Alkeria
IMAGINACTION

ARIA series

Board level USB3 camera

USER GUIDE

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

Release 2.0 - December, 2021

Summary

1 Getting started **7**
1.1 Requirements 7
1.2 Software download 8
1.3 Connections 11
1.4 First Start 12
1.4.1 Status LED 12
1.4.2 Run Alkeria player 12
1.4.3 How to get a good image 14

2 Introducing ARIA Cameras **15**
2.1 ARIA 16
2.2 ARIA family 16
2.2.1 Detailed Specifications 18
2.3 Optional accessories 23
2.4 Ordering Informations 23
2.5 USB 3.2 Gen 1x1 SuperSpeed interface 24
2.5.1 Isochronous endpoint 24
2.5.2 Bulk endpoint 24
2.5.3 USB 3.2 Gen 1x1 controller cards 25
2.6 Installation requirements 25
2.7 MaestroUSB3 SDK 25
2.7.1 Software License 26
2.8 Safety and Precautions 26

3 Hardware Description **28**
3.1 Sensors 28
3.1.1 TeleDyne e2v Sensors 28
3.1.1.1 Monochrome Spectral response 28
3.1.1.2 Color Spectral response 29
3.1.2 SONY IMX Sensors 31
3.1.2.1 Monochrome Spectral response 31
3.1.2.2 Color Spectral response 32
3.2 Filters 33
3.3 Mechanical description 35

3.4 Adapters for standard lens 36
3.4.1 C-mount adapter 36
3.4.1.1 Maximum lens protrusion 37
3.4.2 S-mount adapter 38
3.4.2.1 Maximum lens protrusion 38
3.5 Back casing for ARIA cameras 40
3.5.1 Back casing with C-mount adapters 40
3.5.2 Back casing with S-mount adapters 41
3.6 Flange Focal Distance 42
3.7 Handling precautions 42
3.7.1 Temperature and humidity 42
3.7.2 Warranty limitations 42
3.7.3 Mechanical stress 43
3.7.4 Heat dissipation 43
3.7.5 Board level recommendations 43
3.7.6 Monitoring internal temperature 45
3.7.7 Automatic internal temperature safety mechanism 46
3.7.8 EMI/ESD precautions: noise and electrostatic discharge 47
3.8 Cleaning ARIA camera 47
3.8.1 Before starting 48
3.8.2 Cleaning ARIA camera housing 48
3.8.3 Optical path cleaning 48
3.8.4 Impurities 48
3.8.5 Filter cleaning 49
3.8.6 Sensor cleaning 50
3.8.7 Using compressed air 50
3.8.8 After cleaning procedure 50

4 Interfacing to the world 51
4.1 I/O module 52
4.1.1 Optoisolated input module structure (port 2) 52
4.1.1.1 Debouncing module 53
4.1.1.2 Input events 53
4.1.2 Optoisolated output module structure (port 3) 54
4.1.2.1 Examples of output module connection: Led drive 55
4.1.2.2 Example of Relay driving using OptoIsolated Output 55
4.1.3 Bidirectional module structure (port 0 and port 1) 57
4.1.4 I/O switching time 57
4.2 Using input signals 59
4.2.1 Encoder module 59
4.2.1.1 Encoder hysteresis 60
4.2.1.2 Direction reversal 61
4.2.1.3 Configuring the encoder module 61
4.2.1.4 Reading and resetting encoder position 62
4.2.2 Frequency Multiplier (Phase Locked Loop (PLL)) 62

4.2.3 Trigger manager . . . 64
4.3 Controlling output ports . . . 64
4.3.1 Polarity . . . 65
4.3.2 Software output . . . 66
4.3.3 Trigger Output . . . 66
4.3.3.1 Exposure . . . 66
4.3.3.2 Strobe . . . 66
4.3.3.3 Trigger . . . 67
4.3.3.4 Trigger ready . . . 67
4.3.4 Custom sources . . . 68
4.3.5 Input and output ports usage examples . . . 68
4.4 Serial Interface Module . . . 68

5 System Setup **71**
5.1 USB Setup . . . 71
5.1.1 How to choose a suitable USB cable . . . 71
5.1.2 Recommended USB interface . . . 72
5.2 Status LED . . . 73
5.3 When one ARIA is not enough... . . . 73

6 Trigger **74**
6.1 Trigger sources . . . 74
6.2 Acquisition start trigger . . . 75
6.3 Frame start trigger . . . 75
6.4 End-of-exposure trigger . . . 75
6.5 Trigger delay . . . 76
6.5.1 Encoder delay . . . 76
6.5.2 Trigger overrun . . . 76
6.6 Trigger configuration example . . . 77
6.6.1 External trigger frame acquisition . . . 77
6.6.2 Signal-driven exposure time . . . 78
6.6.3 Encoder synchronization . . . 79

7 The processing Chain **80**
7.1 Light Meter . . . 80
7.2 Black level offset . . . 81
7.3 Chunk data . . . 82
7.3.1 Frame number . . . 82
7.3.2 Time stamp . . . 83
7.3.2.1 Time stamp resolution . . . 83
7.3.3 Encoder position . . . 83
7.3.4 Input status . . . 83
7.3.5 Burst number . . . 83
7.3.6 Configuration example . . . 84
7.4 Frame Combiner . . . 84
7.4.1 Flush and AutoFlush . . . 84

7.4.2 Configuration example . . . 84

8 Capturing images **85**

8.1 Video modes . . . 85
8.1.1 Mode 0 (Normal) . . . 85
8.1.2 Mode 1 (RAW) . . . 85
8.2 Region of Interest (ROI) . . . 86
8.3 Binning . . . 87
8.4 Pixel format . . . 91
8.5 ADC resolution . . . 92
8.6 Frame rate and bandwidth . . . 93
8.6.1 Reserving bandwidth for isochronous channel . . . 93
8.6.2 Bandwidth limits for isochronous endpoint . . . 93
8.6.3 Bandwidth limit for bulk endpoint . . . 94
8.6.4 Packet size for isochronous endpoint . . . 95
8.6.4.1 Calculating the required bandwidth . . . 96
8.6.5 Packet size for bulk endpoint . . . 97
8.6.6 Connecting multiple devices to the same host . . . 97
8.6.7 Frame rate . . . 97
8.6.8 Preserve rates . . . 98
8.6.9 Maximizing frame rate . . . 99
8.7 Controls . . . 99
8.7.1 Brightness . . . 100
8.7.2 Contrast . . . 100
8.7.3 Color correction matrix (color models only) . . . 101
8.7.4 Hue (color models only) . . . 102
8.7.5 Saturation (color models only) . . . 103
8.7.6 White Balance (color models only) . . . 104
8.7.7 Gamma . . . 104
8.7.8 Shutter . . . 106
8.7.8.1 High resolution shutter . . . 107
8.7.8.2 Shutter resolution . . . 107
8.7.8.3 Shutter floor . . . 107
8.7.8.4 Automatic exposure . . . 107
8.7.9 Analog Gain . . . 108
8.7.10 Digital Gain . . . 109
8.7.11 User Look-Up Table (LUT)s . . . 110
8.7.12 Luma . . . 111
8.7.13 Time Stamp . . . 111
8.7.14 Flip and Rotate . . . 111
8.8 Saving device configuration . . . 112
8.8.1 Export and import XML . . . 112
8.8.2 User configuration (deprecated) . . . 113
8.9 Configuration Example . . . 113
8.10 Pattern Generator . . . 114

9 Alkeria player 115
9.1 Device Connection and Initialization 115
9.2 Toolbar buttons 116
9.3 Settings Panel 116
9.3.1 Features 116
9.3.2 Video Settings 117
9.3.3 Trigger 118
9.3.4 Advanced features 119
9.4 Save Sequence Panel 121
9.5 Display Frames Panel 121
9.6 Camera Menu 121
9.6.1 Saving and Exporting Configuration 122
9.6.2 I/O panel 122
9.6.3 Black Level Calibration (e2V models only) 123
9.6.4 LUT Editor 125

10 Profilometer extension for laser triangulation 127
10.1 Laser triangulation 127
10.2 Laser line position detection 128
10.2.1 Thresholds 128
10.2.2 Main Processing 129
10.3 Profile acquisition 129
10.3.1 Video Mode 0 129
10.3.2 Video Mode 1 130
10.3.3 Video Mode 2 130
10.4 Exploiting frame combiner with Profilometer 130
10.5 Player utilities 130
10.5.1 Live Profile 131
10.5.2 Profilometer Settings 131
10.6 Parameters set-up 131

11 Warranty 133
11.1 RMA 133

12 Firmware Update 134

13 EMC certification and compliance 135
13.1 CE conformity 135
13.2 FCC conformity - Class A (USA) 136
13.3 ICES-003 (Canada) 136
13.4 Limitation of liability 136
13.5 RoHS Conformity 136
13.6 Information for Users 137

List of Acronyms 138

Example Code 140

1 Getting started

1.1 REQUIREMENTS

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

- A ARIA 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;

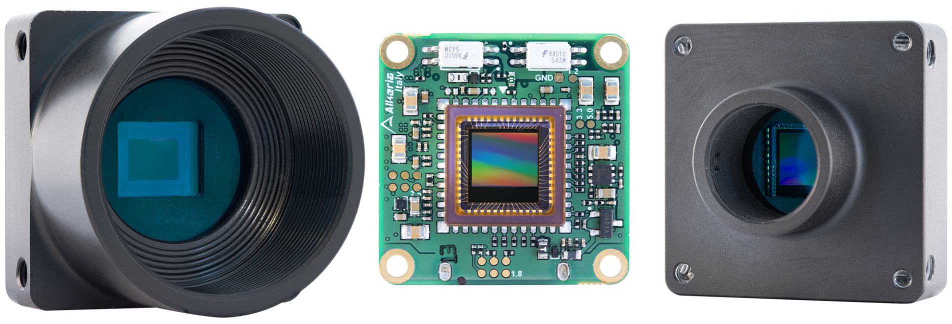


Figure 1.1: *ARIA camera*

(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)

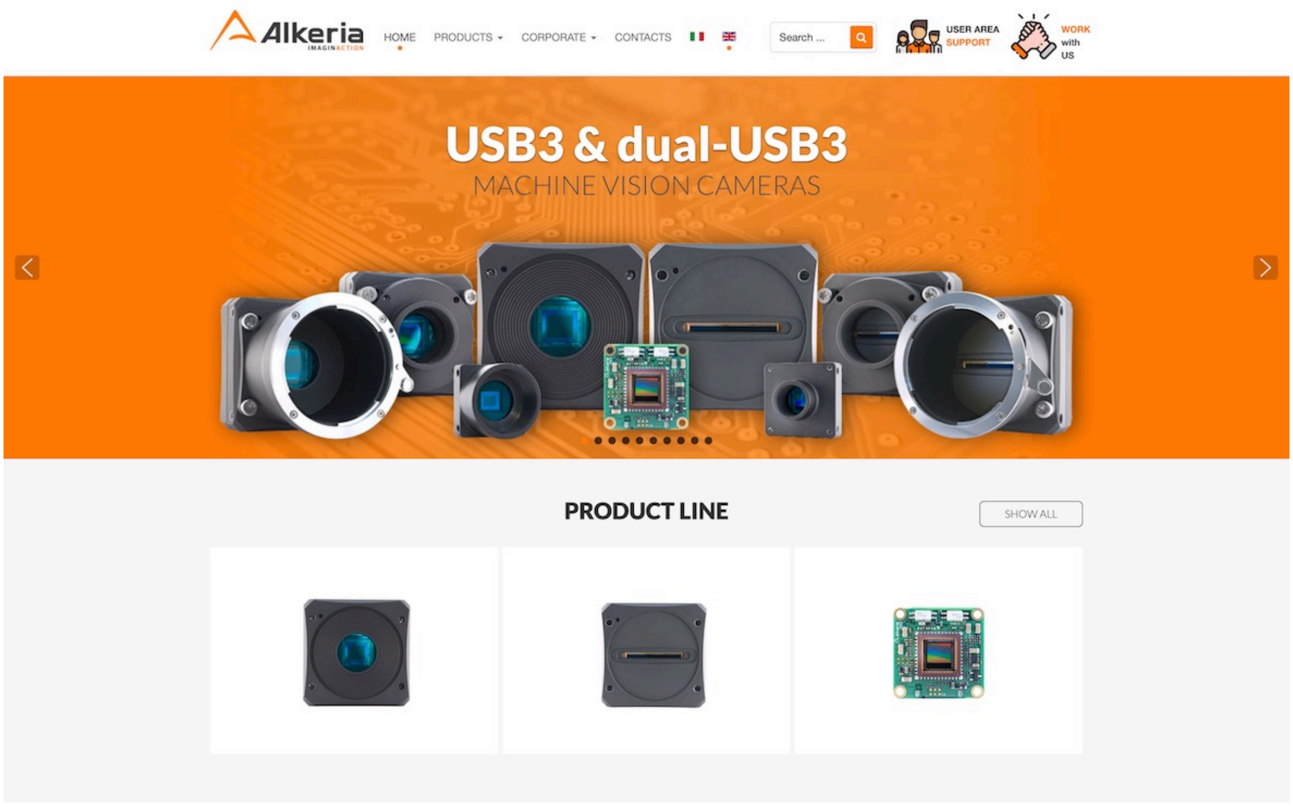


Figure 1.3: *Alkeria website*

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

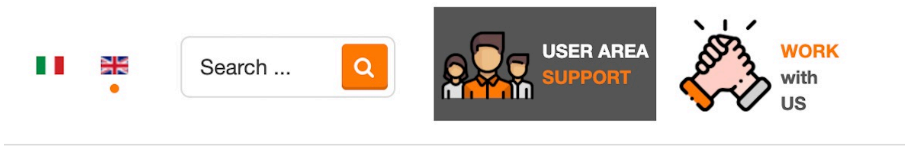


Figure 1.4: *User Area button*

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

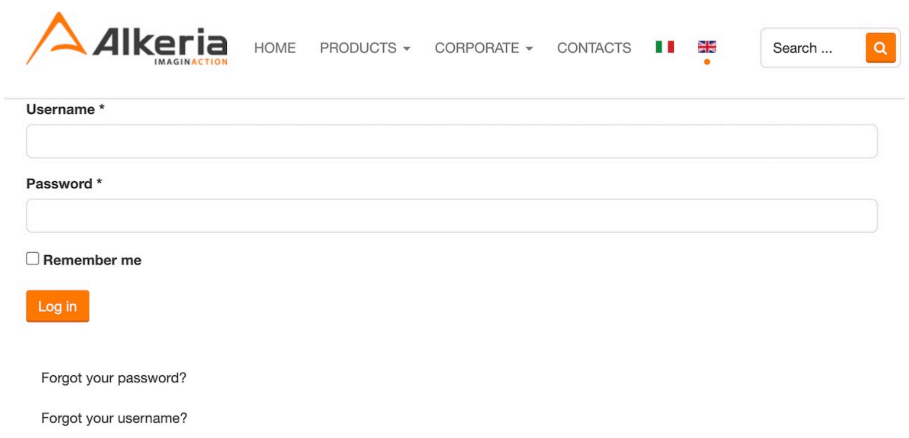


Figure 1.5: *User Area log in form*

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

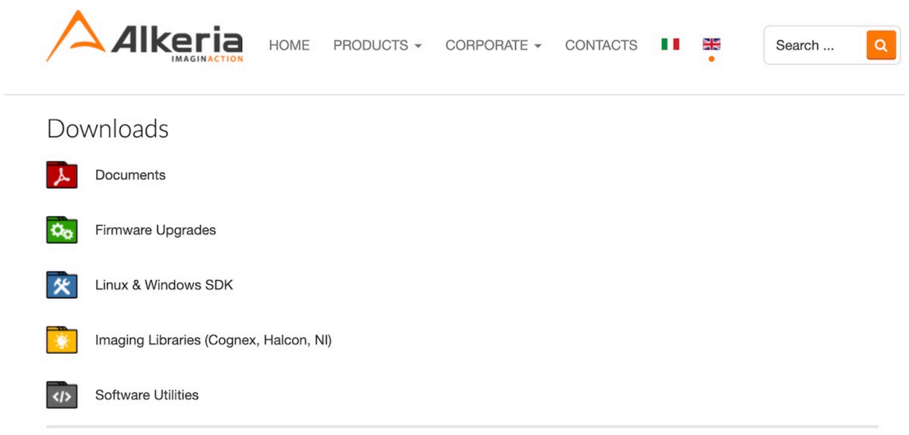
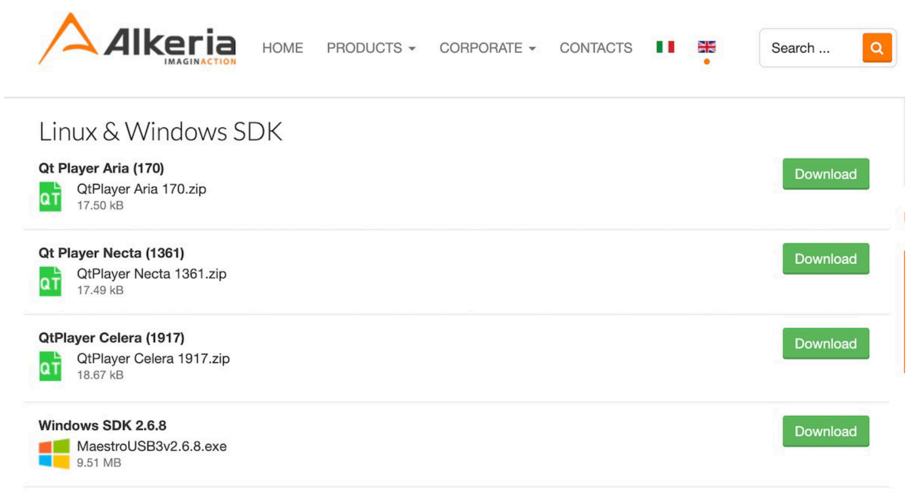


Figure 1.6: *User Area root folder*

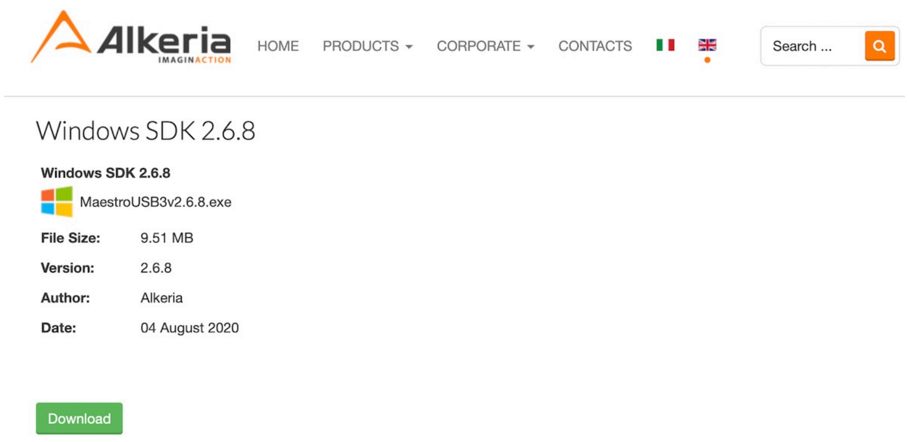
Step 5: Click on “Download” to download a file (fig. 1.7)



Linux & Windows SDK

Qt Player Aria (170)	QtPlayer Aria 170.zip 17.50 kB	Download
Qt Player Necta (1361)	QtPlayer Necta 1361.zip 17.49 kB	Download
QtPlayer Celera (1917)	QtPlayer Celera 1917.zip 18.67 kB	Download
Windows SDK 2.6.8	MaestroUSB3v2.6.8.exe 9.51 MB	Download

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



Windows SDK 2.6.8

Windows SDK 2.6.8
MaestroUSB3v2.6.8.exe

File Size:	9.51 MB
Version:	2.6.8
Author:	Alkeria
Date:	04 August 2020

Download

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.

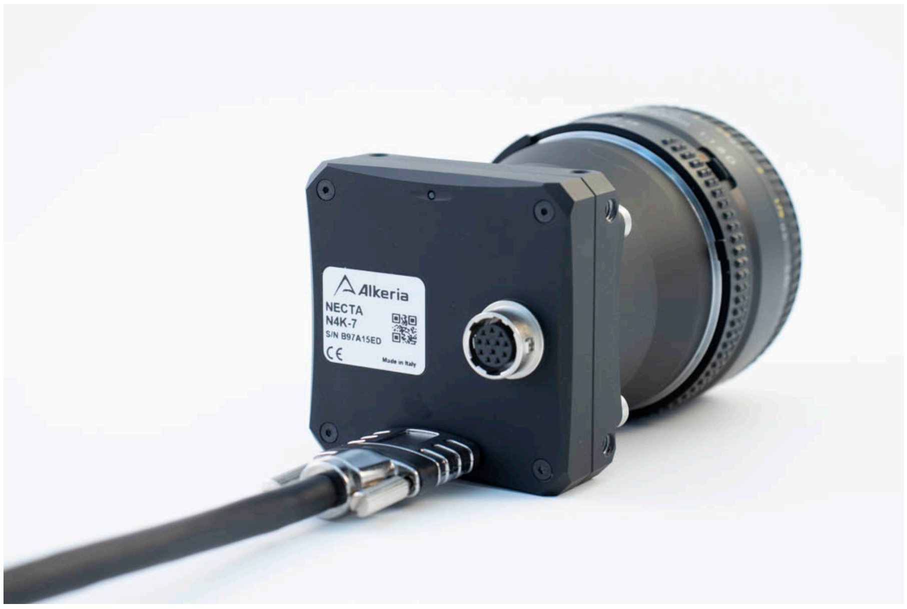
1.3 CONNECTIONS

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

ARIA 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.

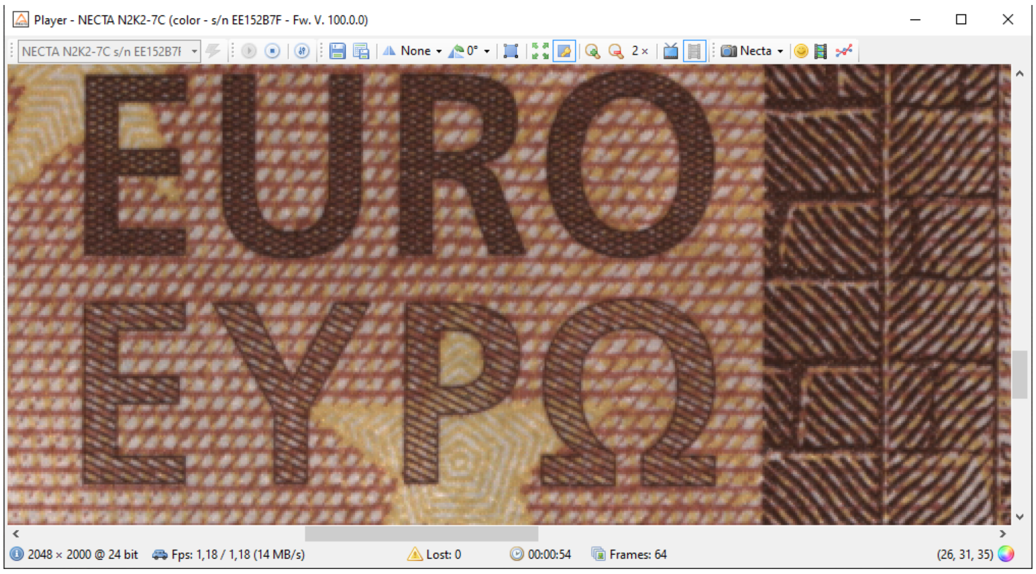
Different behaviours of the status LED may indicate that camera is malfunctioning. To a complete reference on the status LED codes, refer to Table 5.1.

1.4.2 Run Alkeria player

MaestroUSB3 comes with a user-friendly software application, allowing to explore the main features of ARIA 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.



Player - NECTA N2K2-7C (color - s/n EE152B7F - Fw. V. 100.0.0)

2048 × 2000 @ 24 bit   Fps: 1,18 / 1,18 (14 MB/s)   Lost: 0   00:00:54   Frames: 64   (26, 31, 35)

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 ARIA 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 ARIA 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.

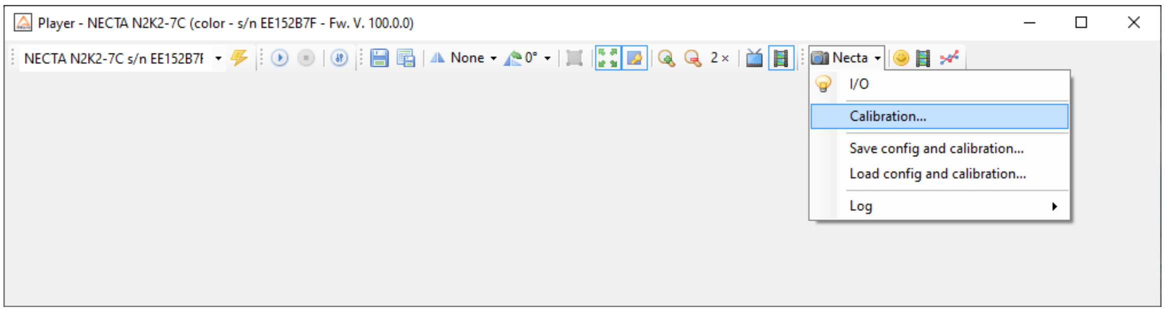

Player - NECTA N2K2-7C (color - s/n EE152B7F - Fw. V. 100.0.0)
NECTA N2K2-7C s/n EE152B7F
None
0°
2 x
Necta
I/O
Calibration...
Save config and calibration...
Load config and calibration...
Log

Figure 1.10: *Alkeria player Calibration Wizard*

2 Introducing ARIA Cameras

ARIA represents Alkeria's answer to the increasing demand of ultralight, ultracompact, versatile and easy to integrate board level cameras.

In the size of a coin and weighting just 5 grams, ARIA is configurable with a wide choice of sensors, bringing all benefits and features of Alkeria's cameras: on-board image pre-processing, flexibility for advanced application thanks to its I/O interface, great customization possibilities, USB3 performances and ease of use. Like Alkeria's higher end cameras, ARIA is equipped with USB3 interface, now a standard in digital imaging applications, due to its high performances and ease of use, eliminating the need for external power adapters.

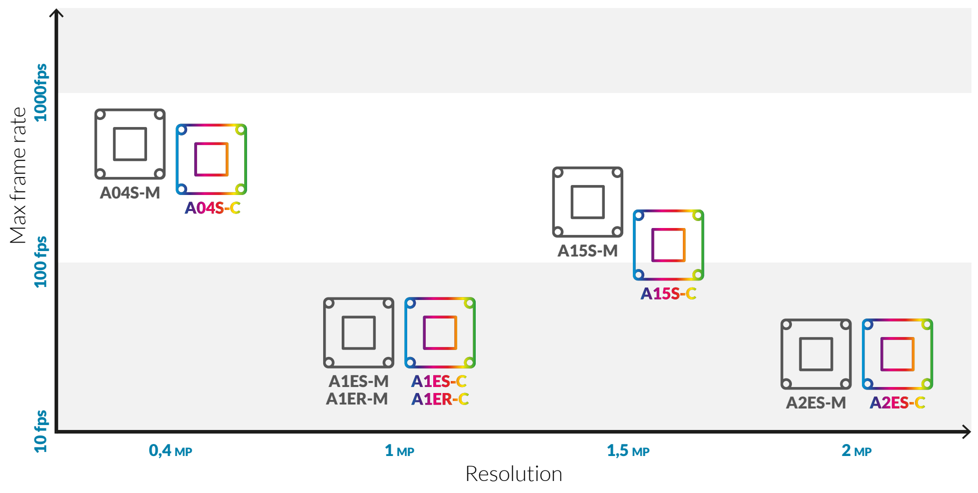


Figure 2.1: *ARIA cameras overview*

2.1 ARIA

ARIA is a family of very compact board level area scan cameras featuring USB 3.2 Gen 1x1 interface. USB3 interface allows to take full advantage of SONY Pregius IMX as well as e2v Sapphire and Ruby series sensors, ensuring high performances -up to and over 520 FPS- and a superior image quality.

ARIA is available with color, black and white or NIR-enhanced CMOS sensor and 8-10-12 bit Analog/Digital conversion system (depending on sensor); it comes with an easy-to-use set of software API, allowing developers to quickly produce fast and clean code both on Windows and Linux, supporting all major programming languages, such as Python, C#, VB.NET, C++ in Visual Studio or QT environment. Our Software Development Kit provided with ARIA cameras, is fully compatible with Cognex VisionPro, MVTech Halcon, NI LabView and other major machine vision software for industrial application.

ARIA 's I/O interface includes 2 I/O (direct encoder interface, RS232, LVTTL) and separate opto-isolated IN and OUT, in order to ensure a great versatility in complex systems. Like all other Alkeria's cameras, ARIA is easy to customize -for enterprise production volumes- with customer specific operations porting and on-board pre-processing.

ARIA is available in board level format, with C or S-mount lens adapter only or even with a complete housing.

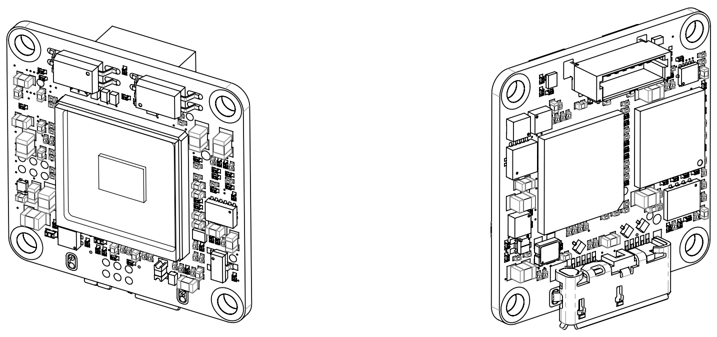


Figure 2.2: *ARIA camera*

2.2 ARIA FAMILY

ARIA cameras use high performance TeleDyne e2v and SONY matrix CMOS sensors. The models in the family differ in size, layout and number of sensor pixels.

The table 2.1 shows all models with their basic features. For detailed specifications please refer to Paragraph 2.2.1 where all available sensor models are listed. All cameras are directly powered by the computer through the USB connector.

Model Name	Sensor	Resolution (H x W)	Chroma
ARIA AE1S-M	E2V EV76C560	1288 x 1032 (1.31 MP)	Mono
ARIA AE1S-C	E2V EV76C560	1288 x 1032 (1.31 MP)	Color
ARIA AE2S-M	E2V EV76C570	1608 x 1208 (1.92 MP)	Mono
ARIA AE2S-C	E2V EV76C570	1608 x 1208 (1.92 MP)	Color
ARIA AE1R-M	E2V EV76C661	1288 x 1032 (1.31 MP)	Mono (NIR)
ARIA AE1R-C	E2V EV76C661	1288 x 1032 (1.31 MP)	Color
ARIA A04S-M	SONY IMX287	728 x 544 (0.39 MP)	Mono
ARIA A04S-C	SONY IMX287	728 x 544 (0.39 MP)	Color
ARIA A15S-M	SONY IMX273	1456 x 1088 (1.58 MP)	Mono
ARIA A15S-C	SONY IMX273	1456 x 1088 (1.58 MP)	Color

Table 2.1: *ARIA cameras family overview*

2.2.1 Detailed Specifications

Specification	ARIA A1ES-M	ARIA A1ES-C
Sensor model	e2V EV76C560	
Sensor type	Mono	Color
Sensor resolution	1288(W) x 1032(H) (1.31 MP)	
Sensor technology	CMOS, global shutter	
Sensor format	1/1.8”	
Pixel size	5.3 µm × 5.3 µm	
Max. frame rate	62 fps	
A/D Conversion	10 bits	
Synchronization	External trigger, software trigger	
Controls	Shutter, gain, frame rate, brightness, contrast, LUT, gamma, saturation, trigger	
Shutter control	16 µs ÷ 0.9 s (1 µs steps)	
Binning factor	2x horizontal, 2x vertical	
Readout	Normal, AOI, binning, frame combiner	
Power supply	1.2 W max, powered by USB 3.2 Gen 1x1	
Inputs/outputs	1 in, 1 out, 2 I/O (direct encoder interface, RS232 with LVTTL levels, LVTTL)	
Formats	Board level | C-mount only | S-mount only | Cased with C-mount or S-mount	
Interface	USB 3.2 Gen 1x1	
Pixel formats	RAW8, RAW16, MONO8, MONO16	RAW16, MONO16, YUV4:2:2, RGB24
Weight	5 g (board level format)	
Size (without adapters)	26.4 mm x 26.4 mm x 7.8 mm	
Size (with C-mount)	29 mm x 29 mm x 24.6 mm	
Size (with S-mount)	29 mm x 29 mm x 19.6 mm	
Size (cased with C-mount)	29 mm x 29 mm x 25.7 mm	
Size (cased with S-mount)	29 mm x 29 mm x 20.7 mm	
Conformity	RoHS	
Operating temperature	0 ÷ 50 °C (referred to housing)	
Software	MaestroUSB3	
Link	`www.alkeria.com/products/ARIA-series`	
Warranty	24 months	

Table 2.2: *ARIA A1ES-M / A1ES-C Specifications*

Specification	**ARIA A2ES-M**	**ARIA A2ES-C**
Sensor model	e2V EV76C570	
Sensor type	Mono	Color
Sensor resolution	1608(W) x 1208(H) (1.92 MP)	
Sensor technology	CMOS, global shutter	
Sensor format	1/1.8”	
Pixel size	4.5 µm × 4.5 µm	
Max. frame rate	51 fps	
A/D Conversion	10 bits	
Synchronization	External trigger, software trigger	
Controls	Shutter, gain, frame rate, brightness, contrast, LUT, gamma, saturation, trigger	
Shutter control	16 µs ÷ 0.9 s (1 µs steps)	
Binning factor	2x horizontal, 2x vertical	
Readout	Normal, AOI, binning, frame combiner	
Power supply	1.2 W max, powered by USB 3.2 Gen 1x1	
Inputs/outputs	1 in, 1 out, 2 I/O (direct encoder interface, RS232 with LVTTL levels, LVTTL)	
Formats	Board level | C-mount only | S-mount only | Cased with C-mount or S-mount	
Interface	USB 3.2 Gen 1x1	
Pixel formats	RAW8, RAW16, MONO8, MONO16	RAW16, MONO16, YUV4:2:2, RGB24
Weight	5 g (board level format)	
Size (without adapters)	26.4 mm x 26.4 mm x 7.8 mm	
Size (with C-mount)	29 mm x 29 mm x 24.6 mm	
Size (with S-mount)	29 mm x 29 mm x 19.6 mm	
Size (cased with C-mount)	29 mm x 29 mm x 25.7 mm	
Size (cased with S-mount)	29 mm x 29 mm x 20.7 mm	
Conformity	RoHS	
Operating temperature	0 ÷ 50 °C (referred to housing)	
Software	MaestroUSB3	
Link	`www.alkeria.com/products/ARIA-series`	
Warranty	24 months	

Table 2.3: *ARIA A2ES-M / A2ES-C Specifications*

Specification	**ARIA A1ER-M**	**ARIA A1ER-C**
Sensor model	e2V EV76C661	
Sensor type	Mono (NIR)	Color
Sensor resolution	1288(W) x 1032(H) (1.31 MP)	
Sensor technology	CMOS, global shutter	
Sensor format	1/1.8”	
Pixel size	5.3 µm × 5.3 µm	
Max. frame rate	62 fps	
A/D Conversion	10 bits	
Synchronization	External trigger, software trigger	
Controls	Shutter, gain, frame rate, brightness, contrast, LUT, gamma, saturation, trigger	
Shutter control	16 µs ÷ 0.9 s (1 µs steps)	
Binning factor	2x horizontal, 2x vertical	
Readout	Normal, AOI, binning, frame combiner	
Power supply	1.2 W max, powered by USB 3.2 Gen 1x1	
Inputs/outputs	1 in, 1 out, 2 I/O (direct encoder interface, RS232 with LVTTL levels, LVTTL)	
Formats	Board level | C-mount only | S-mount only | Cased with C-mount or S-mount	
Interface	USB 3.2 Gen 1x1	
Pixel formats	RAW8, RAW16, MONO8, MONO16	RAW16, MONO16, YUV4:2:2, RGB24
Weight	5 g (board level format)	
Size (without adapters)	26.4 mm x 26.4 mm x 7.8 mm	
Size (with C-mount)	29 mm x 29 mm x 24.6 mm	
Size (with S-mount)	29 mm x 29 mm x 19.6 mm	
Size (cased with C-mount)	29 mm x 29 mm x 25.7 mm	
Size (cased with S-mount)	29 mm x 29 mm x 20.7 mm	
Conformity	RoHS	
Operating temperature	0 ÷ 50 °C (referred to housing)	
Software	MaestroUSB3	
Link	`www.alkeria.com/products/ARIA-series`	
Warranty	24 months	

Table 2.4: *ARIA A1ER-M / A1ER-C Specifications*

Specification	**ARIA A04S-M**	**ARIA A04S-C**
Sensor model	SONY IMX287	
Sensor type	Mono	Color
Sensor resolution	728(W) x 544(H) (0.39 MP)	
Sensor technology	CMOS, global shutter	
Sensor format	1/2.9”	
Pixel size	6.9 µm × 6.9 µm	
Max. frame rate	522 fps	458 fps
A/D Conversion	8, 10, 12 bits	
Synchronization	External trigger, software trigger	
Controls	Shutter, gain, frame rate, brightness, contrast, LUT, gamma, saturation, trigger	
Shutter control	18 µs ÷ 5 s (1 µs steps)	
Binning factor	2x horizontal, 2x vertical	
Readout	Normal, AOI, binning, frame combiner	
Power supply	2 W max, powered by USB 3.2 Gen 1x1	
Inputs/outputs	1 in, 1 out, 2 I/O (direct encoder interface, RS232 with LVTTL levels, LVTTL)	
Formats	Board level | C-mount only | S-mount only | Cased with C-mount or S-mount	
Interface	USB 3.2 Gen 1x1	
Pixel formats	RAW8, RAW16, MONO8, MONO16	RAW16, MONO16, YUV4:2:2, RGB24
Weight	5 g (board level format)	
Size (without adapters)	26.4 mm x 26.4 mm x 7.8 mm	
Size (with C-mount)	29 mm x 29 mm x 24.6 mm	
Size (with S-mount)	29 mm x 29 mm x 19.6 mm	
Size (cased with C-mount)	29 mm x 29 mm x 25.7 mm	
Size (cased with S-mount)	29 mm x 29 mm x 20.7 mm	
Conformity	RoHS	
Operating temperature	0 ÷ 50 °C (referred to housing)	
Software	MaestroUSB3	
Link	`www.alkeria.com/products/ARIA-series`	
Warranty	24 months	

Table 2.5: *ARIA A04S-M / A04S-C Specifications*

Specification	ARIA A15S-M	ARIA A15S-C
Sensor model	SONY IMX273	
Sensor type	Mono	Color
Sensor resolution	1456(W) x 1088(H) (1.58 MP)	
Sensor technology	CMOS, global shutter	
Sensor format	1/2.9”	
Pixel size	3.45 μm × 3.45 μm	
Max. frame rate	237 fps	118 fps
A/D Conversion	8, 10, 12 bits	
Synchronization	External trigger, software trigger	
Controls	Shutter, gain, frame rate, brightness, contrast, LUT, gamma, saturation, trigger	
Shutter control	18 μs ÷ 5 s	22 μs ÷ 5 s
Binning factor	2x horizontal, 2x vertical	
Readout	Normal, AOI, binning, frame combiner	
Power supply	2 W max, powered by USB 3.2 Gen 1x1	
Inputs/outputs	1 in, 1 out, 2 I/O (direct encoder interface, RS232 with LVTTL levels, LVTTL)	
Formats	Board level | C-mount only | S-mount only | Cased with C-mount or S-mount	
Interface	USB 3.2 Gen 1x1	
Pixel formats	RAW8, RAW16, MONO8, MONO16	RAW16, MONO16, YUV4:2:2, RGB24
Weight	5 g (board level format)	
Size (without adapters)	26.4 mm x 26.4 mm x 7.8 mm	
Size (with C-mount)	29 mm x 29 mm x 24.6 mm	
Size (with S-mount)	29 mm x 29 mm x 19.6 mm	
Size (cased with C-mount)	29 mm x 29 mm x 25.7 mm	
Size (cased with S-mount)	29 mm x 29 mm x 20.7 mm	
Conformity	RoHS	
Operating temperature	0 ÷ 50 °C (referred to housing)	
Software	MaestroUSB3	
Link	`www.alkeria.com/products/ARIA-series`	
Warranty	24 months	

Table 2.6: *ARIA A15S-M / A15S-C Specifications*

2.3 OPTIONAL ACCESSORIES

ARIA cameras can be completed with the following accessories:

- AAD-C adapter to C-mount lens (see Section 3.4.1)
- AAD-S adapter to S-mount lens (see Section 3.4.2)
- AIO-03 cable kit for I/O interconnection – 0.3m long available (see Section 4).
- Back casing for ARIA camera for both C-mount and S-mount lens adapters (see Section 3.5)

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.

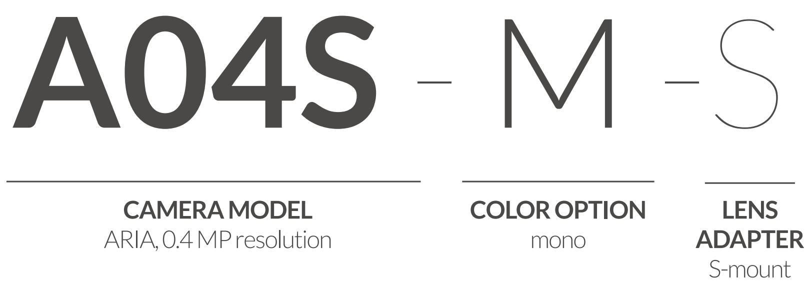


Figure 2.3: *ARIA Ordering information*

2.5 USB 3.2 GEN 1X1 SUPERSPEED INTERFACE

ARIA 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 ARIA 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 ARIA 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.

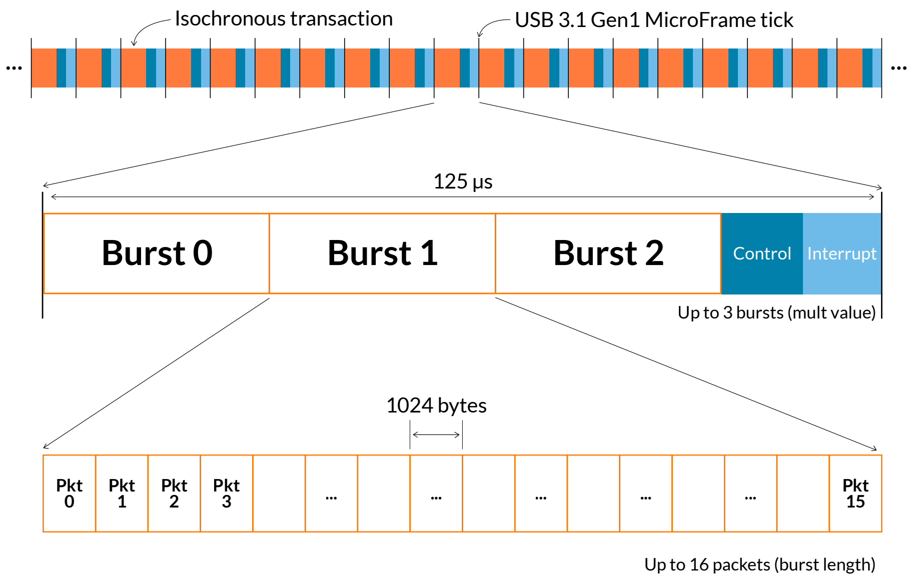


Figure 2.4: *USB 3.2 Gen 1x1 Isochronous transactions structure*

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 ARIA 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 ARIA 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 ARIA 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 ARIA performance, the controlling PC must meet the minimum requirements listed in Table 2.7 for single camera applications and in Table 2.8 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 2.7: *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 2.8: *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 ARIA 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 ARIA 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 ARIA 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

ARIA 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 3.7.

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 7-pin Molex 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.

ARIA 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 ARIA 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 ARIA 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

ARIA cameras employ last-generation CMOS sensors manufactured by TeleDyne e2v and SONY. Please refer to the table above (see par. 1.2) to identify the sensor used in each camera model, indicating the resolution and size of the pixel cell.

3.1.1 TeleDyne e2v Sensors

3.1.1.1 Monochrome Spectral response

The graphs above show the quantum efficiency of the sensors used in ARIA cameras equipped with Tele-Dyne e2v sensors. The graphs refer to the sensors used in BW and NIR versions.

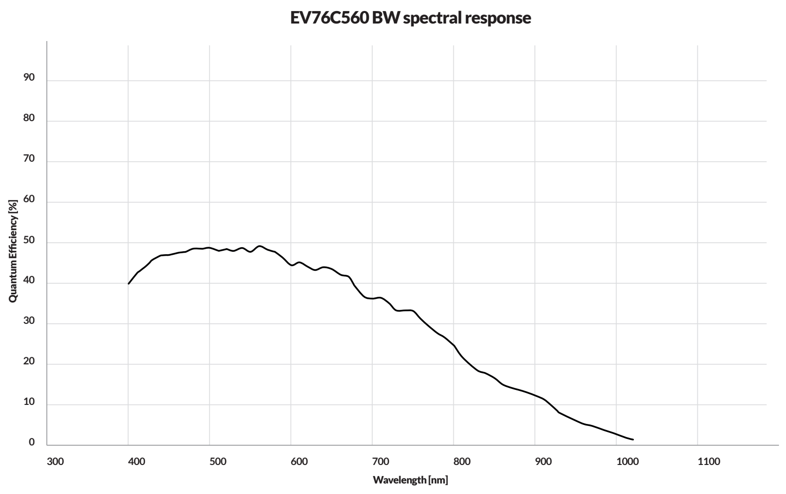


Figure 3.1: *Quantum efficiency for EV76C560 monochrome sensor – Source: TeleDyne e2v*

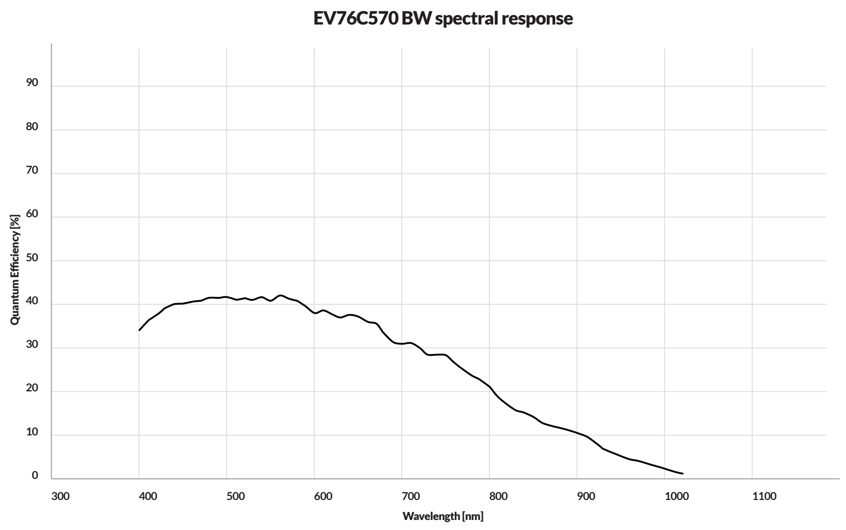


Figure 3.2: *Quantum efficiency for EV76C570 monochrome sensor – Source: TeleDyne e2v*

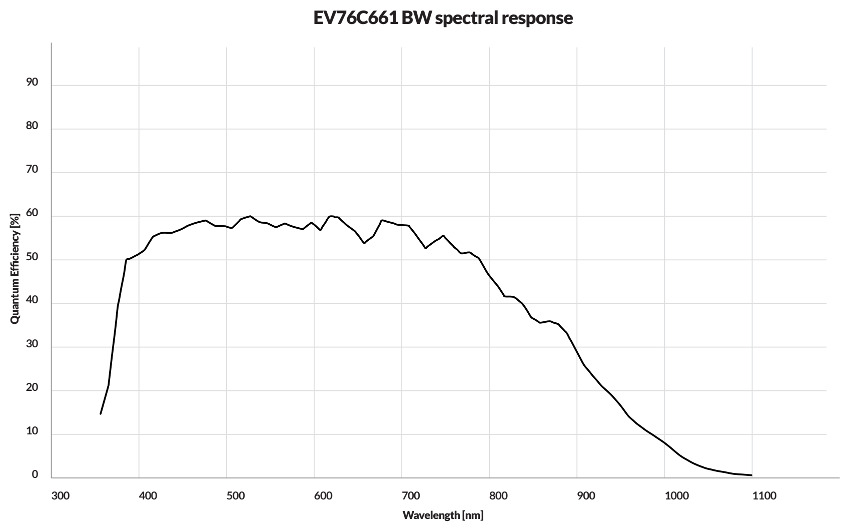


Figure 3.3: *Quantum efficiency for Ev76C661 monochrome NIR Enhanced sensor – Source: TeleDyne e2v*

3.1.1.2 Color Spectral response

TeleDyne e2v Ruby and Sapphire sensors discriminate colors using the Bayer filter, making each pixel sensitive to one of the primary colors only.

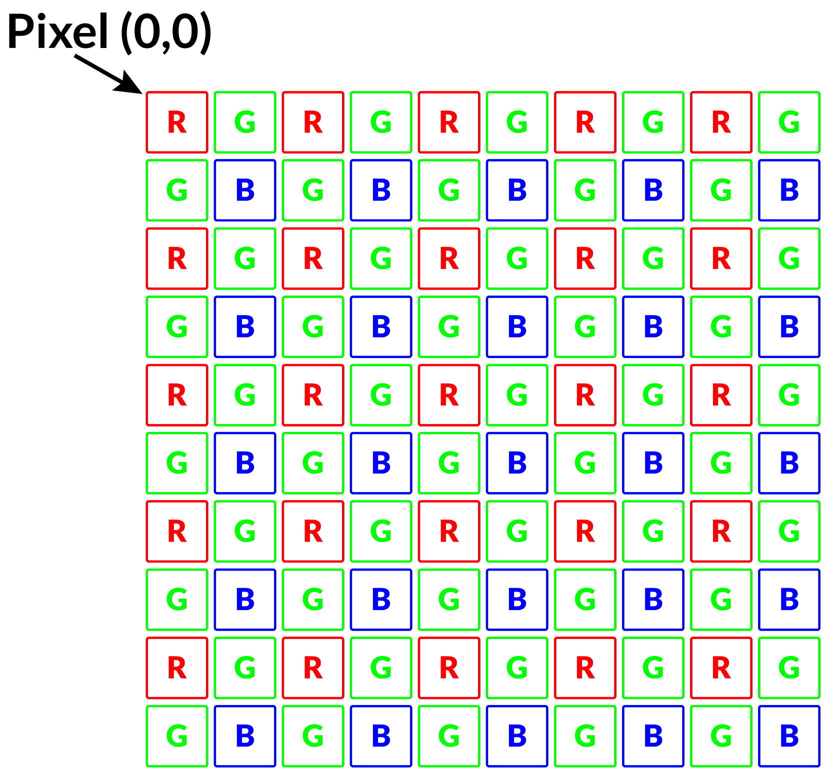


Figure 3.4: *BAYER filter pattern*

The graphs above show the quantum efficiency of the TeleDyne e2v sensors.

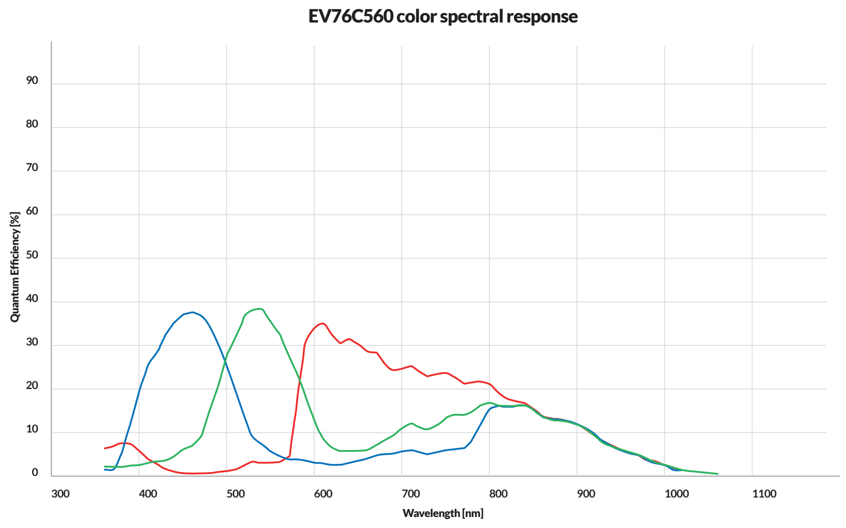


Figure 3.5: *Quantum efficiency for EV76C560 Color sensors – Source: TeleDyne e2v*

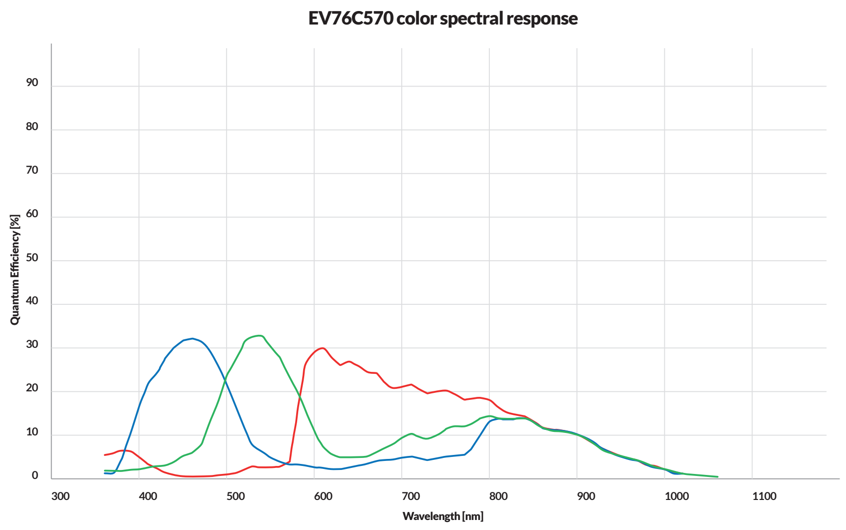


Figure 3.6: *Quantum efficiency for EV76C570 Color sensors - Source: TeleDyne e2v*

3.1.2 SONY IMX Sensors

3.1.2.1 Monochrome Spectral response

The graph in Figure 10 shows the quantum efficiency of the sensors used in ARIA cameras equipped with SONY IMX sensors. The graph refers to the sensors used in BW cameras.

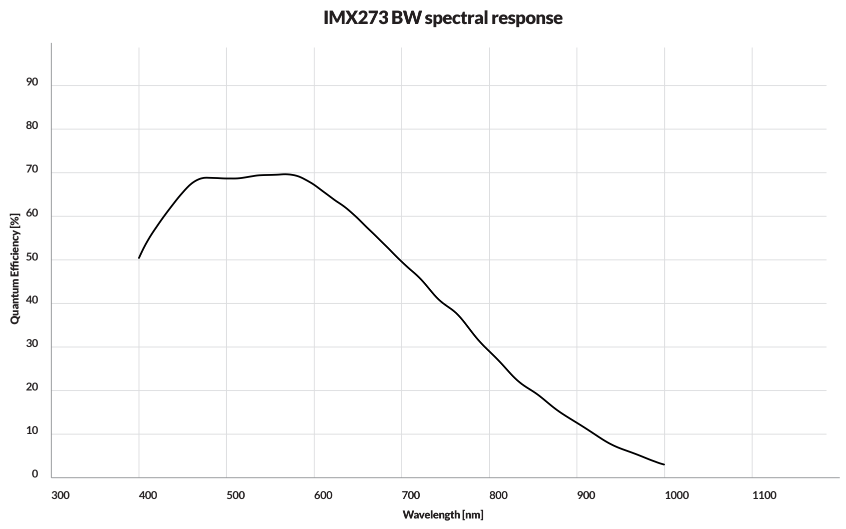


Figure 3.7: *Quantum efficiency for SONY IMX273 monochrome sensor – Source: SONY.*

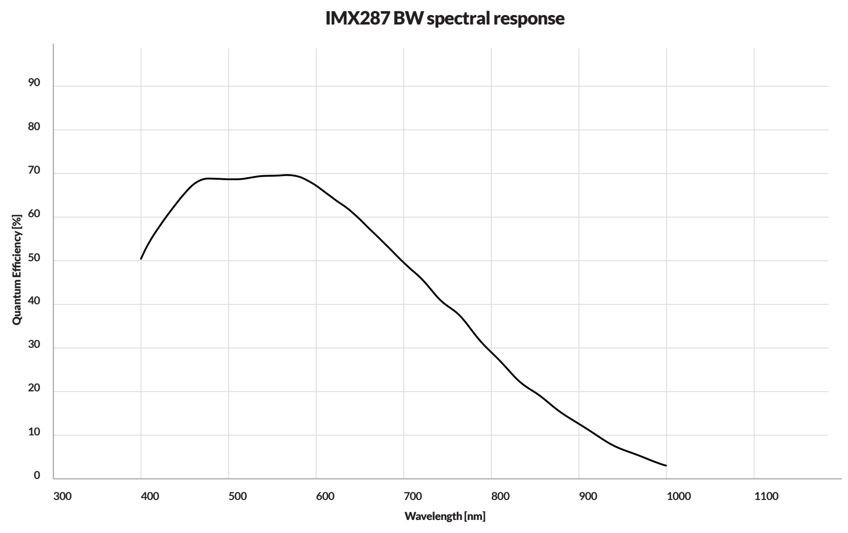


Figure 3.8: *Quantum efficiency for SONY IMX287 monochrome sensors - Source: SONY.*

3.1.2.2 Color Spectral response

SONY IMX sensors discriminate colors using the Bayer filter, making each pixel sensitive to one of the primary colors only. The graph in figures above show SONY IMX sensor quantum efficiency.

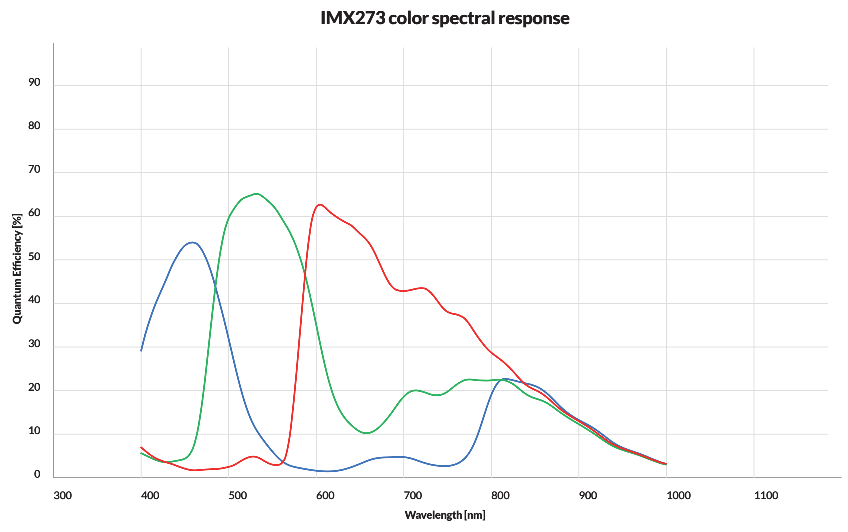


Figure 3.9: *Quantum efficiency for SONY IMX273 Color sensor – Source: SONY.*

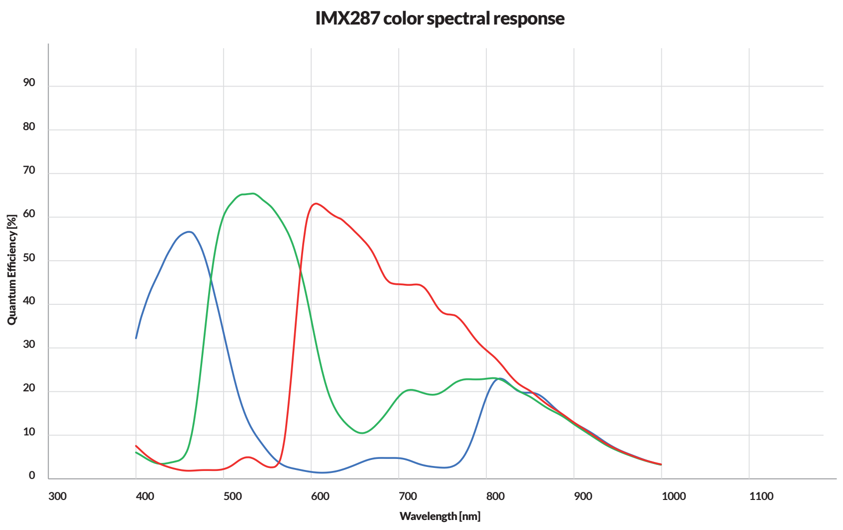


Figure 3.10: *Quantum efficiency for SONY IMX287 Color sensor - Source: SONY.*

3.2 FILTERS

ARIA C-Mount color camera models are equipped with Infrared cut-off filters to reduce unwanted camera sensitivity in the near infrared spectrum.

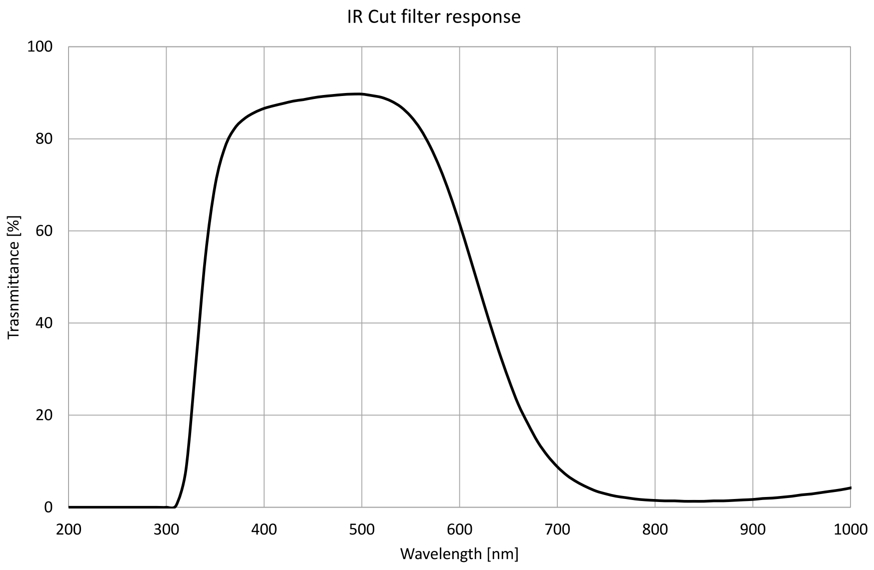


Figure 3.11: *ARIA IR Cut filter response - Source: Hoya Candeo Optronics Corporation.*

ARIA C-Mount monochrome models are provided without protective glass or IR filter installed.

Please note that ARIA cameras can also be provided with transparent protective glass or IR filter installed. Transparent protective glass has the following properties:

Type	Anti-reflective
Material	Schott B 270 © i
Thickness	1 ± 0.05 mm

Table 3.1: *Protection glass characteristics*

Please contact Alkeria SRL Sales Offices for information about monochrome C-Mount Aria cameras equipped with protective glass or IR filter.

ARIA S-Mount models are provided without protective glass or IR filter installed. IR filter or protective glasss can be installed as an option on Aria cameras with S-Mount adapter. Please contact Alkeria SRL Sales Offices for information about S-Mount cameras equipped with protective glass or IR filter.

3.3 MECHANICAL DESCRIPTION

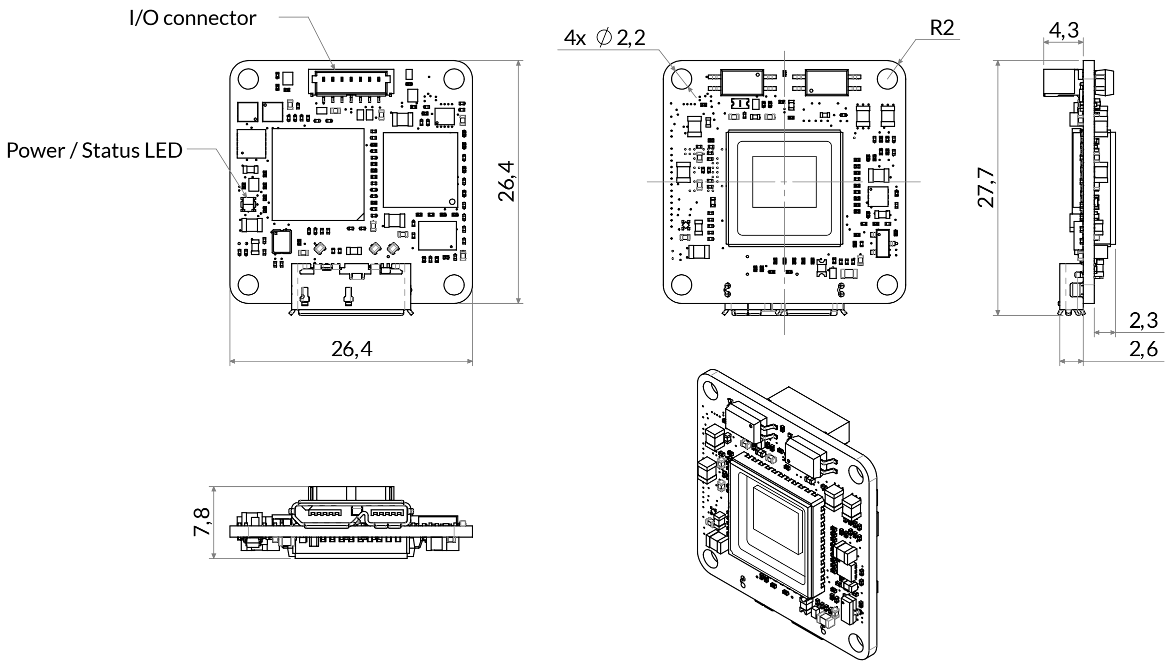


Figure 3.12: *ARIA e2v sensor equipped camera dimensions (without lens adapter)*

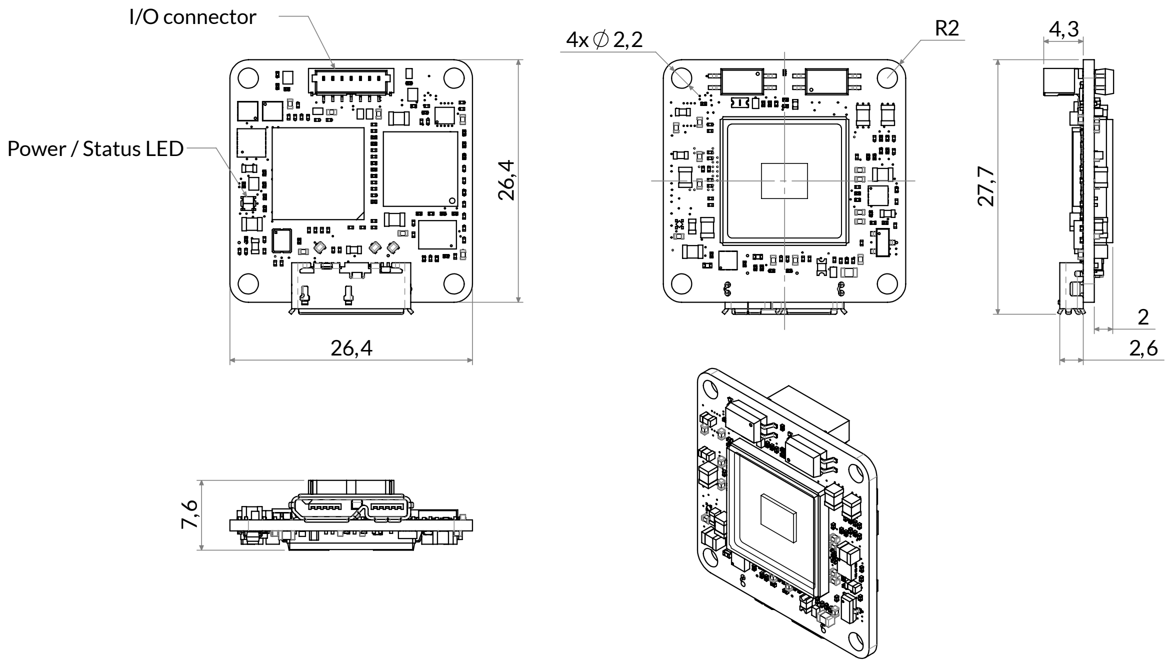


Figure 3.13: *ARIA Sony sensor equipped camera dimensions (without lens adapter)*

3.4 ADAPTERS FOR STANDARD LENS

ARIA family has some adapters available (described below) which allow using standard lens.

The adapters of ARIA cameras are made of black anodized aluminum alloy, using high precision machining. They are provided with two M4 holes at each side of the camera, for connecting to the mount, as shown in the drawings below. When designing the bracket that will support your ARIA , please use the reference planes shown in Figure 16 and Figure 18 (marked as A and B); using these reference planes will allow to achieve a precise positioning of the devices with significant repeatability.

3.4.1 C-mount adapter

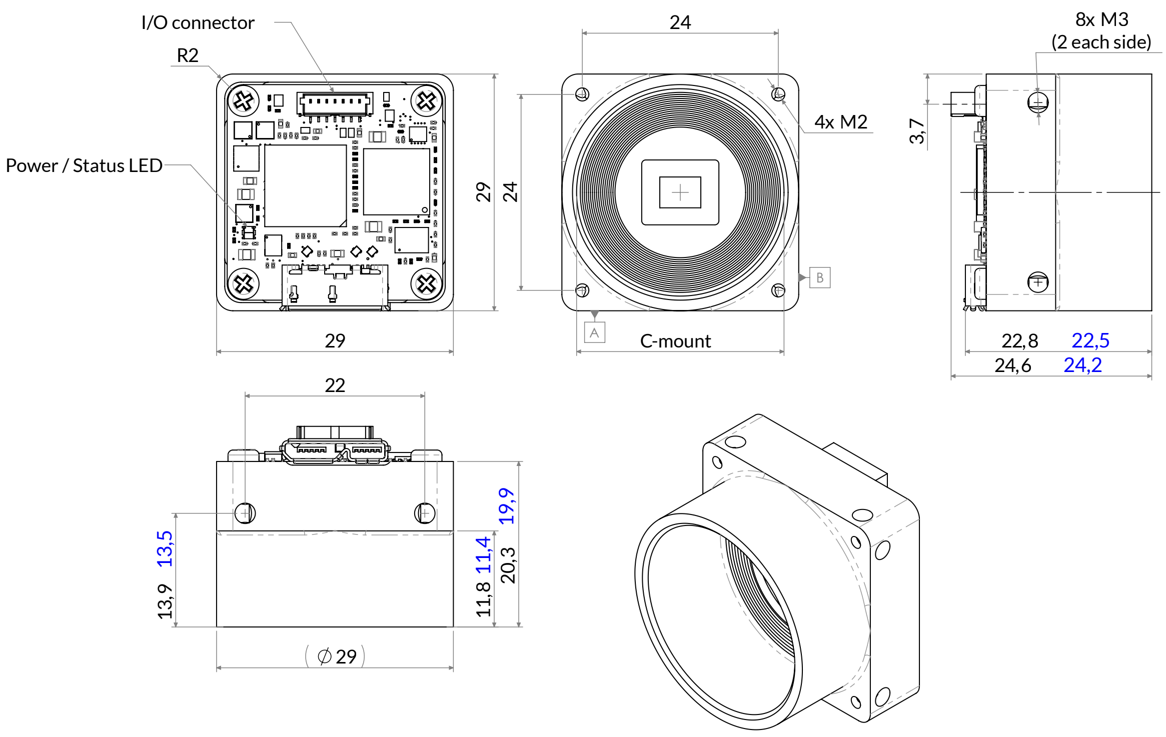


Figure 3.14: *ARIA C-mount mechanical drawings*

Blue-marked measures refers to B/W models.

The ARIA C-mount adapter allows interfacing ARIA camera models with standard C-mount lenses. ARIA comes with the C-mount adapter already assembled. If you need to disassemble it, just remove the four M2 back screws. Please perform this operation in a clean and dust-free environment. To reconnect the adapter, simply align ARIA with the adapter keeping USB3 connector on his seating and tighten the four M2 screws supplied with it.

Warning

The maximum tightening torque for screws is 1.0 Nm. 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.

3.4.1.1 Maximum lens protrusion

ARIA C-Mount adapter allows to install only lens with a maximum protrusion of 11.4 mm.

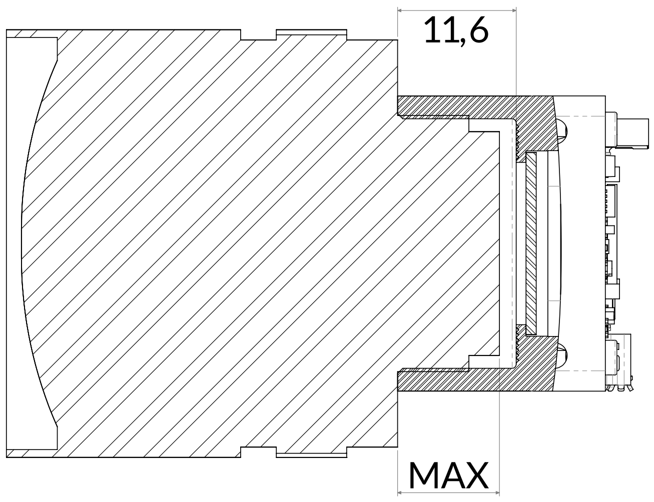


Figure 3.15: *Maximum protrusion for C-Mount type lens*

Warning

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

3.4.2 S-mount adapter

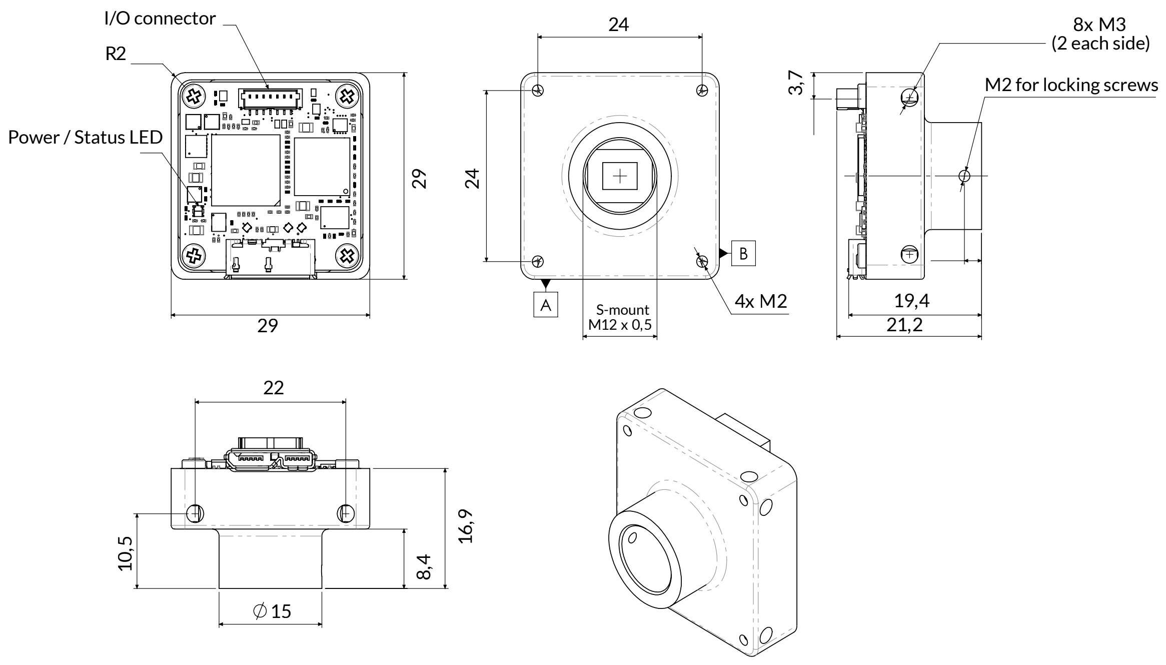


Figure 3.16: *ARIA S-mount mechanical drawings*

The ARIA S-mount adapter allows interfacing ARIA camera models with standard S-mount lenses. ARIA comes with ARIA S-mount adapter already assembled. If you need to disassemble it, just remove the four M2 back screws. Please perform this operation in a clean and dust-free environment. To reconnect the adapter, simply align ARIA with the adapter keeping USB3 connector on his seating and tighten the four M2 screws supplied with it.

Warning

The maximum tightening torque for screws is 1.0 Nm. 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

ARIA S-Mount adapter allows to install only lens with a maximum protrusion of 11.5 mm.

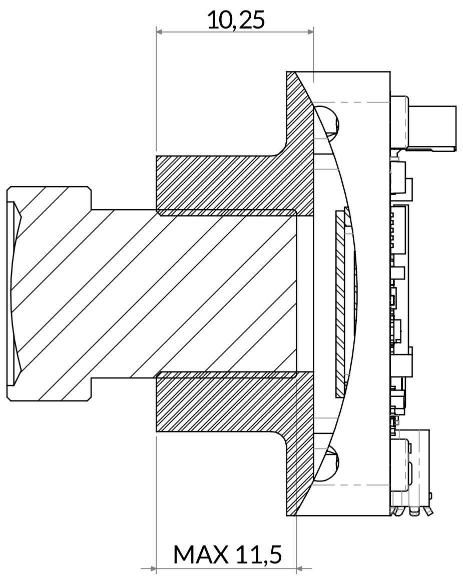


Figure 3.17: *Maximum protrusion for S-Mount type lens*

Warning

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

3.5 BACK CASING FOR ARIA CAMERAS

3.5.1 Back casing with C-mount adapters

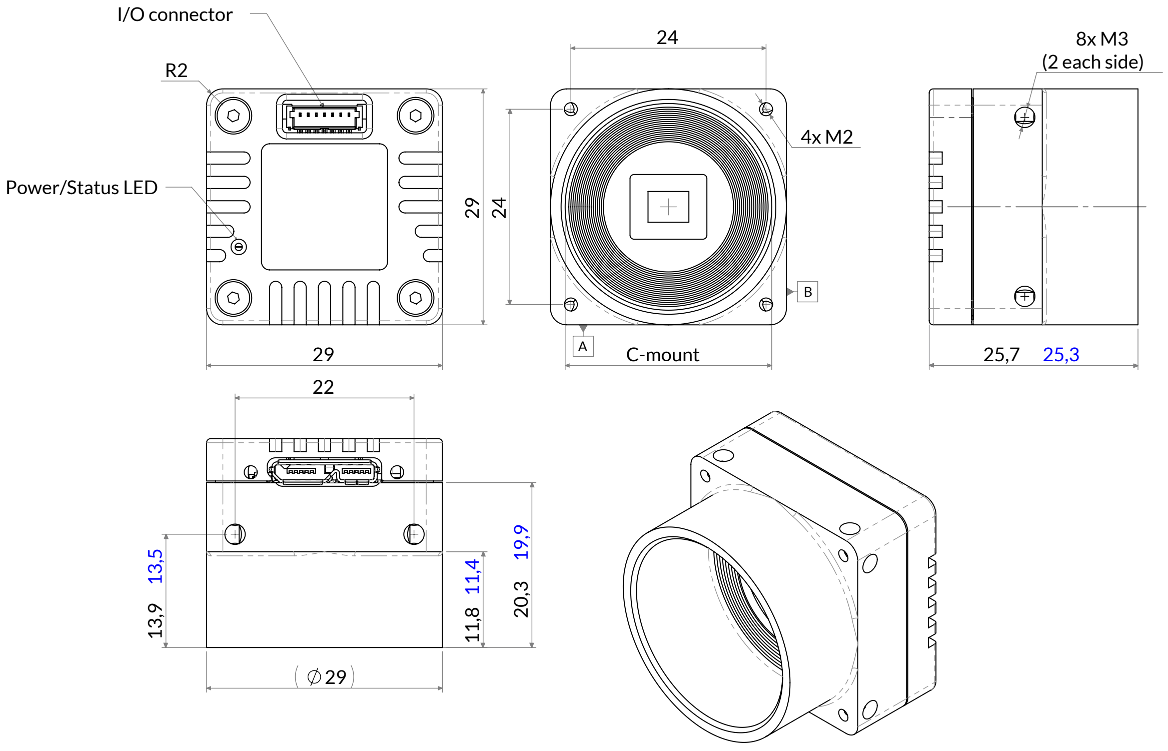


Figure 3.18: *ARIA Mechanical drawings of ARIA with back casing and C-mount lens adapter*

Blue-marked measures refers to B/W models.

Warning

The maximum tightening torque for screws is 1.0 Nm. Exceeding the maximum tightening torque may damage the camera.

Warning

Please take care to not damage I/O socket during connection and disconnection operations.

Note

We suggest you to use USB3 cables with screws if your ARIA camera has back casing enclosure.

3.5.2 Back casing with S-mount adapters

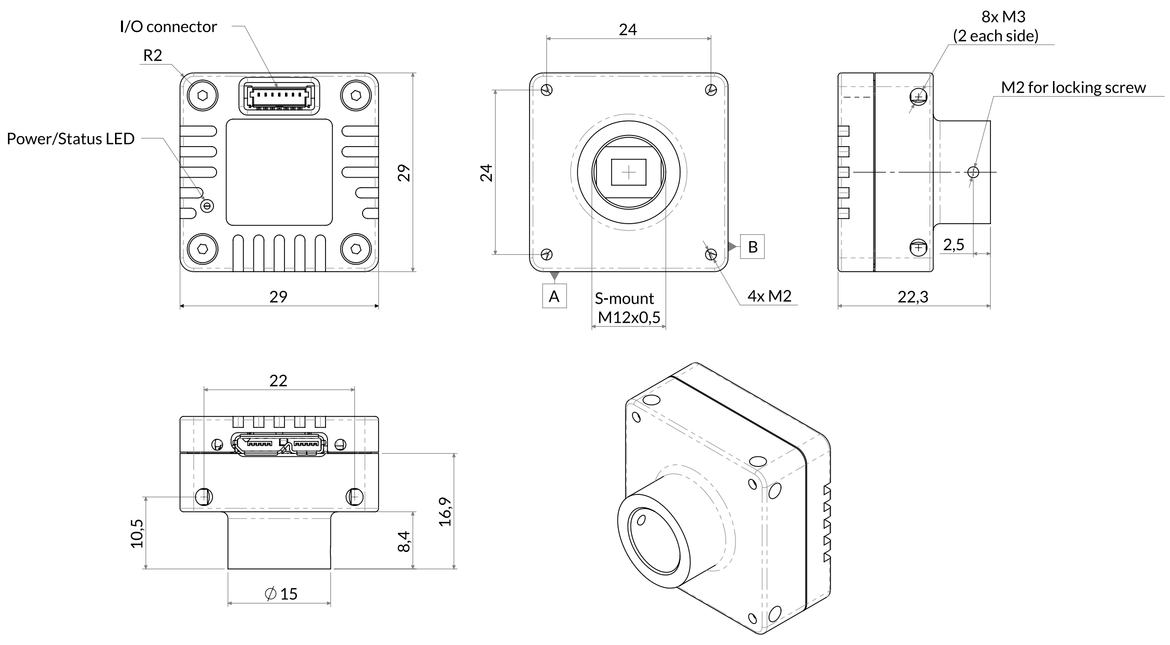


Figure 3.19: *ARIA Mechanical drawings of ARIA with back casing and S-mount lens adapter*

The ARIA C-mount adapter allows interfacing ARIA camera models with standard C-mount lenses. ARIA comes with the C-mount adapter already assembled. If you need to disassemble it, just remove the four M2 back screws. Please perform this operation in a clean and dust-free environment. To reconnect the adapter, simply align ARIA with the adapter keeping USB3 connector on his seating and tighten the four M2 screws supplied with it.

Warning

The maximum tightening torque for screws is 1.0 Nm. Exceeding the maximum tightening torque may damage the camera.

3.6 FLANGE FOCAL DISTANCE

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

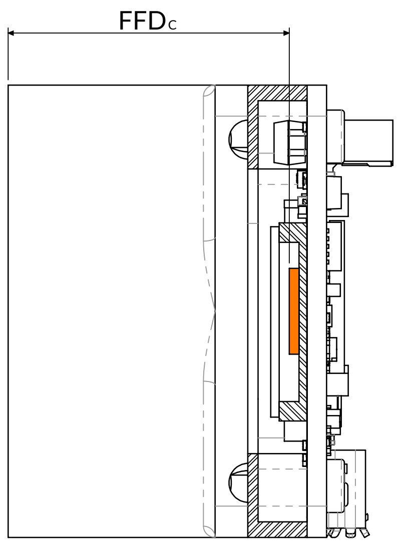


Figure 3.20: *ARIA camera equipped with C-Mount*

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

3.7 HANDLING PRECAUTIONS

The optical parts of ARIA 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.

3.7.1 Temperature and humidity

	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*

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

3.7.2 Warranty limitations

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

3.7.3 Mechanical stress

ARIA 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. ARIA connector can host cables with screws or locking systems; it is highly recommended to use these cables when ARIA is exposed to shocks, vibrations or harsh environments.

3.7.4 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.

3.7.5 Board level recommendations

If your application has been implemented with the board level ARIA , make sure that the circuit board has no conductive contact with other objects. Conductive contact can cause short circuits or overvoltage which can damage the camera. Red areas on Figure 3.21 represent the only conductive contacts allowed for the board level. The custom designed contacts have to be planar with the PCB surface not to stress it asymmetrically and must not contain oxides otherwise they will damage the electronic components.

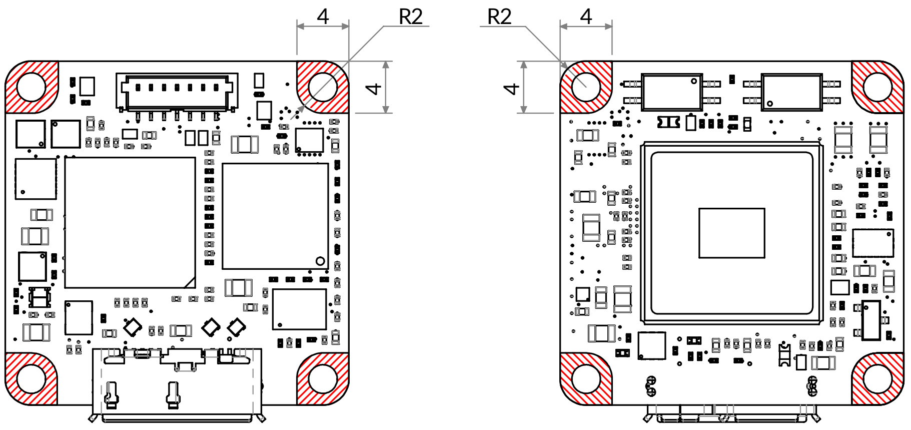


Figure 3.21: *ARIA board level clearance*

Figure 3.22 shows the recommended set of components to build a custom design enclosure. Four screws (1) to couple the board level (2) to the lens mount (6). Between 6 and 2 a thermal compound (e.g. Sil-Pad marked as 5) will improve thermal coupling, moving heat outside the camera. If the lens is not dust proof, it's important to place a filter (4) and a filter holder (3) to block dust intrusion. For color sensors an IR filter is also crucial to block unwanted frequencies that can alter images in some conditions. For monochrome sensor a neutral filter is recommended.

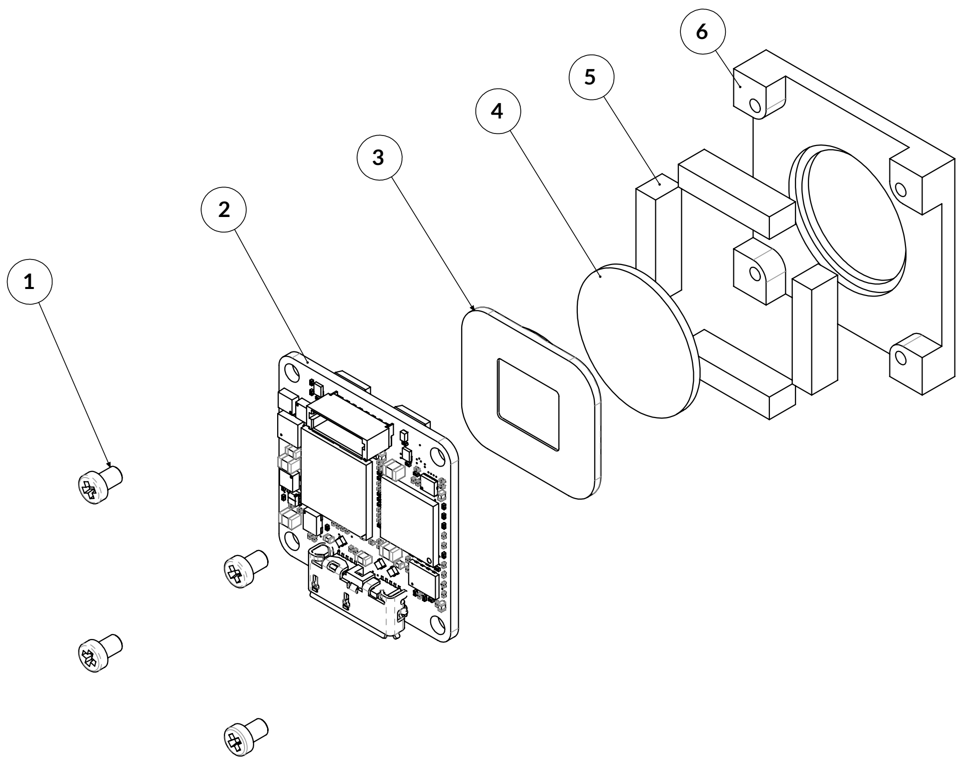


Figure 3.22: *ARIA custom-designed enclosure example*

It's also important to provide sufficient heat dissipation to keep the temperature of your ARIA board module within the specified range. Using a metal lens (instead of plastic) will produce a better heat dissipation as well as using the back cover accessories, as shown in Figure 3.23.

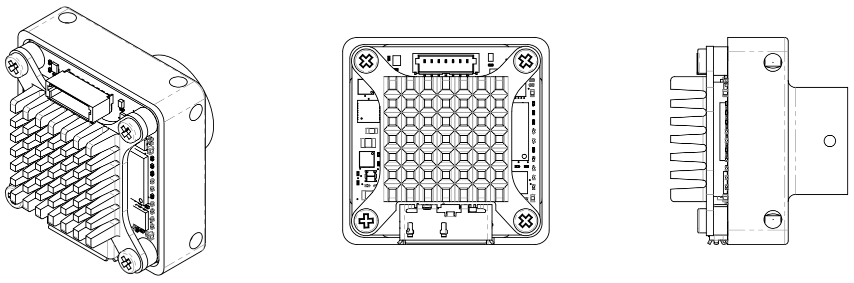


Figure 3.23: *Heat sink example*

Remember to place a thermal compound between PCB and metal to improve heat dissipation and avoid short circuits.

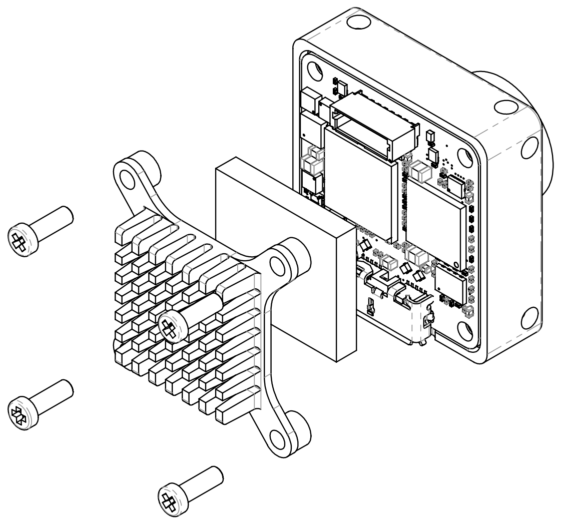


Figure 3.24: *Heat sink with thermal compound*

Also forcing a constant air flow around the camera will increase heat dissipation and keep temperature down.

3.7.6 Monitoring internal temperature

During continuous acquisition, ARIA temperature keeps rising until a steady temperature state is reached. ARIA 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 ARIA 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()));
```

3.7.7 Automatic internal temperature safety mechanism

An automatic temperature monitoring mechanism, shown in Figure 3.25, has been implemented in ARIA 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.

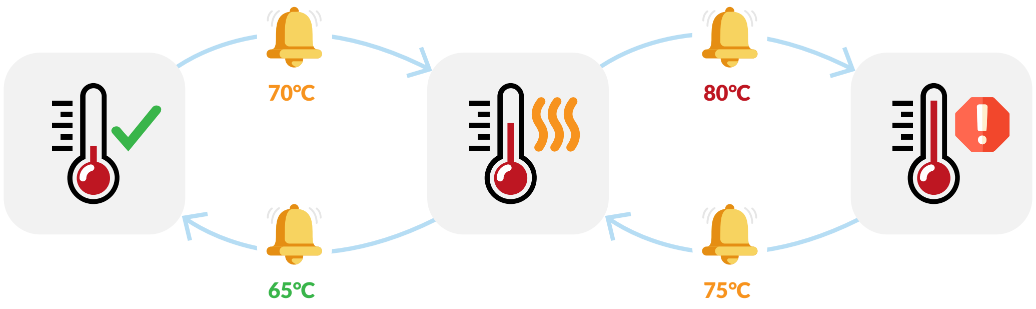


Figure 3.25: *Temperature status transition*

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:

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;
```

3.7.8 EMI/ESD precautions: noise and electrostatic discharge

ARIA cameras are typically used in industrial environments, where many devices may generate electromagnetic interference and electrostatic discharge. Although ARIA 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.

3.8 CLEANING ARIA CAMERA

Each ARIA 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.

3.8.1 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.

3.8.2 Cleaning ARIA 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.

3.8.3 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.

3.8.4 Impurities

Before cleaning any parts, it is crucial to understand where impurities are located. You can distinguish particles on the lens, filter or 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 <u>on the sensor</u> appear as dark well-focused spots that remain still while you move the camera respect to the subject (refer to Figure 3.27). Particles located <u>on the filter</u> surface look blurred, regardless of the lens focus, and remain still even when rotating the lens (refer to Figure 3.26). Dust <u>on the lens</u> looks blurred and rotates together with the lens.

Figure 3.26: *Example of lens or filter impurities*

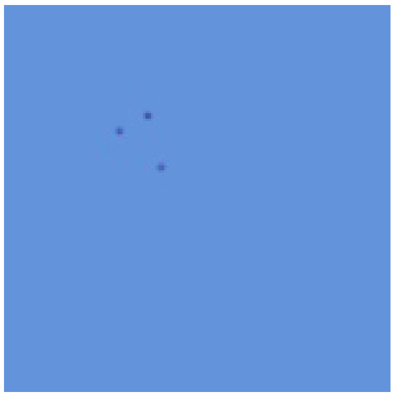


Figure 3.27: *Example of sensor impurities*

3.8.5 Filter cleaning

Keep camera facing downwards when removing lens, lens adapter or dust cap. This prevents dust from collecting on the filter surface.

For a better access to the sensor filter, it is recommended to remove the lens adapter.

Use a lens blower to remove impurities from the filter surface and the lens adapter. Reassemble the optical components and repeat the acquiring procedure. For a perfect result, it might be necessary to repeat the above procedure a few times.

If spots are still present, it may be necessary to mechanically clean the filter, following the procedure below:

- apply a small amount of cleaning liquid to a new lens cleaning wipe,
- wipe the filter surface in a delicate circular motion from the center to the sides to move impurities out of the sensor field of view,
- repeat cleaning motion until all impurities are removed.

If you are not satisfied with your cleaning results, please contact rma@alkeria.com to arrange the camera return procedure (refer to Section 11.1).

Warning

Improper cleaning procedure may lead to filter surface scratches. If you are not familiar with glass cleaning technique please contact rma@alkeria.com to arrange the camera return procedure (refer to Section 11.1).

3.8.6 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 11.1).

3.8.7 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.

3.8.8 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 11.1).

4 Interfacing to the world

ARIA features an I/O connector (TE Connectivity p/n 1734595-7, mating p/n 1470364-7) interfacing to external signals. The I/O connector provides two bidirectional lines, one optoisolated output and one optoisolated input. The I/O connector is located on the back of the camera as indicated in the following figure:

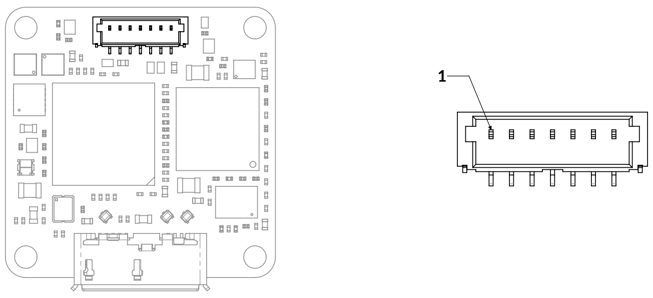


Figure 4.1: *ARIA I/O Connector*

Pin	Name	Function
1	OPGND	External Ground
2	P3	Optoisolated Output
3	P2	Optoisolated Input
4	GND	Ground
5	P0	Bidirectional I/O
6	P1	Bidirectional I/O
7	DC Out	3.3 Vdc, 200 mA source max

Table 4.1: *ARIA 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.

4.1 I/O MODULE

4.1.1 Optoisolated input module structure (port 2)

Figure 4.2 shows the internal structure of ARIA optoisolated input module.

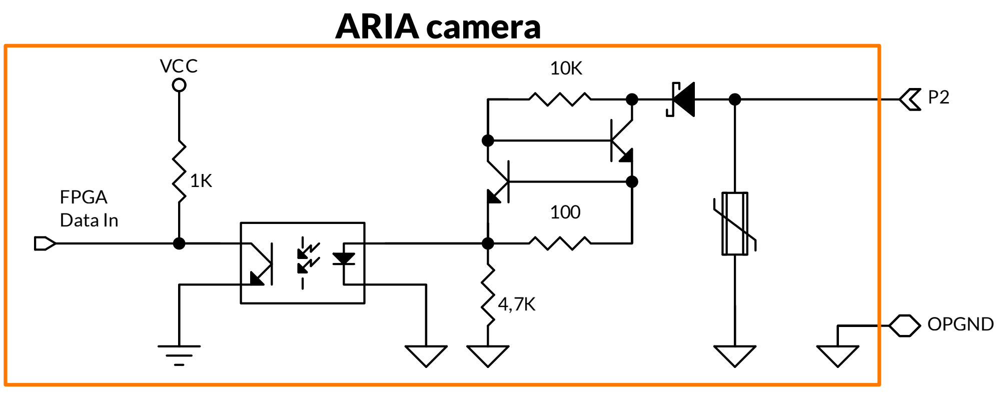


Figure 4.2: *ARIA optoisolated input port internal structure*

Warning

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

Warning

To correctly perform signal isolation do not connect OPGND to camera ground.

Input state	Vin min (V)	Vin max (V)
Logic 0	-24	0.8
Indeterminate	0.8	2.5
Logic 1	2.5	24

Table 4.2: *ARIA I/O Connector Pinout*

4.1.1.1 Debouncing module

Each ARIA 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 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 39); 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 4.3). The debounce time can be set between 0 (debouncing disabled) and 65535 µs, with a 1 µs granularity.

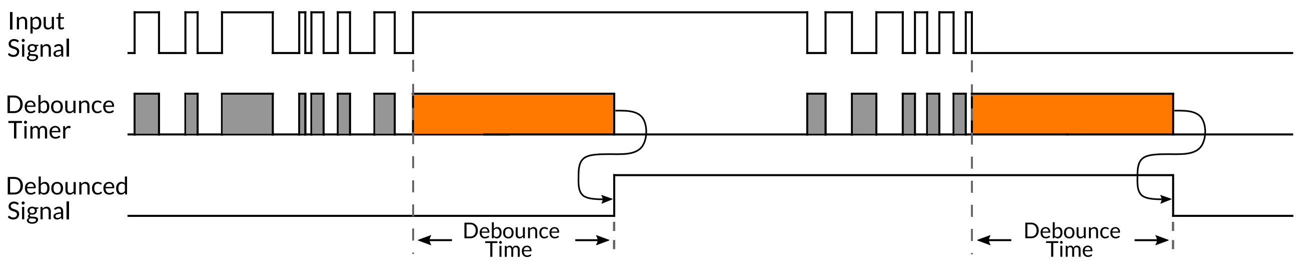


Figure 4.3: *Time diagram of the debouncing filter*

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.

4.1.1.2 Input events

ARIA 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 2:

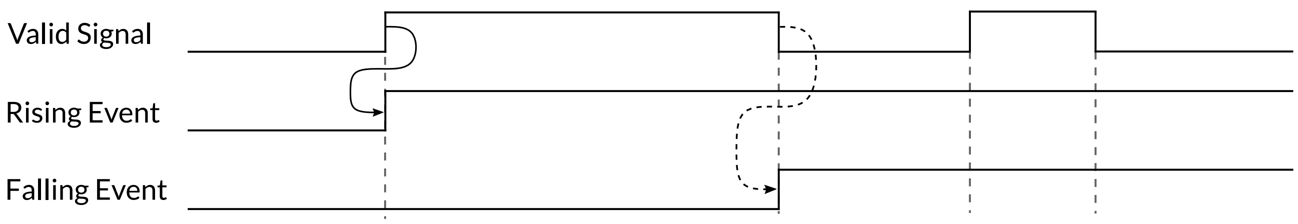


Figure 4.4: *Input event generation and readout*

Example Code 4.1 | *Input Change Event*

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

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.

4.1.2 Optoisolated output module structure (port 3)

Figure 4.5 shows the internal structure of ARIA optoisolated input module.

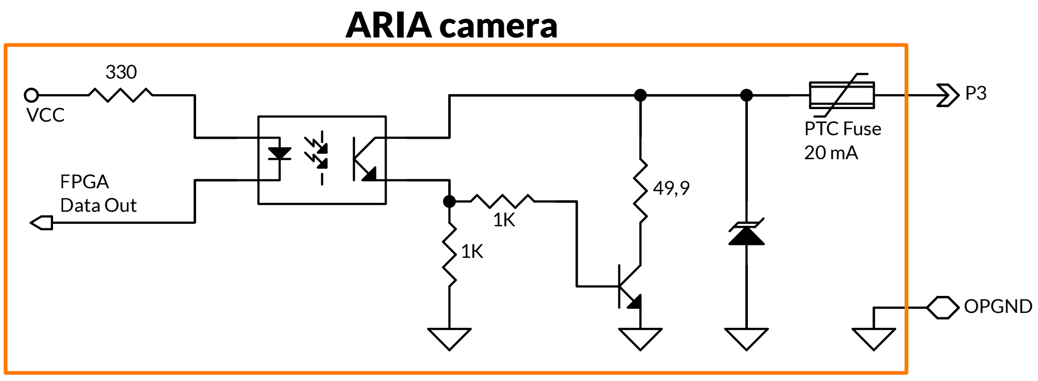


Figure 4.5: *ARIA optoisolated output port internal structure*

ARIA optoisolated output port is open-collector type that requires external circuitry to produce high state level. Pin is protected against polarity reversal and overcurrent sink (20 mA).

Warning

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

4.1.2.1 Examples of output module connection: Led drive

P3 output pin can be used to directly control LEDs. Due to max current limitation of internal circuitry a suitable resistance R must be chosen to limit the current.

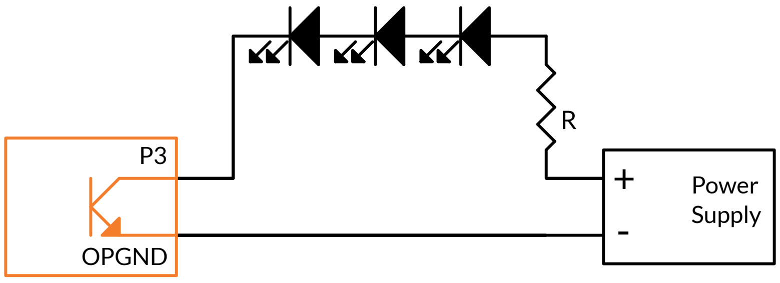


Figure 4.6: *Example of Low-Current leds driving*

If a higher current is required the configurations of Figure 43 and Figure 44 can be used: the first one polarizes the OPGND reference, making the P2 pin not to be used. The second one uses one more transistor but allows to connect an external ground reference to OPGND and so the P2 input can be used as well.

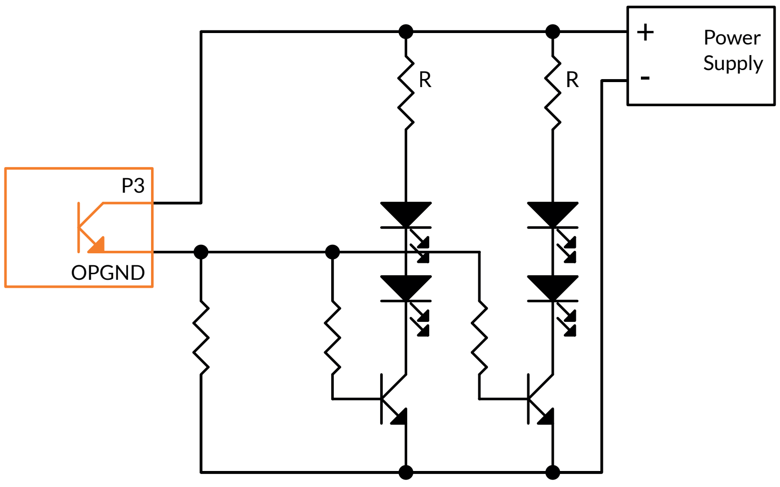


Figure 4.7: *Example of High-Current Leds driving with OPGND polarized*

4.1.2.2 Example of Relay driving using OptoIsolated Output

To prevent damage to P3 output do not directly connect inductive loads (such as motors and coils); use instead an external transistor driven by P3. A simple configuration is proposed in Figure 45: transistor Q is biased through R and it lets current flow into relay's coil, even when ARIA camera is not powered. Writing a logic 1 to P3 disable transistor and turns off relay: the logic is inverted.

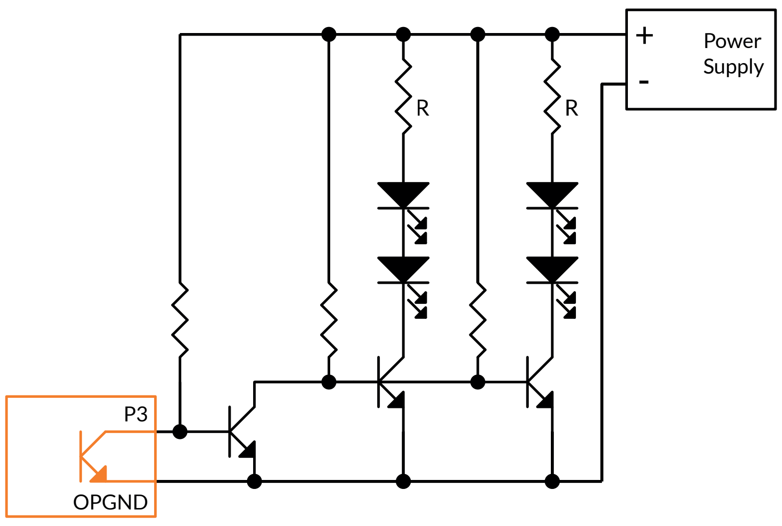


Figure 4.8: *High-Current led driving with OPGND not polarized*

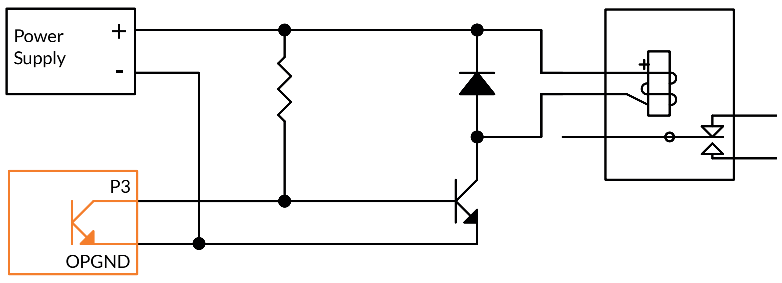


Figure 4.9: *Driving normally closed relay*

To prevent the relay's coil to be active when ARIA camera is not powered two circuits are proposed in Figure 4.10 and Figure 4.11: the first one polarizes the OPGND reference, so only the P3 pin can be used. The second one uses one more transistor but allows to connect an external ground reference to OPGND and so the P2 input can be used as well.

Both configurations require a logic 1 to be written on P3 to turn on the relay: the logic is positive.

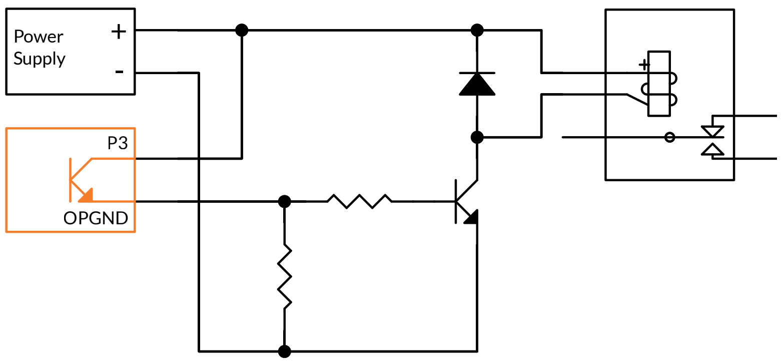


Figure 4.10: *Driving normally opened relay with OPGND polarized*

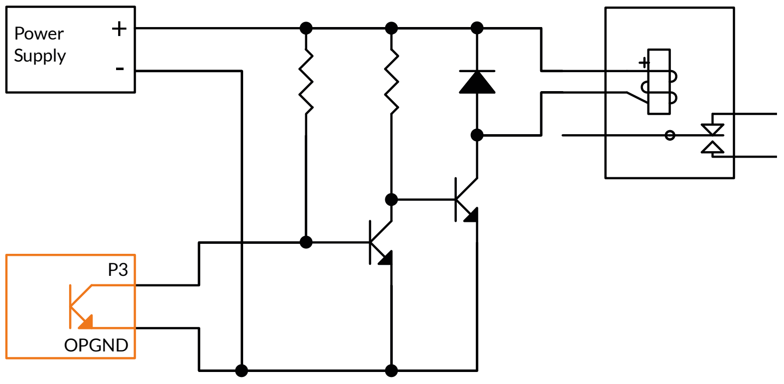


Figure 4.11: *Driving normally opened relay without OPGND polarized*

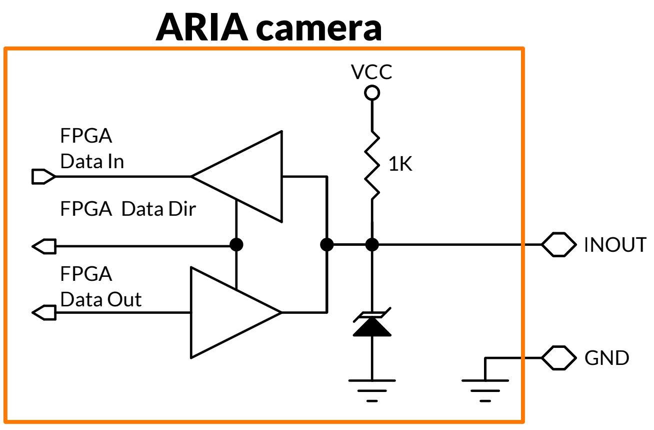


Figure 4.12: *ARIA bidirectional module structure*

4.1.3 Bidirectional module structure (port 0 and port 1)

ARIA port 0 and port 1 can be software-configured either as an input or as an output. When the camera is powered on, the ports are initially configured as an input; they can be set as an output by software. All the statements above about input and output ports apply also to port 0 and port 1.

For more information on how to configure and use the input/output port please refer to the code samples in Section 4.3.4 and to the examples found in the MaestroUSB3 SDK.

4.1.4 I/O switching time

The presence of optocouplers and the related circuitry is a limitation for switching time. Table 4.3 shows measured times for various external power supply values and pullup resistors on the P3 output pin:

V_{ext} [V]	R_{pullup} [kΩ]	V_{OL} [V]	t_{HL} [µs]	t_{PHL} [µs]	t_{LH} [µs]	t_{PLH} [µs]
3.3	1	0.75	0.6	0.4	20.6	18.1
3.3	4.9	0.57	0.6	0.3	46.6	19.3
5	1.5	0.76	0.7	0.4	24.2	19.7
5	4.7	0.65	0.7	0.4	41.9	20.8
12	1.5	0.97	1.1	0.4	34.1	25.0
12	4.7	0.78	1.1	0.4	41.1	26.0
24	1.5	1.11	1.5	0.4	46.7	30.3
24	4.7	0.75	1.5	0.4	52.3	31.8

Table 4.3: *P3 output Timing*

You can refer to the Figure 4.13 to better understand the times shown in Table 4.3. The internal signal was shown in black, while the signal coming out from P3 was shown in orange. The propagation times (t_{PLH}, t_{PHL}) are measured between 50% of the internal signal and 10% of the output. The transition times (t_{LH}, t_{PH}) are measured between 10% and 90% of the output signal.

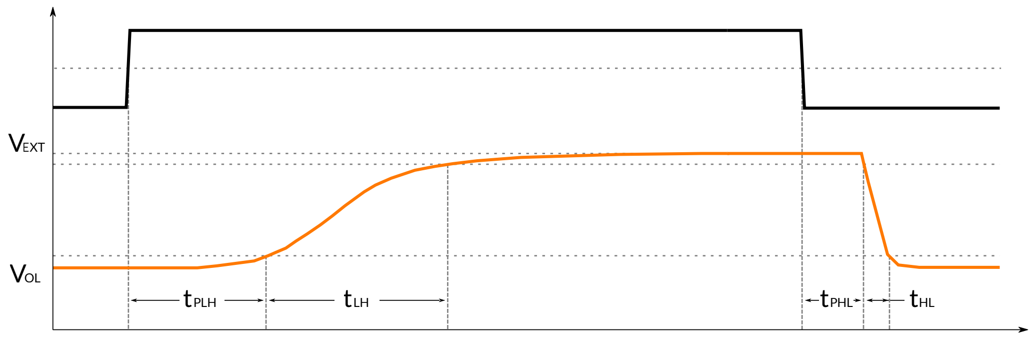


Figure 4.13: *P3 output characteristic*

Table 4.4 shows measured times for various external power supply values and pullup resistors on P2 input pin when it is driven by an Open-Drain source:

V_{ext} [V]	R_{pullup} [kΩ]	t_{HL} [µs]	t_{PHL} [µs]	t_{LH} [µs]	t_{PLH} [µs]
3.3	0.18	6.7	5.3	2.6	1.3
3.3	0.68[1]	6.6	3.2	3.8	1.8
5	0.68	2.3	1.2	6.2	6.1
5	1.84[1]	6.6	3.2	3.8	1.7
12	1.5	1.6	0.8	7.0	7.3
12	4.7	5.6	2.1	5.6	3.6
24	1.5	1.2	0.7	7.3	7.9
24	4.7	1.9	0.9	6.9	6.7

Table 4.4: *P2 input timing driven by Open-Drain source*

[1]Maximum admitted resistor for proper levels detection.

You can refer to the Figure 4.14 to better understand the times shown in Table 14. The input signal was shown in orange, while the internal signal processed from the optocoupler was shown in black. The propagation times (t_{PLH}, t_{PHL}) are measured between 50% of the input signal and 10% of the internal signal. The transition times (t_{LH}, t_{PH}) are measured between 10% and 90% of the internal signal.

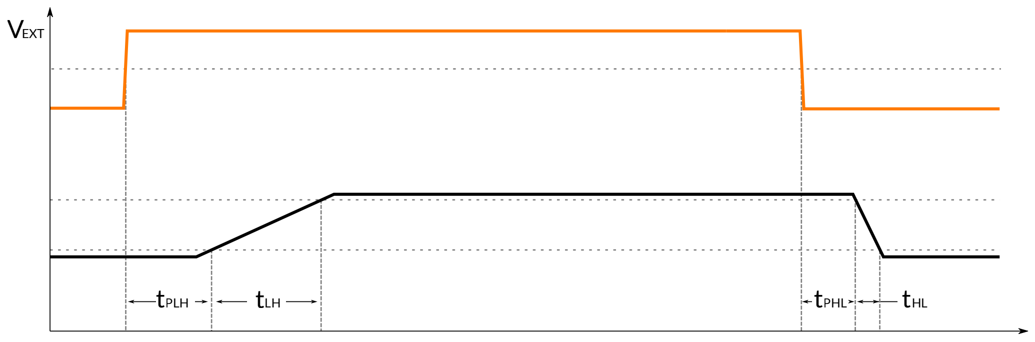


Figure 4.14: *P2 input characteristic*

Bidirectional I/O are only limited by parasitic capacitance of external inputs, these are tested to support a serial interface up to 1 MBaud/s.

Warning

Input signals are usually processed by the debouncing module (see Section 4.1.1.1), 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.

4.2 USING INPUT SIGNALS

As shown in Figure 4.15, 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.

4.2.1 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 4.16 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 4.1.1.1).

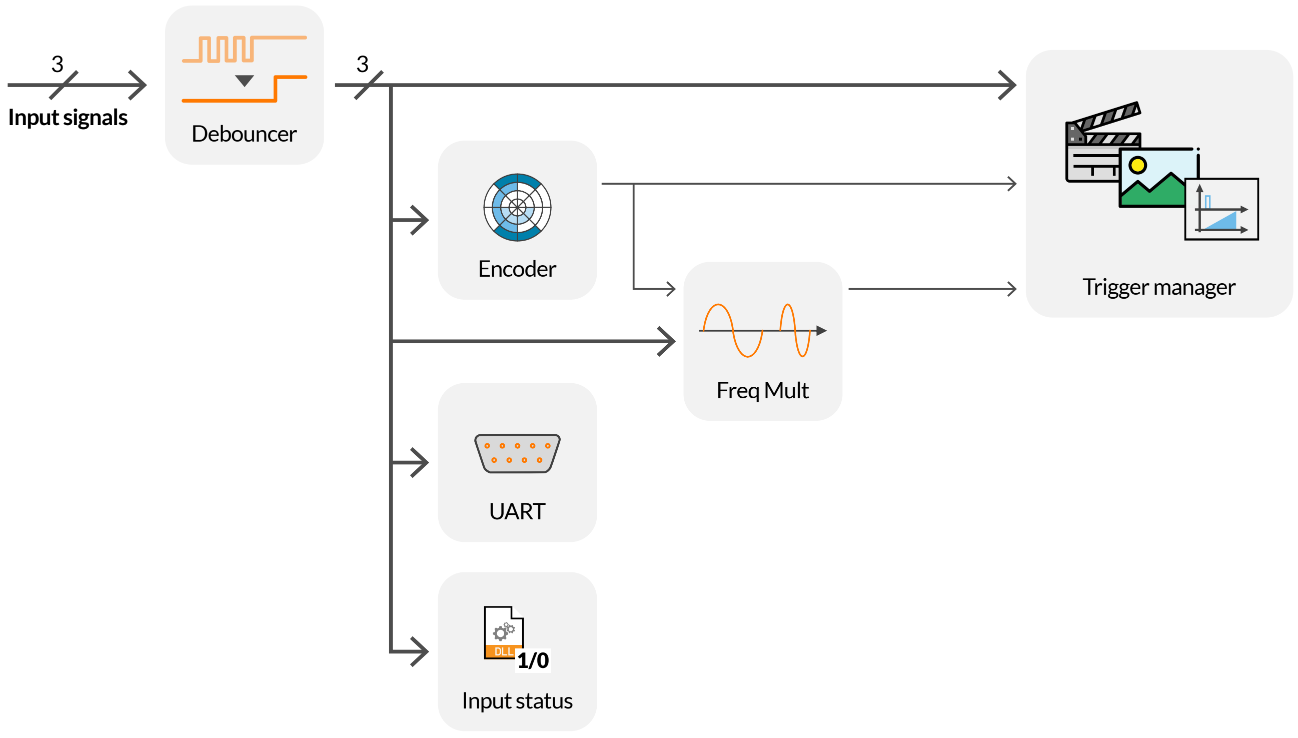


Figure 4.15: *Input port internal routing*

4.2.1.1 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 4.16) and input jitter due to mechanical vibrations (Figure 4.17).

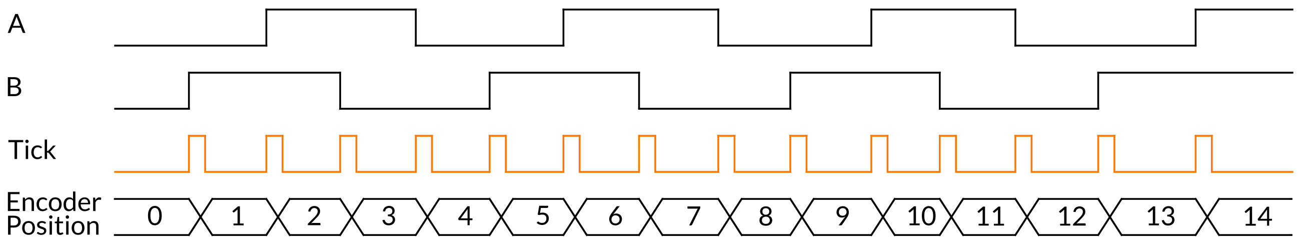


Figure 4.16: *Ticks generated in normal operations*

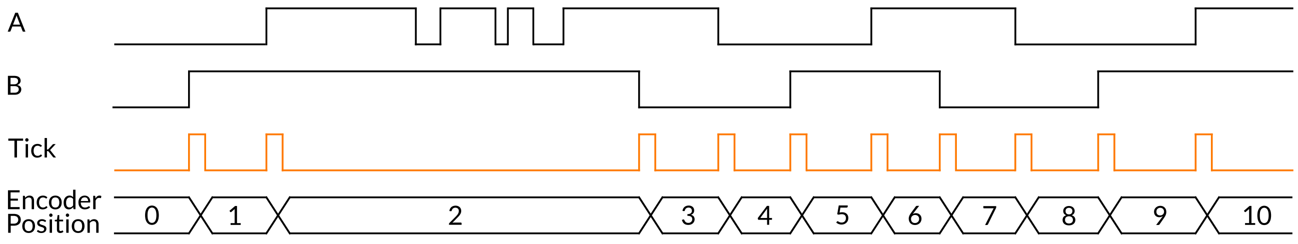


Figure 4.17: *Ticks generated in the presence of jitter*

4.2.1.2 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 4.18 and Figure 4.19).

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.

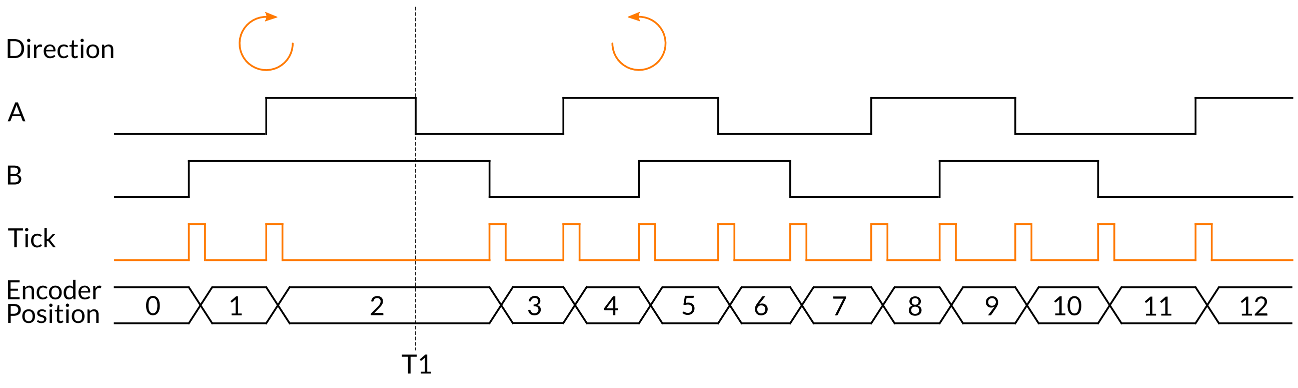


Figure 4.18: *Ticks generated when ignore direction is enabled*

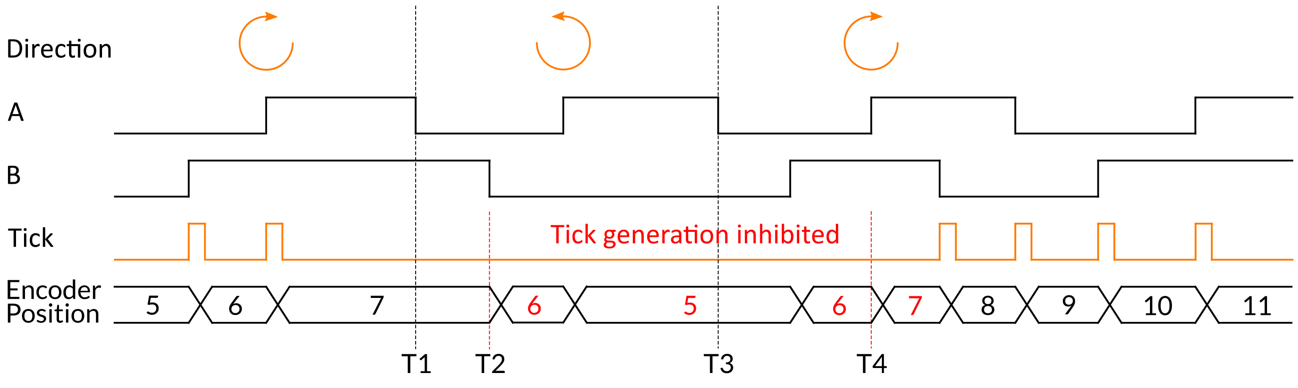


Figure 4.19: *Ticks generated when ignore direction is disabled*

4.2.1.3 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 4.2 | *Encoder ports configuration*

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

4.2.1.4 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 4.3 | *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 below configures rising edge on port 2 as encoder reset source:

Example Code 4.4 | *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;
    }
}
```

4.2.2 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 ARIA 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:

$$f_o = f_i \cdot \frac{M}{D} \tag{4.1}$$

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:

$$\frac{1}{f_i \cdot M} \tag{4.2}$$

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 erratic acquisition cadence. Minimize input jitter in your application for optimum performance.

Note

To get a more stable triggering signal it is recommended to use a programmable encoder instead of PLL.

The following code selects the source for the *frequency multiplier* and the multiplying factor, enable `FrameStartTrigger` and set PLL as trigger source:

Example Code 4.5 | *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 4.6 | *PLL boundaries readout*

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

4.2.3 Trigger manager

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

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

For further information refer to Chapter 6.

4.3 CONTROLLING OUTPUT PORTS

ARIA 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 ARIA acquisition.

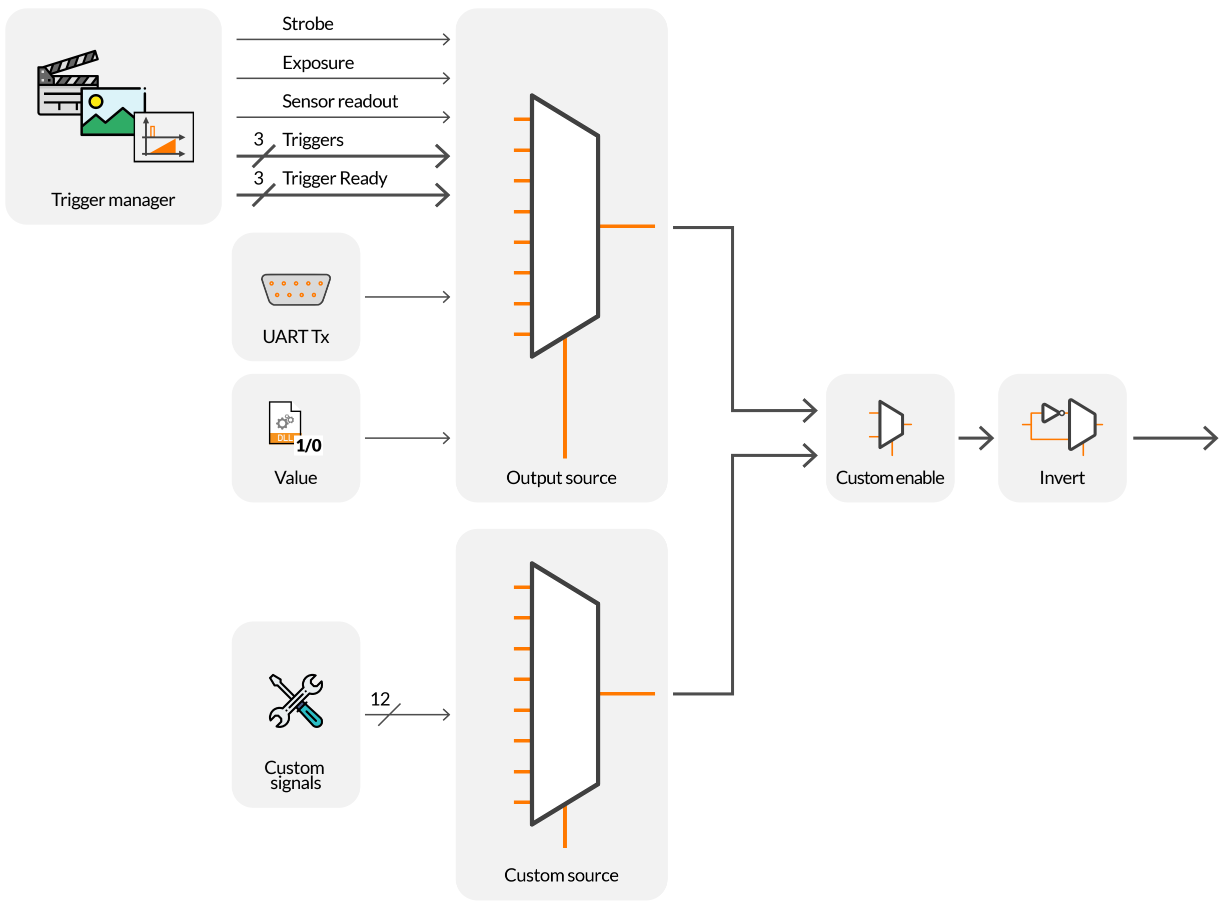


Figure 4.20: *Output port internal routing*

4.3.1 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 4.7 | *Output inversion*

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

Note

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

4.3.2 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 4.8 | *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` property (see Section 4.3.1).

4.3.3 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. Next sections illustrate how these signals can be routed to output ports and how to configure them.

4.3.3.1 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 4.9 | *Exposure as P3 source*

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

The *exposure* signal can be used to synchronize multiple ARIA 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 is 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 4.3.3.2.

4.3.3.2 Strobe

The *strobe* pulse can be generated by setting two programmable delays from the trigger event, for rising and falling delays. These two delays can be properly set with 1 μs resolution, according to your needs.

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

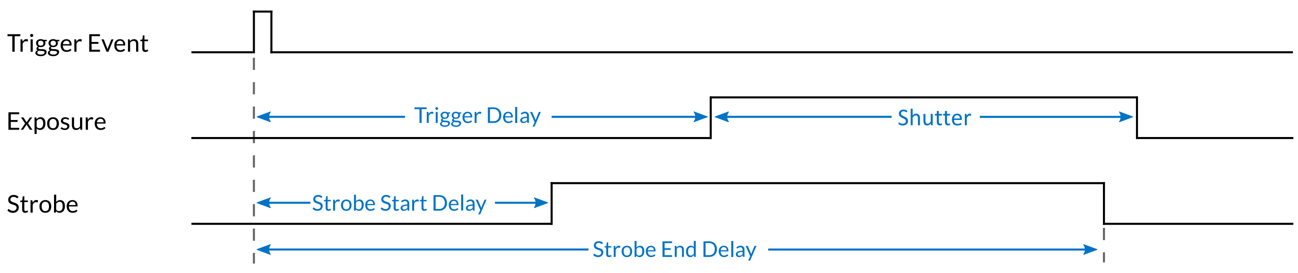


Figure 4.21: *Strobe pulse*

Example Code 4.10 | *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 4.3.3.1, but actually a different signal with its own use: both signals originate from the *frame start* event trigger, although they can have different delays with respect to it. E.g., starting from a trigger event (e.g. crossing of an encoder target position), ARIA 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.

4.3.3.3 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, 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 4.11 | *Frame start trigger as P2 source*

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

4.3.3.4 Trigger ready

The *trigger manager* module also generates a ready signal for each trigger module (acquisition, frame, end-of-exposure – see Chapter 6), indicating that the corresponding trigger module is 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 4.12 | *Frame start trigger ready as P2 source*

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

4.3.4 Custom sources

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

Example Code 4.13 | *Custom signal 10 as P0 source*

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

4.3.5 Input and output ports usage examples

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

Example Code 4.14 | *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 4.15 | *P0 configuration as output*

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

4.4 SERIAL INTERFACE MODULE

ARIA 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.

The Application Programming Interface (API) provides methods to check the input buffer level and size, to set the requested baud rates 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 4.16 | *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 `device.UART.TxFIFOSize` and wait until the output FIFO is empty before calling the Send method.

Example Code 4.17 | *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 4.18 | *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

ARIA 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).

ARIA 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 ARIA , it must not transfer data while camera is operating. ARIA 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 3.7.8 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 ARIA 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 ARIA 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 ARIA operation.

5.2 STATUS LED

ARIA 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 ARIA IS NOT ENOUGH...

You can even connect multiple ARIA cameras at the same PC and use them simultaneously: API makes it possible to balance each video data stream.

When designing the application, you can choose how the connected ARIA cameras will share the available USB 3.2 Gen 1x1 bandwidth, freely modulating acquisition rate, pixel format and Region-Of-Interest (ROI) of each ARIA based on the specific application needs. Refer to Section 8.6.6 for further details.

6 Trigger

ARIA supports asynchronous triggering, which allows capturing a specified number of images, controlling exposure start and duration through configurable events called "triggers".

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

- Acquisition-start trigger;
- Frame-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 ARIA are listed in Table 6.1

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 4.2.2).
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*

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;
```

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. When the module is enabled[3], the camera waits for a frame start trigger before starting an exposure.

When the exposure is complete and the required ROI has been acquired, the camera waits for a new *frame-start* trigger, that will trigger 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 start triggers are internally generated based on the value of `FrameRate` property.

6.4 END-OF-EXPOSURE TRIGGER

The *end-of-exposure* trigger (`ExposureEndTrigger`) event terminates the exposure of a frame. The only available trigger source for this module is `External`. When the module is disabled[5], the duration of the exposure is controlled by the Shutter control value (see Section 8.7.8).

The end-of-exposure trigger is only available for ARIA equipped with Sony sensors.

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

6.5 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 4.1.1.1).

Note

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

For *frame-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 4.21 a representation of the statement above is depicted in Figure 6.1

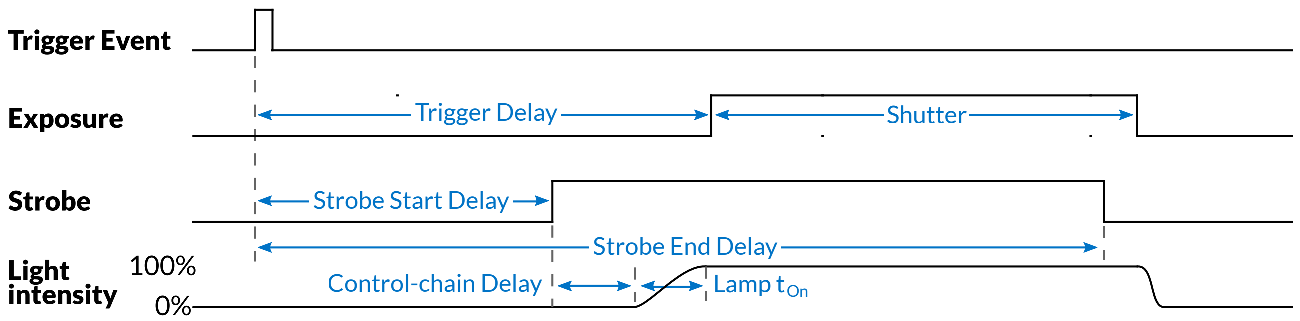


Figure 6.1: *Trigger delay with external light*

6.5.1 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.

6.5.2 Trigger overrun

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

- `AcquisitionStartTriggerReady`
- `FrameStartTriggerReady`
- `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 *frame-start* trigger-related signals.

The exposure begins after the specified trigger delay and a new trigger signal can be accepted only when `FrameStartTriggerReady` 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 `FrameStartTriggerReady` 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.

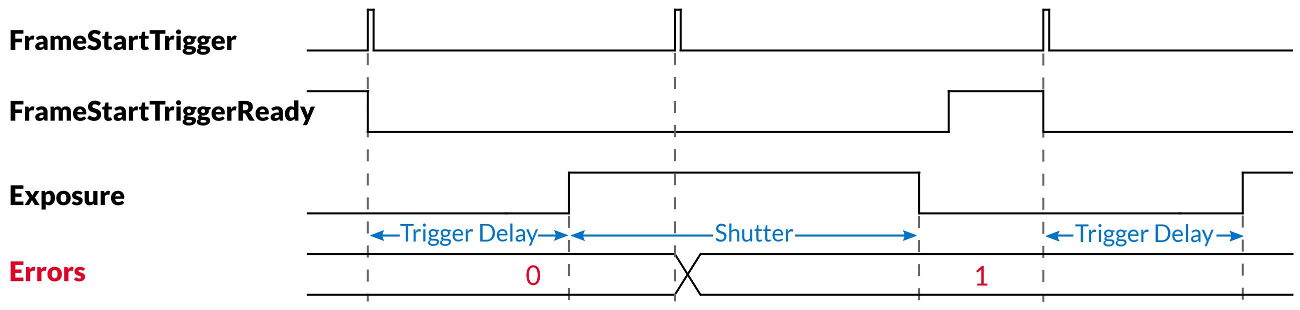


Figure 6.2: *Exposure timings*

6.6 TRIGGER CONFIGURATION EXAMPLE

6.6.1 External trigger frame acquisition

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

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.

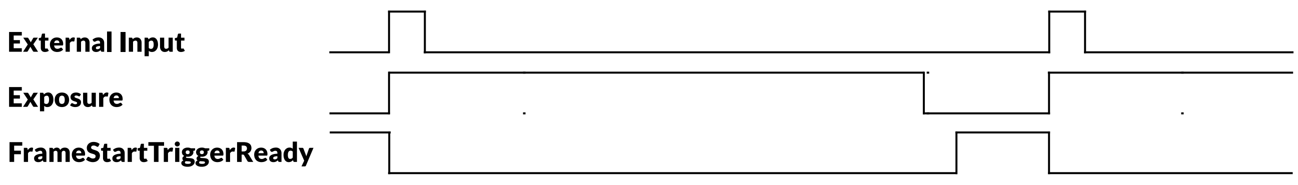


Figure 6.3: *External trigger frame acquisition timings*

Example Code 6.3 | *External trigger frame acquisition*

```
device.Shutter = 1000; // Set shutter time to 1 ms
device.FrameStartTrigger.Source=TriggerSource.External; // Select the external trigger
    source
device.FrameStartTrigger.ExternalInput = 1; //Select input port 1
device.PIOPorts[1].DebounceTime = 1; // Set debounce time to 1 us
device.FrameStartTrigger.DetectExternalInputEdge = true; // Detect an edge of the
    trigger source
device.FrameStartTrigger.InvertExternalInput = false;// Choose the rising edge of the
    trigger source
device.FrameStartTrigger.Enabled = true;//Enable frame trigger
device.PIOPorts[3].Source = OutputSource.FrameStartTriggerReady;// Select output port 3
    for the trigger ready signal
```

6.6.2 Signal-driven exposure time

The following example assumes that you want to acquire a frame 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. Frame start trigger ready signal is routed to output port 3, while `ExposureEndTriggerReady` signal is routed to output port 4.

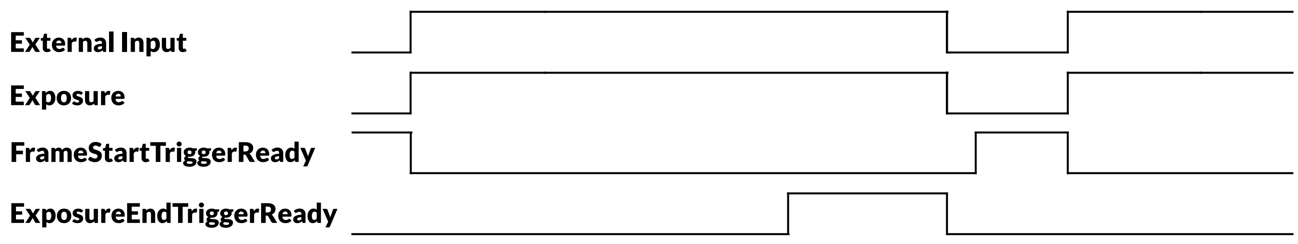


Figure 6.4: *Signal-driven exposure timings*

As shown in Figure 6.4, the `FrameStartTriggerReady` 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*

```
device.PIOPorts[1].DebounceTime = 1; // Set debounce time to 1 us
device.FrameStartTrigger.Source = TriggerSource.External;
device.FrameStartTrigger.ExternalInput = 1; // Select input port 1
device.FrameStartTrigger.DetectExternalInputEdge = true;
device.FrameStartTrigger.InvertExternalInput = false; // Rising edge
device.FrameStartTrigger.Enabled = true; // Enable frame trigger
device.PIOPorts[3].Source = OutputSource.FrameStartTriggerReady; // Trigger ready
device.ExposureEndTrigger.Source = TriggerSource.External;
device.ExposureEndTrigger.ExternalInput = 1; // Select input port 1
device.ExposureEndTrigger.DetectExternalInputEdge = true;
device.ExposureEndTrigger.InvertExternalInput = true; // Falling edge
device.ExposureEndTrigger.Enabled = true; // Enable frame trigger
device.PIOPorts[4].Source = OutputSource.ExposureEndTriggerReady; // Trigger ready
```

6.6.3 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 is acquired every 1000 steps of an encoder, whose signals are connected to ports 1 and 2.

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 FPGA, saving computational power on your PC and making it available for the application. Please refer to Section 8.7 for more information about camera controls such as white balance, digital gain, etc.

Figure 7.1 shows the FPGA image processing chain, referred to a color camera.

Figure 7.2 shows the FPGA image processing chain, referred to a monochrome camera.

7.1 LIGHT METER

ARIA evaluates the compensation parameters (white balance, luma) over an image area called *LMR (light meter ROI)*. 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 Look-Up Table (LUT).

Specifying the Light Meter ROI (LMR) requires setting its position and size through the `startX`, `startY`, `sizeX` and `sizeY` parameters[1].

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

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

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

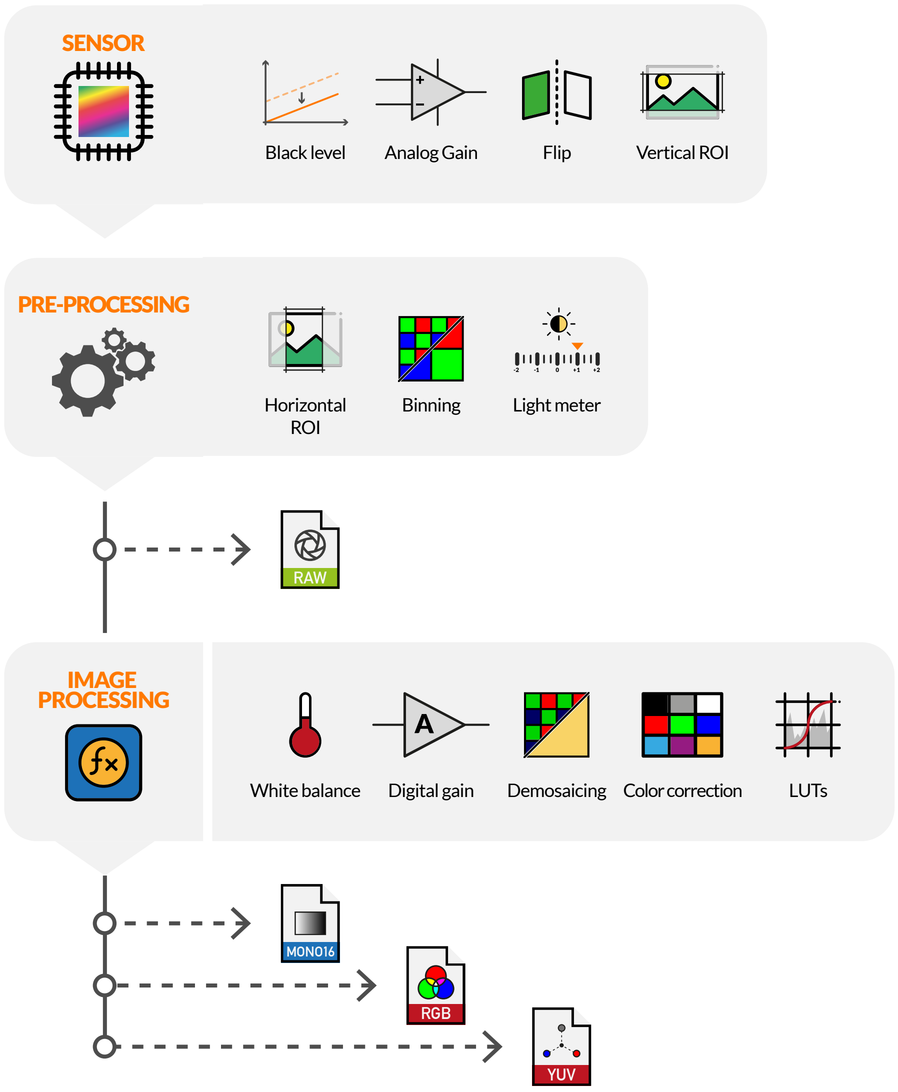


Figure 7.1: *Color camera processing chain diagram*

7.2 BLACK LEVEL OFFSET

The black level is the common offset exhibited by all pixels of the image sensor while not illuminated (see Section 9.6.3). It is inevitably inherent in the nature of sensitive elements and amplification chain; it is therefore important to keep it as low as possible to preserve the dynamics of the ADC and amplification chain.

ARIA cameras with e2V sensors 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.

Example Code 7.2 | *Set Black Level offset correction*

```
device.Calibration.BlackLevelOffset = 200;
```

ARIA cameras with IMX sensors feature automatic black level offset compensation. The `BlackLevel-`

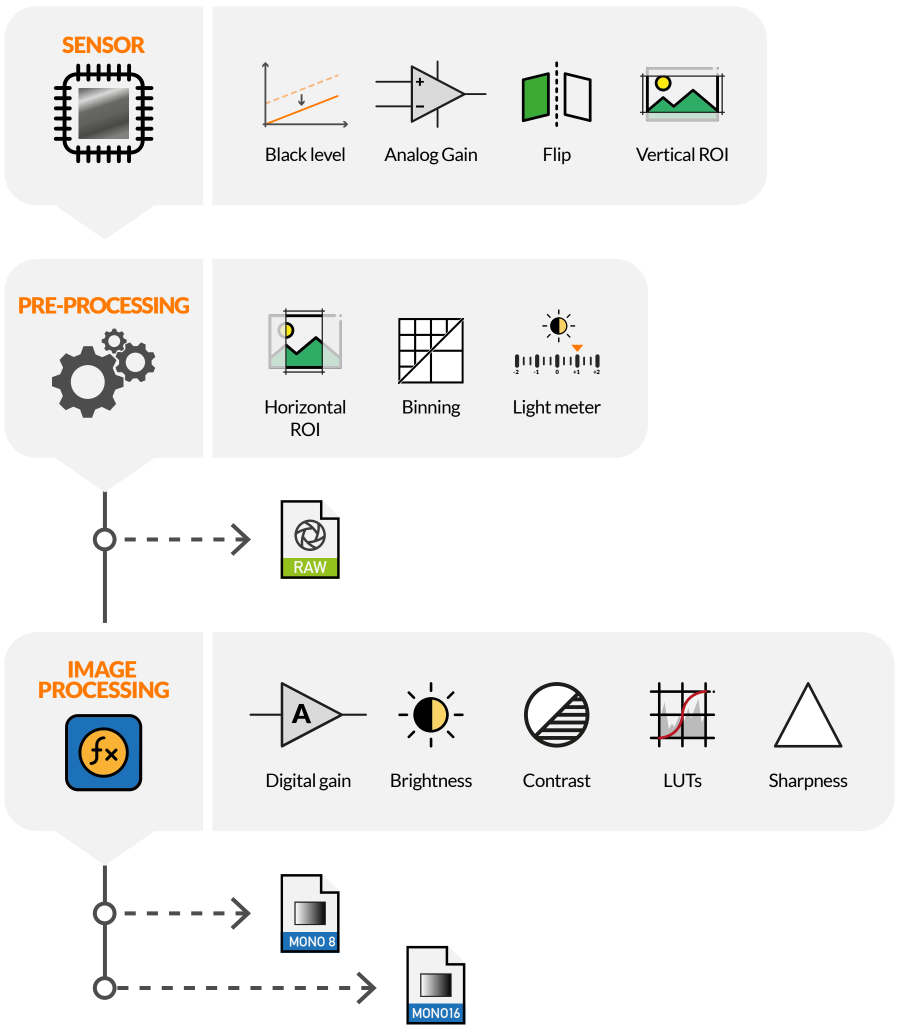


Figure 7.2: *Monochrome camera processing chain diagram*

`Offset` property is therefore not available on these models.

7.3 CHUNK DATA

ARIA cameras can optionally transmit a set of additional information along with the acquired image, that can be used for debugging purposes and data analysis. For instance, each frame can be identified by a progressive number or a time stamp and can also be associated to the encoder position and the inputs' status.

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

7.3.1 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 for each acquisition startup and counts from 0 to `Acquisition-BurstLength` – 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);
```

7.3.2 Time stamp

The `TimeStamp` is a 16-bit unsigned integer generated from a 1 ms timer and cleared when acquisition is started and captured at the frame-start trigger event.

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

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

7.3.2.1 Time stamp resolution

The timer resolution can be changed using the `TimeStampExponent` property. 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 valid values for the `TimeStampExponent` property can be retrieved using the `GetAvailable-TimeStampExponents` method.

7.3.3 Encoder position

The `EncoderPosition` is a 32-bit unsigned value indicating the position of the encoder as captured at the frame-start trigger event.

Example Code 7.5 | *Enable encoder position chunk data*

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

7.3.4 Input status

The `InputStatus` is a bitfield indicating the input status, captured at the frame-start trigger event.

Example Code 7.6 | *Enable input status chunk data captured at frame-start trigger event*

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

7.3.5 Burst number

The `BurstNumber` field is a 16-bit progressive unsigned number. If the acquisition-start trigger is active (see Section 6.1), the counter is incremented at each acquisition-start trigger.

Example Code 7.7 | *Enable burst number chunk data field*

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

7.3.6 Configuration example

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

```
device.EnableChunkData = true; // Enable chunk data transmission
device.SetEnabledChunkDataFields(new ChunkDataField[] {
    ChunkDataField.FrameNumber,
    ChunkDataField.FrameTimeStamp,
    ChunkDataField.FrameEncoderPosition,
    ChunkDataField.FrameInputStatus
});
device.Acquire = true; // Start acquisition
// Wait for a frame and extract chunk data
Pair<Bitmap, ChunkData> pair = device.GetImageChunk(true);
ChunkData data = pair.Second;
Debug.WriteLine("Frame number: " + data.FrameNumber);
Debug.WriteLine("Timestamp: " + data.TimeStamp);
Debug.WriteLine("Encoder position: " + data.EncoderPosition);
```

7.4 FRAME COMBINER

The *frame combiner* module allows to join a programmable number of frames together into a single picture. The frames are vertically stitched together in their chronological order.

7.4.1 Flush and AutoFlush

When the *frame combiner* module is active, the output picture becomes available only when all the requested input frames have been triggered and acquired. If the acquisition is stopped before a whole set of frames have been acquired, you can perform flush operation to retrieve a "partial/incomplete" picture, where the missing tiles are replaced by black images. A *flush* operation can be both manually invoked or automatically triggered programming an `auto-flush` timeout.

7.4.2 Configuration example

Example Code 7.9 | *Configure chunk data, combine 10 frames, set 1 second timeout, flush operation triggered*

```
device.FrameCombinerSize = 10; // Combines 10 frames into a bigger one
// If no frames are received after 1s, an auto flush operation is triggered:
// (set timeout to 0 to disable the timer)
device.FrameCombinerTimeout = 1000;
device.Acquire = true; // Start acquisition
device.FrameCombinerFlush(); // This manually triggers a flush operation
```

8
Capturing images

8.1 VIDEO MODES

Before starting acquiring with ARIA you must choose a *video mode*, associated with a specific configuration (ROI, pixel format, frame rate, etc.). Video modes allow selecting image resolution, region of interest (ROI) and output format.

ARIA video mode can be selected through the `VideoMode` property, as per the following example:

Example Code 8.1 | *Select camera video mode*

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

Warning

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 those pixel formats (see Section 8.4) that make 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.

8.2 REGION OF INTEREST (ROI)

If you don't need to acquire the whole sensor image, you can get just part of it setting a 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 `ImageStartX`, `ImageStartY`, `ImageSizeX` and `ImageSizeY` parameters as shown below):

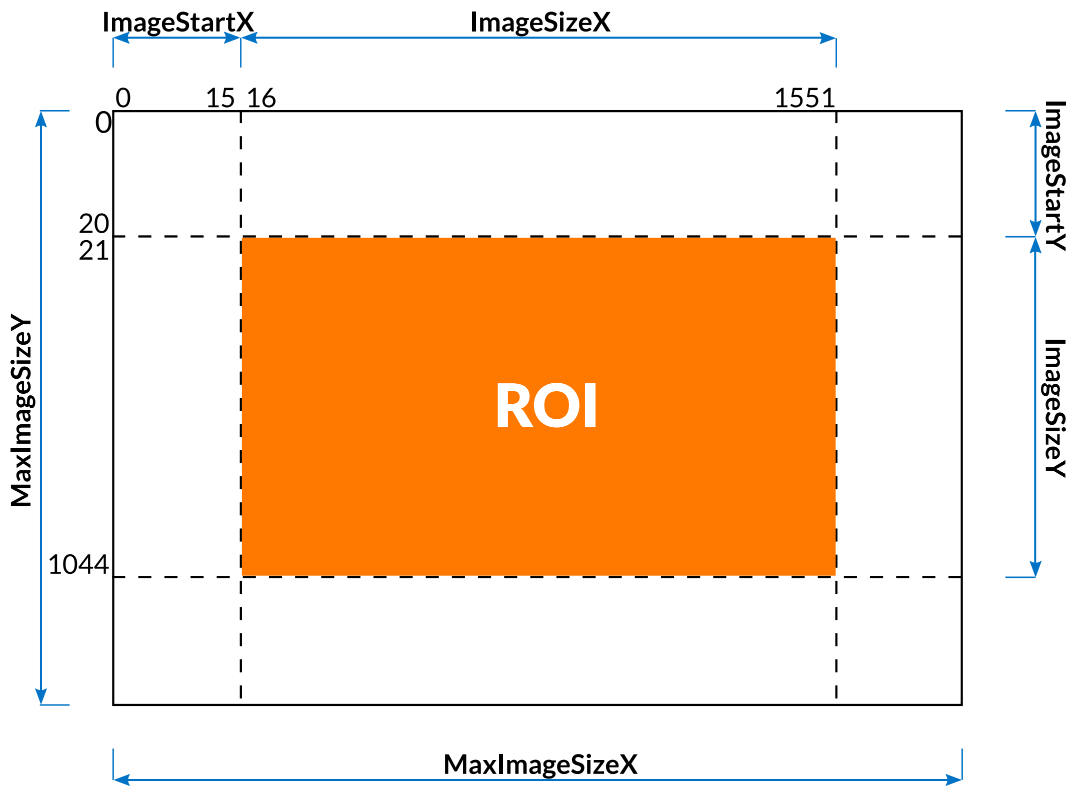


Figure 8.1: *ROI description*

Example Code 8.2 | *ROI configuring example*

```
// Ensures that ROI start position plus size does not exceed max sensor area
device.ImageStartX = 0;
device.ImageStartY = 0;
// Set ROI size
device.ImageSizeX = 1536;
device.ImageStartX = 16;
// Set ROI starting position
device.ImageSizeY = 1024;
device.ImageStartY = 21;
```

Warning

ROI parameters can be set only when camera is not acquiring.

Note

To avoid errors, please care that `ImageStartX + ImageSizeX` does not exceed `MaxImageSizeX`. The same applies also to `ImageStartY` and `ImageSizeY`.

Note

Changing ROI parameters involves automatic recomputing of the *packet size* boundaries (see Section 8.6.4).

8.3 BINNING

ARIA camera has the ability to digitally sum two or more adjacent pixels into one pixel. This procedure increases the sensor response in cases of low lighting. Binning is a digital aggregation that is applied after the analog to digital conversion. Please refer to Section 2.2.1 for further information about available binning modes.

Two types of binning are available: *vertical binning* and *horizontal binning*.

The *vertical binning* mode combines the values of adjacent pixels on the same column (see 8.2), and the vertical size of the acquired image is then reduced by a factor related to the chosen binning factor. For example, starting from the ROI position depicted in 8.1, a vertical binning factor of 2 would reduce the frame height to 512 lines and image vertical start position to 11, as shown below:

$$Y_{resolution_{Bin}} = \frac{Y_{resolution_{NoBin}}}{VB} \tag{8.1}$$

$$SizeY_{Bin} = \frac{SizeY_{NoBin}}{VB} = 512 \tag{8.2}$$

$$StartY_{Bin} = \frac{StartY_{NoBin}}{VB} = 11 \tag{8.3}$$

Similarly, the *horizontal binning* mode combines the values of adjacent pixels on the same line (see Figure 8.3), and the horizontal lenght of the acquired image is then reduced by a factor related to the chosen binning factor. For example, starting from the same position, a horizontal binning factor of 2 would produce the following values:

$$X_{resolution_{Bin}} = \frac{X_{resolution_{NoBin}}}{HB} \tag{8.4}$$

$$SizeX_{Bin} = \frac{SizeX_{NoBin}}{HB} = 768 \tag{8.5}$$

$$StartX_{Bin} = \frac{StartX_{NoBin}}{VB} = 8 \tag{8.6}$$

All values will be truncated to the nearest lower integer.

The *combined binning* mode enables both binning modes (see Figure 8.4) and generates an image whose resolution is reduced in both dimensions.

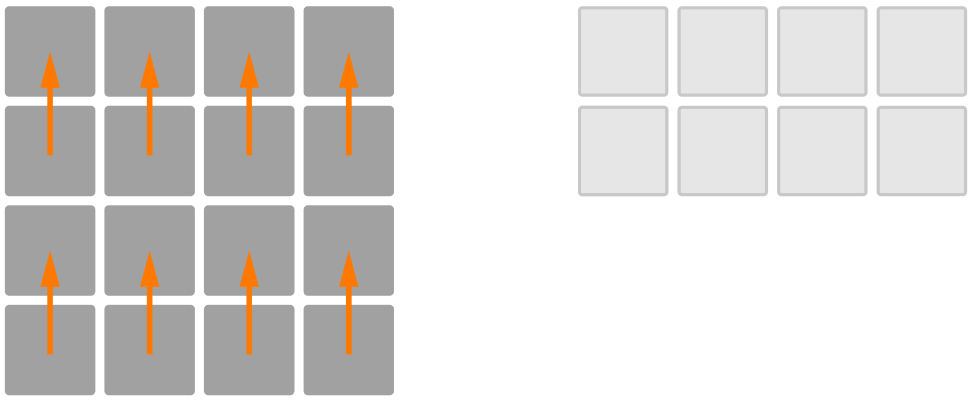


Figure 8.2: *Monochrome vertical binning*

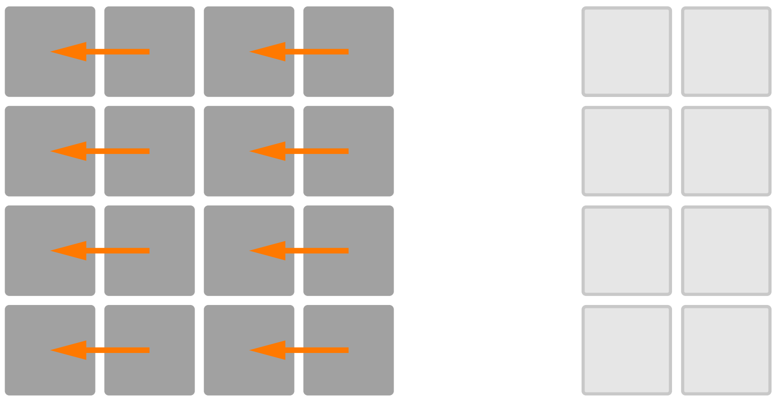


Figure 8.3: *Monochrome horizontal binning*

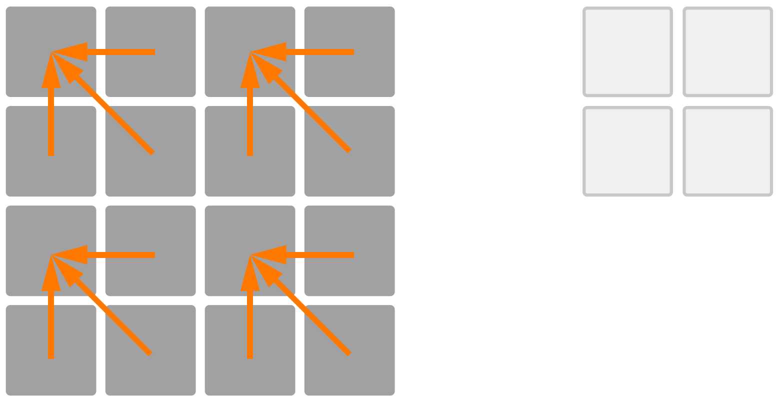


Figure 8.4: *Monochrome combined binning*

Binning is also available in color cameras: pixel values for identical colors are vertically and/or horizontally summed.

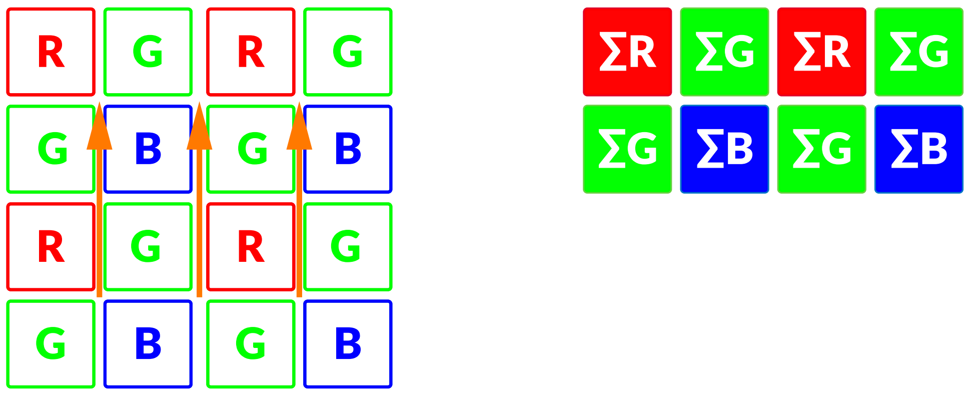


Figure 8.5: *Color vertical binning*

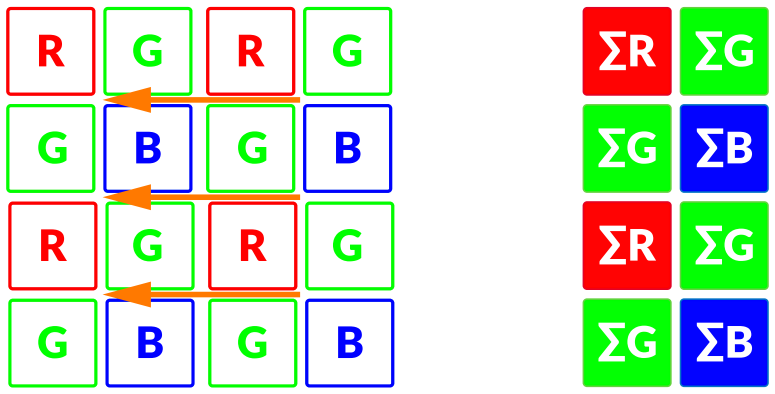


Figure 8.6: *Color horizontal binning*

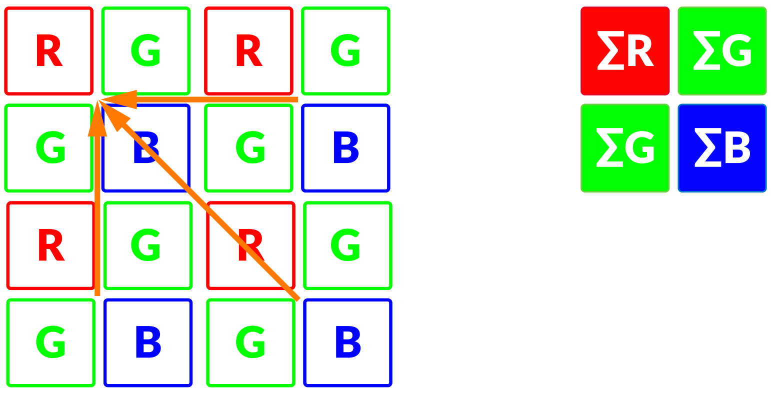


Figure 8.7: *Color combined binning*

Example Code 8.3 | *Set combined binning mode*

```
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 8.4 | *Disable binning mode*

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

Note

When enabling horizontal binning, `ImageStartX` and `ImageSizeX` values will be reduced by the corresponding binning factor. *Vertical binning* mode affects `ImageSizeY` and `UnitImageSizeY` as well.

8.4 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	ARIA Color model	ARIA Monochrome model	Video Mode	Bytes per pixel
MONO8	N.A.	✓	0	1
MONO16	✓	✓	0	2
YUV422	✓	N.A.	0	2
RGB24	✓	N.A.	0	3
RAW8	N.A.	✓	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 bits 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 represented in *little endian*.

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*, reconstructing the actual color of each pixel sampled according to the Bayer filter (see Figure 3.4). The RGB24 format uses 3 bytes (24 bits) 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 bytes for luminance (one per pixel) and 2 bytes 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 ARIA cameras, please consider that the camera output will be a bayer filtered image.

Example Code 8.5 | *Set pixel format to RGB24*

```
device.ColorCoding = ColorCoding.RGB24;
```

Warning

Pixel format can be set only when camera is not acquiring.

Note

Changing pixel format involves automatic recomputing of the packet size boundaries (see Section 8.6.4).

8.5 ADC RESOLUTION

All CMOS camera sensors incorporate an on-chip ADC to digitize the pixel values. Each sensor family features different ADC technology and their resolution may differ.

The *ADC resolution* influences the conversion speed, i.e. the time it takes for the ADC to convert is proportional to the selected *ADC resolution* and affects the maximum frame rate. The *ADC resolution* may hence be reduced to speed up image acquisition. Please refer to the detailed specification tables (Section 2.2).

When the maximum allowed *ADC resolution* is selected, ARIA 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 codings) 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* uses a lower number of bits to represent pixel values so that higher frame rates can be achived 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.

Example Code 8.6 | *Set* `ADCResolution` *property to 10 bits*

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

Warning

ADC resolution can be set only when camera is not acquiring.

Note

Changing *ADC resolution* involves automatic recomputing of the *packet size* boundaries (see Section 8.6.4).

8.6 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. ARIA 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.

8.6.1 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 ARIA cameras throughput, refer to Section 2.5.

8.6.2 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 8.7 | *Sets a bandwidth limits of 32 KiB per microframe on ARIA*

```
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.

8.6.3 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.

8.6.4 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*, *Analog to Digital Converter (ADC) resolution*, *pixel format*, etc.) influencing both acquisition speed and size of data to be sent. Refer to Section 8.6.4.1 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 8.6.2.

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 8.8 | *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.

8.6.4.1 Calculating the required bandwidth

Whenever an operating parameter of the camera is set, ARIA 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 ARIA 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:

$$\mu_B = (I_B + CD_B) \cdot R \cdot \frac{125\mu s}{1s} \tag{8.7}$$

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 7.3). 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:

$$R = \frac{f_{SENSOR}}{W \cdot ROI_H} \tag{8.8}$$

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.

8.6.5 Packet size for bulk endpoint

Similarly to the bandwidth limit (Section 8.6.3), 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.

8.6.6 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 ARIA 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.

8.6.7 Frame rate

The frame rate parameter controls the acquisition period, i.e. the time between the acquisition of two consecutive images, and is expressed in frames per second (Hz).

The *frame rate* control has effects only when the *frame start trigger* is disabled. Otherwise, the time between two consecutive acquisitions is determined by the trigger frequency. ARIA provides two *read-only* variables reporting the allowed range (upper and lower limits) for the *frame rate*. The lower limit is constant (1 fps), the upper limit depends on the available bandwidth (see Section 8.6.4.1): any direct or indirect change of the *packet size* control involves recomputing the maximum frame rate.

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

Note

When changing one of the controls affecting bandwidth (see the complete list in Section 8.6.4.1), the upper *frame rate* limit is recomputed and the current *frame rate* may automatically be reset to the maximum allowed value. Please see Section 8.6.8 for further details.

Note

The actual *frame rate* can be lower than the requested one when the current shutter time exceeds the acquisition period .

In general, the maximum value for the *frame rate* control is determined by three factors:

1. time required for converting and transferring the image from the sensor to the camera;
2. bandwidth available on the USB 3.2 Gen 1x1 bus;
3. size of the selected ROI.

To achieve the largest *frame rate* allowed by the application (see also Section 8.6.9), please:

1. decrease the *ADC resolution* (decreasing the conversion time for each frame);
2. ensure that the *shutter* time is not such large as to affect (increase) the minimum acquisition period;
3. set the optimal *packet size*, i.e. the maximum allowed given current operating conditions;
4. use the lowest ROI size compatibile with the application.

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

```
// Set 30 fps, without exceeding the maximum allowed value.
device.FrameRate = Math.Min(device.MaxFrameRate, 30);
```

8.6.8 Preserve rates

As mentioned before, ARIA updates the upper *frame rate* limit when the current *packet size* is modified. This happens either upon direct request (e.g. `device.PacketSize = 32`) or when changing parameters affecting *packet size* limits (see Section 8.6.4.1).

Whenever the upper *frame rate* limit is updated, ARIA can increase the current *frame rate* value to that value in order to guarantee the maximum available acquisition speed, or the current *frame rate* value can be preserved.

This behavior is controlled by the `PreserveRates` property. When true, ARIA keeps the previously frame rate 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 *frame rate* will always be maximized.

Example Code 8.10 | *Enable the* `PreserveRates` *functionality*

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

Note

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

8.6.9 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;

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 ARIA cameras at once on a single PC, achieving sufficient performances for most applications and minimizing the overall hardware cost.

8.7 CONTROLS

ARIA 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

8.7.1 Brightness

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

Example Code 8.11 | *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.

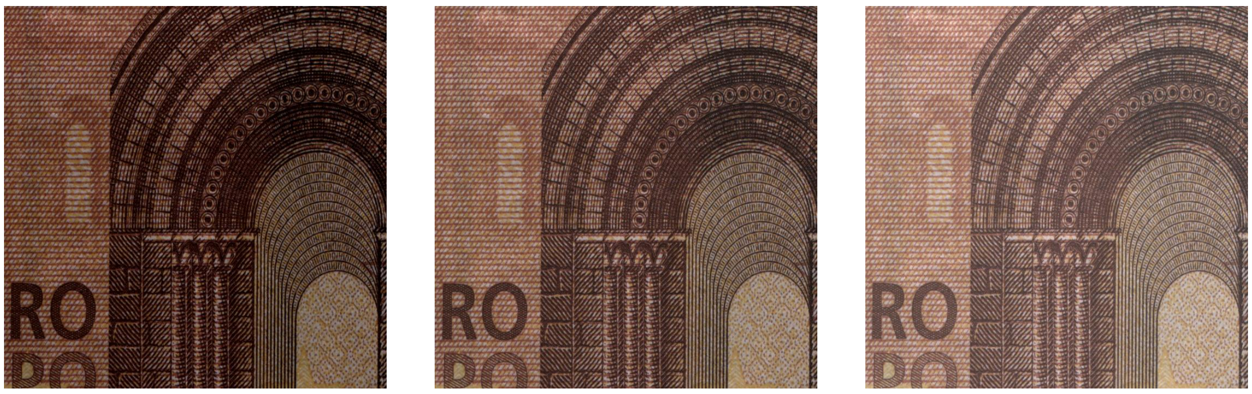

(a) *Brightness 64* **(b)** *Brightness 128 (neutral)* **(c)** *Brightness 192*

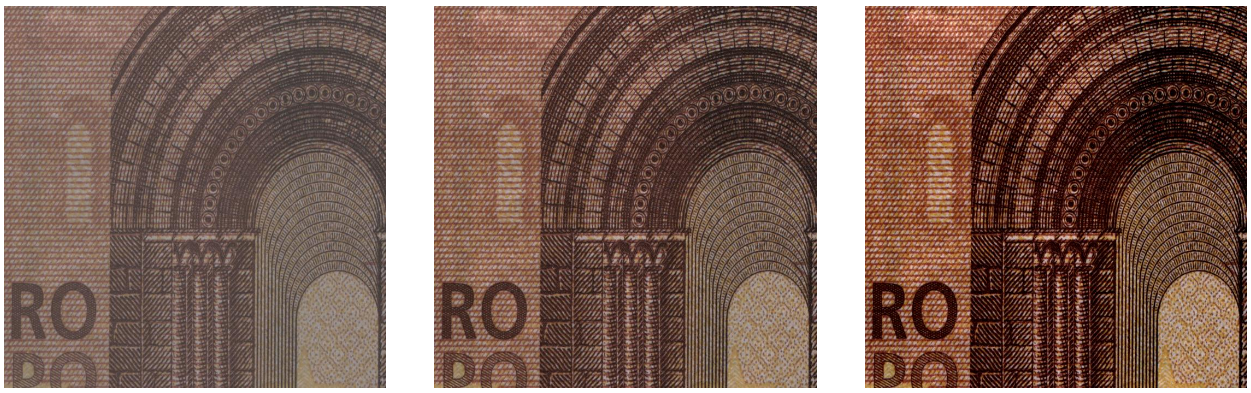
Figure 8.8: *Example of* `Brightness` *control effect.*

8.7.2 Contrast

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

Example Code 8.12 | *Set the contrast level*

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



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

***Figure 8.9:** Example of* `Contrast` *control effect.*

8.7.3 Color correction matrix (color models only)

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

$$\begin{bmatrix} R' \\ G' \\ B' \end{bmatrix} = CCM \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix} + CCO \tag{8.9}$$

$$CCM = \begin{bmatrix} k_{RR} & k_{GR} & k_{BR} \\ k_{RG} & k_{GG} & k_{BG} \\ k_{RB} & k_{GB} & k_{BB} \end{bmatrix} \tag{8.10}$$

$$CCO = \begin{bmatrix} O_R \\ O_G \\ O_B \end{bmatrix} \tag{8.11}$$

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

$$CCM_{Default} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{8.12}$$

$$CCO_{Default} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{8.13}$$

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 8.13 | *Set the color correction matrix coefficients*

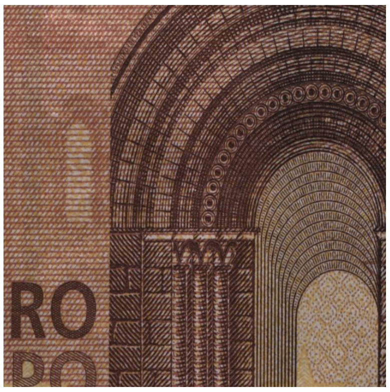
```
if (device.ColorCorrectionMatrixAvailable)
{
    float[,] matrix = new float[3, 3];
    // Populate coefficient matrix
    matrix[0, 0] = (float)kRR;
    matrix[0, 1] = (float)kGR;
    matrix[0, 2] = (float)kBR;
    matrix[1, 0] = (float)kRG;
    matrix[1, 1] = (float)kGG;
    matrix[1, 2] = (float)kBG;
    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;
}
```

8.7.4 Hue (color models only)

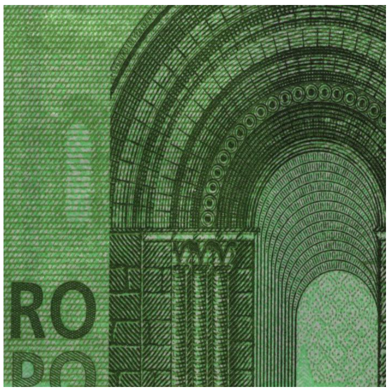
The Hue control modifies the dominant color of the acquired image to provide color enhancement.

Example Code 8.14 | *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)

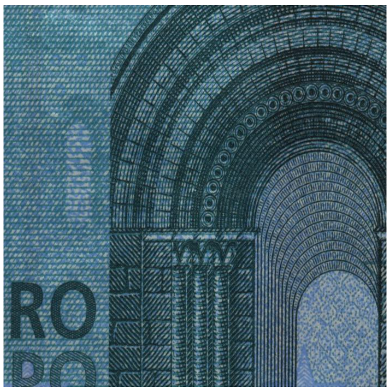

(d) *Hue 128* (180 deg)

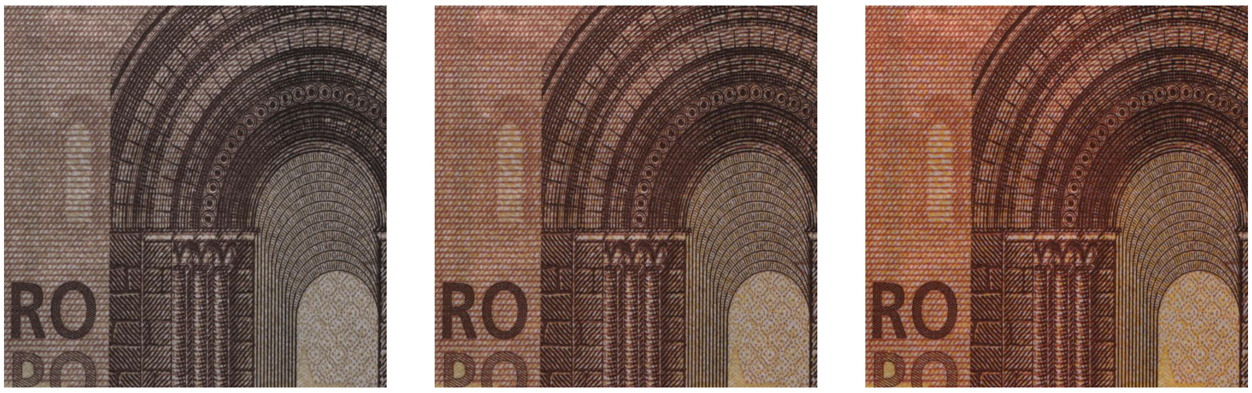
Figure 8.10: *Example of* `Hue` *control effect.*

8.7.5 Saturation (color models only)

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

Example Code 8.15 | *Set the saturation level*

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



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

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

8.7.6 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 8.16 | *Set white balance automatically*

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

Example Code 8.17 | *Set white balance manually*

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

8.7.7 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:

$$P' = C_{Max} \cdot \left(\frac{P}{O_{Max}} \right)^{\gamma} \tag{8.14}$$

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 8.12) and $O_{Max} = 2^{ADC\_resolution-1}$ is the maximum achievable value of original pixel (e.g. 2047 for curves in Figure 8.12).

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

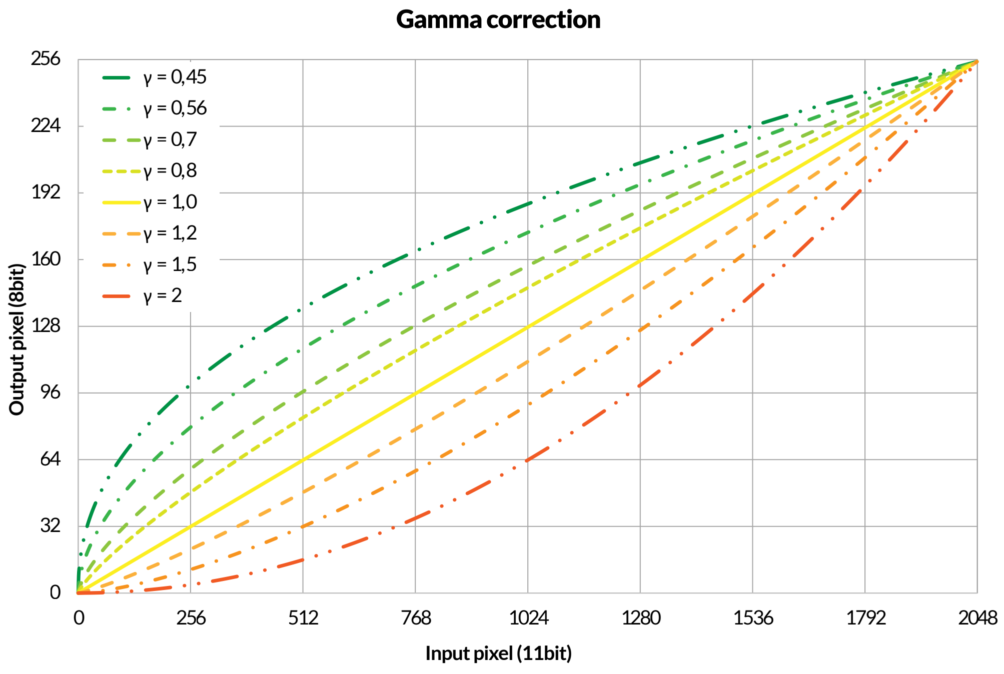


Figure 8.12: *Gamma correction representations*

Note

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

Example Code 8.18 | *Set gamma*

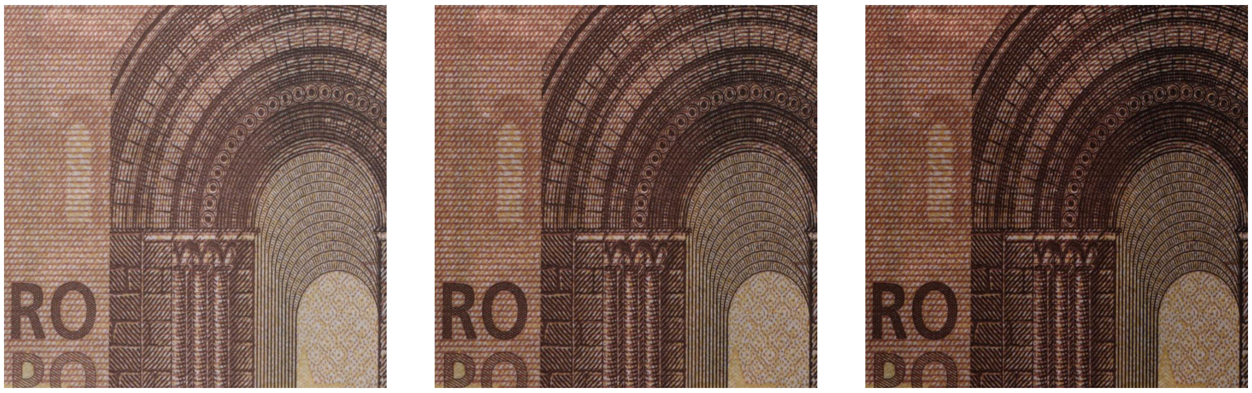
```
UnitInfo unit = device.GetFeatureUnitInfo(Feature.Gamma);
double gamma = 1.23;
device.Gamma = (uint)(gamma / unit.Factor);          // Convert to Gamma units
device.GammaEnabled = true;                      // Enable Gamma Correction
```

Warning

User LUT and gamma correction share the same processing chain resources and cannot be used simultaneously. At power-on, ARIA 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 ARIA could ignore external triggers and/or lose frames.



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

Figure 8.13: *Example of* `Gamma` *control effect.*

8.7.8 Shutter

The `Shutter` control adjusts the exposure time of the frames acquired by the camera. The following example sets the duration of the exposure time to 100 μs:

Example Code 8.19 | *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 frame period decreases the actual frame rate. Consequently, current frame period becomes slightly larger[a] than the currently selected exposure time.

[a] due to analog-to-digital conversion time

MaestroUSB3 viewer and application examples show in detail how to use the Shutter control and allow to experience the interactions with frame rate.

8.7.8.1 High resolution shutter

Due to the sensor architecture, the shutter time is set with limited precision and it is rounded to meet the time constraint of the sensor in use. The camera has a read-only property returning the actual duration of the resulting exposure.

Example Code 8.20 | *Get the exposure length, expressed in seconds*

```
float shutter = device.HighResShutter;  // Read the actual exposure in seconds
```

8.7.8.2 Shutter resolution

As stated in Section 8.7.8.1, actual exposure time may be slightly different from the value requested by your application.

The camera is able to change the shutter time in steps whose duration depends on several parameters (`PacketSize`, `ShutterFloor`, `DisableOverlap`). The shutter resolution is a read-only control that lets you know the actual duration of these steps. A smaller value means you can approximate the desired shutter time with greater precision.

8.7.8.3 Shutter floor

As with all other controls, there is a maximum and minimum value for the shutter control. Its minimum value depends on several factors, including *ADC resolution* and available bandwidth (*packet size*). It happens as meeting the user-selected *packet size* may require slowing down the sensor data transmission rate at which it transmits data; due to sensor's internal time constraint, slowing data transfer may affect the minimum allowed shutter time.

In some applications, you may need smaller shutter times. This control can be used to set the sensor so that these shutter times can be reached.

Please note that after reducing the minimum shutter time, the sensor's ability to adapt its data rate to the selected transmission bandwidth (*packet size*) is limited: as a side effect of this setting, in some configurations the maximum reached *frame rate* may be reduced.

8.7.8.4 Automatic exposure

ARIA *automatic exposure* control allows automatically setting the shutter to achieve a specific average image intensity (*luma*). In order to use this feature, you must specify a *setpoint* (the target *luma*) and enable *automatic exposure*. The camera will calculate the appropriate *shutter*, frame by frame, to keep the *luma* at the required *setpoint*.

You can also limit the maximum *shutter* value set by the routine to avoid excessive *frame rate* slowdown.

Example Code 8.21 | *Set up automatic exposure*

```
// Check if automatic exposure is available
if (device.GetFeatureAutoCapability(Feature.Shutter))
{
    device.AutoExposureRef = 128; // Set the desired luma target
    device.AutoExposureLimit = 3333; // Limit the maximum shutter to 33.33ms (30fps)
    device.AutoExposure = true; // Enable auto exposure feature
}
```

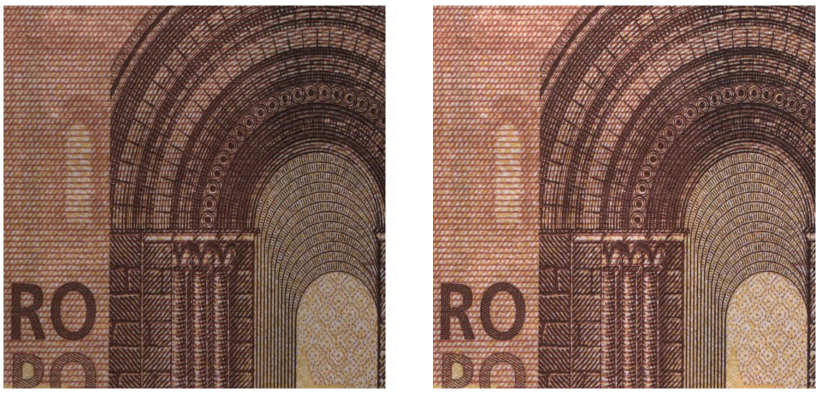
A typical *automatic exposure* setup procedure includes the following steps: issue an initial *manual* camera setup (*shutter*, *gain*, etc.) first, so that the resulting image has the desired brightness level; invoke Auto-ExposureRefSetPoint, updating the AutoExposureRef property with the current *luma* value. The following code explain how to take advantage of this method:

Example Code 8.22 | *Set up automatic exposure*

```
// After reaching the desired image brightness you can invoke the following commands:
device.AutoExposureRefSetPoint();
device.AutoExposure = true;
```

8.7.9 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*

Figure 8.14: *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.

8.7.10 Digital Gain

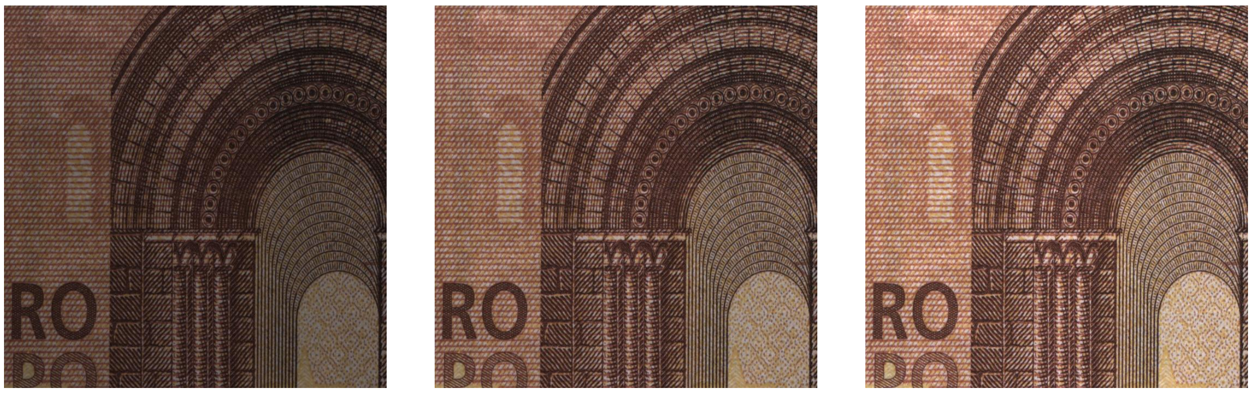
When, despite using the sensor analog amplifiers (Gain control), the resulting images cannot achieve the desired brightness level, you can enable a digital amplifier module scaling the acquired signal up.

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 8.23 | *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 8.15: *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.

8.7.11 User LUTs

ARIA 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, ARIA 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 8.24 | *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 ARIA could ignore external triggers and/or lose frames.

8.7.12 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 8.25 | *Read the Luma value of the last image*

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

Note

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

8.7.13 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 8.26 | *Read the TimeStamp value*

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

8.7.14 Flip and Rotate

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

Example Code 8.27 | *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 8.28 | *Rotate the image of 270 deg*

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

8.8 SAVING DEVICE CONFIGURATION

You can save the status of ARIA 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.

ARIA 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 8.29 | *Retreive the maximum number of user configurations*

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

Example Code 8.30 | *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 8.31 | *Erase configuration*

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

8.8.1 Export and import XML

ARIA 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 8.32 | *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 8.33 | *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);
```

8.8.2 User configuration (deprecated)

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

Example Code 8.34 | *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 8.35 | *Erase user configuration from camera memory*

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

8.9 CONFIGURATION EXAMPLE

Example Code 8.36 | *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
```

8.10 PATTERN GENERATOR

ARIA 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 8.37 | *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 ARIA 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.

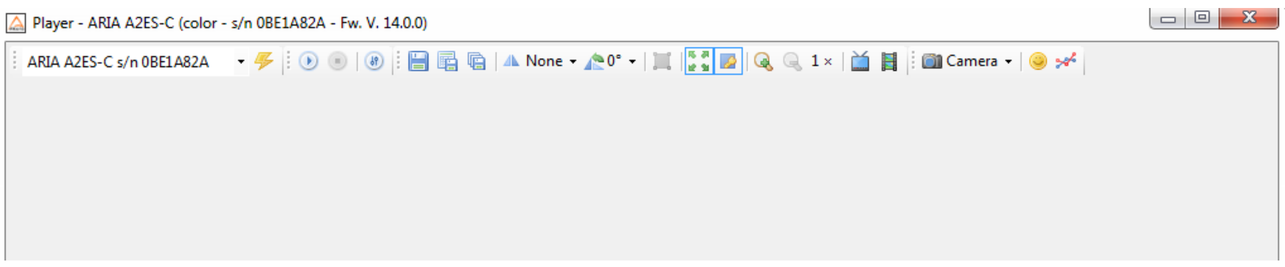


Figure 9.1: *Alkeria player main window*

9.1 DEVICE CONNECTION AND INITIALIZATION

The top left toolbar is a drop-down menu letting you select a specific ARIA camera among the ones connected to the system. When no ARIA 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 3.7.7.
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:

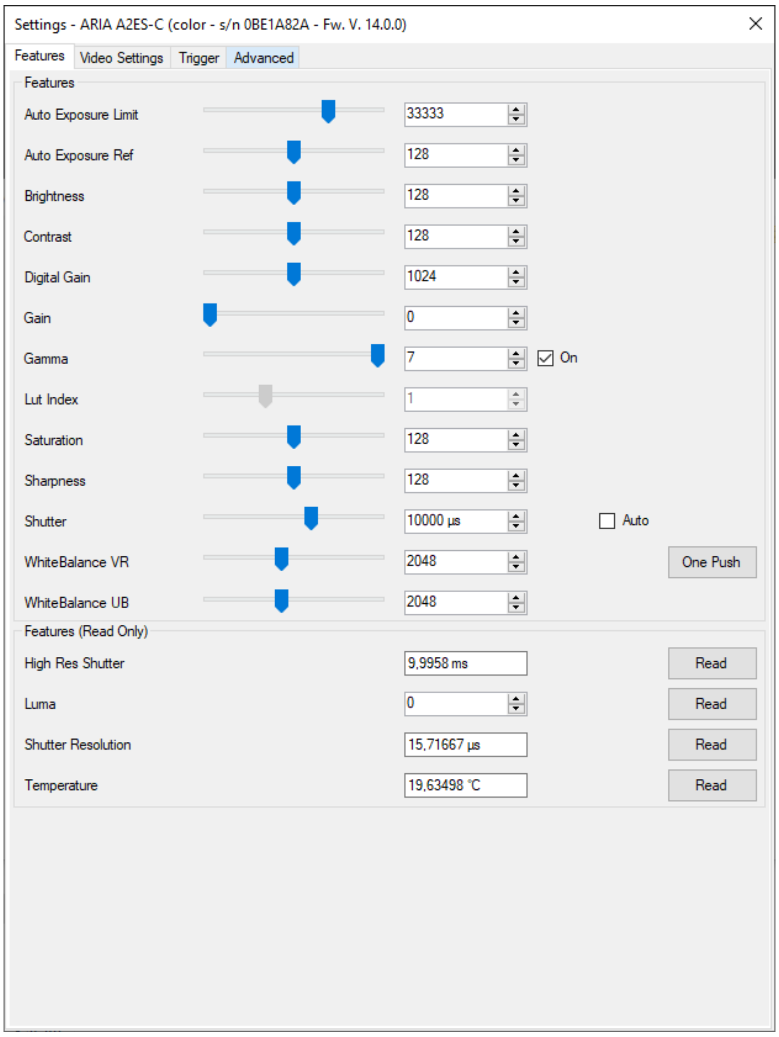
- `Shutter`: defines the exposure time, i.e. the time span in which the sensor is exposed to light;
- `Gamma`: applies a non-linear transformation to the image, defined by the relationship $I_{out} = I_{in}^{\gamma}$, where I is the pixel luminosity;
- `Gain`: controls the analog gain on sensor's circuitry;

- `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.


Settings - ARIA A2ES-C (color - s/n 0BE1A82A - Fw. V. 14.0.0)

Features | Video Settings | Trigger | Advanced

Features

Auto Exposure Limit	33333	
Auto Exposure Ref	128	
Brightness	128	
Contrast	128	
Digital Gain	1024	
Gain	0	
Gamma	7	☑ On
Lut Index	1	
Saturation	128	
Sharpness	128	
Shutter	10000 µs	☐ Auto
WhiteBalance VR	2048	One Push
WhiteBalance UB	2048	

Features (Read Only)

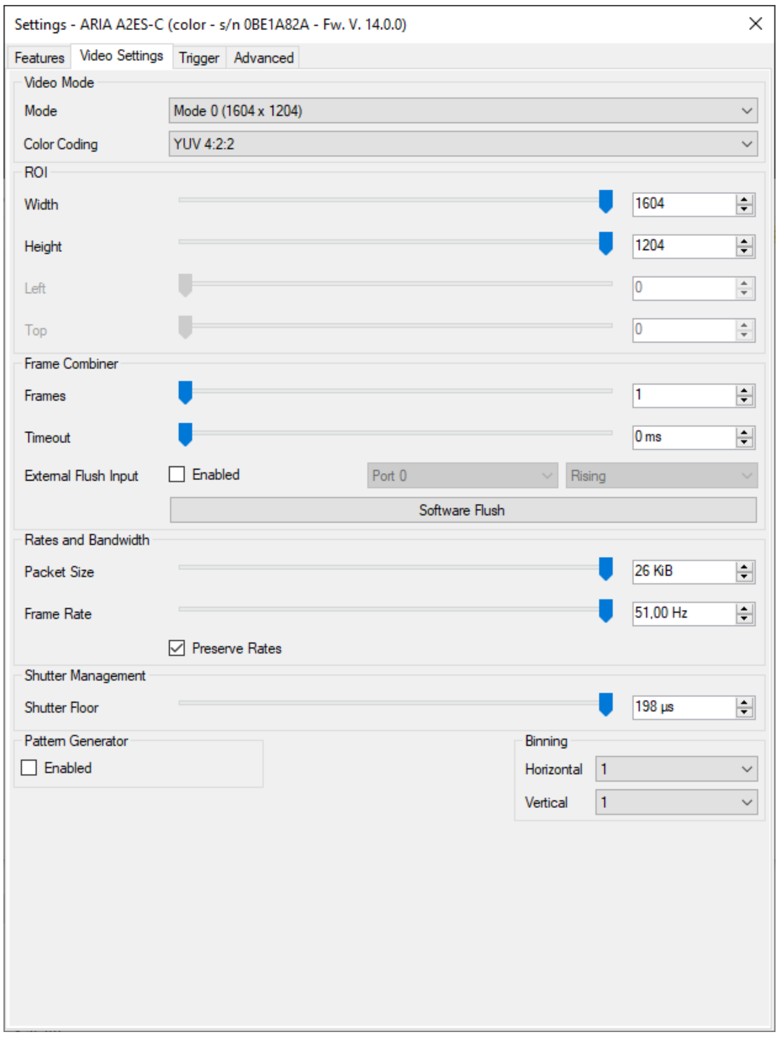
High Res Shutter	9,9958 ms	Read
Luma	0	Read
Shutter Resolution	15,71667 µs	Read
Temperature	19,63498 °C	Read

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;



Settings - ARIA A2ES-C (color - s/n 0BE1A82A - Fw. V. 14.0.0)

Video Mode
Mode: Mode 0 (1604 x 1204)
Color Coding: YUV 4:2:2

ROI
Width: 1604
Height: 1204
Left: 0
Top: 0

Frame Combiner
Frames: 1
Timeout: 0 ms
External Flush Input: ☐ Enabled, Port 0, Rising

Rates and Bandwidth
Packet Size: 26 KiB
Frame Rate: 51,00 Hz
☑ Preserve Rates

Shutter Management
Shutter Floor: 198 µs

Pattern Generator
☐ Enabled

Binning
Horizontal: 1
Vertical: 1

Figure 9.3: *Video settings tab*

9.3.3 Trigger

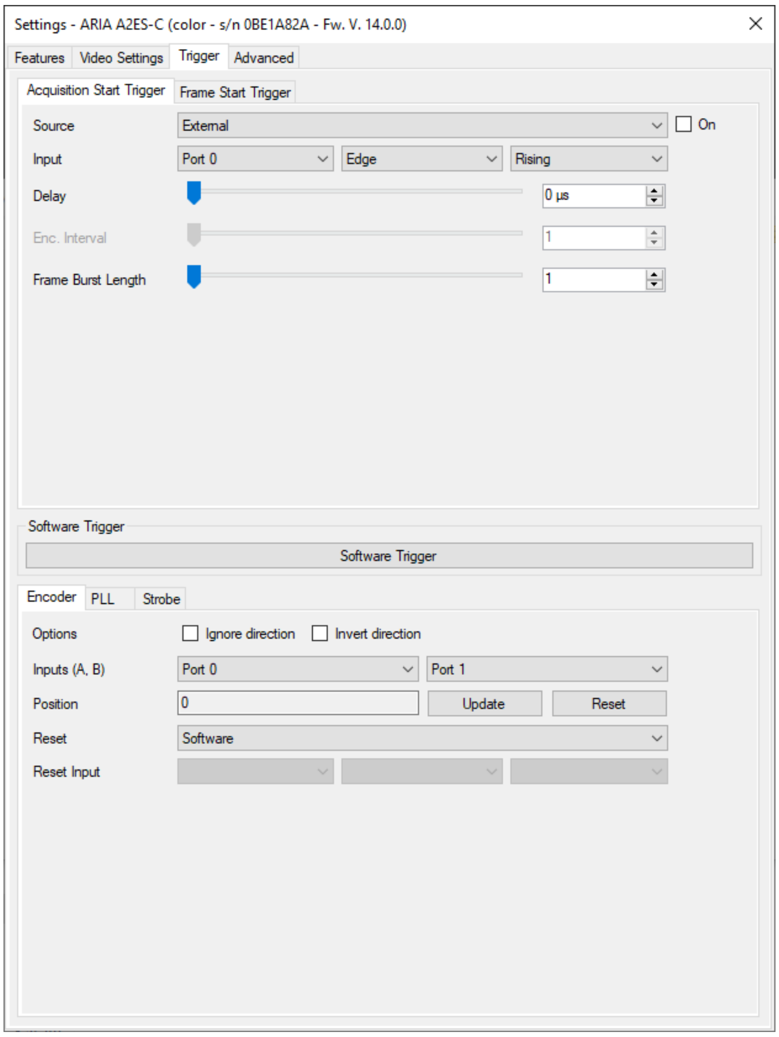
The Trigger panel (Figure 9.4) allows to setup the Trigger Manager module. Various events can be triggered, either via software or external sources (I/O):

- Acquisition start;
- Frame start;

- 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 4.2.1) and PLL (Frequency Multiplier, see Section 4.2.2) 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).


Settings - ARIA A2ES-C (color - s/n 0BE1A82A - Fw. V. 14.0.0)

Source	External		☐ On
Input	Port 0	Edge	Rising
Delay	0 µs		
Enc. Interval	1		
Frame Burst Length	1		

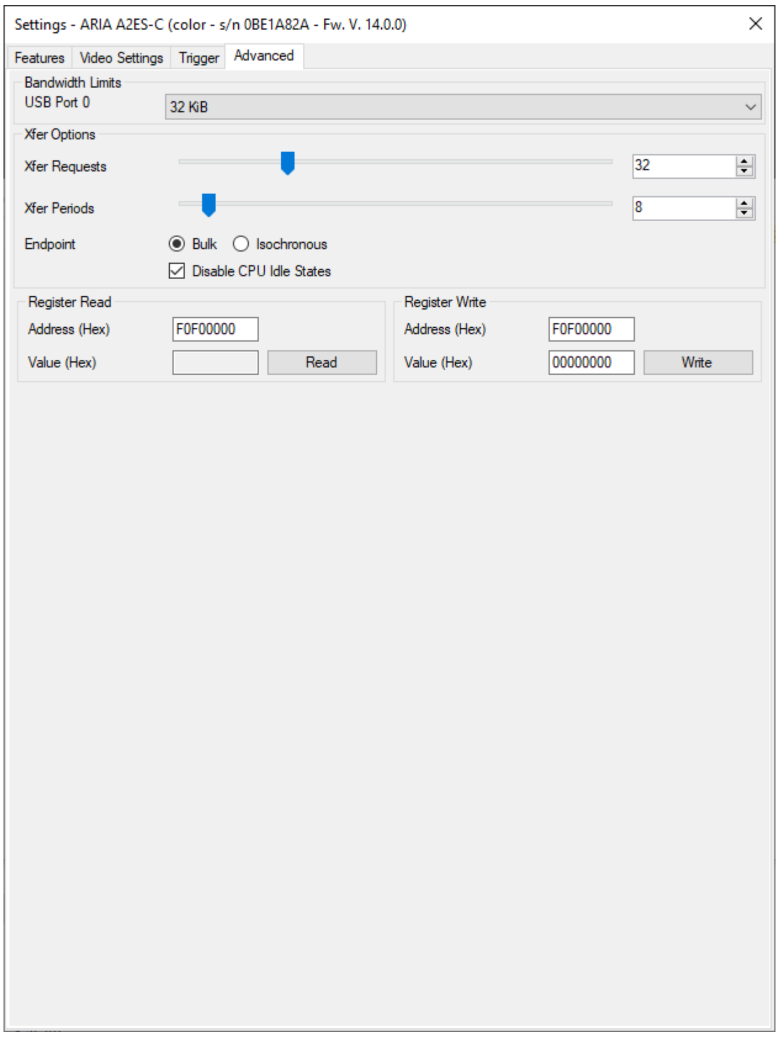
Options	☐ Ignore direction	☐ Invert direction	
Inputs (A, B)	Port 0	Port 1	
Position	0	Update	Reset
Reset	Software		
Reset Input			

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 8.6).


Settings - ARIA A2ES-C (color - s/n 0BE1A82A - Fw. V. 14.0.0)

Bandwidth Limits
USB Port 0: 32 KiB

Xfer Options
Xfer Requests: 32
Xfer Periods: 8
Endpoint: [✓ Bulk] Isochronous
[✓] Disable CPU Idle States

Register Read
Address (Hex): F0F00000
Value (Hex):

Register Write
Address (Hex): F0F00000
Value (Hex): 00000000

Figure 9.5: *Advanced settings tab*

Note

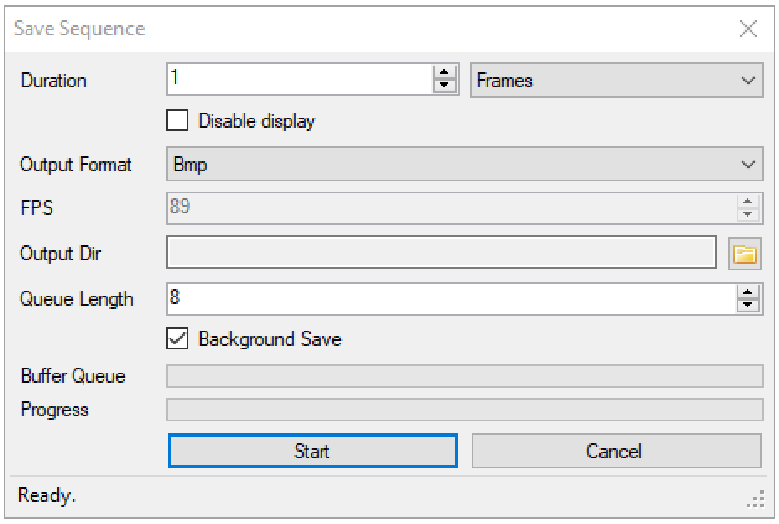
Not all host controllers support the maximum theorical speed. For this reason the camera has a default startup limit of 32 KiB per micro-frame you can override according to your setup.

Note

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

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).


Save Sequence

Duration: 1 Frames
☐ Disable display
Output Format: Bmp
FPS: 89
Output Dir:
Queue Length: 8
☑ Background Save
Buffer Queue
Progress

Ready.

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

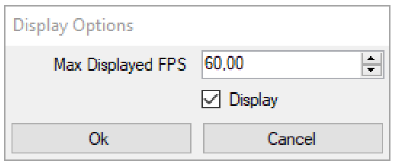
High Queue Length values may be incompatible with some hardware configurations.

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)

9.6 CAMERA MENU

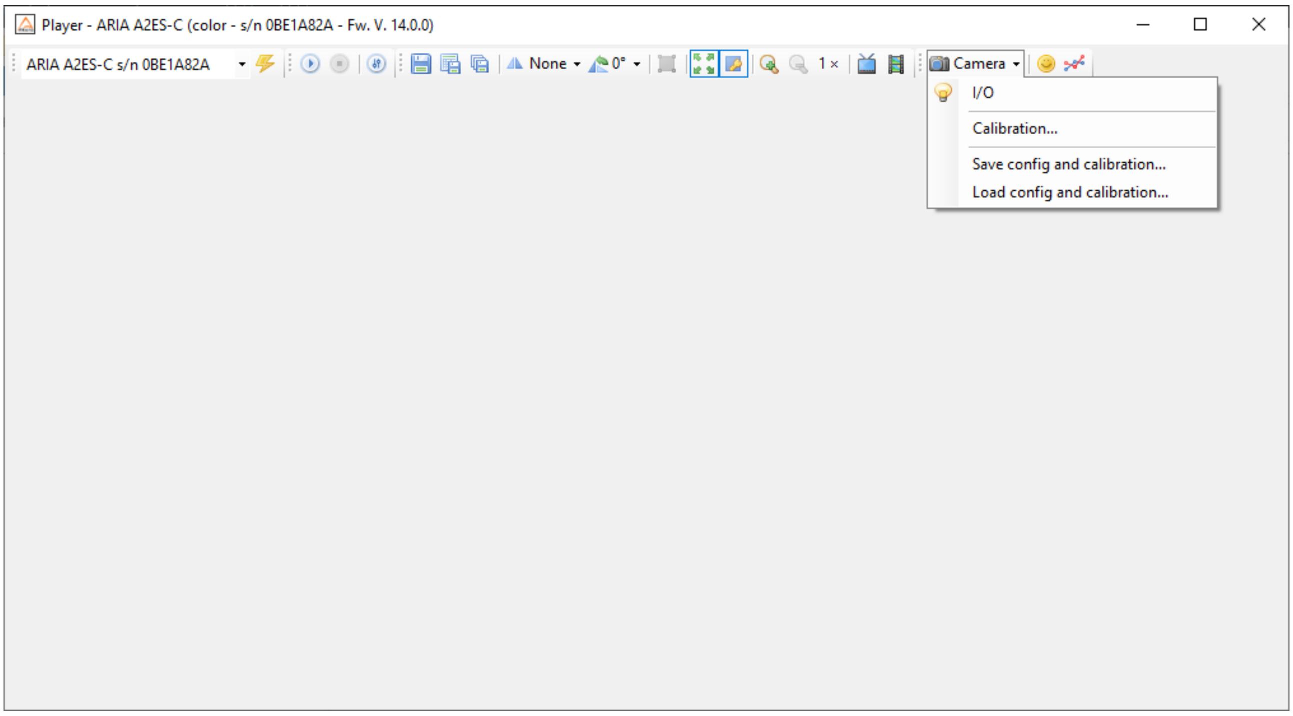
On the rightmost part of the toolbar you can find the Camera menu (Figure 9.8) containing camera-specific features.


Display Options

Max Displayed FPS: 60,00

☑ Display

Figure 9.7: *Display Frames panel*


Player - ARIA A2ES-C (color - s/n 0BE1A82A - Fw. V. 14.0.0)

ARIA A2ES-C s/n 0BE1A82A
None
0°
1 ×
Camera
I/O
Calibration...
Save config and calibration...
Load config and calibration...

Figure 9.8: *ARIA specific features menu*

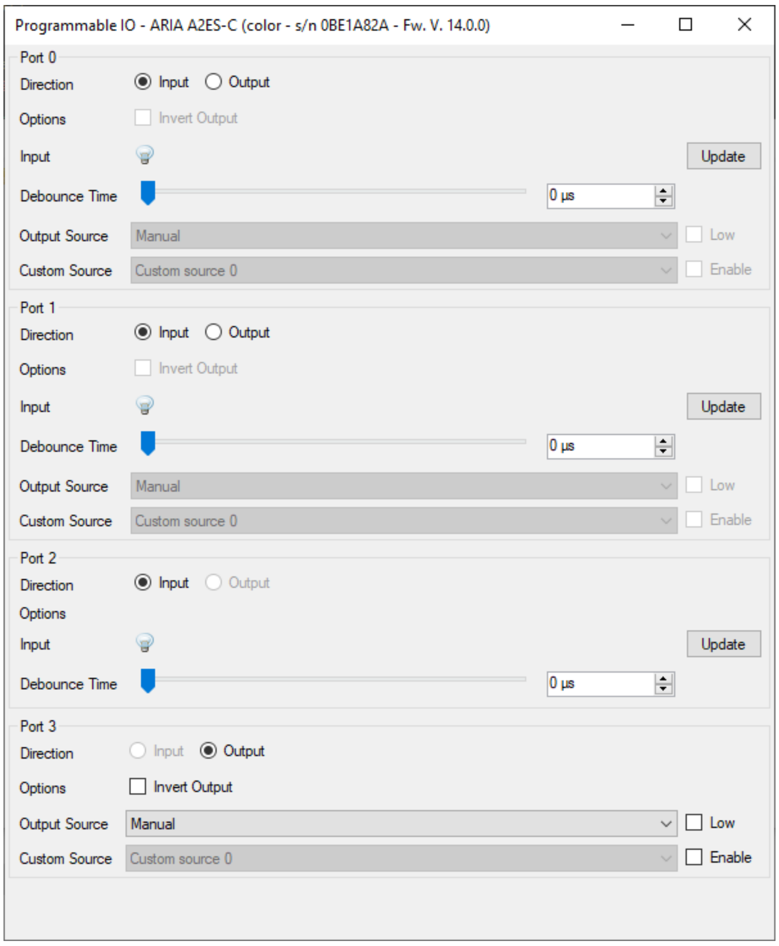
9.6.1 Saving and Exporting Configuration

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 8.8 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).

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 4.1.1), 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.


Programmable IO - ARIA A2ES-C (color - s/n 0BE1A82A - Fw. V. 14.0.0)

Port 0
Direction: [✓ Input] Output
Options: Invert Output
Input: Update
Debounce Time: 0 µs
Output Source: Manual — Low
Custom Source: Custom source 0 — Enable

Port 1
Direction: [✓ Input] Output
Options: Invert Output
Input: Update
Debounce Time: 0 µs
Output Source: Manual — Low
Custom Source: Custom source 0 — Enable

Port 2
Direction: [✓ Input] Output
Options
Input: Update
Debounce Time: 0 µs

Port 3
Direction: Input [✓ Output]
Options: Invert Output
Output Source: Manual — Low
Custom Source: Custom source 0 — Enable

Figure 9.9: *I/O control panel*

9.6.3 Black Level Calibration (e2V models only)

Starting from the first CCDs, image sensors have a certain number of side columns shielded from incident light, in order to measure the electric analog offset due to thermal and parasitic effects and subtract it analogically or digitally from the acquired image. This reference level is called *Black Level* and represents the output of the camera in total darkness, with no compensations applied. The ideal ADC output characterstics should be a straight line resulting in a null output when no light incides into the pixel, and fullscale value (e.g. 255 in 8-bit modes) at saturation. Due to the Black Level offset, the actual pixel conversion may be affected by a small amount that can be compensated using the Black Level Offset control.

Setting the right black level plays a very important role in obtaining a high-quality image from the camera. When the black level is improperly set, the image has poor dynamic range or, in other words, it looks flat. When the black level is set too low, darker portions of the image appear grey and no image detail appears "really black", as if they were "hidden behind a curtain". On the other side, when the black level is set too high, darker details collapse into black and disappear, while white details become grey. Figure 9.10 shows the effect of an improperly set black level.

The effect of a wrong black level can be seen on the images histogram too. When the black level is set too high, the image histogram appears compressed towards the right side (high lights), leaving the left

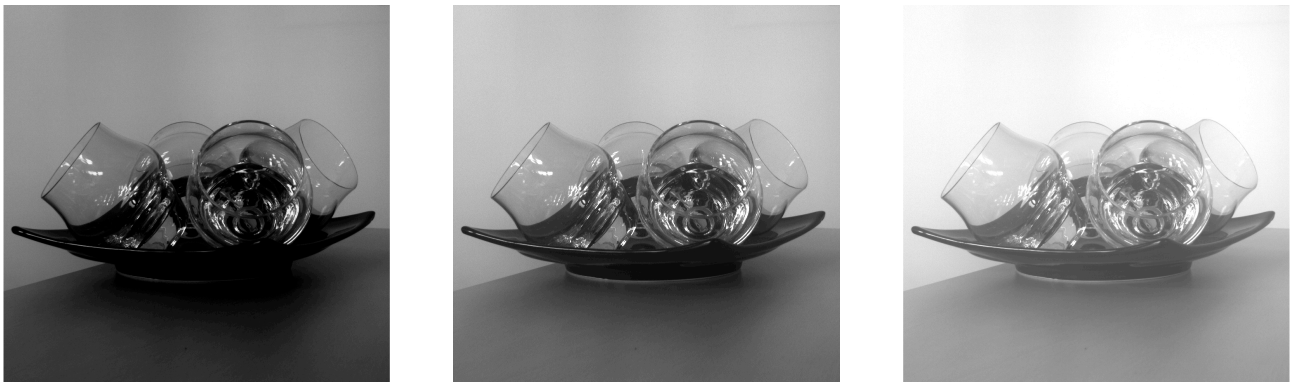


Figure 9.10: Black level calibration examples

side unpopulated. On the other side, setting black level too low causes the image histogram to translate leftwards, leaving the right side unpopulated. The histogram in the middle refers to an image whose black level is properly set. The histogram is well centered and all values from zero to saturation are represented.

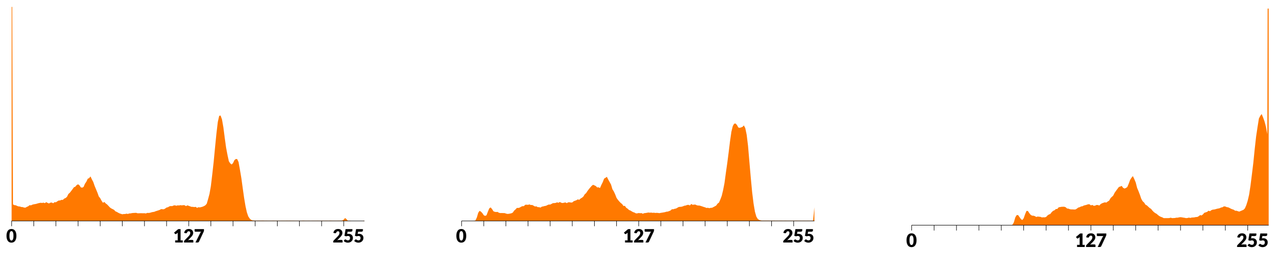
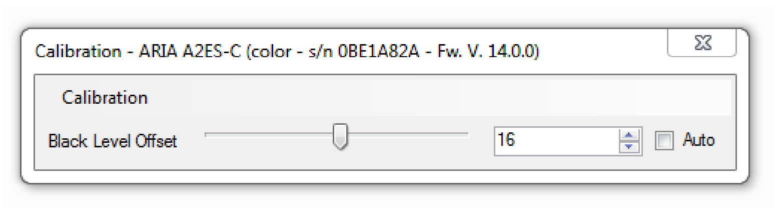


Figure 9.11: Black level histograms examples

Since ideal camera output should be zero for all active pixels not exposed to light, some modern CMOS sensors feature some automatic compensation mechanism in order to adjust black level without your intervention. Some other sensors, like the Sapphire and Ruby series from e2V , use an internal register to fine tune the black level. Since the optimum black level may be different from batch to batch, or even from sensor to sensor, ARIA cameras come already individually calibrated in order to have the best black level preset. Anyway, some small residue offet may arise when long exposures are needed or when the camera is operated at high temperature, so *Alkeria player* features a calibration form allowing manual and automatic black level calibration.


Calibration - ARIA A2ES-C (color - s/n 0BE1A82A - Fw. V. 14.0.0)

Calibration

Black Level Offset 16 Auto

Figure 9.12: Black level panel

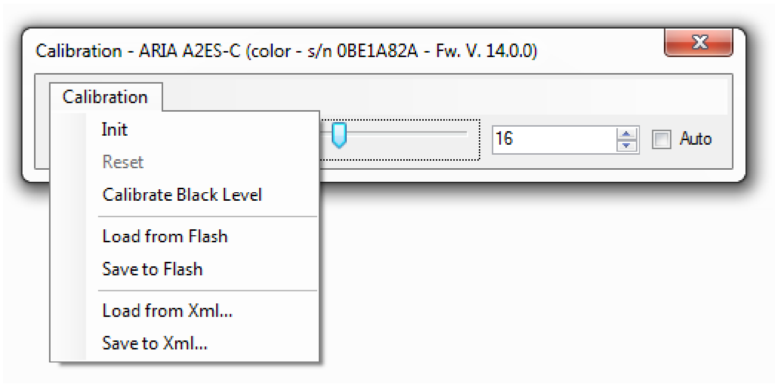


Figure 9.13: *ARIA Black level settings menu*

The calibration form features a manual control and an automatic calibration routine that finds a black level offset value that gives a maximum number of zero pixels not greater than 2% of image.

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 8.7.11). This simple GUI allows you to edit and apply your custom LUT.

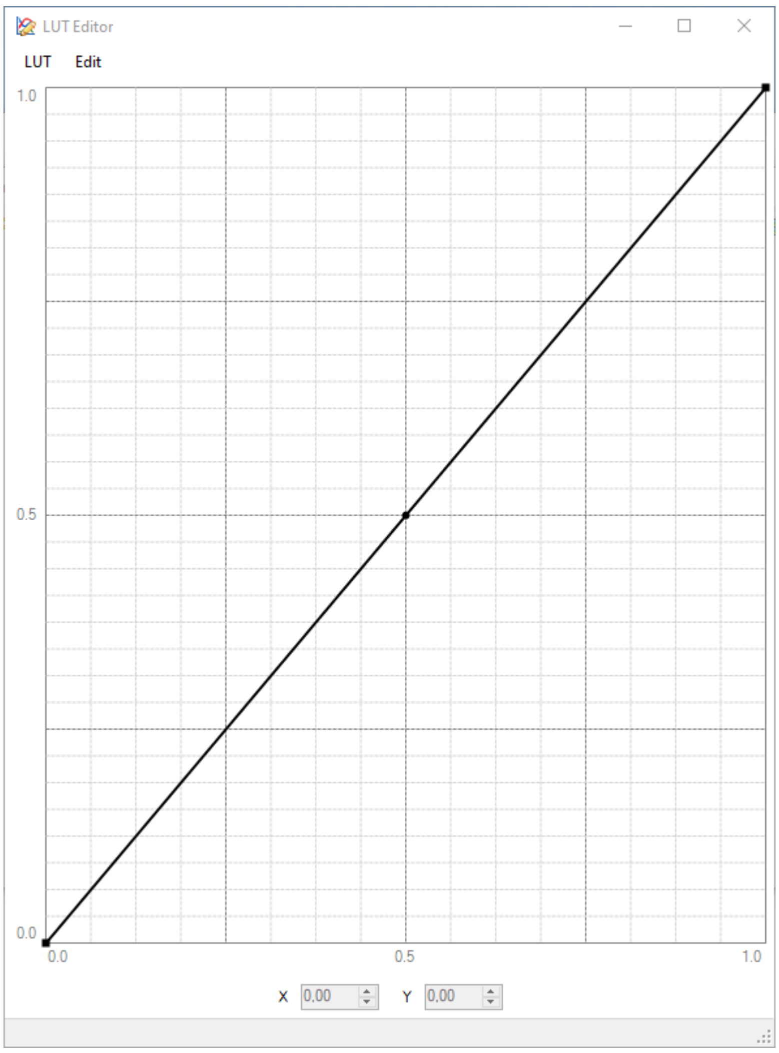


Figure 9.14: *LUT editor GUI*

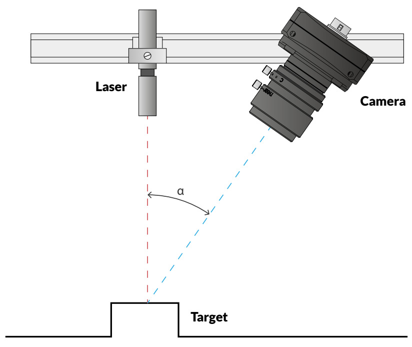
10 Profilometer extension for laser triangulation

10.1 LASER TRIANGULATION

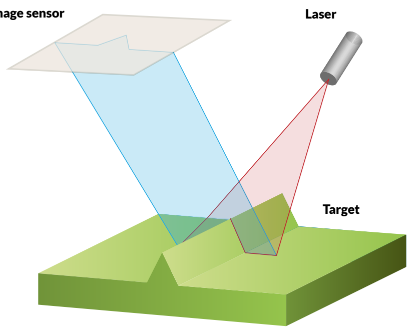
Laser triangulation is a non-contact distance measurement technique widely employed for surface inspection applications. A common setup involves a camera and a laser line, positioned as in Figure 10.1.

The laser line reflected by a target surface is captured by the camera. The first step for distance calculation is to precisely identify the reflected line position in the acquired frame. Resolution of the overall measure is highly dependent on the accuracy of this step.

The Profilometer extension for ARIA monochrome models enables on-camera acceleration of the line



(a)



(b)

Figure 10.1: *Typical laser triangulation setup.*

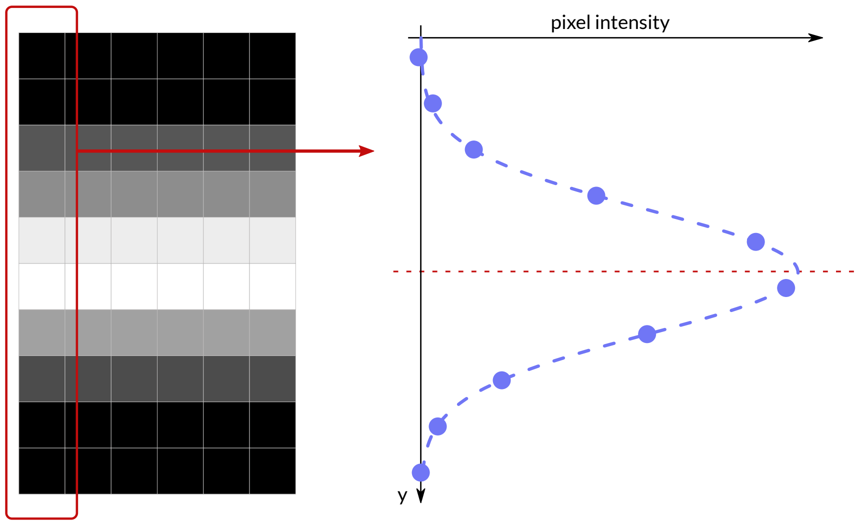


Figure 10.2: *Every frame column is processed independently. For each one, the algorithm provides a 16-bit y-coordinate of the laser line position to the user.*

position detection algorithm. This approach offers two advantages:

- The receiving PC has a lower computational load, since part of the calculations are provided by the camera;
- The acquired frame is no longer needed once line coordinates calculations have been performed. To maximize frame rate and hence speed-up acquisition time, user can exclude full frame from the data sent over USB, and limit the data transmission to coordinates only (see Section 10.3.3).

10.2 LASER LINE POSITION DETECTION

To detect the exact position of the laser line in the image acquired by the sensor, every image column is processed independently, as shown in Figure 10.2. Each pixel is compared to an upper and lower threshold and eventually clamped before main column processing.

10.2.1 Thresholds

Thresholds are available to exclude part of the data from the main processing algorithm (see Figure 10.3):

- Lower threshold clamps to zero every pixel with intensity lower than its value. By setting this value appropriately, background noise can be excluded from calculations.
- Higher threshold clamps to its value every pixel with higher intensity. By setting this value appropriately, spikes effect on calculations may be mitigated.

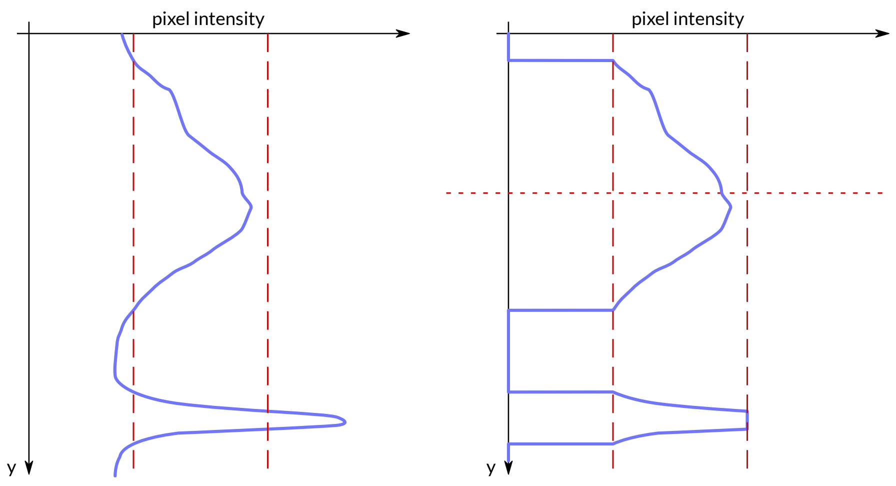


Figure 10.3: *Threshold effect on column data. When multiple clusters are present, the algorithm automatically chooses the bigger one.*

10.2.2 Main Processing

Every column data set is then divided in separated clusters, one for every contiguous group of pixels with non-zero intensity. The algorithm selects the biggest cluster and calculates its center position.

This value is sent to the user as one 16-bit pixel. The pixel should be interpreted as a fixed-point number with a radix point position determined by te user through the `DecimalPrecision` setting. To retrieve the position from the pixel value at column C, the intensity $P^{(C)}$ should be divided by an appropriate factor:

$$Y_{line}^{(C)} = \frac{P^{(C)}}{2^d}$$

Where d is the selected decimal precision.

10.3 PROFILE ACQUISITION

As explained in Section 10.2.2, the line position is expressed as a set of 16-bit unsigned values. The acquisition method depends on the chosen video mode and color coding.

10.3.1 Video Mode 0

In Video Mode 0 the image is processed by the image processing block (see Figure 7.2) and then by the Profilometer hardware accelerator. As in standard monochrome cameras, the whole frame is provided to the user. The profile coordinates are available through the `FrameRawStream` chunk data field (see Section 7.3 and Example 10.2).

10.3.2 Video Mode 1

In Video Mode 1 the raw image is directly sent to the Profilometer hardware accelerator, bypassing the image processing block. As in Video Mode 0, the whole frame is provided to the user and profile coordinates are available through the `FrameRawStream` chunk data field.

10.3.3 Video Mode 2

In Video Mode 2 only the profile data are provided to the user as the main video stream. The color coding (MONO16 or RAW16) determines whether the data is processed or not by the image processing block.

The acquired frame is a single line, representing on 16-bit pixels the line position[1] at each column. This video mode minimizes the amount of data sent through the USB, relaxing packet size requirements for the acquisition. In this way user can easily reach maximum sensor frame rate and collect only the data useful for the measurement.

10.4 EXPLOITING FRAME COMBINER WITH PROFILOMETER

The frame combiner is particularly useful when used in combination with Video Mode 2. When high frame-rate acquisition in Video Mode 2 is taking place, the acquiring PC is forced to access very frequently small data arrays (i.e. the one-line frame providing the profile), reducing the processing time available on those data. Using the frame combiner, an arbitrary number of lines may be packed in a single frame to reduce the overhead. A 1000 fps video stream, by example, could be packed in a 1000 rows frame an then provided to the user once per second.

Frame combiner functionality is embedded in the FPGA processing so that the acquiring PC has no additional tasks to execute. Additionally, the FPGA processing guarantees that no line is skipped between subsequent packed frames.

10.5 PLAYER UTILITIES

ARIA models with profilometer extension use a dedicated camera player to exploit all their additional functionalities. This player can be found on our website, on the User Area inside the *Linux & Windows SDK* folder. This special version of the ARIA player features two additional controls dedicated to the Profilometer extension (see Figure 10.4).

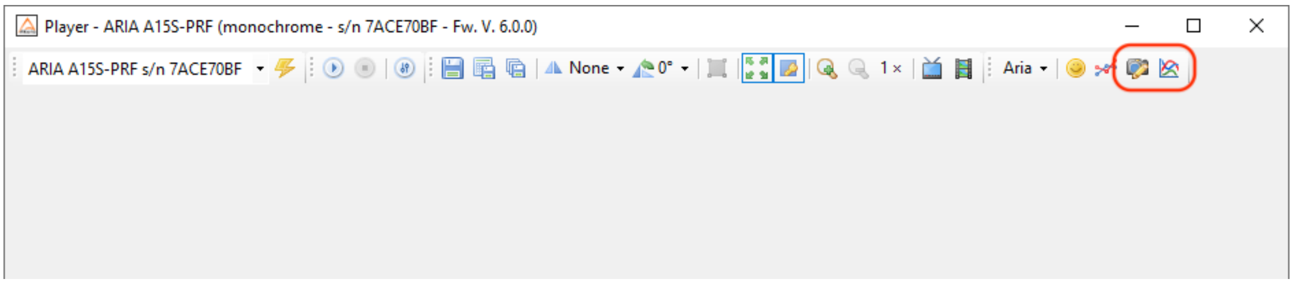

Player - ARIA A15S-PRF (monochrome - s/n 7ACE70BF - Fw. V. 6.0.0)
ARIA A15S-PRF s/n 7ACE70BF
None
0°
1 x
Aria

Figure 10.4: *ARIA player for profilometer extension*

[1]The position is relative to ROI Top

10.5.1 Live Profile

The live profile form provides a chart of the line position in the acquired frame (see Figure 10.5).

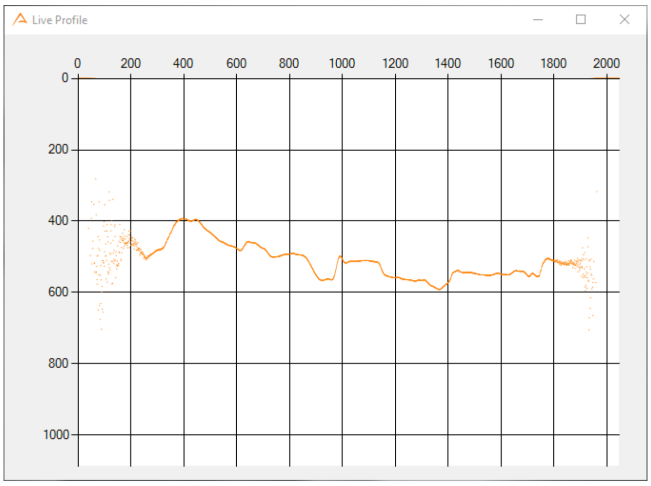
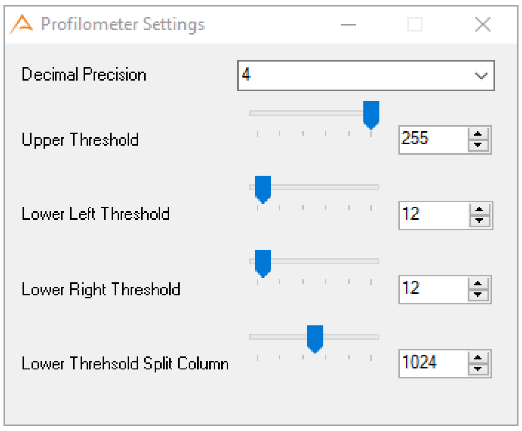


Figure 10.5: *Live Profile player form*

10.5.2 Profilometer Settings

The Profilometer settings form gives access to the main extension features (see Figure 10.6). A second lower threshold and a threshold transition columns are provided for acquiring images where the background noise is not uniform across the scene. All columns from `0` to `ThresholdTransitionColumn − 1` are processed with `LeftLowerThreshold` as the lower threshold while the others are processed with `RightLowerThreshold`. To operate with a single lower column, set both `LeftLowerThreshold` and `RightLowerThreshold` to the same value.


Profilometer Settings

Decimal Precision	4
Upper Threshold	255
Lower Left Threshold	12
Lower Right Threshold	12
Lower Threhsold Split Column	1024

Figure 10.6: *Profilometer Settings form*

10.6 PARAMETERS SET-UP

Profilometer settings are accessible from the `Profilometer` member of the `AriaProfCamera` class.

Example Code 10.1 | *Setup of profilometer parameters*

```
// Set up parameters: upper threshold to maximum value (no effect) and lower
// thresholds to 10 for columns between 0 and 511 and 20 for other columns.
device.Profilometer.UpperThreshold = device.Profilometer.MaxThreshold
device.Profilometer.LeftLowerThreshold = 12
device.Profilometer.RightLowerThreshold = 12
device.Profilometer.LowerThresholdSplitColumn = 1024
device.Profilometer.Precision = 4
```

When acquiring in Video Mode 0 or 1, the profile is accessible by means of chunk data, as shown in the Example Code 10.2.

Example Code 10.2 | *Chunk Data acquisition in Video Mode 1*

```
byte[] chunk;
ushort chunkLength;

device.VideoMode = 1
device.EnableChunkData = true;
device.SetEnabledChunkDataFields(new ChunkDataField[] {ChunkDataField.FrameRawStream});
device.Acquire = true;

BufferPtr ptr = device.GetRawDataPtr(false);    // Get image and chunk
chunk = ptr.RawChunkData;                       // Chunk Data
chunkLength = (ushort)chunk.Length;             // Number of chunk data bytes

byte precision = _Device.Profilometer.Precision;
for (int i = 0; i < chunkLength - 1; i += 2)
{
    ushort intensity = (ushort)chunk[i] | ((ushort)chunk[i + 1] << 8);
    float position = intensity * Math.Pow(2, -precision));
    ...
}
```

In Video Mode 2 enabling Chunk Data is not needed. Profile may be acquired with a simple `GetRaw-DataPtr` instruction, once the live has been started.

Example Code 10.3 | *Setup for profilometer acquisition with frame combiner*

```
device.VideoMode = 2
device.EnableChunkData = false;      // Profile in Video Mode 2 is the main image
device.FrameCombinerSize = 1000;     // Pack 1000 profiles in a single image
device.Acquire = true;

BufferPtr ptr = device.GetRawDataPtr(false);    // Get profile from image
```

11 Warranty

Unless otherwise agreed, ARIA 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.

11.1 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.

12 Firmware Update

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

When applying for MaestroUSB3 free delivery, ARIA 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.

13 EMC certification and compliance

13.1 CE CONFORMITY

ARIA 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.

13.2 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.

13.3 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."

13.4 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.

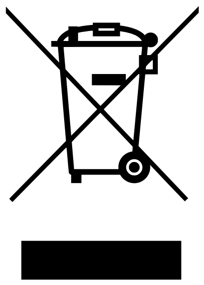
13.5 ROHS CONFORMITY

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

RoHS Compliant

13.6 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

ADC	Analog to Digital Converter
PLL	Phase Locked Loop
LMR	Light Meter ROI
ROI	Region Of Interest
LUT	Look-Up Table
ROI	Region-Of-Interest
API	Application Programming Interface

Example Code

3.1 Temperature Monitoring 46
3.2 Temperature Change Event 47
4.1 Input Change Event 54
4.2 Encoder ports configuration 61
4.3 Encoder position read 62
4.4 Encoder reset with external source 62
4.5 PLL configuration 63
4.6 PLL boundaries readout 64
4.7 Output inversion 65
4.8 Output controlled via software 66
4.9 Exposure as P3 source 66
4.10 Strobe as P3 source 67
4.11 *Frame start trigger* as P2 source 67
4.12 *Frame start trigger ready* as P2 source 68
4.13 Custom signal 10 as P0 source 68
4.14 P0 configuration as input 68
4.15 P0 configuration as output 68
4.16 Serial interface output configuration 69
4.17 Big buffer send over serial interface 69
4.18 Serial interface input configuration 70
6.1 Get all selectable trigger sources 75
6.2 Usage of the `TriggerErrors` property 77
6.3 External trigger frame acquisition 78
6.4 Signal-driven exposure time 78
6.5 Encoder synchronization 79
7.1 Set LMR starting from x = 256, y = 384, and 512 x 800 pixels wide 80
7.2 Set Black Level offset correction 81
7.3 Enable frame number chunk data field 83
7.4 Enable time stamp chunk data field 83
7.5 Enable encoder position chunk data 83
7.6 Enable input status chunk data captured at frame-start trigger event 83
7.7 Enable burst number chunk data field 84
7.8 Configure chunk data, capture a frame and display the relevant data 84

7.9 Configure chunk data, combine 10 frames, set 1 second timeout, flush operation triggered 84
8.1 Select camera video mode 85
8.2 ROI configuring example 86
8.3 Set combined binning mode 90
8.4 Disable binning mode 90
8.5 Set pixel format to RGB24 92
8.6 Set `ADCResolution` property to 10 bits 92
8.7 Sets a bandwidth limits of 32 KiB per microframe on ARIA 94
8.8 Set `PacketSize` property 95
8.9 Set `FrameRate` property 98
8.10 Enable the `PreserveRates` functionality 99
8.11 Set the brightness level 100
8.12 Set the contrast level 101
8.13 Set the color correction matrix coefficients 102
8.14 Set the hue level 102
8.15 Set the saturation level 103
8.16 Set white balance automatically 104
8.17 Set white balance manually 104
8.18 Set gamma 105
8.19 Set Exposure time with unit factor 106
8.20 Get the exposure length, expressed in seconds 107
8.21 Set up automatic exposure 108
8.22 Set up automatic exposure 108
8.23 Set digital gain level 109
8.24 Store lut into the camera 110
8.25 Read the Luma value of the last image 111
8.26 Read the TimeStamp value 111
8.27 Flip the image diagonally 111
8.28 Rotate the image of 270 deg 112
8.29 Retreive the maximum number of user configurations 112
8.30 Save and restore camera configuration and calibration 112
8.31 Erase configuration 112
8.32 Export camera configuration and calibration previously saved on camera memory 113
8.33 Load from XML and save into camera memory 113
8.34 Save and restore camera configuration from camera memory 113
8.35 Erase user configuration from camera memory 113
8.36 Set camera's main parameters and enable live acquisition 113
8.37 Enable pattern generator 114
10.1 Setup of profilometer parameters 132
10.2 Chunk Data acquisition in Video Mode 1 132
10.3 Setup for profilometer acquisition with frame combiner 132

Alkeria

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