, `startY`, `sizeX` and `sizeY` parameters[1].

[1] Light meter ROI position coordinates are set with respect to the upper left point of the current ROI.

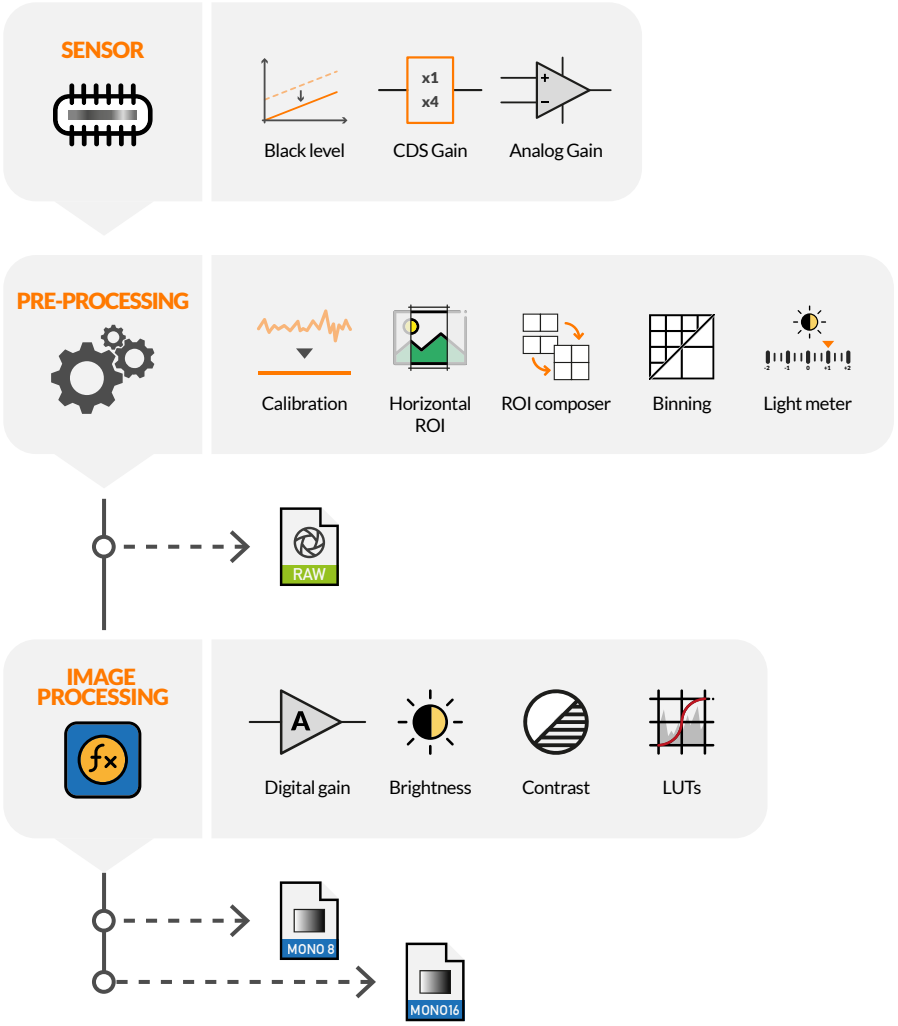


Figure 7.1: *Monochrome camera processing chain diagram*

Example Code 7.1 | *Set LMR starting from x = 256, y = 384, and 512 x 800 pixels wide*

```
device.SetLightMeterROI(256, 384, 512, 800);
```

7.2 CHUNK DATA

NECTA cameras can optionally transmit a set of additional information along with the acquired image, that can be used for debugging purposes and data analysis: a frame is identified by its serial number and time stamp; it can also be associated to the encoder position and the input status captured during its exposure.

Data are user-selectable by software methods. The bandwidth required is very small; however, it might reduce the maximum usable line rate, slightly affecting the performance in some high-demanding applications.

Frame chunk data are generated at the first line-start event of the frame. When both a frame-start and

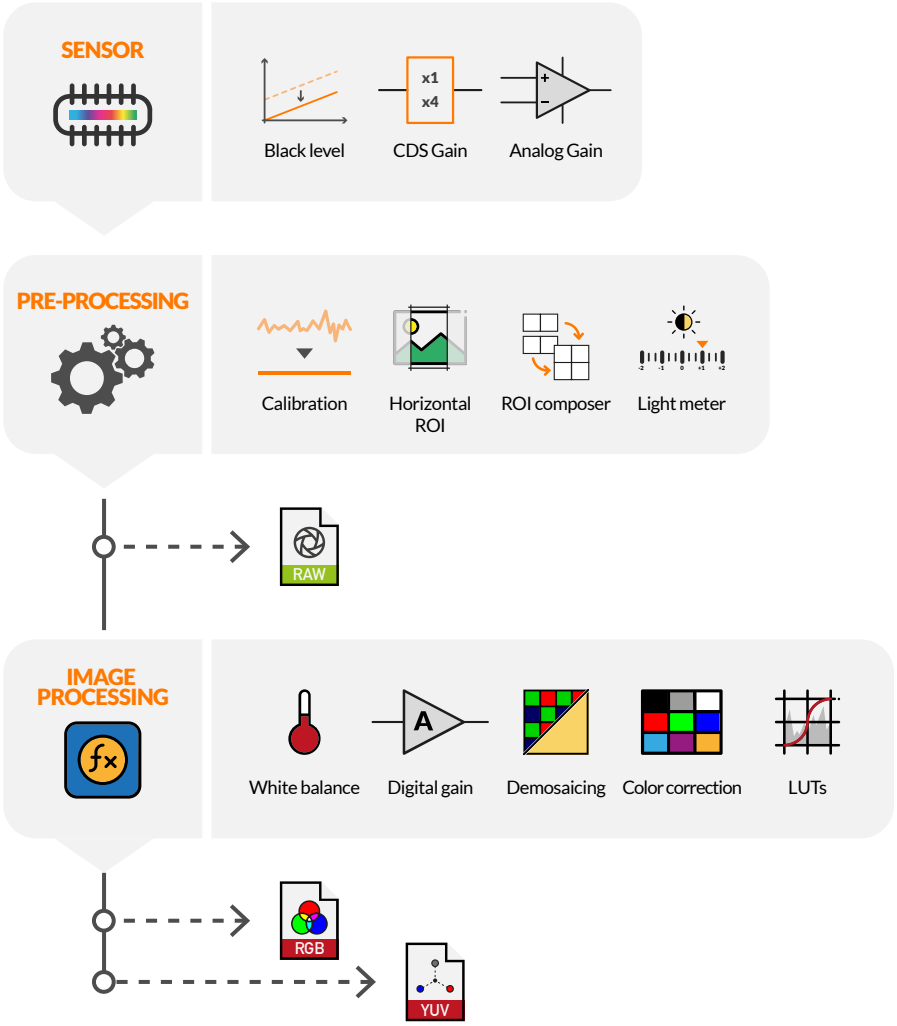


Figure 7.2: *Color camera processing chain diagram. Brightness, Contrast, Saturation and Hue are enclosed in the Color Correction*

a line-start triggers are present, values such as `FrameTimestamp` and `FrameEncoderPosition`, are acquired only at the first line-start trigger, which can be temporally and spatially distant from the frame-start trigger.

7.2.1 Line Number (line related)

The `LineNumber` is a progressive 16-bit unsigned integer. The related counter is reset at the acquisition startup and increment every line-start trigger event until 65535, then starts over.

Example Code 7.2 | *Enable line number chunk data field*

```
device.EnableChunkData = true;
device.EnableChunkDataField(ChunkDataField.LineNumber);
```

> **Note**
>
> Dual-line BW sensors (and color sensors when in Mode 1) expose simultaneously the two lines every line-start trigger event (when line delay is disabled). Thus, `LineNumber` increments once every two lines.

7.2.2 Frame Number

The `FrameNumber` field is a progressive 16-bit unsigned integer.

If the acquisition-start trigger is active (see Section 6.1), the counter is reset at the acquisition startup and counts from 0 to `AcquisitionBurstLength` – 1, then starts over; otherwise, the counter cycles from 0 to 65535, then starts over.

Example Code 7.3 | *Enable frame number chunk data field*

```
device.EnableChunkData = true;
device.EnableChunkDataField(ChunkDataField.FrameNumber);
```

7.2.3 Time stamp

The `TimeStamp` is a 16-bit unsigned integer generated from a 1 ms timer and cleared when acquisition is started. The `TimeStamp` value is the value of the timer counter, as captured at the first line-start trigger event.

Example Code 7.4 | *Enable time stamp chunk data field*

```
device.EnableChunkData = true;
device.EnableChunkDataField(ChunkDataField.FrameTimeStamp);
```

The `TimeStampExponent` property sets the resolution of the timestamp counter. This property represents the unit of time expressed in power of 10, i. e. a value of -3 means 10^{-3} = 1 ms.

A list of all possible values for the `TimeStampExponent` property can be retrieved in the following way:

Example Code 7.5 | *Time stamp exponent property*

```
short[] exp = device.GetAvailableTimeStampExponents();
if (Array.Exists(exp, t => t == -2))
    device.TimeStampExponent = -2;
```

7.2.4 Encoder position

The `EncoderPosition` is a 32-bit unsigned value indicating the position of the encoder as captured at the line-start trigger event (`LineEncoderPositon`) or at the first line-start trigger event of the acquired frame (`FrameEncoderPosition`).

The following example shows how to enable the `LineEncoderPosition32` chunk data field (the same procedure can be done for `FrameEncoderPosition32`):

Example Code 7.6 | *Enable encoder position chunk data*

```
device.EnableChunkData = true;
device.EnableChunkDataField(ChunkDataField.LineEncoderPosition32);
```

A 16-bit unsigned version of the encoder position can be retrieved enabling `LineEncoderPosition16` instead of `LineEncoderPosition32`.

7.2.5 Input Status

The `InputStatus` is the input status bitfield, sampled at the line-start trigger event (`LineInputStatus`) or at the first line-start trigger event (`FrameInputStatus`) of the acquired frame.

Example Code 7.7 | *Enable input status chunk data captured at frame-start trigger event*

```
device.EnableChunkData = true;
device.EnableChunkDataField(ChunkDataField.FrameInputStatus);
```

7.2.6 Configuration example

Example Code 7.8 | *Configure chunk data, capture a frame and display the relevant data*

```
// Enable chunk data transmission
device.EnableChunkData = true;
device.SetEnabledChunkDataFields(new ChunkDataField[] {
    ChunkDataField.FrameNumber,
    ChunkDataField.FrameTimeStamp,
    ChunkDataField.LineNumber,
    ChunkDataField.LineEncoderPosition,
    ChunkDataField.LineInputStatus
});
// Start acquisition
device.Acquire = true;

// Wait for a frame and extract chunk data
var objPair = device.GetImageChunk(true);
var chunkData = objPair.Second;
Debug.WriteLine("Frame number: " + chunkData.FrameNumber);
Debug.WriteLine("Timestamp: " + chunkData.FrameTimeStamp);
foreach (var line in chunkData.Lines)
{
    Debug.WriteLine("Line number: " + line.LineNumber);
    Debug.WriteLine("  Encoder: " + line.LineEncoderPosition);

    for (byte i = 0; i < line.LineInputStatus.Length; i++)
        Debug.WriteLine(
            "  Input[" + i.ToString() + "]: " + line.LineInputStatus[i]
        );
}
```

7.3 FLUSH

When the acquisition is interrupted before a whole set of lines has been acquired, you can still use the flush operation to retrieve a “partial” frame, in which the missing lines are replaced by dummy lines. Dummy lines are black in monochrome modes and green in color ones.

The acquisition can be interrupted by user through the `device.Acquisition = false` command or when external triggers are missing.

A flush operation can be invoked by software, automatically (auto-flush) when a programmable timeout expires or by a selectable external input.

7.3.1 Flush Usage Examples

Example Code 7.9 | *Configure external input to perform a flush operation*

```
// Check if flush is available.
if (device.FrameCombinerAvailable)
{
    // Check if external flush is available.
    if (device.FrameCombinerExternalFlushAvailable)
    {
        // Configure Input port 1 as Flush Input on the rising edge.
        device.FrameCombinerExternalFlushInput = 1;
        // Enables External Flush.
        device.FrameCombinerExternalFlushEnabled = true;
    }
}
```

Example Code 7.10 | *Set a flush timeout to 1s*

```
// Check if flush is available.
if (device.FrameCombinerAvailable)
{
    // Configure 1000 ms timeout.
    UnitInfo unit = device.FrameCombinerTimeoutUnitInfo;
    device.FrameCombinerTimeout = (uint)(1.0 / unit.Factor);
}
```

Example Code 7.11 | *Software flush*

```
// Check if flush is available.
if (device.FrameCombinerAvailable)
    // Software flush.
    device.FrameCombinerFlush();
```

8 Capturing Images

8.1 VIDEO MODES

Despite being a linear camera, NECTA provides *images* (frames) consisting of multiple consecutive lines. A frame is made of a programmable number of lines; both line length and line offset (the number of pixels to skip from the first effective pixel of the sensor) are programmable and specify the Region-Of-Interest (ROI). Like area-scan cameras, NECTA acquires through *video modes*, that can be associated with specific configurations (ROI, pixel format, line period, etc.). The desired video mode can be selected through the `VideoMode` property:

Example Code 8.1 | *Video mode configuration*

```
device.VideoMode = 1;       // Set Video Mode 1
```

Note

Video modes can be selected only when the camera is not acquiring.

8.1.1 Mode 0 (Normal)

Mode 0 is the most commonly used mode; it allows using all pixel formats (see Section 8.5) and makes all processing chain controls available.

8.1.2 Mode 1 (RAW)

Mode 1 allows acquiring RAW data from the sensor, with no additional processing (available pixel formats are RAW8 and RAW16 only). When in mode 1, the processing chain is disabled; this mode can

be used to make calculations starting from the bare raw data coming from the sensor. Fixed Pattern Noise (FPN) calibration remains active; its contribution can be made neutral (e.g. to evaluate a different FPN calibration) using the methods described in Section 8.2.

8.1.3 Mode 2 (Subsampling – Color models only)

Mode 2 performs a subsampling procedure on acquired data. Sensor pixels are grouped four-by-four in clusters called super-pixels (see Figure 8.1). Horizontal resolution is halved, thus data amount of each row is halved compared to the same pixel format of Mode 0. Users can choose between two binning modes: Sum and Average.

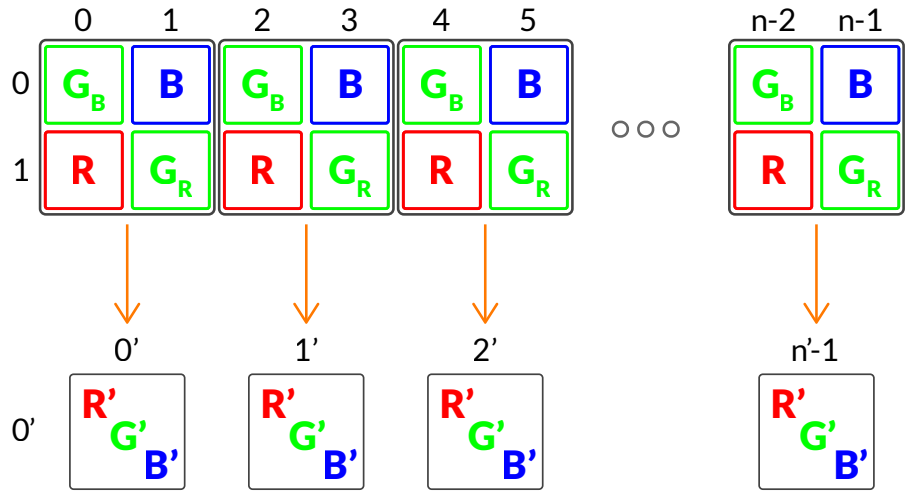


Figure 8.1: *Mode 2 subsampling*

8.1.3.1 Sum

Sum mode is useful when video signal is inadequate for your application. Resulting RGB pixel is calculated as follow:

$$R' = 2R \tag{8.1}$$

$$G' = G_R + G_B \tag{8.2}$$

$$B' = 2B \tag{8.3}$$

8.1.3.2 Average

Average mode is commonly used to reduce image noise. Resulting RGB pixel is calculated as follow:

$$R' = R \tag{8.4}$$

$$G' = \frac{G_R + G_B}{2} \tag{8.5}$$

$$B' = B \tag{8.6}$$

> **Note**
>
> When switching between these two binning modes, `Luma` property doesn't change. `Luma` value is calculated in pre-processing module, while subsampling is performed later in the image processing chain (see Chapter 7).

8.2 CALIBRATION

NECTA camera is able to automatically correct non-uniformities introduced by image sensor, lens and lighting system; the set of techniques used to improve the quality of the acquired images is generally called Shading Correction or Flat-Field Correction (FFC), and is a combination of offset and gain compensations. The calibration procedure combines three corrections involved in different phases of the frame acquisition process:

- Black level offset,
- Dark Signal Non-Uniformity (DSNU),
- Photo Response Non-Uniformity (PRNU).

All reference images must be captured using Video Mode 1 (RAW) and at the same operating temperature required by the application; the acquisition chain parameters (e.g. Shutter, Analog Gain, CDS Gain) must be set to the operating levels expected by the application.

> **Note**
>
> Every major change in the camera Overall System Gain (OSG) will change the calibration conditions, as well as changes in light setup and lens, so a new calibration will be needed. Gain, CDS Gain, ADC Resolution and Temperature will change the OSG.

> **Note**
>
> After calibration you can always use the Shutter control to change the amount of light coming to the sensor without affecting the camera calibration.

Warning

Calibration is performed with camera controls set by the user. To avoid image corruption due to FPN, is mandatory to save and reload calibration and user configuration simultaneously as explained in Section 8.11.

For further information, please refer to MaestroUSB3 manual and application examples.

8.2.1 Black Level Offset

The Black level is the common offset exhibited by all the pixels of the image sensor while not illuminated. It is inevitably inherent in the nature of the sensitive elements and the amplification chain; it is therefore important to keep it as low as possible to preserve the dynamics of the Analog to Digital Converter (ADC) and amplification chain.

NECTA provides a control, accessed through the `BlackLevelOffset` property, acting directly on the analog section of the sensor by subtracting the offset component before the AD conversion.

MaestroUSB3 viewer and application examples show in detail the use of the black level compensation; they also allow to experience how to evaluate the compensation level without obscuring the camera.

8.2.2 Dark Signal Non-Uniformity (DSNU) and Photo Response Non-Uniformity (PRNU)

The camera image sensor may also exhibit small differences in offset and gain between different pixels; this leads to both additive and multiplicative artifacts in the acquired image.

To even out these differences, two correction values are associated with each pixel:

- DSNU correction compensates for the differences in the base level of pixel values (offset) versus the average value of the line, by subtracting a programmable value.
- PRNU correction is used to reduce small differences between the gain of pixels and the average gain of the whole line; the same mechanism allows also to correct the lack of uniformity introduced by lens and lighting system.

The resulting correction for a generic pixel is described below:

$$P'_n = (P_n - DSNU_n) \cdot PRNU_n \tag{8.7}$$

To achieve an effective PRNU correction you must capture a reference picture with a gray background, using even illumination and the average typical value of the application; the acquisition chain parameters (gain, shutter, etc) must be set to the operating levels expected by the application.

Warning

The reference image for evaluating the PRNU correction must not contain saturated pixels. This may introduce artifacts due to an incorrect evaluation of calibration parameters.

The camera evaluates the reference compensation level within the current LMR, possibly smaller than the current ROI; this allows optimizing the algorithm according to a portion of the scanned image only. The PRNU correction algorithm allows also to bring the average level of the compensated image to a given target reference level (target gray level).

Note

The target gray level algorithm uses the Digital Gain control to achieve the required brightness level; when implementing the target gray level and saving the FPN configuration, you must care saving camera parameters according to Section 8.11. Using color NECTA models, White Balance control values will be saved to preserve the correct white compensation parameters.

MaestroUSB3 viewer and application examples show in detail the use of the PRNU compensation and allow you to experience the target grey level algorithm.

8.2.3 Restoring NECTA calibration

After calibration, you can save the result directly in a NECTA internal memory, so you can reload the calibration parameters when necessary. When powered on, NECTA does not preload any calibration; the user can either load one of stored in the internal camera memory during the setup or evaluate a new FPN correction.

Note

If calibration is saved using the deprecated method `device.Calibration.Save()`, relevant control values (Digital Gain, White Balance, Shutter etc.) must be saved and restored using methods described in Section 8.11.

If a calibration has been loaded from memory and needs to be cleared (e.g. choosing Mode 1 instead of Mode 0), user can recall an empty calibration:

Example Code 8.2 | *Empty calibration recall*

```
if (device.Calibration.InitAvailable)
    device.Calibration.Init();
```

8.3 REGION OF INTEREST

If you don't need to acquire the whole sensor image, you can get just part of it setting a Region-Of-Interest (ROI). This may save the time required to transfer the unnecessary parts of the image possibly increasing the resulting frame rate.

A ROI is defined by programming the requested number of rows and columns and the starting position (setting the image `StartX`, `SizeX` and `SizeY` parameters as shown in Figure 8.2):

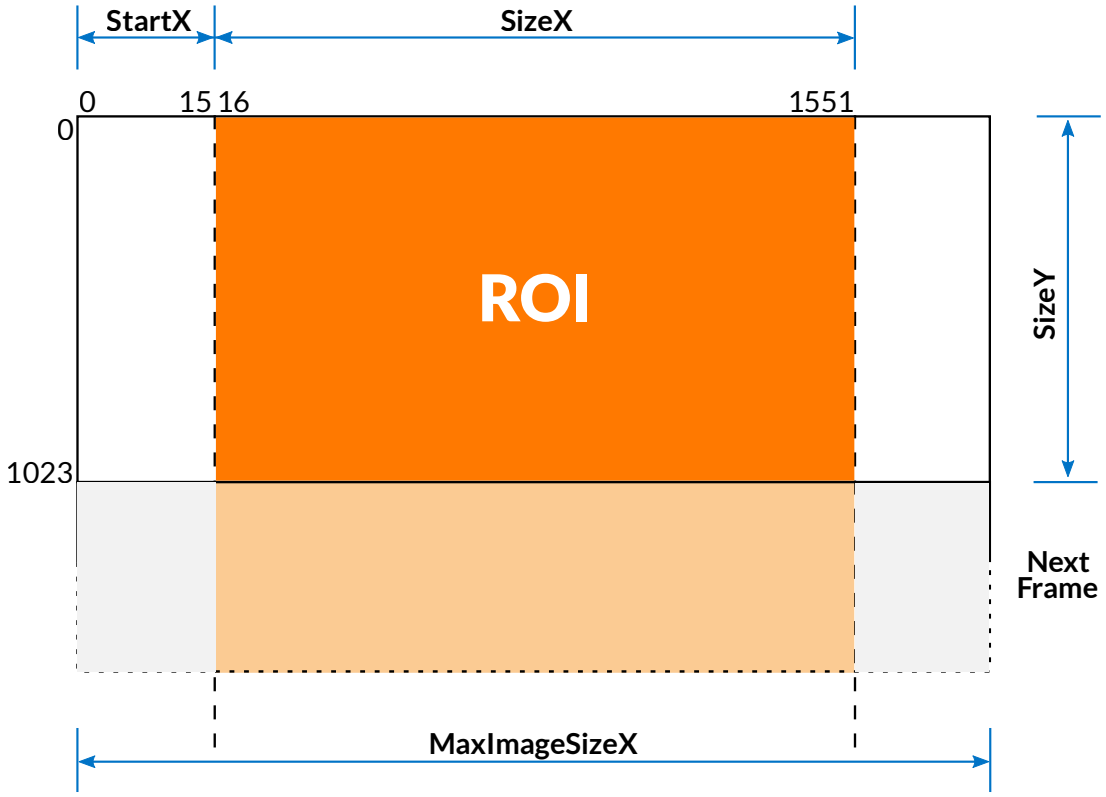


Figure 8.2: *ROI description*

Example Code 8.3 | *ROI configuring example*

```
// Ensures that ROI start position plus size does not exceed max sensor area
device.ImageStartX = 0;
// Set ROI size
device.ImageSizeX = Math.Min(device.MaxImageSizeX, 1536);
// Set ROI starting position
device.ImageStartX = 16;
// Set 1024 lines for each frame assuring it does not exceed the maximum allowed SizeY
device.ImageSizeY = Math.Min(device.MaxImageSizeY, 1024);
```

To avoid errors, the sum of `ImageStartX` and `ImageSizeX` must not exceed `MaxImageSizeX`.

Note

It is possible to set *ROI* parameters only when the camera is not acquiring.

Note

Changing ROI parameters involves automatic recomputing of the *packet size* boundaries (see Section 8.9.4).

8.4 BINNING (MONOCHROME MODELS ONLY)

NECTA cameras using monochrome sensors are able to digitally sum two or more adjacent pixels into one pixel. Binning is a digital aggregation that is applied after the analog to digital conversion. Refer to Section 2.2.1 for further information about available binning modes.

The horizontal binning mode combines the values of adjacent pixels on the same line (see Figure 8.3 and Figure 8.6) and the horizontal lenght of the acquired image is halved.

For example, starting from the ROI position depicted in Figure 8.2, horizontal binning would produce the following values:

$$MaxImageSizeX_{Bin} = \frac{MaxImageSizeX_{No-Bin}}{2} \tag{8.8}$$

$$SizeX_{Bin} = \frac{SizeX_{No-Bin}}{2} = 768 \tag{8.9}$$

$$StartX_{Bin} = \frac{StartX_{No-Bin}}{2} = 8 \tag{8.10}$$

All values will be truncated to their integer part.

Similarly, the vertical binning mode combines the values of adjacent pixels on the same column (see Figure 8.4 and Figure 8.7), and the vertical size of the acquired image is then reduced by a factor related to the chosen binning factor.

The combined binning mode enables both binning modes (see Figure 8.5 and Figure 8.8) and generates an image whose resolution is halved in both dimensions.

User can choose between two binning types: *sum* or *average*.

8.4.1 Sum-type binning

This procedure increases the sensor response in cases of low lighting. This action roughly makes the pixel area grow of the binning factor (x2 or x4).

Figure 8.3: *Monochrome Sum-type horizontal binning*

Figure 8.4: *Monochrome Sum-type vertical binning*

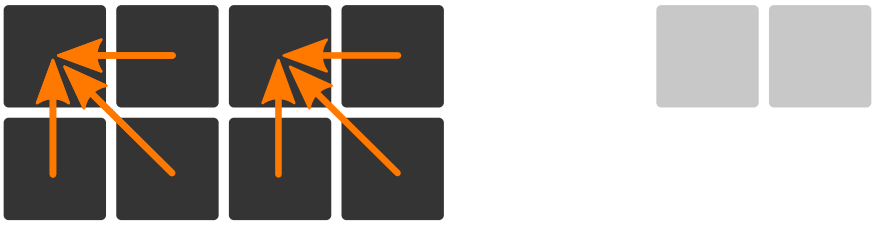


Figure 8.5: *Monochrome Sum-type combined binning*

Example Code 8.4 | *Sum-Type combined binning configuration*

```
if (device.BinningAvailable && device.BinningModeAvailable)
{
    if (device.GetAvailableBinningModes().Contains(BinningMode.Sum))
    {
        // Set sum-type binning.
        device.BinningMode = BinningMode.Sum;
        // Get an array of available horizontal binning values.
        byte[] horizontalBinnings = device.GetAvailableHorizontalBinnings();
        // Check whether 2x horizontal binning value exists in the array or not.
        if (Array.IndexOf(horizontalBinnings, 2) != -1)
            device.HorizontalBinning = 2;
        // Get an array of available vertical binning values.
        byte[] verticalBinnings = device.GetAvailableVerticalBinnings();
        // Check whether 2x vertical binning value exists in the array or not.
        if (Array.IndexOf(verticalBinnings, 2) != -1)
            device.VerticalBinning = 2;
    }
}
```

Example Code 8.5 | *Binning disabling procedure*

```
if (device.BinningAvailable)
{
    device.HorizontalBinning = 1;
    device.VerticalBinning = 1;
}
```

8.4.2 Average-type binning

When using average-type binning, NECTA computes the average value of the data acquired from 2 (or 4) adjacent pixels, thereby reducing the noise due to the acquisition conditions.

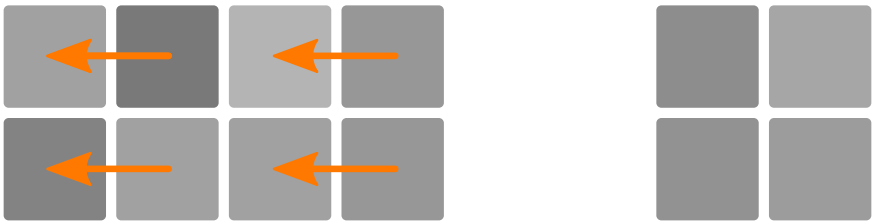


Figure 8.6: *Monochrome Average-type horizontal binning*

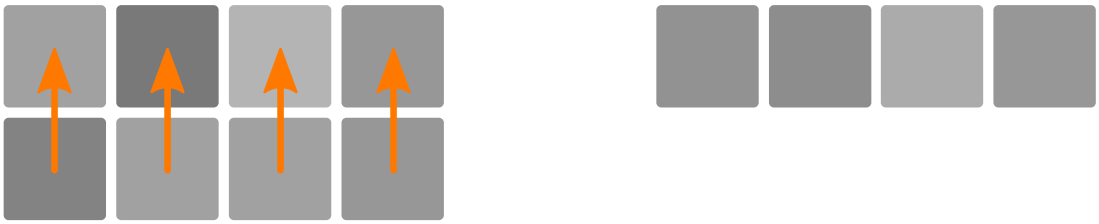


Figure 8.7: *Monochrome Average-type vertical binning*

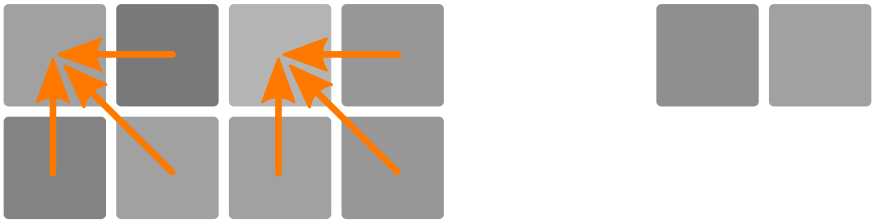


Figure 8.8: *Monochrome Average-type combined binning*

Example Code 8.6 | *Average-Type horizontal binning configuration*

```
if (device.BinningAvailable)
{
    if (device.BinningModeAvailable)
    // Set sum-type binning.
    device.BinningMode = BinningMode.Sum;
    // Get an array of available horizontal binning values.
    byte[] horizontalBinnings = device.GetAvailableHorizontalBinnings();
    // Check whether 2x horizontal binning value exists in the array or not.
    if (Array.IndexOf(horizontalBinnings, 2) != -1)
    device.HorizontalBinning = 2;
    // Get an array of available vertical binning values.
    byte[] verticalBinnings = device.GetAvailableVerticalBinnings();
    // Check whether 2x vertical binning value exists in the array or not.
    if (Array.IndexOf(verticalBinnings, 2) != -1)
    device.VerticalBinning = 2;
}
```

Note

The `BinningAvailable` property allows to check if the camera supports binning modes. Vertical binning mode is available for two-lines monochrome sensors only.

Note

When enabling horizontal binning, `ImageStartX` and `ImageSizeX` values will be halved. Vertical binning mode influences `ImageSizeY` and `UnitImageSizeY` instead.

Note

For two-lines monochrome sensors only: when acquiring a moving target, enabling the `LineDelay` control (see Section 5.7) together with vertical binning may be recommended.

8.5 PIXEL FORMAT

The *pixel format* parameter controls the format used for sending image pixels to the PC. The available formats depend on the camera model and the selected video mode:

Pixel Format	NECTA Color model	NECTA Monochrome model	Video Mode	Bytes per pixel
MONO8	✓	✓	0	1
MONO16	✓	✓	0 / 2	2
YUV422	✓	N.A.	0 / 2	2
RGB24	✓	N.A.	0 / 2	3
RAW8	✓	✓	1	1
RAW16	✓	✓	1	2

Table 8.1: *Available pixel formats*

When MONO8 is selected, the camera sends the 8 most significant bits of the image pixel luminance (Y) to the PC. When MONO16 is selected, the camera sends all 16 bit of the image pixel luminance (Y), sampled using the resolution selected by the *ADC resolution* control and aligned to the most significant bit; pixel data are *little endian* ordered.

Color cameras feature also YUV422 and RGB24 formats, providing color images; when one of these formats is selected, the camera performs an internal processing called demosaicing, which reconstructs the actual color of each pixel sampled according to the Bayer filter (see Figure 3.1).

The RGB24 format uses 3 byte (24 bit) per pixel; it may therefore represent 16.7 million colors.

When the YUV422 format is selected, after demosaicing the camera re-encodes the internal RGB representation: for each pair of pixels, the camera sends 2 byte for luminance (one per pixel) and 2 byte for chrominance, common to both pixels. It thus implements a simple 3:2 compression, still providing excellent image quality.

The RAW8 and RAW16 formats allow receiving raw data from the sensor, bypassing the processing chain; their representation is the same as MONO8 and MONO16 formats.

Note

When using RAW formats from color NECTA cameras, the camera output is a bayer filtered image (see Figure 3.1).

Example Code 8.7 | *Color coding selection*

```
device.ColorCoding = ColorCoding.Raw8;
```

Warning

It is possible to set the *Pixel format* only when the camera is not acquiring.

Note

Changing *pixel format* involves automatic recomputing of the *packet size* boundaries (see Section 8.9.4).

8.6 ADC RESOLUTION

Each sensor pixel has its own 12 bit Analog to Digital Converter (ADC), whose resolution may be possibly reduced to speed up the conversion: the lower selected *ADC resolution*, the lower ADC conversion time, the lower minimum allowable line period.

When the maximum allowed *ADC resolution* is selected, NECTA internal processing chain uses the whole pixel dynamics as supplied by the ADC through all the processing chain. The pixel value is truncated (e.g. in 8 bit color coding) as needed just before sending the image data to the PC. This grants that the highest pixel data accuracy is achieved.

On the other hand, reducing the *ADC resolution*, a lower number of bits is needed to represent pixel values, so that higher frame rates can be achieved on the cost of conversion accuracy. You can then find the best trade-off between frame rate and *ADC resolution* that meets the requirements of your application.

Note

The sensor's analog gain slightly depends on the selected *ADC resolution*; the same Gain value may therefore generate slightly different gain levels when operating at different *ADC resolution*. You should set the operating *ADC resolution* before choosing the Gain level required by the application.

Example Code 8.8 | *Set ADC resolution to 10 bit*

```
device.ADCResolution = 10;
```

Warning

It is possible to set *ADC resolution* only when camera is not acquiring.

Note

Changing *ADC resolution* involves automatic recomputing of the *packet size* boundaries (see Section 8.9.4).

8.7 LINE PERIOD

The *line period* parameter controls the acquisition period, i.e. the time between the acquisition of two lines.

Note

Dual line sensors expose the two rows simultaneously. Thus, *line period* is defined as the time between the beginning of two consecutive exposures.

The *line period* control has effect only when the line trigger is disabled. Otherwise, the time between two consecutive acquisitions is determined by the trigger frequency. NECTA provides two read-only variables reporting the allowed range (upper and lower limits) for the *line period*. The upper limit is constant (100 ms), the lower limit depends on the parameters controlling the operating conditions and is calculated according to them; for example, any direct or indirect change of the *packet size* control involves recomputing the minimum line period.

Unlike other video mode parameters, *line period* can also be modified during frame acquisition.

> **Note**
>
> When the minimum line period is recomputed due to parameter changes (e.g. the *packet size*), the current *line period* is automatically reset to the minimum *line period*. See Section 8.9.7 for further details.

In general, the minimum value for the *line period* is determined by four factors:

- the time required for the conversion and transfer of the line from the sensor to the camera;
- the exposure time (that prevails over the *line period*)[1];
- the bandwidth available on the USB bus;
- the size of the selected ROI.

To achieve the largest *frame rate* allowed by the application (see also Section 8.9.8):

- decrease the *ADC resolution* (decreasing the conversion time for each frame);
- ensure that the shutter time does not increase the minimum acquisition period;
- set the optimal packet size, i.e. the maximum allowed given current operating conditions;
- use the lowest ROI size compatibile with the application.

Example Code 8.9 | *Set* `LinePeriod` *property*

```
// Set 80 us, without exceeding the minimum allowed value.
device.LinePeriod = Math.Max(device.MinLinePeriod, 80.0);
```

8.7.1 High resolution Line period

This read-only property gives the actual value of line period that is configured into the sensor. When the exposure time approach to expected line period (or become longer), line period needs to be incremented by the given amount of time the sensor requires to perform the acquisition[1].

Example Code 8.10 | *Retireve High Resolution line period*

```
float actualLinePeriod = device.HighResLinePeriod;
```

8.8 LINE DELAY

The *line delay* control is available in all double-line (both color and monochrome) cameras.

[1] Actual *line period* is always larger than the current exposure time, plus a 2.1 µs overhead

8.8.1 Color cameras

Using the line delay control improves color reconstruction of the target along the scanning direction. As indicated in Section 3.1.2, NECTA color sensors feature two adjacent lines sensitive to color components according to the Bayer filter; for a proper color reconstruction, the delay between two consecutive exposures should equal the time for the target projection onto the sensor to move from one row to the adjacent one.

The *line period* value must be accurately set to d/v, where d is the size of the projected pixel dimension (refer to Section 2.2.1 for pixel size) on the working plane, and v is the target velocity; using this setup, each sensor line captures the same physical portion of the moving target and its color reconstruction is accurate.

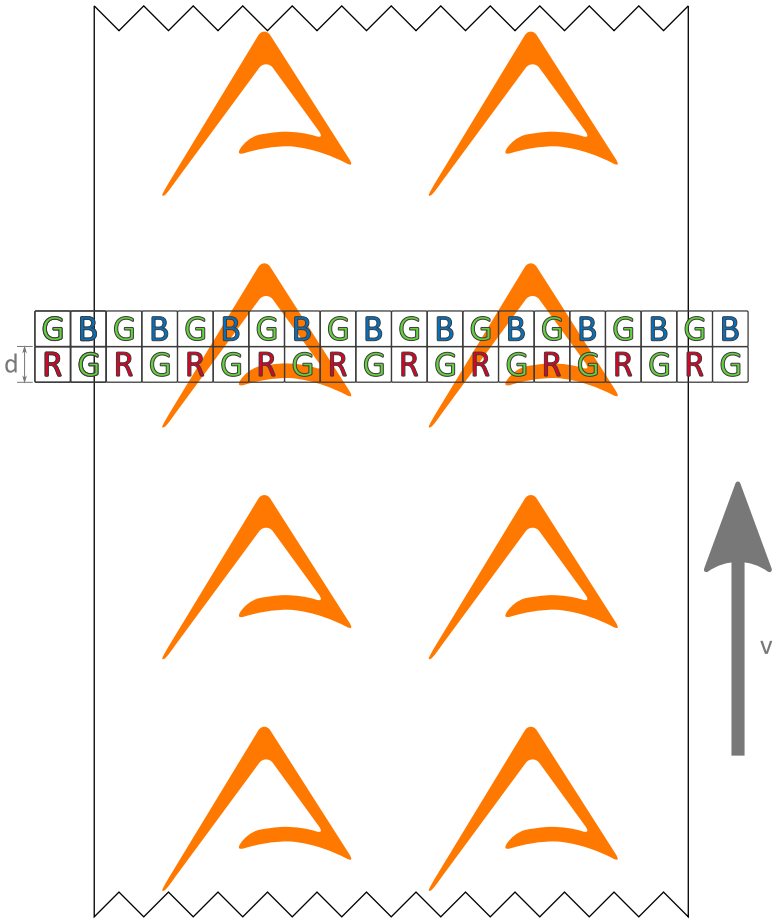


Figure 8.9: *Projection of sensor lines onto the working plane*

For example, being d = 70 µm due to the lens magnification factor, the required line period is 70 µs when the target is moving at v = 1 m/s.

Note

Using the *line delay* control requires setting the target motion direction (Forward or Reverse) according to the acquisition system layout (see Figure 8.10).

Example Code 8.11 | *Setup line delay*

```
// Turn LineDelay mechanism on (default statcamera is disabled)
device.LineDelay.Enabled = true;
// Choose either LineDelayDirection.Forward or LineDelayDirection.Reverse
// according to your setup
device.LineDelay.Direction = LineDelayDirection.Forward;
device.LinePeriod = 70.0;
```

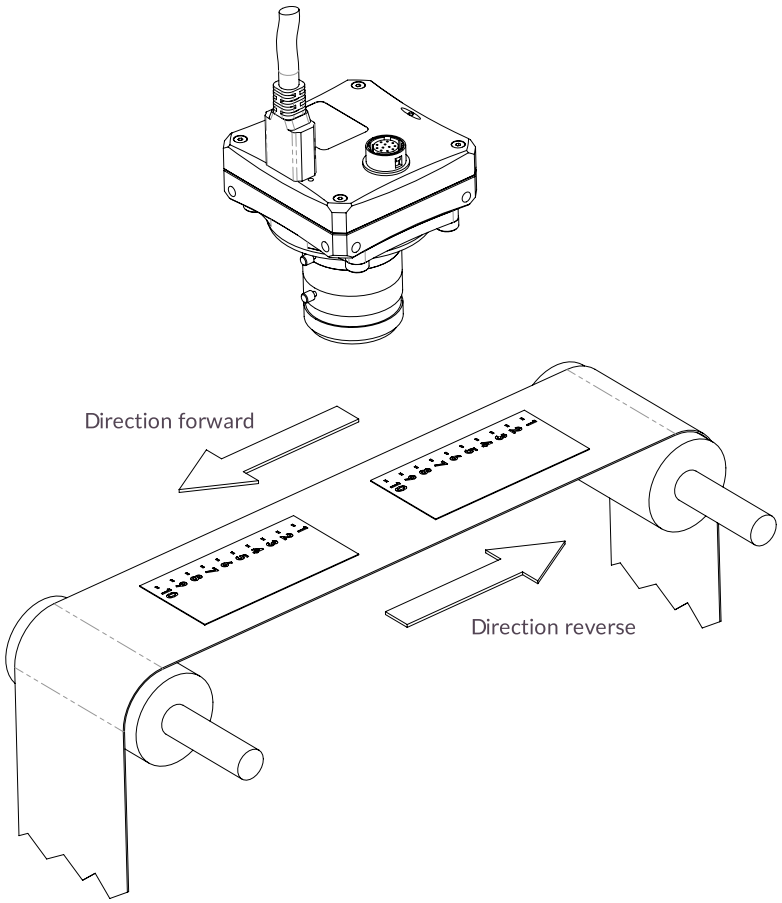


Figure 8.10: *Target Movement vs. Camera positioning (using an inverting lens)*

8.8.2 Monochrome cameras

For double-line monochrome cameras, the line delay control works as in color cameras. It may and must be used only when vertical binning mode is enabled (see Section 8.4) and the target in front of the camera is moving (see Figure 8.10). In these conditions using the line delay control improves image vertical sharpness.

For more details about line delay control settings refer to Section 8.8.1.

> **Note**
>
> Monochrome cameras can follow the target motion direction (Forward or Reverse) according to the acquisition system layout (see Figure 8.10) without enabling *line delay*.

8.9 FRAME RATE AND BANDWIDTH

Acquired frames are sent to the host computer through the USB 3.2 Gen 1x1 connection. This channel has a total bandwidth of about 5 Gbit/s. However not all the theoretical bandwidth is really available for USB data transmission. NECTA user can choose between the USB standard isochronous channel, taking advantage of a reliable bandwidth of up to 3.2 Gbit/s, or the bulk channel, more reliable on USB controllers that are integrated on PC motherboards.

8.9.1 Reserving bandwidth for isochronous channel

As shown in Section 2.5, the USB standard isochronous channel is based on a time interval of 125 µs, i.e. each second is divided into 8000 parts (or micro-frames). The bandwidth available for a device is defined by calculating the amount of KiB that the device can transmit during each micro-frame. The isochronous channel bandwidth cannot be reserved in any quantity: data transmission is made of 1024 Bytes-long packets. Each device can transmit a burst made from 1 to 16 packets, and up to 3 bursts in a micro-frame.

The configuration indicating the number of packets and bursts that can be sent in a micro-frame is called alternate settings (or simply alternate). Each device provides a series of alternates, each one defining a bandwidth to be reserved on the isochronous channel. Selecting one of these alternates means reserving its bandwidth for the current device. The selection of an alternate is automatically performed by API, based on current camera bandwidth requirements.

> **Note**
>
> Not all USB host controllers can handle the total available bandwidth: for more information about the recommended hardware configuration supporting the maximum NECTA cameras throughput, refer to Section 2.5.

8.9.2 Bandwidth limits for isochronous endpoint

User can limit the maximum bandwidth reserved to the camera through the `SetBandwidthLimits` method. This allows connecting multiple cameras on the same host without experiencing bandwidth sharing conflicts. It can also be used to limit camera performance when connected to a poor USB 3.2 Gen 1x1 host controller, as mentioned in Section 5.1.2.

USB Packet size	Burst size	Number of bursts	Bytes per micro-frame	Bandwidth
1 KiB	× 3	× 1	3 KiB	24 000 KiB/s
1 KiB	× 6	× 1	6 KiB	48 000 KiB/s
1 KiB	× 8	× 1	8 KiB	64 000 KiB/s
1 KiB	× 10	× 1	10 KiB	80 000 KiB/s
1 KiB	× 11	× 1	11 KiB	88 000 KiB/s
1 KiB	× 12	× 1	12 KiB	96 000 KiB/s
1 KiB	× 13	× 1	13 KiB	104 000 KiB/s
1 KiB	× 14	× 1	14 KiB	112 000 KiB/s
1 KiB	× 15	× 1	15 KiB	120 000 KiB/s
1 KiB	× 8	× 2	16 KiB	128 000 KiB/s
1 KiB	× 9	× 2	18 KiB	144 000 KiB/s
1 KiB	× 10	× 2	20 KiB	160 000 KiB/s
1 KiB	× 11	× 2	22 KiB	176 000 KiB/s
1 KiB	× 12	× 2	24 KiB	192 000 KiB/s
1 KiB	× 14	× 2	28 KiB	224 000 KiB/s
1 KiB	× 16	× 2	32 KiB	256 000 KiB/s
1 KiB	× 12	× 3	36 KiB	288 000 KiB/s
1 KiB	× 13	× 3	39 KiB	312 000 KiB/s
1 KiB	× 14	× 3	42 KiB	336 000 KiB/s
1 KiB	× 15	× 3	45 KiB	360 000 KiB/s
1 KiB	× 16	× 3	48 KiB	384 000 KiB/s

Table 8.2: *Available isochronous alternate settings*

Example Code 8.12 | *Sets a bandwidth limits of 32 KiB per microframe on NECTA*

```
device.SetBandwidthLimits(new uint[] {32});
```

Note

When calling the `SetBandwidthLimits` method, a valid alt-interface bandwidth –expressed as bytes per micro-frame– must be selected. To get a list of all available bandwidth-limit values the `GetAllowedBandwidthLimits` method can be invoked.

8.9.3 Bandwidth limit for bulk endpoint

In bulk endpoint mode the burst size is fixed at x16, resulting in a 16 KiB of data per bulk request. The bandwidth limit control can be seen as the average data transferred every 125 µs but has no effect on the burst size itself.

8.9.4 Packet size for isochronous endpoint

The packet size parameter controls the maximum amount of data the device can send during a micro-frame. Not all values are permitted: the boundaries for the allowed *packet size* are adjusted by the camera according to the operating conditions; they depend on several parameters (*ROI*, *ADC resolution*, *pixel format*, etc.) influencing both acquisition speed and size of data to be sent. Refer to Section 8.9.4.1 for more details about how the required bandwidth can be estimated.

Note

When the camera automatically recalculates the *packet size* range, it also sets the *packet size* to the current maximum allowable value.

Note

The maximum allowed value for the *packet size* is the optimum value, beyond which there is no frame rate increase. Setting a *packet size* larger than `MaxPacketSize` generates an exception.

Note

The amount of bandwidth available on each interface can be restricted by the `SetBandwidthLimits` method, refer to Section 8.9.2.

Before starting live image acquisition, based on the current *bandwidth limits* and current *packet size*, API selects the lowest available alt-interface providing the required bandwidth.

Note

Selecting a bandwidth limit stricter than the minimum required bandwidth makes acquisition start impossible. An exception is thrown to signal this error condition.

The following code sets the PacketSize to 8192 B per micro-frame (8 KiB per micro-frame), after checking it does not exceed the maximum allowed packet size.

Example Code 8.13 | *Set* `PacketSize` *property*

```
device.PacketSize = Math.Min(device.MaxPacketSize, 8);
```

Warning

`PacketSize` property can be set only when camera is not acquiring.

Note

`PacketSize` property has unit as specified in `PacketSizeUnitInfo` unit info.

8.9.4.1 Calculating the required bandwidth

Whenever an operating parameter of the camera is set, NECTA automatically calculates the upper and lower (optimum) `PacketSize` limits, optimizing camera performance. Nevertheless, it is useful to be aware of parameters affecting the acquisition speed and, consequently, determining bandwidth requirements (and vice versa).

The video stream bandwidth is influenced by video mode, ROI size, binning value, pixel format, ADC resolution, shutter floor, and chunk data. Refer to the relevant sections for information about these controls.

The following explanation is simplified for better understanding: the actual calculations made by NECTA to determine the required bandwidth and maximum frame rate are slightly more complex and take into account some additional details that have been left out here for the sake of simplicity.

In general, the required bandwidth is calculated based on the number of bytes per micro-frame μ_B generated by the sensor:

$$\mu_B = (I_B + CD_B) \cdot R \cdot \frac{125\mu s}{1s} \tag{8.11}$$

The quantity between round brackets is the total size in bytes of the frame and is the sum of two components:

- I_B is the image size in bytes and is the product of image width ROI_W, image height ROI_H and number of bytes per pixel P_B ($I_B = ROI_W \cdot ROI_H \cdot P_B$).
- CD_B depends on the number of enabled chunk data fields, i.e. the size (in bytes) of ancillary information related to each of the lines and the frame (see Section 7.2). CD_B can be calculated as $ROI_H \cdot CD_B^{line} + CD_B^{frame}$.

R is the image rate and is calculated as follows:

$$R = \frac{f_{SENSOR}}{W \cdot ROI_H} \tag{8.12}$$

where W is the overall size of the sensor (number of pixels in a row) and f_{SENSOR} is 100 MHz.

In other words, the above formula states that the number of bytes sent per micro-frame is the size in bytes of each frame multiplied by the image rate and divided by 8000 (number of micro-frames per second).

The value of $I_B + CD_B$ should be increased by 0.4 % due to communication packet headers overhead.

8.9.5 Packet size for bulk endpoint

Similarly to the bandwidth limit (Section 8.9.3), the packet size represents the average data produced by the camera every 125 µs. The calculation is performed in the same manner as the isochronous endpoint, but it is not possible to reserve such bandwidth due to the bulk transfer very nature.

8.9.6 Connecting multiple devices to the same host

When multiple devices are connected to the same host controller, the sum of the bandwidth used by all the devices must not exceed the total bandwidth available on the host (48 KiB max per micro-frame). The bandwidth limit control is the right tool that can be used to define the bandwidth to be reserved to a single device.

Note

A small part of the USB 3.2 Gen 1x1 bus bandwidth is reserved for controlling the camera: depending on the USB 3.2 Gen 1x1 controller you are using, adding another NECTA camera to the same controller may generate a small overhead (3-6%) slightly reducing the total bandwidth available for image transfer.

Warning

When using bulk endpoint, keep in mind that the total bandwidth is shared by all the devices connected to the host. Bandwidth can not be reserved by a single device but you can limit it by reducing the packet size properly.

8.9.7 Preserve rates

As mentioned before, NECTA updates the lower line period limit when the current packet size is modified. This happens either upon direct request (see Code 8.13) or when changing parameters affecting packet size limits (see Section 8.9.4.1).

Whenever the lower line period limit is updated, NECTA can decrease the current line period value to that value in order to guarantee the maximum available acquisition speed, or the current line period value can be preserved.

This behavior is controlled by the `PreserveRates` property. When true, NECTA keeps the previously

line period unaffected, as long as the old value is still inside the range of the allowed rates. Otherwise, if the property is set to false, the line period will always be minimized.

Example Code 8.14 | *Enable the* `PreserveRates` *functionality*

```
device.PreserveRates = true;
```

Note

NECTA cameras `PreserveRates` feature is enabled by default; it can be disabled by setting it to false.

8.9.8 Maximizing frame rate

To achieve maximum performance, you must pay special attention to correctly define some fundamental operating parameters such as lighting, exposure time and pixel format.

First of all, it is advisable to plan the environment where the camera will be operating so that the field is being illuminated with enough light to ensure exposure time is as short as possible; note that NECTA cameras are able to use exposure time down to about 1 µs and line period down as low as 10 µs (see Section 2.2.1 for sensor related line period).

To maximize performance, you must select the pixel format (color coding) using the smallest representation among those suitable for the particular application; for example, if you need to capture monochrome images just to show them on a display, selecting the MONO8 pixel format is possibly enough, as the extra information content provided by the MONO16 format is generally not reproducible by conventional screens.

Finally, avoid using ROI larger than necessary: it would generate unnecessary traffic, increasing the allocated bandwidth on USB 3.2 Gen 1x1 bus and slowing down the PC processing. Planning a proper use of the USB 3.2 Gen 1x1 bus bandwidth (choosing appropriate combinations of packet size) allows operating multiple NECTA cameras at once on a single PC, achieving sufficient performances for most applications and minimizing the overall hardware cost.

8.10 CONTROLS

NECTA cameras feature several controls, affecting the operation of the sensor and acting on the video signal processing chain. The API provides software methods that allow your application to control the camera; for more information about the numerical representation of each parameter, their range of operation and access policy, please refer to MaestroUSB3 SDK manual and examples.

Note

`Brightness`, `Contrast`, `Saturation`, `Hue`, `WhiteBalance`, `Gamma`, `DigitalGain`, `UserLut`, controls have no effect when using a RAW color coding

8.10.1 Brightness

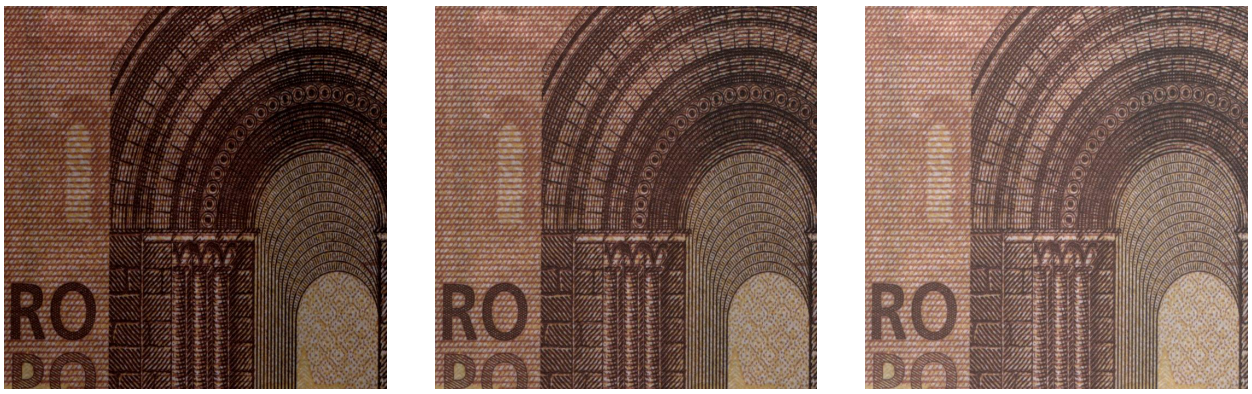
The `Brightness` control modifies the lightness of the acquired image.

Example Code 8.15 | *Set the brightness level*

```
device.Brightness = 64;
```

Note

Raising the exposure time has not the same effect as increasing the brightness: while the brightness equally affects each pixel, exposure has a strong bias on highlights, leaving shadows almost unaltered. To appreciate the difference, you can cover the lens to obtain a dark frame and move exposure and brightness sliders. While the frame remains black when exposure is changed, the color turns gray if the brightness is increased. We suggest to adjust `Gain` and `Shutter` values instead of using Brightness control.



(a) *Brightness 64* ***(b)*** *Brightness 128 (neutral)* ***(c)*** *Brightness 192*

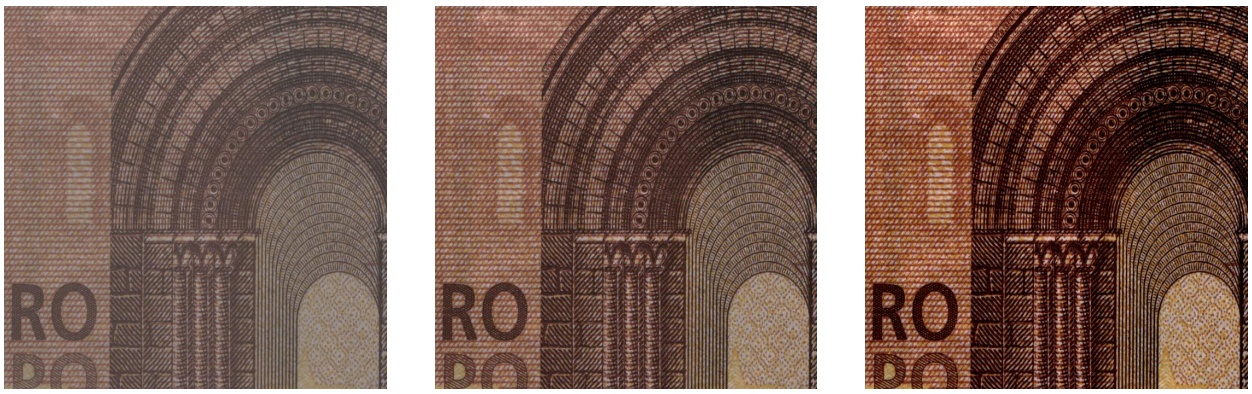
Figure 8.11: *Example of* `Brightness` *control effect.*

8.10.2 Contrast

The `Contrast` control modifies the ratio between the darkest and the lightest spot in the image.

Example Code 8.16 | *Set the contrast level*

```
device.Contrast = 144;
```



(a) *Contrast 64* ***(b)*** *Contrast 128 (neutral)* ***(c)*** *Contrast 192*

Figure 8.12: *Example of* `Contrast` *control effect.*

8.10.3 Color correction matrix (color models only)

Each RGB pixel can be modified using the Color Correction Matrix.

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = CCM \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} + CCO \tag{8.13}$$

$$CCM = \begin{bmatrix} k_{RR} & k_{GR} & k_{BR} \\ k_{RG} & k_{GG} & k_{BG} \\ k_{RB} & k_{GB} & k_{BB} \end{bmatrix} \tag{8.14}$$

$$CCO = \begin{bmatrix} O_R \\ O_G \\ O_B \end{bmatrix} \tag{8.15}$$

Default values for CCM and CCO correspond to the neutral correction:

$$CCM_{Default} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8.16}$$

$$CCO_{Default} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{8.17}$$

Note

When color correction matrix is enabled, Brightness, Contrast, Saturation and Hue controls become disabled. There is no interaction between matrix coefficients and the actual values of the controls above: changing one of CCM or CCO coefficients does not change any control value and viceversa.

Example Code 8.17 | *Set the color correction matrix coefficients*

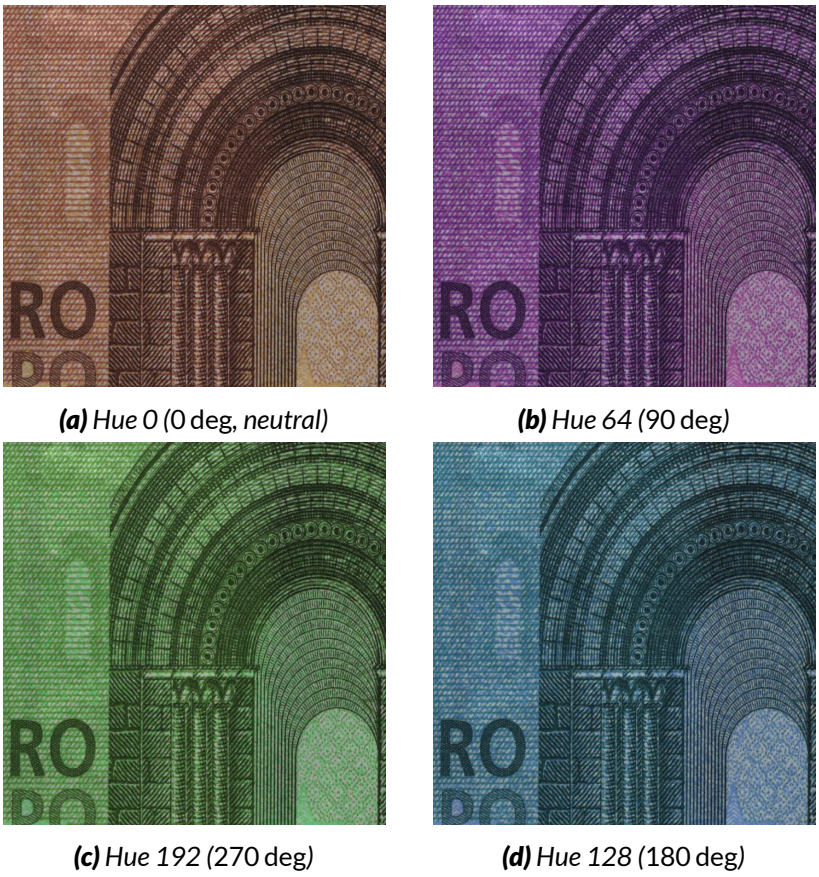
```
if (device.ColorCorrectionMatrixAvailable)
{
    float[,] matrix = new float[3, 3];
    // Populate coefficient matrix
    matrix[0, 0] = (float)kRR;
    matrix[0, 1] = (float)kGR;
    matrix[0, 2] = (float)kBR;
    matrix[1, 0] = (float)kRG;
    matrix[1, 1] = (float)kGG;
    matrix[1, 2] = (float)kBG;
    matrix[2, 0] = (float)kRB;
    matrix[2, 1] = (float)kGB;
    matrix[2, 2] = (float)kBB;
    // Write coefficients on camera
    device.SetColorCorrectionMatrix(matrix);
    float[] offset = new float[3];
    // Populate offset array
    offset[0] = (float)oR;
    offset[1] = (float)oG;
    offset[2] = (float)oB;
    // Write offsets on camera
    device.SetColorCorrectionOffset(matrix);
    // Enable color correction matrix
    device.ColorCorrectionMatrixEnabled = true;
}
```

8.10.4 Hue (color models only)

The Hue control modifies the dominant color of the acquired image to provide color enhancement.

Example Code 8.18 | *Set the hue level*

```
device.Hue = 140; // 0 corresponds to 0°, 255 corresponds to 360°
```



(a) *Hue 0 (0 deg, neutral)*

(b) *Hue 64 (90 deg)*

(c) *Hue 192 (270 deg)*

(d) *Hue 128 (180 deg)*

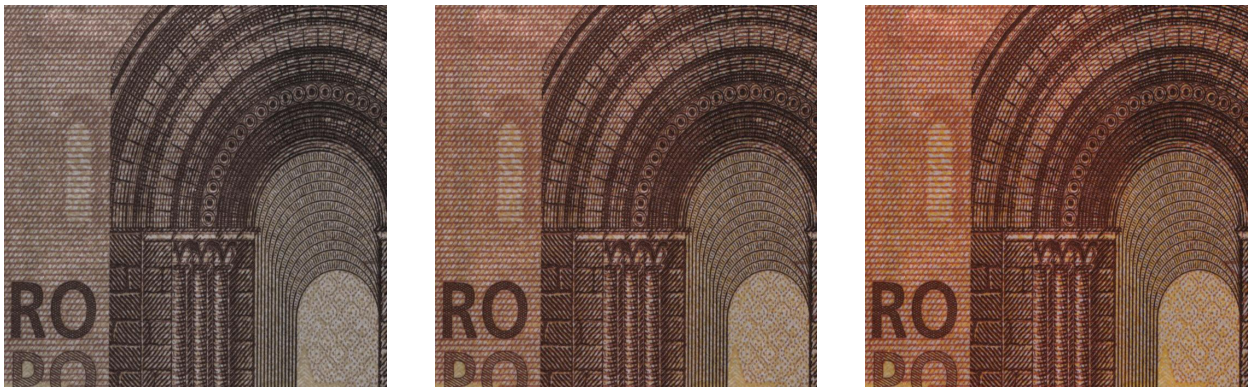
Figure 8.13: *Example of* Hue *control effect.*

8.10.5 Saturation (color models only)

The `Saturation` control modifies the color saturation of the acquired image.

Example Code 8.19 | *Set the saturation level*

```
device.Saturation = 140;
```



(a) *Saturation 64* **(b)** *Saturation 128 (neutral)* **(c)** *Saturation 192*

Figure 8.14: *Example of* `Saturation` *control effect.*

8.10.6 White Balance (color models only)

The `WhiteBalance` controls the relative mixture of primary colors (R, G and B); they allow to separately change the relationship between the Red/Green and Blue/Green image components to correct the colors resulting from the illuminating light and reset the actual white color to a neutral white.

The camera also provides an automatic mode called one-push, that acquires a frame and uses it as a reference for calculating an automatic white compensation. When using automatic compensation, the reference white surface must be acquired with uniform illumination, using the average light level required by the application. Note that the area of the frame where the camera takes the information for the automatic calculation may be changed setting the LMR (see Section 7.1).

Example Code 8.20 | *Set white balance automatically*

```
device.WhiteBalance();
```

Example Code 8.21 | *Set white balance manually*

```
device.WhiteBalanceUB = 1400;
device.WhiteBalanceVR = 800;
```

8.10.7 Gamma

The `Gamma` control (gamma correction) allows compensating the sampled image to match the response of some displays and to boost low light details. Calculation performed by the camera is:

$$P' = C_{Max} \cdot \left(\frac{P}{O_{Max}} \right)^{\gamma} \tag{8.18}$$

where P is the input pixel, P' the compensated pixel value, C_{Max} the maximum achievable value of output pixel (e.g. 255 for graph in Figure 8.15) and $O_{Max} = 2^{ADC\_resolution-1}$ is the maximum achievable value of original pixel (e.g. 2047 for curves in Figure 8.15).

The γ exponent can be selected in a range from 0.40 to 4.00 with a granularity of 0.01.

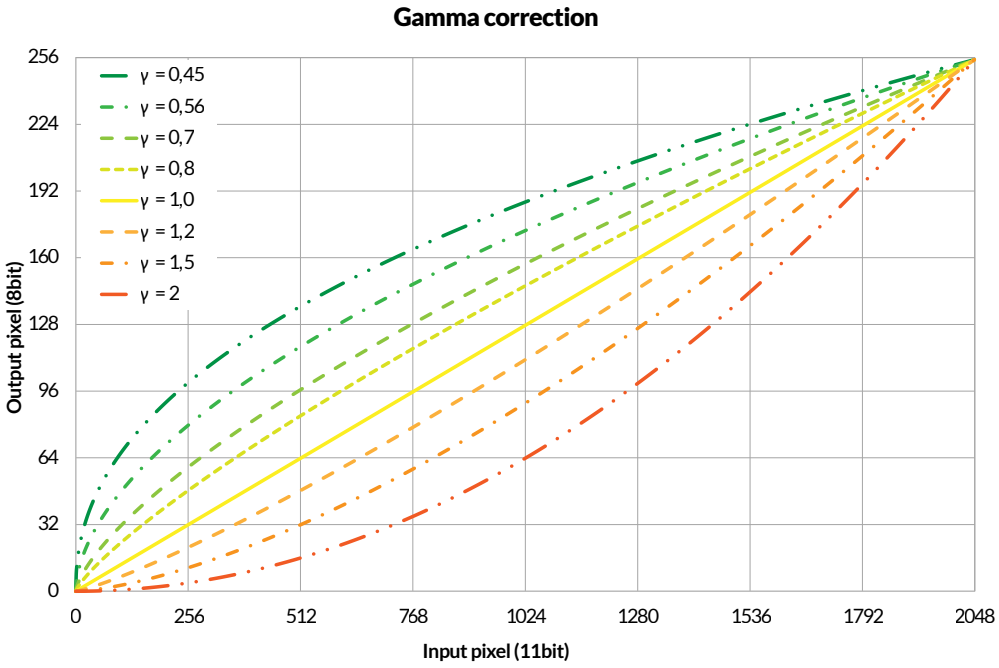


Figure 8.15: *Gamma correction representations*

Note

$\gamma = 1$ corresponds to the neutral correction (i.e. no correction applied).

Example Code 8.22 | *Set gamma*

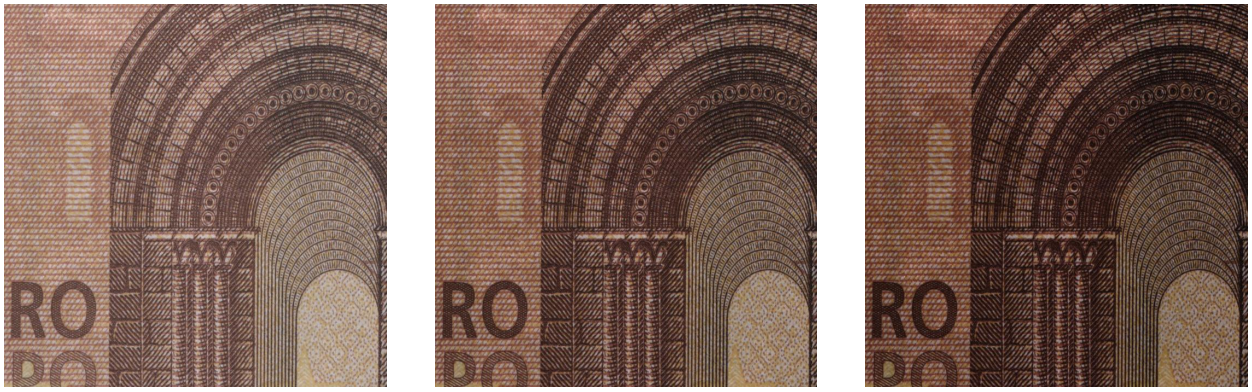
```
UnitInfo unit = device.GetFeatureUnitInfo(Feature.Gamma);
double gamma = 1.23;
device.Gamma = (uint)(gamma / unit.Factor);          // Convert to Gamma units
device.GammaEnabled = true;                        // Enable Gamma Correction
```

Warning

User LUT and gamma correction share the same processing chain resources and cannot be used simultaneously. At power-on, NECTA uses $\gamma = 1.00$ (neutral correction); disabling gamma correction will automatically enable user LUT.

Note

In user application a good approach is setting Gamma value when the camera is not acquiring, since during the Gamma loading operation NECTA could ignore external triggers and/or lose frames.



(a) *Gamma 0.7* ***(b)*** *Gamma 1 (neutral)* ***(c)*** *Gamma 1.3*

Figure 8.16: *Example of* `Gamma` *control effect.*

8.10.8 Shutter

The `Shutter` control adjusts the exposure time of the lines acquired by the camera (with 100 ns resolution). The following example sets the duration of the exposure time to 100 µs:

Example Code 8.23 | *Set Exposure time with unit factor*

```
// Retrieve ShutterUnit
UnitInfo unit = device.GetFeatureUnitInfo(Feature.Shutter);
// Convert exposure time in shutter units
uint reqShutter = (uint)(100e-6 / unit.Factor);
// Request minimum achievable exposure time in shutter units
uint minShutter = device.GetFeatureMin(Feature.Shutter);
// Set the greatest between requested exposure time and minimum shutter
device.Shutter = Math.Max(minShutter, reqShutter);
```

Note

Setting an exposure time longer than the current line period decreases the actual line rate. Consequently, current line period becomes slightly larger [a] than the exposure time.

[a] The line period will be equal to the currently selected exposure time increased by 2.1 µs.

8.10.9 High resolution line period

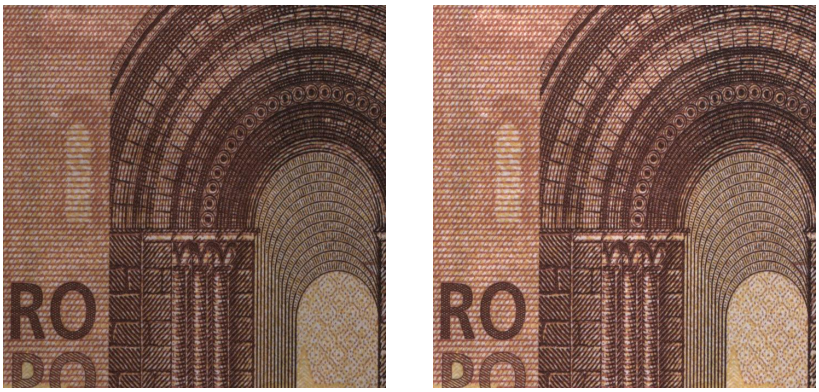
High-Resolution Line period is a read-only property which shows the actual value of line period that is configured into the sensor. When the exposure time approach to expected line period (or become longer), line period needs to be incremented by the given amount of time the sensor requires to perform the acquisition.

Example Code 8.24 | *Get the high resolution line period*

```
float actualLinePeriod = device.HighResLinePeriod;
```

8.10.10 Analog Gain

The `Gain` control adjusts the gain of the analog section of the sensor; it allows to adjust the analog sensitivity to compensate for insufficient lighting.



(a) *Gain 0 (base value)* ***(b)*** *Gain 7*

Figure 8.17: *Example of* `Gain` *control effect.*

Warning

To maximize the signal-to-noise ratio (SNR) of the acquisition, it is recommended to choose a lighting system that allows using the lowest possible `Gain` values.

Note

The sensor's analog gain is slightly dependent on the selected ADC resolution; the same Gain value may therefore generate slightly different gain levels when operating at different ADC resolution. You should set the operating ADC resolution before choosing the Gain level required by the application.

8.10.11 CDS Gain

NECTA sensors have a software-controlled integrated preamplifier which amplifies the signal acquired from active pixels by a factor of about 4; when the analog gain is not enough to ensure good lightness of the acquired image, it may be convenient to use it.

(a) *CDS Gain 0* ***(b)*** *CDS Gain 1*

Figure 8.18: *Example of* CDS `Gain` *control effect.*

Example Code 8.25 | *Set the CDS Gain amplifier*

```
device.CDSGain = 1;
```

Warning

To maximize the signal-to-noise ratio (SNR) of the acquisition it is recommended to use the *CDS Gain*, if necessary, in combination with the minimum suitable *Analog Gain*.

8.10.12 Digital Gain

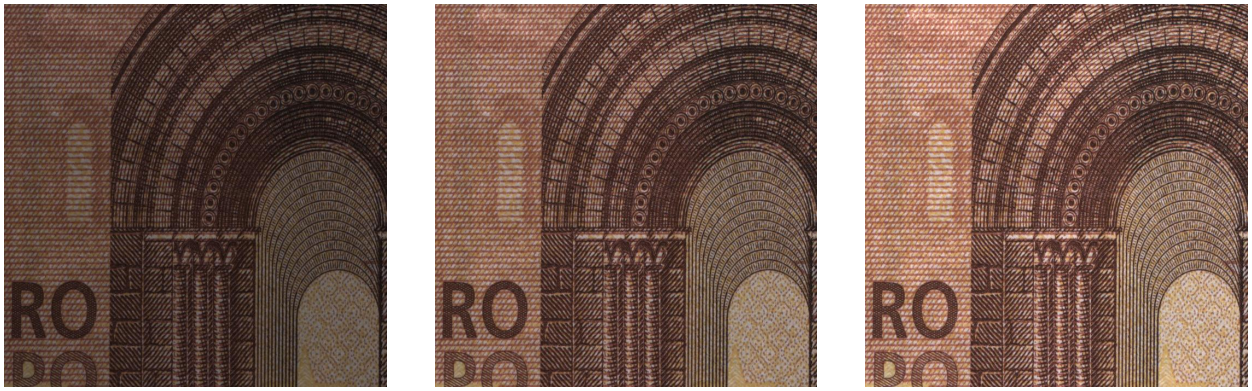
When, despite using the sensor analog amplifiers (Gain control), the resulting images cannot achieve the desired brightness level, you can enable a digital amplifier module scaling the acquired signal up.

Note

The unity gain value is 1024, therefore the digital amplifier module can also be used as an attenuator choosing values below 1024.

Example Code 8.26 | *Set digital gain level*

```
// Set digital gain to 1536/1024 (gain step = 1/1024)
uint maxDigitalGain = device.GetFeatureMax(Feature.DigitalGain);
device.DigitalGain = Math.Min(1536, maxDigitalGain);
```



(a) *Digital Gain 700* ***(b)*** *Digital Gain 1024 (neutral)* ***(c)*** *Digital Gain 1300*

Figure 8.19: *Example of* `Digital Gain` *control effect.*

Warning

By its very nature, the digital amplification may generate missing codes; it may happen that the combination of the ADC resolution and the Digital gain make some brightness values less frequent (or even suppress them). If these artefacts adversely affect the acquisition, they can be reduced by selecting the maximum ADC resolution compatible with the speed requirements of the application.

8.10.13 User LUTs

NECTA can apply a conversion table (look-up table, or LUT) to the incoming image data to change color/brightness distribution. The API functions allow to store up to four LUTs into the camera non-volatile memory; viewer and application examples show in detail how to use User LUTs and experience the advantages.

Warning

User LUT and gamma correction share the same processing chain resources and cannot be used simultaneously. At power-on, NECTA uses $Gamma = 1.00$ (neutral correction); disabling gamma correction will automatically enable user LUT.

To apply a user LUT, you must save it first into the camera; the following example stores a LUT into the camera and selects it:

Example Code 8.27 | *Store lut into the camera*

```
float[] lut = new float[device.UserLut.LutLenght];  // Create a linear sample LUT
for (int i = 0; i < device.UserLut.LutLenght; i++)
    lut[i] = (float)i / ((float)device.UserLut.LutLenght - 1);

if (device.UserLut.LutWritable(1))          // If LUT 1 is writable
    device.UserLut.Write(1, lut);           // Write LUT 1 to camera memory

device.LutIndex = 1;                        // Select LUT 1
device.GammaEnabled = false;                // Disable gamma to enable LUTs
```

Note

In user application a good approach is setting LUT index when the camera is not acquiring, since during the LUT loading operation NECTA could ignore external triggers and/or lose frames.

8.10.14 Luma

Luma is a read-only control that returns the average image luminance; Luma is evaluated within the currently selected LMR (see Section 7.1). The returned value is related to the last acquired image when the request was received by the camera.

Example Code 8.28 | *Read the Luma value of the last image*

```
uint currentLuma = device.Luma;
```

Note

Color cameras evaluate Luma using conversion factors $R = 0.299, G = 0.587, B = 0.114$.

8.10.15 Time Stamp

TimeStamp is a read-only control that returns the status at runtime of timeStamp counter. TimeStamp is a 16-bit unsigned integer generated from a 1 ms timer and cleared when acquisition is started.

Example Code 8.29 | *Read the TimeStamp value*

```
ushort currentTime = device.TimeStamp;
```

8.10.16 Flip and Rotate

The Flip control allows flipping the image horizontally, vertically and diagonally.

Example Code 8.30 | *Flip the image diagonally*

```
device.Flip = FlipMode.Diagonal;
```

The Rotate control rotates the image counterclockwise along the orthogonal axis across its center; the granularity of the rotation is 90 deg.

Example Code 8.31 | *Rotate the image of 270 deg*

```
device.Rotate = RotateMode.R270;
```

8.11 SAVING DEVICE CONFIGURATION

You can save the status of NECTA camera controls and calibration and reload it on demand: camera setup is saved into the internal flash memory and can then be reloaded even after powering off and on the device.

NECTA cameras can store multiple user configuration, allowing user to preset various scenarios and recall them whenever needed. The maximum number of user configuration is camera-dependant;

Example Code 8.32 | *Retreive the maximum number of user configurations*

```
ushort cfgNum = device.MaxFlashIndex() + 1;     // Get max number of user configurations
```

Example Code 8.33 | *Save and restore camera configuration and calibration*

```
ushort cfgIndex = 1;
// Save control and calibration into camera memory
device.Save(cfgIndex);
// Load control and calibration from camera memory
device.Load(cfgIndex);
```

Example Code 8.34 | *Erase configuration*

```
// Erase control and calibration from camera memory
device.Erase(cfgIndex);
```

8.11.1 Export and import XML

NECTA cameras allow user to export and import camera configuration and calibration to an XML file. It can be useful to share camera status during remote assistance session or to configure multiple cameras with the same set of parameters.

Note

Calibration parameters are exported to XML if and only if a calibration process has been conducted or recalled from camera memory.

Example Code 8.35 | *Export camera configuration and calibration previously saved on camera memory*

```
ushort cfgIndex = 2;
// Load control and calibration from camera memory
device.Load(cfgIndex);
// Creating xml file name
string filename = "C:\Users\UserName\Desktop\"CamCfg + cfgIndex.ToString() + "."xml;
// Export xml file
device.SaveXml(filename);
```

Example Code 8.36 | *Load from XML and save into camera memory*

```
ushort cfgIndex = 1;
// Import camera configuration from xml file
device.LoadXml"(C:\Users\UserName\Desktop\CamCfg."xml);
// Save control and calibration into camera memory
device.Save(cfgIndex);
```

8.11.2 User configuration (deprecated)

User can save and load the status of NECTA camera controls through dedicated function.

Example Code 8.37 | *Save and restore camera configuration from camera memory*

```
device.UserConfig.Save();          // Save control status into camera memory
device.UserConfig.Load();          // Load control status from camera memory
```

Example Code 8.38 | *Erase user configuration from camera memory*

```
device.UserConfig.Erase(1);     // Erase control status from camera memory
```

8.11.3 Calibration

After calibration procedure, you can save all calibration parameters into camera memory.

Example Code 8.39 | *Save and restore calibration from camera memory*

```
device.Calibration.Save(3);     // Save calibration data into camera memory
device.Calibration.Load(3);     // Load calibration data from camera memory
```

Example Code 8.40 | *Erase calibration from camera memory*

```
device.Calibration.Erase(2);        // Erase calibration data from camera memory
```

Warning

Calibration is performed with camera controls set by the user. To avoid image corruption due to FPN, is mandatory to save and reload calibration and user configuration simultaneously as explained in Section 8.11.

8.12 CONFIGURATION EXAMPLE

Example Code 8.41 | *Set camera's main parameters and enable live acquisition*

```
device.Camera = 0;                          // Connect to the first camera
device.ImageSizeY = 1024;                   // Set the height of the frame
device.VideoMode = 0;
device.ColorCoding = ColorCoding.Mono16;    // Set the pixel format
device.LiveControl = form.LivePanel;        // Target control for displaying imgs into
device.Acquire = true;                      // Start acquisition
```

8.13 PATTERN GENERATOR

NECTA cameras implement an internal fixed pattern generator, which provides synthetic images and allows you to check if the camera is correctly installed and working over the USB 3.2 Gen 1x1 connection. The pattern generator simulates the behavior of the camera implementing all the parameters controlling the acquisition, except for those strictly dependent on the sensor (e.g. analog gain, CDS gain, shutter, etc.).

While being related to the sensor, the ADC resolution has been emulated as well, as it has a large impact on camera performance: it allows to evaluate the camera performance under real operating conditions.

Example Code 8.42 | *Enable pattern generator*

```
device.PatternGen.Enabled = true;
```

Warning

Enabling or disabling the pattern generator during acquisition may lead to a frame loss.

9 Alkeria player

MaestroUSB3 comes with a user-friendly software application, allowing to explore the main features of NECTA cameras and check the correct camera and software installation. Once installed MaestroUSB3 SDK, you can find the application at the following path:

Program Files\Alkeria\USB3\MaestroUSB3\Players

Alternatively, you can easily start the camera viewer by accessing the Windows Start button and look for *Alkeria player* from the MaestroUSB3 Program folder:

Start->All programs->Alkeria->USB3->MaestroUSB3.

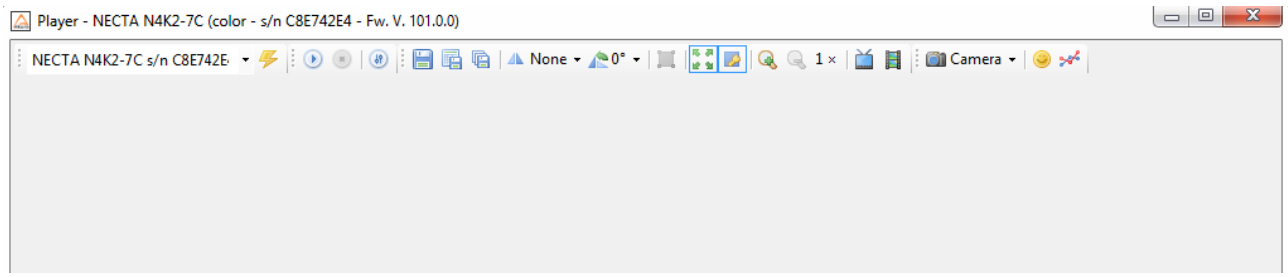


Figure 9.1: *Alkeria player main window*

9.1 DEVICE CONNECTION AND INITIALIZATION

The top left toolbar is a drop-down menu letting you select a specific NECTA camera among the ones connected to the system. When no NECTA camera is connected, the drop-down menu and the toolbar are grayed out.

9.2 TOOLBAR BUTTONS

The top toolbar shows the controls listed in Table 9.1:

Init	Initialize the camera to default settings (camera power-up status).
Play	Start image playback.
Stop	Stop image playback.
Settings	Open the settings panel.
Save	Save the last acquired image as a Windows Bitmap.
Save RAW	Save the last acquired image as a RAW binary file.
Save Sequence	Open the Save Sequence panel.
Flip	Select horizontal and/or vertical flip.
Rotate	Select image rotation mode (0°, 90°, 180°, 270°).
Set LMR	Specify the Light Meter ROI by drawing a rectangle on the image.
Stretch	Enable image stretching.
Maintain Aspect Ratio	Maintain image width/height ratio during stretching operation.
Zoom In	Increase the zoom level (and disable stretching).
Zoom Out	Decrease the zoom level.
Full Screen	Enter full-screen mode.
Temperature Status Normal	Signals that the camera is working within the safe operation area.
Temperature Status Warning	Signals that the current temperature is above 70 °C. Even if the camera is able to acquire images, working at this temperature is not recommended.
Temperature Status Fault	Signals that the current temperature is above 80 °C. The image acquisition is suspended. For more details on the temperature safety mechanism refer to Section 3.6.6.
Display Frames	Open Display Frames panel.
LUT	Launch the Look-Up Table editor.

Table 9.1: *Alkeria player buttons*

9.3 SETTINGS PANEL

9.3.1 Features

The features tab, shown in Figure 9.2, allows adjusting all camera controls, such as:

- `Shutter`: defines the exposure time, i.e. the time span in which the sensor is exposed to light;
- `Gamma`: applies a non-linear transformation to the image, defined by the relationship $I_{out} = I_{in}^{\gamma}$, where I is the pixel luminosity;
- `Gain`: controls the analog gain on sensor's circuitry;

- `Contrast`: modulates the ratio between the darkest points on the image and the lightest ones;

Luma and Temperature features have a “Read” button on the right which updates the value from the camera. Gamma control can be turned on and off (LUT Index control will be activated when Gamma control is turned off).

Note

Shutter control has an “Auto” check-box, allowing you to enable auto-shutter feature.

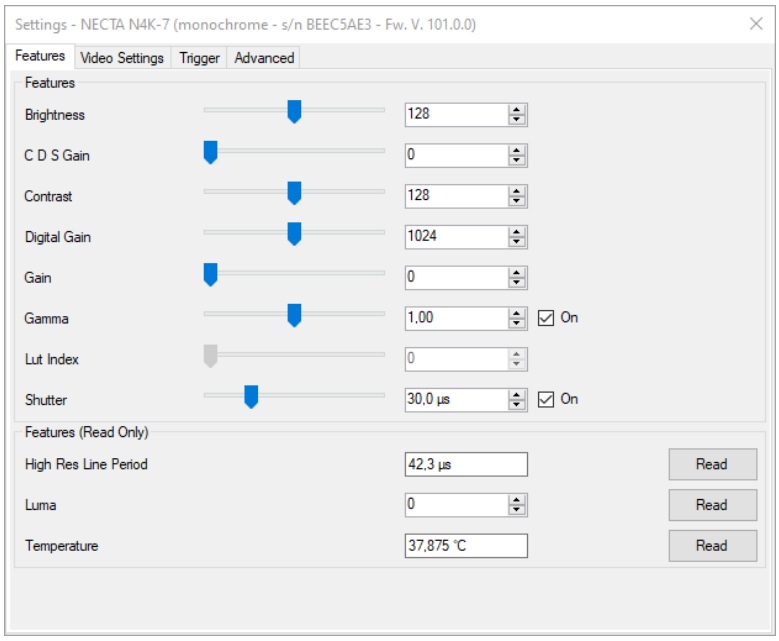


Figure 9.2: *Features controller tab*

9.3.2 Video Settings

The Video Settings tab (Figure 9.3) allows the selection of video mode, color coding and ADC resolution. Other relevant video settings parameters are:

- `Region-Of-Interest (ROI)`: sets the size of the image (width and height), and the displacement from the upper-left corner of the original frame (left and top).
- `Frame Combiner`: allows to pack multiple frames into a single frame.
- `Rates and Bandwidth`: adjusts the acquisition speed;
- `Pattern Generator`: displays a fixed pattern instead of the acquired frame;
- `Binning` (monochrome models only): sets the horizontal and/or vertical binning;

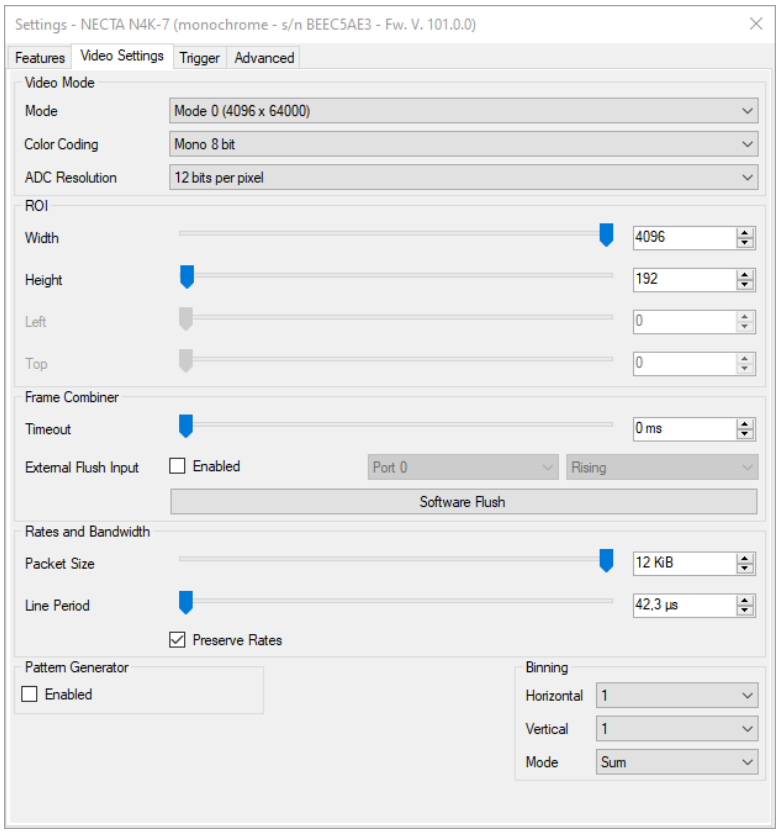


Figure 9.3: *Video settings tab*

9.3.3 Trigger

The Trigger panel (Figure 9.4) allows to setup the Trigger Manager module. Various events can be triggered, either via software or external sources (I/O):

- Acquisition start;
- Frame start;
- Line start;
- End of exposure.

Ticking the check-boxes on the right of the panel enables the related trigger modules. Drop-down lists can be used to select trigger source, delay and encoder step interval.

The bottom part of the panel contains other useful controls: Encoder (see Section 4.4.1) and PLL (Frequency Multiplier, see Section 4.4.2) setup. The Frame Burst bar controls how many frames are acquired during a frame burst when the Acquisition Start Trigger is enabled (see Section 6.1).

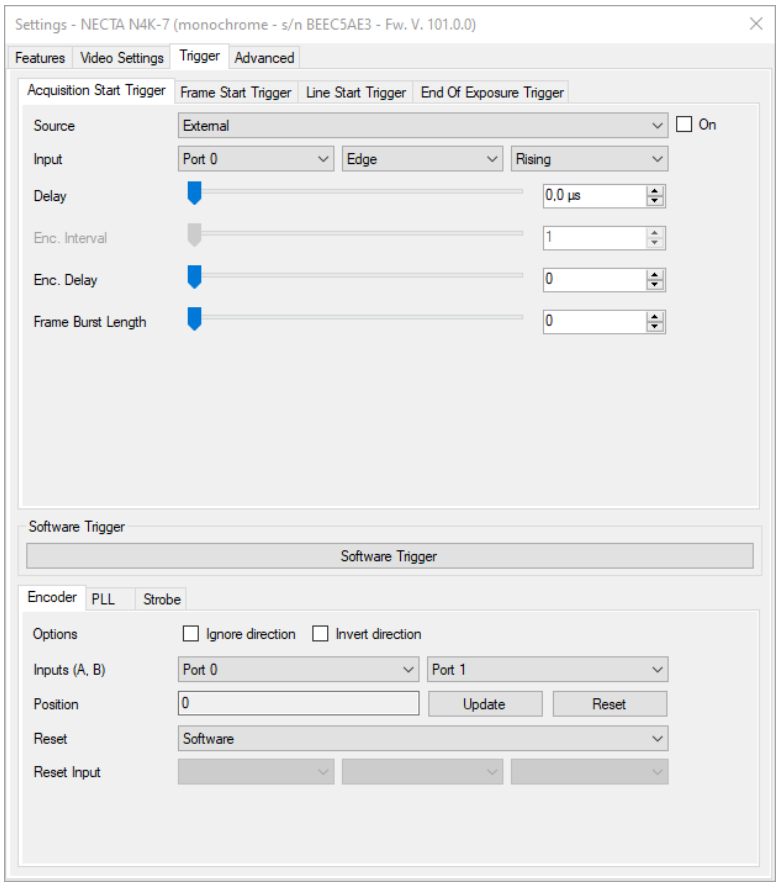


Figure 9.4: *Trigger tab showing trigger modules*

9.3.4 Advanced features

The advanced feature tab (Figure 9.5) contains additional options to control the USB channel. Changing these values may prevent the camera from working correctly.

The Bandwidth Limit control allows you to increase the upper bandwidth limit used by USB 3.2 Gen 1x1 connection. The maximum value you can set is 48 KiB per micro-frame (see Section 8.7).

Note

Not all host controllers support the maximum theorical speed. For this reason the camera has a default startup limit of 32 KiB per micro-frame you can override according to your setup.

Note

The checkbox *Use Bulk Endpoint* switches the USB interface between isochronous and bulk endpoint (see Section 2.5).

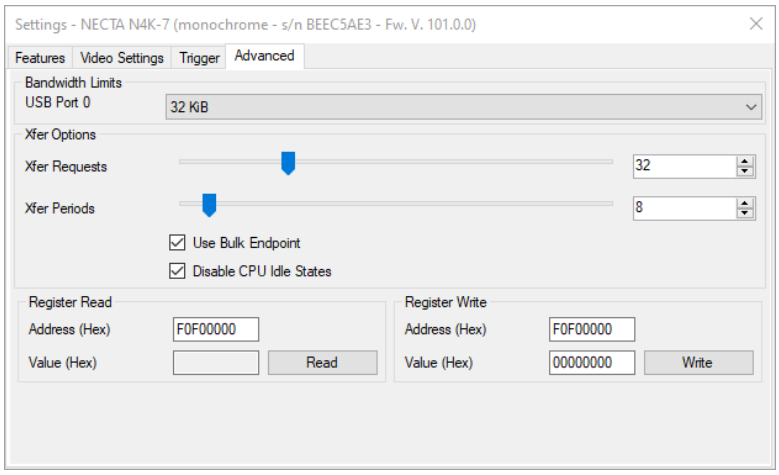


Figure 9.5: *Advanced settings tab*

9.4 SAVE SEQUENCE PANEL

The *Save Sequence Panel* allows to save on the hard drive a sequence of frames captured by the camera, with specific parameters (Figure 9.6).

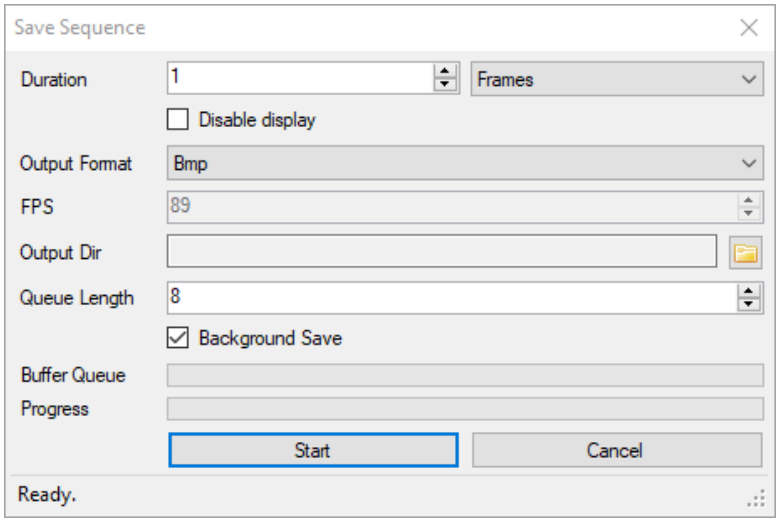


Figure 9.6: *Save Sequence panel*

The *Duration* entry allows to specify the sequence length either in number of frames or in seconds. When the *Output Format* is a still image format, such as `Bmp`, every frame will be saved in a new file. If the output format is set to `AVI`, a single video file is produced. User may specify the video frame rate in the *FPS* entry.

The *Queue Length* control determines the number of frames pre-allocated in RAM during capture. If the *Background Save* checkbox is unchecked, frames will be saved when the RAM queue is full. Otherwise, frames will be automatically saved on the hard drive.

> **Note**
>
> High Queue Length values may be incompatible with some hardware configurations.

9.5 DISPLAY FRAMES PANEL

When clicking on the “Display Frames” button on the Player toolbar, a panel will open. This panel allow the user to set the maximum frame-rate to be displayed on the screen. It’s possible to disable rendering on main UI by un-checking the “Display” box. (Figure 9.7)

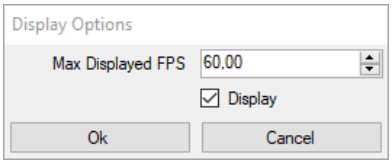


Figure 9.7: *Display Frames panel*

9.6 CAMERA MENU

On the rightmost part of the toolbar you can find the Camera menu (Figure 9.8) containing camera-specific features.

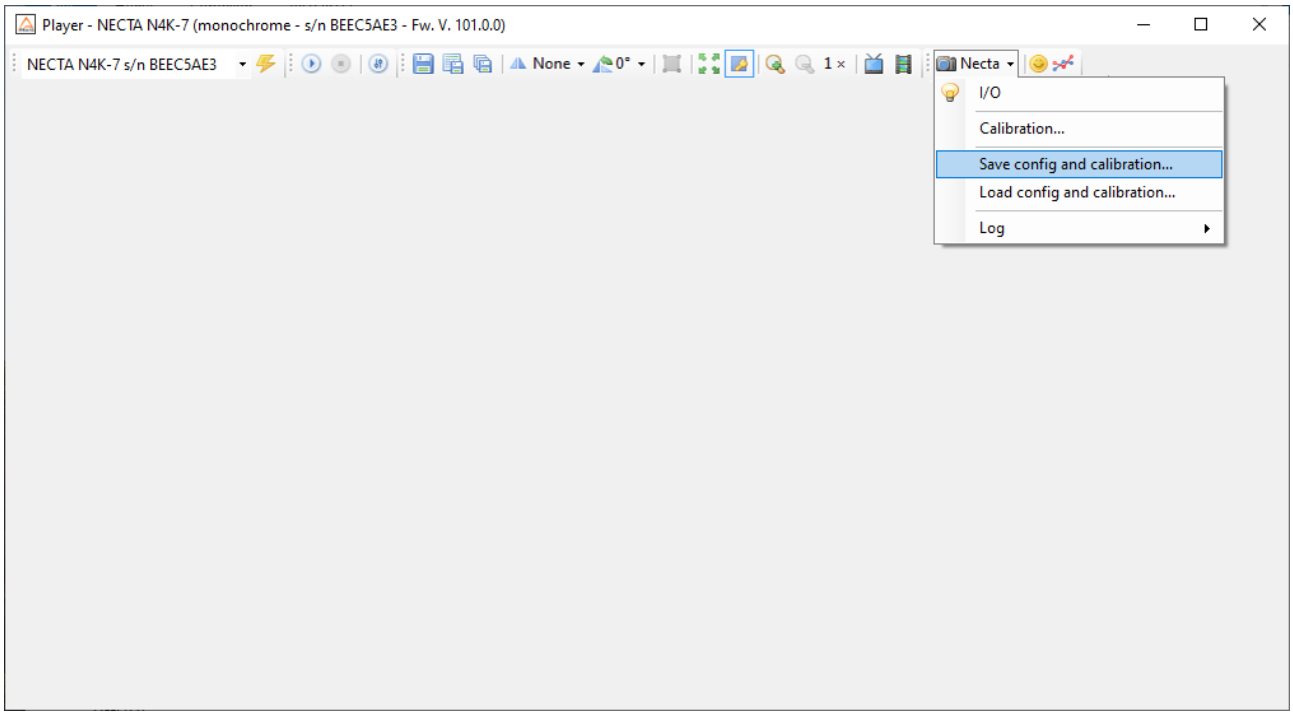


Figure 9.8: *NECTA specific features menu*

9.6.1 Saving and Exporting Configuration

The Config menu allows to save and load camera configuration to and from the camera internal flash memory. Configuration can be exported and imported to and from xml files as well. See Section 8.11 for further information.

9.6.2 I/O panel

The I/O panel (Figure 9.9) allows to control the I/O ports (see Chapter 4 for details).

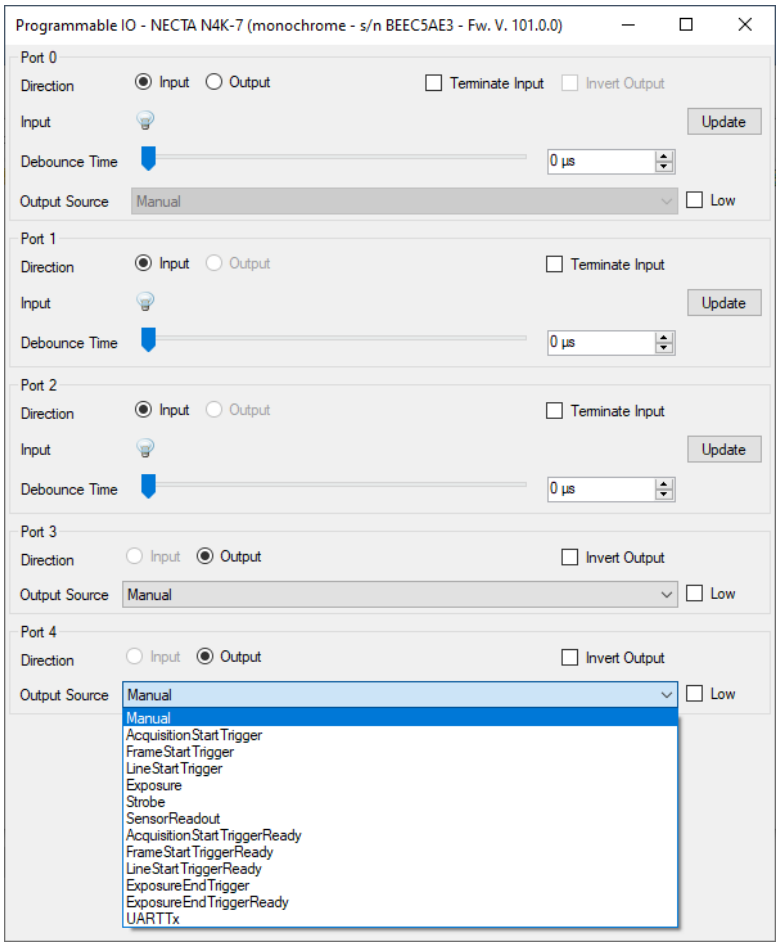


Figure 9.9: *I/O control panel*

Direction for the bidirectional port (Port 0) can be chosen by using the radio button on the top of the panel (Input / Output). Input ports allow enabling input termination (see Section 4.3.1), reading input status through the "Update" button and selecting input debounce time. The "Output Source" drop-down list allows to select how to route internal logical signals to the output pins.

9.6.3 Calibration

NECTA is able to automatically correct non-uniformities introduced by image sensor, lens and lighting system; these non-uniformities are called FPN. The set of techniques used to improve the quality of the acquired images is generally called Shading correction or FFC, and is a combination of offset and gain compensations. The calibration procedure combines three corrections (black level offset, DSNU and PRNU), involved in different phases of the frame acquisition process. The calibration aim will be de-

scribed in Section 8.2. User can find a detailed guide for calibration procedure by pressing Help button.

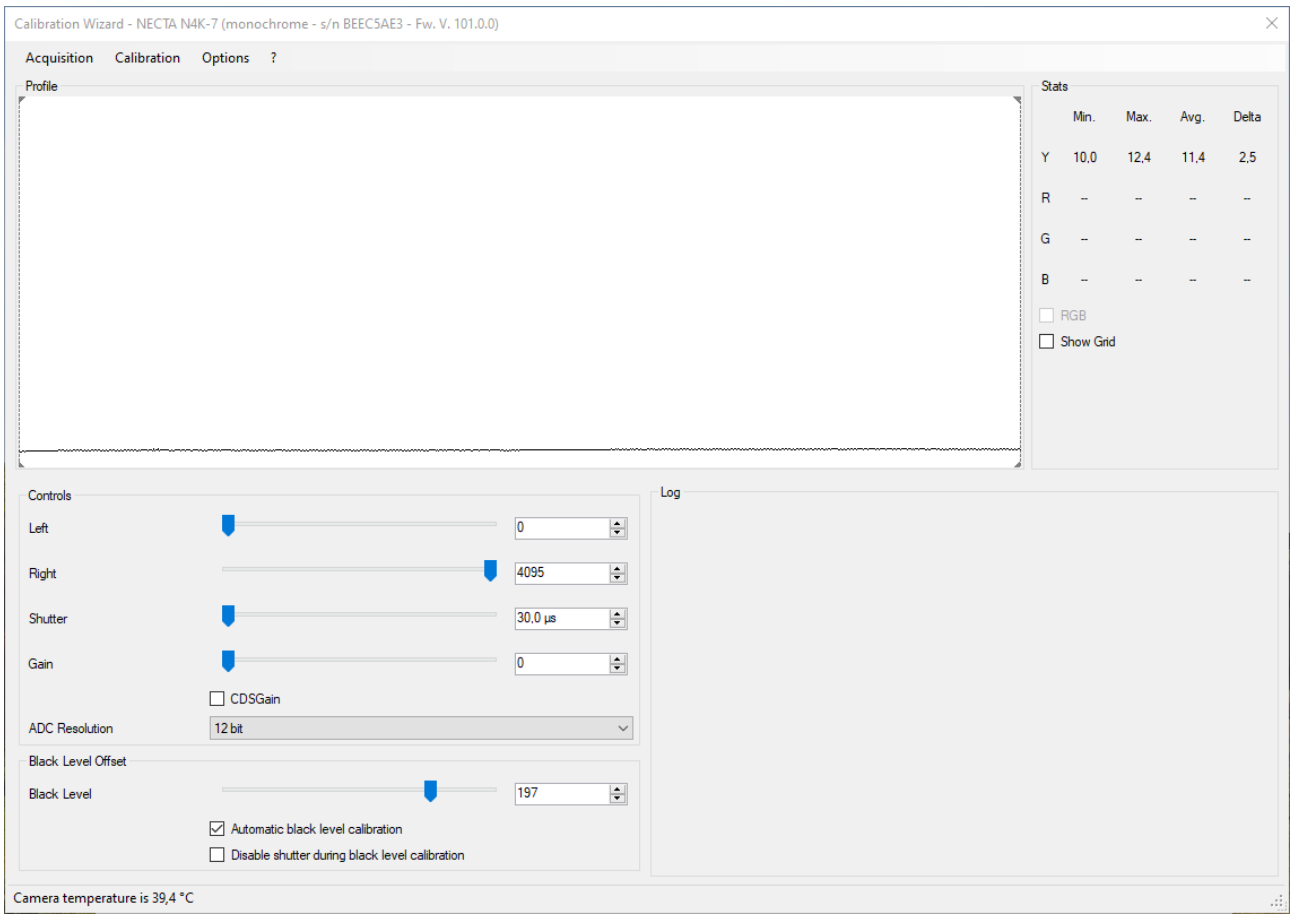


Figure 9.10: *Calibration window*

9.6.4 LUT Editor

A LUT is a function that you can apply to each frame color channel, digitally modifying the camera color response. This function is applied directly in hardware by the camera (see Section 8.10.13). This simple GUI allows you to edit and apply your custom LUT.

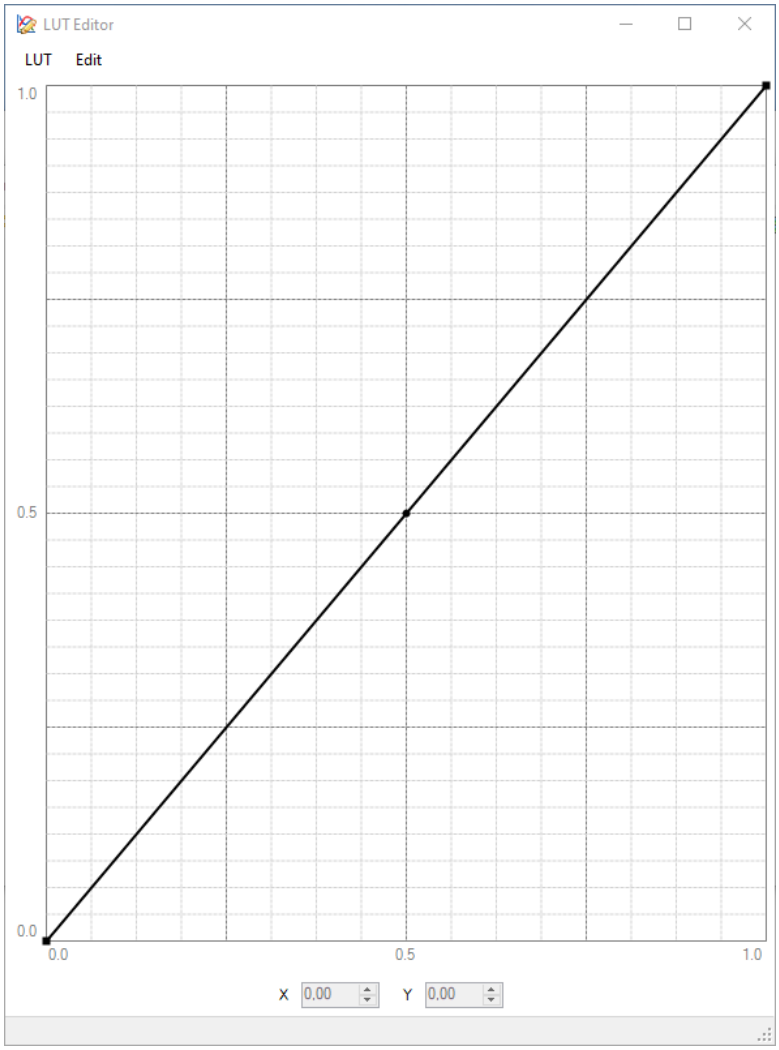


Figure 9.11: *LUT editor GUI*

10 Warranty

Unless otherwise agreed, NECTA cameras are guaranteed for 24 months from date of purchase. The warranty covers any defects due to production and conformity, not due to misuse, tampering or negligence by the user. At discretion of the Alkeria SRL technical service, faulty device may be either repaired or replaced with another one with same characteristics.

10.1 RMA

Before sending a defective camera for repair, you should contact your dealer and follow the procedures for fault verification and eventually returning the camera. If your country is not served by an Alkeria SRL distributor, you must apply directly to the Alkeria SRL technical service sending an e-mail to rma@alkeria.com , indicating your personal details, model and serial number of the camera, along with a brief description of the fault. You will be contacted by Alkeria SRL for inspection and/or receive instructions to send the camera back for repairs. Any unauthorized material will be returned to the sender.

11 Firmware Update

Alkeria SRL development team is constantly working to add new features to the NECTA family.

When applying for MaestroUSB3 free delivery, NECTA customers may be included in an SDK and firmware update notification mailing list. Subscribed users will be notified about new firmware versions available for their devices and will be able to update them, if interested.

Updates can be installed through the USB 3.2 Gen 1x1 connection using a PC running Windows 10 or Linux.

12

EMC certification and compliance

12.1 CE CONFORMITY

CE

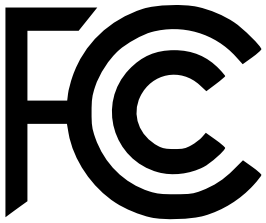
NECTA cameras described in this manual comply with the requirements of the EC EMC Directive 2014/30/EU of February 26, 2014.

Testing standards:

- CEI EN 55032:2015 - CISPR32:2015: Electromagnetic compatibility of multimedia equipment - Emission requirements.
- CEI EN 61000-6-4:2007 + A1:2013 - IEC 61000-6-4:2006 + A1:2010: Generic standards - Emission standard for industrial environments (Class A device).
- CEI EN 55024:2013 + A1:2016 - CISPR 24:2010 + A1:2015: Information technology equipment - Immunity characteristics - Limits and methods of measurement.
- CEI EN 61000-4-2:2011 - IEC 61000-4-2:2008: Testing and measurement techniques - Electrostatic discharge immunity test.
- CEI EN 61000-4-3:2007 + A1:2009 + A2:2011 - IEC 61000-4-3:2006 + A1:2007 +A2:2010: Testing and measurement techniques - Radiated, radio-frequency, electromagnetic field immunity test.

- CEI EN 61000-4-4:2013 - IEC 61000-4-4:2012: Testing and measurement techniques - Electrical fast transient/burst immunity test.
- CEI EN 61000-4-6:2014 + AC:2015 - IEC 61000-4-6:2013: Testing and measurement techniques - Immunity to conducted disturbances, induced by radio-frequency fields.

12.2 FCC CONFORMITY - CLASS A (USA)



This device complies with Part 15 of the FCC Rules. Operation is subject to the following two conditions: (1) this device may not cause harmful interference, and (2) this device must accept any interference received, including interference that may cause undesired operation.

12.3 ICES-003 (CANADA)

Cet appareil numérique respecte les limites bruits radioélectriques applicables aux appareils numériques de Classe A prescrites dans la norme sur le matériel brouilleur: “Appareils Numériques”, NMB-003 édictée par le Ministre Canadian des Communications.

“This digital apparatus does not exceed the Class A limits for radio noise emissions from digital apparatus set out in the interference-causing equipment standard entitled “Digital Apparatus,” ICES-003 of the Canadian Department of Communications.”

12.4 LIMITATION OF LIABILITY

The information in this manual is subject to change without notice. Alkeria SRL reserves the right to make improvements or changes to the devices and operating instructions at any time without prior notice. In no event shall Alkeria SRL be liable to the purchaser for any indirect, special or consequential damages or lost profits arising out of or relating to Alkeria SRL ’s products, or the performance or a breach thereof, even if Alkeria SRL has been advised of the possibility thereof. Alkeria SRL ’s liability, if any, to the purchaser or to the customer of the purchaser shall in no event exceed the total amounts paid to Alkeria SRL by the purchaser for a defective product. Some states do not permit the exclusion of incidental or consequential damages, and in those states the foregoing limitations may not apply. The warranties here give you specific legal rights, and you may have other legal rights which vary from state to state.

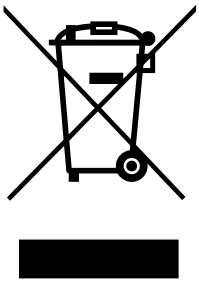
12.5 ROHS CONFORMITY

The NECTA cameras comply with the requirements of the RoHS (Restriction of Hazardous Substances) Directive 2011/65/EU.

RoHS Compliant

12.6 INFORMATION FOR USERS

Alkeria SRL is committed to meeting the requirements of the European Union (EU) Waste Electrical and Electronic Equipment (WEEE) Directive. This Directive requires producers of electrical and electronic equipment to finance the take back, for reuse or recycling, of their products placed on the EU market after August 13, 2005. Alkeria SRL products are within the scope of the Directive and are labelled with a crossed-out "wheelie-bin" symbol, as required by the Directive. It indicates that the product was placed on the market after August 13, 2005 and that end users should segregate the product from other waste at end-of- life. All Alkeria SRL cameras have been manufactured after the 31st of August 2005.



This symbol on Alkeria SRL product or on its packaging means that it should not be disposed of with your other household waste. It is your responsibility to dispose of your waste equipment separately from the municipal waste stream. The correct disposal of your old appliance will help prevent potential negative consequences for the environment and human health.

List of Acronyms

CMOS	Complementary MOS
ADC	Analog to Digital Converter
PLL	Phase Locked Loop
LMR	Light Meter ROI
ROI	Region Of Interest
LUT	Look-Up Table
DSNU	Dark Signal Non-Uniformity
PRNU	Photo Response Non-Uniformity
FPN	Fixed Pattern Noise
FFC	Flat-Field Correction
OSG	Overall System Gain
FPGA	Field-Programmable Gate Array
ROI	Region-Of-Interest
API	Application Programming Interface

Example Code

Via Giuntini 25, int. 36
56021 Cascina (PI) – ITALY
www.alkeria.com