

Manual for uEye Cameras

Version 3.50.00

Status: October 2009



Dimbacher Str. 6-8
D-74182 Obersulm
Fax: +49-(0)7134-96196-99
E-Mail: sales@ids-imaging.com

Content

1	Preface	10
2	Contacting Us	12
3	Welcome	13
3.1	About this Manual	14
3.2	What is New in this Version?	16
3.3	uEye Camera Family	17
4	A: Operation	20
4.1	Quick-Start	20
4.2	Installation	23
	System Requirements	
	Downloading the Software	
	Installing uEye Software (Windows)	
	Installing uEye Software (Linux)	
	Connecting a USB uEye Camera	
	Connecting a GigE uEye Camera	
4.3	Applications	35
	uEye Camera Manager	
	uEye Demo	
	uEye Player	
4.4	uEye Camera Basics	102
	Operating Modes	
	Image Display Modes	
	Sensor	
	Reading out Partial Images	
	Camera Parameters	
	Firmware and Camera Start-up	
	Pixel Preprocessing in GigE uEye Cameras	
	Digital Inputs/Outputs	
	USB Basics	
	GigE Basics	
5	B: Programming	134

5.1	Quick-Start	134
5.2	How To Proceed	136
	Preparing Image Capture	
	Display Mode Selection	
	Image Capture	
	Saving Images and Videos	
	Setting Camera Parameters	
	Using Inputs and Outputs	
5.3	Function Descriptions	166
	is_AddToSequence	
	is_AllocImageMem	
	is_CameraStatus	
	is_CaptureVideo	
	is_ClearSequence	
	is_ConvertImage	
	is_CopyImageMem	
	is_CopyImageMemLines	
	is_DirectRenderer	
	is_DisableEvent	
	is_EnableAutoExit	
	is_EnableEvent	
	is_EnableHdr	
	is_EnableMessage	
	is_ExitCamera	
	is_ExitEvent	
	is_ForceTrigger	
	is_FreelImageMem	
	is_FreezeVideo	
	is_GetActiveImageMem	
	is_GetActSeqBuf	
	is_GetAutoInfo	
	is_GetBusSpeed	
	is_GetCameraInfo	
	is_GetCameraList	
	is_GetCameraLUT	
	is_GetCameraType	
	is_GetCaptureErrorInfo	
	is_GetColorConverter	

is_GetColorDepth
is_GetComportNumber
is_GetDLLVersion
is_GetDuration
is_GetEthDeviceInfo
is_GetError
is_GetExposureRange
is_GetFramesPerSecond
is_GetFrameTimeRange
is_GetGlobalFlashDelays
is_GetHdrKneepointInfo
is_GetHdrKneepoints
is_GetHdrMode
is_GetImageHistogram
is_GetImageInfo
is_GetImageMem
is_GetImageMemPitch
is_GetNumberOfCameras
is_GetOsVersion
is_GetPixelClockRange
is_GetSensorInfo
is_GetSensorScalerInfo
is_GetSupportedTestImages
is_GetTestImageValueRange
is_GetTimeout
is_GetUsedBandwidth
is_GetVsyncCount
is_HasVideoStarted
is_InitCamera
is_InitEvent
is_InquireImageMem
is_IsVideoFinish
is_LoadBadPixelCorrectionTable
is_LoadImage
is_LoadImageMem
is_LoadParameters
is_LockSeqBuf
is_ReadEEPROM
is_ReadI2C

is_RenderBitmap
is_ResetCaptureErrorInfo
is_ResetToDefault
is_SaveBadPixelCorrectionTable
is_SaveImage
is_SaveImageEx
is_SaveImageMem
is_SaveImageMemEx
is_SaveParameters
is_SetAllocatedImageMem
is_SetAOI
is_SetAutoCfgIpSetup
is_SetAutoParameter
is_SetBadPixelCorrection
is_SetBadPixelCorrectionTable
is_SetBinning
is_SetBICompensation
is_SetCameraID
is_SetCameraLUT
is_SetColorConverter
is_SetColorCorrection
is_SetColorMode
is_SetConvertParam
is_SetDisplayMode
is_SetDisplayPos
is_SetEdgeEnhancement
is_SetErrorReport
is_SetExposureTime
is_SetExternalTrigger
is_SetFlashDelay
is_SetFlashStrobe
is_SetFrameRate
is_SetGainBoost
is_SetGamma
is_SetGlobalShutter
is_SetHardwareGain
is_SetHardwareGamma
is_SetHdrKneepoints
is_SetHWGainFactor

- is_SetImageMem
- is_SetImagePos
- is_SetIO
- is_SetIOMask
- is_SetLED
- is_SetOptimalCameraTiming
- is_SetPacketFilter
- is_SetPersistentIpCfg
- is_SetPixelClock
- is_SetRopEffect
- is_SetSaturation
- is_SetSensorScaler
- is_SetSensorTestImage
- is_SetStarterFirmware
- is_SetSubSampling
- is_SetTimeout
- is_SetTriggerCounter
- is_SetTriggerDelay
- is_StopLiveVideo
- is_UnlockSeqBuf
- is_WaitEvent
- is_WriteEEPROM
- is_WriteI2C

5.4 AVI Function Descriptions

365

- isavi_AddFrame
- isavi_CloseAVI
- isavi_DisableEvent
- isavi_EnableEvent
- isavi_ExitAVI
- isavi_ExitEvent
- isavi_GetAVIFileName
- isavi_GetAVISize
- isavi_GetnCompressedFrames
- isavi_GetnLostFrames
- isavi_InitAVI
- isavi_InitEvent
- isavi_OpenAVI
- isavi_ResetFrameCounters
- isavi_SetFrameRate

	isavi_SetImageQuality	
	isavi_SetImageSize	
	isavi_StartAVI	
	isavi_StopAVI	
5.5	Obsolete Functions	386
	is_DisableDDOverlay	
	is_EnableDDOverlay	
	is_GetDC	
	is_GetDDOvlSurface	
	is_HideDDOverlay	
	is_LockDDMem	
	is_LockDDOverlayMem	
	is_PrepareStealVideo	
	is_ReleaseDC	
	is_SetBayerConversion	
	is_SetDDUpdateTime	
	is_SetHwnd	
	is_SetImageAOI	
	is_SetImageSize	
	is_SetKeyColor	
	is_ShowDDOverlay	
	is_StealVideo	
	is_UnlockDDMem	
	is_UnlockDDOverlayMem	
	is_UpdateDisplay	
5.6	Lists and Programming Notes	410
	Programming	
	Complete List of All Return Values	
	Linux Functions	
	Compatibility with FALCON Functions	
6	C: Specifications	421
6.1	Model Comparison	422
6.2	Sensor Data	423
	UI-122x / UI-522x	
	UI-146x / UI-546x	
	UI-148x / UI-548x	
	UI-149x / UI-549x	

	UI-154x / UI-554x	
	UI-155x / UI-555x	
	UI-164x / UI-564x	
	UI-221x / UI-621x	
	UI-222x / UI-622x	
	UI-223x / UI-623x	
	UI-224x / UI-624x	
	UI-225x / UI-625x	
	UI-241x / UI-641x	
6.3	Mechanical Specifications	459
	USB uEye SE	
	USB uEye ME	
	USB uEye RE	
	USB uEye LE	
	GigE uEye SE	
	GigE uEye RE	
	GigE uEye HE	
	Flange Back Distance	
	Filter Glasses	
	Ambient Conditions	
6.4	Electrical Specifications	505
	Digital Inputs/Outputs	
	EEPROM	
	USB uEye SE	
	USB uEye ME	
	USB uEye RE	
	USB uEye LE	
	GigE uEye SE	
	GigE uEye RE	
	GigE uEye HE	
6.5	Accessories	538
	USB uEye SE	
	USB uEye ME	
	USB uEye RE	
	USB uEye LE	
	GigE uEye SE	
	GigE uEye RE	
	GigE uEye HE	

6.6	Components of uEye Cameras	556
7	Appendix	557
7.1	Troubleshooting	557
7.2	Colour and Memory Formats	559
7.3	uEye Parameter File (INI File)	560
7.4	History of uEye Software Versions	565
7.5	History of API Functions	568

1 Preface

Introduction

IDS Imaging Development Systems GmbH has taken every possible care in preparing this manual. We however assume no liability for the content, completeness or quality of the information contained therein. The content of this manual is regularly updated and adapted to reflect the current status of the software. We furthermore do not guarantee that this product will function without errors, even if the stated specifications are adhered to.

Under no circumstances can we guarantee that a particular objective can be achieved with the purchase of this product.

Insofar as permitted under statutory regulations, we assume no liability for direct damage, indirect damage or damages suffered by third parties resulting from the purchase of this product. In no event shall any liability exceed the purchase price of the product.

Please note that the content of this manual is neither part of any previous or existing agreement, promise, representation or legal relationship, nor an alteration or amendment thereof. All obligations of *IDS Imaging Development Systems GmbH* result from the respective contract of sale, which also includes the complete and exclusively applicable warranty regulations. These contractual warranty regulations are neither extended nor limited by the information contained in this manual. Should you require further information on this product, or encounter specific problems that are not discussed in sufficient detail in the manual, please contact your local *uEye* dealer or system installer.

All rights reserved. This manual may not be reproduced, transmitted or translated to another language, either as a whole or in parts, without the prior written permission of *IDS Imaging Development Systems GmbH*.

Status: October 2009

Safety Information

The product must be connected, taken into operation and maintained only by appropriately qualified personnel.

The error-free and safe operation of this product can only be ensured if it is properly transported, stored, set up and assembled, and operated and maintained with due care.

Operating Environment

Please comply with the requirements for the proper use of this product. Failure to do so will render the warranty void.

Do not subject this product to direct sunlight, moisture or shock. The environmental conditions specified in chapter Specifications are required.

EMC Directives

IDS Imaging Development Systems GmbH hereby confirms that this product has been developed, designed and manufactured in compliance with the EC Directive 89/336/EEC (Electromagnetic Compatibility).

Compliance with the directives is demonstrated by meeting the following standards:

Product type	EMC immunity	EMC emission
USB uEye SE (CMOS sensors) *)	EN 61000-6-2:2005	EN 61000-6-3:2001 + A11:2004
USB uEye SE (CCD sensors) *)	EN 61000-6-2:2001	EN 61000-6-4:2001
USB uEye ME (CMOS sensors) *)	EN 61000-6-2:2005	EN 61000-6-3:2001 + A11:2004
USB uEye ME (CCD sensors) *)	EN 61000-6-2:2005	EN 61000-6-3:2001 + A11:2004
USB uEye RE (CMOS sensors) *)	EN 61000-6-2:2001	EN 61000-6-3:2001 + A11:2004
USB uEye RE (CCD sensors) *)	EN 61000-6-2:2001	EN 61000-6-3:2001 + A11:2004
USB uEye LE (CMOS sensors) *)	EN 61000-6-2:2005	EN 61000-6-3:2001 + A11:2004
GigE uEye SE (CMOS sensors) *)	EN 61000-6-2:2005	EN 61000-6-4:2001
GigE uEye SE (CCD sensors) *)	EN 61000-6-2:2005	EN 61000-6-4:2001
GigE uEye RE (CMOS sensors)	EN 61000-6-2:2005	EN 61000-6-4:2007
GigE uEye RE (CCD sensors)	EN 61000-6-2:2005	EN 61000-6-4:2007
GigE uEye HE (CMOS sensors) *)	EN 61000-6-2:2005	EN 61000-6-3:2001 + A11:2004
GigE uEye HE (CCD sensors) *)	EN 61000-6-2:2005	EN 61000-6-3:2001 + A11:2004

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

Modifications not expressly approved by the manufacturer could void the user's authority to operate the equipment under FCC rules.

Trademarks

IDS Imaging Development Systems and *uEye* are registered trademarks of *IDS Imaging Development Systems GmbH*. Microsoft and Windows are trademarks or registered trademarks of Microsoft Corporation. All other products or company names mentioned in this manual are used solely for purposes of identification or description and may be trademarks or registered trademarks of the respective owners.

Copyright

© *IDS Imaging Development Systems GmbH*. All rights reserved.

IDS Imaging Development Systems GmbH hereby grants the purchaser the right to use the software.

2 Contacting Us

Visit our web site <http://www.ids-imaging.com> where you will find all the latest drivers and information about our software and hardware products. The latest *uEye* driver is available on our website <http://www.ueyesetup.com>.

Please contact your local IDS distributors for first level support in your language. For a list of IDS distributors worldwide please go to our website and follow the *Support* link.

Address:	IDS Imaging Development Systems GmbH Dimbacher Str. 6-8 D-74182 Obersulm
Fax:	+49-(0)7134-96196-99
Email:	Sales: sales@ids-imaging.com Support: support@ids-imaging.com
Internet	http://www.ids-imaging.com

3 Welcome

Thank you for purchasing a *uEye* camera from *IDS Imaging Development Systems GmbH*.

The uEye Software Package

For your *uEye* camera, a comprehensive software package is available for download. In addition to the drivers, the *uEye Software Development Kit (SDK)* includes the *uEye Camera Manager*, the *uEye Demo* and the *uEye API* programming interface for creating your own *uEye* programs under Windows 2000, XP and Vista (32-bit) as well as Linux. Numerous demo applications make it easy for you to get started with *uEye* programming.

The current version of the *uEye SDK* is available as a free download at <http://www.ueyesetup.com>. Information on how to install the software and connect the camera can be found in this manual.

Enjoy your new *uEye* camera!

Getting Started

You should first read the following chapters to get a quick overview on what is new in this software version and on getting started with *uEye* programming.



About this Manual

What is New?

Quick-Start

Quick-Start: Programming

Enjoy your new *uEye* camera!

3.1 About this Manual

Contents

The *uEye Manual* contains all the information you need for operating your *uEye* camera. The *uEye Manual* comprises the following parts:

- **Section A: Operating the Camera**

- Quick Start to using your *uEye*
- Installing and Using uEye Software
These sections show how to connect cameras and start operation using the software tools *uEye Camera Manager* and *uEye Demo*.
- Camera Basics
In this section you will find a lot of important information on the technical background of your USB or GigE camera. This section contains explanations on the *uEye's* Operating Modes, on Sensor Technology, important Camera Parameters, and the USB and GigE interfaces. We recommend to read this chapter to become familiar with the general functionality of the *uEye* cameras.

- **Section B: Programming with the *uEye* API**

- Quick-Start to programming with your *uEye*
- How To Proceed
If you are not yet familiar with *uEye* programming, we suggest that you first explore the basic functional flows in this chapter. The function blocks contain almost all the functions available for the *uEye* API ordered by topics. The flowcharts help to easily find the appropriate API function for a certain task.
- Description of Functions / Description of AVI Functions
These chapters cover all the functions of the *uEye* API in alphabetic order.
The AVI functions for video recording are implemented by the *ueye_tools.dll* which is also included in the *uEye Software Development Kit*.
- Obsolete Functions
This chapter lists obsolete API functions and recommended alternatives.
- Lists and Programming Notes
In this chapter, you will find useful information on how to use the *uEye* programming API. Programming environments, modes for *uEye* colour and image display as well as the automatic image control functions are discussed here.

- **Section C: Specifications and Accessories**

- Specifications
All information on the camera's Sensor and Performance, Mechanical as well as Electrical Specifications are contained in this section.
- Accessories
Here you will find a list of accessories for *uEye* cameras sorted by model.

Symbols

Links to other chapters are underlined in the text.



In these boxes, you will find helpful user information.



In these boxes, you will find important notes and warnings.



☐ At a glance

These boxes show the contents of very long chapters at a glance and provide links to the sub-sections.



This symbol indicates interactive graphics. When you click on an active area in a graphic, a chapter containing additional information on that area opens automatically

3.2 What is New in this Version?

Version 3.50 of the *uEye* software package includes many new features and enhancements. The following table gives you an overview of the major new functions.

Please make sure to also read the file named *WhatsNew.txt* which you will find in the C:\Program Files\IDS\uEye\Help directory when the installation is completed. This file contains late-breaking information on new functions and fixed issues.

New in Version 3.50

Functions	Described in chapter
Driver support for Windows 7	System Requirements
Support for 64 bit versions of Windows 7, Windows Vista and Linux	System Requirements

Information in this Manual	Described in chapter
Merge of the previously separated manuals <i>uEye Programming Manual</i> and <i>uEye User Manual</i> .	-
New section: How to Proceed - <i>uEye</i> Programming	How To Proceed
New chapter: Troubleshooting	Troubleshooting
Keyword index added	-

Older versions

See the [History](#) chapter.

3.3 Introduction of the uEye Camera Family

uEye stands for a range of compact and cost-effective cameras for professional use in industrial, security and non-industrial applications. Equipped with the widely used USB 2.0 and Gigabit Ethernet ports, they can easily be interfaced with a vast variety of systems. The images are digitized in the camera and transmitted digitally to the PC. An additional frame grabber is not required.

uEye cameras have state-of-the-art CMOS and CCD sensors. The CMOS models use either the global or the rolling shutter method; the CCD models use only the global shutter method. *uEye* camera resolutions range from 640 x 480 pixels (VGA) to 2560 x 1920 pixels (QXGA), depending on the sensor. Further sensor modules will continuously expand the product portfolio. Depending on the individual model, the *uEye* cameras are available either as monochrome and color versions, or as color versions only.

The [Models Comparison](#) chapter shows the most important features of every series at a glance.

USB uEye SE “Standard Edition”

The *USB uEye SE* series features a robust metal housing with a standard mini-B USB 2.0 connector. Connection is additionally possible via a lockable micro D-sub connector which also carries the opto-isolated I/O signals. A *USB uEye SE* version with C-mount front flange has been developed for OEMs. The camera can also be supplied as PCB stack for special applications.

The USB 2.0 interface is meanwhile available in every standard PC and notebook/laptop and provides a gross bandwidth of 480 Mbps. The camera is connected and powered through the USB port by just a single cable.



Figure 1: USB uEye SE

USB uEye ME “Machine Edition”



Figure 2: USB uEye ME

The *USB uEye ME* features a right-angle rugged metal housing that makes it very compact. With its right-angle and low-profile design, this camera fits into the tightest spaces and is thus ideal for machine integration. Six mounting holes in the housing allow front-mounting the camera onto otherwise hard to reach places when using typical mounting methods.

The USB port of the *USB uEye ME* supports both standard mini-B connectors and lockable connectors. The camera's digital I/Os are interfaced using a lockable Hirose connector designed for use in demanding industrial environments.

USB uEye RE “Rough Environment Edition”

The *RE* versions of the *uEye* cameras are extremely rugged and thus offer an extended area of application. In conjunction with the optional lens tubes, these models meet the requirements of protection classes IP 65 and IP 67. The USB 2.0 and the I/O signals are connected via two ports of the same protection class. The *USB uEye RE* is therefore particularly suited for harsh environments.



Figure 3: USB uEye RE

USB uEye LE “Light Edition”



Figure 4: USB uEye LE versions

The *USB uEye LE* series features extremely compact cameras with high-speed CMOS sensors. The *LE* models are designed for professional use in non-industrial applications. Through the use of the widespread USB 2.0 technology, the cameras can easily be interfaced with a vast variety of systems. *USB uEye LE* cameras are available with a plastic housing with CS-mount lens adapter, as a board-level version with M12 or M14 lens holder, or without a lens holder.

GigE uEye SE “Standard Edition”

The *GigE uEye SE* is a highly compact Gigabit Ethernet camera. With a housing barely larger than that of the *USB uEye* models, the *GigE uEye SE* offers all the benefits of Gigabit Ethernet technology: High bandwidth, cable lengths up to 100 m, and widespread use of this interface.

Besides the lockable Gigabit Ethernet port, the camera provides a 6-pin Hirose connector that carries the power supply as well as the trigger and flash signals.



Figure 5: GigE uEye SE

GigE uEye RE “Rough Environment Edition”



Figure 6: GigE uEye RE

The *RE* version of the *GigE uEye* cameras features the same rugged design and extended application options as the equivalent USB based model. In conjunction with the optional lens tubes, the *RE* models meet the requirements of protection classes IP 65 and IP 67. The GigE interface and the I/O signals are connected via two lockable connectors that are also dustproof and protected against splash water. The *GigE uEye RE* is therefore particularly suited for harsh environments.

The functionality and performance data are the same as for the *GigE uEye SE*.

GigE uEye HE “High-End Edition”

The *GigE uEye HE* offers a rich set of additional features and functions compared to the other *uEye* models. Images can be output at up to 12 bits per channel. The integrated FPGA allows calculating color images in the camera, and various LUT curves can be applied to the images. An integrated 64 MB image memory and two independent processor cores ensure fast and reliable data transfer. Many of the *GigE uEye HE*'s sensors can be operated at increased frame rates. Additional programmable I/Os and a serial RS232 interface in the camera open up new possibilities for camera integration.

The Gigabit Ethernet interface provides further advantages: More than twice the bandwidth of USB 2.0, cable lengths up to 100 m, and widespread use of this interface. The Gigabit Ethernet interface is meanwhile available in every standard PC and notebook/laptop and provides a gross bandwidth of 1000 Mbps.



Figure 7: GigE uEye HE (standard and right-angle versions)

uEye Software

For every *uEye* camera, a comprehensive software package is available as a free download. In addition to the drivers, this software package features the *uEye Camera Manager*, the *uEye Demo* application and a *Software Development Kit (SDK)* for creating your own *uEye* programs under Windows 2000, XP and Vista (32 Bit) as well as Linux. Numerous demo applications make it easy for you to get started with *uEye* programming.

The latest *uEye* software is available for download from our website at <%ADRESS_WWW2%>.

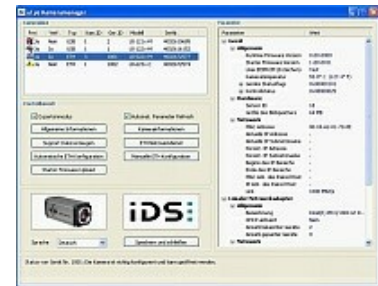


Figure 8: uEye Camera Manager

4 A: Operation

4.1 Quick Start



Connect the camera

Install the latest version of *uEye* software. Then connect the *uEye* camera with the PC. USB cameras are automatically detected as new hardware under Windows. GigE cameras are added to the camera list in the *uEye Camera Manager*. Check the status LEDs on your camera to see if the camera has been correctly identified.

See also:

- [System Requirements](#)
- [Installation](#)
- [USB uEye: Connection - Status LED](#)
- [GigE uEye: Connection - Status LED](#)



Configure the camera

USB *uEye* cameras are ready for use right out of the box. You can assign a unique ID to your camera with the *uEye Camera Manager*.

If you are using a GigE *uEye*, open the *uEye Camera Manager* first and assign an IP address before connecting the camera. You can either assign a fixed (*persistent*) IP address or have the *uEye* driver assign the address automatically. To do this, select the camera in the *uEye Camera Manager* list and click "Automatic ETH configuration" (ETH is short for "Ethernet").

See also:

- [Applications: uEye Camera Manager](#)
- [Assigning a Camera ID in the Camera Manager](#)
- [Firmware and Camera Start](#)



Capture images

The *uEye* software package includes many sample programs that you can use to try out the extensive functionality of your *uEye* camera. We recommend starting off with the *uEye Demo* application. To run the application, simply double-click the corresponding icon on your Windows desktop.

When you select *uEye > Initialize* on the menu bar, the connected *uEye* camera will immediately start capturing live images. The status bar at the bottom displays the frame rate and other important information.

If you are using a high-resolution camera, you can click *View > Render mode* on the menu bar to adjust the size of the rendered image to the application window.

See also:

- [Applications: uEye Demo](#)
- [Camera Basics: Operating Modes](#)



Customize the key camera properties

Select *uEye > Properties...* on the menu bar to open the dialog box for modifying the camera properties.

The *Camera* tab provides all the parameters for adjusting the camera's speed. You can increase the pixel clock to run the camera at a higher frame rate. Reduce the pixel clock if transmission errors occur too often. When you enable *Optimum*, the optimum pixel clock will be set automatically.

On the *Image* tab, you find various sensor gain controllers. Use the *Master gain* controller to increase image brightness if no longer exposure time setting is possible. Switch to the *AES/AGC* tab to enable the Auto Exposure Shutter (*AES*) and Auto Gain Control (*AGC*) features.



Select a low sensor gain to minimize visible noise.

If you are using a color camera, you should activate sensor color correction on the *Color* tab in order to achieve rich vibrant colors for on-screen display. To adapt a color camera to the ambient light conditions, it is essential to carry out Auto White Balance (*AWB*). Aim the camera at a surface of a uniform gray color, then enable the *Image white balance: Enable* and *Run once* check boxes on the *AWB* tab.

See also:

- [uEye Demo: Camera Properties](#)
- [Camera Basics: Camera Parameters](#)



Activate trigger and flash modes

uEye cameras provide the possibility to trigger the image capture and to have the flash controlled by the camera. To switch the camera to trigger mode, go to the camera properties as described above, select the *Trigger* tab and enable the desired mode. To trigger on *falling* or *rising edges*, a digital signal has to be applied to the camera. When you are finished with the trigger settings, select *uEye > Trigger mode...* on the menu bar to start the triggered image capture.

If you have connected the digital output on your *uEye* camera to a flash controller, you can configure the flash function on *Input / Output* tab. Enable *Flash high active* and *Global exposure window*. This way, the *uEye* automatically activates the flash during the exposure time.



The housing version of the *uEye LE* cameras provides no digital I/Os.

See also:

- [uEye Demo: Camera Properties](#)
- [Camera Basics: Digital Input/Output](#)
- [Specifications: Electrical Specifications](#)



Save the camera settings and images

With *uEye Demo*, saving single frames or videos is very easy to do. Just choose the relevant option on the *File* menu. If you have recorded AVI videos, you can play them using the supplied *uEye Player*.

When you have made specific settings for a camera and want to save them so that you can use them again the next time you start the program - or any other *uEye* program - select the *Save parameters* function to save all the camera's properties to an ini file or to the camera memory (*parameter set 1 / 2*). To load the saved settings, select the *Load parameters* option.

See also:

- [uEye Demo: Record Dialog](#)
- [uEye Player](#)

4.2 Installation

4.2.1 System Requirements

For operating the *uEye* cameras, the following system requirements must be met:

USB *uEye* cameras

	Minimum *)	Recommended
CPU speed	600 MHz	2 x 2.4 GHz
Memory (RAM)	256 MB	2048 MByte
For <i>USB uEye</i> cameras: USB host controller	USB 2.0 high speed (480 Mbps)	USB 2.0 high speed (480 Mbps) Intel® or NVIDIA® nForce mainboard chipset
For <i>GigE uEye</i> cameras Network bandwidth Network card type	100 Mbps ---	1000 Mbps Intel Pro/1000 GT (PCI) Intel Pro/1000 PT (PCIe)
Graphics card	Onboard graphics chip	AGP/PCIe graphics card Latest version of <i>DirectX Runtime</i> 9.0c
Operating system	Windows XP 32 bit (Service Pack 2) Windows 2000 (Service Pack 4) Linux (Kernel 2.6)	Windows 7 32 or 64 bit Windows Vista 32 or 64 bit (Service Pack 1) Windows XP 32 bit (Service Pack 3) Linux (Kernel 2.6)

*) Camera performance may be limited.



Onboard USB 2.0 ports usually provide significantly better performance than PCI and PCMCIA USB adapters.



To ensure optimum performance of the network connection, you need to install the latest drivers for your network card. We recommend using the drivers of the following versions:

- Intel® chipsets: version 8.8 or higher
- Realtek chipsets: version 5.7 or higher

Direct3D graphics functions

The *uEye* driver can use Direct3D to display the camera image with overlay information. On Windows systems, you can use the supplied *DXDiag* diagnostic tool to check whether your graphics card supports Direct3D functions. To start the diagnostic tool, click *Run...* on the Windows Start menu (shortcut: Windows+R) and enter "DXDiag" in the input box.

On the *Display* page of the diagnostic tool, click the button for testing the Direct3D functions.

4.2.2 Downloading the Software

The latest camera drivers are available for download at www.ueyesetup.com.



Figure 9: uEye Setup - Language selection

Choose your language on the setup start page and follow the download links.



Figure 10: uEye setup menu



To operate a camera with USB board revision 2.0 or earlier, you will need the uEye driver version 2.40. You can download this driver version from our website at <http://www.ids-imaging.com>. For further information, see also [USB uEye SE Driver Compatibility](#).

The following options are available:

- **Windows Setup (V 3.30)**
This download contains the complete setup with drivers for all cameras, the *uEye Software Development Kit (SDK)* and the manuals.

- *"Driver only" Windows Setup*
This download contains only the drivers for USB and Gigabit Ethernet cameras. The SDK and manuals are not included.
- *USB Bus Checker*
The USB bus checker provides information on the USB interfaces available on your system (Windows only).
- *Camera Hardware Check*
The USB hardware check displays information on whether a connected camera is compatible with the new driver versions (Windows only). To use this feature, a uEye driver has to be installed on your system.
- *LINUX*
This download contains the drivers for the uEye cameras for Linux (kernel version 2.6)
- *Imaging Software Interfaces*
Click this link to download individual software interfaces for using the uEye in conjunction with image processing libraries, such as MVTec HALCON. Please note that all these files are also included in the *Windows Setup (V 3.30)* download (see above).
- *Manuals*
Click this link to access the *uEye Manual* and other manuals for components and tools online as PDF files.
- *RMA Form*
This link displays a form for returning goods to IDS.
- *Support*
Click this link to display IDS support information and additional contact data.

4.2.3 Installing the uEye software under Windows



You need administrator privileges to install the software.

The files are downloaded in ZIP format. They have to be extracted after the download before you can start installing. Double-click the executable (*uEye32_35000.exe*) to start the installation. The uEye driver installation is menu-driven. Please follow the instructions of the setup program.

The setup will prompt you to select a setup type. Please choose one of the following options:

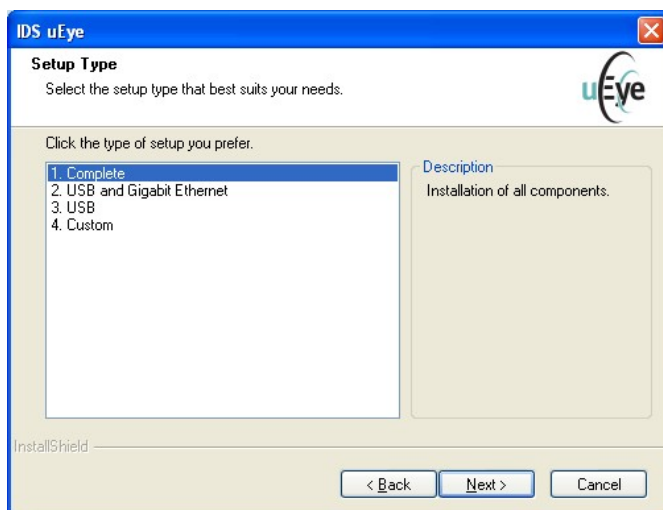


Figure 11: Selecting a setup type

1. **Complete**
Installs all components (recommended).
2. **USB and Gigabit Ethernet**
Installs all USB and GigE components except source code and third-party drivers (e.g. DirectShow or imaging libraries)
3. **USB**
Installs all USB components except source code and third-party drivers
4. **Custom**
When you choose custom installation, you have to individually select the components you want to install. Custom installation is recommended only for advanced users.
If you need to make changes to an existing installation, you can do this with the custom installation.



Once the software has been installed, the *GigE uEye* network service is automatically bound to all local network adapters.

We recommend disabling the network service for all network adapters that will not be used with the *GigE uEye* cameras. To disable the network service, open the [ETH network service](#) dialog box in the *uEye Camera Manager*.

Uninstall

To uninstall the *uEye* drivers and software, you also use the menu-driven *uEye* setup program. The *GigE uEye* network service is uninstalled automatically when you uninstall the *uEye* driver.



After uninstalling the *GigE uEye* driver, you will have to restart your computer. You can only reinstall the driver after restarting the computer.

4.2.4 Installing the uEye Software under Linux

The installation of the *uEye* software on Linux systems is described in the *Readme.txt* file contained in the *uEye* driver download for Linux .

4.2.5 Connecting a USB uEye Camera

Please install the software first as described in the [Installing the uEye Drivers](#) section. Connect the *USB uEye* to the PC, using the USB 2.0 cable. The camera will be recognized automatically. When the camera has been correctly installed, the LED on the back of the camera lights up green.

Under Windows the camera will show up in the *uEye Camera Manager's* camera list.

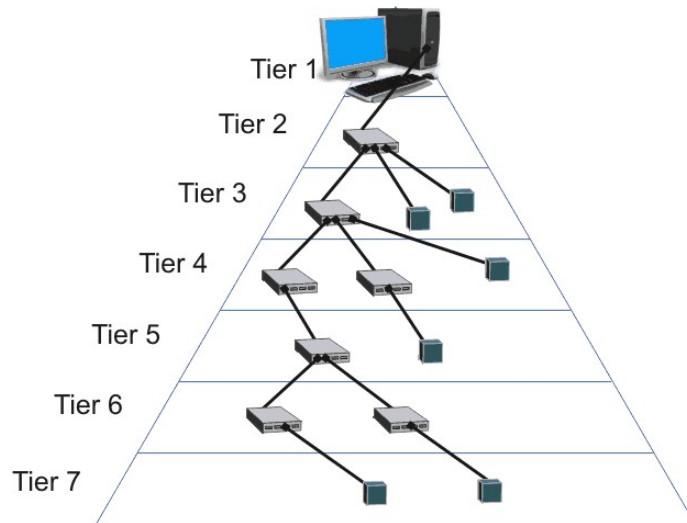


Figure 12: Connecting USB device to a PC

The *uEye* cameras can be connected to a USB port either directly or via hubs and repeaters. A wide range of different hubs and repeaters are available at computer stores or from IDS. The USB 2.0 hubs being used must be *full powered* hubs that are able to provide 500 mA per USB port. *Low Powered* hubs, in comparison, only supply 100 mA per port, which is not sufficient for *uEye* cameras.



To use maximum bandwidth, we recommend connecting the cameras directly to the USB 2.0 ports on the mainboard. Many USB 2.0 ports on PCI/PCIe cards and the USB 2.0 ports on the front of the PC mostly supply lower bandwidths.

Windows Systems



If the camera is not listed in the *uEye Camera Manager*, open the Windows Device Manager to check whether the camera has been correctly recognized. If recognition was successful, you will find an entry in the format "uEye UI-xxx-xx Series" under "Universal Serial Bus Controllers." A question mark or exclamation mark before the entry indicates that camera was not correctly recognized. Disconnect and reconnect the camera. The camera should now be correctly recognized.

4.2.5.1 Status LED on USB uEye cameras

USB uEye SEME//RE

The LED on the rear side of the *USB uEye* camera indicates whether

- the uEye camera is powered on – LED lights up red (only USB board rev. 2.0 or higher).
- the uEye driver has been loaded and the camera is operational – LED lights up green
- an error has occurred – green LED flashes:
 - 2x flash: unknown sensor, please contact our [support](#) team.

If the LED does not light up green, please check the following:

- Has the camera been connected correctly?
- Have the driver and the camera been installed properly in the [uEye Camera Manager](#) on the host PC?
- Does the host PC meet all [system requirements](#)?



Figure 13: USB uEye SE
revision 1.2 (green Status LED)



Figure 14: USB uEye SE revision 2.0
(red/green LED)

USB uEye LE

The *USB uEye LE* camera has a single color status LED. It lights up orange as soon as the camera is supplied with power.

4.2.5.2 USB uEye SE Driver Compatibility



From driver version 3.10 on, only cameras with USB board revision 2.0 or higher are supported. To operate a camera with USB board revision 2.0 or earlier, you will need the uEye driver version 2.40. You can download this driver version from our website at <http://www.ids-imaging.com>.



Windows systems only: You can use the *USB Hardware Check* (see [Software Installation](#)) before installing the driver version 3.10 to check whether your camera is supported. In addition, the *uEye Camera Manager* version 3.10 or above provides information about the compatibility (see [Camera Manager](#)). An incompatible camera will be displayed as *free* and *not available*.

Only the following CMOS camera models of the *USB uEye SE* series are affected:

- UI-121x
- UI-141x
- UI-144x
- UI-154x
- UI-145x
- UI-146x

The LED(s) on the back of the camera housing also indicate the USB board version (see [uEye USB Status LED](#)).

Note on the uEye memory board



The optional memory board of the *USB uEye SE* and *USB uEye RE* camera series has been discontinued.

From version 3.30, the functions required to operate the memory board will no longer be supported in the *uEye* driver.

The *uEye* driver version 3.24 that still supports these functions will continue to be available in the download area of our website at <http://www.ids-imaging.com>.

4.2.6 Connecting a GigE uEye Camera

Please install the software first as described in the [Installing the uEye Drivers](#) section.

Check the power supply to the camera. Suitable AC adapters are available as accessories (see also [GigE uEye HE Accessories](#) and [GigE uEye SE Accessories](#)). Connect the camera to the PC either directly or using switches.

Connecting the camera directly to the PC

Using a suitable network cable (e.g. Cat 5e), connect the uEye camera directly to a Gigabit Ethernet port on your PC. With this connection type, you need a network card for each camera.

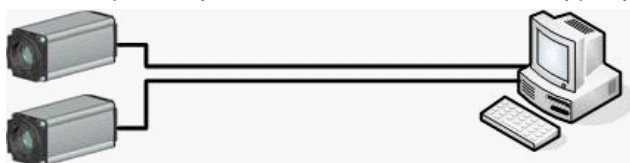


Figure 15: Connecting two cameras directly to a PC

Connecting the camera to a PC via switches

The use of switches allows you to extend the line length, as each switch adds a segment. The maximum cable length for each segment is 100 m.



Using the *GigE uEye* camera through a router is not supported.

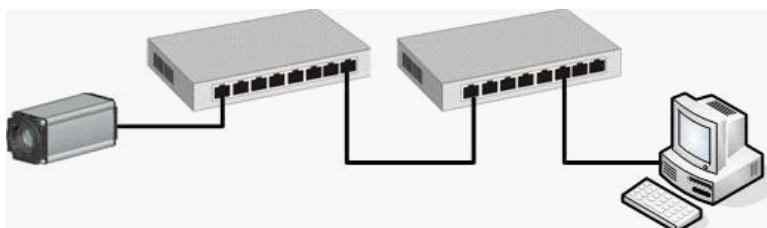


Figure 16: Connecting a camera to a PC, using switches to extend the line length

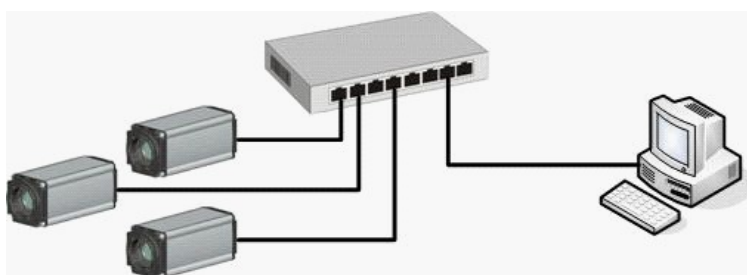


Figure 17: Operating multiple cameras via switch

Connecting multiple cameras to multiple PCs using switches

As soon as one of the cameras is used by a PC, it is visible to other PCs, but no longer available. It can only be used by a different PC when the existing connection to the first PC has been closed.

If the two PCs are on different subnets, each PC can only work with the cameras that have been configured for the relevant subnet.

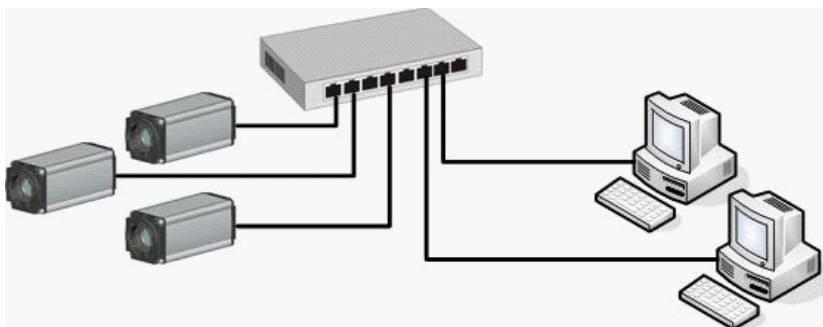
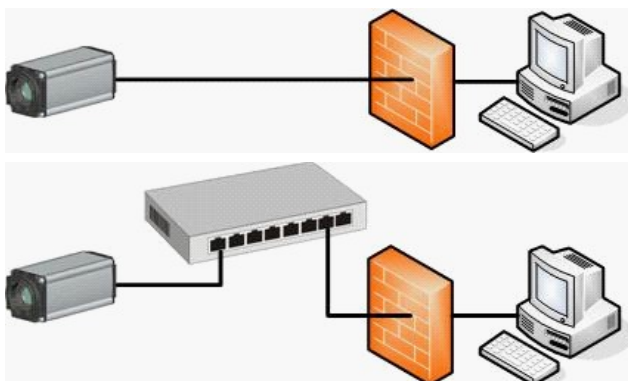


Figure 18: Multiple cameras and PCs networked using a switch

Connecting the camera to a PC behind a firewall



The use of external firewalls between the camera and the host PC is not recommended on image data networks. If you nonetheless want to use a firewall, please ensure that ports 50000, 50001, 50002 and 50003 are open for the UDP protocol.

The built-in Windows Firewall or personal firewalls installed as software in the host PC usually do not cause any problems.

4.2.6.1 Status LED on GigE uEye cameras

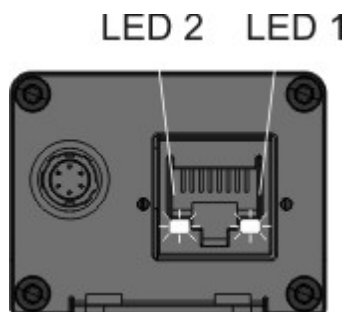


Figure 19: GigE uEye SE/RE status LEDs

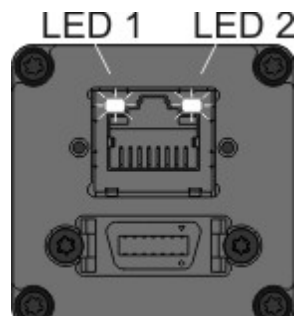


Figure 20: GigE uEye HE status LEDs

The two LEDs indicate the current status of the *GigE uEye* camera.

- LED 1: camera status
- LED 2: network status

Camera status (LED 1)

Camera is booting

Camera off

Starter firmware ok, waiting for connection

Connecting

Firmware update

Normal operation

Freerun mode

Single trigger mode

Group trigger mode *)

Standby mode

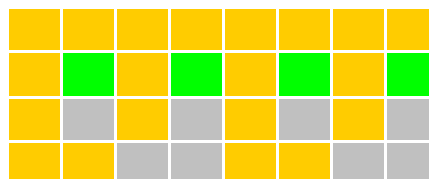
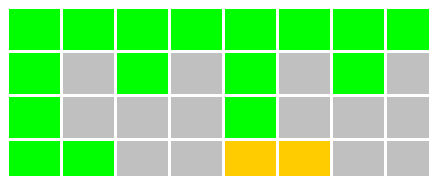
Error

Camera hardware error

Configuration error

Starter firmware not ok, failsafe firmware enabled

Overtemperature (>65°C)



Network status (LED 2)**Normal operation**

No network connection

Network connection OK + data transfer

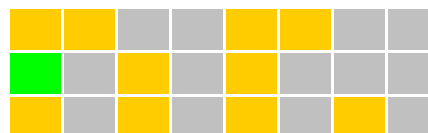
Continuous Data transfer

**Error**

Network error

Camera hardware error

Cable fault *)



*) This function is not supported yet.

4.2.6.2 Notes on connecting GigE cameras

Please read the following notes before setting up *GigE uEye* cameras:

Network card drivers

- To ensure optimum performance of the network connection, you need to install the latest drivers for your network card.

Cabling

- The cable length of the individual segments must not exceed 100 m.
- The network must be cabled throughout at either 100 Mbps or 1000 Mbps. The use of 1000 Mbps networks is recommended.
- The data network and the camera network should be cabled separately. We therefore recommend connecting the *GigE uEye* cameras by a separate network card.
- GigE network adapters for PCI slots do not achieve the maximum transfer rate of 114 MB/sec. In real life conditions, the transfer rate on a PCI bus is approx. 80-90 MB/sec.

IP configuration and DHCP

- Before you can use the *GigE uEye* camera on the network, you need to assign a valid IP address to the camera. The IP address is assigned in the *uEye Camera Manager*. When manually assigning an IP address to the camera, please make sure that the camera is on the same subnet as the connected PC.

The network card can be configured in different ways:

1. Network card with fixed (persistent) IP address:
The network card has a fixed IP address. The *uEye Camera Manager* reads the network card's IP address and uses it to create a valid address range from which an IP address can be automatically assigned to a *GigE uEye* (see also [Automatic ETH Configuration](#)).
2. Network card with DHCP, DHCP host exists
The network card dynamically receives its IP address from an external device (the DHCP host). When using this configuration method, please make sure that the auto IP range used by the *uEye Camera Manager* and the address range maintained by the DHCP host do not overlap. Select a high lease time (ideally infinite) to ensure that the address assigned by the DHCP host cannot change during camera operation. The assignment of a new IP address during operation of the *GigE uEye* camera will cause communication failure between the PC and the camera.
3. Network card with DHCP, no DHCP host exists
The network card is configured for DHCP assignment of the IP address, but there is no DHCP

host. (This may be the case, for example, when you disconnect a laptop from the LAN in order to connect it to a GigE uEye instead.) In this case, the system assigns a link local address in the range between 169.254.1.0 and 169.254.254.255 after the relevant timeout expires. Assigning the new address may take up to a minute. You can then configure the *GigE uEye* by using configuration method 1 described above.



GigE uEye cameras do not support DHCP assignment of the camera's IP address. Only the network card can obtain its IP address from a DHCP server.

We recommend not making any changes to the DHCP configuration of a network card during operation. If problems occur, restart the system with the desired configuration.

Advanced settings

- For operating *GigE uEye* cameras, we recommend setting the value for the receive descriptors of the network connection to the maximum value. Please note that not all network cards provide this option.
To set the receive descriptors, select *Start* → *Settings* → *Network Connections*. Right-click on the network connection and choose *Properties*. Switch to the *Advanced* tab in the dialog box and click the *Configure...* button. You can now set the receive descriptors (*Rx/Tx*).
- The UDP protocol is required for communication between the *GigE uEye* cameras and the computer. Ports 50000 ... 50003 must be available for this purpose.

4.3 Applications

4.3.1 uEye Camera Manager

The *uEye Camera Manager* is the central tool for managing all *uEye* cameras. It displays information on the connected *uEye* USB and *GigE* *uEye* cameras and provides options for configuring them.



The *uEye Camera Manager* is currently only available for Windows operating systems. You can configure the cameras in Linux by using the *uEye SDK*.

The *uEye Camera Manager* can be accessed as follows:

- Start → Programs → IDS → *uEye* → *uEye Camera Manager*
- Program icon on the desktop or Quick Launch toolbar
- Start → Control Panel → *uEye Camera Manager*

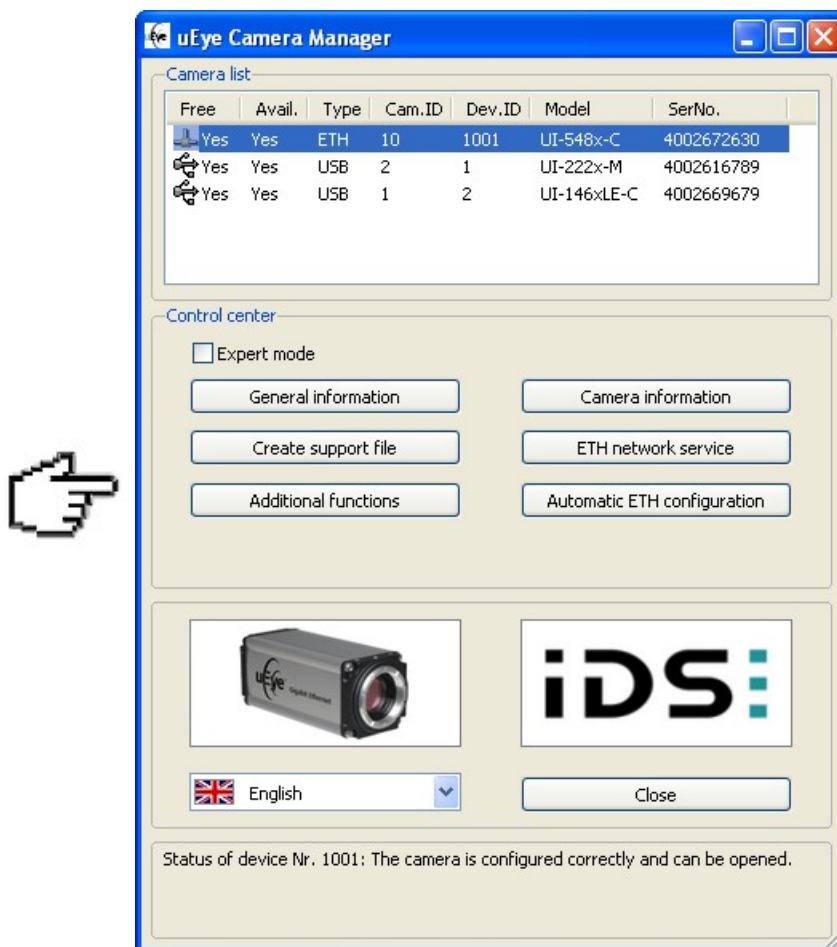



Figure 21: *uEye Camera Manager* (basic view)

- **Camera list**
The camera list displays information on the connected *uEye* cameras.
- **Control center**
In the control center, you can access the configuration and display detailed information on the connected *uEye* cameras.
- In the English drop down box, you can choose the language for the *uEye* Camera Manager. This setting is saved and remains effective even after you close and reopen the

program.

For proper display of Asian languages, the Windows support for East Asian languages has to be installed on your system (in *Control Panel* → *Regional and Language Options*).

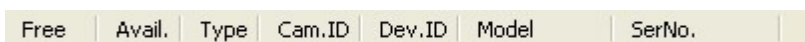
- Click  to close the application; any settings you have made are saved.
- The status box at the bottom of the *Camera Manager* window indicates the current status of the selected camera. If it is available, the status message is shown in black. Otherwise, the status message is shown in red.

If an error has occurred in a camera, a black exclamation mark on a yellow background is shown next to the camera. The status box then indicates the cause of the error and suggests remedies.

4.3.1.1 Camera List

When a camera is activated (switched on or connected to the PC), it appears in the *camera list* of the *uEye Camera Manager* after a few seconds. A Gigabit Ethernet camera requires a little more time to start up and be detected by the network than a USB camera.

After deactivating a *GigE uEye* camera (switching it off or disconnecting it from the network) it takes approximately three seconds before the camera is removed from the camera list. During this time the computer waits to see whether it receives another heartbeat signal from the camera.



Free	Avail.	Type	Cam.ID	Dev.ID	Model	SerNo.
------	--------	------	--------	--------	-------	--------

Figure 22: uEye Camera Manager - Camera list

The data shown in the camera list can be sorted in ascending or descending order by left-clicking on the respective column header.

- *Free/Avail.*
Free indicates whether a camera is currently in use.
Avail. (*Available*) indicates whether a camera can be opened by this computer with the current setup (computer and camera).
 Cameras shown with a red x are currently in use (*Free* = *No*) and are not available (*Avail.* = *No*).
 Cameras shown with an exclamation mark are not in use, but are currently unavailable for various reasons, such as:
 - The camera is not compatible with the driver. Please update the *uEye* driver.
 - The IP configuration of the network card is not configured for use of the *GigE uEye* camera. Please enter a valid configuration in the Manual ETH Configuration.
 - DHCP (automatic assignment of an IP address) is activated in the IP configuration of the network card. Please enter a valid configuration in the Manual ETH Configuration.
 - The driver has not properly detected (initialized) the camera. Please disconnect the camera from the PC and then reconnect it.
 - The camera is currently being removed from the Manager.
 - The camera reports that it is *Not operational*.
- *Type*
 This column indicates whether the camera is a Gigabit Ethernet (*ETH*) or a USB camera (*USB*).
- *Cam.ID*
 The camera ID assigned by the user.

- *Dev.ID*
Unique device identifier sequentially assigned by the system. Different device IDs are assigned for USB and Gigabit Ethernet cameras. USB cameras are assigned device IDs from 1 upwards, Gigabit Ethernet cameras from 1001. After deactivating a *uEye* camera (switching it off or disconnecting it from the network), the device ID is no longer valid and can be assigned again by the system.
- *Model*
Model name of the camera
- *SerNo.*
Serial number of the camera.

4.3.1.2 Control Center

- *Expert mode*
When you select the ☒ Expert mode check box, the *uEye Camera Manager* additionally displays the *Parameters* box on the right. There you will find detailed information on the *uEye* camera selected in the *camera list*. The and buttons are only available in Expert mode and are hidden otherwise

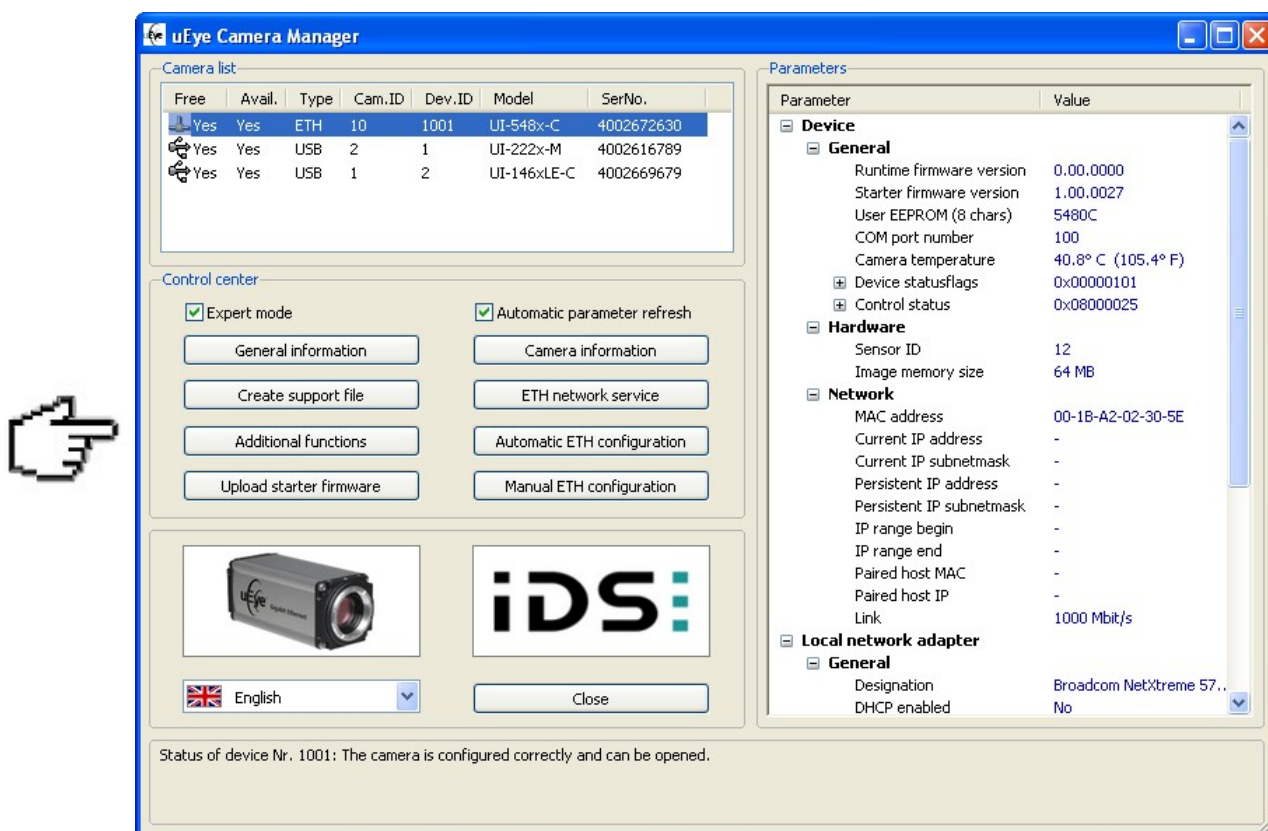


Figure 23: uEye Camera Manager in Expert mode

- *Automatic parameter refresh*
If you select the ☒ Automatic parameter refresh check box, the data shown in the tree structure is updated periodically. If the option is disabled, the data in the tree structure is only updated when a different camera is selected.

All other *Control Center* buttons are described in detail in the following sections.

4.3.1.3 General Information

This dialog box provides information on the installed *uEye* drivers and the available USB controllers and network adapters.

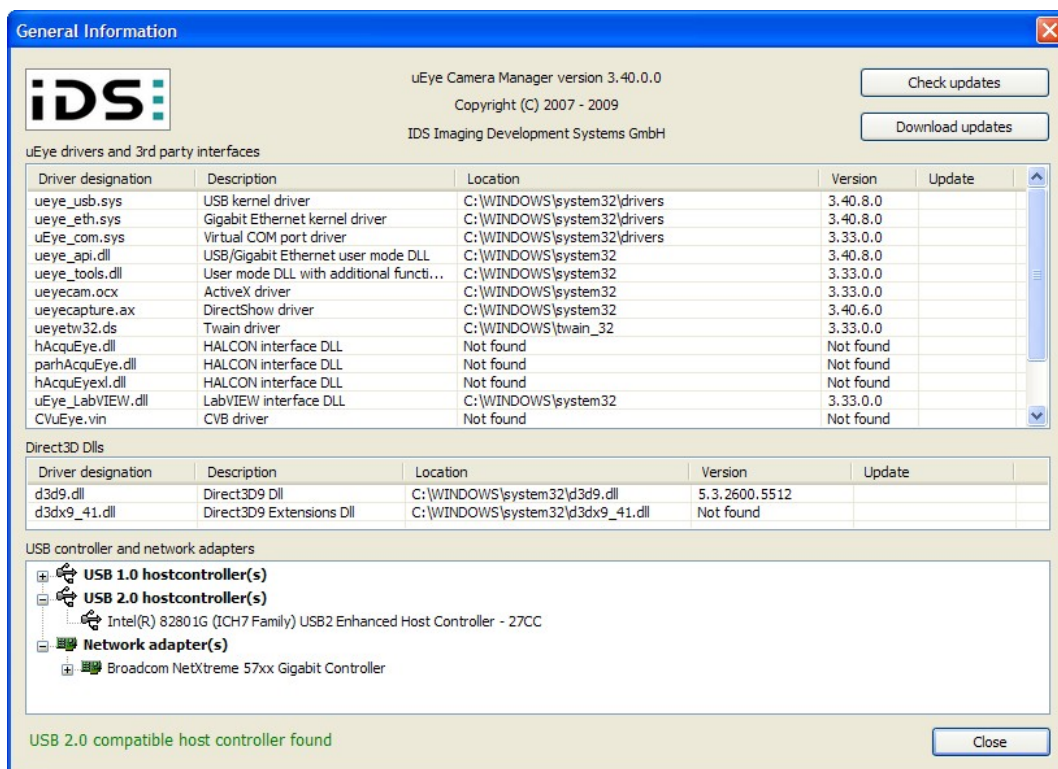


Figure 24: uEye Camera Manager - General information

-

Click this button to check whether new drivers are available on the IDS website. You need a connection to the Internet to use this function. After the version check, the individual files are highlighted by different background colors in the *uEye drivers* and *3rd party drivers* lists:

Version	Update	
3.10.0.0	3.20.0.0	Red: A new driver version is available. It is recommended that you update the software.
3.20.0.0	3.20.0.0	Green: The installed version is up-to-date.
3.21.1.0	3.20.0.0	Yellow: The installed version is more recent than the version on the website.
Nicht vorh.	3.20.0.0	Gray: A file available on the website has not been installed.

-

Click this button to go to the <http://www.ueyesetup.com> website and download the *uEye* software and drivers.

- *uEye drivers*
This list shows the location and version of the *uEye* driver files installed on your system.
- *3rd party drivers*
This list shows the location and version of the *uEye* interface files that have been installed on your system for third-party software.
- *USB controller and network adapters*
All USB controllers and network adapters that are available in your system are shown in a tree structure.

4.3.1.4 Camera Information

In the *Camera information* dialog box, you can assign a unique *ID* to the selected camera and write to the user area of the EEPROM. The data you enter is retained in the camera memory even when the camera is disconnected from the PC or power supply.

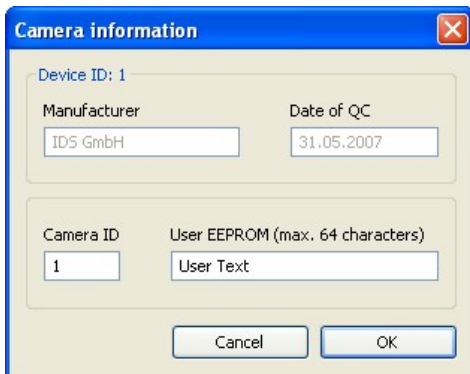


Figure 25: uEye Camera Manager – Camera information



You can only set the camera ID and write to the EEPROM if the camera is marked *Free* and *Available* in the Camera Manager (see also [Camera list](#)).

- **Camera ID**
The camera ID identifies a camera in multi-camera operation. You can assign IDs in a range from 1 to 254. The default value for the camera ID is 1. The same ID can be assigned to multiple cameras. You do not have to assign sequential ID numbers to all connected cameras.



If software accesses cameras through the *uEye DirectShow interface*, the camera IDs must be in a range from 1 to 8.

If software accesses cameras through the *uEye DirectShow interface*, the camera IDs must be assigned consecutively beginning with 1.

- **User EEPROM (max. 64 characters)**
Every *uEye* has a 64-byte user area in its EEPROM (Electrically Erasable and Programmable Read Only Memory) to which you can write text of your choice.


The *Camera information* dialog box displays two additional boxes that are for your information only and cannot be edited:

- *Manufacturer* (e.g. IDS GmbH)
- *Date of QC* (date of final camera quality test)

4.3.1.5 Creating a Support File



A *uEye* support file is a binary file with the extension *.bin*. The file contains camera and driver details that are required for diagnostics by the *uEye* support team. No personal computer data or user data is stored in this file.

The  button opens the "Save as" dialog box, where you can save the

displayed camera information and additional driver information to a file.

4.3.1.6 ETH Network Service

In this dialog box, you can enable and disable the network service of the *GigE uEye* camera for specific network adapters. In addition, network adapters can be assigned a fixed IP address, which is required for operating the *GigE uEye* camera.

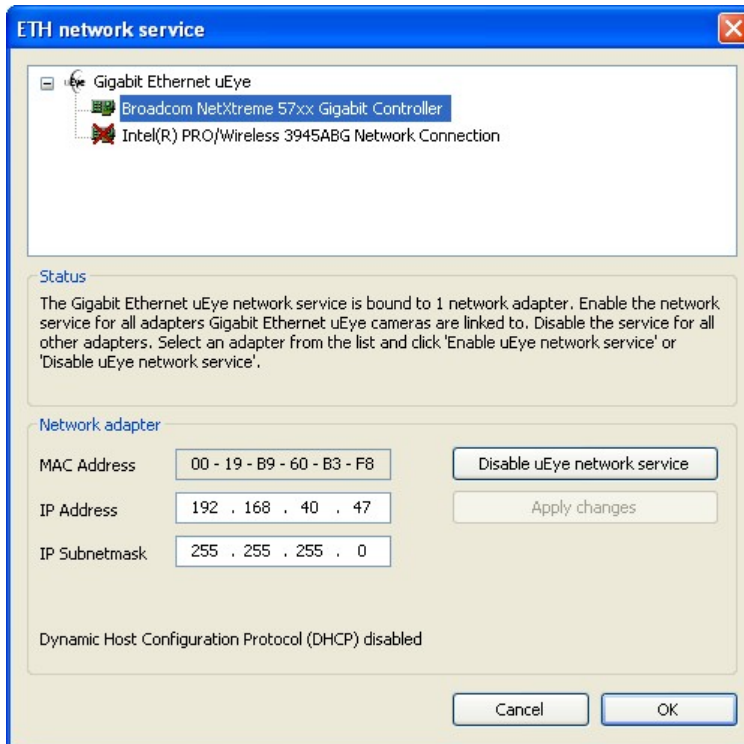




Figure 26: uEye Camera Manager - ETH network service

- **Status**
Displays information on the status of the *GigE uEye* network service and the connected network adapters.
- **Network adapter**
 - Click the  button to disable the *GigE uEye* network service for a network adapter. To enable the network service, click the  button. Before you can enable or disable a network adapter, you need to select it in the tree structure of the dialog box.
We recommend disabling the *GigE uEye* network service for all network adapters that are not being used for *uEye* cameras.
 - In the *IP Address* and *IP Subnetmask* input boxes, you can assign a static IP address and a static IP subnet mask to the selected network adapter. DHCP will be disabled automatically.

4.3.1.7 Additional Functions (COM Port)

The *Additional functions* dialog box allows installing virtual COM ports for communication through the serial interface of the *GigE uEye HE* camera. The following sections show you how to set up and use the serial interface.



This feature is only available for *GigE uEye HE* cameras.

You need administrator privileges to install a virtual COM port.

The *GigE uEye HE* camera you select in the Camera Manager has to be marked *Free* and *Available*.

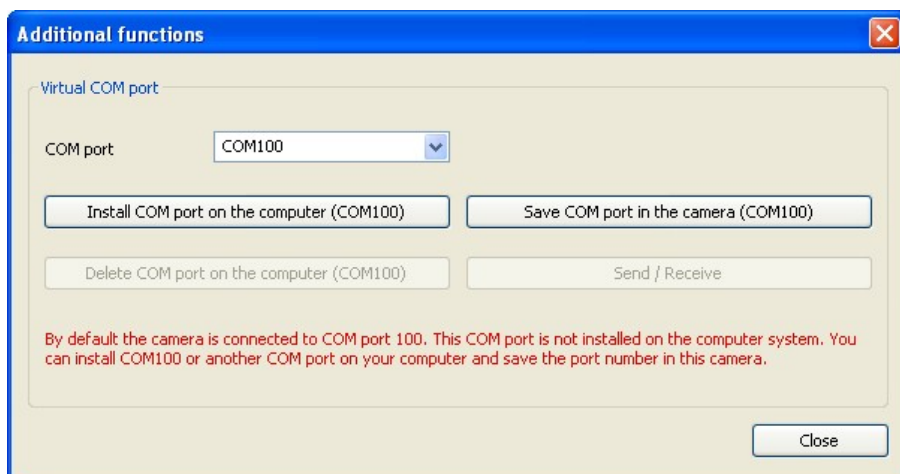


Figure 27: uEye Camera Manager - Additional functions

Setting up the serial interface on the *GigE uEye HE*

Before using the serial interface on the camera, one or more virtual COM ports have to be installed on the PC. Most systems support up to 255 COM ports; COM1 to COM8 are often assigned operating system functions by default. You can check the current port assignment in the *Device Manager* on your computer. Some older systems may not have more than eight ports; in that case you will need to assign the *GigE uEye HE* camera to one of these ports.

- **COM port**

In the drop down box, select the number of the port you want to install (default: 100). COM ports in use are marked (*used*) in the list.

- **Install COM port on the computer (COM100)**

Click this button to install the selected virtual COM port.

During the first installation of a virtual COM port, an additional broadcast port with number 255 is installed. Data sent to this port will be forwarded to all paired cameras.

You can install any number of virtual COM ports on a single system.

- **Delete COM port (COM100)**

With this button, you can release a COM port that is marked "used." If the port number has been saved in that camera, it will be deleted in the camera, too. To release a COM port, select it in the drop down box and then click this button.

- **Save COM port in the camera (COM100)**

Click this button to assign the selected port number to the camera. The port number is saved in the camera's non-volatile memory and retained even when the camera is switched off. You can look up the assigned port number in the *Camera Manager's* expert mode. A COM port number can also be saved in a camera without a virtual COM port installed on the PC.



If you want to control more than one *GigE uEye HE* camera from a PC, each camera should be assigned a unique port number. If multiple cameras are assigned the same port number, only the port of the first camera that is opened will be used.




To send data via the serial interfaces of multiple cameras, you can use the broadcast port with number 255. Before connecting to the broadcast port, ensure that all the cameras that are to receive the broadcast have been opened.

Testing the serial interface on the *GigE uEye HE*



Before a camera can exchange data with a PC through the virtual COM port, the camera has to be paired with that PC (see *Paired*).

To avoid transmission errors, please ensure that both the camera and the receiving end use the same communication parameters (baud rate, data bits, stop bits, parity). Further information on the communication parameters is provided in the Specifications: Serial Interface chapter.

- 

Clicking this button opens a dialog box for transferring data through the COM port. The dialog box is provided as the *uEyeComportDemo.exe* sample program together with the C++ source code and is included in the *uEye SDK*. This program allows sending ASCII characters to the COM port assigned to a camera. The characters are output unchanged on the camera's serial port. To check the proper functionality, you can connect a PC to the camera's serial port and read the transmitted characters on the PC's COM port.

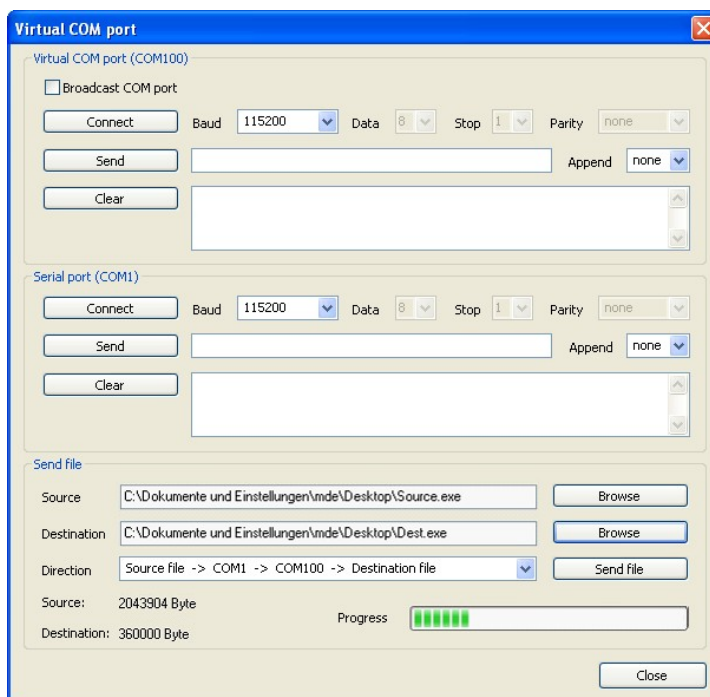



Figure 28: Data transfer through a virtual COM port

- Baud**
 In this drop down box, you can change the data transfer rate of the serial interface.
- Append**
 This drop down box allows appending the special characters *CR* (Carriage Return) and *LF* (Line Feed) to the ASCII text you want to transmit. Some devices with serial interface require ASCII strings to be terminated with CR/LF.
- Send file**
 Using these functions, you can send a file in either direction (output on the camera's virtual COM port or output on the PC's COM port).



Since the sample program has to open the camera, please make sure the selected camera is not used by other applications at the same time.


4.3.1.8 Automatic ETH Configuration

The  button allows configuring a connected *GigE uEye* camera for automatic IP address assignment. The function defines a suitable IP address range, which you can change in the [Manual ETH configuration](#) dialog box, if required. At the same time, it deletes the camera's persistent IP address (i.e. sets it to 0.0.0.0). When the *GigE uEye* camera is opened by an application, the function automatically assigns a free IP address to the camera.



This function is only available for *GigE uEye* cameras.

4.3.1.9 Starter Firmware Upload

The  button uploads a new version of the starter firmware to the selected camera. This button is only available in [Expert mode](#) and is hidden otherwise.




The starter firmware determines the start-up behavior of the *GigE uEye* camera. We recommend that you do not update the starter firmware unless an older firmware version causes start-up problems. If you have questions on the current starter firmware, please contact our technical support (see [Contacting Us](#)).



This function is only available for *GigE uEye* cameras.

4.3.1.10 Manual ETH Configuration

This dialog box allows you to manually set the IP address and subnet mask of a *GigE uEye* camera. The  button is only available in [Expert mode](#) and is hidden otherwise.



This function is only available for *GigE uEye* cameras.

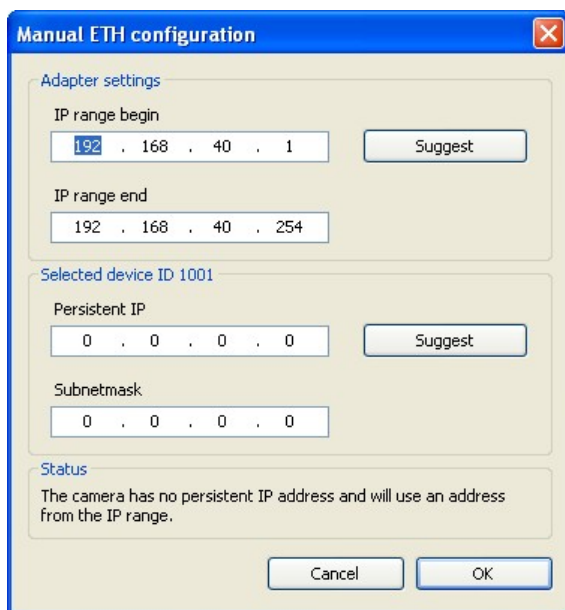


Figure 29: uEye Camera Manager - Manual ETH configuration

- **Adapter settings**
Here, you can change the IP configuration of the network adapter to which the selected *GigE uEye* camera is connected.
- **IP range begin/end**
Defines the IP range from which the *uEye* driver chooses an address during *automatic ETH configuration*. The IP range is not effective if the camera is assigned a persistent IP address.
- **Suggest**
Same as the *Automatic ETH configuration* function.
- **Selected device ID**
Here, you can change the IP configuration of the *GigE uEye* camera.
- **Persistent IP**
The entered IP address is permanently saved in the camera when you close the dialog box. The persistent IP address is retained in the camera memory even when the camera is disconnected from the power supply. If you connect the camera to a different PC, make sure the persistent IP address is valid on that computer, as well.
- **Subnet mask**
Enter a valid subnet mask for the persistent IP address.
- **Suggest**
Selects the first free IP address from the network adapter's range of valid IP addresses. The function then enters that address in the *Persistent IP* box and sets the appropriate subnet mask. The settings made for *IP range begin/end* in the Adapter Settings box have no influence on the suggested address.
- **Status**
This box displays information on the defined IP configuration.



If a DHCP server is running on the network, you need to ensure when configuring the network adapter that the manually assigned address range of the *uEye* driver is outside the DHCP range.

4.3.1.11 Parameters

This box displays the parameters of the camera you have selected in the camera list. The Parameters box is only shown when *Expert mode* is active.

The parameters are organized in a tree structure. Only the information that applies to the selected camera is shown. The data displayed in the *camera list* is not repeated in the *Parameters* box. The data shown in the tree structure cannot be changed.

	Device	
	General	
	Runtime firmware version	0.00.0000
	Starter firmware version	1.00.0027
	User EEPROM (8 chars)	5480C
	COM port number	100
	Camera temperature	39.0° C (102.2° F)
	Device statusflags	0x00000101
	Control status	0x08000033
	Hardware	
	Sensor ID	12
	Image memory size	64 MB
	Network	
	MAC address	00-1B-A2-02-30-5E
	Current IP address	192.168.40.1
	Current IP subnetmask	255.255.255.0
	Persistent IP address	192.168.40.1
	Persistent IP subnetmask	255.255.255.0
	IP range begin	-
	IP range end	-
	Paired host MAC	-
	Paired host IP	-
	Link	1000 Mbit/s
	Local network adapter	
	General	
	Designation	Broadcom NetXtreme 57...
	DHCP enabled	No
	Number of known devices	1
	Number of paired devices	0
	Network	
	MAC address	00-19-B9-60-B3-F8
	IP address	192.168.40.47
	IP subnetmask	255.255.255.0
	IP range begin	192.168.40.10
	IP range end	192.168.40.254
	IP range valid	Yes
	Settings	
	Packet filter	block UEGET
	Local driver	
	Min. compatible starter FW	0.00.0027
	Max. compatible starter FW	255.255.65535

Figure 30: uEye Camera Manager - Parameter list

- USB *)

- *Hub*
Indicates which hub and port a USB camera is connected to. In addition, the full path through all hubs to the USB controller on the computer is displayed.
- *Controller*
Indicates the USB controller to which the camera is connected.
- *Device*
 - *Sensor ID* *)
 - *General*
 - *Runtime firmware version* **)
 - *Starter firmware version* **)
 - *User EEPROM*
The first 8 characters of the user area in the EEPROM are displayed (see [Camera Information](#)).
 - *COM port number* **)
Number of the virtual COM port stored in the camera's memory (see [Serial Interface \(RS232\)](#)).
 - *Camera temperature* **)
Indicates the camera temperature in degrees Celsius.
 - *Device statusflags* **)
Internal camera status flags
 - *Control status* **)
Internal camera status flags
 - *Hardware* **)
 - *Sensor ID* **)
 - *Image memory size* **)
 - *Network* **)
 - *MAC address*
Unique MAC network address of the camera
 - *Current IP address/Current IP subnetmask*
Current IP configuration
 - *Persistent IP address/Persistent IP subnetmask*
IP configuration stored in the camera's memory
 - *IP range begin/IP range end*
IP range assigned by the computer. If the IP address is automatically assigned, the camera accesses this IP range and attempts to find an available IP address within this range.
 - *Paired host MAC/Paired host IP*
Network data of the paired computer
 - *Link*
Bandwidth of the camera's network connection
- *Local network adapter* **)
 - *General* **)
 - *Designation*
Name of the network adapter
 - *DHCP disabled/enabled*
 - *Number of known devices*
Number of devices connected to the computer
 - *Number of paired devices*

Number of cameras that have been opened by this computer

- *Network* **)
 - *MAC address*
Unique network address of the computer
 - *IP address/IP subnetmask*
Network configuration of the computer
 - *IP range begin/IP range end*
Address range stored on the computer for automatic assignment of the camera IP address (see [Automatic ETH Configuration](#))
 - *IP range valid*
Checks that the IP range stored on the computer is valid. The addresses of the IP range are valid if they are on the same subnet as the computer.
- *Settings*
 - *Packet filter* **)
Determines how incoming uEye data traffic is filtered by the network card. Block UEGET is preset and cannot be changed.
- *Local driver*
 - *Min. compatible starter FW* **)
Minimum required version of the starter firmware
 - *Max. compatible starter FW* **)
Last supported version of the starter firmware

*) This information is only displayed for USB *uEye* cameras

**) This information is only displayed for *GigE uEye* cameras

4.3.2 uEye Demo

The *uEye Demo* application demonstrates the functionality and performance of the *uEye* cameras. The application is part of the free *uEye* software package that is available for download from our website.

In *uEye Demo*, you can access all important camera settings and functions of the *uEye* programming library. Apart from controlling and configuring the camera, you can record images as AVI files and save them as BMP or JPEG files.



uEye Demo is currently only available for Windows operating systems. A version with reduced functionality is available for Linux.



Please note that *uEye Demo* does not guarantee completeness and operational reliability in all modes and all computing environments. *uEye Demo* is supplied with source code and is intended solely for demonstrating the *uEye* software library and camera functionality.

uEye Demo can be accessed as follows:

- Start → All Programs → IDS → *uEye* → *uEye Demo*
- Program icon on the desktop or Quick Launch toolbar

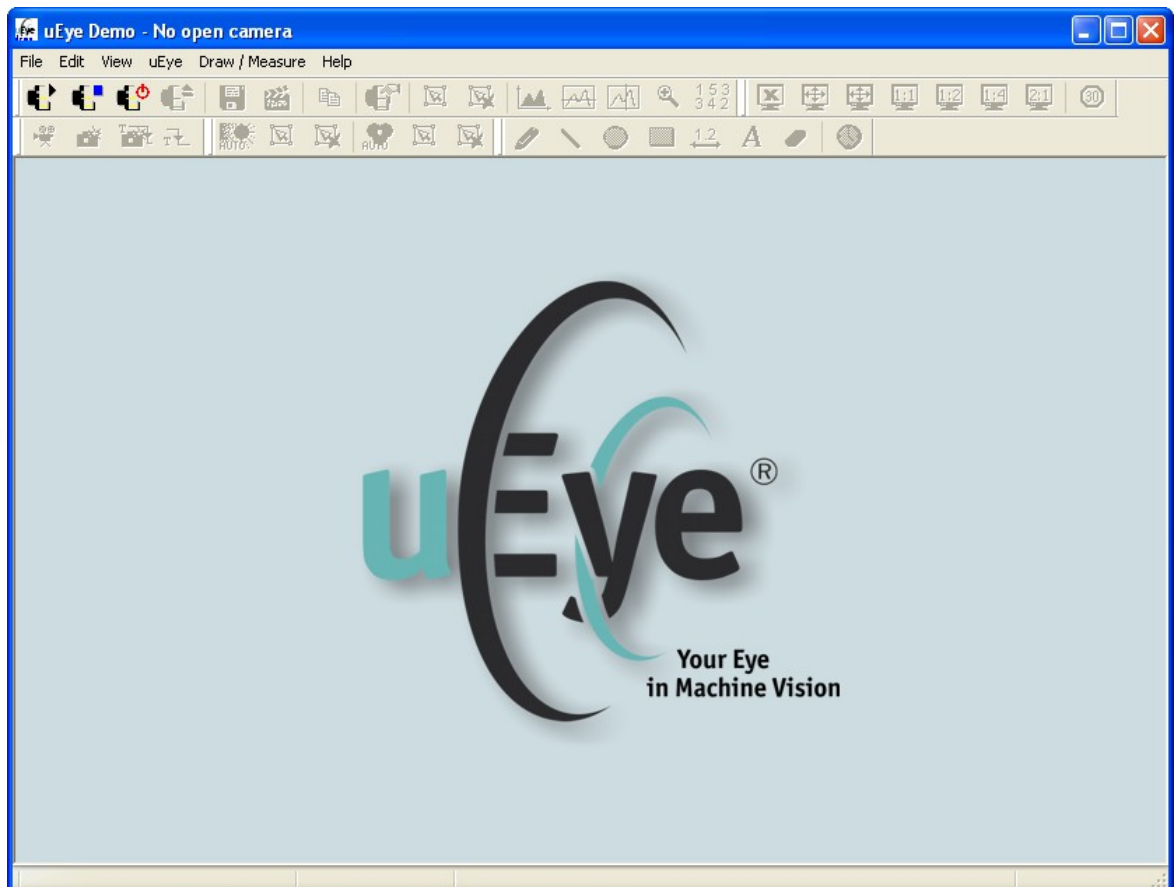


Figure 31: uEye Demo application

4.3.2.1 Camera Selection

Select *uEye menu* → *Open* or click the corresponding icon on the *General toolbar* to select (open) a connected camera. If only one camera is available, this camera is selected automatically. If more than one camera is connected, the Select Camera dialog box is displayed.

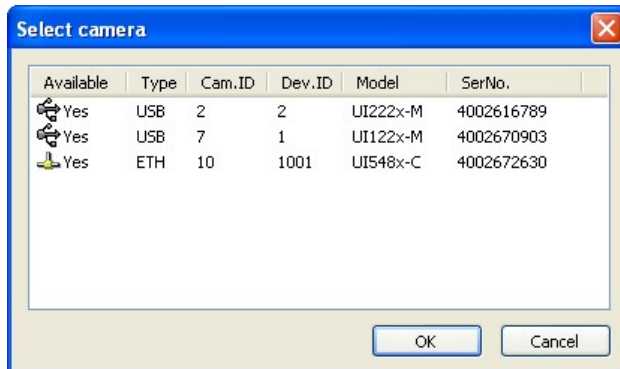


Figure 32: Select Camera dialog box

You can use multiple cameras simultaneously by opening multiple instances of *uEye Demo*. *GigE uEye* cameras that have already been opened or that have not been correctly configured are marked *No* in the *Available* column.

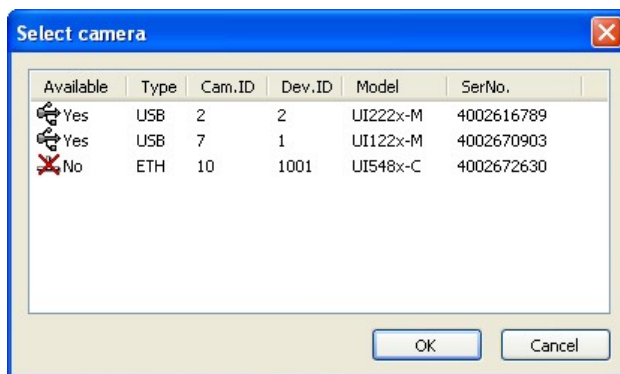
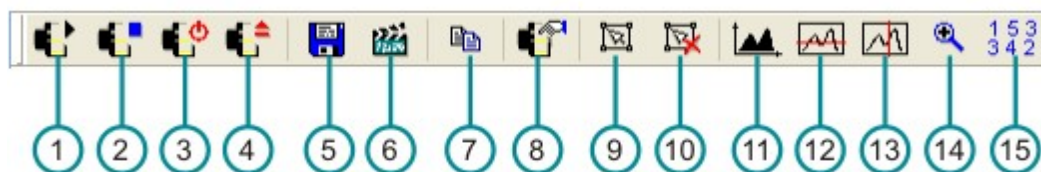


Figure 33: Select camera (cameras in use)

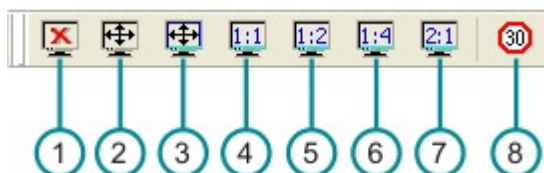
4.3.2.2 Toolbars

uEye



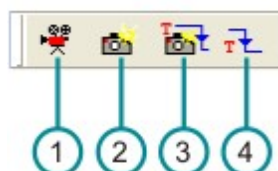
1	Open camera and start in live mode
2	Open camera
3	Close camera
4	Camera changes to standby mode
5	Save image as bitmap
6	Open the dialog box for AVI Recording
7	Copy image to the Clipboard (only in DIB mode)
8	Test the range of camera functions
9	Select AOI (Area Of Interest)
10	Delete selected AOI
11	Open/close Histogram window
12	Open/close Horizontal Line View window Displays the color values of a pixel row
13	Open/close Vertical Line View window Displays the color values of a pixel column
14	Open/close Zoom window
15	Open/close Pixel Peek window

Display



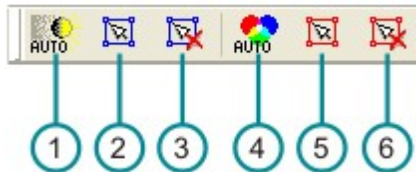
1	Deactivate display
2	Full screen window in overlay or back buffer mode
3	Scale display to window size
4	Display at original size
5	Scale display down to half size
6	Scale display down to quarter size
7	Scale display up to double size
8	Limit max. display frame rate to 30 fps. The image capture frame rate remains unchanged.

Capture



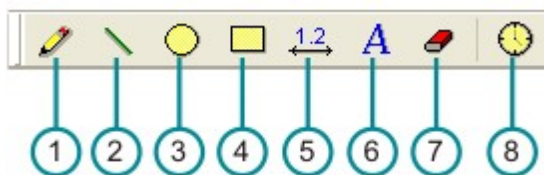
1	Start/stop live video (freerun mode)
2	Snapshot in freerun mode
3	Snapshot in trigger mode
4	Start/stop continuous triggered capture

Auto Features



1	Automatic brightness control (AES/AGC) on/off
2	Set reference area for automatic brightness control
3	Delete reference area for automatic brightness control
4	Automatic white balance (AWB) on/off
5	Set reference area for automatic white balance
6	Delete reference area for automatic white balance

Draw



1	Draw freehand overlay in image
2	Draw overlay line in image
3	Draw overlay circle in image
4	Draw overlay rectangle in image
5	Measure distance
6	Write overlay text in image
7	Clear all drawn elements
8	Timestamp on/off

Status Bar

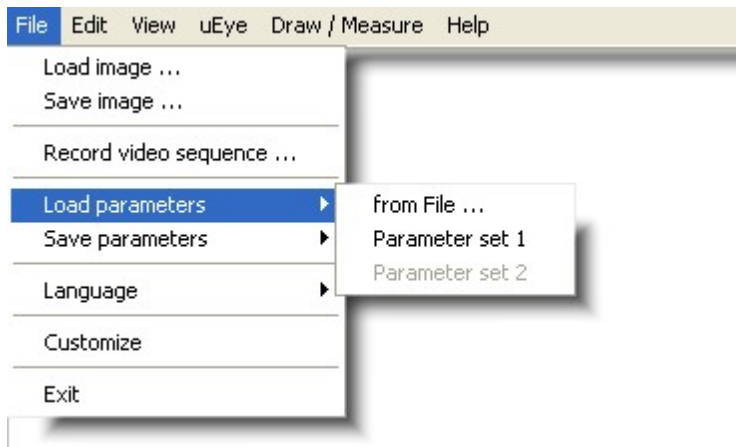


Figure 34: uEye Demo – Status bar

1	Current cursor position in the Zoom/Pixel Peek window and color values at the cursor position	
2	Defined color mode and image resolution	
3	Counters	
	Frames:	Transferred images
	Display:	Displayed images
	Missed:	Hardware trigger events missed. This counter increments each time a hardware trigger is received in trigger mode, but the camera is not ready for image capture
	Failed:	Transmission errors
	Recon.:	This counter increments each time the open camera is removed and reconnected during operation.
4	Status of the current image data transfer (OK/Error)	
5	Current frame rate (fps) of the camera	

4.3.2.3 Menus

File



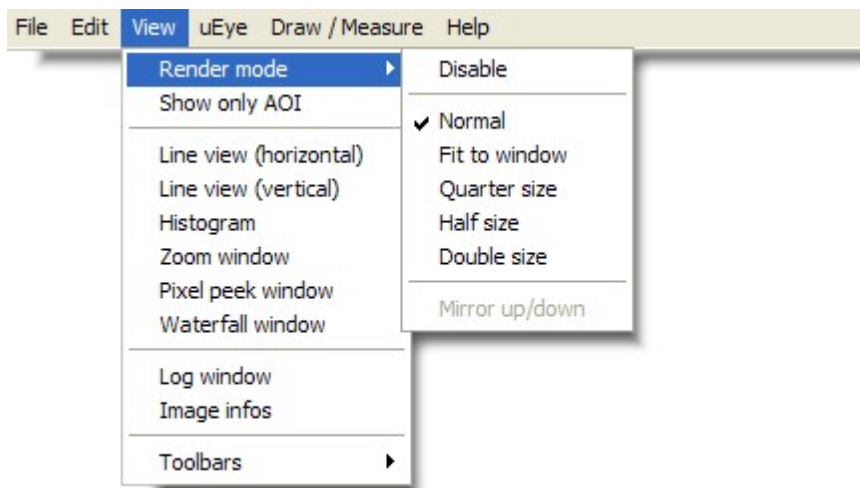
Load image ...	Load bitmap
Save image ...	Save image as bitmap
Record video sequence ...	Opens the Record Dialog box
Load parameters	Load parameters from an ini file or from one of the camera's parameter sets
Save parameters	Save parameters to an .ini file or to one of the camera's internal parameter sets
Language	Select a language for the program. When you change the language, you need to restart <i>uEye Demo</i> to apply the new setting.
Customize	Opens a dialog box where you can make various settings for the startup behavior of <i>uEye Demo</i>
Exit	Exit the demo program

Edit



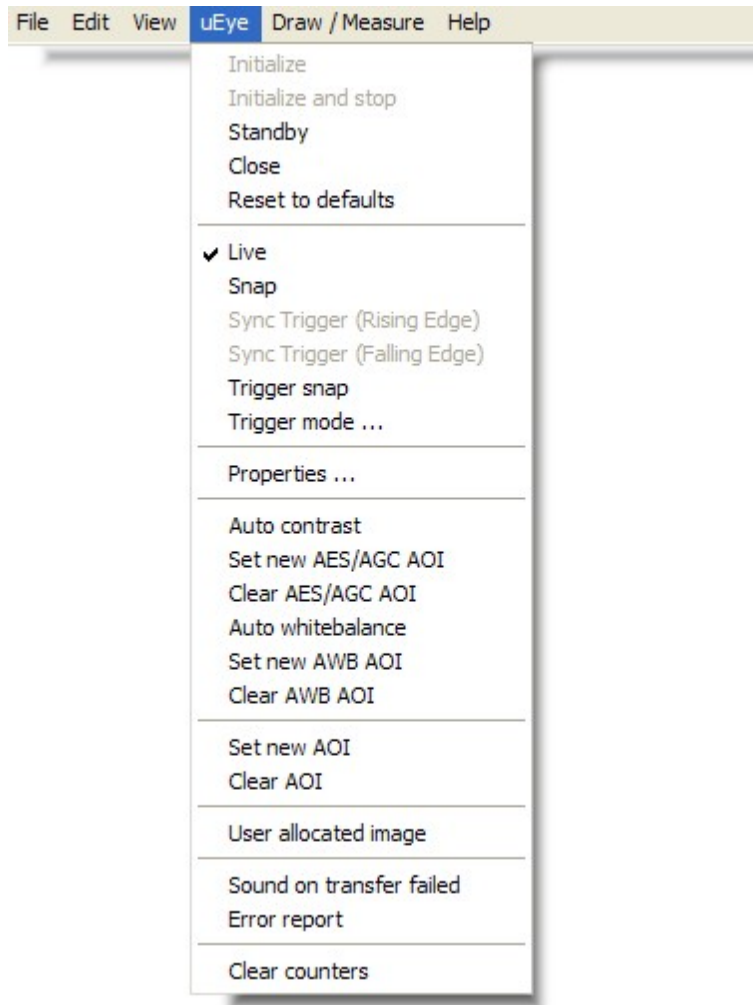
Copy Ctrl+C	Copy the displayed image content to the Clipboard. Overlay data created using the Draw/Measure function is also copied automatically.
-------------	---

View



Render mode	Image display
Disable	Deactivate display
Normal	Display at original size
Fit to window	Scale display to window size
Quarter size	Scale display down to quarter size
Half size	Scale display down to half size
Double size	Scale display up to double size
Mirror up/down	Mirror display on horizontal axis
Show only AOI	AOI is displayed without black border
Line view (horizontal)	Opens the <u>L</u> ine view window (row view)
Line view (vertical)	Opens the <u>L</u> ine view window (column view)
Histogram	Opens the <u>H</u> istogram window
Zoom window	Opens the <u>Z</u> oom window
Pixel peek window	Opens the <u>P</u> ixel peek window
Waterfall window	Opens the <u>W</u> aterfall window
Log window	Opens the <u>L</u> og window
Image infos	Opens the <u>I</u> mage infos window
Toolbars	Show/hide toolbars uEye, View, Capture, Auto Features and Draw/Measure

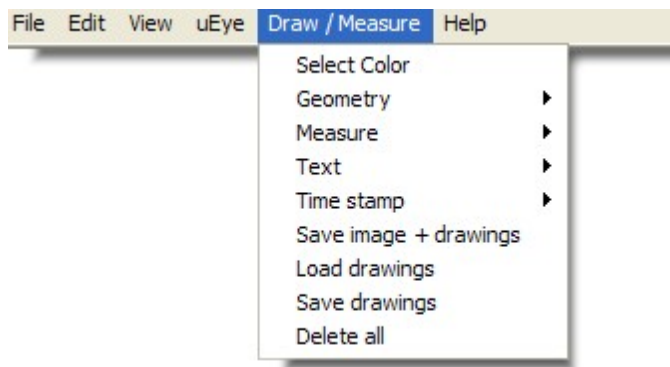
uEye



Initialize	Open camera and show live image
Initialize and stop	Open camera
Standby	The camera changes to standby mode
Close	Close camera
Reset to defaults	Resets all values set in the demo program to the defaults
Live	Live video on/off
Snap	Snapshot from live video
Sync Trigger (Rising Edge)	Triggered snapshot (hardware trigger, rising edge)
Sync Trigger (Falling Edge)	Triggered snapshot (hardware trigger, falling edge)
Trigger snap	Triggered snapshot (software trigger)
Trigger mode ...	Trigger mode on/off; continuous triggered capture
Properties ...	Opens the camera properties dialog (see Properties)
Auto contrast	Activate <u>automatic brightness control</u>
Set new AES/AGC AOI	Define active area for <u>automatic brightness control</u>
Clear AES/AGC AOI	Clear active area defined for <u>automatic brightness control</u>
Auto whitebalance	Activate <u>automatic white balance</u>
Set new AWB AOI	Define active area for <u>automatic white balance</u>
Clear AWB AOI	Clear active area defined for <u>automatic white balance</u>
Set new AOI	After calling <i>Set new AOI</i> , you can select the area to be used as AOI with the mouse
Clear AOI	Resets the area set with <i>Set new AOI</i>

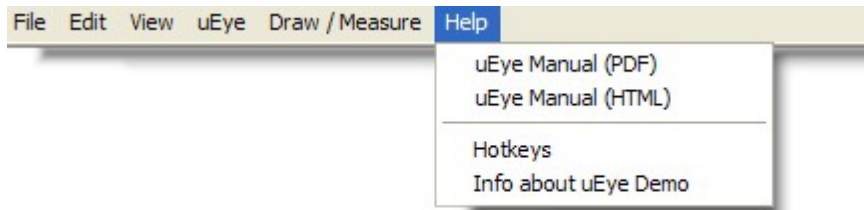
User allocated image	The memory is allocated not via the SDK function <i>is_SetAllocateImageMem()</i> , but by the application
Sound on transfer failed	Output an acoustic signal if a transmission error occurs
Error report	Errors are displayed in a dialog box
Clear counters	Reset the counters (see <u>uEye Demo Status Bar</u>)

Draw/Measure



Select Color	Select color for drawings and text
Geometry	
Select line width	Set line width
Circle	
Show circles	Show/hide circles
New circle	Draw new circles
Delete circles	Delete circles
Freehand	
Show freehand	Show/hide freehand drawings
New freehand	Draw new freehand
Delete freehand	Delete freehand drawing
Line	
Show lines	Show/hide lines
New lines	Draw new lines
Delete lines	Delete lines
Rectangle	
Show rectangles	Show/hide rectangles
New rectangles	Draw new rectangles
Delete rectangles	Delete rectangles
Measure	
Set measuring unit	Set scale
Show measures	Show/hide dimension lines
New measure	New dimension line
Delete measures	Delete all dimension lines
Text	
Select font	Select font
Show texts	Show/hide texts
New text	Write new text
Delete text	Delete text
Time stamp	
Set timestamp position	Timestamp position (top left, top right, bottom left, bottom right)
Show timestamp	Show/hide timestamp
Save image + drawings	Opens the Save As dialog box. The image is stored with all drawings, texts, measures and dimension lines
Load drawings	Loads saved drawings from a *.bin file.
Save drawings	Saves current drawings to *.bin file.
Delete all	Delete all drawings, texts, measures and dimension lines

Help



uEye User Manual	Opens this manual
uEye Programming Manual	Opens the uEye Programming Manual
Hotkeys	Öffnet ein Fenster mit Tastaturkürzeln für die Bedienung von <i>uEye Demo</i> an.
Info über uEye Demo	Öffnet ein Fenster mit Informationen zum Programm und den installieren <i>uEye</i> -Treiberversionen.

4.3.2.4 Dialog Boxes

Record Dialog

Select **File menu** → **Record video sequence ...** to open the *uEye* Record Dialog box. This dialog box provides all the functions you need to create a video file (.avi) from a sequence of images captured with the *uEye*. To reduce the file size, the single frames are stored in the AVI container using an adjustable JPEG compression. It is possible to extract single frames from the AVI file.

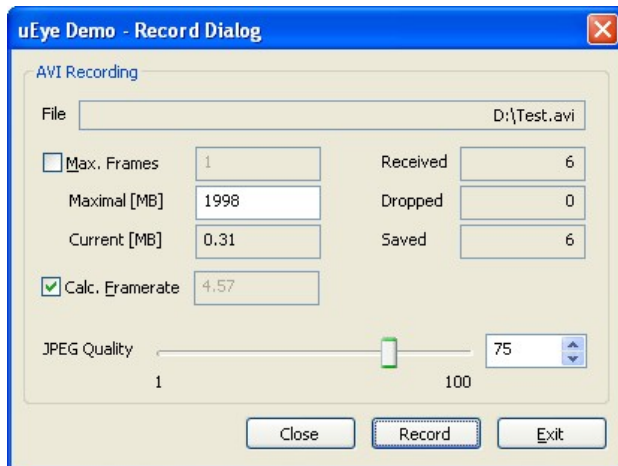


Figure 35: File menu – Record video sequence

AVI capture workflow

Once the AVI file has been created, images transferred from the *uEye* are placed in a buffer. Then, the images are compressed and added to the AVI file which is stored on the hard disk. These operations are not performed in the same thread as the capturing process. If you capture more images while a compression or write operation is in progress, the new images will be discarded.

Supported color formats

The supported input color formats are RGB32, RGB24, Y8 and raw Bayer. The output file will always be in RGB24 format, regardless of the input data format. Once the AVI file has been created, the following parameters of the input image can no longer be changed:

- Color format
- AOI, binning and subsampling
- Show only AOI



AVI recording is only possible in the *Device Independent Bitmap (DIB) display mode*.

Record dialog box

- *File*
Name of the AVI file opened for recording.
- *Max. Frames*
If you select this check box, you can set the number of frames after which recording should stop.
- *Maximal [MB]*
Sets the maximum size for the AVI file. Recording stops when the AVI file reaches the size limit. You can edit the entry in this box (default: 1998).
- *Current [MB]*
Indicates the current size of the AVI file.
- *Received*

Indicates the number of images transferred by the camera.

- *Dropped*

Indicates the number of images discarded during compression. A image is dropped if it arrives while compression is in progress.

- *Saved*

Indicates the number of images saved to the AVI file.

- *Calc. Framerate*

If you select this check box, the frame rate of the AVI file is determined automatically during recording. It can also be set manually. The frame rate value is stored in the AVI file and evaluated by the movie player. The usual value is 25 or 30 frames per second.

The recording speed of the video depends on the selected color format, the image size and the compression level of the AVI file as well as the PC performance.

- *JPEG Quality*

This slider sets the JPEG compression level (1 = lowest quality/highest compression, 100 = highest quality/lowest compression, default = 75).

- *Create.../Close*

Click this button to create a new AVI file for recording, and to close the file again when you are done. If you select an existing AVI file, the file contents will be overwritten during recording.

- *Record/Stop*

Starts/stops AVI recording.

- *Exit*

Closes the Record dialog box. A recording in progress is stopped.

Horizontal/Vertical Line View

Select *View menu* → *Line view (horizontal / vertical)* to open the Line View windows, which show the color values of a pixel row or pixel column. The line view always displays 256 values per channel. For color modes with a bit depth of more than 8 bits, the function evaluates the 8 most significant bits (MSBs).

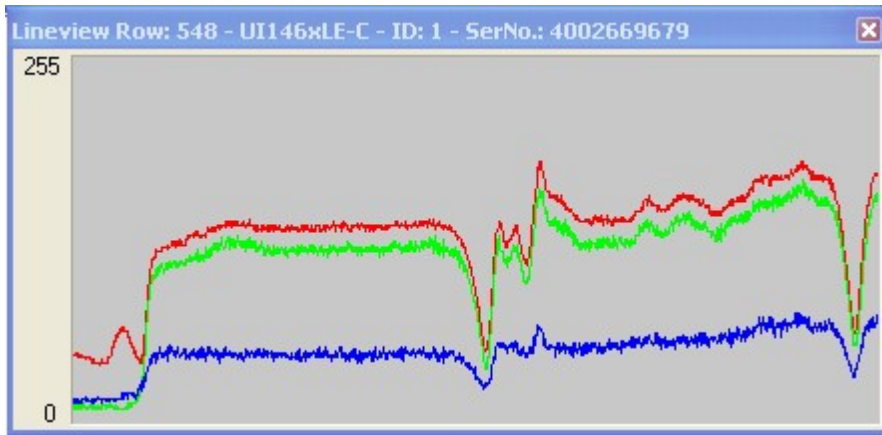


Figure 36: View menu - Line view

Histogram

Select **View menu** → **Histogram** to open the Histogram window. This window graphically displays the frequency distribution of the color values in the image captured by the camera. The histogram always displays 256 values per channel. For color modes with a bit depth of more than 8 bits, the function evaluates the 8 most significant bits (MSBs).

The following options are available:

- **Channel**
With the Red, Green, and Blue check boxes, you can enable or disable the display for each color channel. Avg. displays the average of each color value.
For monochrome images, the average grayscale value is displayed.
- **Outlined**
The Outlined check box enables you to toggle the color value display between an area diagram and a line diagram.
- **Logarithmic**
If you select this check box, the values are displayed with logarithmic scaling.
- **Show Bayer RGB**
This function is only available when a color camera is running in Direct raw Bayer mode. If you select this check box, the histogram represents the individual Bayer color components as red, green and blue channels.

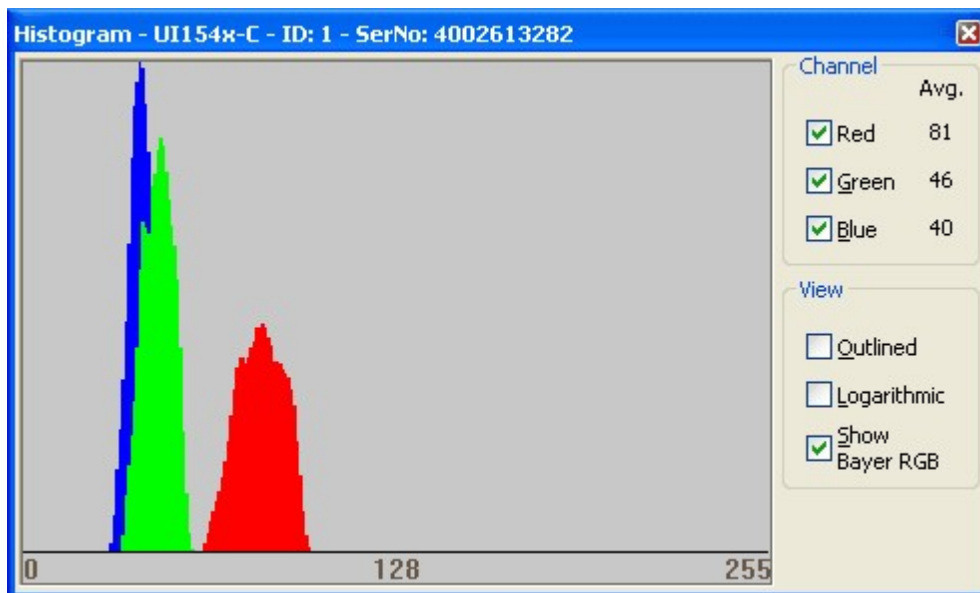


Figure 37: View menu – Histogram window

Zoom Window

Select **View menu** → **Zoom window** to open the Zoomwindow. This window shows an enlarged view of the image area at the selected cursor position. Using the slider at the top of the window, you can set the zoom factor in the range between 0.25 and 20.00. The size of the image area depends on the selected size of the Zoom window.

If you enable the **Pixel Peek** check box at the top of the zoom window, the color values at the cursor position are displayed (see [Pixel Peek Window](#)).

To set the cursor position you want to display in the window, place the cursor at that position in the image, hold the **b** key and right-click. Alternatively, you can set the image position using the context menu.

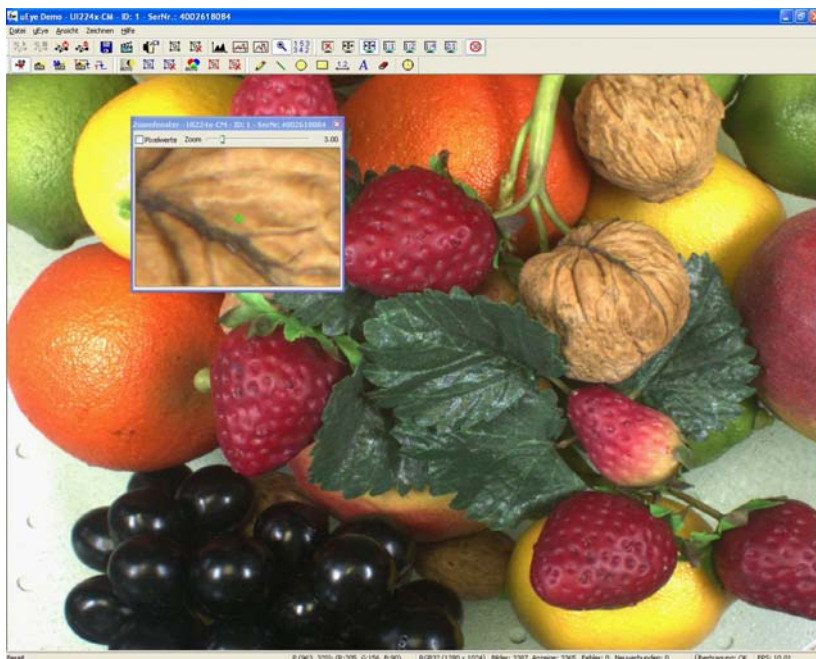


Figure 38: View menu – Zoom window

Pixel Peek Window

Select **View menu** → Pixel peek window to open the Pixel peek window. This window displays the color values at the selected cursor position and those of the neighboring pixels. The color values at the cursor position are surrounded by a yellow rectangle. For monochrome images, the grayscale value is displayed. The window always displays 256 values per channel. For color modes with a bit depth of more than 8 bits, the function evaluates the 8 most significant bits (MSBs).

If you disable the Pixel Peek check box at the top of the window, the Zoom window is displayed (see [Zoom Window](#)).

To set the cursor position you want to display in the window, place the cursor at that position in the image, hold the Ctrl key and right-click. Alternatively, you can set the image position using the context menu.

	815			816			817		
196	177	183	72	167	165	61	152	155	55
197	171	182	71	162	171	58	149	161	52
198	165	180	68	157	169	60	146	163	54
199	159	179	63	152	166	58	143	158	55
200	155	177	54	148	164	50	140	156	49
201	154	174	50	145	163	47	137	155	47
202	154	176	53	146	164	52	138	154	50
203	155	173	56	150	168	52	144	158	49
204	154	175	59	149	171	53	142	158	48
205	151	168	61	142	169	52	132	155	48
206	148	155	59	138	157	52	127	151	47

Figure 39: View menu – Pixel Peek window

Waterfall Window

Select *View menu* → *Waterfall window* to display the Waterfall window. This window shows how a selected image line changes over time. For this purpose, the line at the selected cursor position is copied to the new window. With each new frame, all lines in the Waterfall window are moved one pixel down, and the new line is added at the top. This results in an image that flows from top to bottom and is useful for observing short-term image changes.

To set the cursor position you want to monitor in the Waterfall window, place the cursor at that position in the image, hold the *Ctrl* key and right-click. Alternatively, you can set the image position using the context menu.

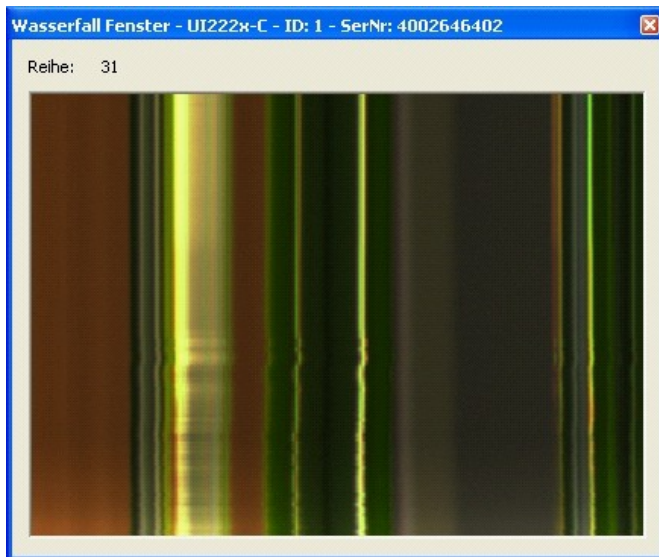


Figure 40: View menu – Waterfall window

Log Window

Select *View menu* → *Log window* to display the logged data. The *uEye* logs events and messages in this window.

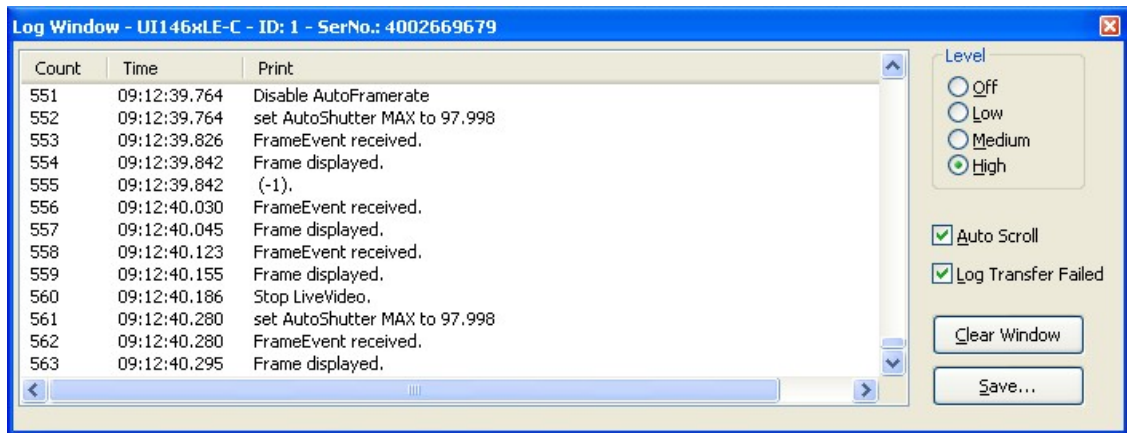


Figure 41: View menu – Log window

The following options are available:

- *Level*
With these radio buttons, you choose the logging level (*Off*, *Low*, *Medium*, *High*). The last level used is saved when you close the window. The next time you open the log window, logging is performed at that level.
- *Auto Scroll*
When you select the *Auto Scroll* check box, the display automatically scrolls up when new entries arrive so that the new entries can be read.
- *Log Transfer Failed*
Select the *Log Transfer Failed* check box if you want to log transfer errors.
- *Clear Window*
The *Clear Window* button deletes the current messages.
- *Save*
The *Save* button opens the Windows *Save as* dialog box, allowing you to save the messages displayed in the log window in ASCII format (.txt file).

Image Infos

Select *View menu* → *Image infos* to display a dialog box containing information on the image capture.

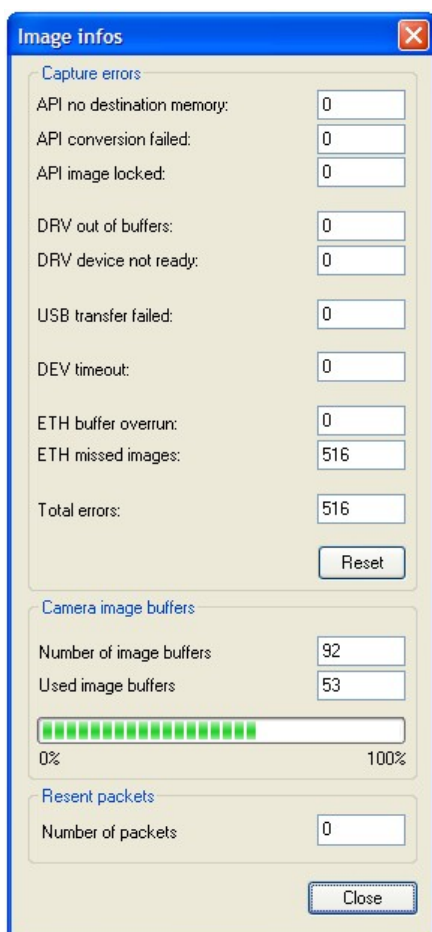


Figure 42: View menu - Image information

Capture errors

This group box provides detail information on errors that occurred during an image capture process:

Error	Description	#
API no destination memory	There is no destination memory for copying the finished image.	1
API conversion failed	The current image could not be processed correctly.	2
API image locked	The destination buffers are locked and could not be written to.	3
DRV out of buffers	No free internal image memory is available to the driver. The image was discarded.	4
DRV device not ready	The camera is no longer available. It is not possible to access images that have already been transferred.	5
USB transfer failed	The image was not transferred over the USB bus.	6

DEV timeout	The maximum allowable time for image capturing in the camera was exceeded.	7
ETH buffer overrun	The sensor transfers more data than the internal camera memory of the <i>GigE uEye</i> can accommodate.	8
ETH missed images	The <i>GigE uEye</i> camera could neither process nor output an image captured by the sensor.	9

#	Possible cause	Remedy
1	Not enough destination memory allocated or all destination buffers locked by the application	<ul style="list-style-type: none"> Reduce the frame rate so that there is more time to process the filled destination memory
2	Internal error during internal processing of the image	-
3	All destination buffers locked by the application	<ul style="list-style-type: none"> Reduce the frame rate so that there is more time to process the filled destination memory
4	The computer takes too long to process the images in the <i>uEye API</i> (e.g. colour conversion)	<ul style="list-style-type: none"> Reduce the frame rate so that there is more time to process the filled image memory of the driver Disable resource-intensive API image pre-processing functions (e.g. edge enhancement, colour correction, choose smaller filter mask for software colour conversion)
5	The camera has been disconnected or closed	-
6	Not enough free bandwidth on the USB bus for transferring the image	<ul style="list-style-type: none"> Reduce the pixel clock frequency Operate fewer cameras simultaneously on a USB bus Check the quality of the USB cabling and components
7	The selected timeout value is too low for image capture	<ul style="list-style-type: none"> Reduce the exposure time Increase the timeout
8	The selected data rate of the sensor is too high	<ul style="list-style-type: none"> Reduce the pixel clock frequency Reduce the frame rate Reduce the image size
9	The camera's frame rate is too high or the bandwidth on the network is insufficient to transfer the image	<ul style="list-style-type: none"> Reduce the frame rate Increase the value for the receive descriptors in the network card settings

Camera image buffers

If you are using a *GigE uEye* camera, this group box displays the number and utilization of the camera's image buffers. If the image buffers are utilized to full capacity, frames might be lost.

Resent packets

With *GigE uEye* cameras, this field displays the number of network packets that were retransmitted due to data loss. This value indicates the quality of the network connection.

4.3.2.5 Properties

When you select **uEye** → **Properties** from the main menu, a dialog box opens where you can set the **uEye** camera parameters. Changes made to camera and image settings here will take effect immediately.

When you close a camera in **uEye Demo**, the current settings are written to the Windows Registry. They will be loaded the next time you open a camera of the same type. To save the settings to the camera or to an ini file, select **File** → **Save parameters** from the main menu. To load settings, select the **Load parameters** option.

Camera

This tab provides parameters for settings the pixel clock frequency, frame rate and exposure time for your **uEye** camera (see also [Pixel Clock](#), [Frame Rate](#) and [Exposure Time](#)).

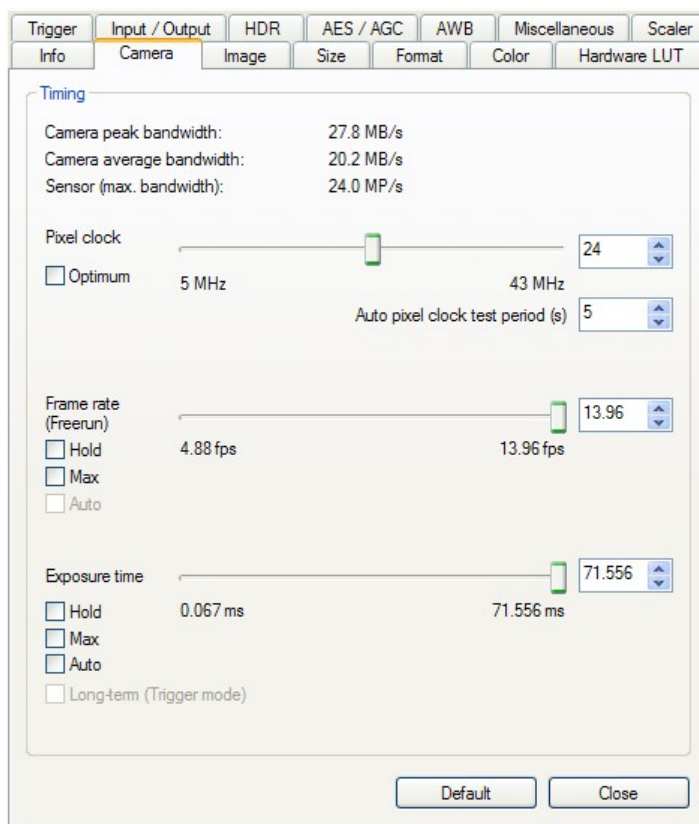


Figure 43: uEye properties - Camera

Timing

- *Camera peak bandwidth*
Maximum required bandwidth in MB/sec (peak load). This value is displayed in red if the available Gigabit Ethernet bandwidth is exceeded.
- *Camera average bandwidth*
Required average bandwidth in MB/sec. This value is displayed in red if the available Gigabit Ethernet bandwidth is exceeded.
The average bandwidth is calculated from the following data: Image size, image format, frame rate, and interface-related protocol overhead.
- *Sensor (max. bandwidth)*
Maximum data volume in Mpixels/sec created by the sensor.



The upper bandwidth limit of a Gigabit Ethernet camera depends on the chipset of the network card and on the network structure. If transfer errors occur, you need to reduce the frame rate.
With USB cameras, the upper limit depends on the USB chipset on the mainboard/USB card and on the number of USB devices connected. If transfer errors occur, reduce the pixel clock frequency.

- *Pixel*
Sets the clock rate at which the image data is read from the sensor. Changes to this parameter affect the frame rate and the exposure time.
Many CMOS sensors allow higher pixel clock frequencies in binning/subsampling mode.
 - *Optimum*
When you select this check box, the highest possible pixel clock is determined and set automatically. The optimum pixel clock is the clock rate at which no transfer errors occur during the time (in seconds) set in the Auto pixel clock test period box. The longer you set the test period, the more reliable the determined pixel clock becomes. The total time it takes to automatically set the pixel clock is a bit longer than the test period setting.
- *Frame rate (Freerun)*
Sets the frame rate in freerun mode. The available frame rate range depends on the pixel clock setting.
 - *Hold*
When you select this check box, the frame rate will remain constant if the pixel clock changes. If the frame rate cannot be maintained, it is set to the nearest possible value.
 - *Max*
The camera is operated at the maximum frame rate that is possible at the current pixel clock setting.
 - *Auto*
Select this check box to activate the Auto Frame Rate function. This function is only available when Auto Exposure Shutter is enabled.
- *Exposure time*
Sets the exposure time. The available exposure time range depends on the pixel clock setting and the frame rate. A low frame rate setting allows long exposure times. A high frame rate setting reduces the maximum possible exposure time.
 - *Hold*
When you select this check box, the exposure time will remain constant if the frame rate changes. If the exposure time cannot be maintained, it is set to the nearest possible value.
 - *Max*
The camera is operated at the maximum exposure time that is possible at the current frame rate.
 - *Auto*
Select this check box to activate the Auto Exposure Shutter function. If the Auto check box is selected, the exposure time and pixel clock can no longer be adjusted manually. Selecting

the Hold or Max check box deselects the Auto check box.

- *Long-term*

If you select this check box, you can set an exposure time of up to 10 minutes on many uEye CCD cameras. This function is only available in trigger mode.

Default

Click this button to reset all parameters to the model-specific defaults.

Image

On this tab you can set the sensor gain parameters for your *uEye* camera (see also [Gain and Offset](#)).

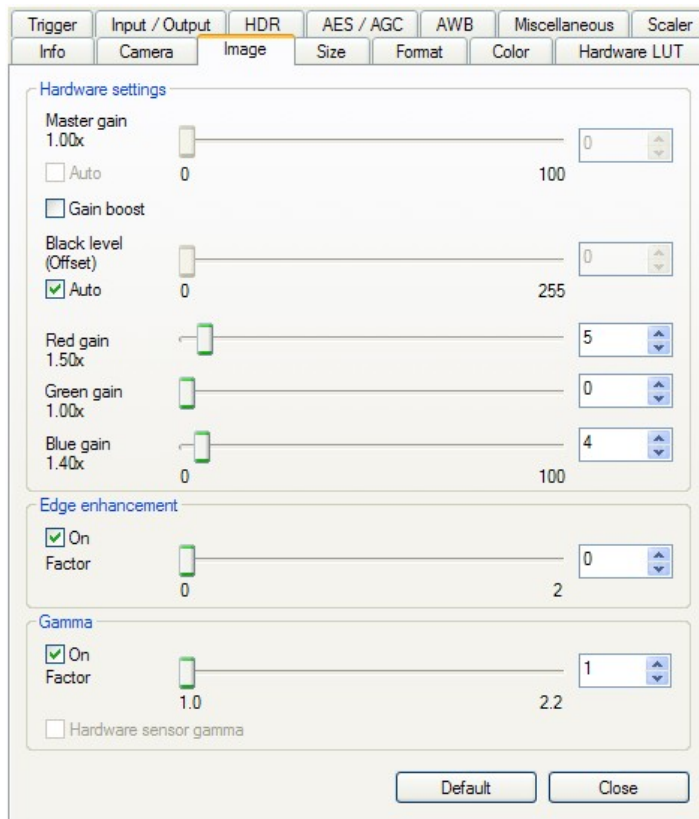


Figure 44: uEye properties - Image

Hardware settings

The following functions control the analog image signal gain and the black level. The analog adjustments are made directly in the sensor, which achieves better results than image adjustments via software.

- **Master gain** [0 ... 100]

Gain for overall image brightness. Some camera models have no master gain. Master gain = 100 means maximum gain; the actual factor is displayed. A gain factor of 1x disables master gain. The maximum possible gain factor depends on the model you are using.

 - *Auto*

Select this check box to activate the [automatic gain control](#) function. Manually changing the master gain setting disables the Auto function.
 - *Gain boost*

Additional analog camera hardware gain. The gain factor ranges between 1.5x and 2x, depending on the camera model.
- **Black level (offset)** [0 ... 255]

Offset for the black level of the sensor signal. The sensor adjusts the black level of the pixels automatically by default. If the environment is very bright, it can be necessary to adjust the black level manually. High gain may offset the black level. Only an additive offset is possible (increase of the black level).

 - *Auto*

The black level is automatically corrected by the sensor (recommended).

- *Red gain* [0 ... 100]
Amplifies the red color values
- *Green gain* [0 ... 100]
Amplifies the green color values
- *Blue gain* [0 ... 100]
Amplifies the blue color values

The RGB gain sliders are only enabled for color cameras.



With cameras featuring both master gain and RGB gain, the two gain factors are multiplied. Very high gain values can be achieved in this way.

If you want to use the RGB sliders for color adjustment, we recommend setting green gain to 0 and using only red and blue gain.

Edge enhancement

This function activates a software filter that emphasises the edges in the image. Enabling the *Edge enhancement* function increases the CPU load during image capture.

Gamma

This function activates the gamma function and sets the factor for the gamma curve. The gamma function emphasises dark image areas according to a non-linear curve. When you are using a *GigE uEye HE* camera and have activated hardware color calculation, the gamma curve is calculated in the camera. In all other cases, the gamma curve is calculated in the PC.

- *Hardware sensor gamma*
Select this check box to enable gamma correction by the hardware, using a fixed factor. This function is currently only available for the UI-122X-X/UI-522X-X.

Default

Click this button to reset all parameters to the model-specific defaults.

Size

On this tab, you can set the image size parameters for your *uEye* camera (see also [Reading out Partial Images](#)).

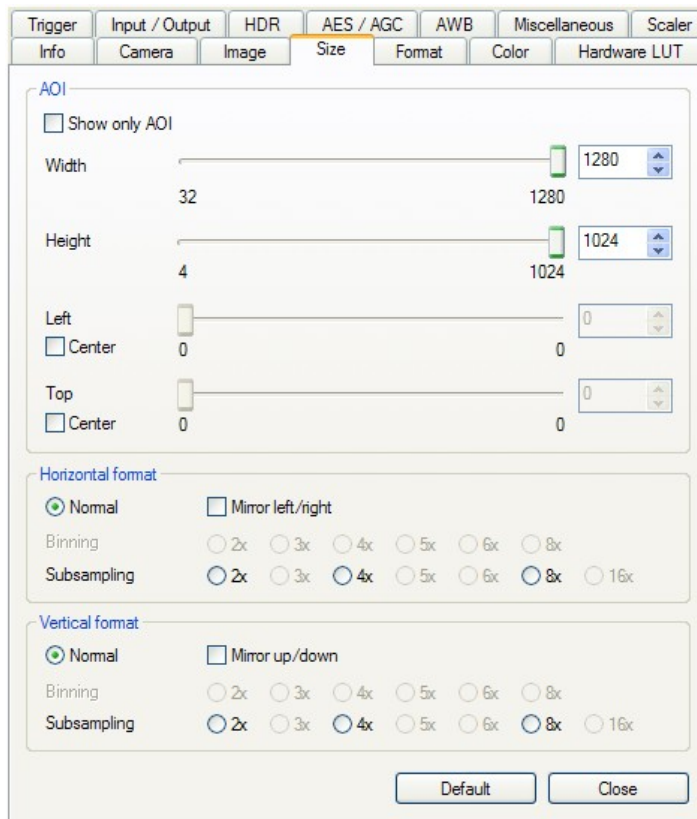


Figure 45: uEye properties - Size

AOI

These parameters allow selecting the size and position of the area of interest.

- *Show only AOI*

The AOI is displayed without a black border. Internally, the image is managed at the AOI resolution and not the full sensor resolution. This function saves memory and computing time when rendering the image.

- *Width*

Sets the AOI width.

- *Height*

Sets the AOI height.

- *Left*

Sets the left-hand position of the AOI.

- *Center*

Select this check box to center the AOI horizontally.

- *Top*

Sets the top position of the AOI.

- *Center*

Select this check box to center the AOI vertically.

Horizontal format / Vertical format

With these check boxes and radio buttons, you can select the binning and subsampling settings for the image.

- *Normal*
Disables binning and subsampling.
- *Mirror left/right / Mirror up/down*
Select this check box to flip the image horizontally/vertically. On CMOS camera models, vertical mirroring is performed directly in the sensor.
- *2x/3x/4x/5x/6x/8x Binning*
These radio buttons allow setting the binning factor. The image resolution is then reduced by the selected factor. You can use binning to increase the frame rate.
- *2x/3x/4x/5x/6x/8x/16x Subsampling*
These radio buttons allow setting the subsampling factor. The image resolution is then reduced by the selected factor. You can use subsampling to increase the frame rate.



The Specifications: Sensors chapter shows you which binning and subsampling factors the individual camera models support.



Some color cameras perform only mono binning/subsampling due to the sensors they use. If mono binning or subsampling is used in a color camera, the color information will be lost.

Some monochrome cameras perform only color binning/subsampling due to the sensors they use. If color binning or subsampling is used in a monochrome camera, image artefacts might become visible.

Default

Click this button to reset all parameters to the model-specific defaults.

Format

On this tab you can set parameters for the color format and display mode of your *uEye* camera (see also [Color Filter \(Bayer Filter\)](#)).

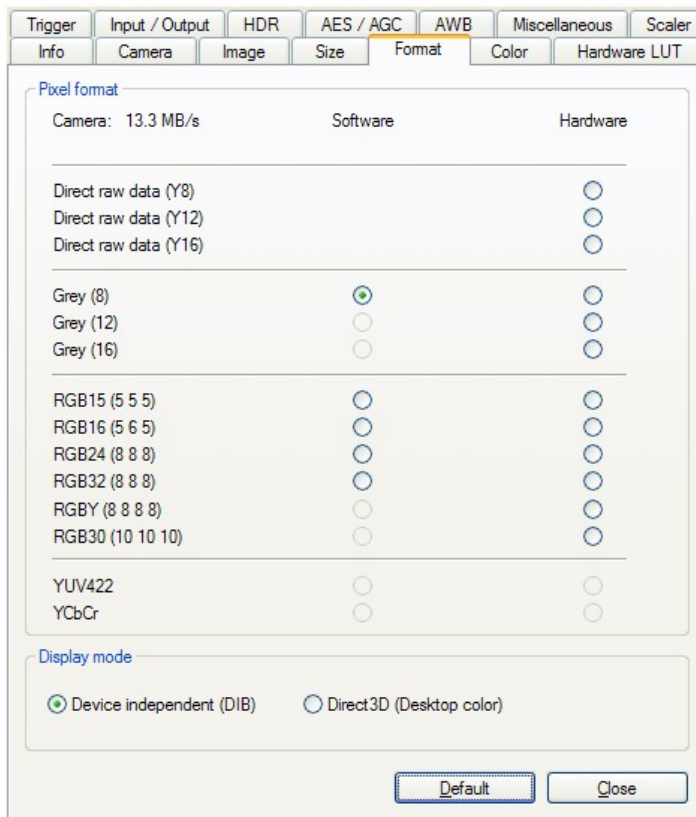


Figure 46: uEye properties - Format

Pixel format

With these settings you define the target format to which you want to convert the sensor's raw data (raw Bayer). The [Color Formats](#) chapter provides a detailed list of all *uEye* color formats and their representation in the memory.

▪ Debayering quality

With this setting you select the [conversion algorithm](#).

○ Software Normal

The conversion is performed by software in the PC. The standard filter mask is used for the conversion.

○ Software High

The conversion is performed by software in the PC. A large filter mask is used for the conversion.

○ Hardware Normal

The conversion is performed in the camera. The standard filter mask is used for the conversion. This radio button is only available for *GigE uEye HE* cameras. When you select hardware de-Bayering, you can also perform the [LUT](#), [Gamma](#) and [Hotpixel correction](#) functions directly in the camera.

With the format radio buttons you specify the format in which the image data are written to the memory. The following formats are available:

○ Direct raw bayer (8)

Direct output of the sensor's raw data (8 bits per pixel). If you are using a color camera, the pattern of the Bayer color filter is visible. With monochrome cameras, *raw Bayer* corresponds to the grayscale format with the exception of the LUT/gamma curves.

- *Direct raw bayer (12)*
Direct output of the sensor's raw data (12 bits per pixel, starting from the least significant bit (LSB)).
- *Direct raw bayer (16)*
Direct output of the sensor's raw data (12 bits per pixel, starting from the most significant bit (MSB)).
- *Grayscale (8)*
Output of a grayscale image to which the LUT/gamma curve has been applied (8 bits per pixel).
- *Grayscale (12)*
Output of a grayscale image to which the LUT/gamma curve has been applied (12 bits per pixel, starting from least significant bit (LSB)).
- *Grayscale (16)*
Output of a grayscale image to which the LUT/gamma curve has been applied (12 bits per pixel, starting from most significant bit (MSB)).
- *RGB15 (5 5 5)*
Output of an image converted according to RGB 15 (5 bits per pixel for R, G and B)
- *RGB16 (5 6 5)*
Output of an image converted according to RGB 16 (5 bits per pixel for R and G, 6 bits per pixel for B)
- *RGB24 (8 8 8)*
Output of an image converted according to RGB 24 (8 bits per pixel for R, G and B)
- *RGB32 (8 8 8)*
Output of an image converted according to RGB 32 (8 bits per pixel for R, G and B, 8 bit padding)
- *RGBY (8 8 8 8)*
Output of an image converted according to RGB 24 (8 bits per pixel for R, G and B) and an additional gray channel (8 bits per pixel)
- *RGB30 (10 10 10)*
Output of an image converted according to RGB 30 (10 bits per pixel for R, G and B, 2 bit padding (MSB = 0))
- *YUV422*
Output of an image converted according to YUV (8 bits per pixel for U, Y, V and Y)
- *YCbCr (8 8 8 8)*
Output of an image converted according to YUV (8 bits per pixel for Cb, Y, Cr and Y)



We recommend 32-bit RGB mode for TrueColor applications. Y8 mode is usually used for monochrome applications.

The Color Formats chapter provides a detailed list of all uEye color formats and their representation in the memory.

Display mode

With these radio buttons you select the display mode for the image.

- *Device independent (DIB)*

The processor actively renders the image. This color format is supported by all graphics hardware and is recommended for applications that will be used on different PCs.

- *Direct3D (Desktop color)*

In this mode the images are written directly to an invisible area of the graphics card, mixed with optional overlay image data and displayed by the card without load on the CPU. The mode also allows scaling the images in real time.



The display mode *Direct3D* is only supported by graphics cards with *DirectX* functionality.

Default

Click this button to reset all parameters to the model-specific defaults.

Color

This tab provides color rendering settings for your *uEye* camera (see also [Color Filter](#)).

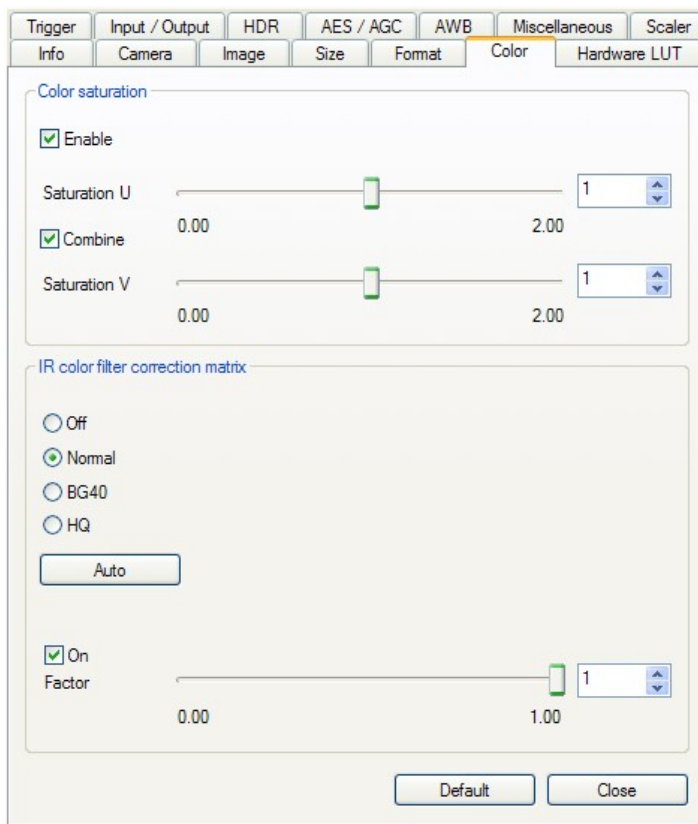


Figure 47: *uEye* properties - Color

Color saturation

This function enables and configures color saturation control.

In the YUV format, color information (i.e. the color difference signals) is provided by the U and V channels. In the U channel they result from the difference between the proportion of blue and Y (luminance), in the V channel from the difference between the proportion of red and Y. For the use in other color formats than YUV, U and V are converted using a driver matrix.

- **Combine**

Selecting this check box synchronizes the two *Saturation U* and *Saturation V* sliders.

IR color filter correction matrix

When using color cameras with IR filter glass, you need to set the appropriate color correction matrix to ensure correct color rendering. The driver detects the IR filter type and sets this value automatically (*Auto button*). You can also select the correction matrix manually.

Sensor color correction

This function corrects the color values of a pixel. The colors are rendered more accurately after the color crosstalk of the individual [Bayer pattern](#) filters has been eliminated by the color correction. The color correction factor is steplessly adjustable between 0 (no correction) and 1 (full correction).

Activating the sensor color correction may increase CPU load.

Default

Click this button to reset all parameters to the model-specific defaults.

Hardware LUT

This tab provides settings for the hardware LUT curve of the *GigE uEye HE* camera. Each look-up table (LUT) for the *uEye* contains modification values for the image brightness and contrast parameters. When a LUT is used, each brightness value in the image will be replaced by a value from the table. LUTs are typically used to enhance the image contrast or the gamma curve.

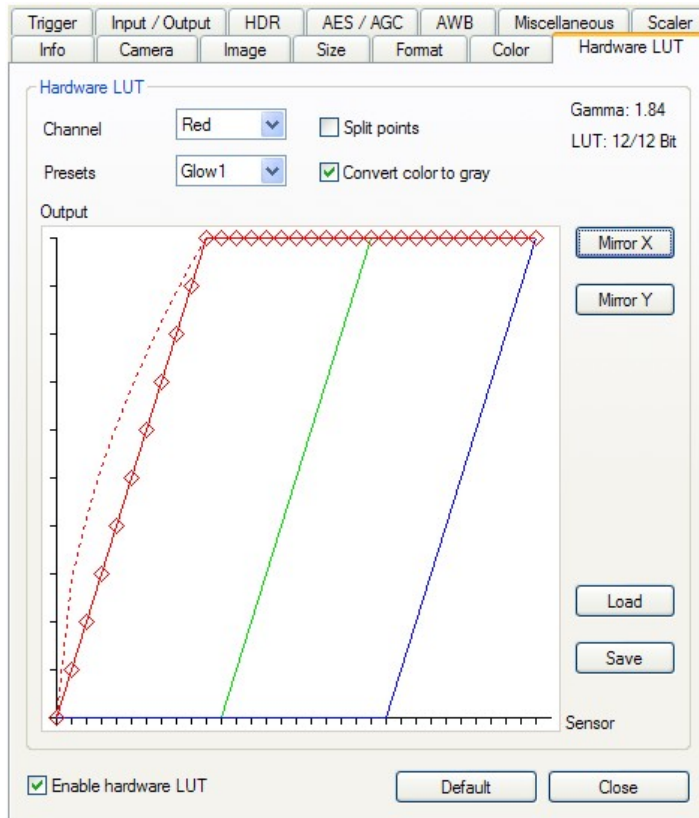


Figure 48: uEye properties - Hardware LUT



This feature is only available for *GigE uEye* cameras.

Hardware LUT

- Channel**
 In this drop down box, you can choose whether you want to display the LUT for all channels or just for red, green or blue.
- Split points**
 The LUT has 32 knee points by default. Knee points are used for defining the individual sections of the curve. When you select the *Split points* check box, each knee point is split into two separate points. Only the start and end points of each curve section can be defined independently of the adjacent sections.
- Presets**
 In this drop down box, you can select and load predefined LUT curves. The following LUT curves are available:

Linear	Linear LUT curve without effect
Negative	Predefined LUT, inverts the image
Glow1	Predefined LUT, false-color representation of the image
Glow2	Predefined LUT, false-color representation of the image
Astro1	Predefined LUT, false-color representation of the image
Rainbow1	Predefined LUT, false-color representation of the image
Map1	Predefined LUT, false-color representation of the image
Cold/Hot	Predefined LUT, false-color representation of the image
Sepic	Predefined LUT, uses sepia toning for coloring the image
Only red	Predefined LUT, shows only the red channel of the image
Only green	Predefined LUT, shows only the green channel of the image
Only blue	Predefined LUT, shows only the blue channel of the image

- *Convert color to gray*

When you are using a *GigE uEye HE* color camera, you can convert the color images to monochrome in the camera before applying the LUT curve. This setting is recommended if you want to use a LUT for false-color representation. The images are transferred in RGB format.

- *Output*

The diagram shows that part of the LUT curve that is selected in the *Channel* box. You can drag and drop each knee point of the curve. To draw a smooth curve for the selected channel, left-click on a blank space next to the curve.

- *Mirror X/Y*

These buttons allow mirroring the curve about the X and Y axes.

- *Load/Save*

Click *Save* to save the current LUT curve to a text file. With *Load*, you can load a saved LUT curve from a file.

- *Enable hardware LUT*

Enables/disables the current LUT curve.

Default

Click this button to reset all parameters to the model-specific defaults.

Trigger

This tab provides the settings for triggered image capture with your *uEye* camera (see also [Digital Input \(Trigger\)](#)).

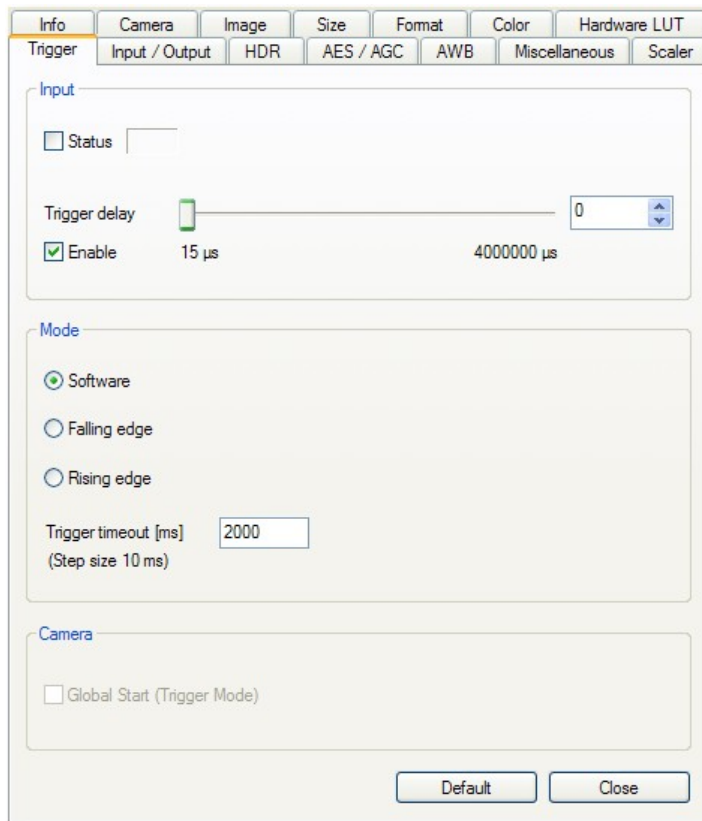


Figure 49: uEye properties - Trigger

Input

- **Status**
If you select this check box, the signal level applied at the camera's trigger input is polled and displayed.
- **Trigger delay**
Select this check box to set a delay between the arrival of a software or hardware trigger signal and the start of exposure.

Mode

With these radio buttons, you choose which trigger mode you want to activate in the camera:

- **Software**
The camera is running in software trigger mode without a signal applied. The images are captured continuously.
- **Falling edge**
The camera captures an image on the falling edge of the signal applied to the trigger input.
- **Rising edge**
The camera captures an image on the rising edge of the signal applied to the trigger input.
- **Single Trigger Timeout [ms]**
Specifies a timeout for the trigger mode. If the camera does not receive a trigger signal within this time, a timeout message is transmitted and the transmission error counter is incremented.

Camera

- *Global Start (Trigger Mode)*

If you select this check box, all rows of a rolling shutter sensor are exposed simultaneously.

Activating Global Start only makes sense when using a flash. This function is not supported by all models.



Please note that the frame rate in trigger mode is always lower than in freerun mode. This is due to the sequential transmission. First the exposure takes place, then the transfer. A new exposure can only be performed after the transmission is completed. Therefore, the freerun mode is faster.

High trigger rates are achieved only with short exposure times and a high pixel clock setting.

Default

Click this button to reset all parameters to the model-specific defaults.

Input/Output

On this tab, you can set the parameters for the flash output and the GPIOs on your *uEye* camera (see also [Digital Output \(Flash Strobe\)](#) and [General Purpose I/O](#)).

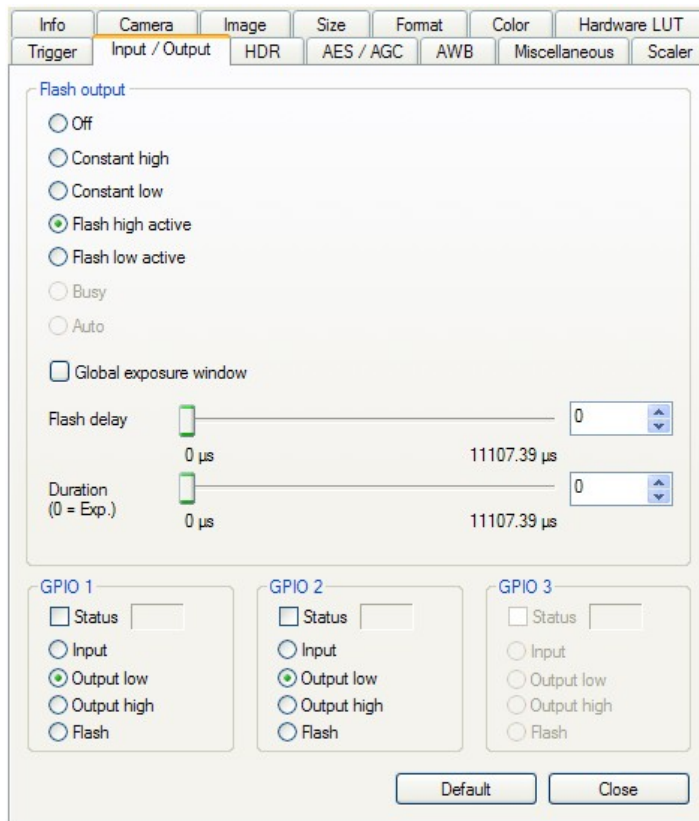


Figure 50: uEye properties - Input/Output



When you are using the *uEye*'s flash function, you need to re-enable the flash (i.e. disable and then activate it again) whenever you change the pixel clock setting or horizontal image geometry. This is necessary to newly synchronize the internal timing settings of the flash output with the start of sensor exposure.

Flash output

With these radio buttons, you choose which digital output function you want to activate on the camera:

- *Off*
The digital output is disabled.
- *Constant high*
The digital output is set to High regardless of the exposure.
- *Constant low*
The digital output is set to Low regardless of the exposure.
- *Flash high active* (only in trigger mode)
The digital output is set to High during the exposure.
- *Flash low active* (only in trigger mode)
The digital output is set to Low during the exposure.

- *Busy*
The digital output signalizes whether the camera is ready for the next trigger. This function is not yet implemented.
- *Auto*
Automatic adjustment of the flash duration when using automatic brightness control. This function is not yet implemented.
- *Global exposure window* (only in trigger mode)
The values for flash delay and duration are calculated to trigger the flash when all sensor rows are exposed simultaneously for sensors with rolling shutters. If you are using the Global Start function, the flash delay has to be set to 0 (see also [Shutter Methods](#)).
- *Flash delay* (only in trigger mode)
Sets the delay for the digital output. After an exposure has started, actuating the digital output is delayed by the time set in Flash delay.
For some models, and depending on the operating mode, delays of up to 200 μ s must be set in order to exactly hit the beginning of the exposure time of the pixels. You can look up the exact value for each camera under *Sensor delay to exposure start* in the [Sensors](#) chapter.
- *Duration* (only in trigger mode)
Sets the switching time of the digital output. The digital output is actuated for the time set in Duration. The value 0 means that the signal is active over the entire exposure time (default).

GPIO 1/2/3 (USB uEye LE and GigE uEye HE cameras only)

The GPIOs (General Purpose I/O) of the *uEye* cameras can be used as inputs or outputs.

- *Status*
Polls the signal level applied to the GPIO.
- *Input*
Sets the *GPIO* as trigger input.
- *Output low*
Sets the *GPIO* as output. The output signal is statically set to *low*.
- *Output high*
Sets the *GPIO* as output. The output signal is statically set to *high*.
- *Flash*
Sets the *GPIO* as flash output. The settings you made in the *Flash output* box are used.



Detailed information on wiring the *uEye* inputs and outputs is provided in the *Electrical Specifications* section of the [Specifications](#) chapter.

Default

Click this button to reset all parameters to the model-specific defaults.

HDR

On this tab, you can configure the HDR mode settings for your *uEye* camera.

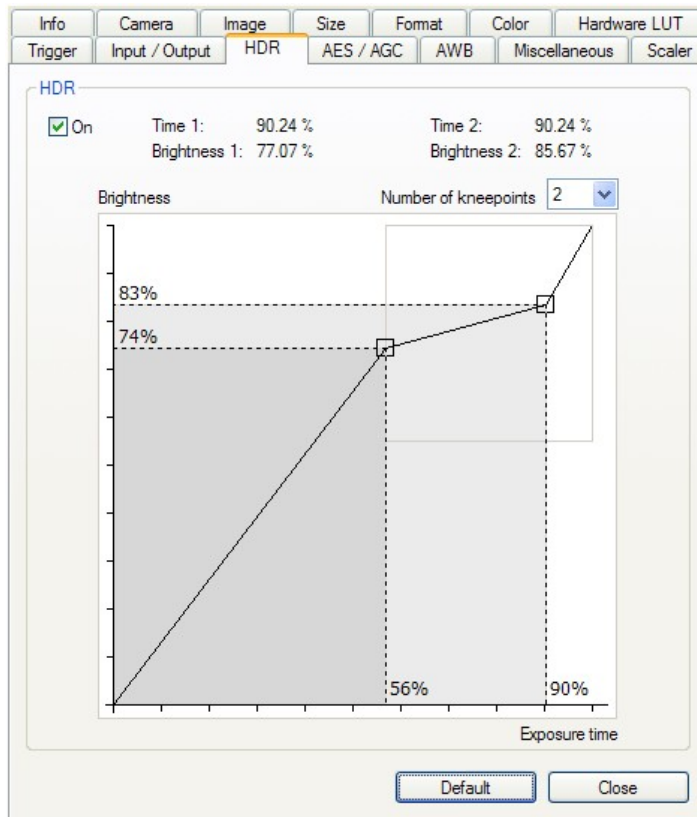


Figure 51: *uEye* properties - HDR

By exposing the sensor cells in two or three phases, the *HDR (High Dynamic Range)* function enhances dynamics during image capture. This means that very bright and very dark image parts are displayed in one image without causing overexposure.

- *Number of knots*

In this drop down box, you can choose whether the HDR curve should have one or two definable knee points.

- *Brightness/Time*

The x value of a knee point indicates the first phase in percent of the currently set exposure time. The y value gives the proportion of maximum pixel intensity for this phase in percent.

Example: A setting of $x = 60$, $y = 80$ would therefore produce the following results:

The first exposure phase lasts for 60% of the set exposure time. In this first exposure phase, all pixels are exposed to at most 80% of maximum pixel intensity and remain at 80% until this phase is over. In the second exposure phase, they are exposed again and may reach the full pixel intensity.



HDR is supported by the UI-122X-X and UI-522X-X models.

For cameras of types UI-122X-C and UI-522X-C, the RGB gain controls must be set to the same values to ensure accurate color rendition in HDR mode.



Using automatic brightness control in HDR mode may lead to short-term brightness fluctuations. To determine the optimum knee point coordinates, we recommend operating the camera at a medium frame rate. A high frame rate reduces the time frame for post-exposure.

Default

Click this button to reset all parameters to the model-specific defaults.

AES/AGC

On this tab, you can set parameters for automatically adjusting the exposure time and sensor gain of your *uEye* camera (see also [Automatic Image Control](#)).

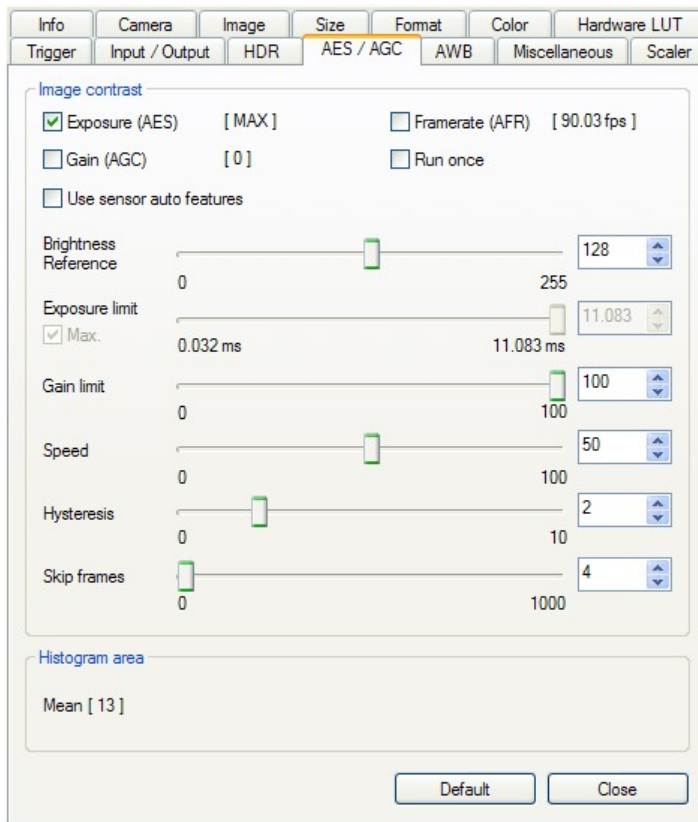


Figure 52: uEye properties - AES/AGC



- **Auto Exposure Shutter (AES)** automatically adjusts image brightness based on the exposure shutter setting. Long exposure times may cause motion blur.
- **Auto Gain Control (AGC)** automatically adjusts image brightness based on the hardware gain control setting of the camera sensor. You can activate this function in addition to AES if ambient light conditions are poor. High gain settings may cause artefacting.
- **Auto Frame Rate (AFR)** adjusts the *frame rate* to allow longer exposure times (see also [Pixel Clock, Frame Rate and Exposure Time](#)). Selecting this check box might decrease the frame rate substantially. This function is only available when AES is active.



The sensor's auto features are only supported by the sensor of the UI-122x/522x camera models. Please also read the notes on using this sensor, which are provided in the [Specifications: Sensor Data](#) chapter.

Image contrast

Use the following settings to configure automatic brightness control:

- **Exposure (AES)**
Select this check box to automatically adjust the image brightness based on the exposure shutter setting.

- *Framerate (AFR)*
Select this check box to adjust the frame rate in such a way that the exposure shutter is optimized. This option is only available when AES is active.
- *Gain (AGC)*
Select this check box to automatically adjust the image brightness based on the gain setting.
- *Run once*
Select this check box to automatically disable the adjustment functionality as soon as the target value is reached.
- *Use sensor auto features*
If your sensor offers automatic image brightness adjustment, you can select this check box to enable the feature.
- *Brightness reference*
Default average grayscale value (brightness) of the image.
- *Exposure limit*
Sets the upper limit for the exposure time. Gain control is activated as soon as the upper limit of the exposure time is reached.
The maximum value for automatic exposure time control is linked to the camera settings. If the maximum possible exposure time value has changed, e.g. through changes in the timing parameters, then this value is applied as the maximum Auto Exposure value. Set values that are less than the maximum value are not affected.
- *Gain limit*
Sets the maximum gain limit. On reaching the lower limit (gain = 0), the exposure time adjustment range is activated.
- *Speed*
Sets the adjustment speed. The higher the speed control is set, the faster the adjustment. Setting the speed control to 0 disables the adjustment functionality.
- *Hysteresis*
Sets the hysteresis range for control stabilization (see also [Automatic Image Control: Hysteresis](#)).
- *Skip frames*
Sets the number of frames that will be skipped during automatic image control when in freerun mode is active (see also [Automatic Image Control: Control Speed](#)).

Histogram area

The histogram area specifies which area of the image will be used for calculating the average grayscale value of the image.

You can set the size of the histogram area by using the tools on the toolbar (see [Toolbar: Auto Features](#)).

Default

Click this button to reset all parameters to the model-specific defaults.

AWB (Auto White Balance)

On this tab, you can set parameters for automatically adjusting the white balance of your *uEye* camera (see also [Automatic Image Control](#)).

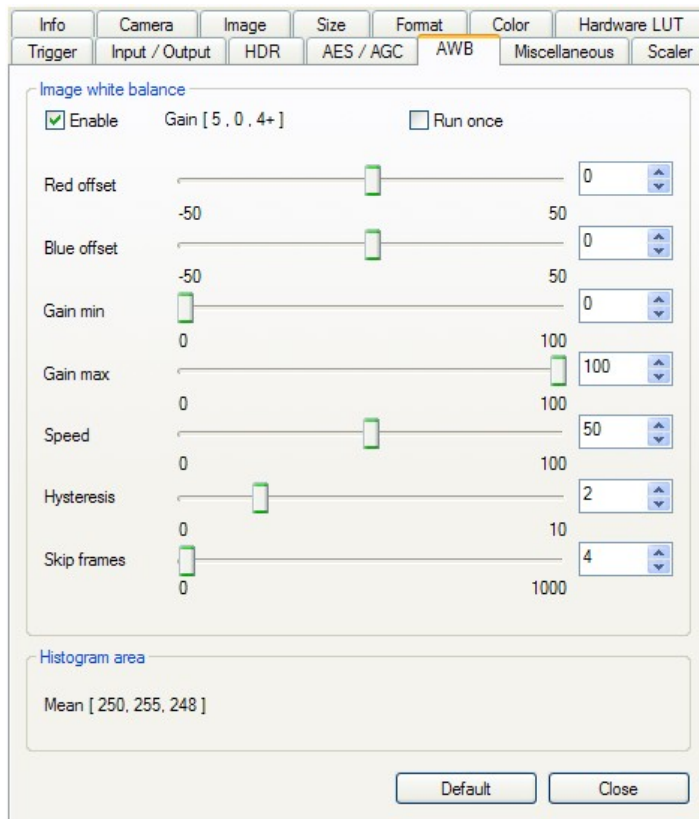


Figure 53: uEye properties - AWB



Every light source has a specific color temperature. Images captured with a camera can have a reddish (low color temperature) or bluish hue (high color temperature), depending on the color temperature of the light. This hue can be corrected by white balancing.

To carry out white balancing, aim the camera at a surface of a uniform gray color. You can perform white balancing either manually with the RGB gain control or by using the *uEye's Auto White Balance (AWB)* function.

Image white balance

- *Enable*
Activates automatic white balance.
- *Run once*
Select this check box to automatically disable the adjustment functionality as soon as the target value is reached.
- *Red offset/Blue offset*
With these sliders you can manually adjust the red and blue content of the image.
- *Gain min*
With this slider you can set a basic gain value for white balancing. Color cameras without master gain can emulate this base value by using the Gain min value set for AWB.
- *Gain max*
Upper adjustment limit.

- *Speed*
Sets the adjustment speed. The higher the speed control is set, the faster the adjustment. Setting the speed control to 0 disables the adjustment functionality.
- *Hysteresis*
Sets the hysteresis range for control stabilization (see also [Automatic Image Control: Hysteresis](#)).
- *Skip frames*
Sets the number of frames that will be skipped during automatic image control when in freerun mode is active (see also [Automatic Image Control: Control Speed](#)).

Histogram area

The histogram area specifies which area of the image will be used for calculating the average grayscale value of the image.

You can set the size of the histogram area by using the tools on the toolbar (see [Toolbar: Auto Features](#)).

Default

Click this button to reset all parameters to the model-specific defaults.

Miscellaneous

This tab provides parameters for setting the hotpixel correction and test image function for your *uEye* camera.

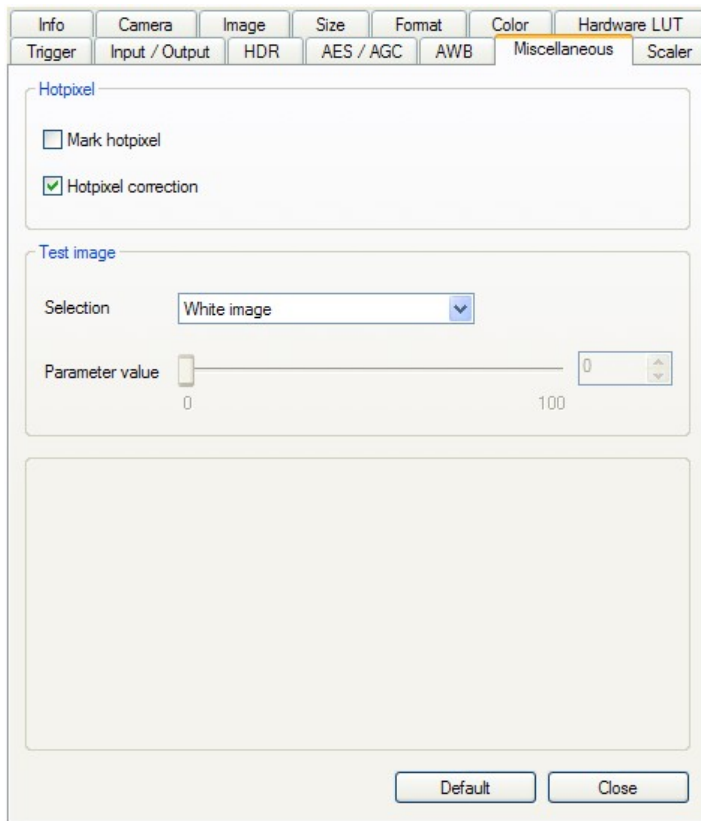


Figure 54: *uEye* properties - Miscellaneous

Hotpixel

Hotpixels are sensor pixels whose brightness value deviates significantly from the brightness value of the surrounding pixels in the case of long exposure times or a high gain setting. Basically every sensor has a small number of hot pixels. CCD sensors have less hotpixels than CMOS sensors, which is due to the different sensor systems. The hot pixels are detected during *uEye* camera testing and written to a coordinate list in the camera. If hotpixel correction is enabled, the correction function eliminates these pixels before debayering.

- *Mark hotpixel*
If you select this check box, the hotpixel positions stored in the camera are graphically represented in the image.
- *Hotpixel correction*
Select this check box if you want to enable a software-based hotpixel correction.

Test image

The camera transmits a selectable test image that you can use for testing the data transmission. You can choose from different types of test images, depending on the camera type.

- *Selection*
Some of the test images (e.g. *Black image* and *White image*) are created by the sensor and are available in both, USB-based and GigE-based *uEye* models.
With all *GigE uEye HE* cameras (CMOS and CCD sensors) you can also choose moving and stationary test images that are created by the camera hardware (e.g. *Colored wedge*,

Animated line, Coldpixel/Hotpixel grid).

Test images marked (RAW) are only properly displayed in raw Bayer mode (see [Color](#)).



Figure 55: Selecting test images
(here: UI-5480-C)

- **Parameter value**

You can adjust the appearance of some of the test images with the *Parameter value* slider.



Animated test images are ideal for testing recorded sequences.

With some sensors, the sensor gain setting has an influence on the test image.

For USB cameras, you can use a white test image to check the camera's maximum load on the USB bus. Due to the transmission process, completely white camera images require a somewhat more bandwidth on the USB bus than completely black images.

Default

Click this button to reset all parameters to the model-specific defaults.

Scaler

On this tab, you can configure the image scaling settings provided by specific *uEye* sensors.

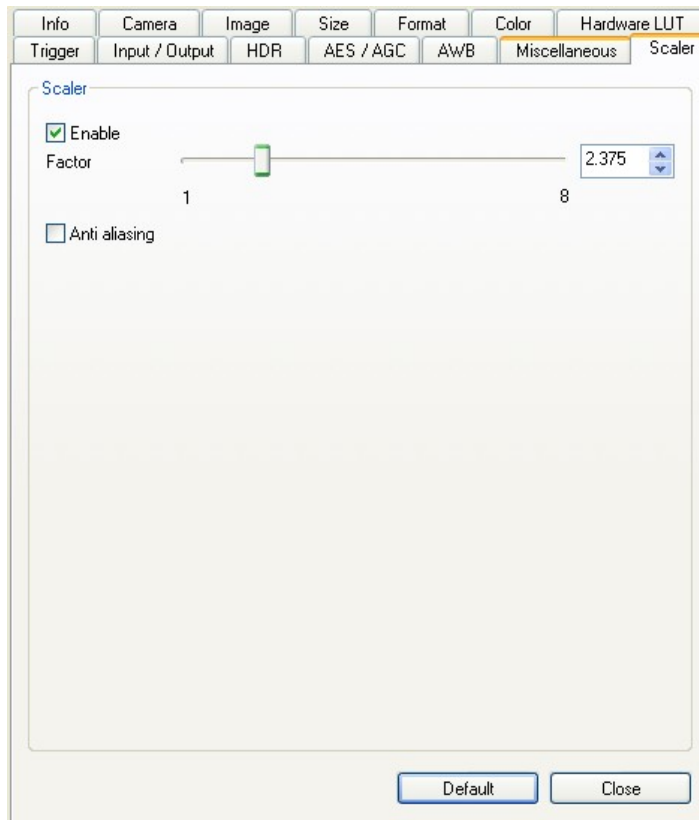


Figure 56: uEye properties - Scaler



The scaler is currently only supported by the sensors of the UI-149x/549x series.



Using the scaler has no effect on the maximum possible frame rate.

Scaler

- **Enable**
Enables the sensor's image scaling feature.
- **Factor**
With this slider, you can set the scaling factor in a range from 1.0 to 8.0.
- **Anti aliasing**
The anti aliasing function smoothes the image edges during scaling.

Default

Click this button to reset all parameters to the model-specific defaults.

4.3.3 uEye Player

4.3.3.1 Functionality



Only one instance of the uEye Player can be opened at a time. This means that the player cannot be displayed multiple times on screen.

Using the *uEye Player*, you can open and play back AVI files, e.g. created with the *uEye Demo* program, in MJPEG format. Images stored in JPG/BMP format can also be displayed.

The *uEye Player* can be accessed as follows:

- *Start* → *All Programs* → *IDS* → *uEye* → *uEye Player*

After program start, the *uEye Player* will display the following dialog box:




Figure 57: uEye Player

The user interface of the *uEye Player* adjusts to the language of the operating system. After the player has started, only the button for loading a video file is active. How to load a video file will be explained in the following section.

4.3.3.2 Loading an AVI File



After clicking the  button, the "Open File" dialog box opens where you can select one or more files to be opened. If you select multiple files, they will be played back one after the other in alphabetical order.

Alternatively, AVI files can also be opened and played back simply by drag and drop. To do this, drag the files with the left mouse button pressed into the uEye Player dialog box; then, release the mouse button.

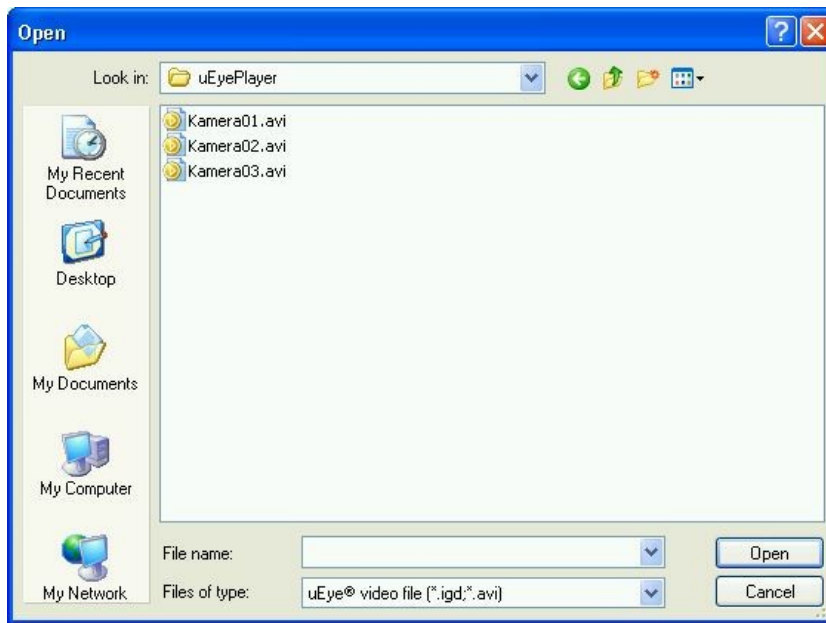


Figure 58: Playing AVI files

Once the required file has been opened, another window for the video film appears on top of the uEye Player. You can move this window freely around the screen, independently of the player window.




















When you place the mouse pointer inside the video window, its display changes to a small magnifying glass and the zoom function of the uEye Player is enabled. Using the left mouse button, you can now select an area within the video image that will be resized to fill the window (Zoom In), even during playback. Double-clicking the left mouse button inside the window will revert the display to its original size (Zoom Out).



Figure 59: uEye Player - Playback

4.3.3.3 Operation Controls

The buttons in the *uEye Player* user interface are for the most part self-explanatory and are based on the keys and symbols of a standard video recorder.

	Reverse: play video backwards
	Play: play video forwards
	Stop: stop playing the video (symbol appears after you click the Play button). The last frame will be frozen.
	Jump to start of video
	Fast rewind
	One frame back
	One frame forward
	Fast forward
	Jump to end of video
	Go to specific frame. When you click this button, a small dialog box will open where you can enter the frame number.
	Start Loop mode (blue text)
	Stop Loop mode (red text)
	Start of <u>playback loop</u>
	End of <u>playback loop</u>
	Save current frame as BMP file or JPEG file
	Print current frame
	Sound on/off
	Open video file
	Close video file

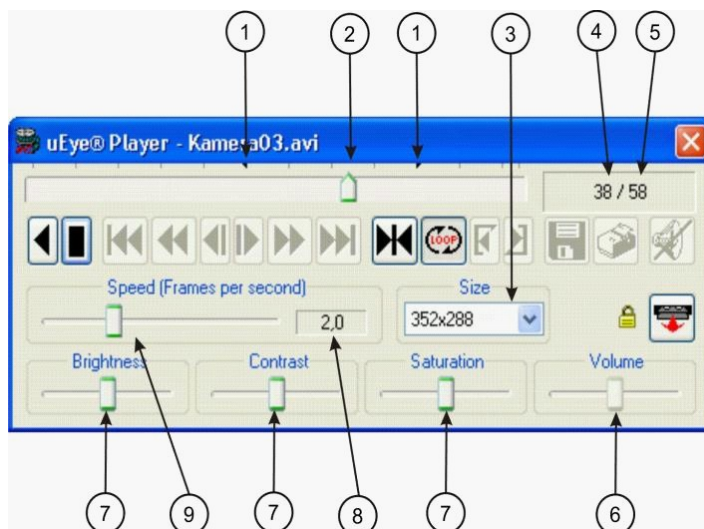


Figure 60: uEye Player operation controls

1	Position markers for the defined playback loop
2	Current position in video file
3	Size of the video display window in pixels
4	Current frame number
5	Number of frames in video file
6	Volume control
7	Frame display parameter settings. Each slider can be reset to its default setting by clicking it with the right mouse button. The settings are also applied to the subsequent files.
8	Current playback speed setting
9	Playback speed in relation to the recording speed from 0.1 to 200 fps. The values are set in increments.



Using the  button, you can jump directly to a specific frame. To do so, enter a numerical value between 1 and the total number of frames in the video sequence.





Figure 61: Jump to specific frame

4.3.3.4 Loop Mode

When using the *uEye Player*, you can select specific periods of time within the video sequence

and play them back in an endless loop. To do this, click the  button. This enables the two icons for marking the start and end of the playback loop. To select the start position, use the mouse to drag the position marker to the desired start position in the endless loop and then click

the  button. Then, set the position marker to the desired end position using the same method.

Click the  button to complete defining markers.

4.3.3.5 Video Window and Full Screen Mode

The video window is displayed dynamically. The possible display sizes are determined based on the capture resolution and the screen resolution.



The video is played back at the same aspect ratio that was used for the capture.

To enable full screen mode, press the **b+F** keys. In this mode, you can use the keyboard for playback control. To quit this mode, press **^** or press **b+F** again.

Key combinations in full screen mode

b+F	Start/stop full screen mode
^	Quit full screen mode
b+O	Open video file
Z (left arrow)	One frame back
X (right arrow)	One frame forward
k (spacebar)	Start/stop video playback

4.4 uEye Camera Basics

4.4.1 Operating Modes

4.4.1.1 Freerun Mode

In freerun mode, the camera sensor captures one image after another at the set frame rate. Exposure and readout/transfer of the image data are performed in parallel. This allows the maximum camera frame rate to be achieved. The frame rate and the exposure time can be set separately. The captured images can be transferred one by one or continuously to the PC. If trigger mode is active, you need to disable it before activating freerun mode.

Single frame mode (snap mode)

The next image exposed by the sensor will be transferred. You cannot use the uEye flash outputs in this mode.

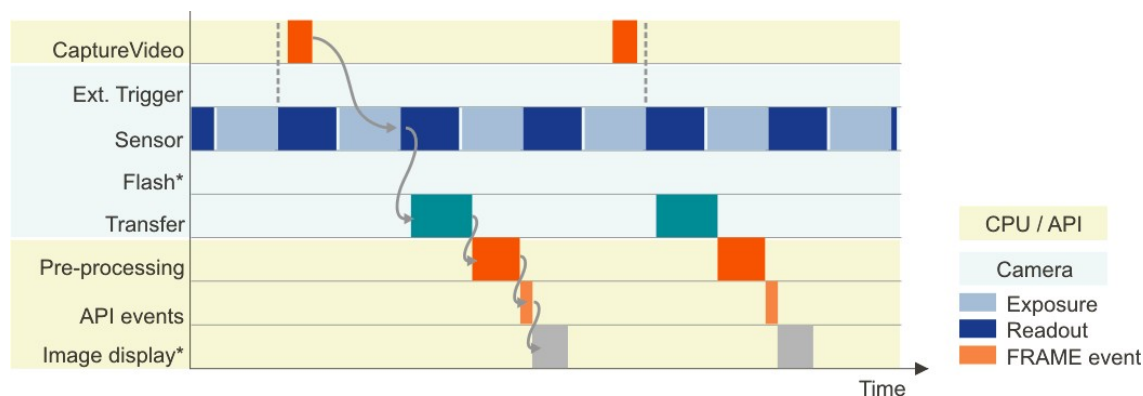


Figure 62: Freerun mode (snap mode)

Continuous mode (live mode)

Images are captured and transferred continuously. You can use the uEye flash outputs.

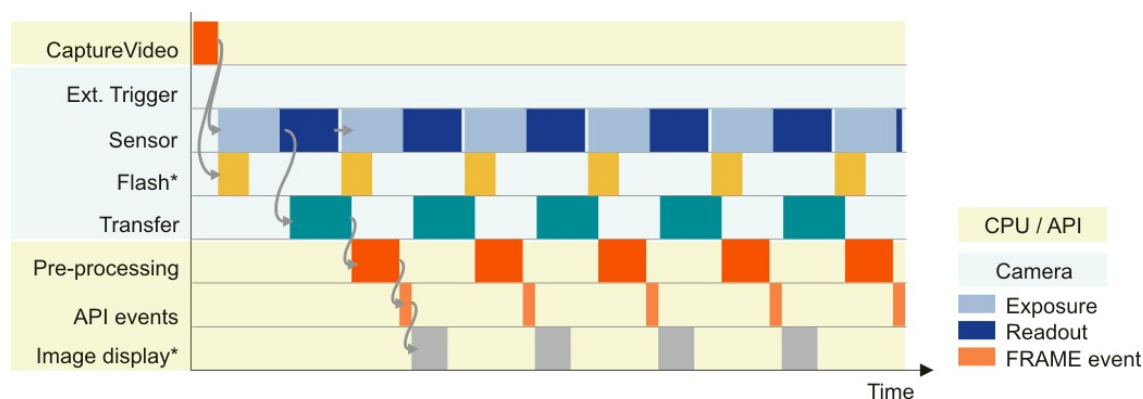


Figure 63: Freerun mode (live mode)

*) Optional function. The start time and duration of the flash signal are defined by the *Flash delay* and *Duration* parameters (see also [Camera Settings: I/O](#)).



These illustrations show a schematic view of the image capture sequence. The sensor exposure and readout times and the transmission times depend on the camera settings. The pre-processing time depends on the API functions you are using (e.g. color conversion, edge enhancement).

4.4.1.2 Trigger Mode

In trigger mode, the sensor is on standby and starts exposing on receipt of a trigger signal. A trigger event can be initiated by a software command (software trigger) or by an electrical signal via the camera's digital input (hardware trigger). For the specifications of the electrical trigger signals, see the [Electrical Specifications](#) chapter.



In trigger mode, the maximum frame rate is lower than in freerun mode because the sensors expose and transfer sequentially. The possible frame rate in trigger mode depends on the exposure time. The time required for acquiring a frame in trigger mode can be approximated with the following formula:

$$t_{\text{acquisition}} = \text{Current exp. time} + \left(\frac{1}{\text{max. frame rate}} \right)$$

Example: At the maximum exposure time, the frame rate is about half as high as in freerun mode; at the minimum exposure time, the frame rate is about the same.

In the [camera properties](#), choose which trigger mode you want to use.

Software trigger mode

When this mode is enabled, calling the Snap function triggers the capture of an image, which is then transferred to the PC. If you call the Live function in this mode, the image capture is triggered continuously and images are transferred continuously.

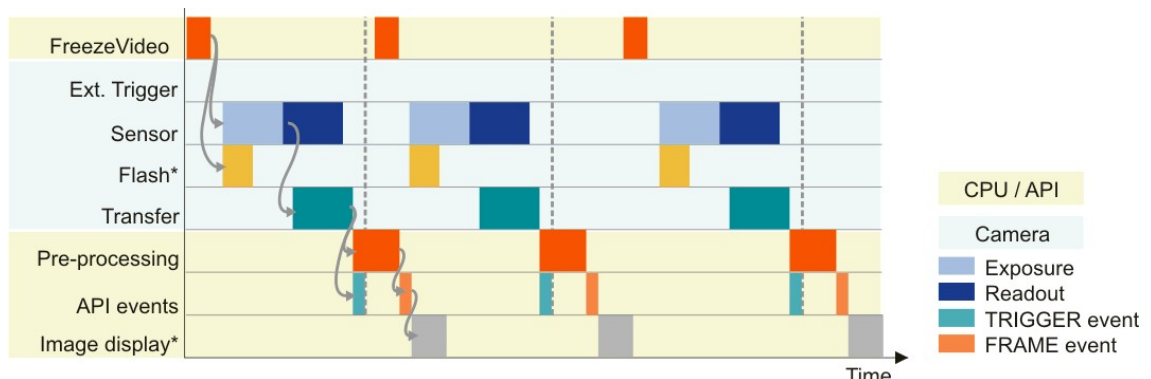
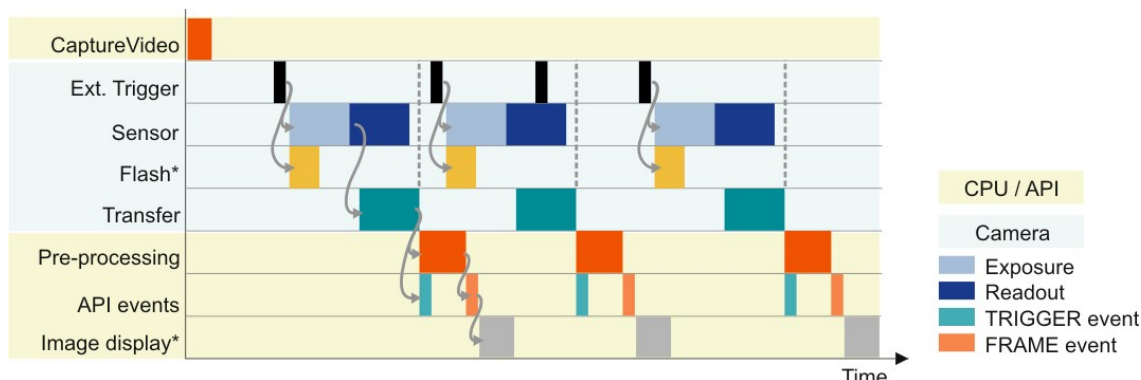


Figure 64: Software trigger mode with continuous image capture

Hardware trigger mode

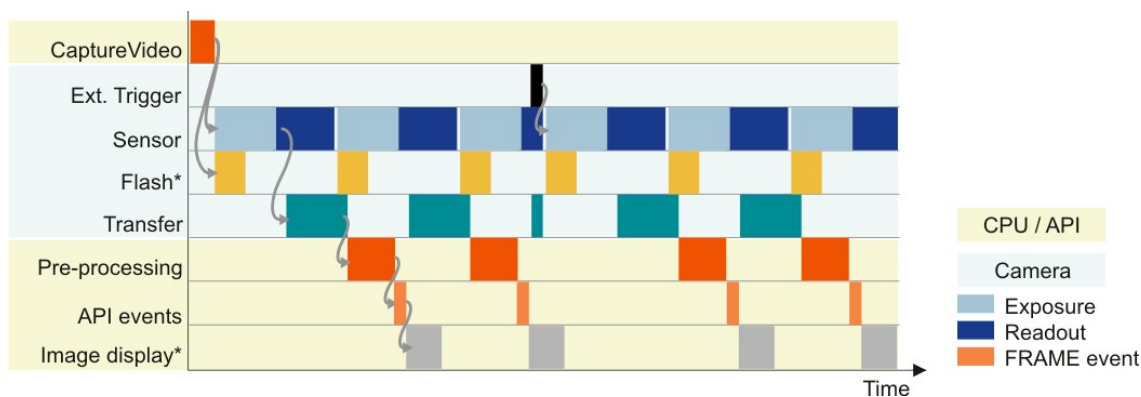
When this mode is enabled, calling the Snap function makes the camera ready for triggering just once. When the camera receives an electrical trigger signal, one image is captured and transferred.

If you call the Live function, the camera is made ready for triggering continuously. An image is captured and transferred each time an electrical trigger signal is received; the camera is then ready for triggering again (recommended procedure).



Freerun synchronization

In this mode, cameras running in freerun mode (live mode, see above) can be synchronized with an external trigger signal. The cameras still remain in freerun mode. The trigger signal stops and restarts the current image capture process. You can use this mode to synchronize multiple cameras that you are operating in the fast live mode. Not all camera models support this mode.



*) Optional function. The start time and duration of the flash signal are defined by the *Flash delay* and *Duration* parameters (see also [Camera Settings: I/O](#)).



These illustrations show a schematic view of the image capture sequence. The sensor exposure and readout times and the transmission times depend on the camera settings. The pre-processing time depends on the API functions you are using (e.g. color conversion, edge enhancement).

4.4.1.3 Standby

uEye cameras can be set to a power-saving standby mode. Standby mode switches off the sensor of CMOS cameras and the timing board of CCD cameras. The camera remains open in the software.

In standby mode, the camera cools down and the number of hot pixels visible when longer exposure times are used is reduced.

Standby is the default state when the camera is not open in the software. When you open the camera or switch to a different mode (*freerun* or *trigger* mode), the camera wakes up from standby mode.



In standby mode, you can continue to use the camera's digital inputs or outputs.

4.4.2 Image Display Modes

The *uEye* driver provides different modes for displaying the captured images on Windows systems. We recommend using the Bitmap mode or the Direct3D functions, depending on your specific application.

For a list of API functions for image display see [How To Proceed: Image Display](#).



The *DirectDraw BackBuffer* and *DirectDraw Overlay Surface* display modes are obsolete. Please use the Direct3D functions instead (see also [Obsolete Functions](#)).

1. Bitmap mode (Device Independent Bitmap, DIB)

In Bitmap mode, images captured by the *uEye* are written to the random access memory of the PC. Programming the image display is up to the user. The application software uses the `is_RenderBitmap()` function to initiate the image display by the graphics card. This may result in a slightly higher CPU load as compared to the Direct3D display.

The advantage of Bitmap mode is that it is compatible with all graphics cards and that image data in the memory is directly accessible. Programming of overlay functions is up to the user. Since the operating system controls the image display, the image may be completely or partly overlapped by other windows and dialog boxes.

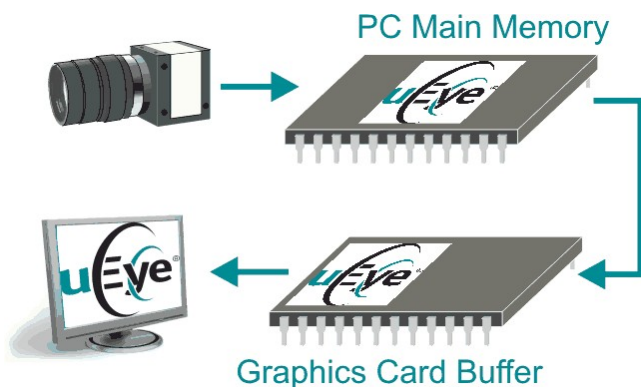


Figure 67: How the Bitmap mode works

2. Direct3D mode (only under Windows with DirectX)

In this mode, the *uEye* driver writes the image data to the invisible area of the graphics card. This process runs automatically and does not have to be controlled by the application software. It requires an installed Direct3D driver, sufficient memory on the graphics card and Direct3D function support by the graphics card. For this purpose, graphics cards generally provide better performance than graphics chips integrated on the mainboard. In Direct3D mode, the CPU load may be lower than in Bitmap mode. You can display overlay data and also scale the video image.

The Direct3D mode and the overlay functions can be configured using the `is_DirectRenderer()` API function.

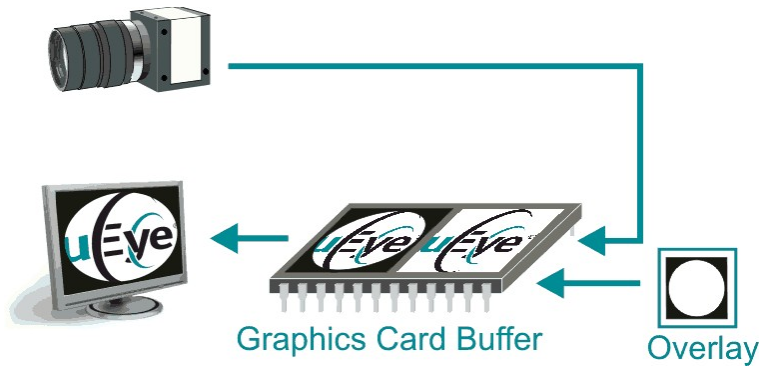


Figure 68: How the Direct3D mode works

Comparison of the display modes

The following table illustrates the major differences between the display modes:

	Bitmap mode	Direct3D mode
Graphics card requirements	Low. No special graphics hardware required. Runs on all systems.	High. Graphics card has to support Direct3D. Does not run on all systems.
Operating system	Windows, Linux	Only Windows with DirectX
Programming effort	Greater. Memory management, event handling and display performed by the application.	Low. Memory management, event handling and display performed by DirectX.
CPU load	Slightly increased by copying of data.	Low. Display performed by graphics card.
Overlay functions	Not available. A simple overlay can be programmed by the user.	Integrated. Complex overlays can be displayed without flicker.
Access to image memory	Direct access possible. Image data already provided in user memory.	Possible using Steal Mode. Single images can be copied to the user memory.

4.4.3 Sensor

4.4.3.1 Sensor Sizes

The size of a digital camera sensor is usually specified in inches. However, the specified value does not indicate the actual size of the active sensor area. The sensor size specifications date back to the formerly used tube systems: The curvature of the imaging surface of the camera tube caused distortions to the display, reducing the usable capture area of a 1" tube to a rectangle with a diagonal of 16 mm.

With the introduction of the semiconductor sensor technology, the dimensional specifications were taken over from tube systems. For this reason, a sensor whose active area diagonal measures 16 mm is specified as a 1-inch sensor. The following illustrations show the most common sensor sizes. The diameter in inch multiplied with 2/3 equals approximately the actual sensitiv area in millimeters.

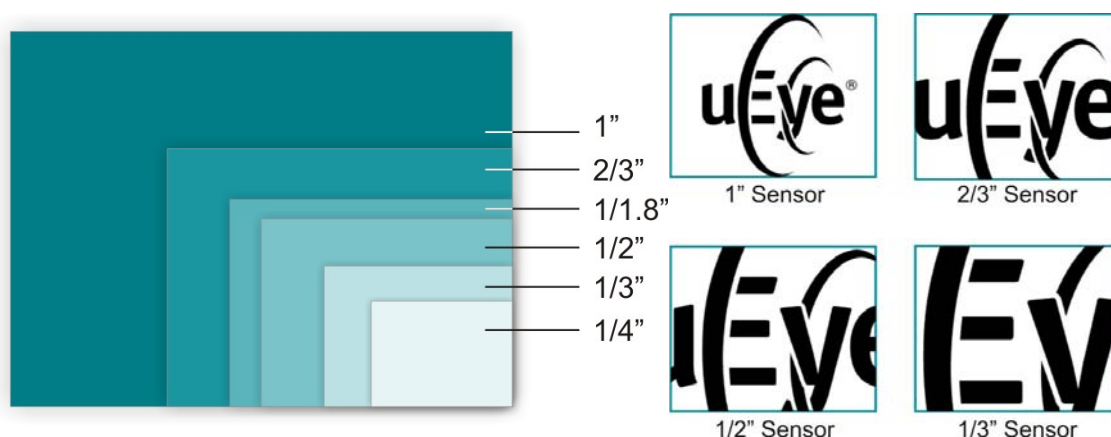


Figure 69: Comparison of common sensor sizes and examples for different fields of view.

The size of each single sensor cell (pixel) depends on the size of the active sensor area and the resolution. In general, less pixels over the same sensor area (or a larger sensor area with the same resolution) will result in greater photosensitivity of the sensor.

4.4.3.2 Fill Factors

The fill factor is the percentage of the pixel area that is exposed to light during exposure. Ideally this would be 100%. Since other elements are located on the sensor surface besides the light-sensitive photodiodes, this value may be reduced to approx. 30 - 50%, depending on the sensor technology. The use of micro lenses compensates for this and increases the fill factor to 90% or more. Micro lenses collect the light that falls onto a photocell, thus increasing the useable sensor area.

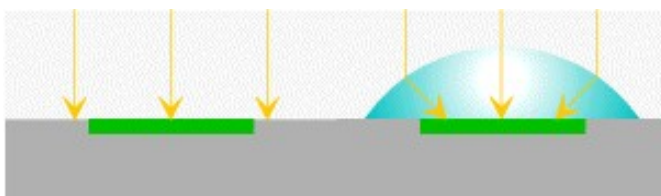


Figure 70: Using micro lenses to increase the effective fill factor

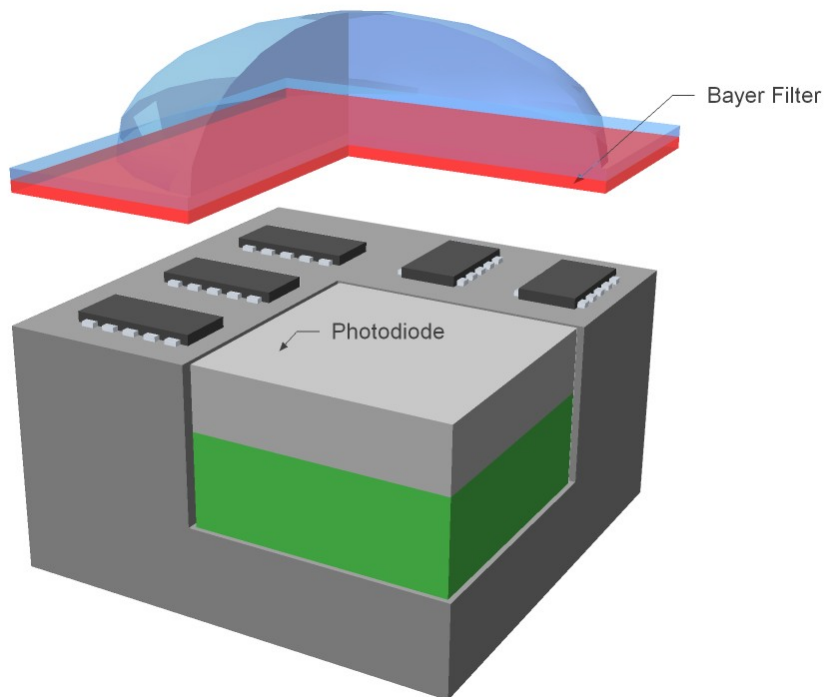


Figure 71: CMOS pixel design with Bayer filter (red) and micro lens



Some CMOS sensors have micro lenses offset to the sensor edge. They compensate for shadows created by obliquely incident light. The use of parallel light causes slight color variations. These may occur if telecentric stops or lenses with large apertures whose last optical element is located at a great distance are used. The following *uEye* models are equipped with CMOS sensors with offset micro lenses:

- 164x-C and 564x-C
- 155x-C and 555x-C
- 148x-M/C and 548x-M/C

4.4.3.3 Color Filter (Bayer Filter)

For technical reasons, digital image sensors can only detect brightness information, but no color information. To produce color sensors, a color filter is applied to each photocell (pixel). The arrangement of the color filters is illustrated in the following figure. Two out of every four pixels have a green filter, one pixel has a red filter and one has a blue filter. This color distribution corresponds to the color sensitivity of the human eye, and is called the Bayer filter pattern. With the help of the Bayer pattern the correct brightness and color information can be calculated for each pixel. Full sensor resolution is retained.

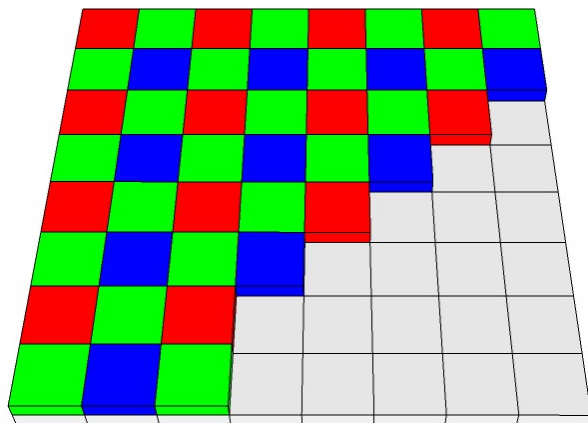


Figure 72: Bayer RGB filter pattern

Bayer conversion

A Bayer conversion, also referred to as de-Bayering, is carried out to determine the color information from the raw sensor data (raw Bayer). By default all *uEye* cameras transmit the image data to the PC in *raw Bayer* format. The PC then uses the functions of the *uEye API* to convert the image data to the color format you need for displaying or further processing the data.

GigE uEye cameras additionally allow de-Bayering in the camera. In this case, the color images are already finished when they are transmitted to the PC. This reduces the load on the computer's CPU and increases the transmission bandwidth required by the camera.

To convert the colors, a filter mask moves over the image and calculates a color value for each pixel from the surrounding pixels. The *uEye API* provides two filter masks that differ in image quality and CPU load (see also [is_SetColorConverter\(\)](#)).

- **Normal Quality** (Mode `IS_CONV_MODE_SOFTWARE_3x3` / `IS_CONV_MODE_HARDWARE_3x3`)
A smaller filter mask is used for conversion. This algorithm has a low load on the CPU. The filter's averaging function may cause a slight blur. Noise is reduced. This filter is recommended for image processing tasks.
- **High Quality** (Mode `IS_CONV_MODE_SOFTWARE_5x5`)
A large filter mask is used for conversion. This algorithm offers very accurate color positioning and an increased level of detail. The CPU load is higher than with the normal filter. This filter is recommended for visualization applications.

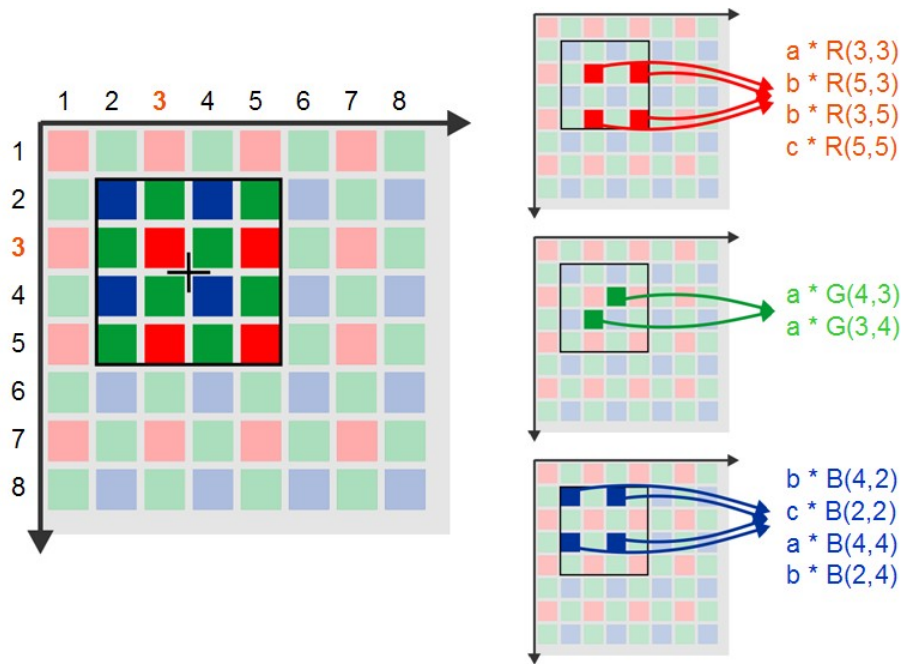


Figure 73: Bayer conversion using the standard mask

4.4.3.4 Shutter Methods

The image is recorded in the sensor in four phases:

- Reset pixels of the rows to be exposed
- Exposure of pixel rows
- Charge transfer to sensor
- Data readout

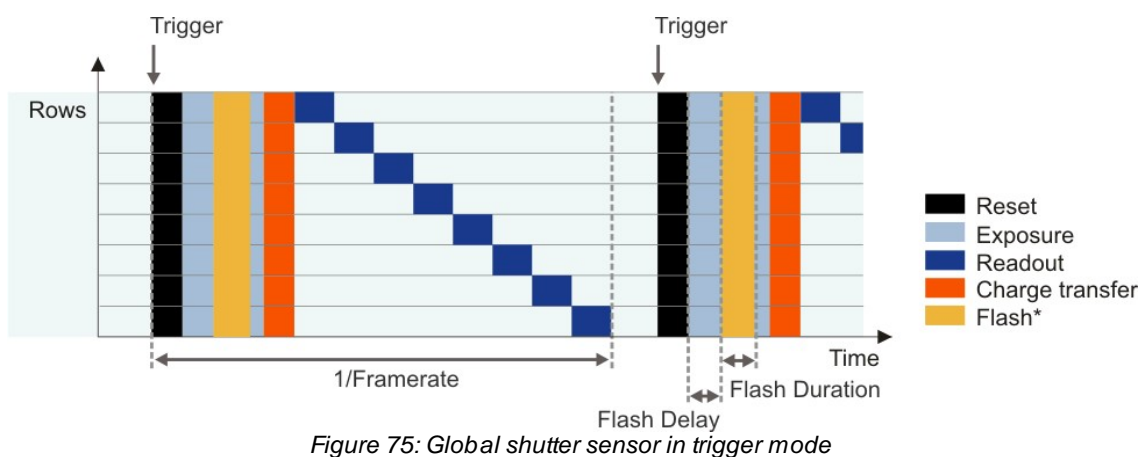
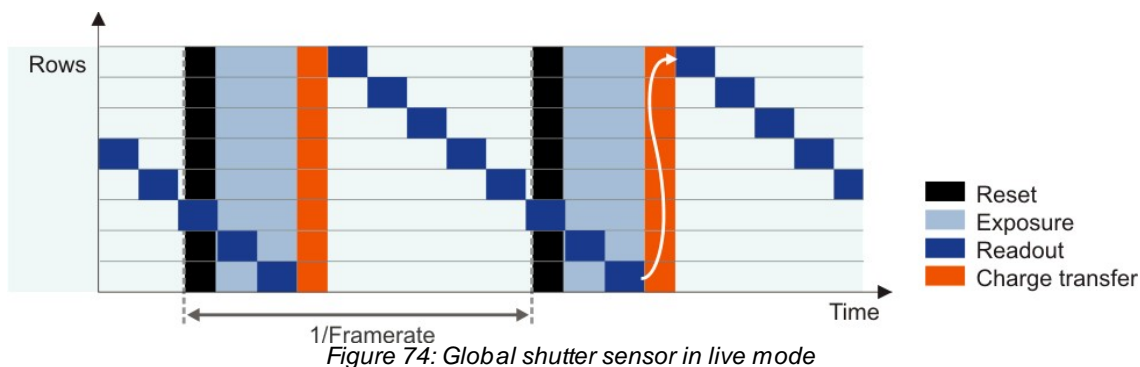
The sensor cells must not be exposed during the readout process. The sensors of the *uEye* cameras have no mechanical shutters, but work with electronic shutter methods instead. Depending on the sensor type, either the rolling shutter method or the global shutter method is used.

Global Shutter

On a global shutter sensor, all pixel rows are reset and then exposed simultaneously. At the end of the exposure, all rows are simultaneously moved to a darkened area of the sensor. The pixels are then read out row by row.

Exposing all pixels simultaneously has the advantage that fast-moving objects can be captured without geometric distortions. Sensors that use the global shutter system are more complex in design than rolling shutter sensors.

All *uEye* CCD sensors as well as some CMOS sensors use the global shutter method.



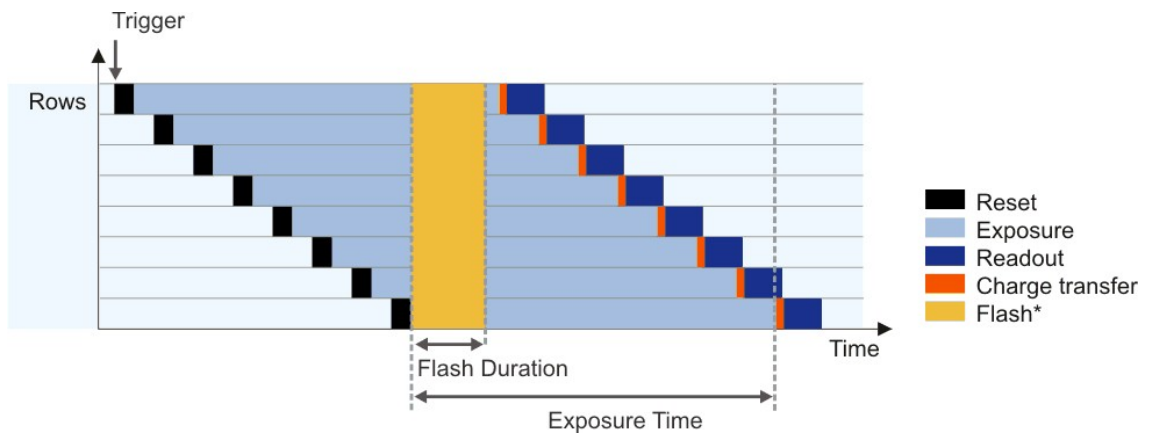
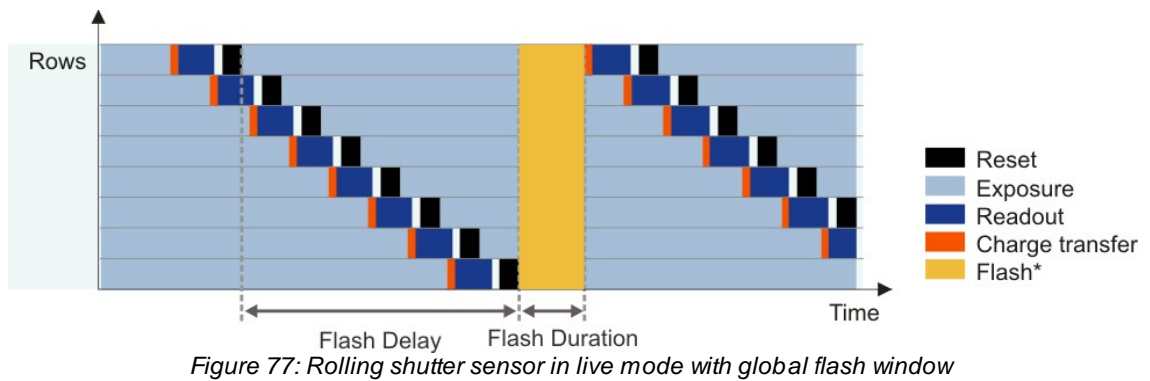
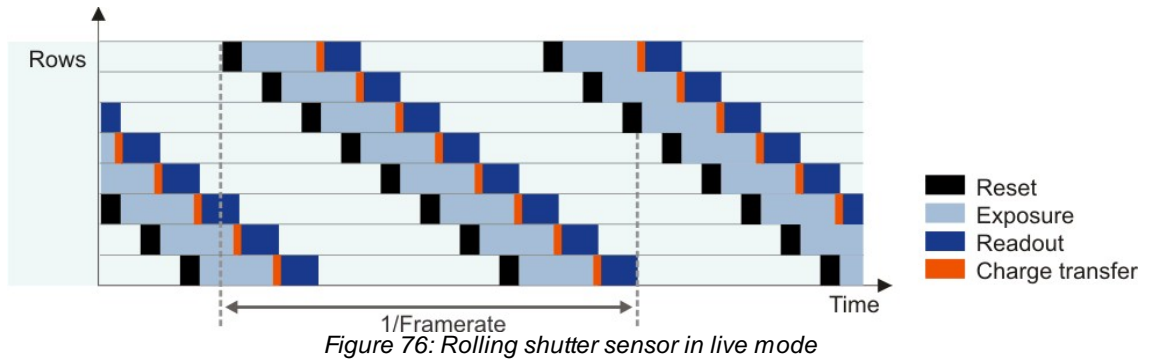
*) Optional flash function. The start time and duration are defined by the *Flash delay* and *Duration* parameters (see also [Camera Settings: I/O](#)).

Rolling Shutter

With the rolling shutter method, the pixel rows are reset and exposed one row after another. At the end of the exposure, the lines are read out sequentially. As this results in a time delay between the exposure of the first and the last sensor rows, captured images of moving objects are distorted.

To counteract this effect, the *uEye* software provides a Global Flash window where you set the time by which flash activation is delayed. You can also specify the flash duration. This allows implementing a global flash functionality which exposes all rows of a rolling shutter sensor simultaneously.

Rolling shutter sensors offer a higher pixel density compared to global shutter CMOS sensors. The rolling shutter system is used in *uEye* cameras with high-resolution CMOS sensors.



^{*)} Optional flash function. The start time and duration are defined by the *Flash delay* and *Duration* parameters (see also [Camera Settings: I/O](#)).

Rolling Shutter with Global Start

Some rolling shutter sensors also provide a global start mode, which starts exposure of all rows simultaneously (see illustration). For best results, use a flash for this mode. No light is allowed to fall on the sensor outside the flash period because otherwise the image brightness will be distributed unevenly.

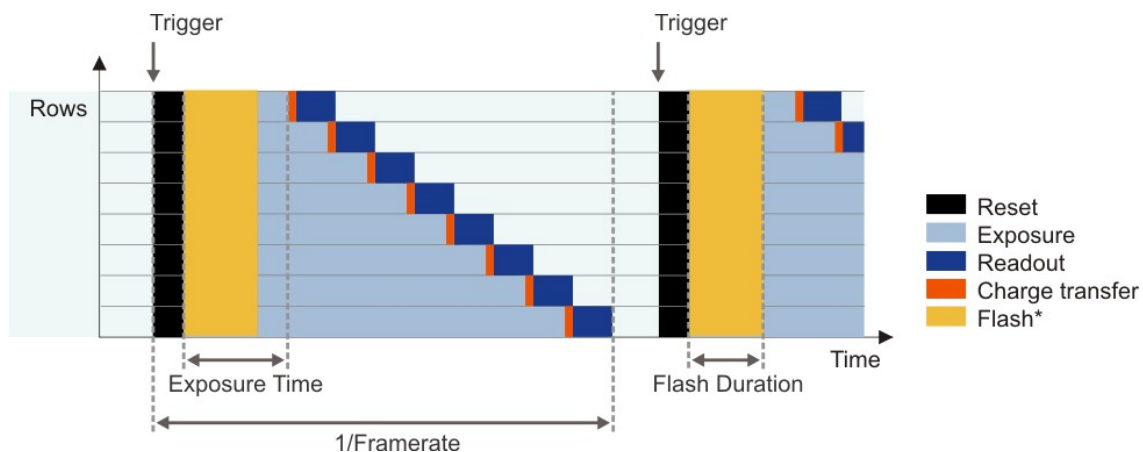


Figure 79: Rolling shutter sensor in trigger mode with Global Start function

^{*)} Optional flash function. The start time and duration are defined by the *Flash delay* and *Duration* parameters (see also [Camera Settings: I/O](#)).

4.4.4 Reading out Partial Images

The camera sensors have defined resolutions which are given as the number of pixels (width x height). However, for some applications it may be necessary to read out only a selected part of the sensor area or to reduce the local resolution. For this purpose, the *uEye* cameras provide various functions:

- Area of Interest (AOI)
- Binning (combining) pixels
- Subsampling (skipping) pixels

These functions reduce the amount of data to be transferred and thus allow you to increase the frame rate considerably, depending on the camera model.

4.4.4.1 Area of Interest (AOI)

Using this function, you can set the size and position of an area of interest (AOI) within an image. In this case, only data included in this AOI will be read out and transferred to the computer. The smaller partial image enables the camera to use a higher frame rate.

For the maximum frame rates that can be obtained with a specific camera model using AOI, please refer to the [Specifications: Sensors](#) chapter.

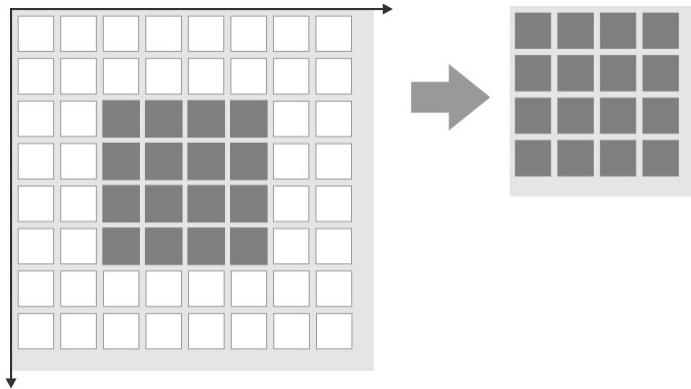


Figure 80: AOI readout on monochrome sensors

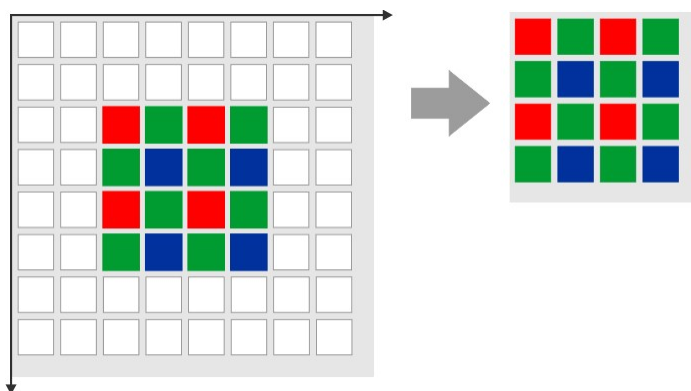


Figure 81: AOI readout on color sensors



Please note that, after defining an AOI, the resulting image may be darker if the camera cannot maintain the originally set exposure time due to the increased frame rate.

4.4.4.2 Subsampling

Subsampling is a technique that skips multiple sensor pixels when reading out image data. This reduces the amount of data to be transferred and enables higher camera frame rates. The captured image has a lower resolution but still the same field of view compared to the full-resolution image. This mode can be used as a fast preview mode for high-resolution cameras.

Color subsampling as performed by most color sensors skips pixels while maintaining colors (see illustration). For some monochrome sensors, the camera also performs color subsampling, resulting in slight artifacts.

Monochrome sensors and some color sensors ignore the Bayer pattern and the color information gets lost (mono subsampling).

Depending on the model, *uEye* cameras support different subsampling factors. Subsampling of horizontal and vertical pixels can be enabled independently.

The [Specifications: Sensors](#) chapter lists the subsampling methods and factors supported by each camera model.

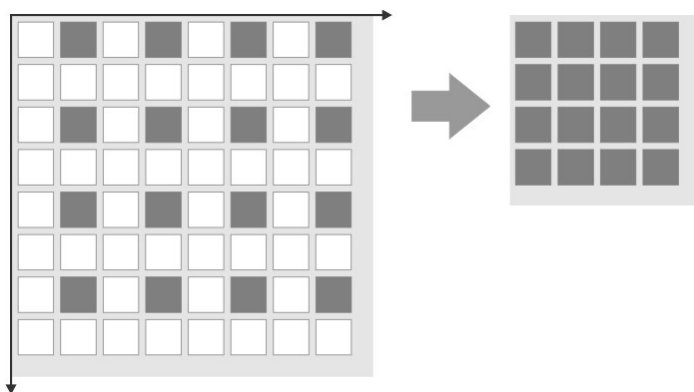


Figure 82: Subsampling on monochrome sensors

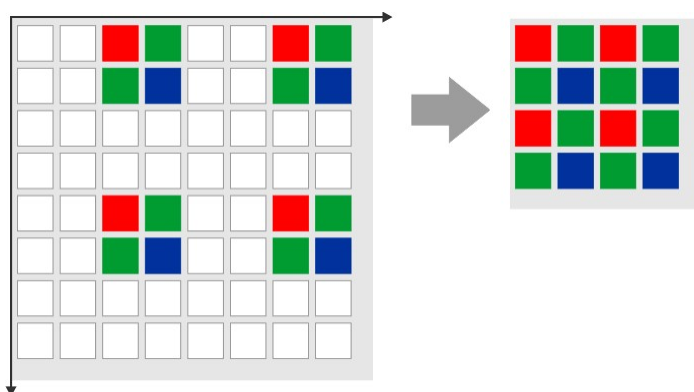


Figure 83: Subsampling on color sensors

4.4.4.3 Binning

Binning is a function that averages or adds multiple sensor pixels to obtain a single value. This reduces the amount of data to be transferred and enables higher camera frame rates. The captured image has a lower resolution but still the same field of view compared to the full-resolution image. This mode can be used as a fast preview mode for high-resolution cameras.

Color binning, as performed by most color sensors, combines only pixels of the same color (see also the [Color Filter \(Bayer Filter\)](#) chapter). For some monochrome sensors, the camera also performs color binning, resulting in slight artifacts.

Most monochrome sensors and some color sensors combine neighboring Bayer pattern pixels; in this case, the color information gets lost (mono binning).

With CCD sensors, binning makes the images brighter because the pixel values are added up. With CMOS sensors, pixel values are usually averaged; this reduces image noise.

Depending on the model, *uEye* cameras support different binning factors. Binning of horizontal and vertical pixels can be enabled independently.

The Specifications: Sensors chapter lists the binning methods and factors the individual camera models support.

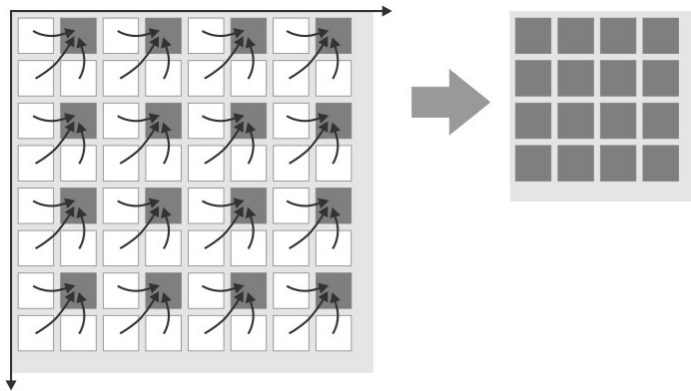


Figure 84: Binning on monochrome sensors

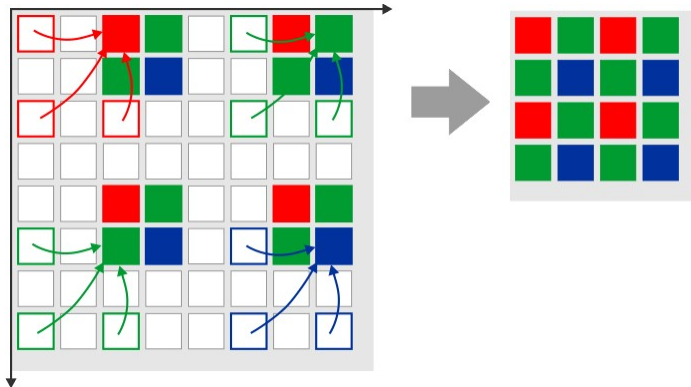


Figure 85: Binning on color sensors

4.4.5 Camera Parameters

4.4.5.1 Pixel Clock, Frame Rate, Exposure Time

Pixel clock

The basic parameter for camera timing is the pixel clock. It determines the speed at which the sensor cells can be read out.



We recommend not setting the pixel clock any higher than necessary to achieve the desired frame rate.

An excessive pixel clock can cause transmission errors or delays. If the data is read from the sensor at a higher speed (high pixel clock), you will also need a faster transmission over the data connection. Thus, by controlling the pixel clock, you can also influence the bandwidth required for a camera.

The pixel clock influences the connected load and consequently the temperature inside the camera.

Frame rate

The possible range of settings for the frame rate depends on the currently selected pixel clock. You can select a lower frame rate without changing the pixel clock. To set a higher frame rate, however, you need to increase the pixel clock.

Exposure time

The exposure time depends on the currently selected frame rate and is preset to its reciprocal value. You can select a shorter exposure time without changing the frame rate. To set a longer exposure time, however, you need to reduce the frame rate.

4.4.5.2 Gain and Offset

Gain

In digital imaging, a voltage proportional to the amount of incident light is output by the sensor. To increase image brightness and contrast, this signal can be amplified by an analog gain and offset before the digitizing process. The results of analog signal processing are usually better than the results of digital post-processing.

Analog amplification of the read-out pixel values increases overall image brightness and contrast. Depending on the sensor type, a global gain value for all pixels (*master gain*) or a separate gain value for each color (*RGB gain*) can be set.



A signal gain will also result in a noise gain. High gain settings are therefore not recommended. We suggest the following gain settings:

1. Enable the *Gain boost function*.
2. If required, adjust the gain setting with the master gain control.

Offset

Every digital image sensor has light-insensitive cells next to the active image area. These dark pixels are used to measure a reference voltage (*black level*) which is subtracted from the image signal. This compensates thermally generated voltages on the sensor which would otherwise falsify the signals.

Normally, the sensor adjusts the black level automatically. If the environment is very bright or if exposure times are very long, it may be necessary to adjust the black level manually.

4.4.5.3 Automatic Image Control

The uEye driver provides various options to automatically adjust the image capture parameters to the lighting situation. These include:

- Auto Exposure Shutter (AES)
- Auto Gain Control (AGC)
- Auto White Balance (AWB)
- Auto Frame Rate (AFR)

The auto functions are used to adjust the average brightness and color rendering of the camera image to their setpoint values, while trying to keep the frame rate at the highest possible value.

All controls are configured using the `is_SetAutoParameter()` SDK function.

Auto Exposure Shutter (AES)

The control of the average brightness is preferably achieved by adjusting the exposure, i.e. you set the highest possible exposure time before gain is controlled. Auto Exposure always uses the current exposure range which results from the selected pixel clock frequency and the frame rate. You can set separate control range limits for exposure and gain.

Auto Gain Control (AGC)

The auto gain feature controls the camera master gain in a range from 0-100%. You can set separate control range limits for exposure and gain.

Auto Frame Rate (AFR)

With the exposure control function enabled, you can still change the frame rate manually or automatically to maintain a dynamic exposure control range. A lower frame rate allows for longer exposure times, but then the live image display may exhibit jitter. The objective of the automatic frame rate control is to set the frame rate to an optimum value. This way, in all situations, the automatic exposure control can use the required control range at the highest possible frame rate.

Auto White Balance (AWB)

Depending on the lighting source, light can have different color temperatures so that the images may have a color cast. At low color temperatures (e.g. light from incandescent lamps), the white content is offset towards a red hue. At high color temperatures (e.g. light from fluorescent lamps), the white content is offset towards a blue hue.

The white balance control feature uses the RGB gain settings of the camera to correct the white level. This is achieved by adjusting the gain controls within the 0-100% range until the red or blue channel matches the average brightness of the green channel. In order to manually influence the color rendering, you can adjust the setpoint values for the red and blue channels relative to the green channel by using an offset value.

Automatically Disabling the Control Function

You can disable the control functionality automatically once the target value has been reached (API parameters `IS_SET_AUTO_WB_ONCE` and `IS_SET_AUTO_BRIGHTNESS_ONCE`). An event / a message notifies the system of this (see also `is_InitEvent()`). Alternatively, you can keep the control feature enabled so that it responds to deviations from the target value.

Control Speed

You can set the auto function speeds in a 0-100% range. This influences the control step width. High speed (100%) causes a little attenuation of a fast-responding control and vice versa. The control functions for average brightness and for color rendering use separate speeds.

In trigger mode, every frame is evaluated for automatic control. The freerun mode skips a number of frames by default because in that mode, changes to the image parameters only become effective after one or more image captures (see also [Applying New Parameters](#)). With the *Skip Frames* parameter (API parameter `IS_SET_AUTO_SKIPFRAMES`), you can select how many frames should be skipped in freerun mode (default: 4). This parameter strongly influences the control speed. Choosing small values can destabilize the automatic control.

Hysteresis

The automatic control feature uses a hysteresis function for stabilization. When the actual value is in the range of (setpoint - hysteresis value) up to (setpoint + hysteresis value), control is temporarily disabled. It is reactivated when the actual value drops below (setpoint - hysteresis value) or exceeds (setpoint + hysteresis value). If the hysteresis value is increased, the achieved target value is maintained for a longer time in case of lighting changes. This makes the automatic control more sluggish, but can be useful in some situations.

4.4.5.4 Applying New Parameters

New capture parameters (such as exposure time or gain settings) can be transferred to the camera via software at any time. Depending on the operating mode, these settings will not always be immediately effective for next image, however.

- Freerun mode
In freerun mode, the camera is internally busy with capturing the next image while new parameters are transmitted to the camera. Depending on the exact time of transmission, new parameters might only come into effect two or even three images later.
- Trigger mode
In this mode, the camera reverts to idle state between two images. When you change the camera parameters, the new settings will be applied immediately to the next image (delayed by one additional image for the UI-122x-C/M or UI-522x-C/M cameras due to the sensor).

4.4.6 Firmware and Camera Start

Every *uEye* camera has its own *firmware* that handles internal processes in the camera. The camera firmware varies from model to model.

USB uEye cameras

USB uEye cameras have a two-tier firmware that is uploaded to the camera each to you connect it to a PC:

1. Common firmware (*uEye boot*)
The general firmware identifies what camera model you have connected, and uploads the corresponding firmware.
2. Model-specific firmware (e.g.: *uEye UI-154x series*)
The model-specific firmware is named after the camera type and provides the functions of the relevant model.



When you connect a *USB uEye* with a Windows PC or a new USB port for the first time, it is detected as a new device. This is normal standard behavior of the operating system.

The *USB uEye* cameras firmware is part of the driver. The automatic upload always loads the firmware that matches the driver installed in the camera.

GigE uEye HE cameras

With *GigE uEye HE* cameras, the firmware consists of three components. As opposed to *USB uEye* cameras, the first two firmware components are stored in the camera's non-volatile EEPROM.

1. Starter firmware

When the *GigE uEye* camera has been connected to the power supply and the network, it loads the starter firmware stored in the camera's EEPROM. The starter firmware enables the camera to register on the network and establish a connection to a host PC. As soon as the starter firmware is loaded, the camera sends a heartbeat broadcast to the network once every second.

In the next step, the camera checks whether a persistent IP address is stored. If it is, the camera uses the persistent IP address. Otherwise the camera is assigned the IP address 0.0.0.0.



If a new *uEye* driver version requires a new starter firmware, you can update it from the *uEye Camera Manager*.

2. Failsafe firmware

If the starter firmware cannot be found or is corrupted, the failsafe firmware is loaded from a write-protected memory area in the *GigE uEye*. The failsafe firmware allows starting the camera so that an executable starter firmware can be uploaded.

3. Runtime firmware

The runtime firmware is the camera's actual operating software. The runtime firmware is not uploaded to the camera until it is paired with a PC (see also Pairing). This ensures that the currently installed driver and the runtime firmware are always compatible. After the IP address has been assigned, the host PC transfers the runtime firmware to the camera and the camera is restarted. The time it takes to transfer the runtime firmware and restart the camera depends on the camera model and might take a few seconds.

GigE uEye SE / RE cameras

The starter firmware for *GigE uEye SE / RE* cameras also contains the runtime firmware. If the camera firmware version is incompatible with the driver version on the PC, the appropriate firmware version will be loaded into the camera the first time the *GigE uEye SE* is paired.



If you have initialized the camera with a driver version earlier than 3.40, you need to update the camera firmware manually using the *uEye Camera Manager*. Driver versions 3.40 and higher support automatic firmware updates.



Updating the *GigE uEye SE* firmware can take up to 20 seconds. It is important not to disconnect the camera from the PC or power supply during this time. Otherwise, malfunctions could occur in the camera.

The firmware programming status is indicated by the uEye Camera Manager and by the camera's status LED.

4.4.6.1 Establishing a Connection

If a computer wants to use a camera, it sends a connection request to that camera. When the camera has signaled that it is available, the system first checks whether the camera has a valid IP address. If it does not, the computer sends a range of valid IP addresses to the camera. The camera picks a free IP address, i.e. one that is not yet in use on the network, from these IP addresses and notifies the computer that the IP address has been assigned.

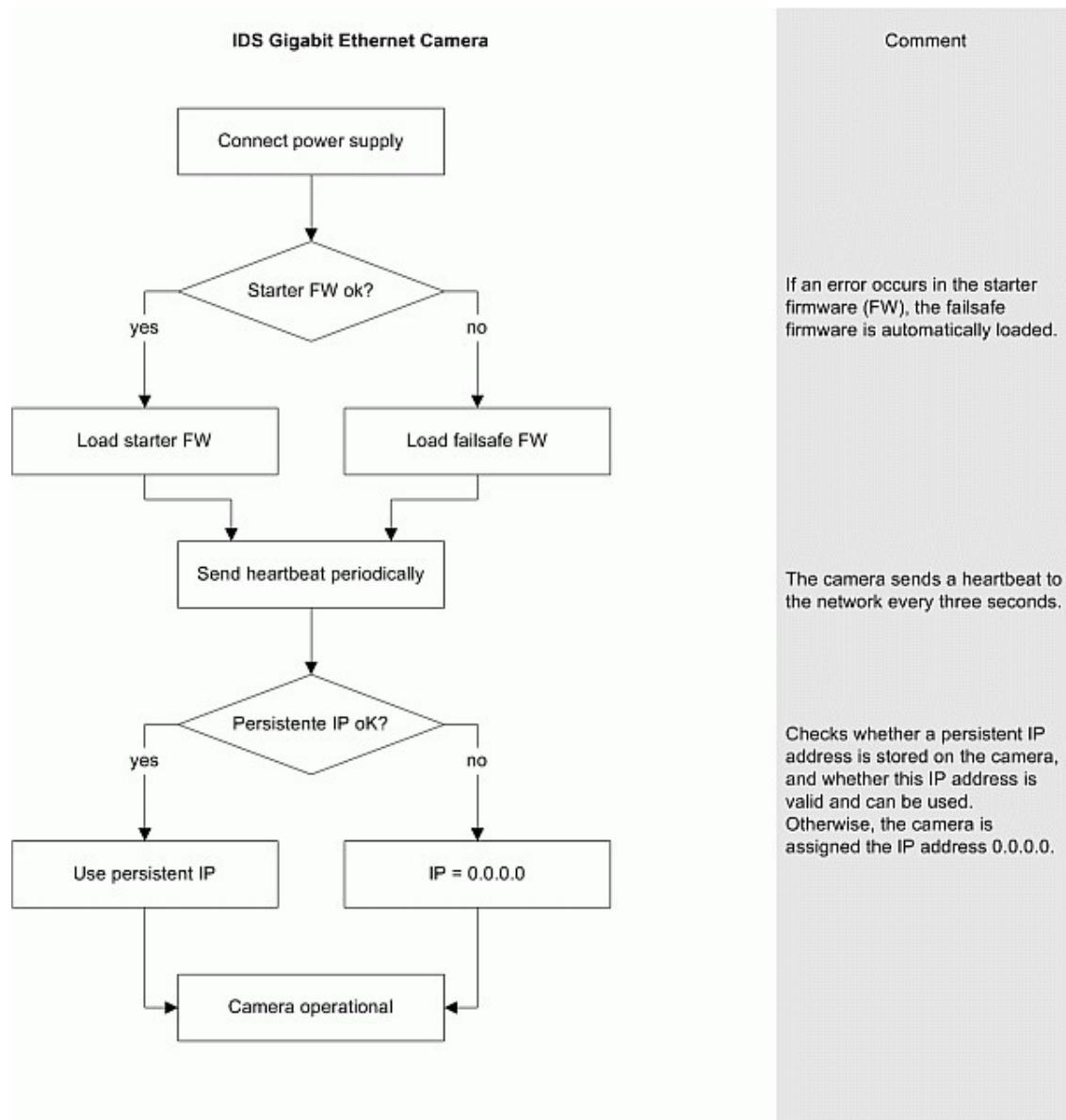
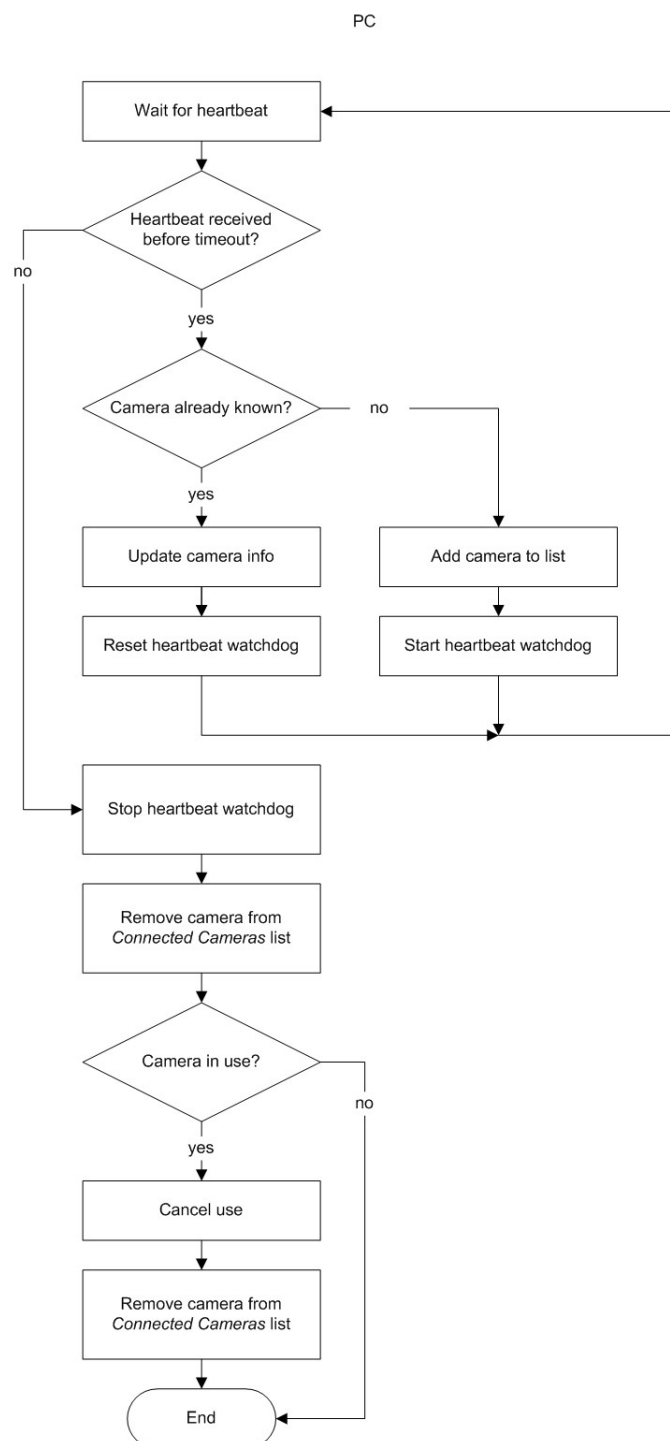


Figure 86: GigE uEye Camera Start-up - Flow Chart

Connection termination

As soon as the camera is connected to a host PC, it cannot connect to any other computer on the network. When a connection has been successfully established, the host PC also transmits a heartbeat, which is received and evaluated by the camera. If the heartbeat of the host PC cuts out, the camera is restarted and the starter firmware loaded. This allows the camera to connect to another computer on the network.

If the heartbeat of the camera cuts out, the host PC closes the connection and removes the camera from its camera list. Once this has taken place, the camera is no longer displayed in the Camera Manager. This process may take up to three heartbeat periods (see above).



Comment

The camera periodically sends a heartbeat signal to the network.

The PC checks whether it has received a heartbeat signal from the camera within a definable period (default: 3 seconds).

As soon as it receives a heartbeat signal, the PC checks whether signals have previously been received from this camera. If so, the camera data in the camera list are updated. Otherwise, the camera is added to the camera list and the heartbeat watchdog is started.

If the camera does not send a signal within the specified wait time (timeout), communication with this camera is cancelled and its entries in the camera lists are deleted.

Figure 87: Flowchart of camera recognition for the GigE uEye HE

4.4.7 Pixel Preprocessing in GigE uEye Cameras

Cameras of the *GigE uEye* series use an integrated FPGA processor for pixel preprocessing. The following flowcharts illustrate the sequence of preprocessing operations in the *GigE uEye SE* and *GigE uEye RE* as well as *GigE uEye HE* cameras.

Operations marked with an asterisk * are optional and can be selected in the software depending on the camera configuration.

4.4.7.1 GigE uEye SE / RE

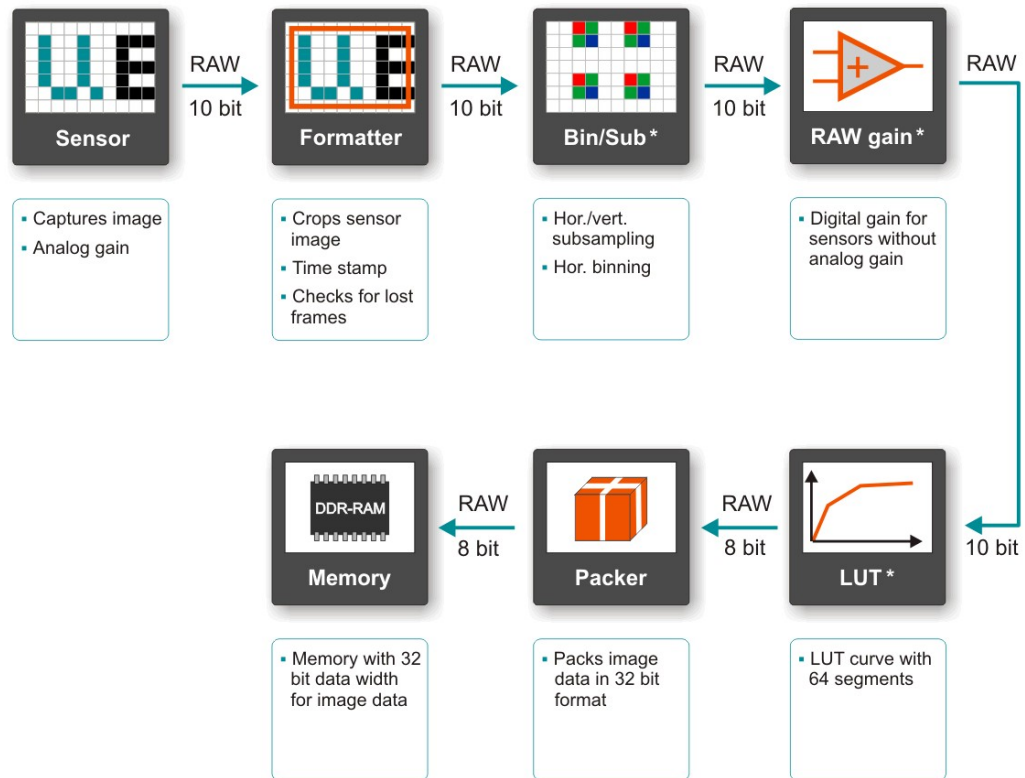


Figure 88: Pixel preprocessing in GigE uEye SE / RE cameras

4.4.7.2 GigE uEye HE

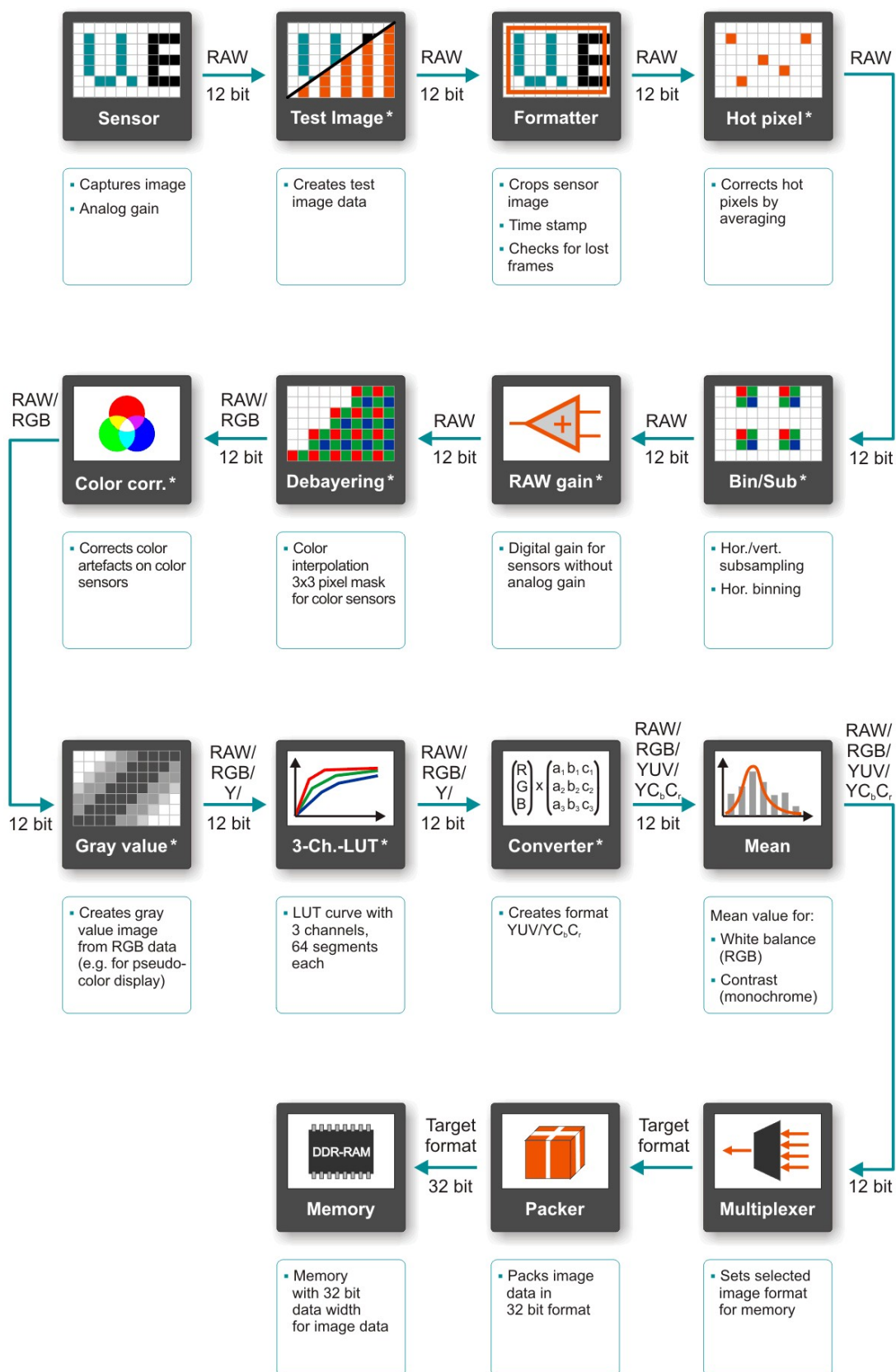


Figure 89: Pixel preprocessing in GigE uEye HE cameras

4.4.8 Digital Inputs/Outputs

4.4.8.1 Digital Input (Trigger)

In trigger mode, image capture by the *uEye* cameras can be controlled through external events. For this purpose, a digital signal must be applied to the camera's digital input.

You can determine whether the camera will respond to the rising or falling edge of the digital signal. After an internal delay, the sensor is exposed for the defined exposure time. The captured image is then transferred to the PC.



The delay is due to internal camera switching times and depends on the sensor type and the parameters that have been set. It is always below 100 μ s. You can find the exact values for each camera in the [Specifications: Sensors](#) chapter.

You can optionally set an additional delay (*trigger delay*).

In case of a triggered image capture, the camera is only ready to process the next trigger signal after completion of the data transfer to the PC. Trigger events that occur during image exposure or data transfer are ignored. An internal counter records the number of ignored trigger events and can be read out from the PC.

You can query the status of the digital input using the software. This enables you to use the input for other purposes as well.

4.4.8.2 Digital Output (Flash Strobe)

The digital output can be set statically by software or depending on the exposure time.

In *uEye* models equipped with an opto-coupler output, it is possible to control a DC voltage which is applied to the output. This allows controlling a flash, either directly or via a separate flash controller unit depending on the sensor exposure. In exposure-dependent mode, you can set the *delay* and the *duration* of the flash. By selecting suitable delay and duration settings, you can minimize the rolling shutter effect (see also [Shutter Methods](#)).

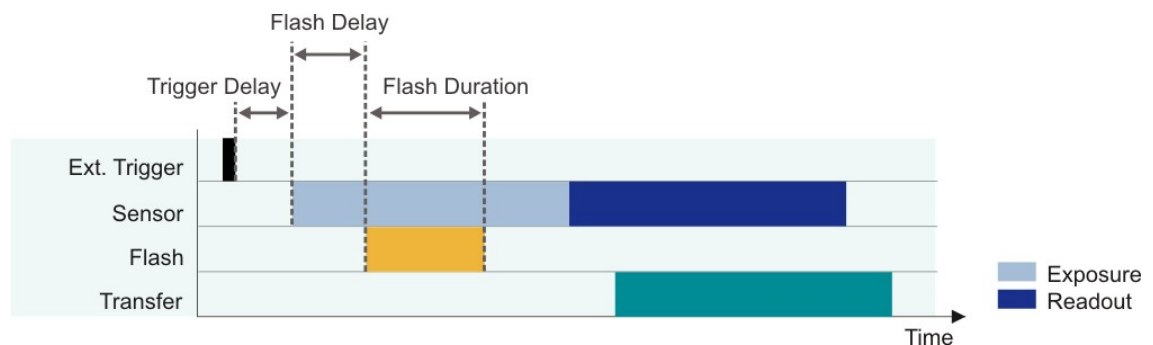


Figure 90: Triggered image capture and flash with delay and duration



The delay is due to internal camera switching times and depends on the sensor type and the parameters that have been set. It is always below 100 μ s. You can find the exact values for each camera in the [Specifications: Sensors](#) chapter. You can set an additional delay and the duration of the flash signal in the software.



The settings specified for the digital output will be reset in the following situations:

- a *GigE uEye* camera loses its pairing (i.e. it is closed in the software)
- a *USB uEye* camera is disconnected from the PC or the PC is powered down

4.4.8.3 General Purpose I/O

Some *uEye* models have freely programmable General Purpose Inputs/Outputs (GPIO) which can be programmed as inputs or outputs using the *uEye SDK* (see also GPIO Specifications).



The GPIO settings will be reset when the camera is closed in the software.

4.4.8.4 Serial Interface (RS232) of the GigE uEye HE

Cameras of the *GigE uEye HE* series are equipped with a serial interface (RS232). It provides functionality for communication with peripheral devices (e.g. lighting controller, lens controller) or the serial port of a PC. Before you can send data through the camera's serial interface, one or more virtual COM ports have to be defined on the PC. Once defined, they can be used for data communication with appropriate software just like any physical COM port.

To set up and use the serial interface, the *Additional functions dialog box* is provided in the *uEye Camera Manager*. For the serial interface specifications, please refer to the Serial Interface (RS232) chapter.

4.4.9 USB Basics

4.4.9.1 History and Development

The *Universal Serial Bus* (USB) is an interface which enables you to easily connect various devices to a PC. As all data exchange is controlled by the PC, no additional interface controller is needed. Further advantages of USB are:

- the PC does not have to be shut down when connecting USB devices (*hot plugging*)
- USB devices can be supplied with power from the PC
- High bandwidth for data transmission

The USB standard was developed by a group of companies including Compaq, IBM, Intel, and Microsoft. Version 1.0 was presented in 1995. The slightly faster USB 1.1 standard followed in 1998.

At first, the USB interface was designed to connect peripheral devices such as printers, mice, or keyboards. With the introduction of USB 2.0 in 2000, the transfer rate increased to 480 Mbps, making USB 2.0 suitable for connecting devices with higher data volumes (such as mass storage devices, scanners, or cameras).

4.4.9.2 Structure and Topology

USB uses a tree topology and is host-controlled. That means that a PC with host functionality is mandatory for using USB. Therefore, it is not possible to directly connect two USB devices (with the exception of USB On-the-go compliant devices). Neither is it possible to connect a camera to a PDA device.

Theoretically, 127 devices can be connected to a host controller. Using external hubs or repeaters, even more devices can be connected, and from a greater distance. Provided that a maximum of 5 hubs/repeaters may be daisy-chained, USB devices can be connected in up to seven levels.

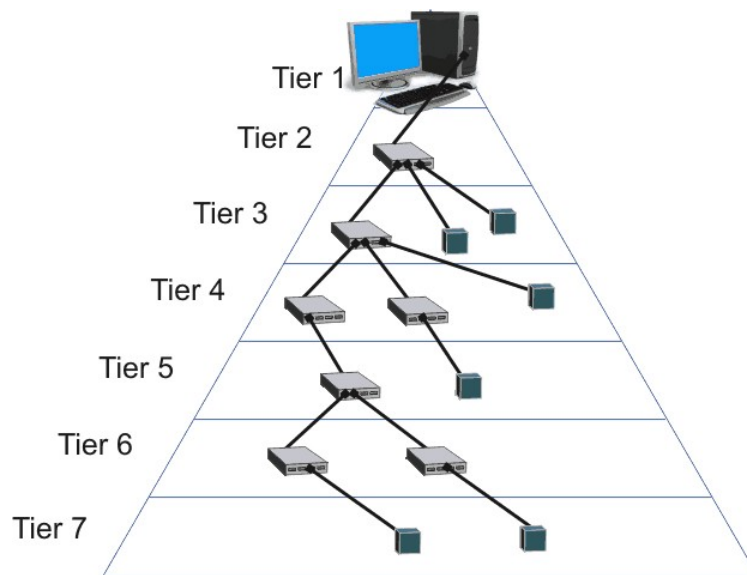


Figure 91: USB 2.0 Topology



The maximum bandwidth of 480 Mbps per USB 2.0 host cannot be exceeded. Therefore, the maximum possible frame rate will be reduced if image data from multiple USB cameras is transferred simultaneously.

The available bandwidth might also be decreased when you use hubs or repeaters. You can reduce the bandwidth required for each camera by lowering the frame rate or the image size.

4.4.9.3 Cabling and Connection

In order to comply with the specifications, the maximum length of USB 2.0 cables is limited to 5 m. Longer cables may be connected if you use high-quality material. For cameras of the *USB uEye RE* series, IDS offers cables with a length of up to 10 m (see also [USB uEye RE Accessories](#)).

The USB bus provides power supply with 5 V and 500 mA max. Many USB devices use the bus power and do not need external power supply (*bus-powered* devices).

Cable design

The following illustration shows the basic design of a shielded USB cable:

- D+/D-: data transfer
- +5 V/GND: power supply

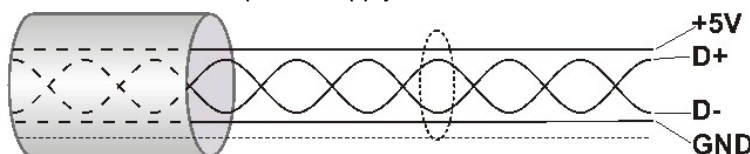


Figure 92: Basic design of a USB cable

Connector types

On the PC side, USB 2.0 cables are equipped with a standard A type plug (four pins) and on the device side either with a standard B plug (four pins) or a mini-B plug (five pins). In addition, some *USB uEye* camera models feature non-standard connectors which are lockable or splash water proof.



Figure 93: USB standard-A socket (four pins)

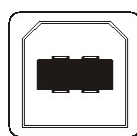


Figure 94: USB standard-B socket (four pins)

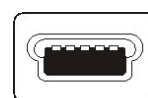


Figure 95: USB mini-B socket (five pins)

4.4.9.4 Data Transmission and Bandwidth

The USB 2.0 standard specifies an overall bandwidth of 480 Mbps shared between different transmission modes. *uEye* cameras use the USB 2.0 *bulk* mode for transmitting images. This mode uses error correction to ensure correct delivery of the image data, but does not guarantee a fixed bandwidth. To ensure error-free communication with all connected devices at all times, the maximum bandwidth for payload data is limited to 416 Mbps.

Theoretically, up to 50 MB/s of data can be transmitted in this mode, but in practice, this value is hardly ever reached. A high-performance desktop PC can transmit about 40 MB/s, most notebooks or embedded PC systems even less than that.

The overall bandwidth can be increased by the use of USB 2.0 expansion cards. These cards are available for the PCI and PCIe buses and have their own host controller chip.



To achieve optimum USB bandwidth, it is important to use a powerful mainboard chipset. The mainboard chipsets from e.g. Intel® or NVIDIA® provide very good results.

If you need recommendations on the most appropriate hardware to use, please contact [uEye Support](#).

4.4.10 GigE Basics

4.4.10.1 General

Gigabit Ethernet was developed on the basis of the Fast Ethernet (100 Mbps) standard. In June 1999, the IEEE 802.3ab 1000 Mbps standard was defined by the *IEEE* (Institute of Electrical and Electronics Engineers). Using at least Cat 5e copper cables, transmission rates of 1 Gbps can be obtained. This makes Gigabit Ethernet 10 times faster than Fast Ethernet. The main advantages of Gigabit Ethernet include:

- Higher bandwidth, allowing for better network performance and the elimination of bottlenecks
- Full-duplex capability virtually doubles the effective bandwidth
- Low purchasing and operating costs through the use of common hardware
- Full compatibility with the large number of installed Ethernet and Fast Ethernet nodes
- Fast transfer of large amounts of data over the network



Figure 96: Structure of a Cat 5e cable

For connecting Gigabit Ethernet cables, RJ45 connectors are used. The following illustrations show schematic views of an RJ45 socket (with cable configuration) and of an RJ45 plug.

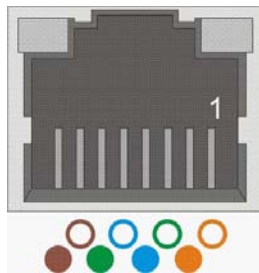


Figure 97: RJ45 socket
(EU type acc. to
EIA/TIA-568B)

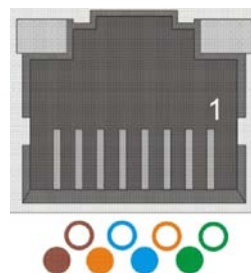


Figure 98: RJ45
socket (US type acc. to
EIA/TIA-568A)

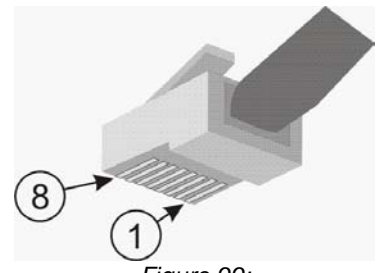


Figure 99:
RJ45 plug



The *GigE uEye* camera automatically recognizes whether an Ethernet cable with crossed wiring or straight wiring is connected. The camera adjusts accordingly.

4.4.10.2 Glossary

UDP

UDP stands for *User Datagram Protocol* and contains mechanisms that allow applications to easily send messages to each other. UDP is session-oriented and has no protective measures to guard against message loss or duplication. The header contains the sender port, the recipient port, the length of the datagram and a checksum.

Port

Ports are address components used in network protocols to assign data segments to the correct services (protocols).

Firewall

A firewall is a software or hardware shield that protects a local network or a computer from Internet-based attacks.

Among a firewall's main uses are protection from hacker attacks, computer viruses, trojans, worms and spyware.

ARP

The *Address Resolution Protocol* uses network messages, called broadcasts, to determine on which logical segment of the network the recipient of a packet is located.

The responses to the broadcast contain all the IP addresses of the available subnet and the associated MAC addresses. Every IP address is stored in an ARP table together with the associated MAC address. These tables are necessary because the two addresses are independent of each other and therefore cannot be calculated by means of an algorithm.

Subnet

Subnets are small units of a network. Using subnets makes it easier to manage networks and increases performance, as connecting devices such as routers or switches can be used to limit data traffic to specific subnets.

The address is made up of the IP address of the network, the subnet address and the host address.

Switch

The term *switch* refers to the connecting units in a *LAN (Local Area Network)*. They are used to connect subnets of the same topology. Contrary to hubs, switches dispatch incoming data packets only to the specific recipients.

Router

Routers are connection units that connect different networks or LANs.

Hub

A hub is a coupling unit that connects several network units on one line (star topology). Contrary to a switch, the message of a network member is dispatched to all other network members.

DHCP

The *Dynamic Host Configuration Protocol* controls the dynamic configuration of IP addresses. When a workstation which is configured for the use of DHCP is started up on a LAN, it registers with a server running this service. The server then assigns an available IP address, which is stored locally so that reassignment is not necessarily required on the next start-up.

Broadcast

A *broadcast* is a data packet that is transmitted to all stations on a network. This is done by sending a data packet to the reserved IP address .255 of a network or subnet (broadcast address).

Heartbeat

Network devices send a *heartbeat* to signal that they are operational and fully functional. If this heartbeat signal is not detected, the recipient system assumes that the remote device is no longer available.

Paired

Paired describes the logical connection of a network camera and a host PC. When a camera and a host PC are paired, they are exclusively connected. Simultaneous pairing with several host PCs is not possible.

A request for image data is only possible in paired state.

5 B: Programming

5.1 Quick-Start

The *uEye API* offers you over 150 commands with which you can access all the parameters and functions of your *uEye* camera. As complicated as that may sound, it is really quite easy to get your first own *uEye* program up and running in a short time. Just follow the six main steps outlined in this quick start guide.



Open (initialize) the camera

Connect your *uEye* camera with the PC. If you are using a *GigE uEye*, open the *uEye Camera Manager* first and assign an IP address before connecting the camera. See also the [Installing the GigE Camera](#) and [uEye Camera Manager](#) chapters

The `is_InitCamera()` function initializes the *uEye* camera. The camera is assigned a unique handle through which it is accessed in subsequent function calls.



Select a display mode

The *uEye API* provides two different modes you can use to display the camera's images on the PC. To quickly show a live image under Windows, it is easiest to use the *Direct3D* mode. This mode has the advantage that no image memory has to be allocated, and that image capture is handled by the driver. Call `is_SetDisplayMode()` to select the display mode. You can then customize the *Direct3D* mode by using `is_DirectRenderer()`.

For advanced users:

You can also access the image data directly by selecting the *Bitmap (DIB)* mode. To use *DIB* mode, you first have to allocate one or more memories by using `is_AllocImageMem()`, add them to a [memory sequence](#), if required, and then activate a memory with `is_SetImageMem()` before each image capture. To show the image on-screen, call the `is_RenderBitmap()` function after each completed image capture. From the [events or messages](#) you can see when an image is available for display.

Under Linux:

The display functions of the *uEye API* are not available under Linux. You need to allocate and activate the relevant image memory as described above. The application then displays the image data via the Linux function library used.



Capture images

Recording live images with the *uEye* is very simple. Just call the `is_CaptureVideo()` function and the camera captures the live images at the default frame rate. To capture single frames, use the `is_FreezeVideo()` function. Every *uEye* camera of course also provides different trigger modes for image capture. Use `is_SetExternalTrigger()` to activate the desired mode before starting the image capture.



Adjust the frame rate, brightness and colors

All function calls with which you can change camera settings start with `is_Set`. To change the frame rate, for example, you call `is_SetFrameRate()`. Image brightness is adjusted through the exposure time set with `is_SetExposureTime()`. You can also implement automatic control of image brightness and other parameters by using `is_SetAutoParameter()`.

If you are using a color camera, you should activate color correction in order to achieve rich vibrant colors for on-screen display (`is_SetColorCorrection()`). To adapt a color camera to the ambient light conditions, it is essential to carry out white balancing. This is also done using the `is_SetAutoParameter()` function.



Save an image

Use the `is_SaveImageEx()` function to save the current image as a BMP or JPEG file. To save a specific image, it is better to use the Snap function (single frame mode) than the Live function (continuous mode).



Close the camera

When you want to exit your application, close the camera with `is_ExitCamera()`. The camera and the allocated memory are automatically released. All previously set camera parameters will be lost, however. So, if you want to save specific settings, use the `is_SaveParameters()` function before closing the camera. The next time you start the application, you can simply load the settings again by using `is_LoadParameters()`.



You will find comprehensive lists of the API functions, sorted by task, in the How To Proceed chapter.

The *uEye SimpleLive* and *uEye SimpleAcquire* C++ programming samples included in the SDK illustrate the steps described above.

5.2 How To Proceed

This chapter shows function blocks and workflows for important camera functions. The charts are structured as follows:

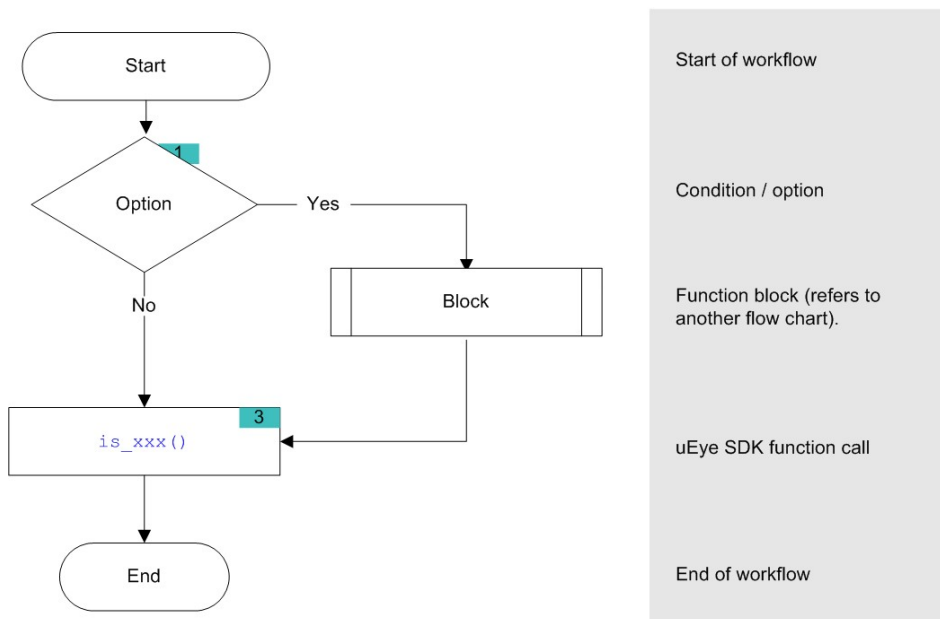


Figure 100: uEye flowchart structure



Click the function names in the flowcharts to open the corresponding function description!

5.2.1 Preparing Image Capture



- **Querying information:** Before you open one or more *uEye* cameras, we recommend querying some key information.
- **Opening and closing a camera**
- **Allocating an image memory:** This is necessary if you want to access image memory contents or if you are not using Direct3D for image display.
- Creating an **image memory sequence** is required when capturing live images.
- **GigE configuration:** These functions for configuring GigE *uEye* cameras are also provided by the *uEye Camera Manager*.

5.2.1.1 Query Information

It is recommended to query the following important information before opening one or more *uEye* cameras.

<code>is_GetNumberOfCameras()</code>	Determines the number of cameras connected to the system.
<code>is_GetCameraList()</code>	Returns information on all connected cameras.

It is also very useful to have the message boxes for error output enabled during the programming process

<code>is_SetErrorReport()</code>	Enables / disables dialogue messages for error output.
----------------------------------	--

With the following functions, you can read out additional information on cameras and software.

<code>is_CameraStatus()</code>	Returns the event counters and other information. Enables standby mode.
<code>is_GetCameraType()</code>	Returns the camera type.
<code>is_GetDLLVersion()</code>	Returns the version of the <i>ueye_api.dll</i> .
<code>is_GetOsVersion()</code>	Returns the operating system version.

5.2.1.2 Open and Close Camera

The following functions are required to open and close a *uEye* camera.

<code>is_InitCamera()</code>	Hardware initialisation
<code>is_ExitCamera()</code>	Closes the camera and releases the created image memory.

When multiple cameras are used on one system you should assign every camera a unique camera ID.

<code>is_SetCameraID()</code>	Sets a new camera ID.
-------------------------------	-----------------------

5.2.1.3 Allocating Image Memory

When you are programming an application that

- requires direct access to the image data in stored in memory, or

- uses Bitmap mode (DIB) for display

use the following functions to allocate and manage image memories (see also [Quick Start: Image Display](#)).

<code>is_AllocImageMem()</code>	Allocates an image memory.
<code>is_SetAllocatedImageMem()</code>	The user provides pre-allocated memory for image capturing.
<code>is_FreeImageMem()</code>	Releases an allocated image memory.

An image memory has to be activated before each image capture:

<code>is_SetImageMem()</code>	Makes an image memory active.
-------------------------------	-------------------------------

To query image memory information and access the data in the image memories, you can use these functions:

<code>is_CopyImageMem()</code>	Copies the image to the user-defined memory.
<code>is_CopyImageMemLines()</code>	Copies selected image lines to the user-defined memory.
<code>is_GetActiveImageMem()</code>	Returns the number and address of the active image memory.
<code>is_GetImageMem()</code>	Returns the pointer to the starting address of the image memory.
<code>is_GetImageMemPitch()</code>	Returns the line offset used in the image memory.
<code>is_InquireImageMem()</code>	Returns the properties of an image memory.



Image memory sequences should be used for frame sequence capture.

Flowchart: Allocating memory

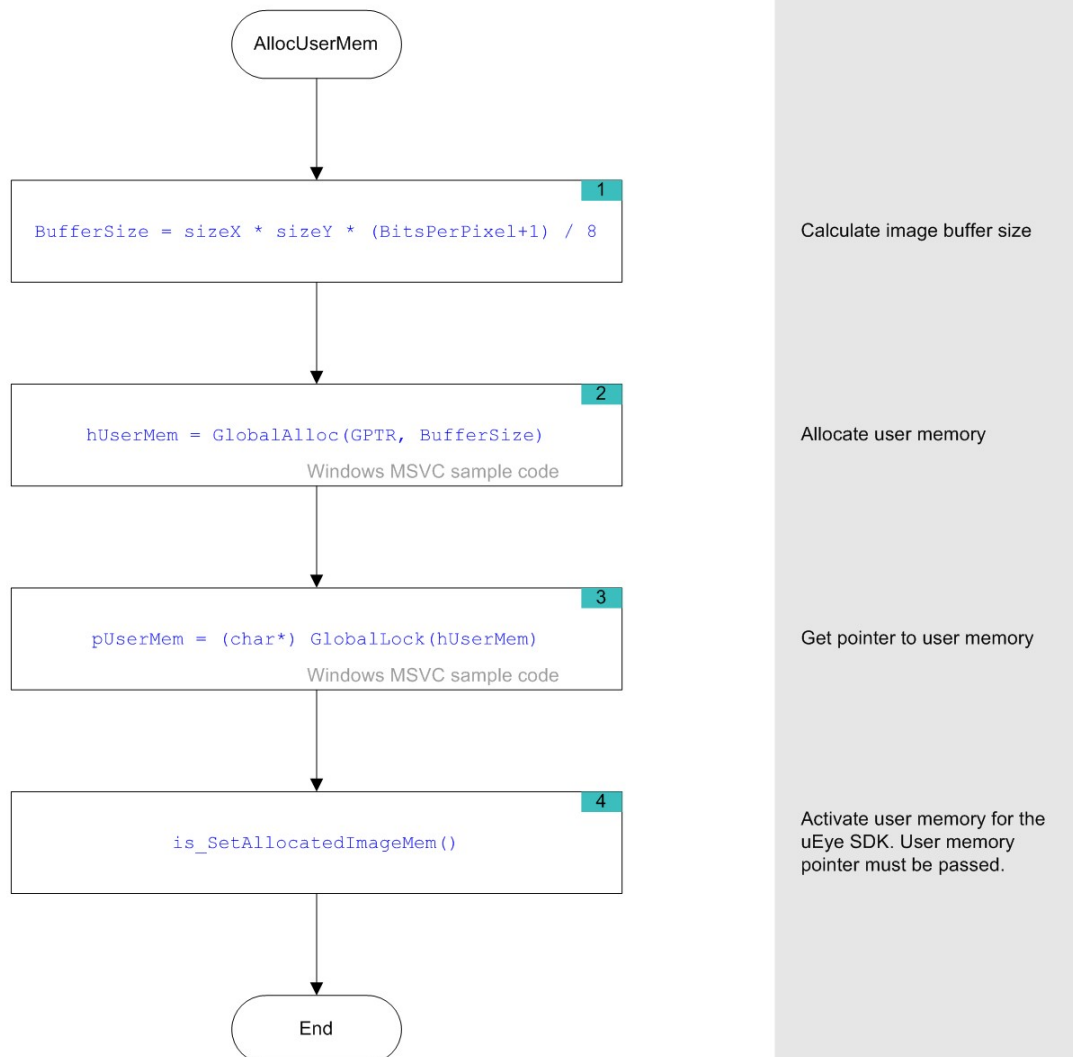


Figure 101: Flowchart - Allocating memory using system functions

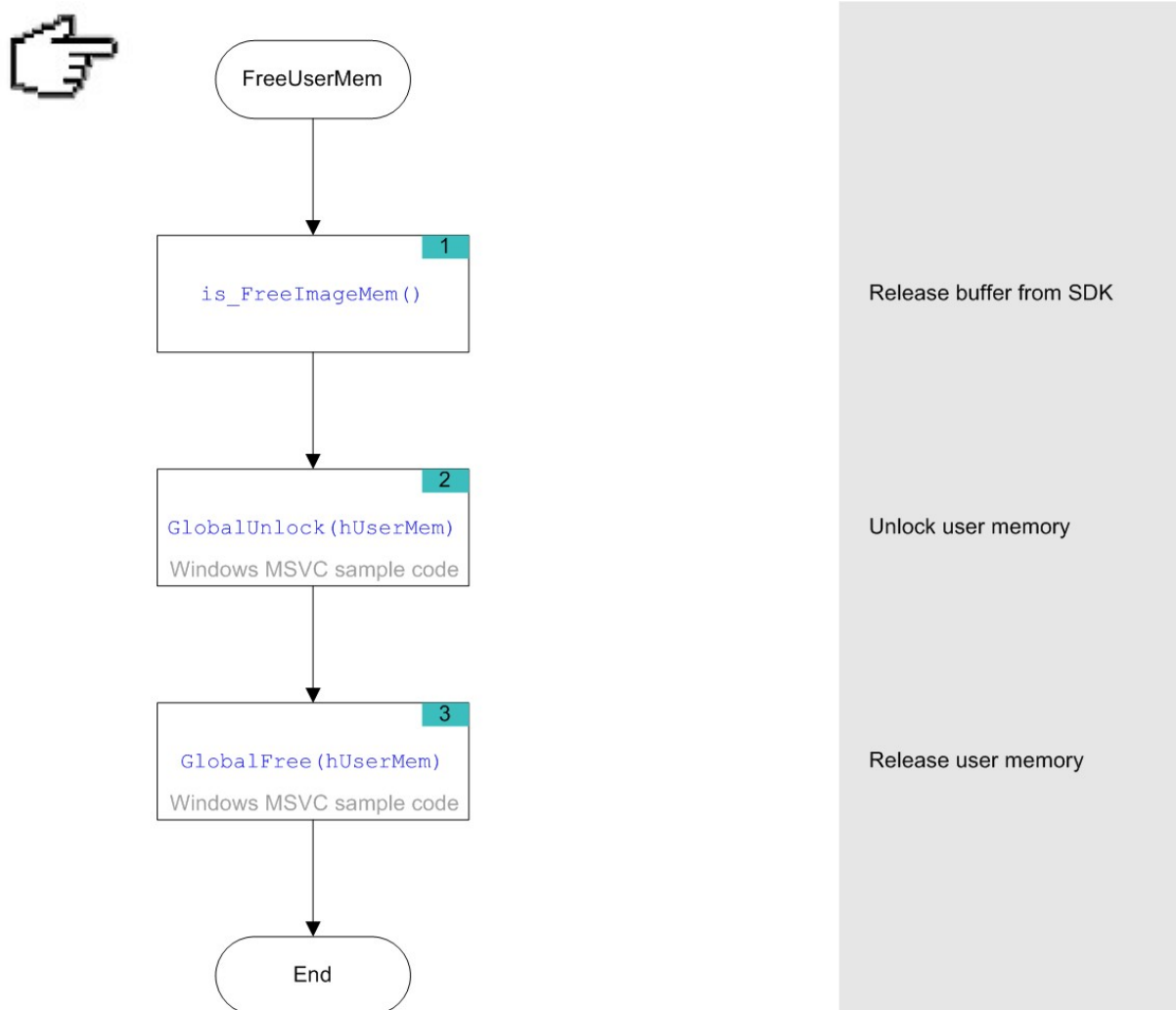


Figure 102: Flowchart - Releasing memory

5.2.1.4 Image Memory Sequences

When you are capturing and displaying frame sequences (e.g. live display), it is advisable to provide appropriate image memory sequences. The *uEye* driver offers a set of easy-to-use features for this purpose. For example, the system automatically cycles through the specified sequence of image memories and can generate an event when it reaches the end of a sequence cycle.

Before you can use a memory sequence, you have to allocate the relevant image memories (see Allocating an Image Memory).

<u>is_AddToSequence()</u>	Adds image memory to the sequence list.
<u>is_ClearSequence()</u>	Deletes the entire sequence list.
<u>is_GetActSeqBuf()</u>	Determines the image memory currently used for the sequence.
<u>is_SetImageMem()</u>	Makes the indicated image memory the active memory.
<u>is_LockSeqBuf()</u>	Protects the sequence image memory from being overwritten.
<u>is_UnlockSeqBuf()</u>	Releases the sequence image memory for overwriting.

5.2.1.5 GigE Configuration

When you are using a *GigE uEye* camera, the following set of functions allows you to make special settings for your GigE camera. These functions are also provided by the *uEye Camera Manager*.

<code>is_GetEthDeviceInfo()</code>	Queries information on connected cameras.
<code>is_SetAutoCfgIpSetup()</code>	Sets the IP auto configuration properties for a network adapter.
<code>is_SetPacketFilter()</code>	Sets the packet filter for a network adapter.
<code>is_SetPersistentIpCfg()</code>	Sets the persistent IP configuration properties for a connected camera.
<code>is_SetStarterFirmware()</code>	Updates the starter firmware of a connected camera.

5.2.2 Display Mode Selection



The *uEye* driver provides different modes for displaying the captured images on Windows systems. We recommend using the Bitmap mode or the Direct3D functions, depending on your specific application.

For further information on the different display modes, see [Basics: Image Display Modes](#).

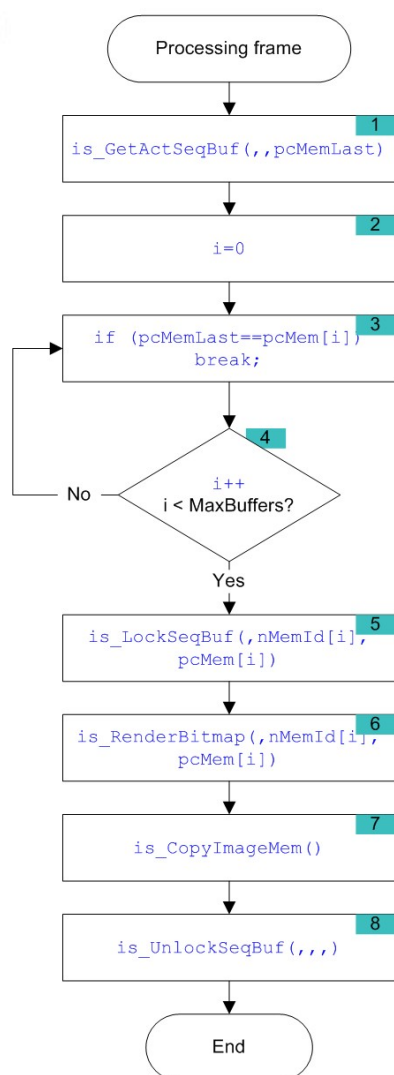
Select the desired mode. The display mode has to be set before you start image capture.

<code>is_SetDisplayMode()</code>	Selection of the display mode.
----------------------------------	--------------------------------

When Bitmap mode (DIB) is active, image display has to be called explicitly for each image.

<code>is_RenderBitmap()</code>	Outputs the contents of the active image memory to a window.
<code>is_SetDisplayPos()</code>	Enables offsetting the image output inside the window.

Flowchart: Image display in DIB mode



Find the buffer ID according to the pointer.

Lock buffer

Perform processing / display.

Optional: Copy image buffer to the application memory for processing

Unlock the buffer

Figure 103: Flowchart - Displaying images in Bitmap mode (DIB)

5.2.3 Image Capture



- *uEye* cameras support the capture of single frames (snap) and frame sequences (live) in trigger mode and untriggered (freerun) mode. Select the image capture mode that best meets your application requirements.
- Using events or messages, the *uEye* driver can provide information to an application, e.g. about the completion of image capture. You will need events and messages, for example, when you are using image memory sequences.

5.2.3.1 Image Capture Modes

For more information on the *uEye* capture modes see also Camera Basics: Freerun and Camera Basics: Trigger.

Freerun Mode

In freerun mode, the camera sensor captures one image after another at the set frame rate. Exposure of the current image and readout/transfer of the previous image data are performed simultaneously. This allows the maximum camera frame rate to be achieved. The frame rate and the exposure time can be set separately. The captured images can be transferred one by one or continuously to the PC.

If trigger mode is active, you need to disable it with `is_SetExternalTrigger()` before activating freerun mode.

- *Single frame mode (snap mode)*
When `is_FreezeVideo()` is called, the next image exposed by the sensor is transferred. You cannot use the *uEye* flash outputs in this mode.
- *Continuous mode (live mode)*
When `is_CaptureVideo()` is called, images are captured and transferred continuously. You can use the *uEye* flash outputs.

Trigger Mode

In trigger mode, the sensor is on standby and starts exposing on receipt of a trigger signal. A trigger event can be initiated by a software command (software trigger) or by an electrical signal via the camera's digital input (hardware trigger). For the specifications of the electrical trigger signals, see the Specifications: Electrical Specifications chapter.

The trigger mode is selected using `is_SetExternalTrigger()`.

- *Software trigger mode*
When this mode is enabled, calling `is_FreezeVideo()` immediately triggers the capture of an image and then transfers the image to the PC. If `is_CaptureVideo()` is called, the triggering of image capture and the transfer of images are performed continuously.
- *Hardware trigger mode*
When this mode is enabled, calling `is_FreezeVideo()` makes the camera ready for triggering just once. When the camera receives an electrical trigger signal, one image is captured and transferred.
If you call `is_CaptureVideo()`, the camera is made ready for triggering continuously. An image is captured and transferred each time an electrical trigger signal is received; the camera is then ready for triggering again (recommended procedure).
- *Freerun synchronisation*
In this mode, cameras running in freerun mode (*live mode*, see above) can be synchronized with an external trigger signal. The cameras still remain in freerun mode. The trigger signal stops and restarts the current image capture process. You can use this mode to synchronise multiple cameras that you are operating in the fast live mode. Not all camera models support this mode (see `is_SetExternalTrigger()`).



In trigger mode, the maximum frame rate is lower than in freerun mode because the sensors expose and transfer sequentially. The possible frame rate in trigger mode depends on the exposure time.
Example: At the maximum exposure time, the frame rate is about half as high as in freerun mode; at the minimum exposure time, the frame rate is about the same.

Overview of Image Capture Modes

Image capture	Trigger	Function calls	Allowed flash modes		Frame rate
			Standard	Global Start	
Continuous	Off	<u>is_SetExternalTrigger</u> (OFF) <u>is_CaptureVideo()</u>	X		Freely selectable
	Software	<u>is_SetExternalTrigger</u> (SOFTWARE) <u>is_CaptureVideo()</u>	X	X	Depending on exposure time and trigger delay
	Hardware	<u>is_SetExternalTrigger</u> (e.g. HI_LO) <u>is_CaptureVideo()</u>	X	X	Depending on exposure time and trigger delay
	Freerun sync.	<u>is_SetExternalTrigger</u> (e.g. HI_LO_SYNC) <u>is_CaptureVideo()</u>	X		Freely selectable
Single frame	Off	<u>is_SetExternalTrigger</u> (OFF) <u>is_FreezeVideo()</u>			Freely selectable
	Software	<u>is_SetExternalTrigger</u> (SOFTWARE) <u>is_FreezeVideo()</u>	X	X	Depending on exposure time and trigger delay
	Hardware	<u>is_SetExternalTrigger</u> (e.g. HI_LO) <u>is_FreezeVideo()</u>	X	X	Depending on exposure time and trigger delay

Timeout Values for Image Capture

When you call `is_FreezeVideo()` or `is_CaptureVideo()`, the timeout value for the image capture is determined from the `wait` parameter. If no image arrives within this timeout period, a timeout error message is issued. Under Windows, a dialogue box is displayed if you have enabled error reports (see `is_SetErrorReport()`). Information on the error cause can be queried using `is_GetCaptureErrorInfo()`.

The following table shows the effect of the `wait` parameter depending on the image capture mode:

Parameter Wait	Image capture mode	Function returns	Timeout for 1st image	Timeout for subsequent images ¹⁾
IS_DONT_WAIT	HW trigger	Immediately	API default or user-defined value ³⁾	API default or user-defined value ³⁾
IS_WAIT	HW trigger	When 1st image in memory	API default or user-defined value ³⁾	API default or user-defined value ³⁾
Time t	HW trigger	When 1st image in memory	Time t	API default or user-defined value ³⁾
IS_DONT_WAIT	Freerun/SW trigger	Immediately	Calculated internally by API ²⁾	Calculated internally by API ²⁾
IS_WAIT	Freerun/SW trigger	When 1st image in memory	Calculated internally by API ²⁾	Calculated internally by API ²⁾
Time t	Freerun/SW trigger	When 1st image in memory	Time t	Calculated internally by API ²⁾

¹⁾ Only with continuous image capture using `is_CaptureVideo()`

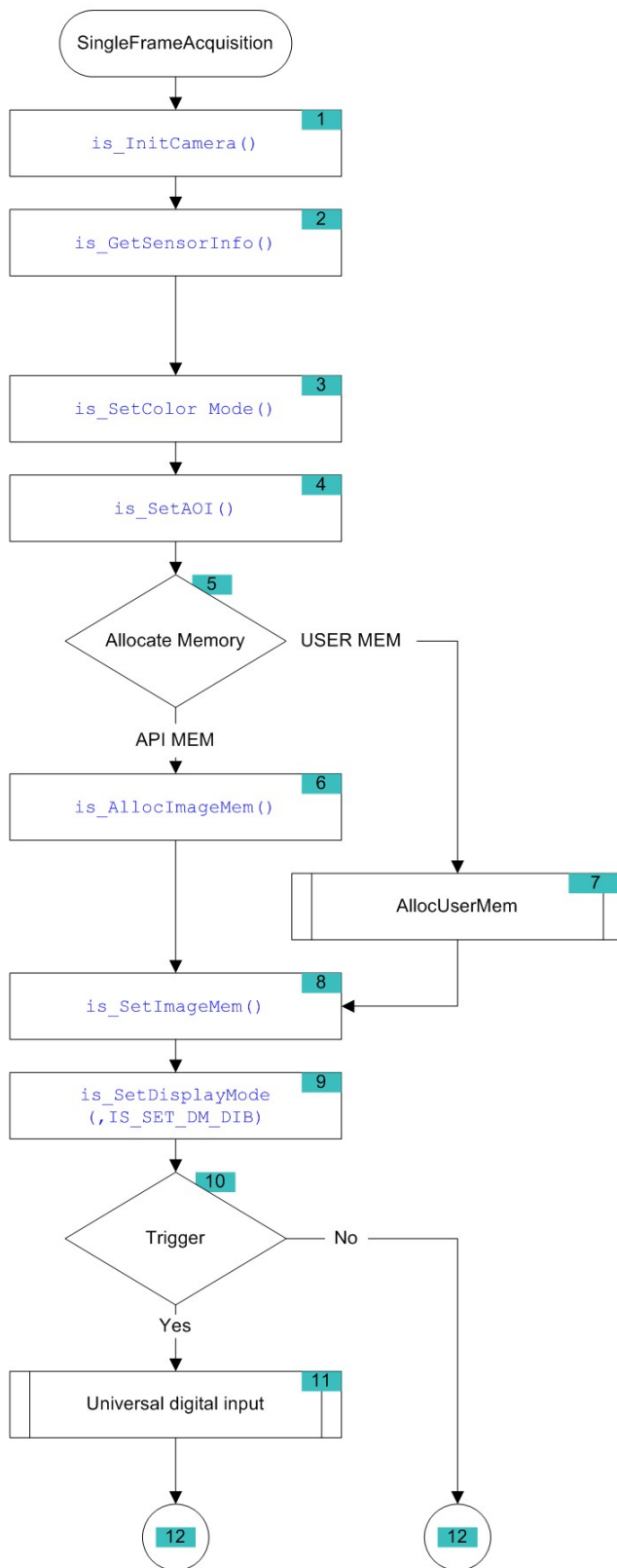
²⁾ The timeout is calculated from the exposure time setting, the image transfer time (depending on the pixel clock) and the optional trigger delay (see `is_SetTriggerDelay()`); it is at least 40 ms.

³⁾ The default value of the *uEye* API is 60 s. User-defined values can be set using the `is_SetTimeout()` function.

Function List

<code>is_CaptureVideo()</code>	Captures a live video.
<code>is_FreezeVideo()</code>	Captures an image and writes it to the active image memory.
<code>is_ForceTrigger()</code>	Simulates a trigger signal in hardware trigger mode.
<code>is_HasVideoStarted()</code>	Returns whether the capture process has been started or not.
<code>is_IsVideoFinish()</code>	Returns whether the capture process has been terminated or not.
<code>is_SetSensorTestImage()</code>	Enables test image output from sensor (all cameras).
<code>is_StopLiveVideo()</code>	Terminates the capturing process (live video or single frame).

Flowchart: Single Capture



Open Camera and get device handle

Get useful information about the sensor and the camera

- Sensor resolution
- Colour format (monochrome, Bayer)
- Rolling/global shutter
- Camera model (e.g. UI-122X-M)

Set color mode for the acquisition
(e.g. IS_SET_CM_Y8, IS_SET_CM_RGB24)

Set image size or AOI

Memory allocation by driver: continue with 6
Memory allocation by user: continue with 7

Set Display mode to DIB

Figure 104: Flowchart - Capturing a single frame (1 of 2)

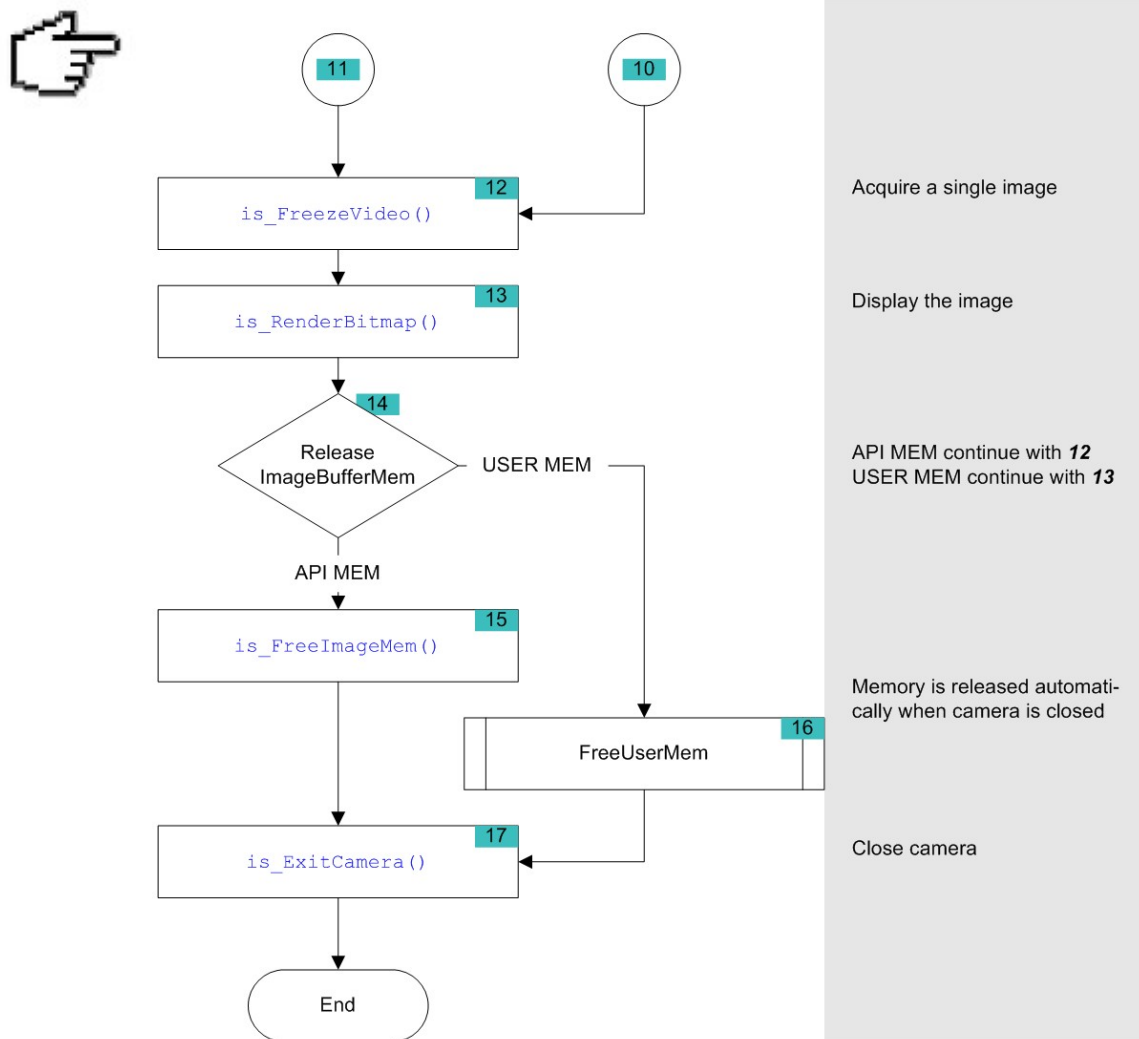


Figure 105: Flowchart - Capturing a single frame (2 of 2)

Flowchart: Sequence Capture

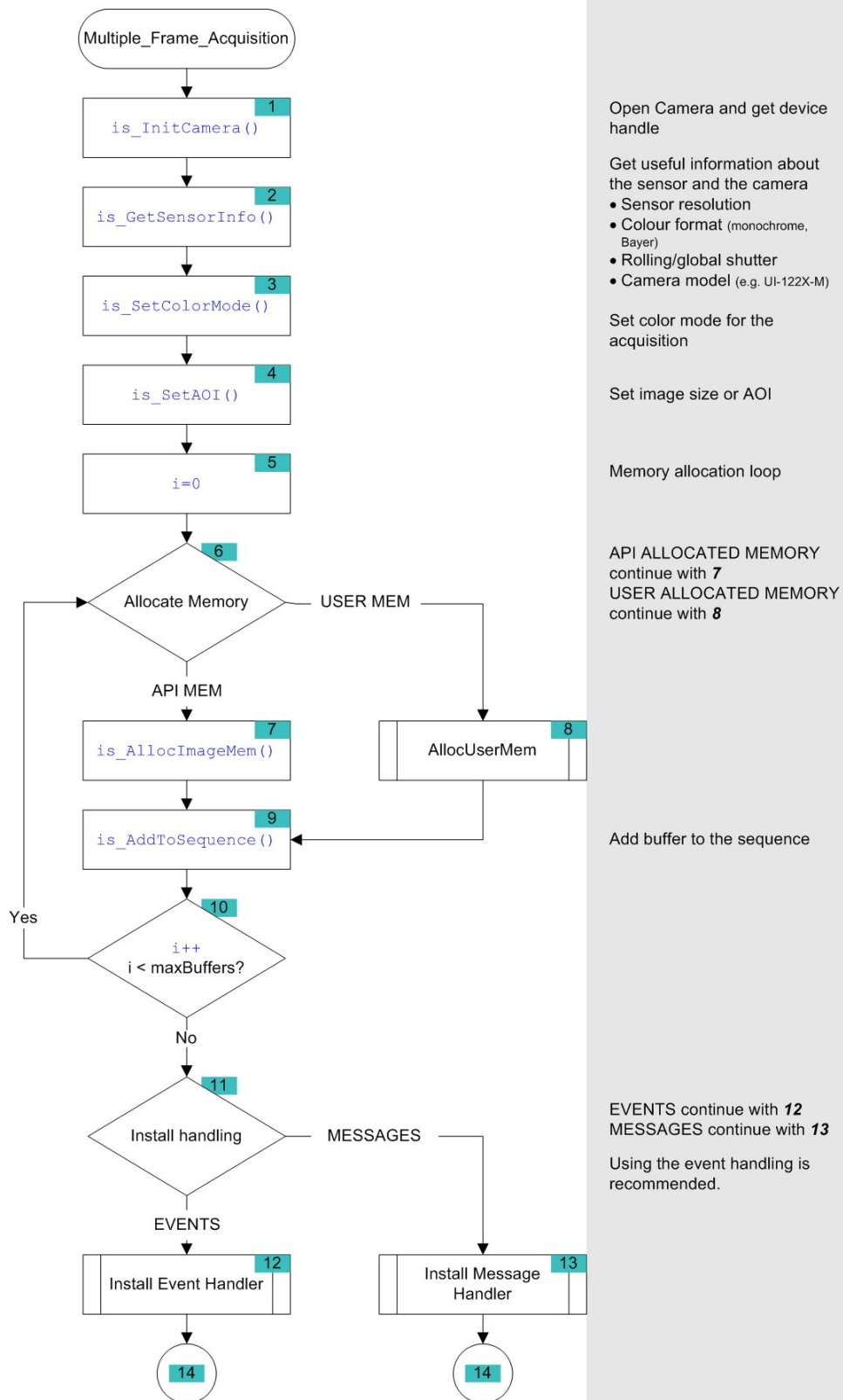


Figure 106: Flowchart - Capturing a frame sequence (1 of 2)

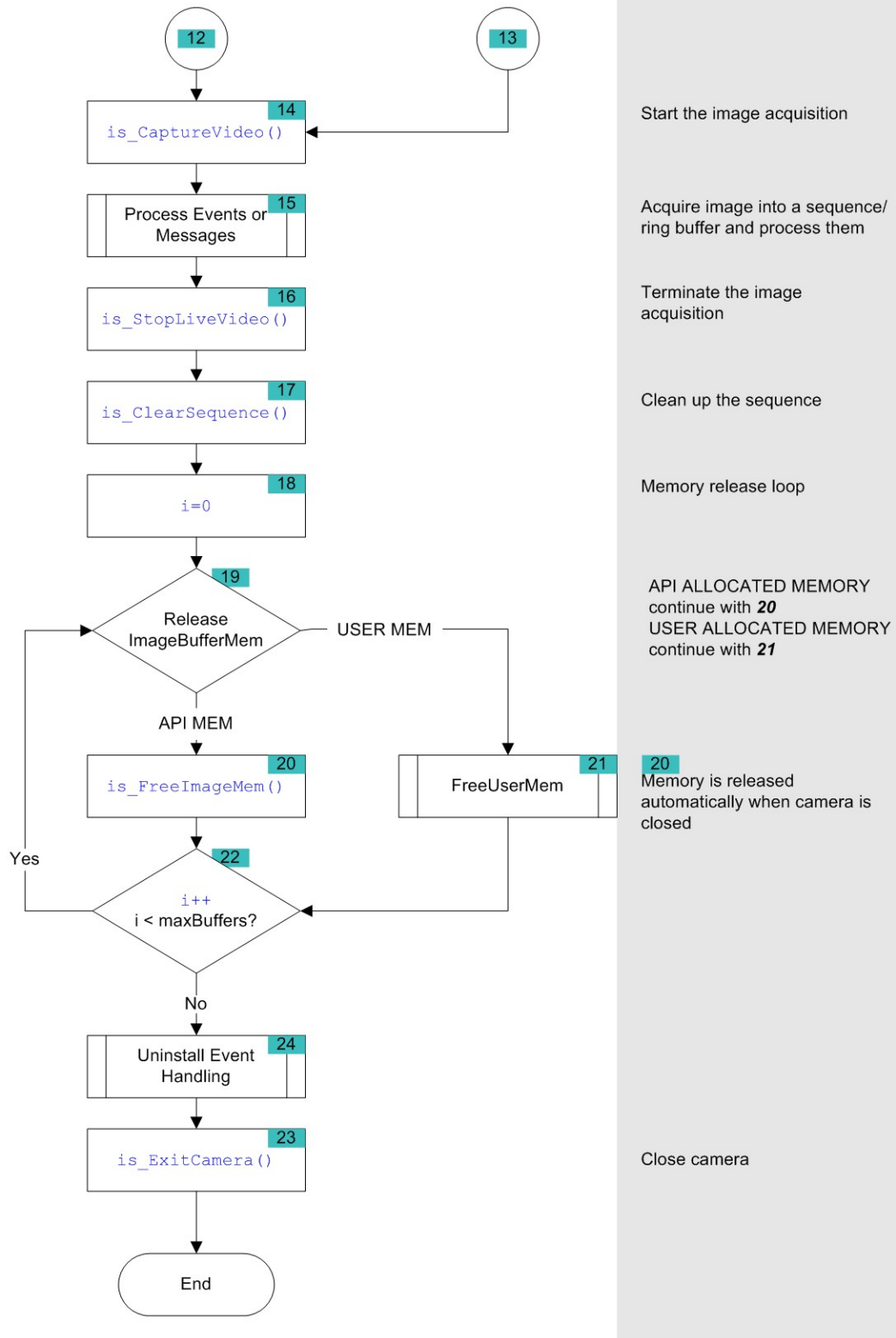


Figure 107: Flowchart - Capturing a frame sequence (2 of 2)

5.2.3.2 Event / Message Handling

Triggering Events for Single-Frame Capture

The following figure shows the time sequence when triggering the `IS_TRIGGER` and `IS_FRAME` events. The camera is prepared for triggered capture using the `is_SetExternalTrigger()` command. An incoming trigger signal at the camera starts the exposure and the subsequent image transfer. Upon completion of the data transfer, the `IS_TRIGGER` event signals that the camera is ready for the next capture. The `IS_FRAME` event is set once pre-processing (e.g. colour conversion) is complete and the finished image is available in the user memory.



The following illustrations show a schematic view of the image capture sequence. The sensor exposure and readout times and the transmission times depend on the camera settings. The pre-processing time depends on the API functions you are using (e.g. colour conversion, edge enhancement).

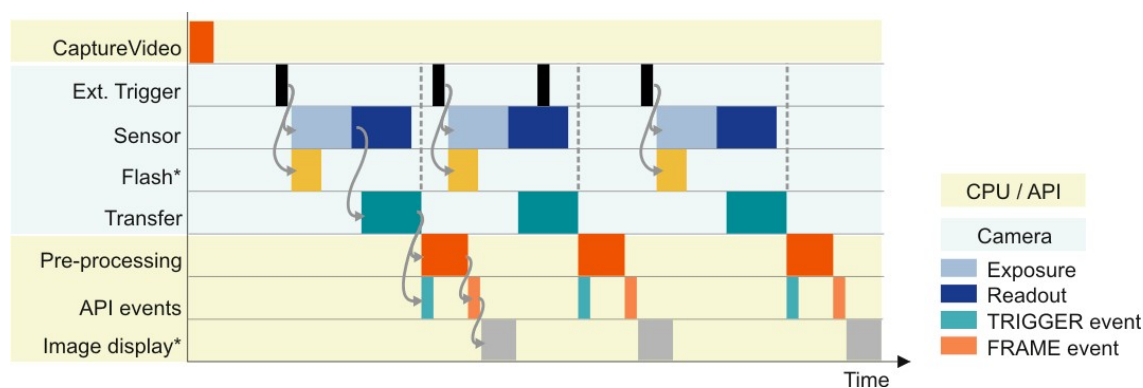


Figure 108: Events in hardware trigger mode

Events in Live Mode (Image Sequence)

The following figure shows the time sequence when triggering the `IS_FRAME` and `IS_SEQUENCE` events. The camera is set to live mode using `is_CaptureVideo()` so that it continuously captures frames. The `IS_FRAME` event is set once pre-processing (e.g. colour conversion) is complete and a finished image is available in the user memory. The `IS_SEQUENCE` event is set after one cycle of a storing sequence has been completed (see also `is_AddToSequence()`).

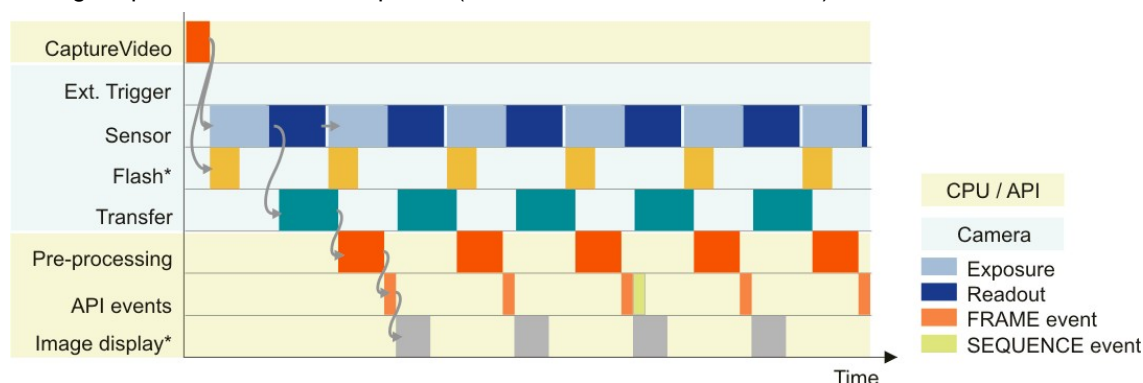
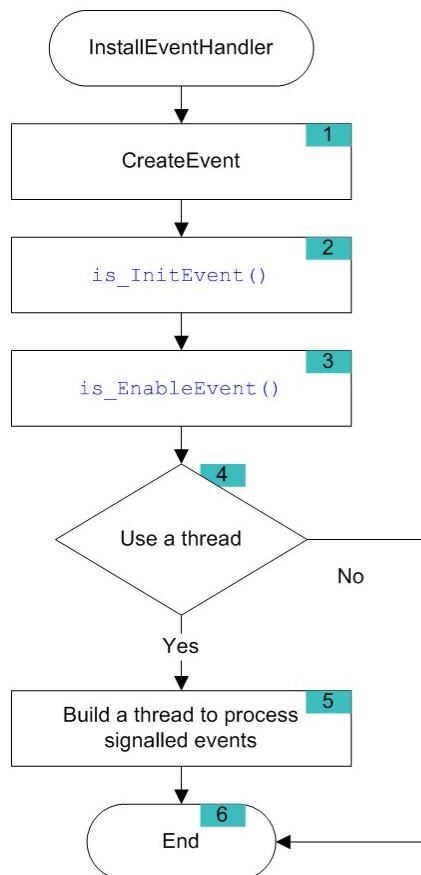


Figure 109:
Events in live mode

*) Optional function. The start time and duration of the flash signal are defined by the *Flash delay* and *Duration* parameters (see `is_SetFlashDelay()`).

Function List

<code>is_DisableEvent()</code>	Disables a single event object.
<code>is_EnableEvent()</code>	Enables a single event object.
<code>is_EnableMessage()</code>	Turns the Windows messages on / off.
<code>is_ExitEvent()</code>	Closes the event handler.
<code>is_InitEvent()</code>	Initialises the event handler.
<code>is_EnableAutoExit()</code>	Automatically releases the camera resources when the camera is disconnected from the PC.

Flowchart: Enable Events

Create an event and receive the handle of the event object

Initialise the uEye event handling. The event handle from CreateEvent is required.

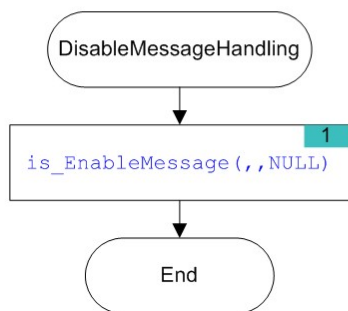
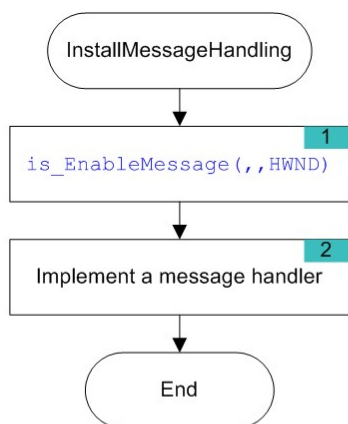
Enable a specific event.

Using a thread (recommended) continue with **5** else continue with **6**.

Windows functions:
WaitForSingleObject()
WaitForMultipleObjects()

Figure 110: Flowchart - Enabling event handling

Flowchart: Enabling Messages



MS Windows only

Enable the message handling.

This may be different in each programming language. Hook on the message map.



When the messages are sent to the applications main window and the GUI is locked for some reasons the uEye messages cannot be processed.

Disable the message handling.

Figure 111: Flowchart - Enabling message handling

Flowchart: Event / Message Handling

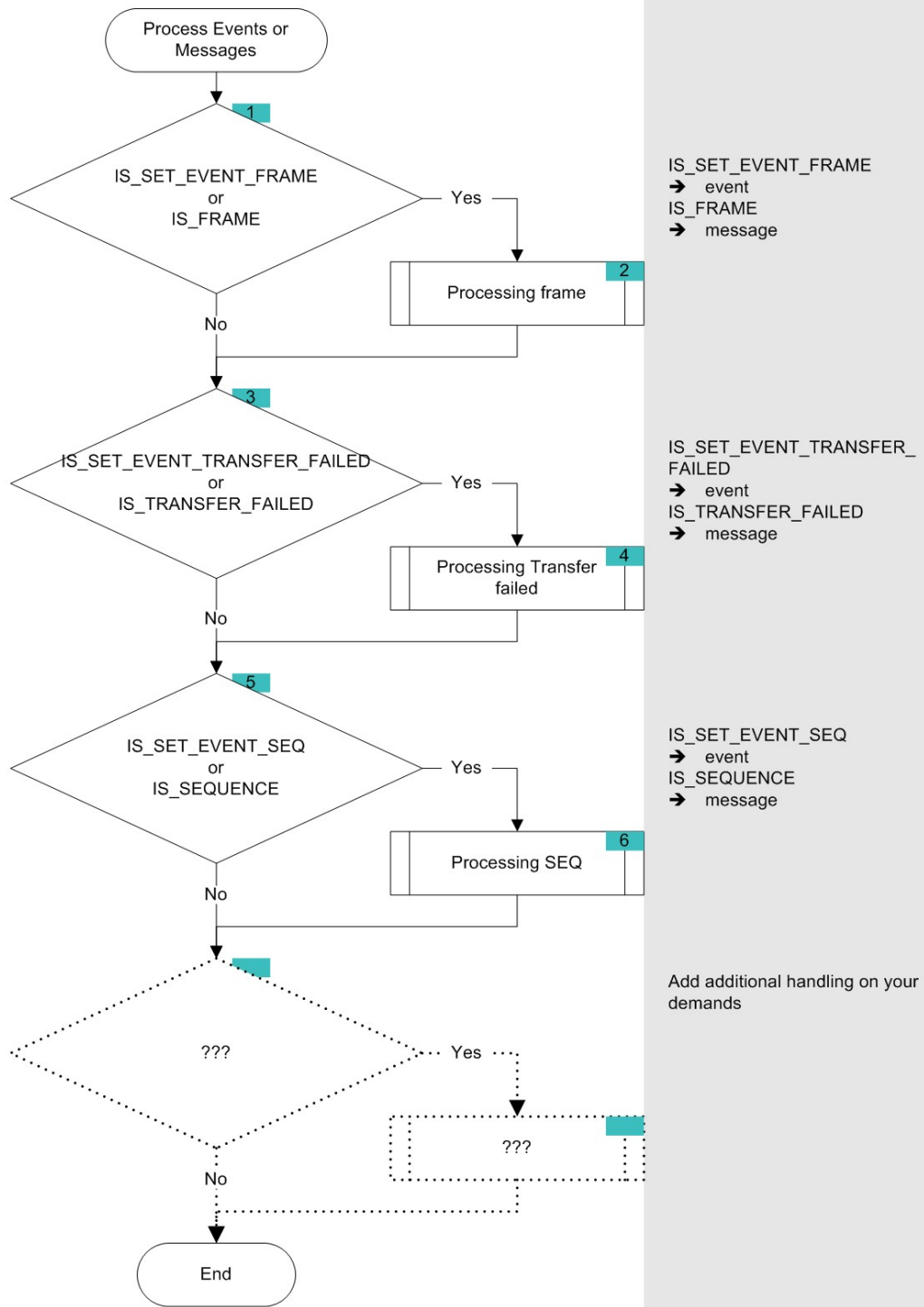


Figure 112: Flowchart - Handling events / messages

5.2.4 Saving Images and Videos



Using the *uEye API*, you can

- save and load single frames and
- capture an AVI frame sequence

5.2.4.1 Saving and Loading Single Frames

Using the following functions, you can save the image data of the current image memory to a BMP or JPG file, and load saved image data into an image memory:

<u>is_LoadImage()</u>	Loads an image from a bitmap file into the current image memory.
<u>is_LoadImageMem()</u>	Loads an image from a bitmap file into a new image memory.
<u>is_SaveImage()</u>	Saves the current image as a BMP file.
<u>is_SaveImageEx()</u>	Saves the current image as a JPEG or BMP file.
<u>is_SaveImageMem()</u>	Saves the contents of the image memory as a BMP file.
<u>is_SaveImageMemEx()</u>	Saves the contents of the image memory as a JPEG or BMP file.

5.2.4.2 Capturing AVIs

The functions of the *uEye_tools.dll* enable you to save images captured with the *uEye* as sequences to an AVI file. In order to reduce the file size, the single frames are stored in the AVI container using an adjustable JPEG compression. It is possible to extract single frames from the AVI file.

AVI Capture Workflow

<code>isavi_InitAVI()</code>	Initialises the <i>uEye</i> AVI interface.
<code>isavi_ExitAVI()</code>	Terminates and closes the <i>uEye</i> AVI interface.
<code>isavi_OpenAVI()</code>	Opens an AVI file for capturing.
<code>isavi_CloseAVI()</code>	Closes an AVI file.
<code>isavi_GetAVIFileName()</code>	Returns the name of the current AVI file.

The following settings should also be done prior to starting the recording.

<code>isavi_SetFrameRate()</code>	Sets the frame rate of the AVI video.
<code>isavi_SetImageQuality()</code>	Sets the compression level / image quality of the AVI video.
<code>isavi_SetImageSize()</code>	Sets the size and offset of the input image memory.

Once the AVI file has been created, captured images are placed in a buffer. Then, the images are compressed and added to the AVI file which is stored on the hard disk. These operations are not performed in the same thread as the capturing process. If you capture more images while a compression or write operation is in progress, the new images will be discarded.

<code>isavi_StartAVI()</code>	Starts AVI recording.
<code>isavi_AddFrame()</code>	Adds a compressed image to the AVI file.
<code>isavi_StopAVI()</code>	Stops AVI recording.

With these functions, you can query additional information on the ongoing recording.

<code>isavi_GetAVISize()</code>	Returns the size of the current AVI file.
<code>isavi_GetnCompressedFrames()</code>	Returns the number of frames in the current AVI file.
<code>isavi_GetnLostFrames()</code>	Returns the number of frames that have been discarded so far.
<code>isavi_ResetFrameCounters()</code>	Resets the counters for discarded and saved frames to 0.

Events can be used to get signalled when a frame was added.

<code>isavi_DisableEvent()</code>	Disables a <i>uEye</i> AVI event.
<code>isavi_EnableEvent()</code>	Enables a <i>uEye</i> AVI-Event.
<code>isavi_ExitEvent()</code>	Turns off <i>uEye</i> AVI event handling.
<code>isavi_InitEvent()</code>	Turns on <i>uEye</i> AVI event handling.

Supported Colour Formats

The supported input colour formats are RGB32, RGB24, Y8 and raw Bayer. The output file will always be in RGB24 format, regardless of the input data format. You can adjust the size of the

images to be stored by defining a freely selectable area of interest (AOI).

Capture Speed

The possible speed of capture depends on the selected colour format, the image size and the compression level of the AVI file as well as the PC performance.

Playback in External Applications

AVI files you have captured using the *uEye_tools.dll* can also be played back in external applications, such as Windows Media Player. To do this, you need to install the *uEye* MJPEG codec on your system:

- Open the *uEye* installation directory (default: *C:\Program Files\IDS\uEye\Program*).
- Right-click the *IdsMjpeg.inf* file.
- Select "Install". The codec is installed automatically.

In player or recording software, the codec will show up as "Intermedia-X MJPEG Codec".

Flowchart: AVI Capture

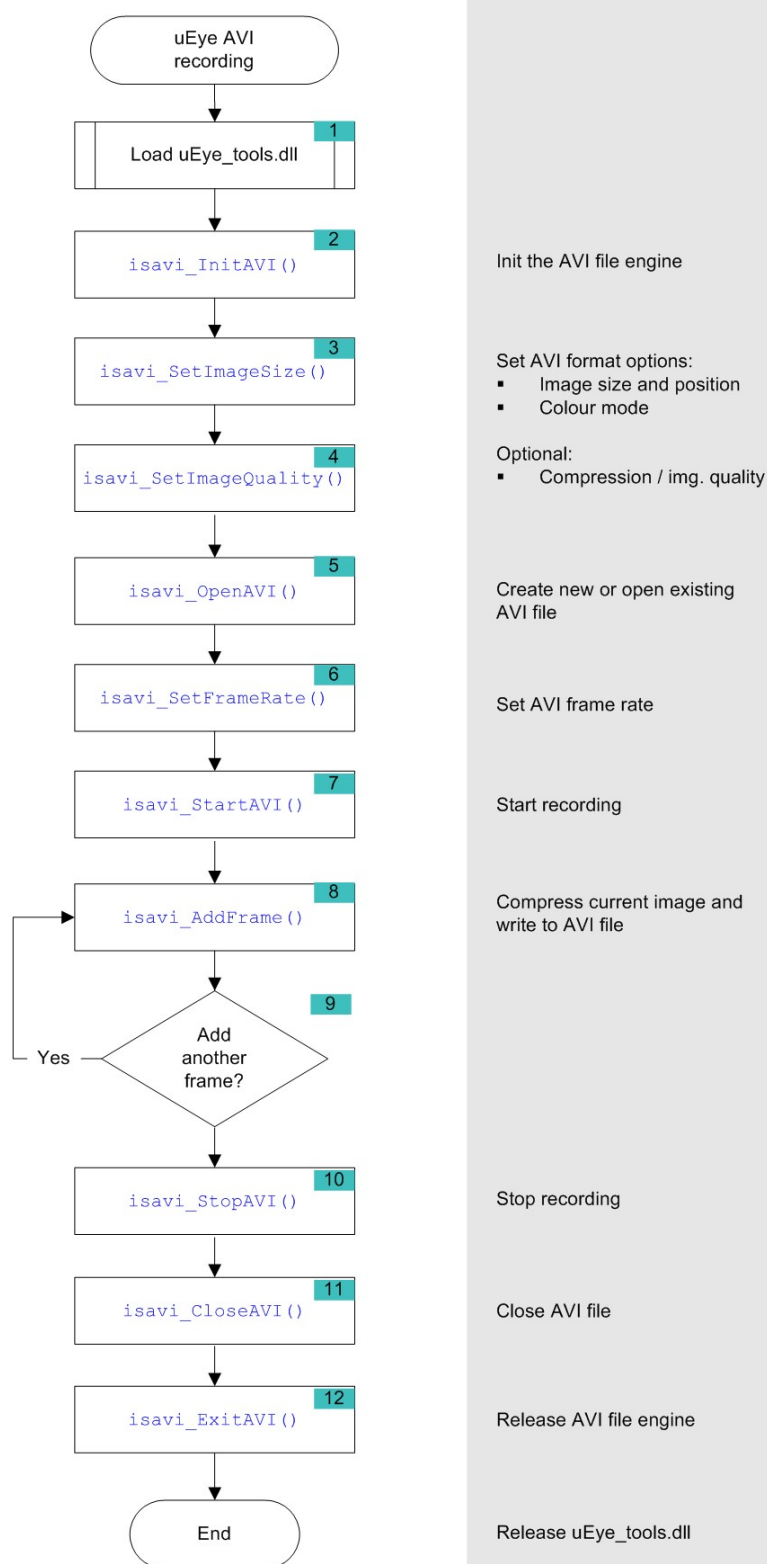


Figure 113: Flowchart - Capturing AVIs with the uEye

5.2.5 Setting Camera Parameters



- **Setting and getting parameters:** Using these functions, you can make settings for the camera and for image capture and preprocessing.
- The *uEye*'s **automatic image control** features allow automatically adjusting image brightness and image color to changing ambient conditions.
- **Image preprocessing:** These functions specify e.g. how color images are processed after image capture.
- **Querying the camera status:** With these functions, you can query additional useful information on the camera status.
- **Using the camera EEPROM:** All *uEye* cameras have a non-volatile EEPROM where you can save the camera settings or any other information.

5.2.5.1 Setting and Getting Parameters

This set of functions specifies the camera's image capture parameters, such as exposure, pixel clock and frame rate:

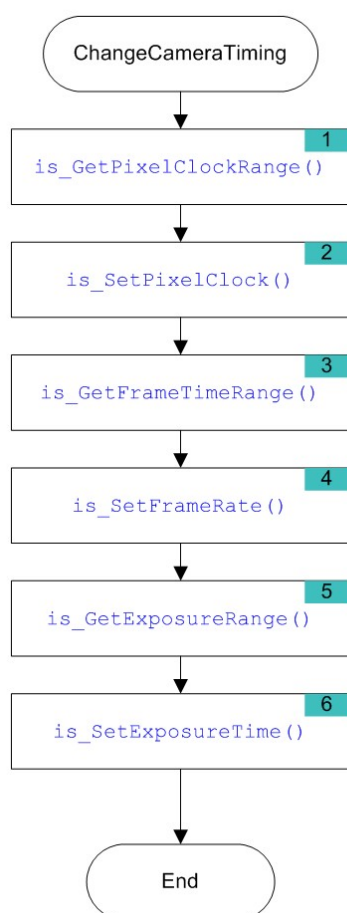
<code>is_GetExposureRange()</code>	Returns the adjustable exposure range.
<code>is_GetFramesPerSecond()</code>	Returns the current frame rate in live mode.
<code>is_GetFrameTimeRange()</code>	Returns the adjustable frame rate range.
<code>is_GetPixelClockRange()</code>	Returns the adjustable pixel clock range.
<code>is_SetAutoParameter()</code>	Enables / disables automatic imaging functions.
<code>is_SetBlCompensation()</code>	Turns black level correction on / off.
<code>is_SetExposureTime()</code>	Sets the exposure time.
<code>is_SetFrameRate()</code>	Sets the frame rate.
<code>is_SetGainBoost()</code>	Sets additional sensor hardware gain boost.
<code>is_SetGamma()</code>	Sets the gamma value (digital post-processing).
<code>is_SetGlobalShutter()</code>	Enables / disables the Global Start shutter.
<code>is_SetHardwareGain()</code>	Enables the sensor hardware gain.
<code>is_SetHardwareGamma()</code>	Sets the sensor gamma control.
<code>is_SetHWGainFactor()</code>	Sets the sensor hardware gain factor.
<code>is_SetPixelClock()</code>	Sets the pixel clock frequency.
<code>is_ResetToDefault()</code>	Resets the camera parameters to its default values.

This set of functions lets you influence the image geometry for image capture, e.g. the area of interest:

<code>is_SetAOI()</code>	Sets the size and position of an area of interest (AOI) or of a reference AOI for auto imaging functions.
<code>is_SetBinning()</code>	Sets the binning modes.
<code>is_SetRopEffect()</code>	Makes real-time geometry changes to an image (Rop = <i>raster operation</i>)
<code>is_SetSubSampling()</code>	Sets the subsampling modes.

The following set of functions refers to the further processing of image data in the PC:

<code>is_GetColorDepth()</code>	Determines the desktop color mode set in the graphics card.
<code>is_GetTimeout()</code>	Returns the user-defined timeout values.
<code>is_LoadBadPixelCorrectionTable()</code>	Loads a user-defined hot pixel list from a file.
<code>is_SaveBadPixelCorrectionTable()</code>	Saves the current user-defined hot pixel list.
<code>is_SetBadPixelCorrection()</code>	Turns hot pixel correction on / off.
<code>is_SetBadPixelCorrectionTable()</code>	Enables a user-defined hot pixel list.
<code>is_SetColorConverter()</code>	Selects Bayer conversion mode.
<code>is_SetColorCorrection()</code>	Sets color correction.
<code>is_SetColorMode()</code>	Selects a color mode.
<code>is_SetConvertParam()</code>	Conversion parameters for raw Bayer conversion.
<code>is_SetEdgeEnhancement()</code>	Sets edge enhancement.
<code>is_SetSaturation()</code>	Sets the image saturation (digital post-processing).
<code>is_SetSensorTestImage()</code>	Enables test image output from sensor.
<code>is_SetTimeout()</code>	Sets user-defined timeout values.

Flowchart: Changing camera timing

Recommended order of function calls for timing changes.

Return the possible range for the pixel clock

The pixel clock determines the range of frame rate settings.

Return the possible range of the frame rate

The frame rate defines the range for the exposure time.

Return the possible exposure range

Set the exposure time

Figure 114: Flowchart - Changing camera timing

5.2.5.2 Automatic Image Control

The uEye driver provides various options to automatically adjust the image capture parameters to the lighting situation. All controls are configured using the `is_SetAutoParameter()` SDK function.

For more information on the automatic image control see [Camera Basics: Automatic Image Control](#).

Flowchart: Enable Auto Brightness

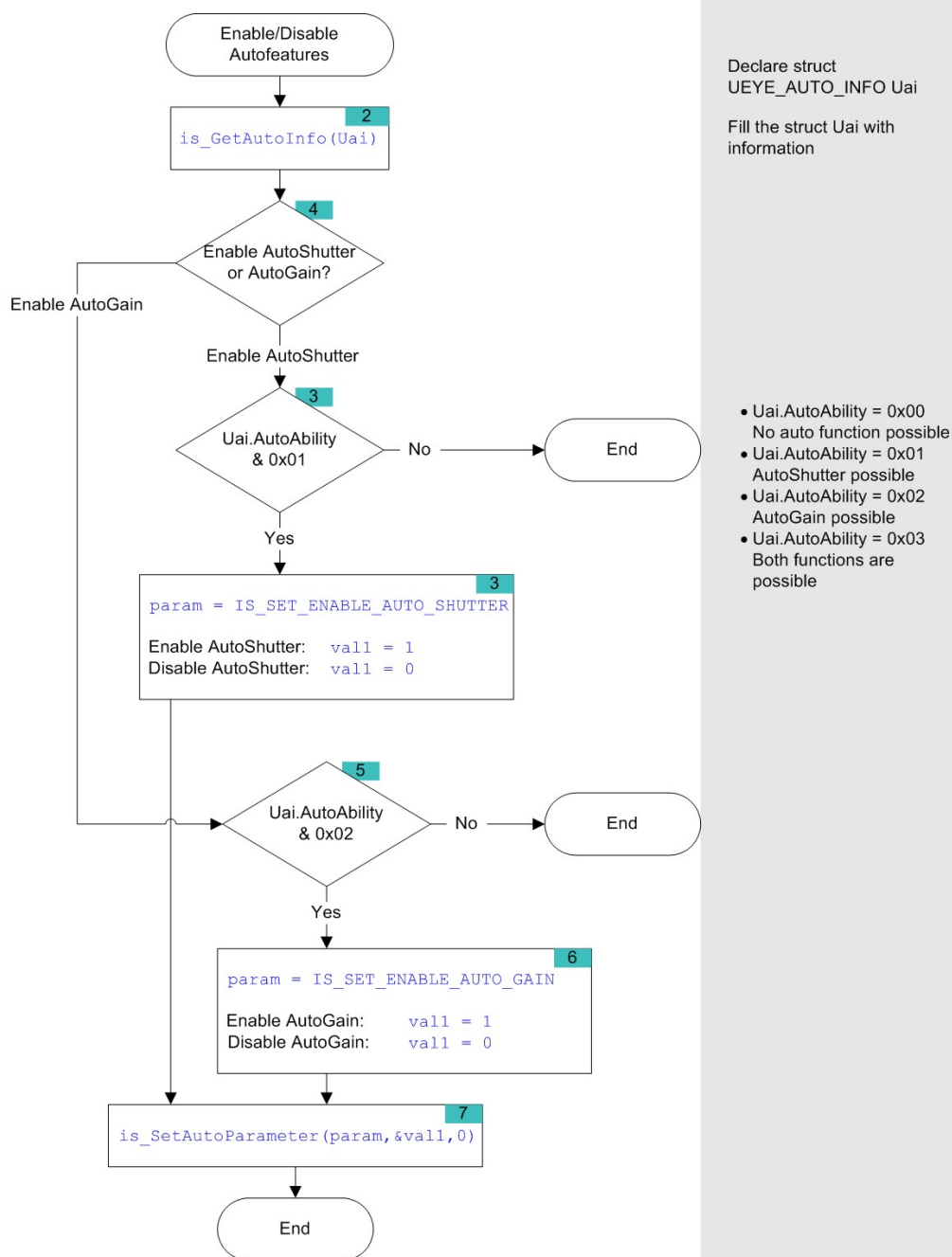


Figure 115: Flowchart - Enabling automatic image brightness control

5.2.5.3 Image Pre-processing

The following functions enable and adjust the Bayer conversion (see [Color Filter \(Bayer Filter\)](#)).

<code>is_ConvertImage()</code>	Converts a Bayer raw image into the desired output format
<code>is_GetColorConverter()</code>	Returns the currently set Bayer conversion mode
<code>is_SetBayerConversion()</code>	Sets the algorithm for Bayer conversion
<code>is_SetColorConverter()</code>	Sets the algorithm for Bayer conversion in the camera (<i>GigE uEye HE</i> cameras only)

Using Look-Up-Table (LUT) functions, you can e. g. adjust brightness or contrast after the acquisition.

<code>is_GetCameraLUT()</code>	Read out current hardware LUT
<code>is_SetCameraLUT()</code>	Activate/deactivate hardware LUT
<code>is_GetImageHistogram()</code>	Computes a histogram for the image buffer passed to the function

5.2.5.4 Get Camera Status

Using these functions, you can read out additional useful information on the camera status.

<code>is_CameraStatus()</code>	Returns the event counters and other information. Enables standby mode.
<code>is_GetAutoInfo()</code>	Returns status information on the auto features.
<code>is_GetCameraList()</code>	Returns information on all connected cameras.
<code>is_GetCameraType()</code>	Returns the camera type.
<code>is_GetCaptureErrorInfo()</code>	Displays information on errors that have occurred.
<code>is_GetError()</code>	Displays errors that have occurred.
<code>is_GetUsedbandwidth()</code>	Returns the bus bandwidth (in Mbit/s) currently used by all initialised or selected cameras.
<code>is_GetVsyncCount()</code>	Returns the VSYNC counter. It will be incremented by 1 each time the sensor starts capturing an image.
<code>is_SetErrorReport()</code>	Enables / disables dialogue messages for error output.

5.2.5.5 Using the Camera EEPROM

The non-volatile EEPROM of every *uEye* camera can hold user data or camera settings.

<code>is_GetCameraInfo()</code>	Returns the factory-set information (e.g. revision information for the individual <i>uEye</i> components).
<code>is_GetSensorInfo()</code>	Returns the sensor information.
<code>is_ReadEEPROM()</code>	Reads out the writable data area of the EEPROM.
<code>is_WriteEEPROM()</code>	Writes user data to the EEPROM.

5.2.6 Using Inputs and Outputs

Depending on the model, *uEye* cameras have one or more digital inputs and outputs designed for different purposes.



- **Input/output control:** Here, you will find functions for setting the *uEye*'s I/Os and for using the trigger and flash modes.
- **I2C bus:** The board-level versions of the *USB uEye ME* and *USB uEye LE* provide an I2C interface for transferring data to external devices.
- **Serial interface:** The *GigE uEye HE* provides a serial RS232 interface for transferring data to external devices.

5.2.6.1 Input / Output Control

With these functions you can use the camera's digital in-/outputs for trigger and flash control.

<u><code>is_SetExternalTrigger()</code></u>	Enables the digital input for trigger operation or returns the applied signal level.
<u><code>is_SetFlashStrobe()</code></u>	Sets the digital output for flash control or a static output level.
<u><code>is_SetFlashDelay()</code></u>	Sets the delay and power-on time of the flash output.
<u><code>is_SetTriggerDelay()</code></u>	Sets the trigger signal delay time.
<u><code>is_GetGlobalFlashDelays()</code></u>	Determines the delay and power-on times of the flash output to obtain a global shutter effect when using rolling shutter sensors.
<u><code>is_ForceTrigger()</code></u>	Simulates a trigger signal in hardware trigger mode.

With these commands you can activate additional functions or use GPIOs on some *uEye* cameras.

<u><code>is_SetIO()</code></u>	Sets the additional digital outputs (GPIO).
<u><code>is_SetIOMask()</code></u>	Defines each port as a digital input or output (GPIO).
<u><code>is_SetLED()</code></u>	Toggles the colour of the status LED for the <i>USB uEye SE</i> camera series.

Flowchart: Digital input

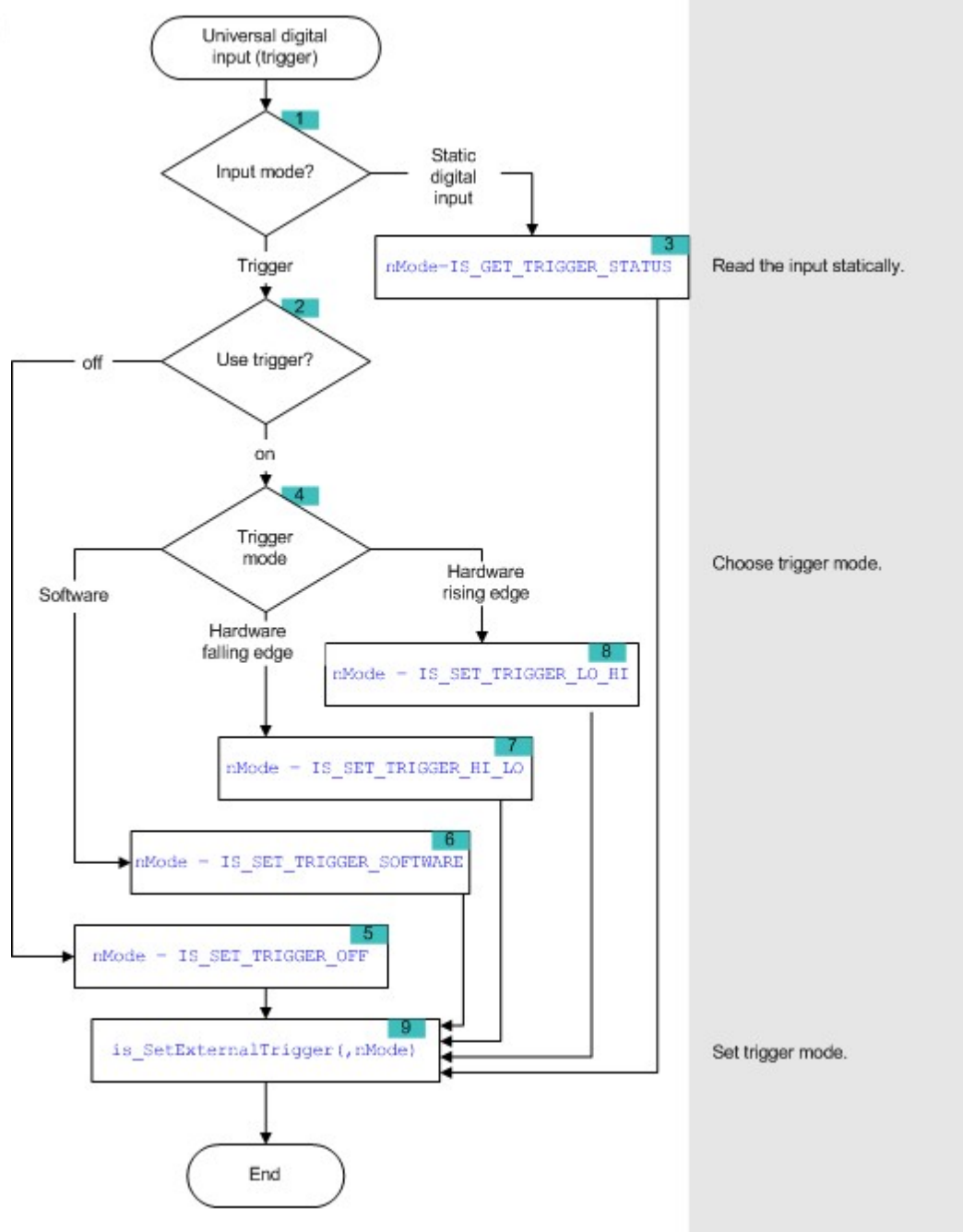


Figure 116: Flowchart - Digital input

Flowchart: Digital output

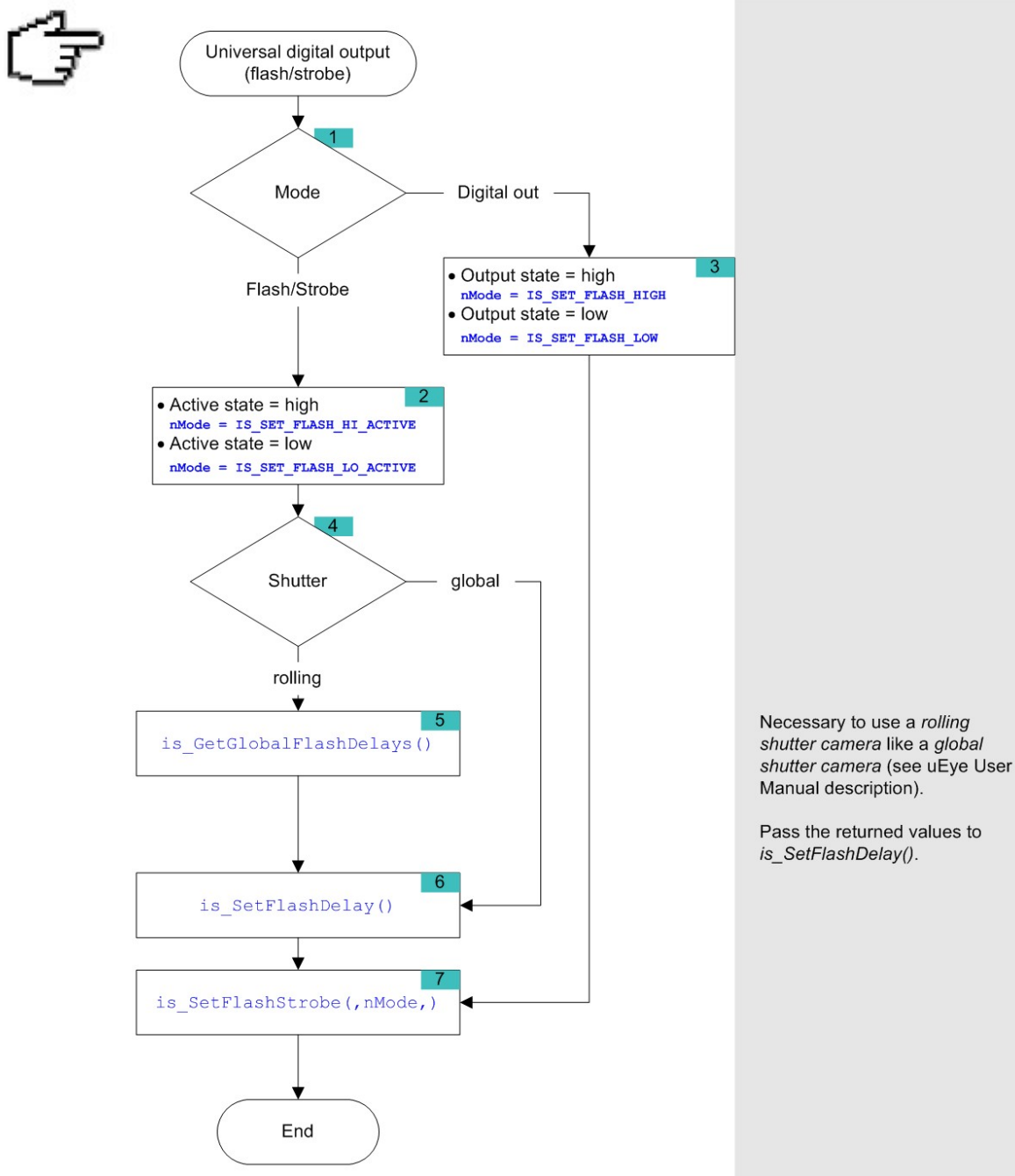


Figure 117: Flowchart - Digital output

5.2.6.2 I2C Functions

Function List

<code>is_ReadI2C()</code>	Reads data via the I2C bus.
<code>is_WriteI2C()</code>	Writes data via the I2C bus.

5.2.6.3 Serial Interface (GigE uEye HE only)



The *GigE uEye HE* has a serial interface which can be accessed from the PC through a virtual COM port (please also refer to the [Camera Basics: Serial Interface](#) chapter).

The Windows API provides all necessary functions for a COM port-based communication. The three Windows API commands listed in the table below are sufficient to perform a basic COM communication. You can use additional Windows API functions to further configure the COM port. The *uEyeComportDemo.exe* COM port demo program which is supplied in C++ source code with the *uEye SDK* shows how to use these functions.

<code>CreateFile()</code>	Windows API function. You can use this function to initialise the specified communication resource – in this case, the virtual COM port. The function returns a handle for accessing the port.
<code>WriteFile()</code>	Windows API function. You can use this function to write data to the port identified by the handle returned by <code>CreateFile()</code> .
<code>ReadFile()</code>	Windows API function. You can use this function to read data from the port identified by the handle returned by <code>CreateFile()</code> .
<code>is_GetComportNumber()</code>	<i>uEye SDK</i> function. Retrieves the currently set COM port number of the camera.

5.3 Function Descriptions

To integrate the *uEye* cameras into your own programs, you can use the functions and parameters provided by the *uEye SDK*. These are described in this chapter. The descriptions are listed alphabetically by function and are structured as follows:

	
USB 2.0 GigE	USB 2.0 GigE

This table shows the availability of the function. For both Windows and Linux the table shows which *uEye* camera series supports the function.

Syntax

Prototype of the function from the `ueye.h` header file

Description

Description of the function with cross-references to related functions

Input Parameters

Description of the function parameters including their value ranges

Return Value

Description and value range of the return value. If a function returns the `IS_NO_SUCCESS (-1)` value, you can get information on the error from the `is_GetError()` function.

Related Functions

List with similar or related SDK functions

Code Sample



For some functions, C++ programming samples are have been added.

Sample Programs

Some descriptions include references to *uEye SDK* sample programs. When you install the *uEye* software, the demo applications are copied to the `C:\Programs\IDS\uEye\Samples` directory. The associated source code can be found under `C:\Programs\IDS\uEye\Develop\Source`.

All sample programs are described in the *uEye Samples Manual*.

5.3.1 `is_AddToSequence`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_AddToSequence (HIDS hCam, char* pcImgMem, INT nID)
```

Description

`is_AddToSequence()` adds an image memory to the list of image memories used for ring buffering. The image memory must have been previously requested using `is_AllocImageMem()`. Using the `is_SetAllocatedImageMem()` function, you can set a memory that has been allocated before as image memory. Image memories that are used for ring buffering must all have been allocated with the same colour depth (bits per pixel).

Input Parameters

<code>hCam</code>	Camera handle
<code>pcMem</code>	Pointer to image memory
<code>nID</code>	Image memory ID

Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message



Related Functions

- `is_AllocImageMem()`
- `is_SetImageMem()`
- `is_SetAllocatedImageMem()`

Sample Programs

- `uEyeSequence` (C++)

5.3.2 is_AllocImageMem

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_AllocImageMem (HIDS hCam,
                     INT width, INT height, INT bitspixel,
                     char** ppcImgMem, INT* pid)
```

Description

`is_AllocImageMem()` allocates an image memory for an image having its dimensions defined by width and height and its colour depth defined by bitspixel. The memory size is at least:

size = [width * ((bitspixel + 1) / 8) + adjust] * height (for details on adjust, see below)

The line increment is calculated as:

```
line      = width * [(bitspixel + 1) / 8]
lineinc   = line + adjust.
adjust    = 0, if line can be divided by 4 without remainder
adjust    = 4 - rest(line / 4), if line cannot be divided by 4 without remainder
```

To read out the line increment, you can use the `is_GetImgMemPitch()` function.

The starting address of the memory area is returned in `ppcImgMem`.

`pid` returns an ID for the allocated memory. A newly allocated memory is not directly active, i.e. digitised images will not be stored immediately in this new memory. It must first be made active using `is_SetImageMem()`.

The returned pointer must be write-protected and may not be altered because it will be used for all further ImageMem functions. To release the memory, you can use `is_FreeImageMem()`.



In the Direct3D modes, image memory allocation is not necessary.



In case the operating system is short of physical memory, today's OS versions swap individual areas of the RAM that have not been used for some time out to the slower hard disk. This can slow down image capture if more image memory has been allocated than can be provided by the RAM at a time.

Input Parameters

<code>hCam</code>	Camera handle
<code>width</code>	Image width
<code>height</code>	Image height
<code>bitapixel</code>	Image colour depth (bits per pixel). RGB16 and RGB15 require the same amount of memory, but can be distinguished by the <code>bitapixel</code> parameter.
<code>ppcImgMem</code>	Returns the pointer to the memory starting address
<code>pid</code>	Returns the ID of this memory

Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message



Related Functions

- [`is_FreeImageMem\(\)`](#)
- [`is_AddToSequence\(\)`](#)
- [`is_SetImageMem\(\)`](#)
- [`is_SetAllocatedImageMem\(\)`](#)
- [`is_GetColorDepth\(\)`](#)
- [`is_GetImgMemPitch\(\)`](#)

Sample Programs

- `uEyeRotationDemo (C++)`

5.3.3 is_CameraStatus

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
ULONG is_CameraStatus (HIDS hCam, INT nInfo, ULONG ulValue)
```

Description

Using `is_CameraStatus()`, you can query and partly set various status information and settings.

Input Parameters

hCam	Camera handle
nInfo	
IS_FIFO_OVR_CNT	Number of FIFO overruns. Is increased if image data gets lost because the USB bus is congested.
IS_SEQUENCE_CNT	Returns the sequence count. For <code>is_CaptureVideo()</code> , this parameter is set to 0. Each time the sequence buffer (image counter) changes, the counter is increased by 1.
IS_SEQUENCE_SIZE	Returns the number of sequence buffers.
IS_EXT_TRIGGER_EVENT_CNT	Returns the camera's internal count of external trigger events.
IS_TRIGGER_MISSED	Returns the number of unprocessed trigger signals. Is reset to 0 after each call.
IS_LAST_CAPTURE_ERROR	Returns the last image capture error, e.g. after a 'transfer failed' event. For a list of all possible error events, see <code>is_GetCaptureErrorInfo()</code> .
IS_PARAMETER_SET_1	Indicates whether parameter set 1 including camera settings is present on the camera (read-only). See also <code>is_SaveParameters()</code> . Return values: TRUE Parameter set 1 present FALSE Parameter set 1 not present
IS_PARAMETER_SET_2	Indicates whether parameter set 2 including camera settings is present on the camera (read-only). See also <code>is_SaveParameters()</code> . Return values: TRUE Parameter set 2 present FALSE Parameter set 2 not present
IS_STANDBY	Sets the camera to standby mode. Return values: TRUE Camera changes to standby mode FALSE The camera changes to freerun mode
IS_STANDBY_SUPPORTED	Queries whether the camera supports standby mode (read-only).

	Return values: TRUE The camera supports standby mode FALSE The camera does not support standby mode
ulValue	
IS_GET_STATUS	Returns the information specified by nInfo.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
Returns the information specified by nInfo.	Only if ulValue = IS_GET_STATUS
When used with IS_LAST_CAPTURE_ERROR	Returns the last image capture error. For a list of all possible error events, see <u>is_GetCaptureErrorInfo()</u> .

Related Functions

- is_GetCameraInfo()
- is_GetError()
- is_SetErrorReport()
- is_SetTriggerCounter()

5.3.4 is_CaptureVideo

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_CaptureVideo (HIDS hCam, INT Wait)
```

Description

`is_CaptureVideo()` digitises video images in real time and transfers the images to an allocated image memory or, if Direct3D is used, to the graphics card. The image data (DIB mode) is stored in the memory created using `is_AllocImageMem()` and designated as active image memory using `is_SetImageMem()`. Using `is_GetImageMem()`, you can query the memory address.

If ring buffering is used, the image capturing function cycles through all image memories used for storing the images of a capture sequence in an endless loop. Sequence memories locked by `is_LockSeqBuf()` will be skipped. If the last available sequence memory has been filled, the sequence event or message will be triggered. Capturing always starts with the first element of the sequence.

For further information on the image capture modes of the *uEye* camera, see the [How To Proceed: Image Capture](#) section.

Input Parameters

hCam	Camera handle
Wait	
IS_DONT_WAIT	Timeout value for image capture (see also the How To Proceed: Timeout Values for Image Capture section)
IS_WAIT	
Time t	
IS_GET_LIVE	Returns if live capture is enabled.

Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
When used with IS_GET_LIVE	TRUE if live capture is enabled



Related Functions

- [is_FreezeVideo\(\)](#)
- [is_StopLiveVideo\(\)](#)
- [is_SetExternalTrigger\(\)](#)
- [is_ForceTrigger\(\)](#)
- [is_SetTimeout\(\)](#)
- [is_GetCaptureErrorInfo\(\)](#)

Sample Programs

- SimpleLive (C++)
- uEyeC# Demo (C#)
- uEye VB Simple Demo (VB6)

5.3.5 is_ClearSequence

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

INT is_ClearSequence (HIDS hCam)

Description

is_ClearSequence() removes all image memories from the sequence list that were added using [is_AddToSequence\(\)](#). After a call of is_ClearSequence(), there is no more active image memory. To make an image memory the active memory, call [is_SetImageMem\(\)](#).

Input Parameters

hCam	Camera handle
------	---------------



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [is_AddToSequence\(\)](#)
- [is_FreeImageMem\(\)](#)
- [is_SetImageMem\(\)](#)

5.3.6 is_ConvertImage

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_ConvertImage(HIDS hCam,
                   char* pcSource, INT nIDSource,
                   char** ppcDest, INT* nIDDest,
                   INT* reserved)
```

Description

is_ConvertImage() converts a raw Bayer image to the desired format. This conversion is done in the PC. You can use is_SetConvertParam() to define the conversion settings.

Input Parameters

hCam	Camera handle
pcSource	Pointer to the input image
nIDSource	Memory ID of the input image
ppcDest	Pointer to the output image In case a NULL value is passed, a new memory is allocated internally.
nIDDest	Memory ID of the output image
reserved	Reserved. NULL must be passed here.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- is_SetConvertParam()
- is_SetColorMode()
- is_SetBayerConversion()

5.3.7 is_CopyImageMem

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_CopyImageMem (HIDS hCam, char* pcSource, INT nID, char* pcDest)
```

Description

`is_CopyImageMem()` copies the contents of the image memory described by `pcSource` and `nID` to the memory area to whose starting address `pcDest` points.



The allocated memory must be large enough to accommodate the entire image in its current format (bits per pixel).

Input Parameters

<code>hCam</code>	Camera handle
<code>pcSource</code>	Pointer to the image memory
<code>nID</code>	ID of this image memory
<code>pcDest</code>	Pointer to the destination memory to copy the image to



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message

Related Functions

- [`is_AllocImageMem\(\)`](#)
- [`is_SetAllocatedImageMem\(\)`](#)

5.3.8 `is_CopyImageMemLines`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_CopyImageMemLines (HIDS hCam, char* pcSource,
                          INT nID, INT nLines, char* pcDest)
```

Description

`is_CopyImageMemLines()` copies the contents of the image memory described by `pcSource` and `nID` to the memory area to whose starting address `pcDest` points. The function only copies the number of lines indicated by `nLines`.



The allocated memory must be large enough to accommodate the entire image in its current format (bits per pixel).

Input Parameters

<code>hCam</code>	Camera handle
<code>pcSource</code>	Pointer to the image memory
<code>nID</code>	ID of this image memory
<code>nLines</code>	Number of lines to be copied
<code>pcDest</code>	Pointer to the destination memory to copy the image to



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message

Related Functions

- [`is_AllocImageMem\(\)`](#)
- [`is_SetAllocatedImageMem\(\)`](#)

5.3.9 is_DirectRenderer

	
USB 2.0 GigE	-

Syntax

```
INT is_DirectRenderer (HIDS hCam, UINT nMode, void* pParam, UINT nSize)
```

Description

`is_DirectRenderer()` provides a set of advanced rendering functions and allows inserting overlay data into the camera's live image without flicker. The graphics card functions of the Direct3D library are supported under Windows.

Notes on using the function:

The second input parameter `nMode` specifies the effect of the `is_DirectRenderer()` call. The value of the third parameter `pParam` depends on the mode selected with `nMode`: For example, when setting the overlay size (`nMode = DR_SET_OVERLAY_SIZE`), a pointer to an array of two values (x and y) is passed (see code samples). When you load a bitmap image (`nMode = DR_LOAD_OVERLAY_FROM_FILE`), `pParam` passes the path to the file (see code samples). The required parameters are illustrated in the sample codes at the end of this section.



- To use the Direct3D functionality, the appropriate version of the Microsoft DirectX Runtime has to be installed in your PC.
 - When you are using high-resolution cameras, the maximum texture size supported by the graphics card should be at least 4096 x 4096 pixels. You can check the maximum texture size by reading out the `D3D_GET_MAX_OVERLAY_SIZE` parameter.
 - The Direct3D mode automatically uses the Windows Desktop color depth setting for the display.
- Please also read the notes on graphics cards which are provided in the [System Requirements](#) chapter.

Input Parameters

<code>hCam</code>	Camera handle
<code>nMode</code>	
<code>DR_GET_OVERLAY_DC</code>	<p>Returns the device context (DC) handle to the overlay area of the graphics card.</p> <p>More details</p> <p>In Direct3D mode, the <code>DR_GET_OVERLAY_DC</code> mode returns the device context (DC) handle of the overlay area. Using this handle, it is possible to access the overlay using the Windows GDI functionality. Thus, all Windows graphics commands (e.g. Line, Circle, Rectangle, TextOut) are available. To transfer the drawn elements to the overlay, release the DC handle by calling <code>DR_RELEASE_OVERLAY_DC</code>.</p> <p>- Code sample</p>
<code>DR_RELEASE_OVERLAY_DC</code>	<p>Releases the device context (DC) handle.</p> <p>More details</p>

	Using DR_RELEASE_OVERLAY_DC, you can release the DC handle and update the overlay data. - Code sample
DR_GET_MAX_OVERLAY_SIZE	Returns the width x and height y of the maximum overlay area supported by the graphics card. Code sample
DR_SET_OVERLAY_SIZE	Defines the size of the overlay area (default: current camera image size). Code sample
DR_GET_OVERLAY_SIZE	Returns the size of the overlay area. (Sample: see DR_SET_OVERLAY_SIZE)
DR_SET_OVERLAY_POSITION	Defines the position of the overlay area. Code sample
DR_GET_OVERLAY_KEY_COLOR	Returns the RGB values of the current key color (default: black). Code sample
DR_SET_OVERLAY_KEY_COLOR	Defines the RGB values of the key color. More details The key color specifies where the camera image will be visible in the overlay area. For example: if you fill the complete overlay with the key color, the whole camera image will be visible. If you fill part of the overlay with a different color, the camera image will be covered by the overlay in those places. The key color has no effect in semi-transparent mode! - Code sample
DR_SHOW_OVERLAY	Enables overlay display on top of the current camera image. Code sample
DR_HIDE_OVERLAY	Disables overlay display. Code sample
DR_ENABLE_SCALING	Enables real-time scaling of the image to the size of the display window. The overlay is scaled together with the camera image. Code sample
DR_ENABLE_IMAGE_SCALING	Enables real-time scaling of the image to the size of the display window. The overlay is not scaled. (Sample: see DR_ENABLE_SCALING)
DR_DISABLE_SCALING	Disables real-time scaling. Code sample
DR_ENABLE _SEMI_TRANSPARENT_OVERLAY	Enables a semi-transparent display of the overlay area. More details In semi-transparent mode, the values of the camera image and the overlay data are added up for each pixel. Since black has the value 0, the complete camera image will be visible if the overlay is black; if the overlay is white, only the overlay will be visible. With all other colors, the camera image will be visible with the overlay superimposed. The key color has no effect in semi-transparent mode! - Code sample
DR_DISABLE _SEMI_TRANSPARENT_OVERLAY	Disables the semi-transparent display of the overlay area. Code sample

DR_SET_VSYNC_AUTO	Enables synchronization of the image display with the monitor's image rendering. The image is displayed upon the monitor's next VSYNC signal. Code sample				
DR_SET_VSYNC_OFF	Disables image display synchronization. The image is displayed immediately. Code sample				
DR_SET_USER_SYNC	<p>Enables synchronization of the image display with a monitor pixel row specified by the user. More details</p> <p>When displaying very large camera images, the auto-VSYNC function might not always optimally synchronize image rendering. In this case, you can eliminate flicker by manually setting a suitable position for synchronization. The position needs to be determined individually, based on the camera type and the graphics card.</p> <p>- Code sample</p>				
DR_GET_USER_SYNC_POSITION_RANGE	Returns the minimum and maximum row position for DR_SET_USER_SYNC. Code sample				
DR_LOAD_OVERLAY_FROM_FILE	Loads a bitmap image (*.BMP file) into the overlay area. If the bitmap image is larger than the overlay area, the bitmap image is clipped. Code sample				
DR_CLEAR_OVERLAY	Deletes the data of the overlay area by filling it with black color. Code sample				
DR_STEAL_NEXT_FRAME	<p>Copies the next image to the active user memory (Steal function). More details - Code sample</p> <p>Using the pParam parameter, you specify when the function should return:</p> <table data-bbox="831 1189 1433 1375"> <tr> <td>IS_WAIT</td><td>The function waits until the image save is complete.</td></tr> <tr> <td>IS_DONT_WAIT</td><td>The function returns immediately.</td></tr> </table>	IS_WAIT	The function waits until the image save is complete.	IS_DONT_WAIT	The function returns immediately.
IS_WAIT	The function waits until the image save is complete.				
IS_DONT_WAIT	The function returns immediately.				
DR_SET_STEAL_FORMAT	<p>Defines the color format for the Steal function. More details - Code sample</p> <p>For a list of all available color formats, see the function description for <code>is_SetColorMode()</code>. The default is <code>IS_CM_BGRA8_PACKED</code> (RGB 32).</p>				
DR_GET_STEAL_FORMAT	Returns the color format setting for the Steal function. Code sample				
DR_SET_HWND	Sets a new window handle for image output in Direct3D. Code sample				
DR_CHECK_COMPATIBILITY	Returns whether the graphics card supports the uEye Direct3D functions. Code sample				
pParam	void-type pointer to a data object or an array of objects (depending on the mode selected using nMode).				
nSize	Size (in bytes) of the data object or array.				

Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
When used with DR_CHECK_COMPATIBILITY	IS_DR_DEVICE_CAPS_INSUFFICIENT The graphics hardware does not fully support the <i>uEye</i> Direct3D functions

Related Functions

- `is_SetDisplayMode()`
- `is_SetColorMode()`
- `is_SetImageMem()`
- `is_RenderBitmap()`

Code Samples

```
//-----
//          DC-Handle
//-----

// Get DC handle for Overlay
HDC hDC;
is_DirectRenderer (hCam, DR_GET_OVERLAY_DC, (void*)&hDC, sizeof (hDC));

// Release DC handle
is_DirectRenderer (hCam, DR_RELEASE_OVERLAY_DC, NULL, NULL);

//-----
//          Size of overlay
//-----

// Query maximum size of overlay area
UINT OverlaySize[2];
is_DirectRenderer (hCam, DR_GET_MAX_OVERLAY_SIZE,
(void*)OverlaySize, sizeof(OverlaySize));
INT nWidth = OverlaySize[0];
INT nHeight = OverlaySize[1];

// Set size of overlay area
UINT Size[2];
Size[0] = 100;
Size[1] = 120;
is_DirectRenderer (hCam, DR_SET_OVERLAY_SIZE,
(void*)Size, sizeof (Size));

// Set position of overlay area
UINT Position[2];
```

```
Position[0] = 20;
Position[1] = 0;
is_DirectRenderer (hCam, DR_SET_OVERLAY_POSITION,
void*)Position, sizeof (Position));

//-----
//          Key color
//-----

// Get current key color
UINT OverlayKeyColor[3];
is_DirectRenderer (hCam, DR_GET_OVERLAY_KEY_COLOR,
(void*)OverlayKeyColor, sizeof(OverlayKeyColor));

INT nRed = OverlayKeyColor[0];
INT nGreen = OverlayKeyColor[1];
INT nBlue = OverlayKeyColor[2];

// Set new key color
OverlayKeyColor[0] = GetRValue(m_rgbKeyColor);
OverlayKeyColor[1] = GetGValue(m_rgbKeyColor);
OverlayKeyColor[2] = GetBValue(m_rgbKeyColor);
is_DirectRenderer (hCam, DR_SET_OVERLAY_KEY_COLOR,
(void*)OverlayKeyColor, sizeof(OverlayKeyColor));

//-----
//          Display
//-----

// Show overlay
is_DirectRenderer (hCam, DR_SHOW_OVERLAY, NULL, NULL);

// Hide overlay
is_DirectRenderer (hCam, DR_HIDE_OVERLAY, NULL, NULL);

//-----
//          Scaling
//-----

// Enable scaling
is_DirectRenderer (hCam, DR_ENABLE_SCALING, NULL, NULL);

// Disable scaling
is_DirectRenderer (hCam, DR_DISABLE_SCALING, NULL, NULL);

//-----
//          Transparency
```

```
//-----  
  
// Enable semi-transparent overlay  
is_DirectRenderer (hCam, DR_ENABLE_SEMI_TRANSPARENT_OVERLAY, NULL, NULL);  
  
// Disable semi-transparent overlay  
is_DirectRenderer (hCam, DR_DISABLE_SEMI_TRANSPARENT_OVERLAY, NULL, NULL);  
  
//-----  
//          Synchronization  
//-----  
  
// Enable auto-synchronization  
is_DirectRenderer (hCam, DR_SET_VSYNC_AUTO, NULL, NULL);  
  
// User defined synchronization: Query range and set position  
UINT UserSync[2];  
is_DirectRenderer (hCam, DR_GET_USER_SYNC_POSITION_RANGE,  
(void*)UserSync, sizeof (UserSync));  
INT Min = UserSync[0];  
INT Max = UserSync[1];  
INT SyncPosition = 400;  
is_DirectRenderer (hCam, DR_SET_USER_SYNC,  
void*)&SyncPosition, sizeof (SyncPosition));  
  
// Disable synchronization  
is_DirectRenderer (hCam, DR_SET_VSYNC_OFF, NULL, NULL);  
  
//-----  
//          BMP file  
//-----  
  
// Load overlay from BMP file  
is_DirectRenderer (hCam, DR_LOAD_OVERLAY_FROM_FILE,  
(void*)"c:\test.bmp", NULL);  
  
//-----  
//          Delete overlay  
//-----  
  
// Delete overlay area  
is_DirectRenderer (hCam, DR_CLEAR_OVERLAY, NULL, NULL);  
  
//-----  
//          Steal mode  
//-----
```

```
// Get and set color mode for image to be copied
INT nColorMode;
is_DirectRenderer (hCam, DR_GET_STEAL_FORMAT,
(void*)&nColorMode, sizeof (nColorMode));

nColorMode = IS_CM_MONO8;
is_DirectRenderer (hCam, DR_SET_STEAL_FORMAT,
void*)&nColorMode, sizeof (nColorMode));

// Copy image with function returning immediately
INT nwait = IS_DONT_WAIT;
is_DirectRenderer(hCam, DR_STEAL_NEXT_FRAME,
(void*)&nwait, sizeof (nwait));

//-----
//          Handle to window
//-----

// Set new window handle for image display
is_DirectRenderer (hCam, DR_SET_HWND,
(void*)&hWnd, sizeof (hWnd));

//-----
//          Compatibility
//-----



// Check graphics card compatibility
INT nRet = is_DirectRenderer (hCam, DR_CHECK_COMPATIBILITY, NULL, NULL);

if (nRet == IS_DR_DEVICE_CAPS_INSUFFICIENT )
// Graphics card does not support Direct3D
```

Sample Programs

- uEyeDirectRenderer
- uEyeSteal

5.3.10 [is_DisableEvent](#)

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_DisableEvent (HIDS hCam, INT which)
```

Description

Using [is_DisableEvent\(\)](#), you disable the event indicated here. The event (e.g. image capture completed) will usually still occur, but will no longer trigger an event signal. Disabled events are no longer signaled to the application. You can re-enable the desired event using [is_EnableEvent\(\)](#). See also [is_InitEvent\(\)](#).

Input Parameters

hCam	Camera handle
which	ID of the event to be disabled. See also is_InitEvent() .



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [is_EnableEvent\(\)](#)
- [is_ExitEvent\(\)](#)
- [is_InitEvent\(\)](#)

5.3.11 is_EnableAutoExit

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_EnableAutoExit (HIDS hCam, INT nMode)
```

Description

`is_EnableAutoExit()` enables automatic closing of the camera handle after a camera has been removed on-the-fly. Upon closing of the handle, the entire memory allocated by the driver will be released.

Input Parameters

hCam	Camera handle
nMode	
IS_ENABLE_AUTO_EXIT	Enables automatic closing.
IS_DISABLE_AUTO_EXIT	Disables automatic closing.
IS_GET_AUTO_EXIT_ENABLED	Returns the current setting.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
Current setting when used together with IS_GET_AUTO_EXIT_ENABLED	

Related Functions

- `is_ExitCamera()`

5.3.12 `is_EnableEvent`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_EnableEvent (HIDS hCam, INT which)
```

Description

Using `is_EnableEvent()`, you release an event object initialized with `is_InitEvent()`. Following the release, the event messages for the created event object are enabled.

Input Parameters

<code>hCam</code>	Camera handle
<code>which</code>	ID of the event to be released. See also <code>is_InitEvent()</code> .

Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message



Related Functions

- `is_InitEvent()`
- `is_ExitEvent()`
- `is_DisableEvent()`
- `is_WaitEvent()`

Sample Programs

- SimpleLive (C++)
- uEyeEvent (C++)

5.3.13 is_EnableHdr

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_EnableHdr (HIDS hCam, INT Enable)
```

Description

Some sensors support HDR (High Dynamic Range) mode. You can use the `is_EnableHdr()` function to enable / disable it. To set the knee points of the HDR curve, you can use the `is_SetHdrKneepoints()` function.

For further information on HDR mode, please refer to the [HDR properties](#) section of the *uEye Demo* chapter.



HDR mode is currently only supported by sensors of the [UI-122x/522x](#) camera models.

Input Parameters

hCam	Camera handle
Enable	
IS_ENABLE_HDR	Enables HDR mode.
IS_DISABLE_HDR	Disables HDR mode.

Return Values

IS_SUCCESS	Function executed successfully This value is also returned in case an unsupported sensor type and IS_DISABLE_HDR are used.
IS_NO_SUCCESS	General error message
IS_NOT_SUPPORTED	In case an unsupported sensor type and IS_ENABLE_HDR are used.



Related Functions

- [is_GetHdrMode\(\)](#)
- [is_SetHdrKneepoints\(\)](#)
- [is_GetHdrKneepointInfo\(\)](#)
- [is_GetHdrKneepoints\(\)](#)

Sample Programs

- [uEyeC# Hdr Demo \(C#\)](#)

5.3.14 `is_EnableMessage`

	
USB 2.0 GigE	-

Syntax

```
INT is_EnableMessage (HIDS hCam, INT which, HWND hWnd)
```

Description

Using `is_EnableMessage()`, you can enable Windows messages. If a particular event occurs, the messages are sent to the application.

Each message is structured as follows:

Message: `IS_UEYE_MESSAGE`

wParam: Event (see table)

lParam: *uEye* camera handle associated with the message

Input Parameters

hCam	Camera handle
which	ID of the message to be enabled/disabled
<code>IS_FRAME</code>	A new image is available.
<code>IS_SEQUENCE</code>	The sequence is completed.
<code>IS_TRANSFER_FAILED</code>	An error occurred during the data transfer.
<code>IS_TRIGGER</code>	An image which was captured following the arrival of a trigger has been transferred completely. This is the earliest possible moment for a new capturing process. The image must then be post-processed by the driver and is available after the <code>IS_FRAME</code> message has occurred.
<code>IS_DEVICE_REMOVED</code>	A camera initialised with <code>is_InitCamera()</code> was disconnected.
<code>IS_DEVICE_RECONNECTED</code>	A camera initialised with <code>is_InitCamera()</code> and disconnected afterwards was reconnected.
<code>IS_NEW_DEVICE</code>	A new camera was connected.
<code>IS_DEVICE_REMOVAL</code>	A camera was removed.
<code>IS_WB_FINISHED</code>	Automatic white balance control is completed (only if this control was started using the <code>IS_SET_AUTO_WB_ONCE</code> function).
<code>IS_AUTOBRIGHTNESS_FINISHED</code>	Automatic brightness control is completed (only if this control was started using the <code>IS_SET_AUTO_BRIGHTNESS_ONCE</code> function).
hWnd	Application window for receiving the message <code>NULL</code> disables the message designated by the <code>which</code> parameter.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [is_InitEvent\(\)](#)

5.3.15 `is_ExitCamera`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_ExitCamera (HIDS hCam)
```

Description

`is_ExitCamera()` disables the `hCam` camera handle and releases the data structures and memory areas taken up by the *uEye* camera. Image memory allocated using the `is_AllocImageMem()` function which has not been released yet is automatically released.



We recommend that you call the following functions only from a single thread in order to avoid unpredictable behaviour of the application.

- `is_InitCamera()`
- `is_SetDisplayMode()`
- `is_ExitCamera()`

See also [Programming: Thread Programming](#).

Input Parameters

<code>hCam</code>	Camera handle
-------------------	---------------



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message

Related Functions

- `is_InitCamera()`
- `is_EnableAutoExit()`

5.3.16 is_ExitEvent

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

INT is_ExitEvent (HIDS hCam, INT which)

Description

is_ExitEvent() deletes an existing event object. After an event has been deleted, you can no longer enable it by calling the [is_EnableEvent\(\)](#) function.

Input Parameters

hCam	Camera handle
which	ID of the event to be deleted. See also is_InitEvent() .

Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message



Related Functions

- [is_InitEvent\(\)](#)
- [is_EnableEvent\(\)](#)
- [is_WaitEvent\(\)](#)

Example

See also [is_ForceTrigger\(\)](#)

5.3.17 `is_ForceTrigger`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_ForceTrigger (HIDS hCam)
```

Description

You can use `is_ForceTrigger()` to force a software-controlled capture of an image while a capturing process triggered by hardware is in progress. This function can only be used if the triggered capturing process was started using the `IS_DONE_WAIT` parameter.

Input Parameters

<code>hCam</code>	Camera handle
-------------------	---------------

Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message

Related Functions

- [`is_FreezeVideo\(\)`](#)
- [`is_CaptureVideo\(\)`](#)
- [`is_SetExternalTrigger\(\)`](#)

Code Sample

Enable trigger and wait 1 second for the external trigger. If no trigger signal has arrived, force an exception using `is_ForceTrigger()`.



```
HANDLE hEvent = CreateEvent(NULL, TRUE, FALSE, "");
if ( hEvent != NULL )
{
    is_InitEvent(hCam, m_hEvent, IS_SET_EVENT_FRAME);
    is_EnableEvent(hCam, IS_SET_EVENT_FRAME);

    is_SetExternalTrigger(hCam, IS_SET_TRIGGER_HI_LO);
    is_FreezeVideo(hCam, IS_DONT_WAIT);

    if ( WaitForSingleObject(m_hEvent, 1000) != WAIT_OBJECT_0 )
    {
        // No trigger has been received, so force image capture
        is_ForceTrigger(hCam);
    }

    is_DisableEvent(hCam, IS_SET_EVENT_FRAME);
    is_ExitEvent(hCam, IS_SET_EVENT_FRAME);
}
```

5.3.18 is_FreeImageMem

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_FreeImageMem (HIDS hCam, char* pcImgMem, INT id)
```

Description

`is_FreeImageMem()` releases an image memory that was allocated using `is_AllocImageMem()` and removes it from the driver management.



If the memory was not allocated using an SDK function, you need to call `is_FreeImageMem()` as well. Otherwise, there may be errors when the driver keeps trying to access this memory. This does however not release the memory. So you need to make sure that the memory will be released again.

Input Parameters

hCam	Camera handle
pcImgMem	Points to the starting address of the memory (e.g. set in the <code>is_AllocImageMem()</code> function)
id	ID of this memory



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- `is_AllocImageMem()`

5.3.19 is_FreezeVideo

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_FreezeVideo (HIDS hCam, INT Wait)
```

Description

`is_FreezeVideo()` acquires a single image from the camera. In DIB mode, the image is stored in the active image memory. If ring buffering is used in DIB mode, the captured image is transferred to the next available image memory of the sequence. Once the last available sequence memory has been filled, the sequence event or message will be triggered.

In Direct3D mode, the image is directly copied to the graphics card buffer and then displayed.

Image capture will be started by a trigger if you previously enabled the trigger mode using `is_SetExternalTrigger()`. A hardware triggered image acquisition can be cancelled using `is_StopLiveVideo()` if exposure has not started yet.

For further information on the image capture modes of the *uEye* camera, see the [How To Proceed: Image Capture](#) section.

Input Parameters

hCam	Camera handle
Wait	
IS_DONT_WAIT	Timeout value for image capture (see also the How To Proceed: Timeout Values for Image Capture section)
IS_WAIT	
Time t	

Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [is_HasVideoStarted\(\)](#)
- [is_IsVideoFinish\(\)](#)
- [is_SetExternalTrigger\(\)](#)
- [is_ForceTrigger\(\)](#)
- [is_CaptureVideo\(\)](#)
- [is_SetTimeout\(\)](#)
- [is_GetCaptureErrorInfo\(\)](#)

Example



Enable trigger mode, set high-active flash mode and capture an image:

```
is_SetExternalTrigger(hCam, IS_SET_TRIGGER_SOFTWARE);  
is_SetFlashStrobe(hCam, IS_SET_FLASH_HI_ACTIVE);  
is_FreezeVideo(hCam, IS_WAIT);
```

Sample Programs

- SimpleAcquire (C++)
- uEyeC# Demo (C#)
- uEye VB Simple Demo (VB6)

5.3.20 `is_GetActiveImageMem`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_GetActiveImageMem (HIDS hCam, char** ppcMem, INT* pnID)
```

Description

`is_GetActiveImageMem()` returns the pointer to the starting address and the ID number of the active image memory.

If a Direct3D mode is active and image memory was nevertheless allocated, the pointer to the image memory and its ID will be returned. However, in Direct3D mode, the image will not be copied automatically to this image memory.

Input Parameters

<code>hCam</code>	Camera handle
<code>ppcMem</code>	Returns the pointer to the starting address of the active image memory.
<code>pnID</code>	Returns the ID of the active image memory.



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message

Related Functions

- [`is_AllocImageMem\(\)`](#)
- [`is_GetImageMem\(\)`](#)
- [`is_SetImageMem\(\)`](#)
- [`is_SetAllocatedImageMem\(\)`](#)

5.3.21 is_GetActSeqBuf

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_GetActSeqBuf (HIDS hCam, INT* pnNum,
                    char** ppcMem, char** ppcMemLast);
```

Description

Using `is_GetActSeqBuf()`, you can determine the image memory which is currently used for capturing an image (`ppcMem`) or the image memory that was last used for capturing an image (`ppcMemLast`). This function is only available if you have enabled ring buffering.



This number is not the ID of the image memory that was allocated using the `is_AllocImageMem()` function, but the running number from the order in which memory was allocated by the `is_AddToSequence()` function.

Input Parameters

<code>hCam</code>	Camera handle
<code>pnNum</code>	Contains the number of the image memory currently used for image capturing. If image capturing is already in progress when <code>is_GetActSeqBuf()</code> is called, <code>pnNum</code> will return the value 0 until the sequence arrives at the first image memory again.
<code>ppcMem</code>	Contains the starting address of the image memory currently used for image capturing.
<code>ppcMemLast</code>	Contains the starting address of the image memory last used for image capturing.



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message

Related Functions

- [`is_AddToSequence\(\)`](#)
- [`is_GetImageMem\(\)`](#)

5.3.22 is_GetAutoInfo

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_GetAutoInfo (HIDS hCam, UEYE_AUTO_INFO* pInfo)
```

Description

Using the `is_GetAutoInfo()` function, you can query status information of the auto control features. This information is written to the `UEYE_AUTO_INFO` structure.

For further information on automatic control, please refer to the Functions for Automatic Image Control chapter.



The status information returned in the `UEYE_AUTO_INFO` structure is only valid if at least one of the auto control features has been enabled using `is_SetAutoParameter()`.

Input Parameters

hCam	Camera handle
pinfo	UEYE_AUTO_INFO structure (see below)

Contents of the UEYE_AUTO_INFO Structure

INT	AutoAbility	Supported auto control features
	AC_SHUTTER	Auto Shutter available
	AC_SENSOR_SHUTTER	Sensor based Auto Shutter available
	AC_FRAMERATE	Auto Frame Rate available
	AC_SENSOR_FRAMERATE	Sensor based Auto Frame Rate available
	AC_GAIN	Auto Gain available
	AC_SENSOR_GAIN	Sensor based Auto Gain available
	AC_WHITEBAL	Auto White Balance available
AUTO_BRIGHT_STATUS	sBrightCtrlStatus	See AUTO_BRIGHT_STATUS
AUTO_WB_STATUS	sWBCtrlStatus	See AUTO_WB_STATUS
DWORD	reserved	Reserved space for extensions

Contents of the UEYE_AUTO_INFO::AUTO_BRIGHT_STATUS Structure

INT	curValue	Current average greyscale value (actual value)
INT	curError	Current control deviation (error)
INT	curController	Current parameter value
	AC_SHUTTER	Exposure time (shutter)
	AC_GAIN	Gain
INT	curCtrlStatus	Current control status
	ACS_ADJUSTING	Control is active.
	ACS_FINISHED	Control is completed.
	ACS_DISABLED	Control is disabled.

Contents of the UEYE_AUTO_INFO::AUTO_WB_STATUS Structure

INT	curController	Current white balance control
	AC_WB_RED_CHANNEL	Value of the red channel
	AC_WB_GREEN_CHANNEL	Value of the green channel
	AC_WB_BLUE_CHANNEL	Value of the blue channel
AUTO_WB_CHANNEL_STATUS	RedChannel	See AUTO_WB_CHANNEL_STATUS
AUTO_WB_CHANNEL_STATUS	GreenChannel	See AUTO_WB_CHANNEL_STATUS
AUTO_WB_CHANNEL_STATUS	BlueChannel	See AUTO_WB_CHANNEL_STATUS

**Contents of the UEYE_AUTO_INFO::AUTO_WB_STATUS::
AUTO_WB_CHANNEL_STATUS Structure**

INT	curValue	Current average greyscale value (actual value)
INT	curError	Current control deviation (error)
INT	curCtrlStatus	Current control status
	ACS_ADJUSTING	Control is active.
	ACS_FINISHED	Control is completed.
	ACS_DISABLED	Control is disabled.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [is_SetAutoParameter\(\)](#)

5.3.23 is_GetBusSpeed

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_GetBusSpeed (HIDS hCam)
```

Description

Using `is_GetBusSpeed()`, you can query whether a camera is connected to a USB 2.0 host controller.

When the value 0 is passed for `hCam`, the function checks whether a USB 2.0 controller is present in the system.



Input Parameters

hCam	Camera handle
------	---------------

Return Values

IS_SUCCESS	
IS_NO_SUCCESS	Only if <code>hCam=0</code> is passed: No USB 2.0 controller present
IS_USB_10	The controller to which the camera is connected does not support USB 2.0.
IS_USB_20	The camera is connected to a USB 2.0 controller.

5.3.24 is_GetCameraInfo

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_GetCameraInfo (HIDS hCam, CAMINFO* pInfo)
```

Description

`is_GetCameraInfo()` reads out the data hard-coded in the EEPROM and writes it to the data structure that `pInfo` points to.



The information from this data structure should not be used to find a specific camera (e.g. in order to control this specific camera). Instead, we recommend identifying a camera by a fixed camera ID or by the sensor ID (see `is_GetCameraList()`).

Contents of the CAMINFO Structure

char	SerNo[12]	Serial number of the camera	
char	ID[20]	Manufacturer of the camera e.g. "IDS GmbH"	
char	Version[10]	For USB cameras, this value indicates the USB board hardware version. e.g. "v2.10"	
char	Date[12]	System date of the final quality check e.g. "01.08.2008" (DD.MM.YYYY)	
unsigned char	Select	Camera ID	
unsigned char	Type	Camera type IS_CAMERA_TYPE_UEYE_USB_SE IS_CAMERA_TYPE_UEYE_USB_ME IS_CAMERA_TYPE_UEYE_USB_RE IS_CAMERA_TYPE_UEYE_USB_LE IS_CAMERA_TYPE_UEYE_ETH_HE IS_CAMERA_TYPE_UEYE_ETH_SE	USB uEye SE USB uEye ME USB uEye RE USB uEye LE GigE uEye HE GigE uEye SE
char	Reserved[8]	Reserved	

Input Parameters

hCam	Camera handle
pInfo	Pointer to a CAMINFO data structure



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- is_GetCameraType()
- is_CameraStatus()

5.3.25 is_GetCameraList

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_GetCameraList (UEYE_CAMERA_LIST* pucl)
```

Description

Using `is_GetCameraList()`, you can query information about the connected cameras. To get all information that is available, you need to adjust the field size to the number of connected cameras. The following tables explain the structures used for that purpose.

Input Parameters

pucl	Handle to the UEYE_CAMERA_LIST structure
------	--

Contents of the UEYE_CAMERA_LIST Structure

ULONG	dwCount	Number of cameras connected to the system
UEYE_CAMERA_INFO	uci[1]	Placeholder for 1 .. n UEYE_CAMERA_INFO structures

Contents of the UEYE_CAMERA_LIST::UEYE_CAMERA_INFO Structure

DWORD	dwCameraID	Customisable camera ID. This ID is stored in the camera and is persistent.
DWORD	dwDeviceID	Internal device ID. This ID is generated by the driver depending on order of connection and camera type. The device ID is not persistent.
DWORD	dwSensorID	Sensor ID
DWORD	dwInUse	1 = camera is being used. 0 = camera is not being used.
Char	SerNo[16]	Serial number of the camera *)
Char	Model[16]	Camera model *)
DWORD	dwReserved[16]	Reserved for later use



*) The information from this data structure should not be used to find a specific camera (e.g. in order to control this specific camera). Instead, we recommend identifying a camera by a fixed camera ID or by the sensor ID.

Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
IS_ACCESS_VIOLATION	Not enough memory allocated for the UEYE_CAMERA_LIST structure
IS_CANT_OPEN_DEVICE	Camera cannot be selected or initialised.
IS_IO_REQUEST_FAILED	Driver communication failed.

Related Functions

- [is_GetNumberOfCameras\(\)](#)



Code Sample

```
// At least one camera must be available
INT nNumCam;
if( is_GetNumberOfCameras( &nNumCam ) == IS_SUCCESS)
{
    if( nNumCam >= 1 )
    {
        // Create new list with suitable size
        UEYE_CAMERA_LIST* pucl;
        pucl = (UEYE_CAMERA_LIST*) new char [sizeof (DWORD)
                                           + nNumCam
                                           * sizeof
                                           (UEYE_CAMERA_INFO)];

        pucl->dwCount = nNumCam;

        //Retrieve camera info
        if (is_GetCameraList(pucl) == IS_SUCCESS)
        {
            int iCamera;
            for (iCamera = 0; iCamera < (int)pucl->dwCount;
                iCamera++)
            {
                //Test output of camera info on the screen
                printf("Camera %i Id: %d", iCamera,
                    pucl->uci[iCamera].dwCameraID);
            }
        }
    }
}
```

5.3.26 is_GetCameraLUT

	
GigE	GigE

Syntax

```
INT is_GetCameraLUT (HIDS hCam,
                    UINT Mode, UINT NumberOfEntries,
                    double* pRed_Grey, double* pGreen, double* pBlue)
```

Description

`is_GetCameraLUT()` returns the current LUT values. Using the `is_SetCameraLUT()` function, you can select a different LUT for the camera.



The `is_GetCameraLUT()` function is only supported by cameras of the *GigE uEye* series.

Input Parameters

<code>hCam</code>	Camera handle
<code>Mode</code>	
<code>IS_GET_CAMERA_LUT_USER</code>	Returns the LUT values set by the user without modifications.
<code>IS_GET_CAMERA_LUT_COMPLETE</code>	Returns the LUT values set by the user after the gamma, contrast and brightness values have been taken into account.
<code>NumberOfEntries</code>	Number of the LUT values
<code>IS_CAMERA_LUT_64</code>	LUT with 64 values
<code>pRed_Grey</code>	Pointer to the array to which the red channel values or the greyscale value (<i>GigE uEye SE</i> cameras) of the LUT are written.
<code>pGreen</code>	Pointer to the array to which the green channel values of the LUT are written.
<code>pBlue</code>	Pointer to the array to which the blue channel values of the LUT are written.



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message
<code>IS_NOT_SUPPORTED</code>	This function is not supported by the current camera.

Related Functions

- `is_SetCameraLUT()`

5.3.27 is_GetCameraType

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

INT is_GetCameraType (HIDS hCam)

Description

is_GetCameraType() returns the camera type.

Input Parameters

hCam	Camera handle
------	---------------



Return Values

IS_CAMERA_TYPE_UEYE_USB_SE	USB uEye SE camera
IS_CAMERA_TYPE_UEYE_USB_ME	USB uEye ME camera
IS_CAMERA_TYPE_UEYE_USB_RE	USB uEye RE camera
IS_CAMERA_TYPE_UEYE_USB_LE	USB uEye LE camera
IS_CAMERA_TYPE_UEYE_ETH_HE	GigE uEye HE camera
IS_CAMERA_TYPE_UEYE_ETH_SE	GigE uEye SE camera

Related Functions

- [is_GetCameraInfo\(\)](#)

5.3.28 is_GetCaptureErrorInfo

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_GetCaptureErrorInfo (HIDS hCam,
                           UEYE_CAPTURE_ERROR_INFO* CaptureErrorInfo,
                           UINT SizeCaptureErrorInfo)
```

Description

`is_GetCaptureErrorInfo()` returns detail information on errors that occurred during an image capture process. The function lists all errors that occurred since the last call of the `is_ResetCaptureErrorInfo()` function.

Input Parameters

<code>hCam</code>	Camera handle
<code>CaptureErrorInfo</code>	Structure of the <code>UEYE_CAPTURE_ERROR_INFO</code> type that is filled by the <i>uEye</i> driver. This structure then contains the error list.
<code>SizeCaptureErrorInfo</code>	Indicates the size of the <code>CaptureErrorInfo</code> structure.

Contents of the UEYE_CAPTURE_ERROR_INFO Structure

DWORD	<code>dwCapErrCnt_Total</code>	Returns the total number of errors occurred since the last reset.
BYTE	<code>reserved[60]</code>	Reserved for an internal function
DWORD	<code>adwCapErrCnt_Detail[CapErr]</code>	This array returns the current count for each possible error. The possible errors are listed below. To query the counter of a specific error type, pass its definition in the <code>CapErr</code> parameter.

Possible CapErr Error Types

Definition	Description	#
<code>IS_CAPERR_API_NO_DEST_MEM</code>	There is no destination memory for copying the finished image.	1
<code>IS_CAPERR_API_CONVERSION_FAILED</code>	The current image could not be processed correctly.	2
<code>IS_CAPERR_API_IMAGE_LOCKED</code>	The destination buffers are locked and could not be written to.	3
<code>IS_CAPERR_DRV_OUT_OF_BUFFERS</code>	No free internal image memory is available to the driver. The image was discarded.	4
<code>IS_CAPERR_DRV_DEVICE_NOT_READY</code>	The camera is no longer available. It is not possible to access images that have already been transferred.	5
<code>IS_CAPERR_USB_TRANSFER_FAILED</code>	The image was not transferred over the USB bus.	6
<code>IS_CAPERR_DEV_TIMEOUT</code>	The maximum allowable time for image capturing in	7

	the camera was exceeded.	
IS_CAPERR_ETH_BUFFER_OVERRUN	The sensor transfers more data than the internal camera memory of the <i>GigE uEye</i> can accommodate.	8
IS_CAPERR_ETH_MISSED_IMAGES	The <i>GigE uEye</i> camera could neither process nor output an image captured by the sensor.	9

#	Possible cause	Remedy
1	Not enough destination memory allocated or all destination buffers locked by the application	<ul style="list-style-type: none"> • Release locked destination memory • Allocate more destination memory • Reduce the frame rate so that there is more time to process the filled destination memory
2	Internal error during internal processing of the image	-
3	All destination buffers locked by the application	<ul style="list-style-type: none"> • Release locked destination memory • Allocate more destination memory • Reduce the frame rate so that there is more time to process the filled destination memory
4	The computer takes too long to process the images in the <i>uEye API</i> (e.g. colour conversion)	<ul style="list-style-type: none"> • Reduce the frame rate so that there is more time to process the filled image memory of the driver • Disable resource-intensive API image pre-processing functions (e.g. edge enhancement, colour correction, choose smaller filter mask for software colour conversion)
5	The camera has been disconnected or closed	-
6	Not enough free bandwidth on the USB bus for transferring the image	<ul style="list-style-type: none"> • Reduce the pixel clock frequency • Operate fewer cameras simultaneously on a USB bus • Check the quality of the USB cabling and components
7	The selected timeout value is too low for image capture	<ul style="list-style-type: none"> • Reduce the exposure time • Increase the timeout
8	The selected data rate of the sensor is too high	<ul style="list-style-type: none"> • Reduce the pixel clock frequency • Reduce the frame rate • Reduce the image size
9	The camera's frame rate is too high or the bandwidth on the network is insufficient to transfer the image	<ul style="list-style-type: none"> • Reduce the frame rate • Increase the value for the receive descriptors in the network card settings



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [is_ResetCaptureErrorInfo\(\)](#)
- [is_GetError\(\)](#)
- [is_CameraStatus\(\)](#)
- [is_SetErrorReport\(\)](#)

5.3.29 `is_GetColorConverter`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```

INT is_GetColorConverter (HIDS hCam,
                        INT ColorMode,
                        INT* pCurrentConvertMode,
                        INT* pDefaultConvertMode,
                        INT* pSupportedConvertModes)

```

Description

For colour cameras, `is_GetColorConverter` returns the set mode or all available Bayer conversion modes for the specified colour mode. The return value depends on the selected colour mode. For further information, please refer to the [Appendix: Colour and Memory Formats](#) section.

Input Parameters

<code>hCam</code>	Camera handle
<code>ColorMode</code>	Colour mode for which the converter is to be returned For a list of all available colour formats and the associated input parameters, see the Appendix: Colour and Memory Formats section.
<code>pCurrentConvertMode</code>	Currently selected converter for this colour mode
<code>pDefaultConvertMode</code>	Default converter for this colour mode
<code>pSupportedConvertModes</code>	All converters supported for this colour mode



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message
<code>IS_INVALID_COLOR_FORMAT</code>	The <code>ColorMode</code> parameter is invalid or not supported.

Related Functions

- [`is_SetColorConverter\(\)`](#)
- [`is_SetColorMode\(\)`](#)

5.3.30 is_GetColorDepth

	
USB 2.0 GigE	-

Syntax

```
INT is_GetColorDepth(HIDS hCam, INT* pnCol, INT* pnColMode)
```

Description

`is_GetColorDepth()` retrieves the current Windows Desktop colour setting and returns the bit depth per pixel and the matching *uEye* colour mode. The colour mode can be passed directly to the `is_SetColorMode()` function. You need to pass the bit depth when allocating an image memory.

Input Parameters

hCam	Camera handle
PnCol	Returns the bit depth of the colour setting.
pnColMode	Returns the <i>uEye</i> colour mode that corresponds to <code>pnCol</code> . For a list of all available colour formats and the associated input parameters, see the Appendix: Colour and Memory Formats section.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [is_SetColorMode\(\)](#)
- [is_AllocImageMem\(\)](#)

5.3.31 is_GetComportNumber

	
GigE	GigE

Syntax

```
INT is_GetComportNumber(HIDS hCam, UINT* pComportNumber)
```

Description

`is_GetComportNumber()` returns the current COM port number of a *GigE uEye HE* camera. The default port number is 100. You can change the port number in the Camera Manager. For further information, please also refer to the [Camera Basics: Serial Interface](#) chapter.



The `is_GetComportNumber()` function is only supported by cameras of the *GigE uEye HE* series.



Input Parameters

<code>hCam</code>	Camera handle
<code>pComportNumber</code>	Pointer to the variable that is supposed to contain the port number

Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message
<code>IS_NOT_SUPPORTED</code>	This function is not supported by the camera

5.3.32 is_GetDLLVersion

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_GetDLLVersion()
```

Description

Returns the version of the *ueye_api.dll*.

Input Parameters

<none>

Return Values



The return value contains the version number which is coded as follows:

Bits 31-24:	Major version
Bits 23-16:	Minor version
Bits 15-0:	Build version

Related Functions

- [is_GetOsVersion\(\)](#)

5.3.33 `is_GetDuration`

	
GigE	GigE

Syntax

```
INT is_GetDuration (HIDS hCam, UINT nMode, INT* pnTime)
```

Description

Using `is_GetDuration()`, you can read out the estimated time it will take the *uEye* driver to execute specific processes (e.g. update the camera firmware).

Input Parameters

<code>hCam</code>	Camera handle
<code>nMode</code>	
<code>IS_SE_STARTER_FW_UPLOAD</code>	Estimated time for uploading the starter firmware to a <i>GigE uEye SE</i> camera
<code>pnTime</code>	Returns the estimated time in ms



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message

Related Functions

- [`is_InitCamera\(\)`](#)
- [`Is_SetStarterFirmware\(\)`](#)

5.3.34 is_GetEthDeviceInfo

	
GigE	GigE

Syntax

```
INT is_GetEthDeviceInfo (HIDS hDev,
                        UEYE_ETH_DEVICE_INFO* pDeviceInfo,
                        UINT uStructSize)
```



At a Glance

- [Description](#)
- [Input Parameters](#)
- [Contents of the UEYE_ETH_DEVICE_INFO Structure](#)
- [Contents of the UEYE_ETH_DEVICE_INFO_HEARTBEAT Structure](#)
- [Status Flags in UEYE_ETH_DEVICE_INFO_HEARTBEAT::dwStatus](#)
- [Contents of the UEYE_ETH_DEVICE_INFO_CONTROL Structure](#)
- [Status Flags in UEYE_ETH_DEVICE_INFO_CONTROL::dwControlStatus](#)
- [Contents of the UEYE_ETH_ADAPTER_INFO Structure](#)
- [Value Range of UEYE_ETH_ADAPTER_INFO::wPacketFilter](#)
- [Contents of the UEYE_ETH_DRIVER_INFO Structure](#)
- [Return Values](#)
- [Related Functions](#)
- [Code Sample](#)

Description

Using `is_GetEthDeviceInfo`, you can query information about the connected *Gigabit Ethernet uEye* cameras. The resulting information is written to the `UEYE_ETH_DEVICE_INFO` structure. For this purpose, the cameras need not be initialised.



The `is_GetEthDeviceInfo()` function is only supported by cameras of the *GigE uEye* series.

Input Parameters

hDev	DevID IS_USE_DEVICE_ID, DevID = internal device ID of the camera from the UEYE_CAMERA_INFO structure (see also is_GetCameraList())
pDeviceInfo	Pointer to a UEYE_ETH_DEVICE_INFO object
uStructSize	Size of the UEYE_ETH_DEVICE_INFO structure in bytes



The `is_GetEthDeviceInfo()` function does not accept a camera handle as the `hDev` parameter. As stated above, the internal device ID must be used for the call. The advantage is that you can also query information related to *Gigabit Ethernet uEye* cameras that are currently not initialised.

This information can also be queried using the *uEye Camera Manager*.

Contents of the UEYE_ETH_DEVICE_INFO Structure

UEYE_ETH_DEVICE_INFO_HEARTBEAT	infoDevHeartbeat	Camera-related data retrieved from the camera (from the heartbeat telegram) See below: DEVICE_INFO_HEARTBEAT
UEYE_ETH_DEVICE_INFO_CONTROL	infoDevControl	Camera-related driver data See below: DEVICE_INFO_CONTROL
UEYE_ETH_ADAPTER_INFO	infoAdapter	Network-card related driver data See below: ADAPTER_INFO
UEYE_ETH_DRIVER_INFO	infoDriver	General driver data See below: DRIVER_INFO

Contents of the UEYE_ETH_DEVICE_INFO::UEYE_ETH_DEVICE_INFO_HEARTBEAT Structure

BYTE	bySerialNumber[12]	Serial number (string)
BYTE	byDeviceType	Type of camera series (0x80 for <i>Gigabit Ethernet uEye</i>)
BYTE	byCameraID	User-defined camera ID
WORD	wSensorID	Sensor ID
WORD	wSizeImgMem_MB	Image memory size in MB
BYTE	reserved_1[2]	reserved
DWORD	dwVerStarterFirmware	Starter firmware version
DWORD	dwVerRuntimeFirmware	Runtime firmware version
DWORD	dwStatus	Status word
BYTE	reserved_2[4]	reserved
WORD	wTemperature	Camera temperature in degrees Celsius Bits 15: Algebraic sign Bits 10...4: Temperature (places before the decimal point) Bits 3...0: Temperature (places after the decimal point) See conversion example
WORD	wLinkSpeed_Mb	Link bandwidth in Mbits/s
UEYE_ETH_ADDR_MAC	macDevice	MAC address of the camera
BYTE	reserved_3[2]	reserved
UEYE_ETH_IP_CONFIGURATION	ipcfgPersistentIpCfg	Persistent IP configuration

UEYE_ETH_IP_CONFIGURATION	ipcfgCurrentIpCfg	Current IP configuration
UEYE_ETH_ADDR_MAC	macPairedHost	MAC address of the connected PC, if any
BYTE	reserved_4[2]	reserved
UEYE_ETH_ADDR_IPV4	ipPairedHostIp	IP address of the connected PC, if any
UEYE_ETH_ADDR_IPV4	ipAutoCfgIpRangeBegin	First IP address of the auto configuration range
UEYE_ETH_ADDR_IPV4	ipAutoCfgIpRangeEnd	Last IP address of the auto configuration range
BYTE	abyUserSpace[8]	The first eight bytes of the user EEPROM
BYTE	reserved_5[84]	reserved
BYTE	reserved_6[64]	reserved

Status Flags in UEYE_ETH_DEVICE_INFO::UEYE_ETH_DEVICE_INFO_HEARTBEAT::dwStatus

IS_ETH_DEVSTATUS_READY_TO_OPERATE	Camera is ready to operate
IS_ETH_DEVSTATUS_TESTING_IP_CURRENT	Camera is testing current IP address
IS_ETH_DEVSTATUS_TESTING_IP_PERSISTENT	Camera is testing persistent IP address
IS_ETH_DEVSTATUS_TESTING_IP_RANGE	Camera is testing auto config IP range
IS_ETH_DEVSTATUS_INAPPLICABLE_IP_CURRENT	Current IP address already assigned on the network
IS_ETH_DEVSTATUS_INAPPLICABLE_IP_PERSISTENT	Persistent IP address already assigned on the network
IS_ETH_DEVSTATUS_INAPPLICABLE_IP_RANGE	IP addresses of auto config IP range already assigned on the network
IS_ETH_DEVSTATUS_UNPAIRED	Camera has not been initialised (paired)
IS_ETH_DEVSTATUS_PAIRING_IN_PROGRESS	Camera is being initialised (paired)
IS_ETH_DEVSTATUS_PAIRED	Camera has been initialised (paired)
IS_ETH_DEVSTATUS_FORCE_100MBPS	Camera configured for 100 Mbps/s
IS_ETH_DEVSTATUS_NO_COMPORT	Camera supports no uEye COM port
IS_ETH_DEVSTATUS_RECEIVING_FW_STARTER	Camera is receiving starter firmware

IS_ETH_DEVSTATUS_RECEIVING_FW_RUNTIME	Camera is receiving runtime firmware
IS_ETH_DEVSTATUS_INAPPLICABLE_FW_RUNTIME	Runtime firmware cannot be used
IS_ETH_DEVSTATUS_INAPPLICABLE_FW_STARTER	Starter firmware cannot be used
IS_ETH_DEVSTATUS_REBOOTING_FW_RUNTIME	Camera is rebooting runtime firmware
IS_ETH_DEVSTATUS_REBOOTING_FW_STARTER	Camera is rebooting starter firmware
IS_ETH_DEVSTATUS_REBOOTING_FW_FAILSAFE	Camera is rebooting failsafe firmware
IS_ETH_DEVSTATUS_RUNTIME_FW_ERR0	Checksum error (error 0) in runtime firmware

Contents of the UEYE_ETH_DEVICE_INFO::UEYE_ETH_DEVICE_INFO_CONTROL Structure

DWORD	dwDeviceID	Internal device ID of the camera
DWORD	dwControlStatus	Status word for driver-based camera management (see below)
BYTE	reserved_1[80]	reserved
BYTE	reserved_2[64]	reserved

Status Flags in UEYE_ETH_DEVICE_INFO::UEYE_ETH_DEVICE_INFO_CONTROL::dwControlStatus

IS_ETH_CTRLSTATUS_AVAILABLE	The camera is available
IS_ETH_CTRLSTATUS_ACCESSIBLE1	Camera has valid IP address and can be accessed over the network
IS_ETH_CTRLSTATUS_ACCESSIBLE2	Camera has no persistent IP address; the auto IP range is valid
IS_ETH_CTRLSTATUS_PERSISTENT_IP_USED	Camera can be accessed over the network by its persistent IP address
IS_ETH_CTRLSTATUS_COMPATIBLE	Camera is compatible with the installed driver
IS_ETH_CTRLSTATUS_ADAPTER_ON_DHCP	DHCP is enabled on the PC network card
IS_ETH_CTRLSTATUS_UNPAIRING_IN_PROGRESS	Camera is being closed on this PC
IS_ETH_CTRLSTATUS_PAIRING_IN_PROGRESS	Camera is being initialised on this PC
IS_ETH_CTRLSTATUS_PAIED	Camera has been initialised on this PC

IS_ETH_CTRLSTATUS_FW_UPLOAD_STARTER	Starter firmware is being loaded onto the camera
IS_ETH_CTRLSTATUS_FW_UPLOAD_RUNTIME	Runtime firmware is being loaded onto the camera
IS_ETH_CTRLSTATUS_REBOOTING	Camera is rebooting
IS_ETH_CTRLSTATUS_INITIALIZED	Camera has been initialised in the driver
IS_ETH_CTRLSTATUS_TO_BE_DELETED	Camera is being removed from driver management
IS_ETH_CTRLSTATUS_TO_BE_REMOVED	Camera is being removed from driver management

Contents of the UEYE_ETH_DEVICE_INFO::Structure

DWORD	dwAdapterID	Network adapter ID as defined internally in the driver
DWORD	dwDeviceLinkspeed	Possible values: IS_ETH_LINKSPEED_100MB = 100 Mbits/s IS_ETH_LINKSPEED_1000MB = 1000 Mbits/s
UEYE_ETH_ETHERNET_CONFIGURATION	ethcfg	Ethernet configuration of the network adapter
BYTE	reserved_2[2]	reserved
BOOL	bIsEnabledDHCP	The adapter is configured for DHCP
UEYE_ETH_AUTOCFG_IP_SETUP	autoCfgIp	Setting of the IP address auto configuration
BOOL	bIsValidAutoCfgIpRange	The IP auto configuration setting is valid
DWORD	dwCntDevicesKnown	Number of cameras detected at this network adapter
DWORD	dwCntDevicesPaired	Number of cameras initialised using this network adapter
WORD	wPacketFilter	Filter settings for incoming packets (see below)
BYTE	reserved_3[38]	reserved
BYTE	reserved_4[64]	reserved

Value Range of UEYE_ETH_DEVICE_INFO::UEYE_ETH_ADAPTER_INFO::wPacketFilter

IS_ETH_PCKTFLT_PASSALL	Forward all packets to the operating system.
IS_ETH_PCKTFLT_BLOCKUEGET	Block <i>Gigabit Ethernet uEye</i> data packets directed to the operating system (recommended).
IS_ETH_PCKTFLT_BLOCKALL	Block all packets directed to the operating system.

Contents of the `UEYE_ETH_DEVICE_INFO::UEYE_ETH_DRIVER_INFO` Structure

DWORD	<code>dwMinVerStarterFirmware</code>	Minimum compatible starter firmware version
DWORD	<code>dwMaxVerStarterFirmware</code>	Maximum compatible starter firmware version
BYTE	<code>reserved_1[8]</code>	reserved
BYTE	<code>reserved_2[64]</code>	reserved

Return Values

<code>IS_SUCCESS</code>	Data was read without errors.
<code>IS_INVALID_PARAMETER</code>	The <code>pDeviceInfo</code> parameter is invalid.
<code>IS_BAD_STRUCTURE_SIZE</code>	The structure size you specified is invalid.
<code>IS_NOT_SUPPORTED</code>	<code>hCam</code> was not designated as a device ID or the device ID specified is not supported by the <i>Gigabit Ethernet uEye</i> .
<code>IS_CANT_OPEN_DEVICE</code>	Driver could not be found.
<code>IS_IO_REQUEST_FAILED</code>	Driver communication failed.

Related Functions

- `is_GetCameraList()`

Code Sample



```
//Create the structure
UEYE_ETH_DEVICE_INFO di;

//Create specific camera handle from the internal device ID, see info
in the box above
HIDS hDev = (HIDS)(dwDeviceID | IS_USE_DEVICE_ID);

//Populate the structure with Gigabit Ethernet uEye information
INT nRet = is_GetEthDeviceInfo( hDev, &di, sizeof(UEYE_ETH_DEVICE_INFO));

//Read out and convert camera temperature
DWORD dwTempFahrenheit;
DWORD dwTempCelsius;
dwTempCelsius = (float) di.infoDevHeartbeat.wTemperature / 16
dwTempFahrenheit = ((float) di.infoDevHeartbeat.wTemperature * 0.1125) + 32
```

5.3.35 is_GetError

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_GetError (HIDS hCam, INT* pErr, IS_CHAR** ppcErr)
```

Description

`is_GetError()` queries the last error that occurred and returns the associated error code and message. We recommend to use this function after an error has occurred that returned `IS_NO_SUCCESS`. Each error message will be overwritten when a new error occurs.

Input Parameters

hCam	Camera handle
PErr	Pointer to the variable containing the error code
PpcErr	Pointer to the string containing the error text



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [is_GetCaptureErrorInfo\(\)](#)
- [is_SetErrorReport\(\)](#)
- [is_CameraStatus\(\)](#)

5.3.36 `is_GetExposureRange`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_GetExposureRange (HIDS hCam,
                        double* min, double* max, double* intervall)
```

Description

Using `is_GetExposureRange()`, you can query the exposure values (in milliseconds) available for the currently selected timing (pixel clock, frame rate). The available time values are comprised between `min` and `max` and can be set in increments defined by the `intervall` parameter.

Input Parameters

<code>hCam</code>	Camera handle
<code>min</code>	Returns the minimum available exposure time.
<code>max</code>	Returns the maximum available exposure time.
<code>intervall</code>	Returns the increment you can use to change the image exposure time.



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message

Related Functions

- [`is_SetExposureTime\(\)`](#)
- [`is_GetPixelClockRange\(\)`](#)
- [`is_GetFrameTimeRange\(\)`](#)
- [`is_GetFramesPerSecond\(\)`](#)
- [`is_SetFrameRate\(\)`](#)

5.3.37 is_GetFramesPerSecond

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_GetFramesPerSecond (HIDS hCam, double* dblFPS)
```

Description

In live capture mode started by [is_CaptureVideo\(\)](#), the [is_GetFramesPerSecond\(\)](#) function returns the number of frames actually captured per second.

Input Parameters

hCam	Camera handle
dblFPS	Returns the current frame rate.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [is_GetFrameTimeRange\(\)](#)
- [is_SetFrameRate\(\)](#)
- [is_GetPixelClockRange\(\)](#)
- [is_GetExposureRange\(\)](#)
- [is_GetExposureTime\(\)](#)

5.3.38 `is_GetFrameTimeRange`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_GetFrameTimeRange (HIDS hCam,
                          double* min, double* max, double* intervall)
```

Description

Using `is_GetFrameTimeRange()`, you can read out the frame rate settings which are available for the current pixel clock setting. The returned values indicate the minimum and maximum frame duration in seconds. You can set the frame duration between `min` and `max` in increments defined by the `intervall` parameter.

The following applies:

$$\text{fps}_{\min} = 1/\text{max}$$

$$\text{fps}_{\max} = 1/\text{min}$$

$$\text{fps}_n = 1/(\text{min} + n * \text{intervall})$$



The use of the following functions will affect the frame duration:

- [`is_SetPixelClock\(\)`](#)
- [`is_SetOptimalCameraTiming\(\)`](#)
- [`is_SetAOI\(\)`](#) (if the image size is changed)
- [`is_SetSubSampling\(\)`](#)
- [`is_SetBinning\(\)`](#)

Changes made to the window size, the frame rate or the read-out timing (pixel clock frequency) also affect the defined frame duration. For this reason, you need to call `is_GetFrameTimeRange()` again after such changes.

Input Parameters

<code>hCam</code>	Camera handle
<code>min</code>	Returns the minimum available frame duration.
<code>max</code>	Returns the maximum available frame duration.
<code>intervall</code>	Returns the increment you can use to change the frame duration.



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message

Related Functions

- [`is_GetFramesPerSecond\(\)`](#)
- [`is_SetFrameRate\(\)`](#)
- [`is_GetPixelClockRange\(\)`](#)
- [`is_GetExposureRange\(\)`](#)
- [`is_GetExposureTime\(\)`](#)

5.3.39 is_GetGlobalFlashDelays

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_GetGlobalFlashDelays (HIDS hCam,
                             ULONG* pulDelay,
                             ULONG* pulDuration)
```

Description

Rolling shutter cameras:

Using `is_GetGlobalFlashDelays()`, you can determine the times required to implement a global flash function for rolling shutter cameras. This way, a rolling shutter camera can also be used as a global shutter camera provided that no ambient light falls on the sensor outside the flash period. If the exposure time is set too short so that no global flash operation is possible, the function returns `IS_NO_SUCCESS`.



To use a rolling shutter camera with the Global Start function, call the `is_SetGlobalShutter()` function before `is_GetGlobalFlashDelays()`. Otherwise, incorrect values will be returned for `Delay` and `Duration`.

Global shutter cameras:

In freerun mode, the exposure of global shutter cameras is delayed if the exposure time is not set to the maximum value. `is_GetGlobalFlashDelays()` determines the required delay in order to synchronise exposure and flash operation. In triggered mode, the return values for delay and flash duration are 0, since no delay is necessary before exposure starts.

For further information, please refer to the [Camera Basics: Shutter Methods](#) chapter.

Input Parameters

<code>hCam</code>	Camera handle
<code>pulDelay</code>	Pointer to the variable that returns the flash delay in μ s.
<code>pulDuration</code>	Pointer to the variable that returns the flash duration in μ s.



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message

Related Functions

- [is_SetFlashStrobe\(\)](#)
- [is_SetFlashDelay\(\)](#)
- [is_SetTriggerDelay\(\)](#)

5.3.40 is_GetHdrKneepointInfo

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_GetHdrKneepointInfo (HIDS hCam,
                           KNEEPOINTINFO* KneepointInfo,
                           INT KneepointInfoSize)
```

Description

Some sensors support HDR mode (High Dynamic Range). You can use the [is_EnableHdr\(\)](#) function to enable / disable it. Using [is_GetHdrKneepointinfo\(\)](#), you can query general information on the knee points. It is returned in a KNEEPOINTINFO structure.

Input Parameters

hCam	Camera handle
KneepointInfo	Pointer to a structure
KneepointInfoSize	Size of the structure

Contents of the KneepointInfo Structure

INT NumberOfSupportedKneepoints	Maximum number of supported knee points
INT NumberOfUsedKneepoints	Currently used number of knee points
double MinValueX	Minimum X value of a knee point
double MaxValueX	Maximum X value of a knee point
double MinValueY	Minimum Y value of a knee point
double MaxValueY	Maximum Y value of a knee point
INT Reserved[10]	Not used



Return Values

IS_SUCCESS	Function executed successfully (supported sensor type)
IS_NO_SUCCESS	General error message (supported sensor type)
IS_NOT_SUPPORTED	Unsupported sensor type

Related Functions

- [is_GetHdrMode\(\)](#)
- [is_GetHdrKneepoints\(\)](#)
- [is_SetHdrKneepoints\(\)](#)
- [is_EnableHdr\(\)](#)

5.3.41 is_GetHdrKneepoints

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_GetHdrKneepoints (HIDS hCam,
                        KNEEPOINTARRAY* KneepointArray,
                        INT KneepointArraySize)
```

Description

Some sensors support HDR mode (High Dynamic Range). You can use the [is_EnableHdr\(\)](#) function to enable / disable it. Using [is_GetHdrKneepoints\(\)](#), you can query the currently set knee points.

Input Parameters

hCam	Camera handle
KneepointArray	Pointer to a KNEEPOINTARRAY See also is_SetHdrKneepoints() .
KneepointArraySize	Size of the array



Return Values

IS_SUCCESS	Function executed successfully (supported sensor type)
IS_NO_SUCCESS	General error message (supported sensor type)
IS_NOT_SUPPORTED	Unsupported sensor type

Related Functions

- [is_GetHdrMode\(\)](#)
- [is_GetHdrKneepointInfo\(\)](#)
- [is_SetHdrKneepoints\(\)](#)
- [is_EnableHdr\(\)](#)

5.3.42 `is_GetHdrMode`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_GetHdrMode (HIDS hCam, INT* Mode)
```

Description

Some sensors support HDR (High Dynamic Range) mode. You can use the `is_EnableHdr()` function to enable / disable it. Using `is_GetHdrMode()`, you can query the HDR mode supported by the sensor.

For further information on HDR mode, please refer to the [HDR properties](#) section of the *uEye Demo* chapter.

Input Parameters

hCam	Camera handle
Mode	Possible return values
IS_HDR_KNEEPOINTS	HDR is supported.
IS_HDR_NOT_SUPPORTED	HDR is not supported.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [is_GetHdrKneepoints\(\)](#)
- [is_GetHdrKneepointInfo\(\)](#)
- [is_SetHdrKneepoints\(\)](#)
- [is_EnableHdr\(\)](#)

5.3.43 is_GetImageHistogram

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_GetImageHistogram (HIDS hCam,
                          int nID, INT ColorMode, DWORD* pHistoMem)
```

Description

`is_GetImageHistogram()` computes the histogram of the submitted image. The histogram always contains 256 values per channel. For colour modes with a bit depth of more than 8 bits, the system evaluates the 8 most significant bits (MSBs).

Input Parameters

hCam	Camera handle
nID	Memory ID
ColorMode	Colour mode of the image with the <code>nID</code> memory ID For a list of all available colour formats and the associated input parameters, see the Appendix: Colour and Memory Formats section.
pHistoMem	Pointer to a <code>DWORD</code> array The array must be allocated in such a way that it can accommodate 3×256 values for colour formats and in raw Bayer mode. In monochrome mode, the array must be able to accommodate 1×256 values.

Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
IS_NULL_POINTER	Invalid Array
IS_INVALID_COLOR_FORMAT	Unsupported colour format
IS_INVALID_PARAMETER	Unknown <code>ColorMode</code> parameter

Code Sample

```
char * pcSource;
INT nIDSource;
is_AllocImageMem (hCam, 256, 256, 24, &pcSource, &nIDSource);

int nX, nY, nBits, nPitch;
is_InquireImageMem (hCam, pcSource, nIDSource, &nX, &nY, &nBits, &nPitch);



//Create RGB test image
for (int j = 0; j < nY; j++)
{
    for (int i = 0; i < nX*3; i += 3)
    {
        pcSource[i + j*nPitch] = 0; // Blue pixels
        pcSource[i + j*nPitch + 1] = i/3; // Green pixels
        pcSource[i + j*nPitch + 2] = 255; // Red pixels
    }
}

// Create memory for RGB histogram
DWORD bgrBuffer [256*3];

//Create pointer for each histogram colour
DWORD * pBlueHisto = bgrBuffer;
DWORD *pGreenHisto = bgrBuffer + 256;
DWORD * pRedHisto = bgrBuffer + 512;

//Retrieve histogram and release memory
is_GetImageHistogram (hCam, nIDSource, IS_SET_CM_RGB24, bgrBuffer);
is_FreeImageMem (hCam, pcSource, nIDSource);
```

5.3.44 is_GetImageInfo

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_GetImageInfo (HIDS hCam,
                    INT nImageBufferID,
                    UEYEIMAGEINFO* pImageInfo,
                    INT nImageInfoSize)
```



Overview

- [Description](#)
- [Input Parameters](#)
- [Contents of the UEYEIMAGEINFO Structure](#)
- [Status Flags in UEYEIMAGEINFO::dwIoStatus](#)
- [Contents of the UEYETIME Structure](#)
- [Return Values](#)
- [Related Functions](#)
- [Code Sample](#)

Description

`is_GetImageInfo()` provides additional information on the images you take. The function returns a timestamp indicating the time of image capture, and the states of the camera I/Os at that point in time. To get information on the last image that was taken, call `is_GetImageInfo` directly after receiving the `IS_FRAME` event.

Using the function with *GigE uEye* cameras

The `UEYEIMAGEINFO` structure returns the camera timestamp `u64TimestampDevice`, which indicates the time of image capture with an accuracy of 100 ns. The time of image capture is defined as:

- The time when a (hardware or software) trigger event is received by the camera in trigger mode. The delay between the receipt of the trigger signal and the start of exposure depends on the sensor. For the delays of the individual sensors, please see the [Specifications: Sensors](#) chapter.
- The time when the sensor starts to output image data in freerun mode (see also [How To Proceed: Image Capture](#)). A rolling shutter sensors starts to output image data after exposure of the first row. With a global shutter sensor, image data is output after exposure of all rows.

The `UEYETIME` structure returns a timestamp with a resolution of 1 ms. The timestamp is synchronized with the PC's system time, and resynchronized every 60 seconds. This may cause minor time shifts in the value passed in `UEYETIME`.

To determine the exact interval between two image captures, it is therefore recommended to read out the camera timestamp `u64TimestampDevice`.

Using the function with *USB uEye* cameras

The `u64TimestampDevice` timestamp returns the time when image data transfer to the PC was completed.

The `UEYETIME` structure returns the timestamp (with a resolution of 1 ms) synchronized with the

PC system time.



Image buffers that are part of a sequence need to be locked using `is_LockSeqBuf()`. This is important to ensure correct assignment between image data and image information. Otherwise, it may happen that an image buffer is filled with new image data. In this case, the image information will not match the image data any more.

Input Parameters

<code>hCam</code>	Camera handle
<code>nImageBufferID</code>	ID of the image buffer for which information is requested
<code>pImageInfo</code>	Pointer to a <code>UEYEIMAGEINFO</code> type structure to which the information will be written
<code>nImageInfoSize</code>	Size of the structure

Contents of the `UEYEIMAGEINFO` Structure

DWORD	<code>dwFlags</code>	Internal status flags (currently not used)
unsigned long long	<code>u64TimestampDevice</code>	Internal timestamp of image capture
UEYETIME	<code>TimestampSystem</code>	Structure with timestamp information in PC system time format, see UEYETIME below
DWORD	<code>dwIoStatus</code>	With <i>GigE uEye HE</i> cameras: Returns the states of the GPIOs (programmable I/Os) at the time of image capture: <ul style="list-style-type: none"> GPIO as input: Pending signal GPIO as output: Set level With all other cameras, <code>dwIoStatus</code> is empty. See dwIoStatus below.
unsigned long long	<code>u64FrameNumber</code>	Internal image number
DWORD	<code>dwImageBuffers</code>	Number of image buffers existing in the camera
DWORD	<code>dwImageBuffersInUse</code>	Number of image buffers in use in the camera

Status Flags in `UEYEIMAGEINFO::dwIoStatus`

0x00 (00)	Both GPIOs return 0
0x01 (01)	First GPIO returns 1, second GPIO returns 0
0x02 (10)	First GPIO returns 0, second GPIO returns 1
0x03 (11)	Both GPIOs return 1

Contents of the UEYEIMAGEINFO::UEYETIME Structure

WORD	wYear	Timestamp year
WORD	wMonth	Timestamp month
WORD	wDay	Timestamp day
WORD	wHour	Timestamp hour
WORD	wMinute	Timestamp minute
WORD	wSecond	Timestamp second
WORD	wMilliseconds	Timestamp millisecond
WORD	wReserved[2]	Reserved

Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
IS_INVALID_PARAMETER	<p>One of the parameters passed is invalid. This may happen when:</p> <ul style="list-style-type: none"> • more memory is allocated than the UEYEIMAGEINFO structure needs • nImageBufferID <= 0 • pImageInfo == NULL • nImageInfoSize <= 0

Related Functions

- [is_GetCaptureErrorInfo\(\)](#)
- [is_LockSeqBuf\(\)](#)
- [is_UnlockSeqBuf\(\)](#)
- [is_SetImageMem\(\)](#)
- [is_SetIO\(\)](#)
- [is_SetIOMask\(\)](#)

Code Sample

```

UEYEIMAGEINFO ImageInfo;
// Read out camera timestamp
INT nRet = is_GetImageInfo( m_hCam,
                           m_lMemoryId, &ImageInfo, sizeof
                           (ImageInfo));

if (nRet == IS_SUCCESS)
{
    unsigned long long u64TimestampDevice;
    u64TimestampDevice = ImageInfo.u64TimestampDevice;

    CString Str; // Read out timestamp in system time
    Str.Format("%02d.%02d.%04d, %02d:%02d:%02d:%03d",
              ImageInfo.TimestampSystem.wDay,
              ImageInfo.TimestampSystem.wMonth,
              ImageInfo.TimestampSystem.wYear,
              ImageInfo.TimestampSystem.wHour,
              ImageInfo.TimestampSystem.wMinute,



```

```
        ImageInfo.TimestampSystem.wSecond,  
        ImageInfo.TimestampSystem.wMilliseconds);  
  
    DWORD dwTotalBuffers = ImageInfo.dwImageBuffers;  
    DWORD dwUsedBuffers = ImageInfo.dwImageBuffersInUse;  
}
```

Sample Program

- uEyeTimestamp (C++)

5.3.45 is_GetImageMem

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_GetImageMem (HIDS hCam, VOID** pMem)
```

Description

`is_GetImageMem()` returns the pointer to the starting address of the active image memory. If you use ring buffering, `is_GetImageMem()` returns the starting address of the image memory last used for image capturing.

Input Parameters

<code>hCam</code>	Camera handle
<code>pMem</code>	Pointer to the starting address of the image memory

Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message



Related Functions

- [`is_GetImageMemPitch\(\)`](#)
- [`is_AllocImageMem\(\)`](#)
- [`is_AddToSequence\(\)`](#)
- [`is_SetImageMem\(\)`](#)
- [`is_SetAllocatedImageMem\(\)`](#)

Sample Programs

- `uEyePixelPeek (C++)`

5.3.46 `is_GetImageMemPitch`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_GetImageMemPitch (HIDS hCam, INT* pPitch)
```

Description

`is_GetImageMemPitch()` returns the line increment (in bytes). The line increment is defined as the number of bytes from the beginning of a line to the beginning of the next line. It may be greater than suggested by the parameters passed when calling `is_AllocImageMem()`. The line increment is always a number that can be divided by 4.

The line increment is calculated as:

```
line      = width * [(bitapixel + 1) / 8]
lineinc   = line + adjust.
adjust    = 0  if line can be divided by 4 without remainder
adjust    = 4 - rest(line / 4) if line cannot be divided by 4 without remainder
```

Input Parameters

<code>hCam</code>	Camera handle
<code>pPitch</code>	Pointer to the variable containing the line increment



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message

Related Functions

- `is_GetImageMem()`
- `is_AllocImageMem()`
- `is_AddToSequence()`
- `is_SetImageMem()`
- `is_SetAllocatedImageMem()`

5.3.47 is_GetNumberOfCameras

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_GetNumberOfCameras (INT* pNumCams)
```

Description

`is_GetNumberOfCameras()` returns the number of *uEye* cameras connected to the PC.

Input Parameters

pNumCams	Returns the number of connected cameras.
----------	--



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [`is_GetCameraList\(\)`](#)
- [`is_GetEthDeviceInfo\(\)`](#)

5.3.48 is_GetOsVersion

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

INT is_GetOsVersion ()

Description

is_GetOsVersion() returns the operating system type used at runtime.

Input Parameters

<none>



Return Values

IS_OS_WIN2000	Windows 2000 operating system
IS_OS_WINXP	Windows XP operating system
IS_OS_WINSERVER2003	Windows Server 2003 operating system
IS_OS_WINVISTA	Windows Vista operating system
IS_OS_WIN7	Windows 7 operating system
IS_OS_LINUX26	Linux 2.6 operating system
IS_OS_UNDETERMINED	Unknown operating system

Related Functions

- [is_GetDLLVersion\(\)](#)

5.3.49 is_GetPixelClockRange

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_GetPixelClockRange (HIDS hCam, INT* pnMin, INT* pnMax)
```

Description

`is_GetPixelClockRange()` returns the adjustable pixel clock range.
The pixel clock limit values can vary, depending on the camera model and operating mode. For detailed information on the pixel clock range of a specific camera model, please refer to the [Specifications: Sensors](#) chapter.

Input Parameters

hCam	Camera handle
pnMin	Returns the lower limit value.
pnMax	Returns the upper limit value.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [is_SetPixelClock\(\)](#)
- [is_GetFramesPerSecond\(\)](#)
- [is_GetFrameTimeRange\(\)](#)
- [is_GetExposureRange\(\)](#)
- [is_GetExposureTime\(\)](#)

5.3.50 `is_GetSensorInfo`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_GetSensorInfo (HIDS hCam, SENSORINFO* pInfo)
```

Description

Using `is_GetSensorInfo()`, you can query information about the sensor type used in the camera. The information contained in the `SENSORINFO` structure is listed in the table below.

The [*ueye.h*](#) file provides a complete up-to-date list of all supported sensor types. To quickly locate the list, search the file for the keyword "Sensor Types".

Input Parameters

<code>hCam</code>	Camera handle
<code>pInfo</code>	Pointer to the <code>SENSORINFO</code> Structure

Contents of the `SENSORINFO` Structure

WORD	<code>SensorID</code>	Returns the sensor type (e.g.: <code>IS_SENSOR_UI224X_C</code>).
Char	<code>strSensorName[32]</code>	Returns the camera model (e.g.: "UI224xLE-C").
Char	<code>nColorMode</code>	Returns the sensor colour mode. <code>IS_COLORMODE_BAYER</code> <code>IS_COLORMODE_MONOCHROME</code>
DWORD	<code>nMaxWidth</code>	Returns the maximum image width.
DWORD	<code>nMaxHeight</code>	Returns the maximum image height.
BOOL	<code>bMasterGain</code>	Indicates whether the sensor provides analogue master gain.
BOOL	<code>bRGain</code>	Indicates whether the sensor provides analogue red channel gain.
BOOL	<code>bGGain</code>	Indicates whether the sensor provides analogue green channel gain.
BOOL	<code>bBGain</code>	Indicates whether the sensor provides analogue blue channel gain.
BOOL	<code>bGlobShutter</code>	Indicates whether the sensor has a global shutter. <code>TRUE</code> = global shutter <code>FALSE</code> = rolling shutter
Char	<code>Reserved[16]</code>	Reserved



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message

Related Functions

- [is_GetCameraInfo\(\)](#)
- [is_CameraStatus\(\)](#)

5.3.51 is_GetSensorScalerInfo

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_GetSensorScalerInfo (HIDS hCam,
                           SENSORSCALERINFO* pSensorScalerInfo,
                           INT nSensorScalerInfoSize)
```

Description

Using `is_GetSensorScalerInfo()` you can query information on the image scaling which is supported by some sensors.



Internal image scaling is only supported by UI-149x/549x series sensors.

Input Parameters

hCam	Camera handle
pSensorScalerInfo	Pointer to a <code>SENSORSCALERINFO</code> type structure to which the information will be written
nSensorScalerInfoSize	Size of the structure

Contents of the `SENSORSCALERINFO` Structure

INT	nCurrMode	Returns the current mode
INT	nNumberOfSteps	Returns the number of steps for the scaling factor
double	dblFactorIncrement	Returns the increment for the scaling factor
double	dblMinFactor	Returns the minimum scaling factor
double	dblMaxFactor	Returns the maximum scaling factor
double	dblCurrFactor	Returns the current scaling factor
BYTE	bReserved[88]	Reserved



Return Values

IS_SUCCESS	Function executed successfully
IS_INVALID_PARAMETER	General error message
IS_NOT_SUPPORTED	The test image function is not supported by the camera.

Related Functions

- [is_SetSensorScaler\(\)](#)

5.3.52 is_GetSupportedTestImages

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_GetSupportedTestImages (HIDS hCam, INT* SupportedTestImages)
```

Description

`is_GetSupportedTestImages()` returns all test images supported by the camera. You can enable the sensor test image feature using `is_SetSensorTestImage()`.

Input Parameters

hCam	Camera handle
SupportedTestImages	Returns a bit mask of all test images supported by the camera.
IS_TEST_IMAGE_NONE	No test image
IS_TEST_IMAGE_WHITE	White image
IS_TEST_IMAGE_BLACK	Black image
IS_TEST_IMAGE_HORIZONTAL_GREYSCALE	Horizontal greyscale
IS_TEST_IMAGE_VERTICAL_GREYSCALE	Vertical greyscale
IS_TEST_IMAGE_DIAGONAL_GREYSCALE	Diagonal greyscale
IS_TEST_IMAGE_WEDGE_GRAY	Grey wedges
IS_TEST_IMAGE_WEDGE_COLOR	Colour wedges
IS_TEST_IMAGE_ANIMATED_WEDGE_GRAY	Grey wedges, animated
IS_TEST_IMAGE_ANIMATED_WEDGE_COLOR	Colour wedges, animated
IS_TEST_IMAGE_MONO_BARS	Monochrome bars
IS_TEST_IMAGE_COLOR_BARS1	Colour bars
IS_TEST_IMAGE_COLOR_BARS2	Colour bars
IS_TEST_IMAGE_GREY_AND_COLOR_BARS	Grey and colour bars
IS_TEST_IMAGE_MOVING_GREY_AND_COLOR_BARS	Grey and colour bars, animated
IS_TEST_IMAGE_ANIMATED_LINE	Line, animated
IS_TEST_IMAGE_ALTERNATE_PATTERN	Alternating pattern (raw Bayer mode only)
IS_TEST_IMAGE_MONOCHROME_HORIZONTAL_BARS	Monochrome bars, horizontal
IS_TEST_IMAGE_MONOCHROME_VERTICAL_BARS	Monochrome bars, vertical
IS_TEST_IMAGE_COLDPIXEL_GRID	Camera image overlaid with a grid of blue dots
IS_TEST_IMAGE_HOTPIXEL_GRID	Camera image overlaid with a grid of red dots
IS_TEST_IMAGE_VARIABLE_GREY	Adjustable greyscale image

IS_TEST_IMAGE_VARIABLE_RED_PART	Image with adjustable red content
IS_TEST_IMAGE_VARIABLE_GREEN_PART	Image with adjustable green content
IS_TEST_IMAGE_VARIABLE_BLUE_PART	Image with adjustable blue content



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
IS_NOT_SUPPORTED	The test image function is not supported by the camera.

Related Functions

- [is_SetSensorTestImage\(\)](#)
- [is_GetTestImageValueRange\(\)](#)

5.3.53 is_GetTestImageValueRange

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_GetTestImageValueRange (HIDS hCam,  
                               INT TestImage,  
                               INT* TestImageValueMin,  
                               INT* TestImageValueMax)
```

Description

Using `is_GetTestImageValueRange()`, you can query the value range of the additional parameter required for some camera test images. You can enable the sensor test image feature using `is_SetSensorTestImage()`.

Input Parameters

hCam	Camera handle
TestImage	Test image for which the value range is queried
TestImageValueMin	Minimum value
TestImageValueMax	Maximum value



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
IS_NOT_SUPPORTED	The test image function is not supported by the camera. In this case, the <code>TestImageValueMin</code> and <code>TestImageValueMax</code> parameters are equal to 0.

Related Functions

- `is_GetSupportedTestImages()`
- `is_SetSensorTestImage()`

5.3.54 `is_GetTimeout`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_GetTimeout (HIDS hCam, UINT nMode, UINT* pTimeout)
```

Description

Using `is_GetTimeout()`, you can read out user-defined timeout values from the *uEye* API.

For further information, please refer to the [How To Proceed: Timeout Values for Image Capture](#) section.

Input Parameters

<code>hCam</code>	Camera handle
<code>nMode</code>	Selects the timeout value to be returned
<code>IS_TRIGGER_TIMEOUT</code>	Returns the timeout value in ms for triggered image capture
<code>pTimeout</code>	Pointer to the variable that holds the timeout value. Returns 0 if the default value of the <i>uEye</i> API is used.

Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message
<code>IS_NOT_SUPPORTED</code>	The value for <code>nMode</code> is invalid
<code>IS_INVALID_PARAMETER</code>	The <code>pTimeout</code> pointer is NULL



Related Functions

- [`is_SetTimeout\(\)`](#)
- [`is_CaptureVideo\(\)`](#)
- [`is_FreezeVideo\(\)`](#)
- [`is_SetExternalTrigger\(\)`](#)

Code Sample

```
// Return user-defined timeout
UINT nTimeout;
INT ret = is_GetTimeout(hCam, IS_TRIGGER_TIMEOUT, &nTimeout);
```

5.3.55 is_GetUsedBandwidth

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_GetUsedBandwidth (HIDS hCam)
```

Description

`is_GetUsedBandwidth()` returns the bus bandwidth (in Mbit/s) currently used by all initialised or selected cameras. This is an approximate value which is calculated based on the pixel clock that has been set and the data format (bits per pixel). The actual data load on the bus can slightly deviate from this value.

Input Parameters

hCam	Camera handle
------	---------------



Return Values

INT value	The total current bus bandwidth (in Mbit/s)
-----------	---

Related Functions

- [`is_SetPixelClock\(\)`](#)
- [`is_SetColorConverter\(\)`](#) (when a *GigE uEye HE* camera is used)

5.3.56 is_GetVsyncCount

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

INT is_GetVsyncCount (HIDS hCam, long* pIntr, long* pActIntr)

Description

is_GetVsyncCount() reads out the VSYNC counter. It will be incremented by 1 each time the sensor starts capturing an image.

Input Parameters

hCam	Camera handle
pIntr	Current VSYNC count
pActIntr	Current Frame SYNC count



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [is_GetFramesPerSecond\(\)](#)

5.3.57 is_HasVideoStarted

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_HasVideoStarted (HIDS hCam, BOOL* pbo)
```

Description

Using `is_HasVideoStarted()`, you can check whether the image digitising process has started. This function is helpful when the `is_FreezeVideo()` function was called with the `IS_DONT_WAIT` parameter.

Input Parameters

hCam	Camera handle
pbo	Returns the digitising status: 0 = Image capturing has not started yet. 1 = Image capturing has started.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- `is_FreezeVideo()`
- `is_IsVideoFinish()`

5.3.58 `is_InitCamera`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_InitCamera (HIDS* phf, HWND hWnd)
```

Description

`is_InitCamera()` starts the driver and establishes the connection to the camera. After successful initialization, this function assigns the camera handle. All subsequent functions require this handle as the first parameter.

When using Direct3D for image display, you can pass a handle to the output window (see also [How To Proceed: Image Display](#)).



When you are using a *GigE uEye SE* camera, initialization will be aborted with an error message if the camera's starter firmware is not compatible with the installed driver. To initialize the camera, you have to write the proper starter firmware to the camera first. This is done using the `is_InitCamera()` function with corresponding parameter setting (see Input Parameters table).

Updating the *GigE uEye SE* firmware can take up to 20 seconds. It is important not to disconnect the camera from the PC or power supply during this time. Otherwise, malfunctions could occur in the camera.

If you have initialized the camera with a driver version earlier than 3.40, you need to update the camera firmware manually using the uEye Camera Manager. Driver versions 3.40 and higher support automatic firmware updates.



Note on multi-camera environments:

When using multiple cameras in parallel operation on a single system, you should assign a unique camera ID to each camera. To initialize or select a camera with `is_InitCamera()`, the `phf` handle must previously have been set to the desired camera ID.

To initialize or select the next available camera without specifying a camera ID, `phf` has to be preset with 0.



We recommend that you call the following functions exclusively from a single thread in order to avoid unpredictable behavior of the application.

- `is_InitCamera()`
- `is_SetDisplayMode()`
- `is_ExitCamera()`

See also [General: Thread Programming](#).

Input Parameters

phf	<p>Pointer to the camera handle</p> <p>When you call this function, the pointer value has the following meaning:</p> <p>0: The first available camera will be initialized or selected.</p> <p>1-254: The camera with the specified camera ID will be initialized or selected.</p>
phCam IS_USE_DEVICE_ID	<p>The camera is opened using the device ID instead of the camera ID. For details on device ID please refer to the is_GetCameraList() chapter.</p>
phf IS_ALLOW_STARTER_FW_UPLOAD	<p>During initialization of the camera, this parameter checks whether a new version of the starter firmware is required. If it is, the new starter firmware is updated automatically (only <i>GigE uEye SE</i> cameras).</p> <p>To ensure backward compatibility of applications, always call <code>is_InitCamera()</code> without the <code>IS_ALLOW_STARTER_FW_UPLOAD</code> parameter first. Only if an error occurs, call the function with this parameter set (see Code Sample below).</p>
hWnd	<p>Pointer to the window where the Direct3D image will be displayed</p> <p>If <code>hWnd = NULL</code>, DIB mode will be used for image display.</p>

Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
IS_STARTER_FW_UPLOAD_NEEDED	The camera's starter firmware is not compatible with the driver and needs to be updated.

Related Functions

- [is_ExitCamera\(\)](#)
- [is_EnableAutoExit\(\)](#)
- [is_GetCameraList\(\)](#)
- [is_SetCameraID\(\)](#)
- [is_GetCameraInfo\(\)](#)
- [is_GetDuration\(\)](#) (when a *GigE uEye SE* camera is used)
- [is_SetStarterFirmware\(\)](#) (when a *GigE uEye SE* camera is used)

Code Sample

```
//Open camera with ID 1
HIDS hCam = 1;
INT nRet = is_InitCamera (&hCam, NULL);

if (nRet != IS_SUCCESS)
{
    //Check if new starter firmware is needed
    if (nRet == IS_STARTER_FW_UPLOAD_NEEDED)
```



```
{
    //Calculate time needed for updating the starter firmware
    INT nTime;
    is_GetDuration (hCam, IS_SE_STARTER_FW_UPLOAD, &nTime);
    /*
    ... e.g. have progress bar displayed in separate thread
    */



    //Upload new starter firmware during initialization
    nRet = is_InitCamera (&hCam | IS_ALLOW_STARTER_FW_UPLOAD, NULL);

    /*
    ... end progress bar
    */
}
```

Sample Programs

- uEyeMultipleCameraScan (C++)
- uEyeConsole (C++)
- uEyeC# Demo (C#)
- uEye VB Simple Demo (VB6)
- uEye VB Dual Camera Demo (VB6)

5.3.59 is_InitEvent

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_InitEvent (HIDS hCam, HANDLE hEv, INT which)
```

Description

`is_InitEvent()` initialises the event handle for the specified event object. This registers the event object in the *uEye* kernel driver.

Input Parameters

hCam	Camera handle
hEv	Event handle created by the <code>CreateEvent()</code> Windows API function. Note: Under Linux, <code>hEv</code> has to be <code>NULL</code> .
which	ID of the event to be initialised
IS_SET_EVENT_FRAME	A new image is available.
IS_SET_EVENT_SEQ	The sequence is completed.
IS_SET_EVENT_STEAL	An image extracted from the overlay is available.
IS_SET_EVENT_TRANSFER_FAILED	During the transfer, data was lost.
IS_SET_EVENT_EXTTTRIG	An image which was captured following the arrival of a trigger has been transferred completely. This is the earliest possible moment for a new capturing process. The image must then be post-processed by the driver and will be available after the <code>IS_FRAME</code> processing event.
IS_SET_EVENT_REMOVE	A camera initialised with <code>is_InitCamera()</code> was disconnected. *)
IS_SET_EVENT_DEVICE_RECONNECTED	A camera initialised with <code>is_InitCamera()</code> and disconnected afterwards was reconnected. *)
IS_SET_EVENT_NEW_DEVICE	A new camera was connected. This is independent of the device handle (<code>hCam</code> is ignored).
IS_SET_EVENT_REMOVAL	A camera was removed. This is independent of the device handle (<code>hCam</code> is ignored).
IS_SET_EVENT_WB_FINISHED	The automatic white balance control is completed.

*) Not available under Linux.

Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [is_EnableEvent\(\)](#)
- [is_DisableEvent\(\)](#)
- [is_ExitEvent\(\)](#)
- [is_WaitEvent\(\)](#)

Code Sample

```
HANDLE hEvent = CreateEvent(NULL, TRUE, FALSE, "");



if (hEvent != NULL)
{
    //Enable frame event, start image capture and wait for event:
    is_InitEvent(hCam, hEvent, IS_SET_EVENT_FRAME);
    is_EnableEvent(hCam, IS_SET_EVENT_FRAME);
    is_FreezeVideo(hCam, IS_DONT_WAIT);

    if (WaitForSingleObject(hEvent, 1000) == WAIT_OBJECT_0)
    {
        // Image was captured successfully
        is_DisableEvent(hCam, IS_SET_EVENT_FRAME);
        is_ExitEvent(hCam, IS_SET_EVENT_FRAME);
    }
}
```

Sample Programs

- SimpleLive (C++)
- uEyeEvent (C++)

5.3.60 is_InquireImageMem

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_InquireImageMem (HIDS hCam,
                        char* pcMem, int nID,
                        int* pnX, int* pnY,
                        int* pnBits, int* pnPitch);
```

Description

`is_InquireImageMem()` reads out the properties of an allocated image memory.

Input Parameters

hCam	Camera handle
pMem	Pointer to the starting address of the image memory as allocated by is_AllocImageMem()
NID	ID of the image memory as allocated by is_AllocImageMem()
pnX	Returns the width used to define the image memory (can be 0).
pnY	Returns the height used to define the image memory (can be 0).
pnBits	Returns the bit width used to define the image memory (can be 0).
pnPitch	Returns the line increment of the image memory (can be 0).



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [is_AllocImageMem\(\)](#)
- [is_SetImageMem\(\)](#)
- [is_SetAllocatedImageMem\(\)](#)
- [is_GetColorDepth\(\)](#)

5.3.61 `is_IsVideoFinish`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_IsVideoFinish (HIDS hCam, BOOL* pbo)
```

Description

Using `is_IsVideoFinish()`, you can check whether an image has been captured and stored completely in the image memory. This function is helpful if the `is_FreezeVideo()` function was called with the `IS_DONT_WAIT` parameter.

By setting the `*pbo==IS_TRANSFER_FAILED` parameter before calling `is_IsVideoFinish()`, you can also check whether a transfer or post-processing error occurred.

Input Parameters

hCam	Camera handle
pbo	<p>By setting <code>*pbo != IS_TRANSFER_FAILED</code> before calling the function, <code>pbo</code> contains the following digitising status:</p> <p><code>IS_VIDEO_NOT_FINISH</code> = Digitising of the image is not completed yet.</p> <p><code>IS_VIDEO_FINISH</code> = Digitising of the image is completed.</p> <p>By setting <code>*pbo == IS_TRANSFER_FAILED</code> before calling the function, <code>pbo</code> contains the following digitising status:</p> <p><code>IS_VIDEO_NOT_FINISH</code> = Digitising of the image is not completed yet.</p> <p><code>IS_VIDEO_FINISH</code> = Digitising of the image is completed.</p> <p><code>IS_TRANSFER_FAILED</code> = Transfer error or conversion problem (e.g. destination memory is invalid)</p>



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message

Related Functions

- `is_FreezeVideo()`
- `is_HasVideoStarted()`

5.3.62 is_LoadBadPixelCorrectionTable

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_LoadBadPixelCorrectionTable (HIDS hCam, const IS_CHAR* File)
```

Description

`is_LoadBadPixelCorrectionTable()` loads a list of sensor hot pixel coordinates that was previously saved using the [is_SaveBadPixelCorrectionTable\(\)](#) function.

Input Parameters

hCam	Camera handle
File	Pointer to a string which contains the name of the file where the coordinates are stored. You can either pass an absolute or a relative path. If <code>NULL</code> is passed, the "Open File" dialogue opens.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [is_SaveBadPixelCorrectionTable\(\)](#)
- [is_SetBadPixelCorrection\(\)](#)
- [is_SetBadPixelCorrectionTable\(\)](#)

5.3.63 is_LoadImage

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_LoadImage (HIDS hCam, char* File)
```

Description

is_LoadImage() loads an image from a file. The image must have been saved in BMP format. It is loaded into the active image memory.

Input Parameters

hCam	Camera handle
File	Pointer to a filename You can either pass an absolute or a relative path. If NULL is passed, the "Open File" dialogue opens.



Return Values

IS_SUCCESS	The image was loaded without errors.
IS_FILE_READ_INVALID_BMP_SIZE	The size of the image to be loaded exceeds the active image memory size.
IS_FILE_READ_INVALID_BMP_ID	The file to be loaded is not a valid bitmap file.
IS_FILE_READ_OPEN_ERROR	The file could not be opened.

Related Functions

- [Is_LoadImageMem\(\)](#)
- [is_GetImageMem\(\)](#)
- [is_SetImageMem\(\)](#)
- [is_SaveImage\(\)](#)
- [is_SaveImageEx\(\)](#)
- [is_SaveImageMem\(\)](#)
- [is_SaveImageMemEx\(\)](#)

5.3.64 is_LoadImageMem

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_LoadImageMem (HIDS hCam, char* File, char** ppcImgMem, int* pid)
```

Description

`is_LoadImageMem()` loads an image from a file. The image must have been saved in BMP format. The image, together with its colour format and colour depth properties, is loaded into a newly allocated image memory.

Using the `is_FreeImageMem()` function, you can release the image memory again.

Input Parameters

hCam	Camera handle
File	Filename You can either pass an absolute or a relative path. If NULL is passed, the "Open File" dialogue opens.
ppcImgMem	Pointer to a variable containing the starting address
pid	Pointer to a variable containing the memory ID



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
IS_FILE_READ_INVALID_BMP_ID	Bitmap format of the image to be loaded is invalid.
IS_FILE_READ_OPEN_ERROR	File cannot be opened.

Related Functions

- `Is_LoadImage()`
- `is_GetImageMem()`
- `is_SetImageMem()`
- `is_SaveImage()`
- `is_SaveImageEx()`
- `is_SaveImageMem()`
- `is_SaveImageMemEx()`

5.3.65 is_LoadParameters

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_LoadParameters (HIDS hCam, char* pFilename)
```

Description

is_LoadParameters() loads the parameters for a camera from a *uEye* ini file or from the camera EEPROM. Using the is_SaveParameters() function, you can save camera parameters in an ini file or in the camera.



Only camera-specific ini files can be loaded. The uEye Parameter File section in the Appendix describes the structure of a *uEye* ini file.

When loading an ini file, make sure that the image size (AOI) and colour depth parameters in the ini file match those in the allocated memory. Otherwise, display errors may occur.

Input Parameters

hCam	Camera handle
pFilename	Pointer to a filename. You can either pass an absolute or a relative path. For the internal camera parameter sets, these would be "\\cam\\set1" or "/cam/set1", or "\\cam\\set2" or "/cam/set2", respectively. If NULL is passed, the "Open File" dialogue is displayed.

You can load the parameter sets stored in the camera EEPROM using specific filenames:

pFilename	
"\\cam\\set1" or "/cam/set1"	Parameter set 1
"\\cam\\set2" or "/cam/set2"	Parameter set 2



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
IS_INVALID_CAMERA_TYPE	This is the ini file of a different camera type

Related Functions

- is_SaveParameters()

5.3.66 is_LockSeqBuf

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_LockSeqBuf (HIDS hCam, INT nNum, char* pcMem)
```

Description

`is_LockSeqBuf()` locks write access to an image memory within a sequence. In the capturing process, locked image memories will be skipped in the sequence list of image memories to be used. This way, you can avoid that image data which are required for further processing will be overwritten by newly captured data. Full access to the image memory is still guaranteed. You can lock any number of image memories at the same time.

Using the `is_UnlockSeqBuf()` function, you can re-enable write access to the image memory.

Input Parameters

hCam	Camera handle
nNum	Number of the image memory to be locked (1 . . max) or <code>IS_IGNORE_PARAMETER</code> : The image memory will be identified by its starting address only.
pcMem	Starting address of the image memory to be locked



nNum indicates the location in the sequence list, not the memory ID assigned using `is_AllocImageMem()`.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- `is_UnlockSeqBuf()`
- `is_AddToSequence()`
- `is_SetImageMem()`
- `is_SetAllocatedImageMem()`

5.3.67 is_ReadEEPROM

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_ReadEEPROM (HIDS hCam, INT Adr, char* pcString, INT Count)
```

Description

Using `is_ReadEEPROM()`, you can read the contents of the camera EEPROM. Besides the hard-coded factory information, the EEPROM of the *uEye* can hold 64 bytes of user data.

Input Parameters

hCam	Camera handle
Adr	Starting address for data reads Value range: 0...63
pcString	Pointer to the buffer for the data to read (min. size = Count)
Count	Number of characters to read

Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message



Related Functions

- [is_WriteEEPROM\(\)](#)

Code Sample

```
char buffer[64];
is_ReadEEPROM( hCam, 0x00, buffer, 64 );
```

5.3.68 is_ReadI2C

	
USB 2.0	USB 2.0

Syntax

```
INT is_ReadI2C (HIDS hCam,
               INT nDeviceAddr, INT nRegisterAddr,
               BYTE* pbData, INT nLen)
```

Description

Using `is_ReadI2C()`, you can read data over the I²C bus of a board level camera.

For information on the signals applied to the I²C bus, refer to the chapters with electrical specifications for the *USB uEye LE* and the *USB uEye ME*.



The `is_ReadI2C()` function is only supported by PCB versions of the *USB uEye ME/LE* camera series.



The *uEye* processes I²C addresses in a 7-bit format that is created from the 8-bit format by a bit shift to the right. The eighth bit indicates whether an address is a read (1) or write (0) address. For example, the 7-bit address 0x48 is the write address 0x90 and the read address 0x91 in 8-bit format. The following addresses for `nRegisterAddr` are assigned to the *uEye* and must not be used:

7-bit format: 0x10, 0x48, 0x4C, 0x50, 0x51, 0x52, 0x55, 0x5C, 0x5D, 0x69

8-bit format: 0x20, 0x90, 0x98, 0xA0, 0xA2, 0xA4, 0xAA, 0xB8, 0xBA, 0xD2

Input Parameters

<code>hCam</code>	Camera handle
<code>nDeviceAddr</code>	Slave device address in 7-bit format
<code>nRegisterAddr</code>	Address of a 8 bit register (only 8-bit addresses are valid)
<code>nRegisterAddr IS_I2C_16_BIT_REGISTER</code>	Address of a 16 bit register
<code>pbData</code>	Pointer to the data to be read
<code>nLen</code>	Data length <code>nLen = 1</code> : 8 bits data <code>nLen = 2</code> : 16 bits data



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message
<code>IS_INVALID_I2C_DEVICE_ADDRESS</code>	Invalid I ² C device address

Related Functions

- [is_WriteI2C\(\)](#)

5.3.69 `is_RenderBitmap`

	
USB 2.0 GigE	-

Syntax

```
INT is_RenderBitmap (HIDS hCam, INT nMemID, HWND hwnd, INT nMode)
```

Description

Using `is_RenderBitmap()`, you can output an image from an image memory in the specified window. For the display, Windows bitmap functionality is used. The image is displayed in the format you specified when allocating the image memory.

The `bitapixel` parameter of the `is_AllocImageMem()` function defines the colour depth and display type. RGB16 and RGB15 require the same amount of memory but can be distinguished by the `bitapixel` parameter.



In bitmap (DIB) mode, the color format UYVY is not supported.

Input Parameters

<code>hCam</code>	Camera handle
<code>nMemID</code>	ID of the image memory whose contents is to be displayed
<code>hwnd</code>	Output window handle
<code>nMode</code>	
<code>IS_RENDER_NORMAL</code>	The image is rendered normally. It will be displayed in 1:1 scale as stored in the image memory.
<code>IS_RENDER_FIT_TO_WINDOW</code>	The image size is adjusted to fit the output window.
<code>IS_RENDER_DOWNSCALE_1_2</code>	Displays the image at 50% of its original size.
The following options can be linked by a logical OR using the <code>nMode</code> parameter:	
<code>IS_RENDER_MIRROR_UPDOWN</code>	Mirrors the displayed image along the horizontal axis.

Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message

Related Functions

- `is_AllocImageMem()`
- `is_SetColorMode()`
- `is_SetDisplayMode()`
- `is_DirectRenderer()`

Code Sample



Fit image to window and display it upside down:

```
is_RenderBitmap (hCam, nMemID, hwnd,  
                IS_RENDER_FIT_TO_WINDOW | IS_RENDER_MIRROR_UPDOWN);
```

Sample Programs

- SimpleAcquire (C++)
- SimpleLive (C++)

5.3.70 `is_ResetCaptureErrorInfo`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_ResetCaptureErrorInfo (HIDS hCam)
```

Description

`is_ResetCaptureErrorInfo()` deletes the list of errors that occurred while images were being captured. You can retrieve this list using the [is_GetCaptureErrorInfo\(\)](#) function.

Input Parameters

<code>hCam</code>	Camera handle
-------------------	---------------



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message

Related Functions

- [is_GetCaptureErrorInfo\(\)](#)
- [is_GetError\(\)](#)
- [is_CameraStatus\(\)](#)

5.3.71 is_ResetToDefault

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_ResetToDefault (HIDS hCam)
```

Description

`is_ResetToDefault()` resets all parameters to the camera-specific defaults as specified by the driver. By default, the camera uses full resolution, a medium speed and colour level gain values adapted to daylight exposure. All optional features are disabled.

Input Parameters

hCam	Camera handle
------	---------------



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [`is_LoadParameters\(\)`](#)
- [`is_SaveParameters\(\)`](#)

5.3.72 `is_SaveBadPixelCorrectionTable`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SaveBadPixelCorrectionTable (HIDS hCam, const IS_CHAR* File)
```

Description

`is_SaveBadPixelCorrectionTable()` saves the user-defined hot pixel list to the specified file.

Input Parameters

<code>hCam</code>	Camera handle
<code>File</code>	Pointer to a string which contains the name of the file where the coordinates are stored. You can either pass an absolute or a relative path. If <code>NULL</code> is passed, the "Save as" dialogue will be displayed.



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message

Related Functions

- [`is_LoadBadPixelCorrectionTable\(\)`](#)
- [`is_SetBadPixelCorrection\(\)`](#)
- [`is_SetBadPixelCorrectionTable\(\)`](#)

5.3.73 is_SaveImage

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SaveImage (HIDS hCam, const IS_CHAR* File)
```

Description

`is_SaveImage()` saves an image in bitmap (*.BMP) format to a file. The images are read out from the current image memory. The bitmap is stored with the colour depth that was used when allocating the image memory (in DIB mode) or that was set for the current colour mode (in Direct3D mode).



In Direct3D mode, overlay data is not saved.

Input Parameters

hCam	Camera handle
File	Pointer to a string containing the BMP filename. You can either pass an absolute or a relative path. If <code>NULL</code> is passed, the "Save as" dialogue will be displayed.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [`is_SaveImageEx\(\)`](#)
- [`is_SaveImageMem\(\)`](#)
- [`is_SaveImageMemEx\(\)`](#)
- [`is_LoadImage\(\)`](#)
- [`Is_LoadImageMem\(\)`](#)
- [`is_GetImageMem\(\)`](#)
- [`is_SetImageMem\(\)`](#)

5.3.74 is_SaveImageEx

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SaveImageEx (HIDS hCam, const IS_CHAR* File,
                   INT fileFormat, INT Param)
```

Description

is_SaveImageEx() saves an image in (*.BMP) or Jpeg (*.JPG) format to a file. The images are read out from the current image memory. The bitmap is stored with the colour depth that was used when allocating the image memory (in DIB mode) or that was set for the current colour mode (in Direct3D mode).



In Direct3D mode, overlay data is not saved.

Input Parameters

hCam	Camera handle
File	Pointer to a string containing the BMP filename You can either pass an absolute or a relative path. If NULL is passed, the "Save as" dialogue will be displayed.
fileFormat	Specifies the output format of the file.
IS_IMG_BMP	Bitmap format
IS_IMG_JPG	JPEG format
Param	When you use IS_IMG_JPG to specify the file format, you can set the quality by specifying a value between 1 and 100 for Param. If Param=0, the system uses the default quality (75). If you use IS_IMG_BMP, Param does not take effect.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
IS_INVALID_PARAMETER	Invalid file format or invalid JPEG quality value

Related Functions

- [is_SaveImage\(\)](#)
- [is_SaveImageMem\(\)](#)
- [is_SaveImageMemEx\(\)](#)
- [is_LoadImage\(\)](#)
- [Is_LoadImageMem\(\)](#)
- [is_GetImageMem\(\)](#)
- [is_SetImageMem\(\)](#)

5.3.75 is_SaveImageMem

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SaveImageMem (HIDS hCam, const IS_CHAR* File,
                    char* pcMem, int nID)
```

Description

is_SaveImageMem() saves an image in bitmap (*.BMP) format to a file. The images are read out from the specified image memory. The bitmap is stored with the colour depth that was used when allocating the image memory (in DIB mode) or that was set for the current colour mode (in Direct3D mode). JPEG files are always saved with a colour depth of 8 or 24 bits.



In Direct3D mode, overlay data is not saved.

Input Parameters

hCam	Camera handle
File	Pointer to a string containing the BMP filename You can either pass an absolute or a relative path. If NULL is passed, the "Save as" dialogue will be displayed.
pcMem	Pointer to the image memory
nID	Image memory ID You can link USE_ACTUAL_IMAGE_SIZE and nID by a logical OR to save the image with the currently set image size.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [is_SaveImage\(\)](#)
- [is_SaveImageEx\(\)](#)
- [is_SaveImageMemEx\(\)](#)
- [is_LoadImage\(\)](#)
- [Is_LoadImageMem\(\)](#)
- [is_GetImageMem\(\)](#)
- [is_SetImageMem\(\)](#)

5.3.76 is_SaveImageMemEx

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SaveImageMemEx (HIDS hCam, const IS_CHAR* File,
                      char* pcMem, int nID,
                      INT fileFormat, INT Param)
```

Description

`is_SaveImageMemEx()` saves an image in bitmap (*.BMP) or Jpeg (*.JPG) format to a file. The images are read out from the specified image memory. The bitmap is stored with the colour depth that was used when allocating the image memory (in DIB mode) or that was set for the current colour mode (in Direct3D mode). JPEG files are always saved with a colour depth of 8 or 24 bits.



In Direct3D mode, overlay data is not saved.

Input Parameters

hCam	Camera handle
File	Pointer to a string containing the BMP filename. You can either pass an absolute or a relative path. If <code>NULL</code> is passed, the "Save as" dialogue will be displayed..
pcMem	Pointer to the image memory
nID	Image memory ID
fileFormat	Specifies the output format of the file.
IS_IMG_BMP	Bitmap format
IS_IMG_JPG	JPEG format
Param	When you use <code>IS_IMG_JPG</code> to specify the file format, you can set the quality by specifying a value between 1 and 100 for <code>Param</code> . If <code>Param=0</code> , the system uses the default quality (75). If you use <code>IS_IMG_BMP</code> , <code>Param</code> does not take effect.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
IS_INVALID_PARAMETER	Invalid file format or invalid JPEG quality value

Related Functions

- is_SaveImage()
- is_SaveImageEx()
- is_SaveImageMem()
- is_LoadImage()
- Is_LoadImageMem()
- is_GetImageMem()
- is_SetImageMem()

5.3.77 is_SaveParameters

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SaveParameters (HIDS hCam, const IS_CHAR* pFilename)
```

Description

`is_SaveParameters()` saves the current camera parameters to an ini file or to the EEPROM of the camera. You can load saved parameters using the [is_LoadParameters\(\)](#) function. The [uEye Parameter File](#) section in the Appendix describes the structure of a *uEye* ini file.

Input Parameters

hCam	Camera handle
pFileName	Pointer to a filename You can either pass an absolute or a relative path. For internal parameter sets, these are "\\cam\\set1" or "/cam/set1", or "\\cam\\set2" or "/cam/set2", respectively. If NULL is passed, the "Save as" dialogue will be displayed..

You can save two parameter sets in the non-volatile EEPROM of the camera using specific filenames:

pFileName	
"\\cam\\set1" or "/cam/set1"	Parameter set 1
"\\cam\\set2" or "/cam/set2"	Parameter set 2



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [is_LoadParameters\(\)](#)
- [is_CameraStatus\(\)](#)

5.3.78 is_SetAllocatedImageMem

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetAllocatedImageMem (HIDS hCam,
                             INT width, INT height, INT bitspixel,
                             char* pcImgMem, int* pid)
```

Description

Using is_SetAllocatedImageMem(), you can make a memory allocated by a user the active memory for storing digitised images in it. The allocated memory must be large enough ($size \geq (width * height * bitspixel / 8)$) and must always be locked globally (see below). You can call the is_AddToSequence() function to add a memory which was set using is_SetAllocatedImageMem() to a sequence.

Please make sure to proceed in the following order:

- Allocate Memory: `HANDLE hgMem = GlobalAlloc (size);`
- Lock memory: `char* pcMem = (char*) GlobalLock (hgMem);`

The address of this memory will be passed to the *uEye* driver. For this, you can use the is_SetAllocatedImageMem() function. In addition, you need to specify the image size, just as you do when calling is_AllocImageMem(). The returned memory ID is required by other functions for memory access.

The memory area must be removed from the driver management again using the is_FreeImageMem() function. Please note that this does not release the memory. You then need to make sure that the memory will be released again:

- Unlock memory: `GlobalUnlock (hgMem);`
- Release Memory: `is_FreeImageMem (hCam, pcMem, ID);`
`GlobalFree (hgMem);`

Input Parameters

hCam	Camera handle
width	Image width
height	Image height
bitspixel	Image colour depth (bits per pixel)
pcImgMem	Pointer to the starting address of the allocated memory
pid	Returns the ID of this memory.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [is_AllocImageMem\(\)](#)
- [is_FreeImageMem\(\)](#)
- [is_AddToSequence\(\)](#)
- [is_SetImageMem\(\)](#)
- [is_GetColorDepth\(\)](#)
- [is_GetImgMemPitch\(\)](#)

5.3.79 is_SetAOI

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetAOI (HIDS hCam,
               INT type,
               INT* pXPos, INT* pYPos,
               INT* pWidth, INT* pHeight)
```

Description

`is_SetAOI()` can be used to set the size and position of an area of interest (AOI) within an image. The following AOIs can be defined:

- Image AOI – display of an image portion
- Auto Brightness AOI – reference area of interest for automatic brightness control
- Auto Whitebalance AOI – reference area of interest of automatic white balance control



By default, the window size for auto AOIs is always maximum, i.e. it corresponds to the current image AOI.

After a change to the image geometry (by resetting an image AOI, by binning or sub-sampling), the auto AOIs will always be reset to the image AOI value (i.e. to maximum size). This means that it might be necessary to set the AOIs for the auto features again manually.



Changes to the image geometry or pixel clock affect the value ranges of the frame rate and exposure time. After executing `is_SetAOI()`, calling the following functions is recommended in order to keep the defined camera settings:

- `is_SetFrameRate()`
- `is_SetExposureTime()`
- If you are using the *uEye*'s flash function: `is_SetFlashStrobe()`

Input Parameters

The `pXPos` and `pYPos` parameters represent an offset with respect to the upper left image corner. The cut window is copied to the start position in the memory. If you want the image to be copied to the same offset within the memory, you can link the new position with a logical OR to the `IS_SET_IMAGEPOS_X_ABS` and `IS_SET_IMAGEPOS_Y_ABS` parameters.

hCam	Camera handle
type	
IS_SET_IMAGE_AOI	Sets an image AOI.
IS_GET_IMAGE_AOI	Returns the current image AOI.
IS_SET_AUTO_BRIGHT_AOI	Sets average AOI values for auto gain and auto shutter.
IS_GET_AUTO_BRIGHT_AOI	Returns the current auto brightness AOI.
IS_SET_AUTO_WB_AOI	Sets an auto white balance AOI.
IS_GET_AUTO_WB_AOI	Returns the current auto white balance AOI.
pXPos	Pointer to the horizontal position of the AOI Returns the current setting when used together with the IS_GET_... parameters.
0...XPosMax IS_SET_IMAGEPOS_X_ABS	Applies the absolute position to the memory as well.
pYPos	Pointer to the vertical position of the AOI Returns the current setting when used together with the IS_GET_... parameters.
0...YPosMax IS_SET_IMAGEPOS_Y_ABS	Applies the absolute position to the memory as well.
pWidth	Pointer to the width of the AOI Returns the current setting when used together with the IS_GET_... parameters.
pHeight	Pointer to the height of the AOI Returns the current setting when used together with the IS_GET_... parameters.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [is_SetImagePos\(\)](#)
- [is_SetBinning\(\)](#)
- [is_SetSubSampling\(\)](#)
- [is_SetAutoParameter\(\)](#)

5.3.80 `is_SetAutoCfgIpSetup`

	
GigE	GigE

Syntax

```
INT is_SetAutoCfgIpSetup (INT iAdapterID,
                        const UYEYE_ETH_AUTOCFG_IP_SETUP* pSetup,
                        UINT uStructSize)
```

Description

You can use `is_SetAutoCfgIpSetup()` to set the IP auto configuration properties of a network adapter. The IP configuration can also be changed using the *uEye Camera Manager*.



The `is_SetAutoCfgIpSetup()` function is only supported by cameras of the *GigE uEye* series.

Input Parameters

<code>iAdapterID</code>	Internal adapter ID of the network adapter. It is returned by the <code>is_GetEthDeviceInfo()</code> function in the <code>UEYE_ETH_ADAPTER_INFO</code> structure.
<code>pSetup</code>	Pointer to a <code>UEYE_ETH_AUTOCFG_IP_SETUP</code> object
<code>uStructSize</code>	Size of the <code>UEYE_ETH_AUTOCFG_IP_SETUP</code> structure in bytes

Contents of the `UEYE_ETH_AUTOCFG_IP_SETUP` Structure

<code>UEYE_ETH_ADDR_IPV4</code>	<code>ipAutoCfgIpRangeBegin</code>	First IPv4 address of the auto configuration range
<code>UEYE_ETH_ADDR_IPV4</code>	<code>ipAutoCfgIpRangeEnd</code>	Last IPv4 address of the auto configuration range
<code>BYTE</code>	<code>reserved[4]</code>	reserved

Return Values

<code>IS_SUCCESS</code>	General message indicating successful execution
<code>IS_INVALID_PARAMETER</code>	<code>pSetup</code> is invalid.
<code>IS_BAD_STRUCTURE_SIZE</code>	The structure size you specified is invalid.
<code>IS_CANT_OPEN_DEVICE</code>	Driver could not be found.
<code>IS_IO_REQUEST_FAILED</code>	Driver communication failed.

Related Functions

- `is_SetPersistentIpCfg()`
- `is_GetEthDeviceInfo()`

Code Sample



```
UEYE_ETH_AUTOCFG_IP_SETUP autocfgip;

// Specify the IP address 192.168.10.15 in hexadecimal format
autocfgip.ipAutoCfgIpRangeBegin.dwAddr= 0xC0A80A0F;

// Specify the IP address 192.168.10.35 in hexadecimal format
autocfgip.ipAutoCfgIpRangeEnd.dwAddr= 0xC0A80A23;

// Set IP address
INT nRet= is_SetAutoCfgIpSetup( iAdapterId, &autocfgip, sizeof
                                (UEYE_ETH_AUTOCFG_IP_SETUP));
```

5.3.81 is_SetAutoParameter

	
USB 2.0 GigE	USB 2.0 GigE



- ☐ At a Glance
- [Syntax](#)
- [Description](#)
- [Input Parameters](#)
- [Enable auto controls and return status information](#)
- [Adjust auto gain/auto exposure](#)
- [Adjust auto white balance](#)
- [Pre-defined values for auto gain/auto exposure](#)
- [Return Values](#)
- [Related Functions](#)
- [Code Samples](#)

Syntax

```
INT is_SetAutoParameter (HIDS hCam,
                        INT param,
                        double* pval1, double* pval2)
```

Description

Using `is_SetAutoParameter()`, you can control the automatic gain, exposure shutter, frame rate and white balance control values.

For further information on automatic control, please refer to the Functions for Automatic Image Control chapter.



- Control is only active as long as the camera is capturing images.
- A manual change of the exposure time and gain settings disables the auto functions.
- When the **auto shutter** function is enabled, you cannot modify the pixel clock frequency.
- The **auto frame rate** function is only available when the auto shutter control is on. Auto frame rate and auto gain cannot be used simultaneously.
- The **auto gain** function can only be used for cameras with master gain control set. Auto white balance is only available for cameras with hardware RGB gain control set.
- The **sensor's internal auto functions** are only supported by the sensor of the [UI-122x/522x](#) camera models. Please also read the notes on using this sensor, which are provided in the [Specifications: Sensors](#) chapter.



Automatic control by the sensor and the software is not possible simultaneously. To use the sensor's control functionality, disable software control, and vice versa.

Input Parameters

hCam	Camera handle
------	---------------

param	Configure auto control
[-] Enable auto controls and return status information	
IS_SET_ENABLE_AUTO_GAIN	Enables / disables the auto gain function. Control parameter pval1 = 1 enables, 0 disables control
IS_GET_ENABLE_AUTO_GAIN	Returns the current auto gain setting. Control parameter pval1 returns the current value
IS_SET_ENABLE_AUTO_SENSOR_GAIN	Enables / disables the sensor's internal auto gain function *). Control parameter pval1 = 1 enables, 0 disables control
IS_GET_ENABLE_AUTO_SENSOR_GAIN	Returns the current setting of the sensor's internal auto gain function *). Control parameter pval1 returns the current value
IS_SET_ENABLE_AUTO_SHUTTER	Enables / disables the auto exposure function. Control parameter pval1 = 1 enables, 0 disables control
IS_GET_ENABLE_AUTO_SHUTTER	Returns the current auto exposure setting. Control parameter pval1 returns the current value
IS_SET_ENABLE_AUTO_SENSOR_SHUTTER	Enables / disables the sensor's internal auto exposure function. *). Control parameter pval1 = 1 enables, 0 disables control
IS_GET_ENABLE_AUTO_SENSOR_SHUTTER	Returns the current setting of the sensor's internal auto exposure function *). Control parameter pval1 returns the current value
IS_SET_ENABLE_AUTO_WHITEBALANCE	Enables / disables the auto white balance function. Control parameter pval1 = 1 enables, 0 disables control
IS_GET_ENABLE_AUTO_WHITEBALANCE	Returns the current auto white balance setting. Control parameter pval1 returns the current value
IS_SET_ENABLE_AUTO_FRAMERATE	Enables / disables the auto frame rate function. Control parameter pval1 = 1 enables, 0 disables control
IS_GET_ENABLE_AUTO_FRAMERATE	Returns the current auto frame rate setting. Control parameter pval1 returns the current value
IS_SET_ENABLE_AUTO_SENSOR_FRAMERATE	Enables / disables the sensor's internal auto frame rate function. *). Control parameter

	pval1 = 1 enables, 0 disables control
IS_GET_ENABLE_AUTO_SENSOR _FRAMERATE	Returns the current setting of the sensor's internal auto frame rate function *). Control parameter pval1 returns the current value
Adjust auto gain/auto exposure	
IS_SET_AUTO_REFERENCE	Sets the setpoint value for auto gain / auto shutter. Control parameter pval1 the setpoint value (average image brightness). Independent of pixel bit depth the setpoint range is: 0 = black 128 = 50% grey (default) 255 = white
IS_GET_AUTO_REFERENCE	Returns the setpoint value for auto gain / auto shutter. Control parameter pval1 returns the current value
IS_SET_AUTO_GAIN_MAX	Sets the upper limit for auto gain. Control parameter pval1 valid value for gain (0...100)
IS_GET_AUTO_GAIN_MAX	Returns the upper limit for auto gain. Control parameter pval1 returns the current value
IS_SET_AUTO_SHUTTER_MAX	Sets the upper limit for auto exposure. Control parameter pval1 valid exposure value (0 sets the value continuously to max. exposure)
IS_GET_AUTO_SHUTTER_MAX	Returns the upper limit for auto exposure. Control parameter pval1 returns the current value
IS_SET_AUTO_SPEED	Sets the speed value for auto gain / auto exposure. Control parameter pval1 defines the control speed (0...100)
IS_GET_AUTO_SPEED	Returns the speed value for auto gain / auto exposure. Control parameter pval1 returns the current

	value
IS_SET_AUTO_HYSTERESIS	Sets the hysteresis for auto gain / auto exposure. Control parameter pval1 defines the hysteresis value (default: 2)
IS_GET_AUTO_HYSTERESIS	Returns the hysteresis for auto gain / auto exposure. Control parameter pval1 returns the current value
IS_GET_AUTO_HYSTERESIS_RANGE	Returns range for the hysteresis value. Control parameter pval1 returns the minimum value pval2 returns the maximum value
IS_SET_AUTO_SKIPFRAMES	Sets the number of frames to be skipped for auto gain / auto exposure. Control parameter pval1 defines the number of frames to be skipped (default: 4)
IS_GET_AUTO_SKIPFRAMES	Returns the number of frames to be skipped for auto gain / auto exposure. Control parameter pval1 returns the current value
IS_GET_AUTO_SKIPFRAMES_RANGE	Returns range for the number of frames to be skipped. Control parameter pval1 returns the minimum value pval2 returns the maximum value
IS_SET_AUTO_BRIGHTNESS_ONCE	Enables / disables automatic disabling of auto gain / auto exposure **). Control parameter pval1 = 1 enables, 0 disables control
IS_GET_AUTO_BRIGHTNESS_ONCE	Returns the automatic disable status of auto gain / auto exposure **). Control parameter pval1 returns the current value
▣ Adjust auto white balance	
IS_SET_AUTO_WB_OFFSET	Sets the offset value for the red and blue channels. Control parameter pval1 defines the red level offset (-50...50) pval2 defines the blue level offset (-50...50)
IS_GET_AUTO_WB_OFFSET	Returns the offset value for the red and blue

	<p>channels. Control parameter</p> <p>pval1 returns the red level offset (-50...50)</p> <p>pval2 returns the blue level offset (-50...50)</p>
IS_SET_AUTO_WB_GAIN_RANGE	<p>Sets the gain limits for the auto white balance function. Control parameter</p> <p>pval1 sets the minimum value</p> <p>pval2 sets the maximum value</p>
IS_GET_AUTO_WB_GAIN_RANGE	<p>Returns the gain limits for the auto white balance function. Control parameter</p> <p>pval1 returns the minimum value</p> <p>pval2 returns the maximum value</p>
IS_SET_AUTO_WB_SPEED	<p>Sets the speed value for the auto white balance. Control parameter</p> <p>pval1 defines the control speed (0...100)</p>
IS_GET_AUTO_WB_SPEED	<p>Returns the speed value for the auto white balance. Control parameter</p> <p>pval1 returns the current value</p>
IS_SET_AUTO_WB_HYSTERESIS	<p>Sets the hysteresis for auto white balance. Control parameter</p> <p>pval1 defines the hysteresis value (default: 2)</p>
IS_GET_AUTO_WB_HYSTERESIS	<p>Returns the hysteresis for auto white balance. Control parameter</p> <p>pval1 returns the current value</p>
IS_GET_AUTO_WB_HYSTERESIS_RANGE	<p>Returns range for the hysteresis value. Control parameter</p> <p>pval1 returns the minimum value</p> <p>pval2 returns the maximum value</p>
IS_SET_AUTO_WB_SKIPFRAMES	<p>Sets the number of frames to be skipped for auto white balance. Control parameter</p> <p>pval1 defines the number of frames to be skipped (default: 4)</p>
IS_GET_AUTO_WB_SKIPFRAMES	<p>Returns the number of frames to be skipped for auto white balance. Control parameter</p> <p>pval1 returns the current value</p>
IS_GET_AUTO_WB_SKIPFRAMES_RANGE	<p>Returns range for the number of frames to be</p>

	skipped. Control parameter pval1 returns the minimum value pval2 returns the maximum value
IS_SET_AUTO_WB_ONCE	Enables / disables automatic disabling of auto white balance **). Control parameter pval1 = 1 enables, 0 disables control
IS_GET_AUTO_WB_ONCE	Returns the automatic disable status of auto white balance **). Control parameter pval1 returns the current value
<div> <div></div> Pre-defined values for auto gain/auto exposure </div>	
For parameters pval1 and pval, NULL must be passed.	
IS_DEFAULT_AUTO_BRIGHT_REFERENCE	Default setpoint value for auto gain / exposure
IS_MIN_AUTO_BRIGHT_REFERENCE	Minimum setpoint value for auto gain / exposure
IS_MAX_AUTO_BRIGHT_REFERENCE	Maximum setpoint value for auto gain / exposure
IS_DEFAULT_AUTO_SPEED	Default value for auto speed
IS_MAX_AUTO_SPEED	Maximum value for auto speed
<div> <div></div> Pre-defined values for auto white balance </div>	
For parameters pval1 and pval, NULL must be passed.	
IS_DEFAULT_WB_OFFSET	Default value for auto white balance offset
IS_MIN_WB_OFFSET	Minimum value for auto white balance offset
IS_MAX_WB_OFFSET	Maximum value for auto white balance offset
IS_DEFAULT_AUTO_WB_SPEED	Default value for auto white balance speed
IS_MIN_AUTO_WB_SPEED	Minimum value for auto white balance speed
IS_MAX_AUTO_WB_SPEED	Maximum value for auto white balance speed
pval1	Control parameter, can have a variable value depending on the corresponding auto function (see below)
pval2	Control parameter, can have a variable value depending on the corresponding auto function (see below)

*) Not supported by all sensors (see info box above)

**) Not available when the sensor's internal controls are activated

Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- `is_GetAutoInfo()`
- `is_SetHardwareGain()`
- `is_SetHWGainFactor()`
- `is_SetExposureTime()`
- `is_SetFrameRate()`
- `is_SetAOI()`



Code Samples

```
//Enable auto gain:
double dEnable = 1;
int ret = is_SetAutoParameter (hCam ,IS_SET_ENABLE_AUTO_GAIN, &dEnable, 0);

//Set brightness setpoint value to 128:
double nominal = 128;
int ret = is_SetAutoParameter (hCam,IS_SET_AUTO_REFERENCE, &nominal, 0);

//Return shutter control limit:
double maxShutter;
int ret = is_SetAutoParameter (hCam, IS_GET_AUTO_SHUTTER_MAX, &maxShutter, 0);
```

5.3.82 is_SetBadPixelCorrection

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetBadPixelCorrection (HIDS hCam, INT nEnable, INT threshold)
```

Description

`is_SetBadPixelCorrection()` enables / disables the software correction of sensor hot pixels.



This correction will not work while sub-sampling or binning are enabled or raw Bayer mode is used.

Input Parameters

hCam	Camera handle
nEnable	
IS_BPC_DISABLE	Disables the correction function.
IS_BPC_ENABLE_SOFTWARE	Enables software correction based on the hot pixel list stored in the EEPROM.
IS_BPC_ENABLE_USER	Enables software correction based on user-defined values. First, the <code>is_SetBadPixelCorrectionTable()</code> function must be called.
IS_GET_BPC_MODE	Returns the current mode.
IS_GET_BPC_THRESHOLD	Returns the current threshold value.
threshold	Currently not used



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
Current mode when used in connection with IS_GET_BPC_MODE	
Current threshold value when used in connection with IS_GET_BPC_THRESHOLD	

Related Functions

- [`is_LoadBadPixelCorrectionTable\(\)`](#)
- [`is_SaveBadPixelCorrectionTable\(\)`](#)
- [`is_SetBadPixelCorrectionTable\(\)`](#)

5.3.83 is_SetBadPixelCorrectionTable

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetBadPixelCorrectionTable (HIDS hCam, INT nMode, WORD* pList)
```

Description

`is_SetBadPixelCorrectionTable()` can be used to set the table containing the hot pixel positions which will be used by the user-defined hot pixel correction function. You can enable hot pixel correction by calling `is_SetBadPixelCorrection()`. Each value in the table consists of a 2-byte `WORD` data type. The first value indicates the number of pixel coordinates in the table, the coordinates are listed subsequently (first X, then Y).

A table with 3 hot pixels must be structured as follows:

3	X1	Y1	X2	Y2	X3	Y3
---	----	----	----	----	----	----

Input Parameters

<code>hCam</code>	Camera handle
<code>nMode</code>	
<code>IS_SET_BADPIXEL_LIST</code>	Sets a new user-defined list. The <code>pList</code> parameter points to a list which has the format described above.
<code>IS_GET_LIST_SIZE</code>	Returns the number of pixel coordinates included in the user-defined list.
<code>IS_GET_BADPIXEL_LIST</code>	Copies the user-defined list to the <code>pList</code> parameter. Make sure to allocate the memory accordingly.
<code>pList</code>	Pointer to the starting address of the hot pixel table

Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message
Number of coordinates in the list when used together with <code>IS_GET_LIST_SIZE</code>	

Related Functions

- `is_LoadBadPixelCorrectionTable()`
- `is_SaveBadPixelCorrectionTable()`
- `is_SetBadPixelCorrection()`

Code Sample



```
WORD *pList = NULL;
// Number of coordinates in the list
DWORD nCount = is_SetBadPixelCorrectionTable (hCam, IS_GET_LIST_SIZE, NULL);

// Allocate memory for the entire list
pList = new WORD[ 1 + 2*nCount ];

// Read out list
is_SetBadPixelCorrectionTable (hCam, IS_GET_BADPIXEL_LIST, pList);

// Release the list again
delete [] pList;
```


5.3.84 is_SetBinning

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetBinning (HIDS hCam, INT mode)
```

Description

Using `is_SetBinning()`, you can enable the binning mode both in horizontal and in vertical direction. This way, the image size in the binning direction can be reduced without scaling down the area of interest. Depending on the sensor used, the sensitivity or the frame rate can be increased while binning is enabled.

To enable horizontal and vertical binning at the same time, you can link the horizontal and vertical binning parameters by a logical OR.

The adjustable binning factors of each sensor are listed in the [Specifications: Sensors](#) chapter.



Some sensors allow a higher pixel clock setting if binning or subsampling has been activated. If you set a higher pixel clock and then reduce the binning/subsampling factors again, the driver will automatically select the highest possible pixel clock for the new settings.



Changes to the image geometry or pixel clock affect the value ranges of the frame rate and exposure time. After executing `is_SetBinning()`, calling the following functions is recommended in order to keep the defined camera settings:

- `is_SetFrameRate()`
- `is_SetExposureTime()`
- If you are using the *uEye's* flash function: `is_SetFlashStrobe()`

Input Parameters

hCam	Camera handle
mode	
IS_BINNING_DISABLE	Disables binning.
IS_BINNING_2X_VERTICAL	Enables vertical binning with factor 2.
IS_BINNING_3X_VERTICAL	Enables vertical binning with factor 3.
IS_BINNING_4X_VERTICAL	Enables vertical binning with factor 4.
IS_BINNING_6X_VERTICAL	Enables vertical binning with factor 6.
IS_BINNING_2X_HORIZONTAL	Enables horizontal binning with factor 2.
IS_BINNING_3X_HORIZONTAL	Enables horizontal binning with factor 3.
IS_BINNING_4X_HORIZONTAL	Enables horizontal binning with factor 4.
IS_BINNING_6X_HORIZONTAL	Enables horizontal binning with factor 6.
IS_GET_BINNING	Returns the current setting.
IS_GET_BINNING_FACTOR_VERTICAL	Returns the vertical binning factor.

IS_GET_BINNING_FACTOR_HORIZONTAL	Returns the horizontal binning factor.
IS_GET_SUPPORTED_BINNING	Returns the supported binning modes.
IS_GET_BINNING_TYPE	Indicates whether the camera uses colour-proof binning (IS_BINNING_COLOR) or not. (IS_BINNING_MONO)



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
Current setting when used together with IS_GET_BINNING IS_GET_BINNING_FACTOR_VERTICAL IS_GET_BINNING_FACTOR_HORIZONTAL	
When used with IS_GET_BINNING_TYPE	Returns IS_BINNING_COLOR if the camera uses colour-proof subsampling; otherwise, IS_BINNING_MONO is returned.
When used with IS_GET_SUPPORTED_BINNING	Returns the supported subsampling modes linked by logical ORs.

Related Functions

- [is_SetSubSampling\(\)](#)
- [is_SetAOI\(\)](#)
- [is_SetImagePos\(\)](#)
- [is_SetPixelClock\(\)](#)

5.3.85 is_SetBlCompensation

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetBlCompensation (HIDS hCam,
                          INT nEnable, INT offset, INT reserved)
```

Description

`is_SetBlCompensation()` enables the black level correction function which might improve the image quality under certain circumstances. By default, the sensor adjusts the black level value for each pixel automatically. If the environment is very bright, it can be necessary to adjust the black level manually.

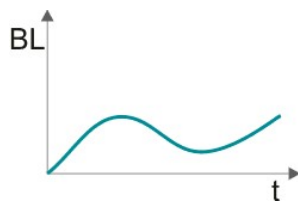


Figure 118: Black level
correction - Auto

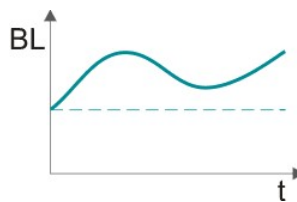


Figure 119: Black level
correction - Auto + offset

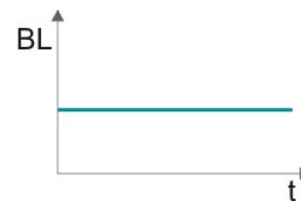


Figure 120:
Schwarzwert-Korrektur nur
Offset



Input Parameters

hCam	Camera handle
nEnable	
IS_BL_COMPENSATION_DISABLE	Disables automatic black level correction. The <code>offset</code> value is used as black level instead. This mode is only supported by sensors of the UI-154x/554x series.
IS_BL_COMPENSATION_ENABLE	Enables automatic black level correction. The <code>offset</code> value is added to the automatic black level value.
IS_GET_BL_COMPENSATION	Returns the current mode.
IS_GET_BL_OFFSET	Returns the currently set value for <code>offset</code> .
IS_GET_BL_DEFAULT_MODE	Returns the default mode.
IS_GET_BL_DEFAULT_OFFSET	Returns the default value for <code>offset</code> .
IS_GET_BL_SUPPORTED_MODE	Returns the supported modes. Possible values: <code>IS_BL_COMPENSATION_ENABLE</code> The sensor supports automatic black level correction. <code>IS_BL_COMPENSATION_OFFSET</code> For the sensor used, it is also possible to set the <code>offset</code> manual.
IS_IGNORE_PARAMETER	The <code>nEnable</code> parameter is ignored.
offset	Contains the offset value used for compensation. Valid values are between 0 and 255.
IS_IGNORE_PARAMETER	The <code>offset</code> parameter is ignored.
reserved	Reserved. 0 must be passed.

Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
Supported modes when used together with <code>IS_GET_BL_SUPPORTED_MODE</code>	
Current mode when used together with <code>IS_GET_BL_COMPENSATION</code>	
Current offset when used together with <code>IS_GET_BL_OFFSET</code>	

5.3.86 `is_SetCameraID`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetCameraID (HIDS hCam, INT nID)
```

Description

Using `is_SetCameraID()`, you can assign a unique camera ID to a camera. Thus, it is possible to access the camera directly with the `is_InitCamera()` function.

The camera ID is stored in the non-volatile memory of the camera. The factory default camera ID is 1. The camera ID can also be changed in the Camera Manager.

Input Parameters

<code>hCam</code>	Camera handle
<code>nID</code>	
1...254	New camera ID
<code>IS_GET_CAMERA_ID</code>	Returns the current ID.



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message
Current ID when used together with <code>IS_GET_CAMERA_ID</code>	

Related Functions

- `is_InitCamera()`
- `is_GetCameraInfo()`
- `is_CameraStatus()`

5.3.87 is_SetCameraLUT

	
GigE	GigE

Syntax

```
INT is_SetCameraLUT (HIDS hCam,
                    UINT Mode, UINT NumberOfEntries,
                    double* pRed_Grey, double* pGreen, double* pBlue)
```

Description

Using `is_SetCameraLUT()`, you can enable a hardware LUT for *GigE uEye HE* cameras. This LUT which will be applied to the image in the camera. A number of predefined LUTs are available. Alternatively, you define your own LUT. It is possible to define a LUT without enabling it at the same time. You can query the current LUT used by the camera by calling the `is_GetCameraLUT()` function.

Each look-up table (LUT) for the *uEye* contains modification values for the image brightness and contrast parameters. When a LUT is used, each brightness value in the image will be replaced by a value from the table. LUTs are typically used to enhance the image contrast or the gamma curve. The values must be in the range between 0.0 and 1.0. A linear LUT containing 64 equidistant values between 0.0 and 1.0 has no effect on the image.

For further information on LUTs, please refer to the [LUT properties](#) section of the *uEye Demo* chapter.



The `is_SetCameraLUT()` function is only supported by cameras of the *GigE uEye* series.

Input Parameters

hCam	Camera handle
Mode	These modes can be linked by a logical OR.
IS_CAMERA_LUT_IDENTITY	Predefined LUT, linear LUT, no image modifications
IS_CAMERA_LUT_NEGATIV	Predefined LUT, inverts the image.
IS_CAMERA_LUT_GLOW1	Predefined LUT, false-colour display of the image
IS_CAMERA_LUT_GLOW2	Predefined LUT, false-colour display of the image
IS_CAMERA_LUT_ASTRO1	Predefined LUT, false-colour display of the image
IS_CAMERA_LUT_RAINBOW1	Predefined LUT, false-colour display of the image
IS_CAMERA_LUT_MAP1	Predefined LUT, false-colour display of the image
IS_CAMERA_LUT_COLD_HOT	Predefined LUT, false-colour display of the image
IS_CAMERA_LUT_SEPIC	Predefined LUT, uses sepia toning for colouring the image.
IS_CAMERA_LUT_ONLY_RED	Predefined LUT, shows only the red channel of the image.
IS_CAMERA_LUT_ONLY_GREEN	Predefined LUT, shows only the green channel of the image.
IS_CAMERA_LUT_ONLY_BLUE	Predefined LUT, shows only the blue channel of the

	image.
<code>IS_SET_CAMERA_LUT_VALUES</code>	Applies the LUT values.
<code>IS_ENABLE_CAMERA_LUT</code>	Enables the LUT. If no other LUT has been defined, the system sets the linear LUT as specified by <code>IS_CAMERA_LUT_IDENTITY</code> .
<code>IS_ENABLE_RGB_GRAYSCALE</code>	The camera converts a colour image to a greyscale image.
<code>NumberOfEntries</code>	Indicates the number of knee points used.
<code>IS_CAMERA_LUT_64</code>	Defines a LUT with 64 knee points. This results in 32 sections with a start and end point each.
<code>pRed_Grey</code>	Array containing the values for the LUT red channel or the LUT greyscale channel (<i>GigE uEye SE</i> cameras).
<code>pGreen</code>	Array containing the values for the LUT green channel.
<code>pBlue</code>	Array containing the values for the LUT blue channel.

Structure of the Knee Point Arrays

The `Red_Grey`, `Green` and `Blue` arrays contain double values between 0.0 and 1.0. The array size must correspond exactly to the value predefined by `NumberOfEntries` (64 or 128). For monochrome cameras, the `Green` and `Blue` parameters are ignored.

The *GigE uEye SE* color and monochrome cameras ignore the `Green` and `Blue` parameters.

GigE uEye HE cameras use all three parameters: `Red_Grey`, `Green` and `Blue`. If you set all three parameters to the same value for *GigE uEye HE* monochrome cameras, the LUT curve creates a monochrome output image. Assigning different values to the three parameters will result in false-color representation.

Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message
<code>IS_NOT_SUPPORTED</code>	This function is not supported by the current camera.

Related Functions

- `is_GetCameraLUT()`

Code Sample



```
//Enable the latest LUT set
INT nRet;
nRet = is_SetCameraLUT (hCam,
                        IS_ENABLE_CAMERA_LUT,
                        IS_CAMERA_LUT_64,
                        NULL, NULL, NULL)

//Set the LUT default "Glow1" and enable it
INT nRet;
double Red[IS_CAMERA_LUT_64];
double Green[IS_CAMERA_LUT_64];
double Blue[IS_CAMERA_LUT_64];

nRet = is_SetCameraLUT (hCam,
```

```
IS_ENABLE_CAMERA_LUT |  
IS_SET_CAMERA_LUT_VALUES |  
IS_CAMERA_LUT_GLOW1,  
IS_CAMERA_LUT_64,  
&Red, &Green, &Blue );
```


5.3.88 `is_SetColorConverter`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetColorConverter (HIDS hCam, INT ColorMode, INT ConvertMode)
```

Description

Using `is_SetColorConverter`, you can select the type of Bayer conversion for colour cameras. Software conversion is done on the PC, while hardware conversion (*Gigabit Ethernet uEye* only) is done in the camera. The use of a larger filter mask results in a higher image quality, but increases the computational load. For further information, please refer to the [Camera Basics: Colour Filters](#) chapter.



Hardware conversion is only supported by *GigE uEye* cameras.



While free run mode is active, you cannot change the colour conversion type. To do so, you must first stop the capturing process using `is_StopLiveVideo()` or set the camera to trigger mode (see also `is_SetExternalTrigger()`).

Input Parameters

<code>hCam</code>	Camera handle
<code>ColorMode</code>	Colour mode for which the converter is to be set. For a list of all available colour formats and the associated input parameters, see the Appendix: Colour and Memory Formats section.
<code>ConvertMode</code>	Conversion mode selection
<code>IS_CONV_MODE_SOFTWARE_3x3</code>	Software conversion using the standard filter mask (default)
<code>IS_CONV_MODE_SOFTWARE_5x5</code>	Software conversion using a large filter mask
<code>IS_CONV_MODE_HARDWARE_3x3</code>	Hardware conversion using the standard filter mask (<i>GigE uEye</i> only)



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message
<code>IS_CAPTURE_RUNNING</code>	This function cannot be executed since the camera is currently in live operation
<code>IS_INVALID_PARAMETER</code>	The <code>ConvertMode</code> parameter is invalid or not supported
<code>IS_INVALID_COLOR_FORMAT</code>	The <code>ColorMode</code> parameter is invalid or not supported

Related Functions

- [is_GetColorConverter\(\)](#)
- [is_SetColorMode\(\)](#)
- [is_ConvertImage\(\)](#)

5.3.89 `is_SetColorCorrection`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetColorCorrection (HIDS hCam, INT nEnable, double* factors)
```

Description

For colour cameras, `is_SetColorCorrection()` enables colour correction in the *uEye* driver. This enhances the rendering of colours for cameras with colour sensors. Colour correction is a digital correction based on a colour matrix which is adjusted individually for each sensor. If you perform Bayer conversion for *GigE uEye HE* colour cameras in the camera itself, colour conversion will automatically also take place in the camera.



After changing this parameter, perform manual or automatic white balancing in order to obtain correct colour rendering (see also `is_SetAutoParameter()`).

Input Parameters

<code>hCam</code>	Camera handle
<code>nEnable</code>	
<code>IS_CCOR_ENABLE_NORMAL</code>	Enables simple colour correction. This parameter replaces <code>IS_CCOR_ENABLE</code> .
<code>IS_CCOR_ENABLE_BG40_ENHANCED</code>	Enables colour correction for cameras with optical IR filter glasses of the BG40 type.
<code>IS_CCOR_ENABLE_HQ_ENHANCED</code>	Enables colour correction for cameras with optical IR filter glasses of the HQ type.
<code>IS_CCOR_DISABLE</code>	Disables colour correction.
<code>IS_GET_CCOR_MODE</code>	Returns the current setting.
<code>IS_GET_SUPPORTED_CCOR_MODE</code>	Returns all supported colour correction modes. See the Return Values section.
<code>IS_GET_DEFAULT_CCOR_MODE</code>	Returns the default colour correction mode.
<code>factors</code>	Currently not used



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
Current setting when used together with IS_GET_CCOR_MODE	
When used together with IS_GET_SUPPORTED_CCOR_MODE	<p>When used for colour cameras and together with IS_GET_SUPPORTED_CCOR_MODE, this parameter returns the supported values linked by a logical OR:</p> <p>IS_CCOR_ENABLE_NORMAL IS_CCOR_ENABLE_BG40_ENHANCED IS_CCOR_ENABLE_HQ_ENHANCED</p> <p>When used for monochrome cameras, the system returns 0.</p>
When used together with IS_GET_DEFAULT_CCOR_MODE	<p>When used for colour cameras and together with IS_GET_DEFAULT_CCOR_MODE, this parameter returns the default colour correction mode:</p> <p>IS_CCOR_ENABLE_NORMAL IS_CCOR_ENABLE_HQ_ENHANCED.</p> <p>When used for monochrome cameras, the system returns 0.</p>

Related Functions

- [is_SetColorConverter\(\)](#)
- [is_SetColorMode\(\)](#)
- [is_SetAutoParameter\(\)](#)

5.3.90 is_SetColorMode

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetColorMode (HIDS hCam, INT Mode)
```

Description

`is_SetColorMode()` sets the colour mode to be used when image data are saved or displayed by the graphics card. For this purpose, the allocated image memory must be large enough to accommodate the data with the selected colour mode. When images are transferred directly to the graphics card memory, make sure that the display settings match the colour mode settings. Otherwise, the images will be displayed with altered colours or are not clearly visible.



For the RGB16 and RGB15 data formats, the MSBs of the internal 8-bit R, G and B colours are used.

Input Parameters

hCam	Camera handle
Mode	Colour mode to be set For a list of all available colour formats and the associated input parameters, see the Appendix: Colour and Memory Formats section.
IS_CM_BAYER_RG16	Raw Bayer (16)
IS_CM_BAYER_RG12	Raw Bayer (12)
IS_CM_BAYER_RG8	Raw Bayer (8)
IS_CM_MONO16	Greyscale (16)
IS_CM_MONO12	Greyscale (12)
IS_CM_MONO8	Greyscale (8)
IS_CM_RGB10V2_PACKED	RGB30 (10 10 10)
IS_CM_RGBA8_PACKED	RGB32 (8 8 8)
IS_CM_RGBY8_PACKED	RGBY (8 8 8 8)
IS_CM_RGB8_PACKED	RGB24 (8 8 8)
IS_CM_BGR10V2_PACKED	BGR30 (10 10 10)
IS_CM_BGRA8_PACKED	BGR32 (8 8 8)
IS_CM_BGR8_PACKED	BGR24 (8 8 8)
IS_CM_BGRY8_PACKED	BGRY (8 8 8)

IS_CM_BGR565_PACKED	BGR16 (5 6 5)
IS_CM_BGR555_PACKED	BGR15 (5 5 5)
IS_CM_UYVY_PACKED	UYVY (8 8 8 8)
IS_CM_UYVY_MONO_PACKED	UYVY (8 8 8 8)
IS_CM_UYVY_BAYER_PACKED	UYVY (8 8 8 8)
IS_CM_CBYCRY_PACKED	CbYCrY (8 8 8 8)
IS_GET_COLOR_MODE	Returns the current setting.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
Current setting when used together with IS_GET_COLOR_MODE	

Related Functions

- [is_SetDisplayMode\(\)](#)
- [is_SetColorConverter\(\)](#)
- [is_SetColorCorrection\(\)](#)
- [is_GetColorDepth\(\)](#)
- [is_AllocImageMem\(\)](#)

5.3.91 `is_SetConvertParam`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetConvertParam (HIDS hCam,
                        BOOL ColorCorrection,
                        INT BayerConversionMode, INT ColorMode, INT Gamma,
                        double* WhiteBalanceMultipliers)
```

Description

Using `is_SetConvertParam()`, you can set the parameters for converting a raw Bayer image to a colour image. To convert the image, use the `is_ConvertImage()` function.

Input Parameters

<code>hCam</code>	Camera handle
<code>ColorCorrection</code>	Enables / disables colour correction.
<code>BayerConversionMode</code>	Sets the Bayer conversion mode.
<code>IS_SET_BAYER_CV_BETTER</code>	Better quality
<code>IS_SET_BAYER_CV_BEST</code>	Optimum quality (higher CPU load)
<code>ColorMode</code>	Sets the colour mode for the output image. For a list of all available colour formats and the associated input parameters, see the Appendix: Colour and Memory Formats section.
<code>Gamma</code>	Gamma value multiplied by 100. Range: [1...1000]
<code>WhiteBalanceMultipliers</code>	Pointer to an array containing the red, green and blue gain values

Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message
<code>IS_INVALID_COLOR_FORMAT</code>	Invalid <code>ColorMode</code> parameter
<code>IS_INVALID_PARAMETER</code>	Other invalid parameter.

Related Functions

- [`is_ConvertImage\(\)`](#)
- [`is_SetColorMode\(\)`](#)
- [`is_SetColorConverter\(\)`](#)

Code Sample

Conversion of a raw Bayer image to RGB24. The memory is allocated automatically.

```
INT nRet;
char * pcSource;
INT nIDSource;
INT nX,nY,nBits,nPitch;

// Create raw Bayer test image
is_AllocImageMem (hCam, 256, 256, 8, &pcSource, &nIDSource);
is_InquireImageMem (hCam, pcSource, nIDSource, &nX, &nY, &nBits, &nPitch);
for (int j = 0; j<nY; j++)
{
    for (int i = 0; i<nX; i++)
    {
        pcSource[i + j * nPitch] = i;
    }
}

// Define conversion parameters (example)
INT Gamma = 120;
double rgbGains[3];
rgbGains[0] = 1.0 ; // Red channel gain
rgbGains[1] = 3.0 ; // Green channel gain
rgbGains[2] = 1.0 ; // Blue channel gain



char* pcDest; // Pointer to the newly allocated image memory
INT nIDDest;  // ID of the newly allocated image memory

// Set conversion parameters
nRet = is_SetConvertParam(hCam, TRUE, IS_SET_BAYER_CV_BETTER, IS_SET_CM_RGB24,
                          Gamma, rgbGains);

// Convert image
if (nRet == IS_SUCCESS)
{
    pcDest = NULL;
    is_ConvertImage(hCam, pcSource, nIDSource, &pcDest, &nIDDest, 0);
}

// Release allocated image memory
is_FreeImageMem (hCam, pcSource, nIDSource);
is_FreeImageMem (hCam, pcDest, nIDDest);
```


5.3.92 is_SetDisplayMode

	
USB 2.0 GigE	-

Syntax

```
INT is_SetDisplayMode (HIDS hCam, INT Mode)
```

Description

Using `is_SetDisplayMode()`, you can set the way in which images will be displayed on the screen.

For live videos including overlays, you can use the Direct3D mode. This mode is not supported by all graphics cards. The graphics card must have sufficient extended memory because Direct3D mode requires additional memory up to the size needed for the current screen resolution.

For further information on the display modes of the *uEye* camera, see the [How To Proceed: Image Display](#) section.



We recommend that you call the following functions exclusively from a single thread in order to avoid unpredictable behaviour of the application.

- [is_InitCamera\(\)](#)
- [is_SetDisplayMode\(\)](#)
- [is_ExitCamera\(\)](#)

See also [Programming: Thread Programming](#).

Input Parameters

hCam	Camera handle
Mode	
IS_SET_DM_DIB	Captures an image in system memory (RAM). Using is_RenderBitmap() , you can define the image display (default).
IS_SET_DM_DIRECT3D	Image display in Direct3D mode
IS_SET_DM_DIRECT3D IS_SET_DM_MONO	Monochrome image display in Direct3D mode
IS_GET_DISPLAY_MODE	Returns the current setting.

The new *Direct3D* mode completely replaces the BackBuffer and Overlay Surface display modes from DirectDraw. It is advisable not to use these modes any longer (see also [Obsolete Functions](#)). To activate the obsolete modes, do the following:

Mode	
IS_SET_DM_DIRECTDRAW IS_SET_DM_BACKBUFFER	Image display in DirectDraw BackBuffer mode
IS_SET_DM_DIRECTDRAW IS_SET_DM_ALLOW_OVERLAY	Image display in DirectDraw Overlay Surface mode
IS_SET_DM_ALLOW_SCALING	Real-time scaling in Overlay Surface mode



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
Current setting when used together with IS_GET_DISPLAY_MODE	

Related Functions

- [is_RenderBitmap\(\)](#)
- [is_SetColorMode\(\)](#)
- [is_DirectRenderer\(\)](#)



Code Sample

```
is_SetDisplayMode (hCam, Mode);  
  
//Bitmap mode (images are digitised and stored in system memory):  
Mode = IS_SET_DM_DIB  
  
//Direct3D mode  
Mode = IS_SET_DM_DIRECT3D
```

Sample Programs

- uEyeOvl (C++)
- uEye VB Simple Demo (VB6)

5.3.93 is_SetDisplayPos

	
USB 2.0 GigE	-

Syntax

```
INT is_SetDisplayPos (HIDS hCam, INT x, INT y)
```

Description

is_SetDisplayPos() allows you to move an area of interest when rendering images using is_RenderBitmap(). This does not alter the image memory contents.

Input Parameters

hCam	Camera handle
x	Offset in x direction
y	Offset in y direction



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- is_SetImagePos()
- is_SetAOI()
- is_RenderBitmap()
- is_SetDisplayMode()

5.3.94 is_SetEdgeEnhancement

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetEdgeEnhancement (HIDS hCam, INT nEnable)
```

Description

`is_SetEdgeEnhancement()` enables a software edge filter. Due to Bayer format colour conversion, the original edges of a colour image may easily become blurred. By enabling the digital edge filter, you can optimise edge representation. This function causes a higher CPU load.

If you perform Bayer conversion for *GigE uEye HE* colour cameras in the camera itself, edge enhancement will automatically also take place in the camera. In this case, the CPU load will not increase.

Input Parameters

hCam	Camera handle
nEnable	
IS_EDGE_EN_DISABLE	Disables the edge filter.
IS_EDGE_EN_STRONG	Enables strong edge enhancement.
IS_EDGE_EN_WEAK	Enables weaker edge enhancement.
IS_GET_EDGE_ENHANCEMENT	Returns the current setting.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
Current setting when used together with IS_GET_EDGE_ENHANCEMENT	

Related Functions

- [`is_SetColorMode\(\)`](#)
- [`is_SetColorConverter\(\)`](#)

5.3.95 `is_SetErrorReport`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetErrorReport (HIDS hCam, INT Mode)
```

Description

Using `is_SetErrorReport()`, you can enable / disable error event logging. If error reporting is enabled, errors will automatically be displayed in a dialogue box. Cancelling the dialogue box disables the error report. Even with disabled error reporting, you can still query errors using the `is_GetError()` function.



`is_SetErrorReport()` can be called before calling `is_InitCamera()`.

You only need to enable the `is_SetErrorReport()` function once for all cameras in the application.

Input Parameters

<code>hCam</code>	Camera handle or 0 if no camera has been initialised yet
<code>Mode</code>	
<code>IS_DISABLE_ERR_REP</code>	Disables error reporting.
<code>IS_ENABLE_ERR_REP</code>	Enables error reporting.



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message
Current setting when used together with <code>IS_GET_ERR_REP_MODE</code>	

Related Functions

- `is_GetError()`
- `is_GetCaptureErrorInfo()`
- `is_CameraStatus()`

5.3.96 is_SetExposureTime

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetExposureTime (HIDS hCam, double EXP, double* newEXP)
```

Description

Using `is_SetExposureTime()`, you can set the exposure time (in milliseconds). Since this value depends on the sensor timing, the exposure time actually used may slightly deviate from the value set here. The actual exposure time is returned by the `newEXP` parameter.

In free-running mode (`is_CaptureVideo()`), any modification of the exposure time will only become effective when the next image but one is captured. In trigger mode (`is_SetExternalTrigger()`), the modification will be applied to the next image.

For minimum and maximum exposure times as well as other sensor-based dependencies, please refer to [Specifications: Sensors](#) chapter.



Newer driver versions sometimes allow an extended value range for the exposure time setting. We recommend to query the value range every time and set the exposure time explicitly.



The use of the following functions will affect the exposure time:

- `is_SetPixelClock()`
- `is_SetOptimalCameraTiming()`
- `is_SetFrameRate()` (if the new image duration is shorter than the exposure time)
- `is_SetAOI()` (if the image size is changed)
- `is_SetSubSampling()`
- `is_SetBinning()`

Changes made to the window size, the frame rate or the read-out timing (pixel clock frequency) also affect the defined exposure time. For this reason, you need to call `is_SetExposureTime()` again after such changes.

Input Parameters

hCam	Camera handle
EXP	New desired exposure time For <code>EXP=0.0</code> , the exposure time is 1/frame rate.
IS_GET_EXPOSURE_TIME	Returns the current exposure time in the <code>newEXP</code> parameter.
IS_GET_DEFAULT_EXPOSURE	Returns the default exposure time.
IS_SET_ENABLE_AUTO_SHUTTER	Enables the auto exposure function.
newEXP	Returns the exposure time actually set.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- is_GetExposureRange()
- is_SetFrameRate()
- is_SetPixelClock()
- is_SetOptimalCameraTiming()
- is_SetAutoParameter()
- is_SetHardwareGain()

5.3.97 is_SetExternalTrigger

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetExternalTrigger (HIDS hCam, INT nTriggerMode)
```

Description

Using `is_SetExternalTrigger()`, you can activate the trigger mode. If the camera is in standby mode, it quits this mode and activates trigger mode.

In hardware trigger mode, image capture is delayed for each function call until the selected trigger event has occurred.

In software trigger mode, an image is captured immediately when `is_FreezeVideo()` is called, or a continuous triggered capture is started when `is_CaptureVideo()` is called. In hardware trigger mode, you can use the `is_ForceTrigger()` command to trigger an image capture even if no electric signal is present.

When you disable the trigger functionality, you can statically query the signal level at the trigger input. This option causes the camera to change to freerun mode.

For further information on the image capture modes of the *uEye* camera, see the [How To Proceed: Image Capture](#) section.



For hardware reasons, the board-level versions of the *USB uEye LE* cameras can only be triggered on the falling edge.

Input Parameters

hCam	Camera handle	
nTriggerMode	Trigger mode	Trigger event
IS_SET_TRIGGER_OFF	Off	-
IS_SET_TRIGGER_HI_LO	Hardware trigger	Falling signal edge
IS_SET_TRIGGER_LO_HI	Hardware trigger	Rising signal edge
IS_SET_TRIGGER_HI_LO_SYNC	Freerun sync./ hardware trigger *)	Falling signal edge
IS_SET_TRIGGER_LO_HI_SYNC	Freerun sync./ hardware trigger *)	Rising signal edge
IS_SET_TRIGGER_SOFTWARE	Software trigger	Call of <code>is_FreezeVideo()</code> (single frame mode) Call of <code>is_CaptureVideo()</code> (continuous mode)
IS_GET_EXTERNALTRIGGER	Returns the trigger mode setting	
IS_GET_TRIGGER_STATUS	Returns the current signal level at the trigger input	
IS_GET_SUPPORTED_TRIGGER_MODE	Returns the supported trigger modes	



*) The freerun synchronisation mode is currently only supported by the UI-146x/546x series sensors.

Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message
Current setting when used together with <code>IS_GET_EXTERNALTRIGGER</code>	
When used with <code>IS_GET_TRIGGER_STATUS</code>	Returns the current signal level at the trigger input
When used with <code>IS_GET_SUPPORTED_TRIGGER_MODE</code>	Returns the supported modes linked by logical ORs

Related Functions

- `is_CaptureVideo()`
- `is_FreezeVideo()`
- `is_ForceTrigger()`
- `is_SetTriggerCounter()`
- `is_SetTriggerDelay()`
- `is_SetFlashStrobe()`



Code Sample

```
//Enable trigger mode and set high-active flash mode.
is_SetExternalTrigger (hCam, IS_SET_TRIGGER_SOFTWARE);
is_SetFlashStrobe (hCam, IS_SET_FLASH_HI_ACTIVE, 0);
is_FreezeVideo (hCam, IS_WAIT);
```

Sample Programs

- uEye Simple Trigger (C++)
- uEye IO (C++)

5.3.98 is_SetFlashDelay

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetFlashDelay (HIDS hCam, ULONG ulDelay, ULONG ulDuration)
```

Description

`is_SetFlashDelay()` sets a delay for driving the flash output. In addition, you can specify the flash duration. This allows the implementation of a global flash functionality which exposes all rows of a rolling shutter sensor. In addition, it is possible, for a camera with global shutter sensors, to set the flash start in free-run mode to the start time of the exposure window. For further information, please refer to the [Digital Output \(Flash/Strobe\)](#) and [Shutter Methods](#) chapters.

Input Parameters

<code>hCam</code>	Camera handle
<code>ulDelay</code>	Time by which the flash start is delayed (in μ s), default = 0
<code>IS_GET_FLASH_DELAY</code>	Returns the currently set delay time.
<code>IS_GET_FLASH_DURATION</code>	Returns the currently set flash duration.
<code>IS_GET_MIN_FLASH_DELAY</code>	Returns the minimum value for the delay.
<code>IS_GET_MIN_FLASH_DURATION</code>	Returns the minimum value for the flash duration.
<code>IS_GET_MAX_FLASH_DELAY</code>	Returns the maximum value for the delay.
<code>IS_GET_MAX_FLASH_DURATION</code>	Returns the maximum value for the flash duration.
<code>IS_GET_FLASH_DELAY_GRANULARITY</code>	Returns the increment of the adjustable delay time.
<code>IS_GET_FLASH_DURATION_GRANULARITY</code>	Returns the increment of the adjustable flash duration.
<code>ulDuration</code>	Time during which the flash is on (in μ s). If 0 is passed, the flash output will be active until the end of the exposure time.



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message
Current setting when used together with <code>IS_GET_FLASH_DELAY</code> <code>IS_GET_FLASH_DURATION</code>	

Related Functions

- [is_SetFlashStrobe\(\)](#)
- [is_GetGlobalFlashDelays\(\)](#)
- [is_SetExternalTrigger\(\)](#)
- [is_SetTriggerDelay\(\)](#)

5.3.99 `is_SetFlashStrobe`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetFlashStrobe (HIDS hCam, INT nMode, INT nLine)
```

Description

`is_SetFlashStrobe()` enables / disables the flash output of the *uEye* camera. In addition, you can set the active level (high or low). By default, the strobe is set to high-active. Alternatively, the strobe output can be used statically as a digital output.

You can set the duration of the flash and the flash delay using the `is_SetFlashDelay()` function. In order to obtain precise synchronization of flash output and exposure time, we recommend to use the values returned by the `is_GetGlobalFlashDelays()` function for flash delay and duration.



When you are using the *uEye*'s flash function, you need to re-enable the flash (i.e. disable and then activate it again) whenever you change the pixel clock setting or horizontal image geometry. This is necessary to newly synchronise the internal timing settings of the flash output with the start of sensor exposure.

Input Parameters

<code>hCam</code>	Camera handle
<code>nMode</code>	
<code>IS_SET_FLASH_OFF</code>	Disables the strobe output.
<code>IS_SET_FLASH_LO_ACTIVE</code>	Sets the strobe output to low-active (trigger mode only).
<code>IS_SET_FLASH_HI_ACTIVE</code>	Sets the strobe output to high-active (trigger mode only).
<code>IS_SET_FLASH_LO_ACTIVE_FREERUN</code>	Sets the strobe output to low-active (freerun mode only).
<code>IS_SET_FLASH_HI_ACTIVE_FREERUN</code>	Sets the strobe output to high-active (freerun mode only).
<code>IS_SET_FLASH_HIGH</code>	Sets the strobe output statically to high.
<code>IS_SET_FLASH_LOW</code>	Sets the strobe output statically to low.
<code>IS_GET_FLASHSTROBE_MODE</code>	Returns the current mode.
<code>IS_SET_FLASH_IO_1</code>	Enable additionally the flash functions for GPIO port 1
<code>IS_SET_FLASH_IO_2</code>	Enable additionally the flash functions for GPIO port 1
<code>IS_GET_SUPPORTED_FLASH_IO</code>	Returns the supported flash-enabled I/O ports
<code>nLine</code>	Currently not used.

Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
Current setting when used together with IS_GET_FLASHSTROBE_MODE	
When used with IS_GET_FLASHSTROBE_MODE	The system returns the supported modes linked by a logical OR.

Related Functions

- [is_SetFlashDelay\(\)](#)
- [is_SetTriggerDelay\(\)](#)
- [is_GetGlobalFlashDelays\(\)](#)
- [is_SetExternalTrigger\(\)](#)
- [is_CaptureVideo\(\)](#)



Code Sample

```
//Enable trigger mode and set high-active flash mode.  
is_SetExternalTrigger (hCam, IS_SET_TRIGGER_SOFTWARE);  
is_SetFlashStrobe (hCam, IS_SET_FLASH_HI_ACTIVE, 0);  
is_FreezeVideo (hCam, IS_WAIT);  
  
// Enable additional flash output on GPIO 2  
is_SetFlashStrobe (hCam, IS_SET_FLASH_HI_ACTIVE | IS_SET_FLASH_IO_2, 0);  
  
// Flash function only on normal digital output  
is_SetFlashStrobe (hCam, IS_SET_FLASH_HI_ACTIVE, 0);
```

Sample Programs

- uEyeFlashStrobe (C++)
- uEyeIO (C++)

5.3.100 `is_SetFrameRate`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetFrameRate (HIDS hCam, double FPS, double* newFPS)
```

Description

Using `is_SetFrameRate()`, you can set the sensor frame rate in freerun mode (live mode). Since this value depends on the sensor timing, the exposure time actually used may slightly deviate from the value set here. After you have called the function, the actual frame rate is returned through the `newFPS` parameter.

If the frame rate is set too high, it might not be possible to transfer every single frame. In this case, the effective frame rate may vary from the set value.

For minimum and maximum frame rates as well as other sensor-based dependencies, please refer to [Specifications: Sensors](#) chapter.



The use of the following functions will affect the frame rate:

- `is_SetPixelClock()`
- `is_SetOptimalCameraTiming()`
- `is_SetAOI()` (if the image size is changed)
- `is_SetSubSampling()`
- `is_SetBinning()`

Changes made to the window size or the read-out timing (pixel clock frequency) also affect the defined frame rate. For this reason, you need to call `is_SetFrameRate()` again after such changes.



Newer driver versions sometimes allow an extended value range for the frame rate setting. We recommend to query the value range every time and set the frame rate explicitly.

Changes to the frame rate affect the value ranges of the exposure time. After executing `is_SetFrameRate()`, calling the function `is_SetExposureTime()` is recommended in order to keep the defined camera settings.

Input Parameters

<code>hCam</code>	Camera handle
<code>FPS</code>	Desired frame rate in frames per second (fps)
<code>IS_GET_FRAMERATE</code>	Only returns the current frame rate in the <code>newFPS</code> parameter.
<code>IS_GET_DEFAULT_FRAMERATE</code>	Returns the default frame rate.
<code>newFPS</code>	Returns the frame rate actually set.



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message

Related Functions

- [is_GetFramesPerSecond\(\)](#)
- [is_GetFrameTimeRange\(\)](#)
- [is_SetPixelClock\(\)](#)
- [is_SetOptimalCameraTiming\(\)](#)
- [is_SetExposureTime\(\)](#)
- [is_SetAutoParameter\(\)](#)
- [is_SetAOI\(\)](#)
- [is_SetSubSampling\(\)](#)
- [is_SetBinning\(\)](#)
- [is_CaptureVideo\(\)](#)

5.3.101 `is_SetGainBoost`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetGainBoost (HIDS hCam, INT mode)
```

Description

In some cameras, `is_SetGainBoost()` enables an additional analogue hardware gain boost feature on the sensor.

Input Parameters

<code>hCam</code>	Camera handle
<code>mode</code>	
<code>IS_GET_GAINBOOST</code>	Returns the current state of the gain boost function.
<code>IS_SET_GAINBOOST_ON</code>	Enables the gain boost function.
<code>IS_SET_GAINBOOST_OFF</code>	Disables the gain boost function.
<code>IS_GET_SUPPORTED_GAINBOOST</code>	Indicates whether the camera supports a gain boost feature or not.



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message
Current setting when used together with <code>IS_GET_GAINBOOST</code>	Returns 0 if the camera does not support a gain boost feature.
Current setting when used together with <code>IS_GET_SUPPORTED_GAINBOOST</code>	Returns <code>IS_SET_GAINBOOST_ON</code> if the function is supported, otherwise it returns <code>IS_SET_GAINBOOST_OFF</code> .

Related Functions

- [`is_SetHardwareGain\(\)`](#)
- [`is_SetHWGainFactor\(\)`](#)
- [`is_SetAutoParameter\(\)`](#)

5.3.102 is_SetGamma

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetGamma (HIDS hCam, INT nGamma)
```

Description

`is_SetGamma()` sets the value for digital gamma correction (brighten dark image portions by applying a non-linear characteristic (LUT)). Valid values are in the range between 0.01 and 10.

Input Parameters

hCam	Camera handle
nGamma	Gamma value to be set, multiplied by 100 (Range: 1...1000. Default = 100, corresponds to a gamma value of 1.0)
IS_GET_GAMMA	Returns the current setting.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
Current setting when used together with IS_GET_GAMMA	

Related Functions

- [`is_SetHardwareGamma\(\)`](#)

5.3.103 `is_SetGlobalShutter`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetGlobalShutter (HIDS hCam, INT mode)
```

Description

`is_SetGlobalShutter()` enables the Global Start shutter function on some sensors.

For further information on the Global Start shutter mode, please refer to the [Camera Basics: Shutter Methods](#) chapter.



The Global Start shutter function is only supported in trigger mode (see also [is_SetExternalTrigger\(\)](#)).

Input Parameters

<code>hCam</code>	Camera handle
<code>mode</code>	
<code>IS_GET_GLOBAL_SHUTTER</code>	Returns the current mode or <code>IS_NOT_SUPPORTED</code> if the camera does not support this function.
<code>IS_SET_GLOBAL_SHUTTER_ON</code>	Enables Global Start shutter mode.
<code>IS_SET_GLOBAL_SHUTTER_OFF</code>	Disables Global Start shutter mode.
<code>IS_GET_SUPPORTED_GLOBAL_SHUTTER</code>	Indicates whether the connected camera supports the Global Start shutter or not.



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message
Current setting when used together with <code>IS_GET_GLOBAL_SHUTTER</code>	
When used together with <code>IS_GET_SUPPORTED_GLOBAL_SHUTTER</code>	Returns <code>IS_SET_GLOBAL_SHUTTER_ON</code> if this function is supported. Otherwise, it returns <code>IS_SET_GLOBAL_SHUTTER_OFF</code> .

Related Functions

- [is_GetGlobalFlashDelays\(\)](#)
- [is_SetExternalTrigger\(\)](#)
- [is_SetFlashStrobe\(\)](#)

5.3.104 is_SetHardwareGain

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetHardwareGain (HIDS hCam, INT nMaster,
                      INT nRed, INT nGreen, INT nBlue)
```

Description

`is_SetHardwareGain()` controls the sensor gain channels. These can be set between 0% and 100% independently of each other. The actual gain factor obtained for the value 100% depends on the sensor and is specified in [Specifications: Sensors](#) chapter.

You can use the `is_GetSensorInfo()` function to query the available gain controls.

Depending on the time when the gain settings are changed, these changes might only become effective when the next image is captured.



Enabling hardware gain increases not only the image brightness, but also the image noise. We recommend to use gain values below 50 for normal operation.



The default setting values for the red, green and blue channel gain factors depend on the colour correction matrix that has been set. If you select a different colour correction matrix, the returned default values might change (see also `is_SetColorCorrection()`).

Input Parameters

<code>hCam</code>	Camera handle
<code>nMaster</code>	Sets the overall gain factor (0...100).
<code>IS_IGNORE_PARAMETER</code>	The master gain factor will not be changed.
<code>IS_GET_MASTER_GAIN</code>	Returns the master gain factor.
<code>IS_GET_RED_GAIN</code>	Returns the red channel gain factor.
<code>IS_GET_GREEN_GAIN</code>	Returns the green channel gain factor.
<code>IS_GET_BLUE_GAIN</code>	Returns the blue channel gain factor.
<code>IS_GET_DEFAULT_MASTER</code>	Returns the default master gain factor.
<code>IS_GET_DEFAULT_RED</code>	Returns the default red channel gain factor.
<code>IS_GET_DEFAULT_GREEN</code>	Returns the default green channel gain factor.
<code>IS_GET_DEFAULT_BLUE</code>	Returns the default blue channel gain factor.
<code>IS_SET_ENABLE_AUTO_GAIN</code>	Enables the auto gain functionality (see also <code>is_SetAutoParameter()</code>). You can disable the auto gain functionality by setting a value for <code>nMaster</code> .
<code>nRed</code>	Sets the red channel gain factor (0...100).
<code>IS_IGNORE_PARAMETER</code>	The channel gain factor will not be changed.
<code>nGreen</code>	Sets the green channel gain factor (0...100).

IS_IGNORE_PARAMETER	The green channel gain factor will not be changed.
nBlue	Sets the blue channel gain factor (0...100).
IS_IGNORE_PARAMETER	The blue channel gain factor will not be changed.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
Current setting When used together with IS_GET_MASTER_GAIN IS_GET_RED_GAIN IS_GET_GREEN_GAIN IS_GET_BLUE_GAIN	
IS_INVALID_MODE	Camera is in standby mode, function not allowed.

Related Functions

- is_SetHWGainFactor()
- is_GetSensorInfo()
- is_SetGainBoost()
- is_SetAutoParameter()

5.3.105 is_SetHardwareGamma

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetHardwareGamma (HIDS hCam, INT nMode)
```

Description

`is_SetHardwareGamma()` enables the hardware gamma control feature of the camera.



The `is_SetHardwareGamma()` function is only supported by cameras of the *GigE uEye* series.

Input Parameters

hCam	Camera handle
nMode	
IS_GET_HW_SUPPORTED_GAMMA	Indicates whether the camera supports hardware gamma control or not.
IS_SET_HW_GAMMA_ON	Enables the gamma control feature.
IS_SET_HW_GAMMA_OFF	Disables gamma control.
IS_GET_HW_GAMMA	Returns the current state of gamma control.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
IS_NOT_SUPPORTED	Unsupported sensor
Current setting when used together with IS_GET_HW_GAMMA	
When used together with IS_GET_HW_SUPPORTED_GAMMA	IS_SET_HW_GAMMA_ON The camera supports gamma control. IS_SET_HW_GAMMA_OFF The camera does not support gamma control.

Related Functions

- [`is_SetGamma\(\)`](#)

5.3.106 `is_SetHdrKneepoints`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetHdrKneepoints (HIDS hCam,
                        KNEEPOINTARRAY* KneepointArray,
                        INT KneepointArraySize)
```

Description

Using `is_SetHdrKneepoints()`, you can define settings for the HDR mode (High Dynamic Range) which is supported by some sensors. You can enable / disable HDR mode by calling `is_EnableHdr()`.

For further information on HDR mode, please refer to the [HDR properties](#) section of the *uEye Demo* chapter.

Input Parameters

<code>hCam</code>	Camera handle
<code>KneepointArray</code>	Pointer to a field (see below).

Contents of the KNEEPOINTARRAY Field

<code>INT NumberOfUsedKneepoints</code>	Number of knee points used.
<code>KNEEPOINT Kneepoint[10]</code>	Knee point

Contents of the KNEEPOINT Structure

<code>double x</code>	Knee point x value
<code>double y</code>	Knee point y value

The x value of a knee point indicates the first phase of the currently set exposure time (in %). The y value indicates the proportion of maximum pixel intensity in percent. If two knee points are used, you can set two phases in which the images will not be exposed. This means that two corresponding times will be set on the x-axis.

For instance, the effects of setting `x = 60`, `y = 80` would be as follows: The first exposure phase takes up 60% of the set exposure time. In this first exposure phase, all pixels are exposed up to a maximum of 80% of the maximum pixel intensity and remain at 80% until this phase is over. In the second exposure phase, they are exposed again and may reach the full pixel intensity.

Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message
<code>IS_NOT_SUPPORTED</code>	Unsupported sensor



Related Functions

- [is_EnableHdr\(\)](#)
- [is_GetHdrMode\(\)](#)
- [is_GetHdrKneepointInfo\(\)](#)
- [is_GetHdrKneepoints\(\)](#)

Sample Programs

- uEyeC# Hdr Demo (C#)
- uEye VB Hdr Demo (VB6)

5.3.107 `is_SetHWGainFactor`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetHWGainFactor (HIDS hCam, INT nMode, INT nFactor)
```

Description

`is_SetHWGainFactor()` uses gain factors to control sensor gain channels. These channels can be set independently of each other. The `is_SetHardwareGain()` does not use factors for setting the gain channels, but standardised values between 0 and 100. The actual gain factor is sensor-dependent and can be found in [Specifications: Sensors](#) chapter.

You can use the `is_GetSensorInfo()` function to query the available gain controls. Depending on the time when the gain settings are changed, these changes might only become effective when the next image is captured.

Input Parameters

<code>hCam</code>	Camera handle
<code>nMode</code>	
<code>IS_GET_MASTER_GAIN_FACTOR</code>	Returns the master gain factor.
<code>IS_GET_RED_GAIN_FACTOR</code>	Returns the red channel gain factor.
<code>IS_GET_GREEN_GAIN_FACTOR</code>	Returns the green channel gain factor.
<code>IS_GET_BLUE_GAIN_FACTOR</code>	Returns the blue channel gain factor.
<code>IS_SET_MASTER_GAIN_FACTOR</code>	Sets the master gain factor.
<code>IS_SET_RED_GAIN_FACTOR</code>	Sets the red channel gain factor.
<code>IS_SET_GREEN_GAIN_FACTOR</code>	Sets the green channel gain factor.
<code>IS_SET_BLUE_GAIN_FACTOR</code>	Sets the blue channel gain factor.
<code>IS_GET_DEFAULT_MASTER_GAIN_FACTOR</code>	Returns the default master gain factor.
<code>IS_GET_DEFAULT_RED_GAIN_FACTOR</code>	Returns the default red channel gain factor.
<code>IS_GET_DEFAULT_GREEN_GAIN_FACTOR</code>	Returns the default green channel gain factor.
<code>IS_GET_DEFAULT_BLUE_GAIN_FACTOR</code>	Returns the default blue channel gain factor.
<code>IS_INQUIRE_MASTER_GAIN_FACTOR</code>	Converts the index value for the master gain factor.
<code>IS_INQUIRE_RED_GAIN_FACTOR</code>	Converts the index value for the red channel gain factor.
<code>IS_INQUIRE_GREEN_GAIN_FACTOR</code>	Converts the index value for the green channel gain factor.
<code>IS_INQUIRE_BLUE_GAIN_FACTOR</code>	Converts the index value for the blue channel gain factor.
<code>nFactor</code>	Gain value (100 = gain factor 1, i. e. no effect)

For converting a gain value from the `is_SetHardwareGain()` function, you can set the `nMode` parameter to one of the `IS_INQUIRE_x_FACTOR` values. In this case, the value range for `nFactor` is

between 0 and 100.

To set the gain using `IS_SET..._GAIN_FACTOR`, you must set the `nFactor` parameter to an integer value in the range from 100 to the maximum value. By calling `IS_INQUIRE_x_FACTOR` and specifying the value 100 for `nFactor`, you can query the maximum value. A gain value of 100 means no gain, a gain value of 200 means gain to the double level (factor 2), etc.

Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message
Current setting when used together with <code>IS_GET_MASTER_GAIN_FACTOR</code> <code>IS_GET_RED_GAIN_FACTOR</code> <code>IS_GET_GREEN_GAIN_FACTOR</code> <code>IS_GET_BLUE_GAIN_FACTOR</code>	
Defined setting when used together with <code>IS_SET_MASTER_GAIN_FACTOR</code> <code>IS_SET_RED_GAIN_FACTOR</code> <code>IS_SET_GREEN_GAIN_FACTOR</code> <code>IS_SET_BLUE_GAIN_FACTOR</code> .	
Default setting when used together with <code>IS_GET_DEFAULT_MASTER_GAIN_FACTOR</code> <code>IS_GET_DEFAULT_RED_GAIN_FACTOR</code> <code>IS_GET_DEFAULT_GREEN_GAIN_FACTOR</code> <code>IS_GET_DEFAULT_BLUE_GAIN_FACTOR</code> .	
When used together with <code>IS_INQUIRE_MASTER_GAIN_FACTOR</code> <code>IS_INQUIRE_RED_GAIN_FACTOR</code> <code>IS_INQUIRE_GREEN_GAIN_FACTOR</code> <code>IS_INQUIRE_BLUE_GAIN_FACTOR</code> .	Converted gain index

Related Functions



- [`is_SetHardwareGain\(\)`](#)
- [`is_SetHardwareGamma\(\)`](#)
- [`is_SetGainBoost\(\)`](#)
- [`is_SetAutoParameter\(\)`](#)
- [`is_GetSensorInfo\(\)`](#)

Code Sample

```
//Set master gain factor to 3.57:
INT ret = is_SetHWGainFactor(hCam, IS_SET_MASTER_GAIN_FACTOR, 357);
//ret has the value 363 for the UI-1460-C

//Query the maximum gain factor for the red channel:
ret = is_SetHWGainFactor(hCam, IS_INQUIRE_RED_GAIN_FACTOR, 100);
//ret has the value 725 for the UI-1460-C
```


5.3.108 `is_SetImageMem`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetImageMem (HIDS hCam, char* pcImgMem, INT id)
```

Description

`is_SetImageMem()` makes the specified image memory the active memory. Only an active image memory can receive image data. When you call `is_FreezeVideo()`, the captured image is stored in the image buffer designated by `pcImgMem` and `id`. For `pcImgMem`, you must pass a pointer which was created by `is_AllocImageMem()`, passing any other pointer will result in an error message. You may pass the same pointer multiple times.



In the Direct3D modes, there is no need to set an image memory.



If you want the application to be compatible with the *FALCON SDK*, make sure to call `is_SetImageSize()` after `is_SetImageMem()`.

Input Parameters

<code>hCam</code>	Camera handle
<code>pcImgMem</code>	Pointer to the starting position in the memory.
<code>id</code>	ID of this memory.



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message

Related Functions

- `is_AllocImageMem()`
- `is_FreeImageMem()`
- `is_AddToSequence()`
- `is_SetAllocatedImageMem()`
- `is_GetColorDepth()`
- `is_GetImageMem()`
- `is_GetImageMemPitch()`

5.3.109 is_SetImagePos

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetImagePos (HIDS hCam, INT x, INT y)
```

Description

`is_SetImagePos()` determines the position of an area of interest (AOI) in the display window. When used together with the `is_SetAOI()` function, you can cut out an area of interest of the full video image.

To avoid a positional mismatch between the display area and the image area, make sure to call the functions in the correct order. Starting from the original image, it is mandatory to keep to the following order:

- `is_SetAOI()`
- `is_SetImagePos()`



With `is_SetAOI()`, you can set the position and size of an area of interest using a single function call.



Changes to the image geometry or pixel clock affect the value ranges of the frame rate and exposure time. After executing `is_SetBinning()`, calling the following functions is recommended in order to keep the defined camera settings:

- `is_SetFrameRate()`
- `is_SetExposureTime()`
- If you are using the *uEye's* flash function: `is_SetFlashStrobe()`

Input Parameters

The `x` and `y` parameters represent an offset with respect to the upper left image corner. The cut window is copied to the start position in the memory. If you want the image to be copied to the same offset within the memory, you can link the new position with a logical OR to the `IS_SET_IMAGE_POS_X_ABS` and `IS_SET_IMAGE_POS_Y_ABS` parameters.

<code>hCam</code>	Camera handle
<code>x</code>	
<code>0 ... xMax</code>	Sets the horizontal position
<code>0 ... xMax</code> <code>IS_SET_IMAGE_POS_X_ABS</code>	Applies the absolute position to the memory as well.
<code>IS_GET_IMAGE_POS_X</code>	Returns the current <code>x</code> position.
<code>IS_GET_IMAGE_POS_X_MIN</code>	Returns the minimum value for the horizontal AOI position.
<code>IS_GET_IMAGE_POS_X_MAX</code>	Returns the maximum value for the horizontal AOI position.
<code>IS_GET_IMAGE_POS_X_INC</code>	Returns the increment for the horizontal AOI position.
<code>IS_GET_IMAGE_POS_X_ABS</code>	Returns the absolute horizontal position in the memory.

<code>IS_GET_IMAGE_POS_Y</code>	Returns the current Y position.
<code>IS_GET_IMAGE_POS_Y_MIN</code>	Returns the minimum value for the vertical AOI position.
<code>IS_GET_IMAGE_POS_Y_MAX</code>	Returns the maximum value for the vertical AOI position.
<code>IS_GET_IMAGE_POS_Y_INC</code>	Returns the increment for the vertical AOI position.
<code>IS_GET_IMAGE_POS_Y_ABS</code>	Returns the absolute vertical position in the memory.
<code>y</code>	
<code>0...yMax</code>	Sets the vertical position
<code>0...yMax</code> <code>IS_SET_IMAGE_POS_Y_ABS</code>	Applies the absolute position to the memory as well.
<code>0</code>	When returning settings via parameter x (s. above)

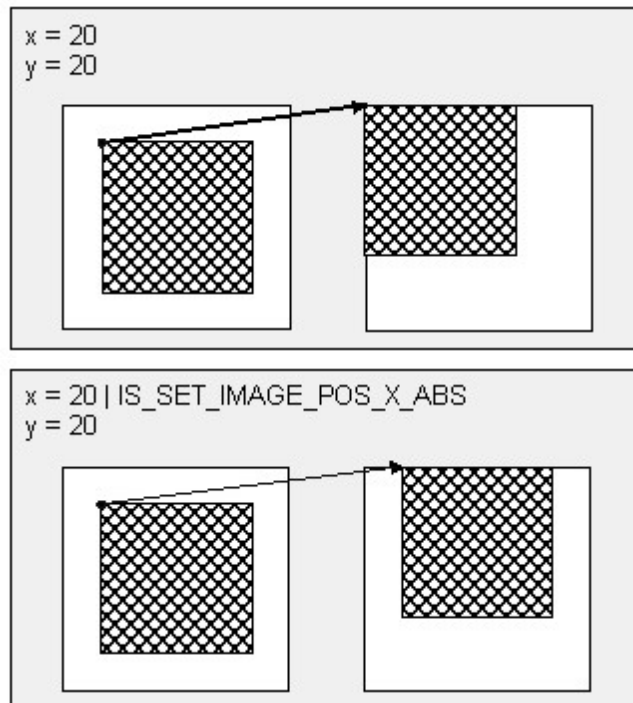
Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message
<code>IS_INVALID_PARAMETER</code>	Parameters <code>x</code> or <code>y</code> are invalid (<code>x, y < 0</code>)
Current setting when used together with <code>IS_GET_IMAGE_POS</code> parameters	
<code>IS_INVALID_MODE</code>	Camera is in standby mode, function not allowed.

Related Functions

- `is_SetAOI()`

Example



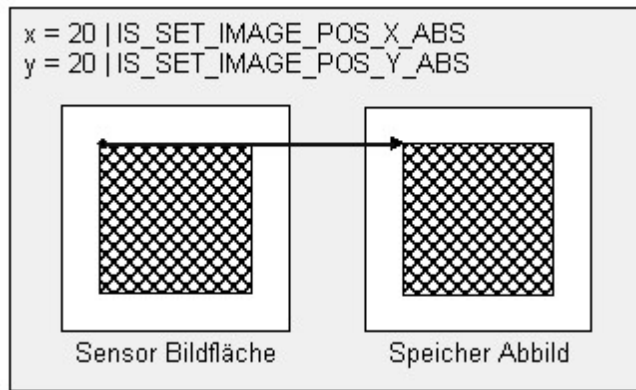




Figure 121: Examples for `is_SetImagePos`

5.3.110 `is_SetIO`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetIO (HIDS hCam, INT nIO)
```

Description

`is_SetIO()` sets the additional digital outputs (GPIOs) of the uEye or returns their current states. Using `is_SetIOMask()`, you can define each GPIO as a digital input or output.



The GPIOs are available with the *GigE uEye HE* and *USB uEye ME/LE* (board level only) cameras. The GPIOs are not provided with optocouplers and use TTL voltages. For information on GPIO wiring, please refer to the [Electrical Specifications](#) chapter.



To connect and control a flash (strobe) unit for the *uEye* cameras, it is recommended to use the flash output provided (see `is_SetFlashStrobe()`).

Input Parameters

<code>hCam</code>	Camera handle
<code>nIO</code>	Bit mask for outputs
<code>0x00 (00)</code>	Sets both outputs to 0.
<code>0x01 (01)</code>	Sets the first output to 1, the second one to 0.
<code>0x02 (10)</code>	Sets the first output to 0, the second one to 1.
<code>0x03 (11)</code>	Sets both outputs to 1.
<code>IS_GET_IO</code>	Reads the signal applied to the input.



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message
Current setting when used together with <code>IS_GET_IO</code>	

Related Functions

- `is_SetIOMask()`
- `is_GetImageInfo()`
- `is_SetFlashStrobe()`
- `is_SetExternalTrigger()`

5.3.111 is_SetIOMask

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetIOMask (HIDS hCam, INT nMask)
```

Description

Using `is_SetIOMask()`, you can define each GPIO as an input or output. The `is_SetIO()` function sets the additional digital outputs (GPIOs) of the uEye or returns the current states.



The GPIOs are available with the *GigE uEye HE* and *USB uEye ME/LE* (board level only) cameras. The GPIOs are not provided with optocouplers and use TTL voltages. For information on GPIO wiring, please refer to the [Electrical Specifications](#) chapter.



To use hardware triggering with the *uEye* cameras, we suggest that you use the trigger input provided for this purpose (see `is_SetExternalTrigger()`).

To connect and control a flash (strobe) unit for the *uEye* cameras, it is recommended to use the flash output provided (see `is_SetFlashStrobe()`).

Input Parameters

hCam	Camera handle
nMask	Bit mask for inputs / outputs.
0x00 (00)	Use both GPIOs as inputs.
0x01 (01)	Use the first GPIO as output, the second one as input.
0x02 (10)	Use the first GPIO as input, the second one as output.
0x03 (11)	Use both GPIOs as outputs.
IS_GET_IO_MASK	Returns the current bit mask.
IS_GET_INPUT_MASK	Returns the IOs to be used as inputs.
IS_GET_OUTPUT_MASK	Returns the IOs to be used as outputs.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
Current setting when used together with IS_GET_IO	
When used with IS_GET_INPUT_MASK IS_GET_OUTPUT_MASK.	Bit masks of the IOs to be used.

Related Functions

- is_SetIO()
- is_GetImageInfo()
- is_SetFlashStrobe()
- is_SetExternalTrigger()

5.3.112 is_SetLED

	
USB 2.0	USB 2.0

Syntax

```
INT is_SetLED (HIDS hCam, INT nValue)
```

Description

Using `is_SetLED()`, you can toggle the colour of the LED on the back of the *USB uEye* camera housing.



The `is_SetLED()` function is only supported by cameras of the *USB uEye SE* and *USB uEye RE* series.

Input Parameters

hCam	Camera handle
nValue	
IS_SET_LED_OFF	Switches LED to red.
IS_SET_LED_ON	Switches LED to green.
IS_SET_LED_TOGGLE	Toggles between red and green.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [`is_SetIO\(\)`](#)
- [`is_SetFlashStrobe\(\)`](#)
- [`is_SetExternalTrigger\(\)`](#)

5.3.113 is_SetOptimalCameraTiming

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetOptimalCameraTiming (HIDS hCam,
                              INT Mode, INT Timeout, INT* pMaxPx1Clk,
                              double* pMaxFrameRate)
```

Description

Using `is_SetOptimalCameraTiming()`, you can determine the highest possible pixel clock frequency for the current configuration. This function sets the pixel clock for which no transfer errors will occur during the `Timeout` period. Moreover, it returns the highest frame rate available for this pixel clock frequency. `is_SetOptimalCameraTiming()` can only be executed in free-run mode (`is_CaptureVideo()`). If the return value is `IS_SUCCESS`, no clock setting will be made.



The function should be executed in a separate thread and run in the background to allow for the computational load caused by additional colour conversions, etc. Otherwise, it will not be able to return the optimum values.

Input Parameters

hCam	Camera handle
Mode	
IS_BEST_PCLK_RUN_ONCE	The function makes one attempt to determine the optimum pixel clock and returns immediately.
Timeout [4000...20000]	Sets the period (in milliseconds) during which no transfer error may occur. The adjustable range is between 4 and 20 seconds. The higher the value you set for this parameter, the more stable the determined pixel clock value will be. This, in turn, increases the runtime of the function correspondingly.
pMaxPx1Clk	Returns the maximum pixel clock frequency (in MHz).
pMaxFrameRate	Returns the maximum frame rate (in fps).



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
IS_AUTO_EXPOSURE_RUNNING	Automatic exposure is active.
IS_INVALID_PARAMETER	The <code>Timeout</code> input parameter is not correct.
IS_TRIGGER_ACTIVATED	The camera is operating in trigger mode.

Related Functions

- [is_SetPixelClock\(\)](#)
- [is_SetFrameRate\(\)](#)
- [is_SetAutoParameter\(\)](#)
- [is_CaptureVideo\(\)](#)

5.3.114 `is_SetPacketFilter`

	
GigE	GigE

Syntax

```
INT is_SetPacketFilter (INT iAdapterID, UINT uFilterSetting)
```

Description

Using `is_SetPacketFilter()`, you can set the packet filter for a network adapter.



The `is_SetPacketFilter()` function is only supported by the cameras of the *GigE uEye* series.



Only incoming packets are filtered.

Regardless of this setting, ICMP (Ping) and ARP packets are always forwarded to the operating system.

Input Parameters

<code>iAdapterID</code>	Internal adapter ID of the network adapter. It is returned by the <code>is_GetEthDeviceInfo()</code> function in the <code>UEYE_ETH_ADAPTER_INFO</code> structure.
<code>uFilterSetting</code>	
<code>IS_ETH_PCKTFLT_PASSALL</code>	Forward all packets to the operating system.
<code>IS_ETH_PCKTFLT_BLOCKUEGET</code>	Block <i>GigE uEye</i> data packets directed to the operating system (recommended).
<code>IS_ETH_PCKTFLT_BLOCKALL</code>	Block all packets directed to the operating system.



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_INVALID_PARAMETER</code>	<code>uFilterSetting</code> is invalid.
<code>IS_CANT_OPEN_DEVICE</code>	Driver could not be found.
<code>IS_IO_REQUEST_FAILED</code>	Driver communication failed.

Related Functions

- `is_GetEthDeviceInfo()`
- `is_SetAutoCfgIpSetup()`

5.3.115 is_SetPersistentIpCfg

	
GigE	GigE

Syntax

```
INT is_SetPersistentIpCfg (HIDS hDev,
                          UEYE_ETH_IP_CONFIGURATION* pIpCfg,
                          UINT uStructSize)
```

Description

Using `is_SetPersistentIpCfg()`, you can set the properties of the persistent IP configuration for a connected camera. The IP configuration can also be changed using the *uEye Camera Manager*.



The `is_SetPersistentIpCfg()` function is only supported by cameras of the *GigE uEye* series.

Input Parameters

hDev	DevID IS_USE_DEVICE_ID DevID = internal device ID of the camera from the UEYE_CAMERA_INFO structure (see also <code>is_GetCameraList()</code>).
pIpCfg	Pointer to a UEYE_ETH_IP_CONFIGURATION object (see below).
uStructSize	Size of the UEYE_ETH_IP_CONFIGURATION structure (in bytes).



The `is_SetPersistentIpCfg()` function does not accept a camera handle in the `hDev` parameter. In the call, please use the internal device ID as described below.
Never modify the IP configuration after a *GigE uEye camera* has been initialised!

Contents of the UEYE_ETH_IP_CONFIGURATION Structure

UEYE_ETH_ADDR_IPV4	ipAddress	IPv4 address
UEYE_ETH_ADDR_IPV4	ipSubnetmask	IPv4 subnet mask
BYTE	reserved[4]	reserved

Return Values

IS_SUCCESS	Function executed successfully
IS_INVALID_PARAMETER	pIpCfg is invalid.
IS_BAD_STRUCTURE_SIZE	The structure size you specified is invalid.
IS_NOT_SUPPORTED	For hDev, no device ID was specified or the ID is no device ID for an Ethernet camera (see above).
IS_CANT_OPEN_DEVICE	Driver could not be found.
IS_IO_REQUEST_FAILED	Driver communication failed.

Related Functions

- `is_SetAutoCfgIpSetup()`
- `is_GetEthDeviceInfo()`

Code Sample



```
//Create the structure
UEYE_ETH_IP_CONFIGURATION ipcfg;

//Create specific camera handle from the internal device ID, see info
in the box above
HIDS hDev = (HIDS)( dwDeviceID | IS_USE_DEVICE_ID);

//Indicate addresses in hexadecimal format
ipcfg.ipAddress.dwAddr = 0xC0A80A02; //IP address 192.168.10.2
ipcfg.ipSubnetmask.dwAddr = 0xFFFFFFFF00; //Subnet mask 255.255.255.0

//Set persistent IP address
INT nRet = is_SetPersistentIpCfg( hDev, &ipcfg, sizeof
(UEYE_ETH_IP_CONFIGURATION));
```

5.3.116 is_SetPixelClock

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetPixelClock (HIDS hCam, INT Clock)
```

Description

`is_SetPixelClock()` sets the frequency used to read out image data from the sensor (pixel clock frequency). Due to an excessive pixel clock for USB cameras, images may get lost during the transfer. If you change the pixel clock on-the-fly, the current image capturing process will be aborted.



Some sensors allow a higher pixel clock setting if binning or subsampling has been activated. If you set a higher pixel clock and then reduce the binning/subsampling factors again, the driver will automatically select the highest possible pixel clock for the new settings.



Changes to the image geometry or pixel clock affect the value ranges of the frame rate and exposure time. After executing `is_SetPixelClock()`, calling the following functions is recommended in order to keep the defined camera settings:

- `is_SetFrameRate()`
- `is_SetExposureTime()`
- If you are using the *uEye's* flash function: `is_SetFlashStrobe()`

Input Parameters

hCam	Camera handle
Clock	Pixel clock frequency to be set (in MHz)
IS_GET_PIXEL_CLOCK	Current pixel clock
IS_GET_DEFAULT_PIXEL_CLK	Returns the default pixel clock frequency

Return Values



IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
Current setting when used together with IS_GET_PIXEL_CLOCK	
IS_INVALID_MODE	Camera is in standby mode, function not allowed.
IS_INVALID_PARAMETER	The value for <code>Clock</code> is outside the pixel clock range supported by the camera.

Related Functions

- `is_SetOptimalCameraTiming()`
- `is_SetFrameRate()`
- `is_SetExposureTime()`
- `is_SetAutoParameter()`

- is_SetBinning()
- is_SetSubSampling()
- is_SetAOI()

5.3.117 is_SetRopEffect

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetRopEffect (HIDS hCam, INT effect, INT param, INT reserved)
```

Description

`is_SetRopEffect()` enables functions for real-time image geometry modification (Rop = *raster operation*).

Input Parameters

hCam	Camera handle
effect	
IS_SET_ROP_MIRROR_UPDOWN	Mirrors the image along the horizontal axis.
IS_SET_ROP_MIRROR_LEFTRIGHT	Mirrors the image along the vertical axis. Depending on the sensor, this operation is performed in the camera or in the PC software.
IS_GET_ROP_EFFECT	Returns the current settings.
param	Turns the Rop effect on / off. 0 = Turn off 1 = Turn on
reserved	Reserved. 0 must be passed.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
Current setting when used together with IS_SET_ROP_EFFECT	
IS_INVALID_MODE	Camera is in standby mode, function not allowed.

Related Functions

- [`is_SetBinning\(\)`](#)
- [`is_SetSubSampling\(\)`](#)
- [`is_SetAOI\(\)`](#)
- [`is_SetImagePos\(\)`](#)

5.3.118 `is_SetSaturation`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetSaturation (HIDS hCam, INT ChromU, INT ChromV)
```

Description

Using `is_SetSaturation()`, you can set the software colour saturation.



In the YUV format, colour information (i.e. the colour difference signals) is provided by the U and V channels. In the U channel, this information results from the difference between the blue level and Y (luminance), in the V channel from the difference between the red level and Y. For use in other colour formats than YUV, U and V are converted using a driver matrix.

Input Parameters

<code>hCam</code>	Camera handle
<code>ChromU</code>	U saturation: value multiplied by 100. Range: [IS_MIN_SATURATION ... IS_MAX_SATURATION]
<code>IS_GET_SATURATION_U</code>	Returns the current value for the U saturation.
<code>ChromV</code>	V saturation: value multiplied by 100. Range: [IS_MIN_SATURATION ... IS_MAX_SATURATION]
<code>IS_GET_SATURATION_V</code>	Returns the current value for the V saturation.



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message
Current setting when used together with <code>IS_GET_SATURATION_U</code> <code>IS_GET_SATURATION_V</code>	
<code>IS_INVALID_PARAMETER</code>	Invalid value for the <code>ChromU</code> or <code>ChromV</code> parameter.

Related Functions

- [`is_SetColorMode\(\)`](#)
- [`is_SetColorCorrection\(\)`](#)
- [`is_SetHardwareGamma\(\)`](#)
- [`is_SetColorConverter\(\)`](#)

5.3.119 is_SetSensorScaler

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetSensorScaler (HIDS hCam, UINT nMode, double dblFactor)
```

Description

`is_SetSensorScaler()` enables internal image scaling for some sensors. This allows to reduce the image resolution by adjustable factors. Thus, the amount of data from high resolution sensors can be reduced.



Internal image scaling is only supported by UI-149x/549x series sensors.

Input Parameters

hCam	Camera handle
nMode	Function mode
IS_ENABLE_SENSOR_SCALER	Enable image scaling
IS_ENABLE_SENSOR_SCALER IS_ENABLE_ANTI_ALIASING	Enable image scaling with smoothed edges (anti aliasing effect)
dblFactor	Scaling factor

Return Values

IS_SUCCESS	Function executed successfully
IS_INVALID_PARAMETER	General error message
IS_NOT_SUPPORTED	The test image function is not supported by the camera.
IS_NOT_SUPPORTED	The sensor does not support image scaling

Related Functions

- [is_GetSensorScalerInfo\(\)](#)



Code Sample

```
SENSORSCALERINFO Info;
INT nRet;
double dblNewFactor;

// Query information on image scaling
nRet = is_GetSensorScalerInfo (hCam, &Info,
                               sizeof(Info));

// Enable scaling with anti aliasing
dblNewFactor = Info.dblMinFactor + Info.dblFactorIncrement;
nRet = is_SetSensorScaler (hCam, IS_ENABLE_SENSOR_SCALER |
                           IS_ENABLE_ANTI_ALIASING, dblNewFactor);
```

5.3.120 `is_SetSensorTestImage`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetSensorTestImage (HIDS hCam, INT TestImage, INT Param)
```

Description

`is_SetSensorTestImage()` enables a test image function in the sensor. You can select different test images. The test images supported by a particular camera can be queried using the [is_GetSupportedTestImages\(\)](#) function. For some test images, the `Param` parameter provides additional options. If the test image does not support additional parameters, `Param` will be ignored.

Input Parameters

<code>hCam</code>	Camera handle
<code>TestImage</code>	The test image to be set. See also is_GetSupportedTestImages() .
<code>Param</code>	Additional parameter for used to modify the test image. Not available for all test images.



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message
<code>IS_INVALID_PARAMETER</code>	The <code>Param</code> parameter is not within the allowed value range.
<code>IS_NOT_SUPPORTED</code>	The test image function is not supported by the camera.

Related Functions

- [is_GetSupportedTestImages\(\)](#)
- [is_GetTestImageValueRange\(\)](#)

5.3.121 is_SetStarterFirmware

	
GigE	GigE

Syntax

```
INT is_SetStarterFirmware (HIDS hDev,
                           const CHAR* pcFilepath,
                           UINT uFilepathLen)
```

Description

Using `is_SetStarterFirmware()`, you can update the starter firmware of a connected camera. This is also possible from within the *uEye Camera Manager*.



The `is_SetStarterFirmware()` function is only supported by the cameras of the *GigE uEye* series.



The starter firmware determines the start-up behaviour of the *GigE uEye*. We recommend that you do not update the starter firmware unless an older firmware version causes start-up problems. If you have questions on the current starter firmware, please contact [Technical Support](#).

Input Parameters

hDev	DevID IS_USE_DEVICE_ID, DevID = internal device ID of the camera from the UYEYE_CAMERA_INFO structure (see also <code>is_GetCameraList()</code>)
pcFilepath	Pointer to a null-terminated ASCII string that contains the full file path.
uFilepathLen	Length of the file path (in bytes).



The `is_SetPersistentIpCfg()` function does not accept a camera handle in the `hDev` parameter. In the call, please use the internal device ID as described below.
Never modify the IP configuration after a *GigE uEye* camera has been initialised!

Return Values

IS_SUCCESS	Function executed successfully
IS_INVALID_PARAMETER	The <code>pcFilepath</code> or <code>uFilepathLen</code> parameter is invalid.
IS_BAD_STRUCTURE_SIZE	The structure size you specified is invalid.
IS_NOT_SUPPORTED	For <code>hDev</code> , no device ID was specified or the ID is no device ID for an Ethernet camera.
IS_CANT_OPEN_DEVICE	Driver could not be found.
IS_IO_REQUEST_FAILED	Driver communication failed.

Related Functions



- `is_GetCameraList()`
- `is_GetDuration()`
- `is_InitCamera()`

Code Sample

```
// Prepare the data parameter.
const CHAR kFilepath[] = "c:\\ids\\firmware.fw";

// Prepare the handle parameter. Mark the given device id with
IS_USE_DEVICE_ID.
HIDS hDev = (HIDS)( dwDeviceID | IS_USE_DEVICE_ID);
INT nRet = is_SetStarterFirmware( hDev, kFilepath, sizeof(kFilepath));
```

5.3.122 is_SetSubSampling

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetSubSampling (HIDS hCam, INT mode)
```

Description

Using `is_SetSubSampling()`, you can enable sub-sampling mode both in horizontal and in vertical directions. This allows you to reduce the image size in the sub-sampling direction without scaling down the area of interest. In order to simultaneously enable horizontal and vertical sub-sampling, the horizontal and vertical sub-sampling parameters can be linked by a logical OR.

Some monochrome sensors are limited by their design to mere colour sub-sampling. In case of fine image structures, this can result in slight artifacts.

The adjustable sub-sampling factors of each sensor are listed in [Specifications: Sensors](#) chapter.



Some sensors allow a higher pixel clock setting if binning or subsampling has been activated. If you set a higher pixel clock and then reduce the binning/subsampling factors again, the driver will automatically select the highest possible pixel clock for the new settings.



Changes to the image geometry or pixel clock affect the value ranges of the frame rate and exposure time. After executing `is_SetBinning()`, calling the following functions is recommended in order to keep the defined camera settings:

- `is_SetFrameRate()`
- `is_SetExposureTime()`
- If you are using the uEye's flash function: `is_SetFlashStrobe()`

Input Parameters

hCam	Camera handle
mode	
IS_SUBSAMPLING_DISABLE	Disables sub-sampling.
IS_SUBSAMPLING_2X_VERTICAL	Enables vertical sub-sampling with factor 2.
IS_SUBSAMPLING_3X_VERTICAL	Enables vertical sub-sampling with factor 3.
IS_SUBSAMPLING_4X_VERTICAL	Enables vertical sub-sampling with factor 4.
IS_SUBSAMPLING_5X_VERTICAL	Enables vertical sub-sampling with factor 5.
IS_SUBSAMPLING_6X_VERTICAL	Enables vertical sub-sampling with factor 6.
IS_SUBSAMPLING_8X_VERTICAL	Enables vertical sub-sampling with factor 8.
IS_SUBSAMPLING_16X_VERTICAL	Enables vertical sub-sampling with factor 16.
IS_SUBSAMPLING_2X_HORIZONTAL	Enables horizontal sub-sampling with factor 2.
IS_SUBSAMPLING_3X_HORIZONTAL	Enables horizontal sub-sampling with factor 3.
IS_SUBSAMPLING_4X_HORIZONTAL	Enables horizontal sub-sampling with factor 4.
IS_SUBSAMPLING_5X_HORIZONTAL	Enables horizontal sub-sampling with factor 5.

IS_SUBSAMPLING_6X_HORIZONTAL	Enables horizontal sub-sampling with factor 6.
IS_SUBSAMPLING_8X_HORIZONTAL	Enables horizontal sub-sampling with factor 8.
IS_SUBSAMPLING_16X_HORIZONTAL	Enables horizontal sub-sampling with factor 16.
IS_GET_SUBSAMPLING	Returns the current setting.
IS_GET_SUBSAMPLING_FACTOR_VERTICAL	Returns the vertical sub-sampling factor
IS_GET_SUBSAMPLING_FACTOR_HORIZONTAL	Returns the horizontal sub-sampling factor
IS_GET_SUBSAMPLING_TYPE	Indicates whether the camera uses colour-proof sub-sampling.
IS_GET_SUPPORTED_SUBSAMPLING	Returns the supported sub-sampling modes.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
Current setting when used together with IS_GET_SUBSAMPLING	Returns IS_SUBSAMPLING_COLOR if the camera uses colour-proof sub-sampling, else IS_SUBSAMPLING_MONO
When used with IS_GET_SUPPORTED_SUBSAMPLING	Returns the supported sub-sampling modes linked by logical ORs

Related Functions

- is_SetBinning()
- is_SetAOI()
- is_SetImagePos()
- is_SetPixelClock()

5.3.123 is_SetTimeout

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetTimeout (HIDS hCam, UINT nMode, UINT Timeout)
```

Description

Using `is_SetTimeout()`, you can change user-defined timeout values of the *uEye* API. If no user-defined timeout is set, the default value of the *uEye* API is used for the relevant timeout.

For further information, please refer to the [How To Proceed: Timeout Values for Image Capture](#) section.



The user-defined timeout only applies to the specified camera at runtime of the program.

Input Parameters

hCam	Camera handle
nMode	Selects the timeout value to be set
IS_TRIGGER_TIMEOUT	Sets the timeout value for triggered image capture
Timeout	Timeout value in 10 ms. Value range [0; 4...429496729] (corresponds to 40 ms to approx. 1193 hours) 0 = use default value of the <i>uEye</i> API For 1...3, the value 4 is used.

Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
IS_NOT_SUPPORTED	The value for nMode is invalid



Related Functions

- [is_GetTimeout\(\)](#)
- [is_CaptureVideo\(\)](#)
- [is_FreezeVideo\(\)](#)
- [is_SetExternalTrigger\(\)](#)

Code Sample

```
// Set user-defined timeout to 120 seconds
is_SetTimeout(hCam, IS_TRIGGER_TIMEOUT, 12000);
```


5.3.124 `is_SetTriggerCounter`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetTriggerCounter (HIDS hCam, INT nValue)
```

Description

Using `is_SetTriggerCounter()`, you can read out the camera's internal counter for incoming hardware triggers.

Input Parameters

<code>hCam</code>	Camera handle
<code>nValue</code>	
<code>IS_GET_TRIGGER_COUNTER</code>	Returns the current hardware trigger count
Other values	Resets the hardware trigger counter



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message

Related Functions

- [`is_SetExternalTrigger\(\)`](#)
- [`is_CameraStatus\(\)`](#)

5.3.125 is_SetTriggerDelay

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetTriggerDelay (HIDS hCam, INT nTriggerDelay)
```

Description

Using `is_SetTriggerDelay()`, you can set the delay time between the arrival of a trigger signal and the start of exposure. The trigger signal can be initiated by hardware or by software.

The delay time set here adds to the delay caused by the sensor. The delay times of each sensor are listed in [Specifications: Sensors](#) chapter.

Input Parameters

<code>hCam</code>	Camera handle
<code>nTriggerDelay</code>	Time by which the image capture is delayed (in μ s).
<code>IS_GET_TRIGGER_DELAY</code>	Returns the currently set delay time.
<code>IS_GET_MIN_TRIGGER_DELAY</code>	Returns the minimum adjustable value.
<code>IS_GET_MAX_TRIGGER_DELAY</code>	Returns the maximum adjustable value.
<code>IS_GET_TRIGGER_DELAY_GRANULARITY</code>	Returns the resolution of the adjustable delay time.



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message
Current setting when used together with <code>IS_GET_TRIGGER_DELAY</code>	

Related Functions

- [`is_SetFlashDelay\(\)`](#)
- [`is_SetFlashStrobe\(\)`](#)
- [`is_GetGlobalFlashDelays\(\)`](#)
- [`is_SetExternalTrigger\(\)`](#)

5.3.126 is_StopLiveVideo

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_StopLiveVideo (HIDS hCam, INT Wait)
```

Description

is_StopLiveVideo() stops live mode or cancels a hardware triggered image capture in case the exposure has not yet started.

Input Parameters

hCam	Camera handle
Wait	
IS_WAIT	The function waits until the image save is complete.
IS_DONT_WAIT	The function returns immediately. Digitising the image is completed in the background.
IS_FORCE_VIDEO_STOP	Digitising is stopped immediately.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [is_FreezeVideo\(\)](#)
- [is_CaptureVideo\(\)](#)
- [is_SetDisplayMode\(\)](#)

5.3.127 is_UnlockSeqBuf

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_UnlockSeqBuf (HIDS hCam, INT nNum, char* pcMem)
```

Description

Using `is_UnlockSeqBuf()`, you unlock a previously locked image memory in order to make it available again for storing captured images. The image memory is re-inserted at its previous position in the sequence list.

Input Parameters

hCam	Camera handle
nNum	Number of the image memory to unlock. When you pass <code>IS_IGNORE_PARAMETER</code> , the image memory is only identified by its starting address. <code>nNum</code> identifies the position in the sequence list, not the memory ID assigned with <code>is_AllocImageMem()</code> .
pcMem	Starting address of the image memory



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- `is_LockSeqBuf()`

5.3.128 is_WaitEvent

	
-	USB 2.0 GigE

Syntax

```
INT is_WaitEvent (HIDS hCam, INT which, INT nTimeout)
```

Description

`is_WaitEvent()` allows waiting for *uEye* events. The function indicates successful execution when the event has occurred within the specified timeout.

Input Parameters

<code>hCam</code>	Camera handle
<code>which</code>	ID of the event (see <code>is_InitEvent()</code>)
<code>nTimeout</code>	Time (in ms) that the function will wait for an event to occur

Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message
<code>IS_TIMED_OUT</code>	The specified timeout expired without the event having occurred.

Related Functions



- `is_InitEvent()`
- `is_EnableEvent()`
- `is_DisableEvent()`
- `is_ExitEvent()`

Code Sample

```
//Activate and initialise FRAME event
is_EnableEvent (hCam, IS_SET_EVENT_FRAME);
is_InitEvent (hCam, IS_SET_EVENT_FRAME);

//Start image capture and wait 1000 ms for event to occur
is_FreezeVideo (hCam, IS_DONT_WAIT);
is_WaitEvent (hCam, IS_SET_EVENT_FRAME, 1000);
```

5.3.129 is_WriteEEPROM

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_WriteEEPROM (HIDS hCam, INT Adr, char* pcString, INT Count)
```

Description

Using `is_WriteEEPROM()`, you can write data to the EEPROM of the camera. Besides the hard-coded factory information, the EEPROM of the *uEye* can hold 64 bytes of user data.

Input Parameters

hCam	Camera handle
Adr	Starting address for data writes (0...63)
pcString	Pointer to the string containing the data to be written
Count	Number of characters to be written (1...64)



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
IS_INVALID_MODE	Camera is in standby mode, function not allowed.

Related Functions

- [`is_ReadEEPROM\(\)`](#)

5.3.130 is_WriteI2C

	
USB 2.0	USB 2.0

Syntax

```
INT is_WriteI2C (HIDS hCam,
                INT nDeviceAddr, INT nRegisterAddr,
                BYTE* pbData, INT nLen)
```

Description

Using `is_WriteI2C()`, you can write data via the I²C bus of a board level camera.

For information on the signals applied to the I²C bus, refer to the chapters with electrical specifications for the *USB uEye LE* and the *USB uEye ME*.



The `is_WriteI2C()` function is only supported by PCB versions of the *USB uEye ME/LE* camera series.



The uEye processes I²C addresses in a 7-bit format that is created from the 8-bit format by a bit shift to the right. The eighth bit indicates whether an address is a read (1) or write (0) address. For example, the 7-bit address 0x48 is the write address 0x90 and the read address 0x91 in 8-bit format.

The following addresses for `nRegisterAddr` are assigned to the uEye and must not be used:

7-bit format: 0x10, 0x48, 0x4C, 0x50, 0x51, 0x52, 0x55, 0x5C, 0x5D, 0x69

8-bit format: 0x20, 0x90, 0x98, 0xA0, 0xA2, 0xA4, 0xAA, 0xB8, 0xBA, 0xD2

Input Parameters

<code>hCam</code>	Camera handle
<code>nDeviceAddr</code>	Device address
<code>nRegisterAddr</code>	Address of a 8 bit register (only 8-bit addresses are valid)
<code>nRegisterAddr IS_I2C_16_BIT_REGISTER</code>	Address of a 16 bit register
<code>pbData</code>	Data to be written
<code>nLen</code>	Data length <code>nLen = 1</code> : 8 bits data <code>nLen = 2</code> : 16 bits data

Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message

Related Functions

- `is_ReadI2C()`

Example

```
// -----
// Write to I2C device:
//
// 16 bit register addressing
// Note:
// Only writing 2 bytes at once is allowed

is_WriteI2C (hCam, DevAdr, RegAdr | IS_I2C_16_BIT_REGISTER, pbData, 2);

// 16 bit register addressing
// Note:
// Writing 1 or 2 bytes at once is allowed

// Writing 1 byte
is_WriteI2C (hCam, DevAdr, RegAdr, pbData, 1);
// Writing 2 bytes
is_WriteI2C (hCam, DevAdr, RegAdr, pbData, 2);

// -----
// Read from I2C device:
//
// 16 bit register addressing
// Note:
// Only reading 2 bytes at once is allowed

is_ReadI2C (hCam, DevAdr, RegAdr | IS_I2C_16_BIT_REGISTER, pbData, 2);

// 8 bit register addressing
// Note:
// Reading 1 or 2 bytes at once is allowed

// Reading 1 byte
is_ReadI2C (hCam, DevAdr, RegAdr, pbData, 1);
// Reading 2 bytes
is_ReadI2C (hCam, DevAdr, RegAdr, pbData, 2);

// -----
// Example values
// Device address (from device data sheet): 0x40
// Device address in 8 bit format (after left shift): 0x80
// Device address for write (generated by is_WriteI2C()): 0x80
// Device address for read (generated by is_ReadI2C()) 0x81

// Write value 0x52 to device with address 0x40 in register 0x00
is_WriteI2C (hCam, 0x80, 0x00, 0x52, 1);



// Read from device with address 0x40, register 0x00
INT nVal;
is_ReadI2C (hCam, 0x80, 0x00, &nVal, 1);
```


5.4 AVI Function Descriptions

The functions of the *uEye_tools.dll* enable you to save images captured with the *uEye* as sequences to an AVI file.

The [How To Proceed: Capturing AVIs](#) chapter shows the command sequence for capturing an AVI video.

5.4.1 isavi_AddFrame

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT isavi_AddFrame (int nAviID, char* pcImageMem)
```

Description

isavi_AddFrame() adds a new frame to an AVI sequence.

Input Parameters

nAviID	Instance ID set by the isavi_InitAVI() function
pcImageMem	Pointer to the memory containing the image.



Return Values

IS_AVI_NO_ERR	Function executed successfully
IS_AVI_ERR_INVALID_ID	The indicated AVI instance could not be found. Either the AVI ID is invalid or the instance has already been deleted using isavi_ExitAVI() .
IS_AVI_ERR_COMPRESSION_RUN	The current image could not be processed since compression is still in progress.
IS_AVI_ERR_INVALID_FILE	The AVI file is not open.

Related Functions

- [isavi_InitAVI\(\)](#)

5.4.2 `isavi_CloseAVI`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT isavi_CloseAVI (int nAviID)
```

Description

`isavi_CloseAVI()` closes an AVI file which was opened using `isavi_OpenAVI()`.

Input Parameters

<code>nAviID</code>	Instance ID set by the <code>isavi_InitAVI()</code> function.
---------------------	---



Return Values

<code>IS_AVI_NO_ERR</code>	Function executed successfully
<code>IS_AVI_ERR_INVALID_ID</code>	The indicated AVI instance could not be found. Either the AVI ID is invalid or the instance has already been deleted using <code>isavi_ExitAVI()</code> .

Related Functions

- `isavi_OpenAVI()`
- `isavi_InitAVI()`
- `isavi_ExitAVI()`

5.4.3 isavi_DisableEvent

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT isavi_DisableEvent (int nAviID, int which)
```

Description

`isavi_DisableEvent()` disables the specified event. The disabled event no longer triggers an event notification when the associated event occurs.

Input Parameters

nAviID	Instance ID set by the <code>isavi_InitAVI()</code> function.
which	Name of the event to be disabled.
IS_AVI_SET_EVENT_FRAME_SAVED	A new frame was saved to the AVI file.



Return Values

IS_AVI_NO_ERR	Function executed successfully
IS_AVI_ERR_INVALID_ID	The indicated AVI instance could not be found. Either the AVI ID is invalid or the instance has already been deleted using <code>isavi_ExitAVI()</code> .
IS_AVI_ERR_PARAMETER	An invalid event was specified for the <code>which</code> parameter.

Related Functions

- `isavi_EnableEvent()`

5.4.4 `isavi_EnableEvent`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT isavi_EnableEvent (int nAviID, int which)
```

Description

`isavi_DisableEvent()` enables the specified event. The enabled event triggers an event notification when the associated event occurs.

Input Parameters

<code>nAviID</code>	Instance ID set by the <code>isavi_InitAVI()</code> function.
<code>which</code>	Name of the event to be enabled.
<code>IS_AVI_SET_EVENT_FRAME_SAVED</code>	A new frame was saved to the AVI file.



Return Values

<code>IS_AVI_NO_ERR</code>	Function executed successfully
<code>IS_AVI_ERR_INVALID_ID</code>	The indicated AVI instance could not be found. Either the AVI ID is invalid or the instance has already been deleted using <code>isavi_ExitAVI()</code> .
<code>IS_AVI_ERR_PARAMETER</code>	An invalid event was specified for the <code>which</code> parameter.

Related Functions

- `isavi_DisableEvent()`

5.4.5 isavi_ExitAVI

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT isavi_ExitAVI (int nAviID)
```

Description

`isavi_ExitAVI()` terminates and deletes the instance of the *uEye* AVI interface.

Input Parameters

nAviID	Instance ID set by the <code>isavi_InitAVI()</code> function.
--------	---



Return Values

IS_AVI_NO_ERR	Function executed successfully
IS_AVI_ERR_INVALID_ID	The indicated AVI instance could not be found. Either the AVI ID is invalid or the instance has already been deleted using <code>isavi_ExitAVI()</code> .
IS_AVI_ERR_INVALID_FILE	The AVI file could not be closed.

Related Functions

- `isavi_InitAVI()`
- `isavi_OpenAVI()`
- `isavi_CloseAVI()`

5.4.6 `isavi_ExitEvent`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT isavi_ExitEvent (int nAviID, int which)
```

Description

`isavi_ExitEvent()` deletes the specified event. The deleted event no longer occurs and needs to be re-created using `isavi_InitEvent()` before it can be enabled and used.

Input Parameters

<code>nAviID</code>	Instance ID set by the <code>isavi_InitAVI()</code> function.
<code>which</code>	Name of the event to be deleted.
<code>IS_AVI_SET_EVENT_FRAME_SAVED</code>	A new frame was saved to the AVI file.



Return Values

<code>IS_AVI_NO_ERR</code>	Function executed successfully
<code>IS_AVI_ERR_INVALID_ID</code>	The indicated AVI instance could not be found. Either the AVI ID is invalid or the instance has already been deleted using <code>isavi_ExitAVI()</code> .
<code>IS_AVI_ERR_PARAMETER</code>	An invalid event was specified for the <code>which</code> parameter.

Related Functions

- `isavi_InitEvent()`
- `isavi_EnableEvent()`
- `isavi_DisableEvent()`

5.4.7 isavi_GetAVIFilename

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT isavi_GetAVIFilename (int nAviID, char* strName)
```

Description

Using `isavi_GetAVIFilename()`, you can read out the filename of the current AVI file. This function is helpful if an AVI file was opened with the `isavi_OpenAVI()` function and a `Null` parameter was specified.

Input Parameters

nAviID	Instance ID set by the <code>isavi_InitAVI()</code> function.
strName	Pointer to the memory location where the filename is written to. The allocated memory must be large enough to accommodate the full file path.
NULL	When NULL is passed the function returns the length of the filename.



Return Values

IS_AVI_NO_ERR	Function executed successfully
IS_AVI_ERR_INVALID_ID	The indicated AVI instance could not be found. Either the AVI ID is invalid or the instance has already been deleted using <code>isavi_ExitAVI()</code> .

Related Functions

- `isavi_GetAVISize()`

5.4.8 `isavi_GetAVISize`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT isavi_GetAVISize (int nAviID, float* size)
```

Description

Use `isavi_GetAVISize()` to retrieve the size of the frame sequence saved to the current AVI file.

Input Parameters

<code>nAviID</code>	Instance ID set by the <code>isavi_InitAVI()</code> function
<code>size</code>	The size in kBytes



Return Values

<code>IS_AVI_NO_ERR</code>	Function executed successfully
<code>IS_AVI_ERR_INVALID_ID</code>	The designated AVI instance could not be found. Either the AVI ID is invalid or the instance was already deleted using <code>isavi_ExitAVI()</code> .

Related Functions

- `isavi_GetAVIFileName()`

5.4.9 isavi_GetnCompressedFrames

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT isavi_GetnCompressedFrames (int nAviID, unsigned long* nFrames)
```

Description

Using `isavi_GetnCompressedFrames()`, you can read out the number of frames saved to the current AVI file.

Input Parameters

nAviID	Instance ID set by the <code>isavi_InitAVI()</code> function.
nFrames	The number of frames



Return Values

IS_AVI_NO_ERR	Function executed successfully
IS_AVI_ERR_INVALID_ID	The indicated AVI instance could not be found. Either the AVI ID is invalid or the instance has already been deleted using <code>isavi_ExitAVI()</code> .

Related Functions

- `isavi_GetnLostFrames()`
- `isavi_ResetFrameCounters()`

5.4.10 `isavi_GetnLostFrames`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT isavi_GetnLostFrames (int nAviID, unsigned long* nFrames)
```

Description

Using `isavi_GetnLostFrames()`, you can read out the number of frames that have been discarded. A frame will be discarded if it cannot be processed because a compression operation is still in progress.

Input Parameters

<code>nAviID</code>	Instance ID set by the <code>isavi_InitAVI()</code> function.
<code>nFrames</code>	The number of frames



Return Values

<code>IS_AVI_NO_ERR</code>	Function executed successfully
<code>IS_AVI_ERR_INVALID_ID</code>	The indicated AVI instance could not be found. Either the AVI ID is invalid or the instance has already been deleted using <code>isavi_ExitAVI()</code> .

Related Functions

- `isavi_GetnCompressedFrames()`
- `isavi_ResetFrameCounters()`

5.4.11 isavi_InitAVI

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT isavi_InitAVI (int* pnAviID, HIDS hCam)
```

Description

`isavi_InitAVI()` initialises an instance of the *uEye* AVI interface. Multiple instances can be created simultaneously.

Input Parameters

pnAviID	Pointer. Returns the instance ID which is needed for calling the other <i>uEye</i> AVI functions.
hCam	Handle of a selected or initialised <i>uEye</i> camera.



Return Values

IS_AVI_NO_ERR	Function executed successfully
IS_AVI_ERR_PARAMETER	The pnAviID pointer contains the value Null.
IS_AVI_ERR_NO_CODEC_AVAIL	The maximum number of instances allowed in this system has been reached. It is not possible to create another instance.
IS_AVI_ERR_INVALID_UEYE	No <i>uEye</i> camera was found.

Related Functions

- [`isavi_ExitAVI\(\)`](#)
- [`isavi_OpenAVI\(\)`](#)
- [`isavi_CloseAVI\(\)`](#)

5.4.12 isavi_InitEvent

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT isavi_InitEvent (int nAviID, int which)
```

Description

`isavi_InitEvent()` creates the specified event. This includes registering the event object in the *uEye* AVI interface and creating an event handler. Before you can use a new event, you must enable it by calling `isavi_EnableEvent()`.

Input Parameters

nAviID	Instance ID set by the <code>isavi_InitAVI()</code> function.
which	Name of the event to be created.
IS_AVI_SET_EVENT_FRAME_SAVED	A new frame was saved to the AVI file.

Return Values

IS_AVI_NO_ERR	Function executed successfully
IS_AVI_ERR_INVALID_ID	The indicated AVI instance could not be found. Either the AVI ID is invalid or the instance has already been deleted using <code>isavi_ExitAVI()</code> .
IS_AVI_ERR_EVENT_FAILED	The event could not be initialised. The Windows <code>SetEvent</code> function failed.
IS_AVI_ERR_PARAMETER	An invalid event was specified for the <code>which</code> parameter.

Related Functions

- `isavi_ExitEvent()`
- `isavi_EnableEvent()`
- `isavi_DisableEvent()`



Code Sample

Create and enable an event object for the "Frame saved" event:

```
HANDLE hEvent = CreateEvent( NULL, TRUE, FALSE, "" );
if ( hEvent != NULL )
{
    isavi_InitEvent( AviDest, hEvent, IS_AVI_SET_EVENT_FRAME_SAVED );
    isavi_EnableEvent( AviDest, IS_AVI_SET_EVENT_FRAME_SAVED );

    if ( WaitForSingleObject( hEvent, 1000 ) == WAIT_OBJECT_0 )
    {
        //Frame was captured successfully...
    }
    isavi_DisableEvent( AviDest, IS_AVI_SET_EVENT_FRAME_SAVED );
    isavi_ExitEvent( AviDest, IS_AVI_SET_EVENT_FRAME_SAVED );
}
```

5.4.13 isavi_OpenAVI

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT isavi_OpenAVI (int nAviID)
```

Description

isavi_OpenAVI() opens a new or existing AVI file.

Input Parameters

nAviID	Instance ID set by the <u>isavi_InitAVI()</u> function.
pFileName	Pointer to the name to be assigned to the AVI file. If NULL is passed, the "Open File" dialogue is displayed.



Return Values

IS_AVI_NO_ERR	Function executed successfully
IS_AVI_ERR_INVALID_ID	The indicated AVI instance could not be found. Either the AVI ID is invalid or the instance has already been deleted using <u>isavi_ExitAVI()</u> .
IS_AVI_ERR_CAPTURE_RUNNING	Another capturing operation is in progress or an AVI file is still open.
IS_AVI_ERR_INVALID_FILE	No valid AVI file was selected in the Windows "Open File..." dialogue.
IS_AVI_ERR_NEW_FAILED	No memory could be allocated for the AVI file.
IS_AVI_ERR_AVIFILEOPEN	The AVI file could not be opened. Please check if the file is corrupted or was opened in another application.
IS_AVI_ERR_CREATESTREAM	No AVI stream could be created.

Related Functions

- isavi_CloseAVI()
- isavi_InitAVI()
- isavi_ExitAVI()

5.4.14 `isavi_ResetFrameCounters`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
IDSAPIEXP isavi_ResetFrameCounters (int nAviID)
```

Description

`isavi_ResetFrameCounters()` resets the counters for saved and discarded images.

Input Parameters

<code>nAviID</code>	Instance ID set by the <code>isavi_InitAVI()</code> function.
---------------------	---



Return Values

<code>IS_AVI_NO_ERR</code>	Function executed successfully
<code>IS_AVI_ERR_INVALID_ID</code>	The indicated AVI instance could not be found. Either the AVI ID is invalid or the instance has already been deleted using <code>isavi_ExitAVI()</code> .

Related Functions

- `isavi_GetnCompressedFrames()`
- `isavi_GetnLostFrames()`

5.4.15 isavi_SetFrameRate

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT isavi_SetFrameRate (int nAviID, double fr)
```

Description

`isavi_SetFrameRate()` sets the frame rate for AVI capturing. You can set the frame rate after opening the AVI file. This value does not have to be equal to the frame rate set for the *uEye* camera.

Input Parameters

nAviID	Instance ID set by the <code>isavi_InitAVI()</code> function.
fr	The frame rate to be set. Default = 25.0



Return Values

IS_AVI_NO_ERR	Function executed successfully
IS_AVI_ERR_INVALID_ID	The indicated AVI instance could not be found. Either the AVI ID is invalid or the instance has already been deleted using <code>isavi_ExitAVI()</code> .
IS_AVI_ERR_WRITE_INFO	The AVI file could not be modified.
IS_AVI_ERR_INVALID_FILE	The AVI file is not open.

Related Functions

- `isavi_SetImageQuality()`
- `isavi_SetImageSize()`

5.4.16 `isavi_SetImageQuality`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT isavi_SetImageQuality (int nAviID,int q)
```

Description

`isavi_SetImageQuality()` indicates the quality for the frames to be compressed. You can change the image quality at any time; it then applies to all subsequent frames. For compression, the system uses the JPEG algorithm.

Input Parameters

<code>nAviID</code>	Instance ID set by the <code>isavi_InitAVI()</code> function.
<code>q</code>	Image quality [1 = lowest ... 100 = highest]



Return Values

<code>IS_AVI_NO_ERR</code>	Function executed successfully
<code>IS_AVI_ERR_INVALID_ID</code>	The indicated AVI instance could not be found. Either the AVI ID is invalid or the instance has already been deleted using <code>isavi_ExitAVI()</code> .
<code>IS_AVI_ERR_INVALID_VALUE</code>	The <code>q</code> parameter is outside the range of 1...100.
<code>IS_AVI_ERR_INVALID_FILE</code>	The AVI file is not open.

Related Functions

- `isavi_SetFrameRate()`
- `isavi_SetImageSize()`

5.4.17 isavi_SetImageSize

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT isavi_SetImageSize (int nAviID,
                        int cMode,
                        long Width, long Height,
                        long PosX, long PosY,
                        long LineOffset)
```

Description

`isavi_SetImageSize()` sets the size and position of the area of interest which will be saved to the AVI file. Only the defined area of interest of each frame will be saved. In addition, this function specifies the input colour format of the frames. You define these settings only once for the entire video.



The supported input colour formats are RGB32, RGB24, Y8 and raw Bayer. The output file will always be in RGB24 format, regardless of the input data format.

For further information on the structure of the different colour formats, see the [Appendix: Colour and Memory Formats](#) section.

Input Parameters

nAviID	Instance ID set by the <code>isavi_InitAVI()</code> function.
cMode	Colour format of the input frames captured by the <i>uEye</i> .
Width	Width of the entire frame or of the area of interest.
Height	Height of the entire frame or of the area of interest.
PosX	X position (offset) of the area of interest.
PosY	Y position (offset) of the area of interest.
LineOffset	Line increment. The total widths of the areas clipped to the right and to the left of the area of interest make up this value.



Return Values

<code>IS_AVI_NO_ERR</code>	Function executed successfully
<code>IS_AVI_ERR_INVALID_ID</code>	The indicated AVI instance could not be found. Either the AVI ID is invalid or the instance has already been deleted using <code>isavi_ExitAVI()</code> .
<code>IS_AVI_ERR_INVALID_FILE</code>	The AVI file is not open.
<code>IS_AVI_ERR_CAPTURE_RUNNING</code>	Another capturing operation is in progress or an AVI file is still open.
<code>IS_AVI_ERR_ALLOC_MEMORY</code>	No memory could be allocated.
<code>IS_AVI_ERR_INVALID_CM</code>	The submitted colour mode is not supported for AVI capturing.
<code>IS_AVI_ERR_INVALID_SIZE</code>	The submitted size is invalid.
<code>IS_AVI_ERR_INVALID_POSITION</code>	The submitted position is invalid.

Related Functions

- `isavi_SetFrameRate()`
- `isavi_SetImageQuality()`

5.4.18 isavi_StartAVI

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT isavi_StartAVI (int nAviID)
```

Description

isavi_StartAVI() starts the image capture thread.

Input Parameters

nAviID	Instance ID set by the isavi_InitAVI() function.
--------	--



Return Values

IS_AVI_NO_ERR	Function executed successfully
IS_AVI_ERR_INVALID_ID	The indicated AVI instance could not be found. Either the AVI ID is invalid or the instance has already been deleted using isavi_ExitAVI() .
IS_AVI_ERR_INVALID_FILE	The AVI file is not open.

Related Functions

- [isavi_StopAVI\(\)](#)
- [isavi_InitEvent\(\)](#)
- [isavi_ExitAVI\(\)](#)

5.4.19 isavi_StopAVI

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT isavi_StopAVI (int nAviID)
```

Description

isavi_StopAVI() stops the image capture thread. Subsequent calls of [isavi_AddFrame\(\)](#) will be ignored.

Input Parameters

nAviID	Instance ID set by the isavi_InitAVI() function.
--------	--

Return Values

IS_AVI_NO_ERR	Function executed successfully
IS_AVI_ERR_INVALID_ID	The indicated AVI instance could not be found. Either the AVI ID is invalid or the instance has already been deleted using isavi_ExitAVI() .
IS_AVI_ERR_INVALID_FILE	The AVI file is not open.

Related Functions

- [isavi_StartAVI\(\)](#)
- [isavi_InitEvent\(\)](#)
- [isavi_ExitAVI\(\)](#)

5.5 Obsolete Functions

We are continuously extending and enhancing the *uEye API*. The resulting product upgrades sometimes require replacing obsolete functions with new ones. We recommend against using the obsolete functions. They will continue to be supported for reasons of backward compatibility, but they will not be documented any longer. The functions listed in this chapter will not be included in future versions of this manual.

The following table lists the obsolete functions and indicates the recommended alternatives.

Obsolete function	Recommended alternative	No longer documented since driver version
<u>is_DisableDDOverlay()</u>	<u>is_DirectRenderer()</u>	
<u>is_EnableDDOverlay()</u>	<u>is_DirectRenderer()</u>	
<u>is_GetDC()</u>	<u>is_DirectRenderer()</u>	
<u>is_GetDDOvlSurface()</u>	<u>is_DirectRenderer()</u>	
<u>is_GetLastMemorySequence()</u>	The <i>uEye</i> memory board is not supported any longer (see below).	3.30
<u>is_GetMemorySequenceWindow()</u>		
<u>is_GetNumberOfMemoryImages()</u>		
<u>is_GetRevisionInfo()</u>	<u>is_GetCameraInfo()</u>	3.20
<u>is_GetWhiteBalanceMultipliers()</u>	<u>is_SetAutoParameter()</u>	3.31
<u>is_HideDDOverlay()</u>	<u>is_DirectRenderer()</u>	
<u>is_IsMemoryBoardConnected()</u>	The <i>uEye</i> memory board is not supported any longer (see below).	3.30
<u>is_LockDDMem()</u>	<u>is_DirectRenderer()</u>	
<u>is_LockDDOverlayMem()</u>	<u>is_DirectRenderer()</u>	
<u>is_MemoryFreezeVideo()</u>	The <i>uEye</i> memory board is not supported any longer (see below).	3.30
<u>is_ReleaseDC()</u>	<u>is_DirectRenderer()</u>	
<u>is_ResetMemory()</u>	The <i>uEye</i> memory board is not supported any longer (see below).	3.30
<u>is_SetBayerConversion()</u>	<u>is_SetColorConverter()</u>	
<u>is_SetBrightness()</u>	<u>is_SetGamma()</u> <u>is_SetHardwareGamma()</u> <u>is_SetBICompensation()</u>	3.40
<u>is_SetContrast()</u>	<u>is_SetExposureTime()</u> <u>is_SetHardwareGain()</u>	3.40
<u>is_SetDDUpdateTime()</u>	<u>is_DirectRenderer()</u>	
<u>is_SetHwnd()</u>	<u>is_DirectRenderer()</u>	
<u>is_SetImageAOI()</u>	<u>is_SetAOI()</u>	
<u>is_SetImageSize()</u>	<u>is_SetAOI()</u>	
<u>is_SetImagePos()</u>	<u>is_SetAOI()</u>	
<u>is_SetKeyColor()</u>	<u>is_DirectRenderer()</u>	
<u>is_SetMemoryMode()</u>	The <i>uEye</i> memory board is not supported any longer (see below).	3.30

<code>is_SetWhiteBalance()</code>	<code>is_SetAutoParameter()</code>	3.31
<code>is_SetWhiteBalanceMultipliers()</code>	<code>is_SetAutoParameter()</code>	3.31
<code>is_ShowDDOverlay()</code>	<code>is_DirectRenderer()</code>	
<code>is_StealVideo()</code>	<code>is_DirectRenderer()</code>	
<code>is_TransferImage()</code>	The <i>uEye</i> memory board is not supported any longer (see below).	3.30
<code>is_TransferMemorySequence()</code>		
<code>is_UnlockDDMem()</code>	<code>is_DirectRenderer()</code>	
<code>is_UnlockDDOverlayMem()</code>	<code>is_DirectRenderer()</code>	
<code>is_UpdateDisplay()</code>	<code>is_DirectRenderer()</code>	



The `is_SetWhiteBalance()` and `is_SetWhiteBalanceMultipliers()` functions have been completely replaced by the `is_SetAutoParameter()` function and are no longer supported by the *uEye* API.





The optional memory board of the *USB uEye SE* and *USB uEye RE* camera series has been discontinued.

From version 3.30, the functions required to operate the memory board will no longer be supported in the *uEye* driver.

The *uEye* driver version 3.24 that still supports these functions will continue to be available in the download area of our website at <http://www.ids-imaging.com>.

5.5.1 is_DisableDDOverlay

	
USB 2.0 GigE	-

Syntax

```
INT is_DisableDDOverlay (HIDS hCam)
```

Description



This function is obsolete and should not be used anymore. We recommend to use the [is_DirectRenderer\(\)](#) function instead (see also [Obsolete Functions](#)).

In DirectDraw BackBuffer mode, `is_DisableDDOverlay()` disables overlay mode and releases the memory allocated to the overlay. This results in discarding of the overlay data.

Input Parameters

hCam	Camera handle
------	---------------



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [is_DisableDDOverlay\(\)](#)
- [is_EnableDDOverlay\(\)](#)
- [is_HideDDOverlay\(\)](#)
- [is_SetDisplayMode\(\)](#)
- [is_ShowDDOverlay\(\)](#)
- [is_GetDDOvlSurface\(\)](#)

5.5.2 `is_EnableDDOverlay`

	
USB 2.0 GigE	-

Syntax

```
INT is_EnableDDOverlay (HIDS hCam)
```

Description



This function is obsolete and should not be used anymore. We recommend to use the `is_DirectRenderer()` function instead (see also [Obsolete Functions](#)).

In DirectDraw BackBuffer mode, `is_EnableDDOverlay()` enables live overlay mode. The overlay cannot be displayed directly, but needs to be visualised before by calling `is_ShowDDOverlay()`. The overlay uses black as the so-called key colour, so that overlay graphics may not contain any black colour.

Input Parameters

<code>hCam</code>	Camera handle
-------------------	---------------



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message

Related Functions

- `is_SetDisplayMode()`
- `is_DisableDDOverlay()`
- `is_ShowDDOverlay()`
- `is_HideDDOverlay()`
- `is_GetDDOvlSurface()`

5.5.3 is_GetDC

	
USB 2.0 GigE	-

Syntax

```
INT is_GetDC (HIDS hCam, HDC* phDC)
```

Description



This function is obsolete and should not be used anymore. We recommend to use the [is_DirectRenderer\(\)](#) function instead (see also [Obsolete Functions](#)).

In DirectDraw BackBuffer mode, [is_GetDC\(\)](#) returns the device context handle of the overlay buffer. Using this handle, it is possible to access the overlay using the Windows GDI functionality. Thus, all Windows graphics commands such as [Line](#), [Circle](#), [Rectangle](#), [TextOut](#),... are available. You should release the device context handle as early as possible using the [is_ReleaseDC\(\)](#) function. While a [GetDC...ReleaseDC](#) block is executed, the overlay buffer on the screen will not be updated.

Input Parameters

hCam	Camera handle
phDC	Pointer to the variable that is supposed to contain the device context handle



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [is_ReleaseDC\(\)](#)
- [is_ShowDDOverlay\(\)](#)
- [is_DisableDDOverlay\(\)](#)
- [is_EnableDDOverlay\(\)](#)
- [is_GetDDOvlSurface\(\)](#)
- [is_SetDisplayMode\(\)](#)

5.5.4 `is_GetDDOvlSurface`

	
USB 2.0 GigE	-

Syntax

```
INT is_GetDDOvlSurface (HIDS hCam, void** ppDDSurf)
```

Description



This function is obsolete and should not be used anymore. We recommend to use the `is_DirectRenderer()` function instead (see also [Obsolete Functions](#)).

In DirectDraw BackBuffer mode, `is_GetDDOvlSurface()` returns the pointer to the internal DirectDraw surface. Thus, the functionality provided by the DirectDraw Surface interface can be used.

Input Parameters

<code>hCam</code>	Camera handle
<code>ppDDSurf</code>	Contains the pointer to the DirectDraw Surface interface



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message

Related Functions

- `is_SetDisplayMode()`
- `is_DisableDDOverlay()`
- `is_EnableDDOverlay()`
- `is_ShowDDOverlay()`
- `is_HideDDOverlay()`

5.5.5 is_HideDDOverlay

	
USB 2.0 GigE	-

Syntax

```
INT is_HideDDOverlay (HIDS hCam)
```

Description



This function is obsolete and should not be used anymore. We recommend to use the [is_DirectRenderer\(\)](#) function instead (see also [Obsolete Functions](#)).

In DirectDraw BackBuffer mode, `is_HideDDOverlay()` hides the overlay. Only the contents of the image buffer will be displayed. This way, the frame rate is higher on some systems than with the overlay shown. By hiding the overlay, its data is not lost.

Input Parameters

hCam	Camera handle
------	---------------



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [is_ShowDDOverlay\(\)](#)
- [is_DisableDDOverlay\(\)](#)
- [is_EnableDDOverlay\(\)](#)
- [is_GetDDOvlSurface\(\)](#)
- [is_SetDisplayMode\(\)](#)

5.5.6 `is_LockDDMem`

	
USB 2.0 GigE	-

Syntax

```
INT is_LockDDMem (HIDS hCam, void** ppMem, INT* pPitch)
```

Description



This function is obsolete and should not be used anymore. We recommend to use the `is_DirectRenderer()` function instead (see also [Obsolete Functions](#)).

`is_LockDDMem()` enables access to the image memory in DirectDraw mode and returns the pointer to the image memory. In most cases, the image memory is located on the graphics card. Using the pointer, you have direct access to the image memory. Make sure to release the memory as early as possible using the `is_UnlockDDMem()` function.

Calling `is_LockDDMem()` will not interrupt the process of digitising an image and saving it to the memory area.

While a `LockDDMem ... UnlockDDMem` block is executed in DirectDraw BackBuffer mode, the contents of the back buffer will not be refreshed on the screen.

Input Parameters

<code>hCam</code>	Camera handle
<code>ppMem</code>	Pointer to the variable that holds the address pointer
<code>pPitch</code>	Pointer to the variable that holds the pitch value



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message

Related Functions

- `is_UnlockDDMem()`
- `is_LockDDOverlayMem()`
- `is_UnlockDDOverlayMem()`
- `is_LockSeqBuf()`
- `is_UnlockSeqBuf()`
- `is_UpdateDisplay()`

5.5.7 is_LockDDOverlayMem

	
USB 2.0 GigE	-

Syntax

```
INT is_LockDDOverlayMem(HIDS hCam, void** ppMem, INT* pPitch)
```

Description



This function is obsolete and should not be used anymore. We recommend to use the [is_DirectRenderer\(\)](#) function instead (see also [Obsolete Functions](#)).

In DirectDraw BackBuffer mode, `is_LockDDOverlayMem()` enables access to the overlay memory and returns the pointer to the starting address of the overlay buffer. This way, you can write data directly to the overlay buffer without the need to use the Windows GDI functions.

`pPitch` returns the line offset (in bytes) from the beginning of a line to the beginning of the next line. Make sure to release the memory again as early as possible using the [is_UnlockDDOverlayMem\(\)](#) function.

While a `LockDDOverlayMem...UnlockDDOverlayMem` block is executed, the contents of the overlay buffer will not be refreshed on the screen.

Input Parameters

<code>hCam</code>	Camera handle
<code>ppMem</code>	Pointer to the variable that holds the address pointer
<code>pPitch</code>	Pointer to the variable that holds the pitch value



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message

Related Functions

- [is_UnlockDDOverlayMem\(\)](#)
- [is_LockDDMem\(\)](#)
- [is_UnlockDDMem\(\)](#)
- [is_LockSeqBuf\(\)](#)
- [is_UnlockSeqBuf\(\)](#)

5.5.8 `is_PrepareStealVideo`

	
USB 2.0 GigE	-

Syntax

```
INT is_PrepareStealVideo (HIDS hCam, int Mode, ULONG StealColorMode)
```

Description



This function is obsolete and should not be used anymore. We recommend to use the `is_DirectRenderer()` function instead (see also [Obsolete Functions](#)).

`is_PrepareStealVideo()` enables steal mode during DirectDraw display. Using the `is_StealVideo()` command, you can remove or copy an image from the DirectDraw video data stream. There are two different steal modes:

- Normal steal
This option redirects a single frame from a DirectDraw video data stream to the active user memory. The frame will not be displayed with DirectDraw.
- Copy steal
This option displays the frame with DirectDraw and copies it to the currently active image memory.

Input Parameters

<code>hCam</code>	Camera handle
<code>Mode</code>	
<code>IS_SET_STEAL_NORMAL</code>	Normal mode
<code>IS_SET_STEAL_COPY</code>	Copy mode
<code>StealColorMode</code>	reserved



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message

Related Functions

- `is_StealVideo()`
- `is_SetDisplayMode()`
- `is_AllocImageMem()`
- `is_SetImageMem()`

5.5.9 is_ReleaseDC

	
USB 2.0 GigE	-

Syntax

INT is_ReleaseDC (HIDS hCam, HDC hDC)

Description



This function is obsolete and should not be used anymore. We recommend to use the [is_DirectRenderer\(\)](#) function instead (see also [Obsolete Functions](#)).

In DirectDraw BackBuffer mode, `is_ReleaseDC()` releases the device context handle of the overlay buffer. If overlay display is enabled using the [is_ShowDDOverlay\(\)](#) function, the contents of the overlay buffer will be updated on the screen after the handle release.

Input Parameters

hCam	Camera handle
hDC	Device context handle returned by is_GetDC()



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [is_GetDC\(\)](#)
- [is_ShowDDOverlay\(\)](#)
- [is_DisableDDOverlay\(\)](#)
- [is_EnableDDOverlay\(\)](#)
- [is_GetDDOvlSurface\(\)](#)
- [is_SetDisplayMode\(\)](#)

5.5.10 `is_SetBayerConversion`

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetBayerConversion (HIDS hCam, INT nMode)
```

Description

`is_SetBayerConversion()` enables you to select one of two algorithms for the Bayer conversion. These algorithms vary in the obtainable quality and in the required computer load.



This function is obsolete and should not be used anymore. We recommend to use the `is_SetColorConverter()` function instead (see also [Obsolete Functions](#)).



This function can only be used for the 24 bit, 32 bit and Y8 colour formats (colour cameras).

Input Parameters

<code>hCam</code>	Camera handle
<code>nMode</code>	
<code>IS_SET_BAYER_CV_BETTER</code>	Good quality, minor colour artifacts, lower computational load
<code>IS_SET_BAYER_CV_BEST</code>	Best quality and edge acuity, higher computational load
<code>IS_GET_BAYER_CV_MODE</code>	Returns the current setting.



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message
Current setting when used together with <code>IS_GET_BAYER_CV_MODE</code>	

Related Functions

- [`is_SetColorConverter\(\)`](#)
- [`is_SetColorMode\(\)`](#)
- [`is_SetColorCorrection\(\)`](#)

5.5.11 is_SetDDUpdateTime

	
USB 2.0 GigE	-

Syntax

```
INT is_SetDDUpdateTime (HIDS hCam, INT ms)
```

Description



This function is obsolete and should not be used anymore. We recommend to use the [is_DirectRenderer\(\)](#) function instead (see also [Obsolete Functions](#)).

`is_SetDDUpdateTime()` sets the timer interval used for the video image update cycle in DirectDraw BackBuffer mode.

Input Parameters

hCam	Camera handle
ms	Time in milliseconds. Valid range: 20...2000 ms



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [is_SetDisplayMode\(\)](#)
- [is_UpdateDisplay\(\)](#)

5.5.12 `is_SetHwnd`

	
USB 2.0 GigE	

Syntax

```
INT is_SetHwnd (HIDS hCam, HWND hwnd)
```

Description



This function is obsolete and should not be used anymore. We recommend to use the `is_DirectRenderer()` function instead (see also [Obsolete Functions](#)).

`is_SetHwnd()` sets a new window handle for image output in DirectDraw mode. The new handle and the image output will only be effective when `is_SetDisplayMode()` is called for the next time.

Input Parameters

<code>hCam</code>	Camera handle
<code>hwnd</code>	Window handle



Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message

Related Functions

- `is_SetDisplayMode()`

5.5.13 is_SetImageAOI

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetImageAOI (HIDS hCam,
                   INT xPos, INT yPos,
                   INT width, INT height)
```

Description



This function is obsolete and should not be used anymore. We recommend to use the [is_SetAOI\(\)](#) function instead (see also [Obsolete Functions](#)).

Input Parameters

hCam	Camera handle
xPos	X position of the upper left corner.
yPos	Y position of the upper left corner.
width	Image width
height	Image height



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
IS_INVALID_MODE	Camera is in standby mode, function not allowed.

Related Functions

- [is_SetAOI\(\)](#)
- [is_SetImagePos\(\)](#)
- [is_SetImageSize\(\)](#)
- [is_SetBinning\(\)](#)
- [is_SetSubSampling\(\)](#)

5.5.14 is_SetImageSize

	
USB 2.0 GigE	USB 2.0 GigE

Syntax

```
INT is_SetImageSize (HIDS hCam, INT x, INT y)
```

Description



This function is obsolete and should not be used anymore. We recommend to use the is_SetAOI() function instead (see also Obsolete Functions).

In conjunction with the is_SetImagePos() settings, is_SetImageSize() determines the size of the captured area of interest (AOI).

To avoid a positional mismatch between the display area and the image area, make sure to call the functions in the correct order. Starting from the original image, it is mandatory to keep to the following order:

1. is_SetImageSize()
2. is_SetImagePos()



is_SetAOI() combines both functions. With is_SetAOI(), you can set the position and size of an area of interest using a single function call.



Changes to the image geometry or pixel clock affect the value ranges of the frame rate and exposure time. After executing is_SetBinning(), calling the following functions is recommended in order to keep the defined camera settings:

- is_SetFrameRate()
- is_SetExposureTime()
- If you are using the *uEye*'s flash function: is_SetFlashStrobe()

Input Parameters

hCam	Camera handle
x	
1...xMax	Sets the image width
IS_GET_IMAGE_SIZE_X	Returns the current image width.
IS_GET_IMAGE_SIZE_X_MIN	Returns the minimum AOI image width.
IS_GET_IMAGE_SIZE_X_MAX	Returns the maximum AOI image width.
IS_GET_IMAGE_SIZE_X_INC	Returns the increment for the AOI image width.
IS_GET_IMAGE_SIZE_Y	Returns the current image height.
IS_GET_IMAGE_SIZE_Y_MIN	Returns the minimum AOI image height
IS_GET_IMAGE_SIZE_Y_MAX	Returns the maximum AOI image height
IS_GET_IMAGE_SIZE_Y_INC	Returns the increment for the AOI image height
y	
1...yMax	Sets the image height
0	Return settings



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message
Current setting when used together with IS_GET_IMAGE_SIZE parameters	
IS_INVALID_MODE	Camera is in standby mode, function not allowed.

Related Functions

- [is_SetAOI\(\)](#)
- [is_SetImagePos\(\)](#)

5.5.15 `is_SetKeyColor`

	
USB 2.0 GigE	-

Syntax

```
INT is_SetKeyColor (HIDS hCam, INT r, INT g, INT b)
```

Description



This function is obsolete and should not be used anymore. We recommend to use the `is_DirectRenderer()` function instead (see also [Obsolete Functions](#)).

Using `is_SetKeyColor()`, you define the key colour for DirectDraw Overlay Surface mode. This function can also be used to return the key colour. Use the `r` parameter to specify the colour value to be returned. Depending on the call type, the function returns either a value reflecting the colour content (0...255) or the corresponding RGB value (0 ... 16777215).

Input Parameters

<code>hCam</code>	Camera handle
<code>r</code>	Red content of the key colour (0...255).
<code>IS_GET_KC_RED</code>	The function returns the red content value.
<code>IS_GET_KC_GREEN</code>	The function returns the green content value.
<code>IS_GET_KC_BLUE</code>	The function returns the blue content value.
<code>IS_GET_KC_RGB</code>	The function returns the RGB colour.
<code>g</code>	Green content of the key colour (0...255).
<code>b</code>	Blue content of the key colour (0...255).

Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message
Colour value when used together with <code>IS_GET_KC_RGB</code> <code>IS_GET_KC_RED</code> <code>IS_GET_KC_GREEN</code> <code>IS_GET_KC_BLUE</code>	

Recommended Alternative



- `is_DirectRenderer()`

Related Functions

- `is_SetDisplayMode()`
- `is_ShowDDOverlay()`
- `is_HideDDOverlay()`
- `is_DisableDDOverlay()`

- is_EnableDDOverlay()
- is_GetDDOvlSurface()
- is_SetColorMode()

5.5.16 `is_ShowDDOverlay`

	
USB 2.0 GigE	-

Syntax

```
INT is_ShowDDOverlay (HIDS hCam)
```

Description



This function is obsolete and should not be used anymore. We recommend to use the `is_DirectRenderer()` function instead (see also [Obsolete Functions](#)).

In DirectDraw BackBuffer mode, `is_ShowDDOverlay()` displays the overlay, i.e. the most recent data stored in the overlay buffer. With some graphics cards, overlay display may reduce the frame rate.

Input Parameters

<code>hCam</code>	Camera handle
-------------------	---------------

Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message



Recommended Alternative

- `is_DirectRenderer()`

Related Functions

- `is_HideDDOverlay()`
- `is_DisableDDOverlay()`
- `is_EnableDDOverlay()`
- `is_GetDDOvlSurface()`
- `is_SetDisplayMode()`

5.5.17 is_StealVideo

	
USB 2.0 GigE	-

Syntax

```
INT is_StealVideo (HIDS hCam, int Wait)
```

Description



This function is obsolete and should not be used anymore. We recommend to use the [is_DirectRenderer\(\)](#) function instead (see also [Obsolete Functions](#)).

Using [is_StealVideo\(\)](#), you can initiate the extraction of an image from the DirectDraw image data stream. The extracted image is written to the active image memory. The data includes the colour format set with the [is_SetColorMode\(\)](#) function.

You can use the [is_PrepareStealVideo\(\)](#) function to specify to remove the image from the DirectDraw data stream or to copy it. If you set the copy option, the image will both be displayed using DirectDraw and copied to the currently active image memory.

See also the *Events in Live Mode* figure in the [Event Handling](#) section.

Input Parameters

hCam	Camera handle
Wait	
IS_WAIT	The function waits until the image save is complete.
IS_DONT_WAIT	The function returns immediately.



Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message

Related Functions

- [is_PrepareStealVideo\(\)](#)
- [is_SetDisplayMode\(\)](#)
- [is_SetColorMode\(\)](#)
- [is_AllocImageMem\(\)](#)
- [is_SetImageMem\(\)](#)
- [is_SetAllocatedImageMem\(\)](#)

5.5.18 `is_UnlockDDMem`

	
USB 2.0 GigE	-

Syntax

```
INT is_UnlockDDMem (HIDS hCam)
```

Description



This function is obsolete and should not be used anymore. We recommend to use the `is_DirectRenderer()` function instead (see also [Obsolete Functions](#)).

Using `is_UnlockDDMem()`, you can unlock the image memory in the DirectDraw modes. This results in a refresh of the BackBuffer contents on the screen.

Input Parameters

<code>hCam</code>	Camera handle
-------------------	---------------

Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message



Recommended Alternative

- `is_DirectRenderer()`

Related Functions

- `is_LockDDMem()`
- `is_LockDDOverlayMem()`
- `is_UnlockDDOverlayMem()`
- `is_LockSeqBuf()`
- `is_UnlockSeqBuf()`
- `is_UpdateDisplay()`

5.5.19 is_UnlockDDOverlayMem

	
USB 2.0 GigE	-

Syntax

```
INT is_UnlockDDOverlayMem (HIDS hCam)
```

Description



This function is obsolete and should not be used anymore. We recommend to use the [is_DirectRenderer\(\)](#) function instead (see also [Obsolete Functions](#)).

Using `is_UnlockDDOverlayMem()`, you can unlock the overlay buffer in DirectDraw BackBuffer mode. This results in an overlay buffer refresh on the screen, provided that the overlay display was enabled using `is_ShowDDOverlay()`.

Input Parameters

hCam	Camera handle
------	---------------

Return Values

IS_SUCCESS	Function executed successfully
IS_NO_SUCCESS	General error message



Recommended Alternative

- [is_DirectRenderer\(\)](#)

Related Functions

- [is_LockDDOverlayMem\(\)](#)
- [is_UnlockDDOverlayMem\(\)](#)
- [is_LockDDMem\(\)](#)
- [is_UnlockDDMem\(\)](#)
- [is_LockSeqBuf\(\)](#)
- [is_UnlockSeqBuf\(\)](#)
- [is_UpdateDisplay\(\)](#)

5.5.20 `is_UpdateDisplay`

	
USB 2.0 GigE	-

Syntax

```
INT is_UpdateDisplay (HIDS hCam)
```

Description



This function is obsolete and should not be used anymore. We recommend to use the `is_DirectRenderer()` function instead (see also [Obsolete Functions](#)).

Using `is_UpdateDisplay()`, you can manually update the screen output in the DirectDraw modes. In normal operation, the driver performs updates automatically. There may be cases, however, where a manual update of the screen output is necessary.

Input Parameters

<code>hCam</code>	Camera handle
-------------------	---------------

Return Values

<code>IS_SUCCESS</code>	Function executed successfully
<code>IS_NO_SUCCESS</code>	General error message

Recommended Alternative

- `is_DirectRenderer()`

Related Functions

- `is_SetDisplayMode()`
- `is_LockDDOverlayMem()`
- `is_UnlockDDOverlayMem()`
- `is_LockDDMem()`
- `is_UnlockDDMem()`

5.6 Lists and Programming Notes

5.6.1 Programming Notes



Apart from camera-specific functions, the *uEye SDK* is almost identical with the SDK for the *FALCON* and *EAGLE* frame grabbers from IDS. The [Compatibility with FALCON Functions](#) chapter includes a list of functions from the *FALCON SDK* which are not supported by the *uEye* camera.

Notes on parameter validity

Functions that refer to an initialized camera have the camera handle `HIDS hCam` as the first parameter. All parameters that are set using these functions remain valid for as long as the handle is valid, that is, until you close the corresponding camera or exit the program. The next time you open the camera, it is initialized with the defaults again.

The uEye.h header file

The *uEye.h* header file contains all the definitions and constants needed for the *uEye API*. You will find this file in the directory `C:\Program Files\IDS\uEye\Develop\include` after installation of the *uEye* drivers.

5.6.1.1 Programming in C / C++

For programming with the *uEye API*, we suggest that you use the C / C++ programming language. This programming language offers efficient access to all functions of the *uEye API*. Enabling access to image memory contents through pointers, C / C++ is especially suitable for image processing applications.

Most of the *uEye* sample programs were created in Microsoft Visual Studio using the C++ programming language.

Required Files

In order to access the *uEye API*, make sure to include the following files in your project:

- Header file: *uEye.h*
- Lib file: *uEye_api.lib*
- Function library (DLL): *uEye_Api.dll*

In order to access the *uEye AVI* functions, make sure to include the following files in your project:

- Header file: *uEye_tools.h*
- Lib file: *uEye_tools.lib*
- Function library (DLL): *uEye_tools.dll*

In order to access the *uEye DirectShow* functions, make sure to include the following files in your project:

- Header file: *uEyeCaptureInterface.h*
- DirectShow interface: *uEyecapture.ax*



We suggest that you keep the function libraries (DLL, AX and OCX files) in the default directory. Under Windows, these files reside in `C:\Windows\System32\` after the installation. Copying these files to other locations may result in version conflicts.

5.6.1.2 Programming in C#

We suggest to use the C# programming language for the creation of visualisation applications. While it is possible to access image memory contents, doing so is more tedious than in C/C++ due to the 'managed code'. To access image memory contents in C#, you can use 'unsafe code' or the 'Marshall class'. Some system-level functions, such as Windows event handling, can be integrated using the Windows API. The *uEye* SDK includes sample programs for programming with Microsoft Visual Studio in the C# programming language.

Required Files

In order to access the *uEye* API in C#, make sure to include the following files in your project:

- Header file: *uEye.cs*
- Function library (DLL): *uEye_Api.dll*

In order to access the *uEye* AVI functions in C#, make sure to include the following files in your project:

- Header file: *uEye_tools.cs*
- Function library (DLL): *uEye_tools.dll*



We suggest that you keep the function libraries (DLL, AX and OCX files) in the default directory. Under Windows, these files reside in C:\Windows\System32\ after the installation. Copying these files to other locations may result in version conflicts.

5.6.1.3 Programming in VB.NET

We suggest to use the Visual Basic programming language for the creation of applications which are exclusively used for visualisation purposes. The access to image memory contents is extremely tedious due to the missing pointer arithmetics.

We suggest to use the *uEye* ActiveX component when programming in VB.Net. The *uEye* SDK includes a sample program for programming with Microsoft Visual Studio in the VB.NET programming language using the *uEye* ActiveX component.

5.6.1.4 Programming in Delphi

The *uEye* SDK does not provide direct integration of the *uEye* API for the Delphi programming language. In order to use the *uEye* API in Delphi, you need to create separate header files. We suggest to use the *uEye* ActiveX component (see also [Programming with ActiveX](#)) when programming in Delphi.



We suggest that you keep the function libraries (DLL, AX and OCX files) in the default directory. Under Windows, these files reside in C:\Windows\System32\ after the installation. Copying these files to other locations may result in version conflicts.

5.6.1.5 Programming with ActiveX

The *uEye* SDK comes with an ActiveX component that allows you to use almost all functions of the *uEye*. Programming the *uEye* ActiveX component is described in the *uEye ActiveX Manual*. After the installation, you will find this manual in the C:\Programs\IDS\uEye\Help directory.

Required Files

In order to access the *uEye* ActiveX component, make sure to include the following file in your project:

- ActiveX control: *uEyeCam.ocx*



We suggest that you keep the function libraries (DLL, AX and OCX files) in the default directory. Under Windows, these files reside in C:\Windows\System32\ after the installation.

Copying these files to other locations may result in version conflicts.

5.6.1.6 Thread Programming

In general, the *uEye* API is *thread-safe*. This means that the *uEye* API can be accessed by multiple threads simultaneously. Simultaneous attempts to call the same function are recognised and prevented by the driver.



We recommend that you call the following functions only from a single thread in order to avoid unpredictable behaviour of the application.

- *is_InitCamera()*
- *is_SetDisplayMode()*
- *is_ExitCamera()*

5.6.2 Complete List of All Return Values

No	Error	Description
-1	IS_NO_SUCCESS	General error message
0	IS_SUCCESS	General message indicating successful execution – no error
1	IS_INVALID_CAMERA_HANDLE	The camera handle is invalid. Most of the <i>uEye SDK</i> functions expect the camera handle as the first parameter.
2	IS_IO_REQUEST_FAILED	An IO request from the <i>uEye</i> driver failed. Possibly the versions of the <i>ueye_api.dll</i> (API) and the driver file (<i>ueye_usb.sys</i> or <i>ueye_eth.sys</i>) do not match.
3	IS_CANT_OPEN_DEVICE	An attempt to initialise or select the camera failed (no camera connected or initialisation error).
11	IS_CANT_OPEN_REGISTRY	Error opening a Windows registry key.
12	IS_CANT_READ_REGISTRY	Error reading settings from the Windows registry.
15	IS_NO_IMAGE_MEM_ALLOCATED	The driver could not allocate memory.
16	IS_CANT_CLEANUP_MEMORY	The driver could not release the allocated memory.
17	IS_CANT_COMMUNICATE_WITH_DRIVER	Communication with the driver failed because no driver has been loaded.
18	IS_FUNCTION_NOT_SUPPORTED_YET	The function is not supported yet.
49	IS_INVALID_MEMORY_POINTER	Invalid pointer or invalid memory ID.
50	IS_FILE_WRITE_OPEN_ERROR	File cannot be opened for writing.
51	IS_FILE_READ_OPEN_ERROR	File cannot be opened for reading.
52	IS_FILE_READ_INVALID_BMP_ID	The specified file is not a valid bitmap file.
53	IS_FILE_READ_INVALID_BMP_SIZE	The bitmap size is not correct (bitmap too large).
108	IS_NO_ACTIVE_IMG_MEM	No active image memory available. You must set the memory to active using the <i>is_SetImageMem()</i> function or create a sequence using the <i>is_AddToSequence()</i> function.
112	IS_SEQUENCE_LIST_EMPTY	The sequence list is empty and cannot be deleted.
113	IS_CANT_ADD_TO_SEQUENCE	The image memory is already included in the sequence and cannot be added again.
117	IS_SEQUENCE_BUF_ALREADY_LOCKED	The memory could not be locked. The pointer to the buffer is invalid.

No	Error	Description
118	IS_INVALID_DEVICE_ID	The device ID is invalid. Valid IDs start from 1 for USB cameras, and from 1001 for GigE cameras.
119	IS_INVALID_BOARD_ID	The board ID is invalid. Valid IDs range from 1 through 255.
120	IS_ALL_DEVICES_BUSY	All cameras are in use.
122	IS_TIMED_OUT	A timeout occurred. An image capturing process could not be terminated within the allowable period.
125	IS_INVALID_PARAMETER	One of the submitted parameters is outside the valid range or is not supported for this sensor or is not available in this mode.
127	IS_OUT_OF_MEMORY	No memory could be allocated.
139	IS_NO_USB20	The camera is connected to a port which does not support the USB 2.0 high-speed standard.
140	IS_CAPTURE_RUNNING	A capturing operation is in progress and must be terminated before you can start another one.
145	IS_IMAGE_NOT_PRESENT	The requested image is not available in the camera memory or is no longer valid.
148	IS_TRIGGER_ACTIVATED	The function cannot be used because the camera is waiting for a trigger signal.
151	IS_CRC_ERROR	A CRC error correction problem occurred while reading the settings.
152	IS_NOT_YET_RELEASED	This function has not been enabled yet in this version.
153	IS_NOT_CALIBRATED	The camera does not contain any calibration data.
154	IS_WAITING_FOR_KERNEL	The system is waiting for the kernel driver to respond.
155	IS_NOT_SUPPORTED	The camera model used here does not support this function or setting.
157	IS_OPERATION_ABORTED	The dialogue was cancelled without a selection so that no file could be saved.
158	IS_BAD_STRUCTURE_SIZE	An internal structure has an incorrect size.
159	IS_INVALID_BUFFER_SIZE	The image memory has an inappropriate size to store the image in the desired format.
160	IS_INVALID_PIXEL_CLOCK	This setting is not available for the currently set pixel clock frequency.

No	Error	Description
161	IS_INVALID_EXPOSURE_TIME	This setting is not available for the currently set exposure time.
162	IS_AUTO_EXPOSURE_RUNNING	This setting cannot be changed while automatic exposure time control is enabled.
163	IS_CANNOT_CREATE_BB_SURF	The BackBuffer surface cannot be created.
164	IS_CANNOT_CREATE_BB_MIX	The BackBuffer mix surface cannot be created.
165	IS_BB_OVLMEM_NULL	The BackBuffer overlay memory cannot be locked.
166	IS_CANNOT_CREATE_BB_OVL	The BackBuffer overlay memory cannot be created.
167	IS_NOT_SUPP_IN_OVL_SURF_MODE	Not supported in BackBuffer Overlay mode.
168	IS_INVALID_SURFACE	Back buffer surface invalid.
169	IS_SURFACE_LOST	Back buffer surface not found.
170	IS_RELEASE_BB_OVL_DC	Error releasing the overlay device context.
171	IS_BB_TIMER_NOT_CREATED	The back buffer timer could not be created.
172	IS_BB_OVL_NOT_EN	The back buffer overlay was not enabled.
173	IS_ONLY_IN_BB_MODE	Only possible in BackBuffer mode.
174	IS_INVALID_COLOR_FORMAT	Invalid colour format.
175	IS_INVALID_WB_BINNING_MODE	Mono binning / mono sub-sampling do not support automatic white balance.
176	IS_INVALID_I2C_DEVICE_ADDRESS	Invalid I2C device address.
177	IS_COULD_NOT_CONVERT	The current image could not be processed.
178	IS_TRANSFER_ERROR	Transfer error.
179	IS_PARAMETER_SET_NOT_PRESENT	Parameter set is not present.
180	IS_INVALID_CAMERA_TYPE	The camera type defined in the .ini file does not match the current camera model.
184	IS_STARTER_FW_UPLOAD_NEEDED	The camera's starter firmware is not compatible with the driver and needs to be updated (see is_InitCamera()).

5.6.3 List of API Functions Available Under Linux

<u>is_AddToSequence()</u>
<u>is_AllocImageMem()</u>
<u>is_CameraStatus()</u>
<u>is_CaptureVideo()</u>
<u>is_ClearSequence()</u>
<u>is_ConvertImage()</u>
<u>is_CopyImageMem()</u>
<u>is_CopyImageMemLines()</u>
<u>is_DisableEvent()</u>
<u>is_EnableAutoExit()</u>
<u>is_EnableEvent()</u>
<u>is_EnableHdr()</u>
<u>is_ExitCamera()</u>
<u>is_ExitEvent()</u>
<u>is_ForceTrigger()</u>
<u>is_FreeImageMem()</u>
<u>is_FreezeVideo()</u>
<u>is_GetActiveImageMem()</u>
<u>is_GetActSeqBuf()</u>
<u>is_GetAutoInfo()</u>
<u>is_GetBusSpeed()</u>
<u>is_GetCameraInfo()</u>
<u>is_GetCameraList()</u>
<u>is_GetCameraLUT()</u>
<u>is_GetCameraType()</u>
<u>is_GetCaptureErrorInfo()</u>
<u>is_GetColorConverter()</u>
<u>is_GetComportNumber()</u>
<u>is_GetDLLVersion()</u>
<u>is_GetError()</u>
<u>is_GetEthDeviceInfo()</u>
<u>is_GetExposureRange()</u>
<u>is_GetFramesPerSecond()</u>
<u>is_GetFrameTimeRange()</u>
<u>is_GetGlobalFlashDelays()</u>
<u>is_GetHdrKneepointInfo()</u>
<u>is_GetHdrKneepoints()</u>
<u>is_GetHdrMode()</u>
<u>is_GetImageHistogram()</u>
<u>is_GetImageInfo()</u>
<u>is_GetImageMem()</u>
<u>is_GetImageMemPitch()</u>
<u>is_GetNumberOfCameras()</u>
<u>is_GetOsVersion()</u>
<u>is_GetPixelClockRange()</u>
<u>is_GetSensorInfo()</u>
<u>is_GetSupportedTestImages()</u>
<u>is_GetTestImageValueRange()</u>
<u>is_GetTimeout()</u>
<u>is_GetUsedbandwidth()</u>
<u>is_GetVsyncCount()</u>
<u>is_HasVideoStarted()</u>

<code>is_InitCamera()</code>
<code>is_InitEvent()</code>
<code>is_InquireImageMem()</code>
<code>is_IsVideoFinish()</code>
<code>is_LoadBadPixelCorrectionTable()</code>
<code>is_LoadImage()</code>
<code>is_LoadImageMem()</code>
<code>is_LoadParameters()</code>
<code>is_LockSeqBuf()</code>
<code>is_ReadEEPROM()</code>
<code>is_ReadI2C()</code>
<code>is_ResetCaptureErrorInfo()</code>
<code>is_ResetToDefault()</code>
<code>is_SaveBadPixelCorrectionTable()</code>
<code>is_SaveImage()</code>
<code>is_SaveImageEx()</code>
<code>is_SaveImageMem()</code>
<code>is_SaveImageMemEx()</code>
<code>is_SaveParameters()</code>
<code>is_SetAllocatedImageMem()</code>
<code>is_SetAOI()</code>
<code>is_SetAutoCfgIpSetup()</code>
<code>is_SetAutoParameter()</code>
<code>is_SetBadPixelCorrection()</code>
<code>is_SetBadPixelCorrectionTable()</code>
<code>is_SetBayerConversion()</code>
<code>is_SetBinning()</code>
<code>is_SetBlCompensation()</code>
<code>is_SetCameraID()</code>
<code>is_SetCameraLUT()</code>
<code>is_SetColorConverter()</code>
<code>is_SetColorCorrection()</code>
<code>is_SetColorMode()</code>
<code>is_SetConvertParam()</code>
<code>is_SetEdgeEnhancement()</code>
<code>is_SetErrorReport()</code>
<code>is_SetExposureTime()</code>
<code>is_SetExternalTrigger()</code>
<code>is_SetFlashDelay()</code>
<code>is_SetFlashStrobe()</code>
<code>is_SetFrameRate()</code>
<code>is_SetGainBoost()</code>
<code>is_SetGamma()</code>
<code>is_SetGlobalShutter()</code>
<code>is_SetHardwareGain()</code>
<code>is_SetHardwareGamma()</code>
<code>is_SetHdrKneepoints()</code>
<code>is_SetHWGainFactor()</code>
<code>is_SetImageAOI() *)</code>
<code>is_SetImageMem()</code>
<code>is_SetImagePos()</code>
<code>is_SetImageSize() *)</code>
<code>is_SetIO()</code>

<u>is_SetIOMask()</u>
<u>is_SetLED()</u>
<u>is_SetOptimalCameraTiming()</u>
<u>is_SetPacketFilter()</u>
<u>is_SetPersistentIpCfg()</u>
<u>is_SetPixelClock()</u>
<u>is_SetRopEffect()</u>
<u>is_SetSaturation()</u>
<u>is_SetSensorTestImage()</u>
<u>is_SetStarterFirmware()</u>
<u>is_SetSubSampling()</u>
<u>is_SetTimeout()</u>
<u>is_SetTriggerCounter()</u>
<u>is_SetTriggerDelay()</u>
<u>is_StopLiveVideo()</u>
<u>is_UnlockSegBuf()</u>
<u>is_WriteEEPROM()</u>
<u>is_WriteI2C()</u>
<u>Isavi_AddFrame()</u>
<u>Isavi_CloseAVI()</u>
<u>Isavi_DisableEvent()</u>
<u>Isavi_EnableEvent()</u>
<u>Isavi_ExitAVI()</u>
<u>Isavi_ExitEvent()</u>
<u>Isavi_GetAVIFileName()</u>
<u>Isavi_GetAVISize()</u>
<u>Isavi_GetnCompressedFrames()</u>
<u>Isavi_GetnLostFrames()</u>
<u>Isavi_InitAVI()</u>
<u>Isavi_InitEvent()</u>
<u>Isavi_OpenAVI()</u>
<u>Isavi_ResetFrameCounters()</u>
<u>Isavi_SetFrameRate()</u>
<u>Isavi_SetImageQuality()</u>
<u>Isavi_SetImageSize()</u>
<u>Isavi_StartAVI()</u>
<u>Isavi_StopAVI()</u>

*) Function ist obsolete, see chapter [Obsolete Functions](#).

5.6.4 Compatibility with FALCON Functions

Compatible FALCON Functions



The functions from the *FALCON* frame grabber series listed below are supported by the *uEye* camera series for compatibility reasons, but we recommend to use the indicated *uEye* SDK functions instead.

Obsolete FALCON function	Recommended <i>uEye</i> SDK function
<code>is_InitBoard()</code>	<code>is_InitCamera()</code>
<code>is_ExitBoard()</code>	<code>is_ExitCamera()</code>
<code>is_InitFalcon()</code>	<code>is_InitCamera()</code>
<code>is_ExitFalcon()</code>	<code>is_ExitCamera()</code>
<code>is_GetBoardType()</code>	<code>is_GetCameraInfo()</code>
<code>is_GetBoardInfo()</code>	<code>is_GetCameraInfo()</code>
<code>is_BoardStatus()</code>	<code>is_CameraStatus()</code>
<code>is_GetNumberOfDevices()</code>	<code>is_GetNumberOfCameras()</code>
<code>is_GetNumberOfBoards()</code>	<code>is_GetNumberOfCameras()</code>

Incompatible FALCON Functions



The functions listed below are specific to the *FALCON* frame grabber series and are not supported by the *uEye* camera series.

`is_GetCurrentField()`

`is_GetIRQ()`

`is_GetPciSlot()`

`is_OvlSurfaceOffWhileMove()`

`is_ScaledDDOverlay()`

`is_SetAGC()`

`is_SetCaptureMode()`

`is_SetDecimationMode()`

`is_SetDisplaySize()`

`is_SetHorFilter()`

`is_SetHue()`

`is_SetKeyOffset()`

`is_SetParentHwnd()`

`is_SetPassthrough()`

`is_SetRenderMode()`

`is_SetSync()`

`is_SetSyncLevel()`

`is_SetToggleMode()`

`is_SetUpdateMode()`

`is_SetVertFilter()`

`is_SetVideoCrossbar()`

`is_SetVideoInput()`

`is_SetVideoMode()`

`is_SetVideoSize()`

`is_ShowColorBars()`

`is_Watchdog()`

`is_WatchdogTime()`

6 C: Specifications

This chapter lists the specifications of the available *uEye* camera models. In the table below, you will find an overview of all the models in the *uEye* camera range.

			UI	–	##	#	#	–	X	–	XX
Short for <i>uEye</i>											
Sensor and shutter system											
12	USB <i>uEye</i> CMOS Global Shutter										
14/15/16	USB <i>uEye</i> CMOS Rolling Shutter										
22/23/24	USB <i>uEye</i> CCD Progressive Scan										
52	GigE <i>uEye</i> CMOS Global Shutter										
54/55/56	GigE <i>uEye</i> CMOS Rolling Shutter										
62/63/64	GigE <i>uEye</i> CCD Progressive Scan										
Resolution											
1	VGA 640 x 480 (0.30 Mpixels)										
2	WVGA 752 x 480 (0.36 Mpixels)										
	PAL 768 x 582 (0.45 Mpixels)										
3	XGA 1024 x 768 (0.78 Mpixels)										
4	SXGA 1280 x 1024 (1.30 Mpixels)										
5	UXGA 1600 x 1200 (2.00 Mpixels)										
6	QXGA 2048 x 1536 (3.10 Mpixels)										
8	QSXGA 2592 x 1944 (5.00 Mpixels)										
9	QHDTV 3840 x 2748 (10.55 Mpixels)										
Housing											
0SE	SE series, C-mount with straight housing										
0ME	ME series, C-mount with angled housing										
0RE	RE series, C-mount with IP65/67 housing										
1SE	SE series, OEM version, C-mount without housing										
2SE	SE series, OEM version, PCB stack										
5LE	LE series, CS-mount without housing										
6LE	LE series, board level with S-mount M12										
7LE	LE series, board level with S-mount M14										
8LE	LE series, board level without S-mount										
9HE	HE series, C-mount with angled housing										
Color format											
M	Monochrome sensor										
C	Color sensor										
Filter glass											
HQ	Infrared cut filter, type HQ (standard for color cameras)										
BG	Infrared cut filter, type BG (discontinued)										
DL	Daylight cut filter (optional)										
GL	Plain glass (standard for monochrome cameras)										

6.1 Model Comparison

The following table outlines the key features of each *uEye* camera series for direct comparison (see also the *uEye Camera Family* chapter).

	USB uEye LE	USB uEye SE	USB uEye ME	USB uEye RE	GigE uEye SE	GigE uEye RE	GigE uEye HE
CMOS / CCD models	+ / -	+ / +	+ / +	+ / +	+ / +	+ / +	+ / +
S- / CS-/C-mount	+ / + / (+)	- / - / +	- / - / +	- / - / +	- / - / +	- / - / +	- / - / +
Housing / board-level (OEM)	+ / +	+ / +	+ / +	+ / -	+ / +	+ / -	+ / -
EMC Compliance of housing	CE B, FCC	CE B, FCC	CE B, FCC	CE B, FCC	CE A, FCC **)	CE B, FCC **)	CE B, FCC **)
WxHxD in mm, CCD in parentheses	44 x 44 x 25.6 36 x 36 *)	32 x 34 x 30.2 (37.2) 30 x 30 *)	44 x 71 x 33.6 (36.2)	41 x 41 x 40.5 (55.5)	44 x 34 x 41.5 (43.5)	53 x 41 x 42.7 (47.7)	38 x 38 x 89.5 (89.5)
Mounting holes bottom/top/side/front	1 / 0 / 0 / 4*	8 / 3 / 3 / 0	0 / 0 / 0 / 6	6 / 6 / 6 / 0	4 / 0 / 0 / 0	4 / 4 / 3 / 0	4 / 4 / 3 / 0
Thread diameter	5/8"	M2 / M3	M4 hole	M3 / M5	M3	M3	M3
Adjustable flange back distance	+	-	-	-	-	-	+
IP protection class	30	30	30	65 / 67	30	65 / 67	30
Interface	USB 2.0	USB 2.0	USB 2.0	USB 2.0	GigE	GigE	GigE
Power Supply	USB	USB	USB	USB	12 V, external	12 V, external	6-24 V, external
Lockable connector	-	+	+	+	+	+	+
I/O connector	10-pin connector	9-pin micro D-sub	6-pin HR10	4-pin Binder	6-pin HR10	7-pin Binder	14-pin MDR
Opto coupler for I/O	- / -	1 / 1	1 / 1	1 / 1	1 / 1	1 / 1	1 / 1
Opto coupler speed	-	100 µs	100 µs	100 µs	100 µs	100 µs	1 µs
Max. cable length (m)	5	5	5	8	100	100	100
Dig. I / O / GPIO	1 / 1 / 2*)	1 / 1 / 0	1 / 1 / 1*)	1 / 1 / 0	1 / 1 / 0	1 / 1 / 0	1 / 1 / 2
RS232	-	-	-	-	-	-	+
I ² C bus	+))	-	+))	-	-	-	-
Hot pixel correction	Software	Software	Software	Software	Hardware	Hardware	Hardware
Color calculation	Software	Software	Software	Software	Software	Software	Hardware
Bit depth: Internal / transferred	8 / 8	8 / 8	8 / 8	8 / 8	10 / 8	10 / 8	12 / 32
LUT: Internal / transferred	-	-	-	-	10 / 8	10 / 8	12 / 12
Max. pixel clock (MHz) at full resolution	43	43	43	43	96	96	103
Image memory	-	-	-	-	32 MB	32 MB	64 MB

*) board-level version only **) planned

6.2 Sensor Data



The diagrams shown in the sensor specifications section indicate the *relative* sensitivities of the *uEye* cameras in the spectral range. Therefore, the characteristic curves cannot be compared to each other.

6.2.1 UI-122x / UI-522x

Sensor specification				
Sensor type	CMOS			
Shutter system	Electronic global shutter			
Readout mode	Progressive scan			
Resolution class	WVGA			
Resolution	752 x 480 pixels (0.36 Mpixels)			
Aspect ratio	14:9			
Bit depth	10 bits			
Optical sensor class	1/3 inch			
Exact sensitive area	4.51 mm x 2.88 mm			
Exact optical sensor diagonal	5.4 mm (1/3.0 inch)			
Pixel size	6.0 μm, square			
Sensor name, monochrome	Micron MT9V032C12STM			
Sensor name, color	Micron MT9V022I77ATC			
Sensor name, color (<i>USB uEye LE</i>)	Micron MT9V032C12STC			
Special features	<ul style="list-style-type: none">• HDR mode with two kneepoints for extended dynamic range• Sensor internal controls for exposure (AES) and gain (AGC), can be used optionally			
Gain				
Monochrome model (master gain)	4.0x			
Color model (master/RGB)	4.0x/5.0x (software)			
Gain boost	1.6x			
Camera timing		USB uEye	GigE uEye SE	GigE uEye HE
Pixel clock range	MHz	5-43 ^{*1)}	5-46 ^{*1)}	5-46 ^{*1)}
Max. pixel clock with subsampling/binning	MHz	60 ^{*1)}	59 ^{*1)}	60 ^{*1)}
Frame rate (freerun mode)	fps	87.2 ^{*2)}	100.0 ^{*2)}	100.0 ^{*2)}
Frame rate (trigger mode, 1 ms exposure)	fps	83.3 ^{*2)}	96.9 ^{*2)}	96.9 ^{*2)}
Exposure time in freerun mode	ms	0.037 ^{*2)} -5580 ^{*3)}	0.032 ^{*2)} -5580 ^{*3)}	0.032 ^{*2)} -5580 ^{*3)}
Exposure time in trigger mode	ms	0.037 ^{*2)} -5580 ^{*3)}	0.032 ^{*2)} -5580 ^{*3)}	0.032 ^{*2)} -5580 ^{*3)}
AOI				
Mode		Horizontal ^{*4)} + Vertical ^{*4)}		
AOI image width, step width	Pixels	16-752, 4	16-752, 4	16-752, 4

AOI image height, step width	Pixels	4-480, 2	4-480, 2	4-480, 2
AOI position grid horizontal, vertical	Pixels	4, 2	4, 2	4, 2
AOI frame rate, 640 x 480 pixels (VGA)	fps	100	115	115
AOI frame rate, 320 x 240 pixels (CIF)	fps	200	231	231
Binning				
Mode		Horizontal ^{*4)} + Vertical ^{*4)}		
Method		H + V: Monochrome binning, averaging		
Factors		2x, 4x		
Frame rate with 2x binning, 376 x 240 pixels	fps	274	274	274
Frame rate with 4x binning, 188 x 120 pixels	fps	466	466	466
Subsampling				
Mode		-		
Hardware trigger				
Mode		Asynchronous	Asynchronous	Asynchronous
Trigger delay with rising edge	µs	20.6 ±0.25	3.3 ±0.25	0.3 ±0.25
Trigger delay with falling edge	µs	38.8 ±0.25	11.8 ±0.25	0.2 ±0.25
Additive trigger delay (optional)	µs	15 µs...4 s	15 µs...4 s	15 µs...4 s
Sensor delay to exposure start	µs	< 200 ^{*2)}	< 200 ^{*2)}	< 200 ^{*2)}
Power consumption ^{*5)}				
	W	USB uEye	GigE uEye SE	GigE uEye HE
		0.4-1.0	2.3-3.1	2.9-4.2

^{*1)} The maximum possible pixel clock frequency depends on the PC hardware used.

^{*2)} Requires maximum pixel clock frequency.

^{*3)} Requires minimum pixel clock frequency.

^{*4)} Use of this function increases the frame rate.

^{*5)} The power consumption depends on the sensor model and the pixel clock setting.

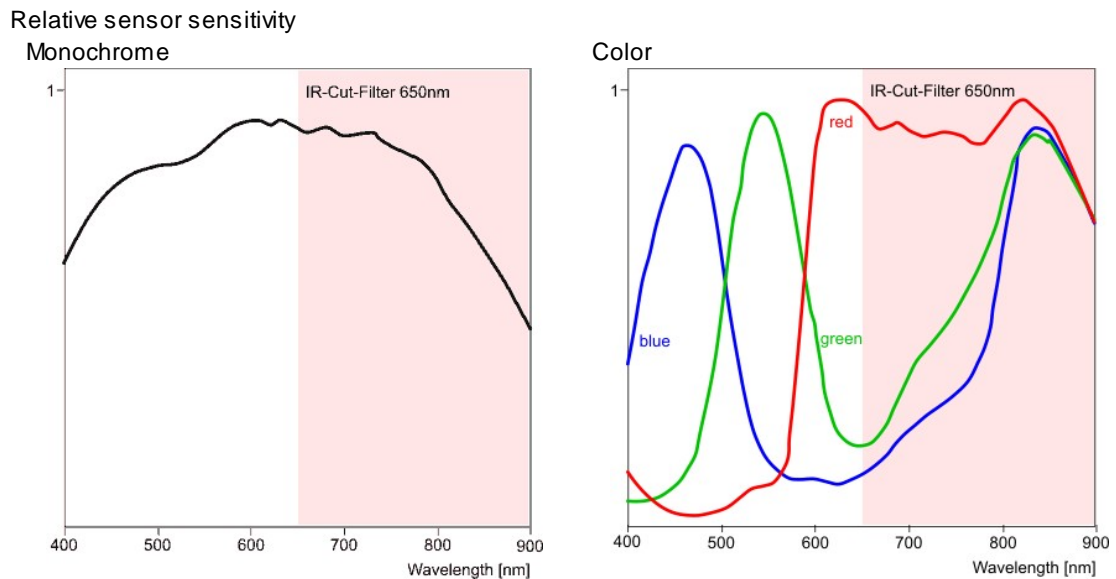


Figure 122: Sensor sensitivity of the UI-122x / UI-522x



Notes on using the UI-122x / UI-522x

- Optimum pixel clock frequency is 27 MHz.
- The color version has no hardware gain controls. The driver simulates these.
- The RGB gain controls have no effect in raw Bayer mode.
- Sensor brightness control: The sensor does not use the average image brightness value as the actual value for brightness control. Instead, it uses a value calculated internally from the histogram. This value is defined with 12% of the pixels being brighter than the actual value.
- Sensor brightness control: The permissible value range for the reference value is [44...235]. You cannot set any smaller values (down to black = 0) or higher values (up to white = 255).
- Sensor speed does not increase for AOI width <608 pixels (constant image height).
- The sensor binning works by averaging pixels, so the image will not become brighter when binning is activated.
- The frame rate is not significantly higher with horizontal 4x binning than with 2x binning.
- Extreme overexposure may shift the black level. As an effect, the white level is no longer reached.
- Functions that modify image content (such as exposure or gain) are applied with a delay of one frame time. This is also the case in trigger mode.
- IR illumination with 900 nm causes blooming.
- With horizontal 4x binning, a dark column appears at the right-hand image border, which is caused by the sensor.
- For sensor reasons, the (black level) offset cannot be modified when HDR mode is active.
- Master gain and gain boost should be disabled when using HDR mode.

6.2.2 UI-146x / UI-546x

Sensor specification				
Sensor type	CMOS			
Shutter system	Electronic rolling shutter			
Readout mode	Progressive scan			
Resolution class	QXGA			
Resolution	2048 x 1536 pixels (3.2 Mpixels)			
Aspect ratio	4:3			
Bit depth	10 bits			
Optical sensor class	1/2 inch			
Exact sensitive area	6.55 x 4.92 mm			
Exact optical sensor diagonal	8.2 mm (1/2.0 inch)			
Pixel size	3.2 μm, square			
Sensor name, monochrome	-			
Sensor name, color	Micron MT9T001			
Gain				
Color model (master/RGB)	12.0x/7.25x			
Gain boost	2.0x			
Camera timing		USB uEye	GigE uEye SE	GigE uEye HE
Pixel clock range	MHz	5-43 ^{*1)}	3-60 ^{*1)}	3-60 ^{*1)}
Max. pixel clock with subsampling/binning	MHz	64 ^{*1)}	64 ^{*1)}	64 ^{*1)}
Frame rate (freerun mode)	fps	11.2 ^{*2)}	15.7 ^{*2)}	15.7 ^{*2)}
Frame rate (trigger mode, 1 ms exposure)	fps	11.2 ^{*2)}	15.6 ^{*2)}	15.6 ^{*2)}
Exposure time in freerun mode	ms	0.057 ^{*2)} -1744 ^{*3)}	0.041 ^{*2)} -2912 ^{*3)}	0.041 ^{*2)} -2912 ^{*3)}
Exposure time in trigger mode	ms	0.057 ^{*2)} -750 ^{*3)}	0.041 ^{*2)} -1251 ^{*3)}	0.041 ^{*2)} -1251 ^{*3)}
AOI				
Mode		Horizontal ^{*4)} + Vertical ^{*4)}		
AOI image width, step width	Pixels	16 - 2048, 4	16 - 2048, 4	16 - 2048, 4
AOI image height, step width	Pixels	4 - 1536, 2	4 - 1536, 2	4 - 1536, 2
AOI position grid horizontal, vertical	Pixels	4, 2	4, 2	4, 2
AOI frame rate, 1920 x 1080 pixels (HD 1080)	fps	16.7	23.3	23.3
AOI frame rate, 1280 x 720 pixels (HD 720)	fps	34.1	47.6	47.6
Binning				
Mode		Horizontal ^{*4)} + Vertical ^{*4)}		
Method		H + V: Color binning, H: additive, V: averaging		
Factor		2x, 3x, 4x, 6x		
Subsampling				

Mode		Horizontal ^{*4)} + Vertical ^{*4)}		
Method		H + V: Color subsampling		
Factor		2x, 3x, 4x, 5x, 6x, 8x		
Frame rate w/ 2x subsampling, 1024 x 768 pixels	fps	37.8	52.8	52.8
Frame rate with 3x subsampling, 680 x 480 pixels	fps	107	113	113
Hardware trigger				
Mode		Asynchronous	Asynchronous	Asynchronous
Trigger delay with rising edge	µs	22.4 ±0.25	3.1 ±0.25	0.3 ±0.25
Trigger delay with falling edge	µs	40.7 ±0.25	11.9 ±0.25	0.2 ±0.25
Additive trigger delay (optional)	µs	15 µs...4 s	15 µs...4 s	15 µs...4 s
Sensor delay to exposure start	µs	< 200 ^{*2)}	< 200 ^{*2)}	< 200 ^{*2)}
Power consumption ^{*5)}				
	W	USB uEye	GigE uEye SE	GigE uEye HE
		0.4-0.7	2.4-2.9	2.8-3.9

*1) The maximum possible pixel clock frequency depends on the PC hardware used.

*2) Requires maximum pixel clock frequency.

*3) Requires minimum pixel clock frequency.

*4) Use of this function increases the frame rate.

*5) The power consumption depends on the sensor model and the pixel clock setting.

Relative sensor sensitivity

Color

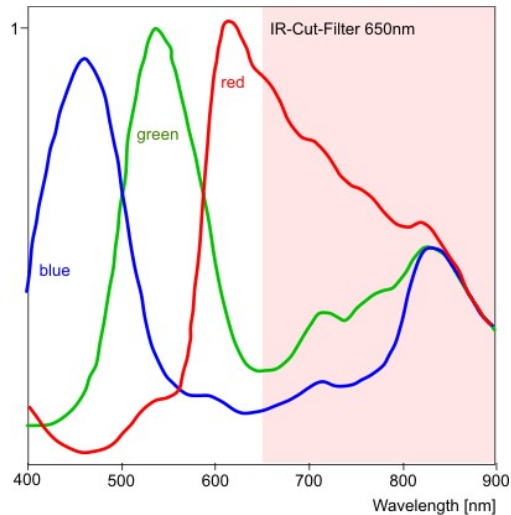


Figure 123: Sensor sensitivity of the UI-146x / UI-546x



Notes on using the UI-146x/ UI-546x

- Master gain is digitally calculated on the sensor and may cause artefacts. Instead use RGB gains first (e.g. by setting a minimum value in the [Auto White Balance](#) function).
- The sensor does not allow changes of exposure time while in trigger mode. If *is_SetExposureTime()* is called in trigger mode, the sensor will temporarily switch to freerun. This results in a longer delay time (depending on the frame rate) at function call.
- Sensor speed does not increase for effective horizontal resolution <256 pixels.
- Changing the frame rate in trigger mode has no effect. The maximum possible exposure time cannot be increased in this way.
- With horizontal 4x binning, a dark column appears at the right-hand image border, which is caused by the sensor.
- For hardware reasons, the sensor can not perform more than 3x vertical binning. When 4x or 6x binning is activated in the uEye software, the driver uses a combination of binning and subsampling instead. Therefore, the image will not become brighter when 4x or 6x horizontal binning is activated.

6.2.3 UI-148x / UI-548x

Sensor specification				
Sensor type	CMOS			
Shutter system	Electronic rolling shutter / global start shutter			
Readout mode	Progressive scan			
Resolution class	QSXGA			
Resolution	2560 x 1920 pixels (4.92 MP)			
Aspect ratio	4:3			
Bit depth	12 bits			
Optical sensor class	1/2 inch			
Exact sensitive area	5.63 x 4.22 mm			
Exact optical sensor diagonal	7.0 mm (1/2.3 inch)			
Pixel size	2.2 μm, square			
Sensor name, monochrome	Micron MT9P031			
Sensor name, color	Micron MT9P031			
Gain				
Monochrome model (master gain)	30.0x			
Color model (master/RGB)	12.0x/6.5x			
Gain boost	1.6x (color model only)			
Camera timing		USB uEye	GigE uEye SE	GigE uEye HE
Pixel clock range	MHz	5-43 ^{*1)}	4-96 ^{*1)}	4-103 ^{*1)}
Max. pixel clock with subsampling/binning	MHz	90 ^{*1)}	100 ^{*1)}	128 ^{*1)}
Frame rate (freerun mode)	fps	6.31 ^{*2)}	14.1 ^{*2)}	15.1 ^{*2)}
Frame rate (trigger mode, 1 ms exposure)	fps	6.38 ^{*2)}	14.1 ^{*2)}	15.2 ^{*2)}
Exposure time in freerun mode	ms	0.075 ^{*2)} - 2745 ^{*3)}	0.034 ^{*2)} - 3404 ^{*3)}	0.031 ^{*2)} -3404 ^{*3)}
Exposure time in trigger mode	ms	0.075 ^{*2)} - 2745 ^{*3)}	0.034 ^{*2)} - 3404 ^{*3)}	0.031 ^{*2)} -3404 ^{*3)}
AOI				
Mode		Horizontal ^{*4)} + Vertical ^{*4)}		
AOI image width, step width	Pixels	32 - 2560, 4	32 - 2560, 4	32 - 2560, 4
AOI image height, step width	Pixels	4 - 1920, 2	4 - 1920, 2	4 - 1920, 2
AOI position grid horizontal, vertical	Pixels	4, 2	4, 2	4, 2
AOI frame rate, 1920 x 1080 pixels (HD 1080)	fps	13.5	30.1	32.3
Binning				
Mode		Horizontal ^{*4)} + Vertical ^{*4)}		
Method		H + V: Color binning. H: additive. V: averaging		
Factor Mono / Color		2x / 2x, 3x, 4x, 6x		
Subsampling				

Mode		Horizontal ^{*4)} + Vertical ^{*4)}		
Method		H + V: Color subsampling		
Factor		2x, 3x, 4x, 5x, 6x		
Frame rate w/ 2x subsampling, 1280 x 960 pixels	fps	25.4	51.3	55.9
Frame rate w/ 4x subsampling, 640 x 480 pixels	fps	100.6	138.5	156.9
Hardware trigger				
Mode		Asynchronous	Asynchronous	Asynchronous
Trigger delay with rising edge	µs	22.0 ±0.25	3.1 ±0.25	0.3 ±0.25
Trigger delay with falling edge	µs	40.2 ±0.25	11.8 ±0.25	0.2 ±0.25
Additive trigger delay (optional)	µs	15 µs...4 s	15 µs...4 s	15 µs...4 s
Sensor delay to exposure start	µs	< 200 ^{*2)}	< 200 ^{*2)}	< 200 ^{*2)}
Power consumption ^{*5)}				
	W	USB uEye	GigE uEye SE	GigE uEye HE
		0.5-0.9	2.6-3.1	3.0-4.4

*1) The maximum possible pixel clock frequency depends on the PC hardware used.

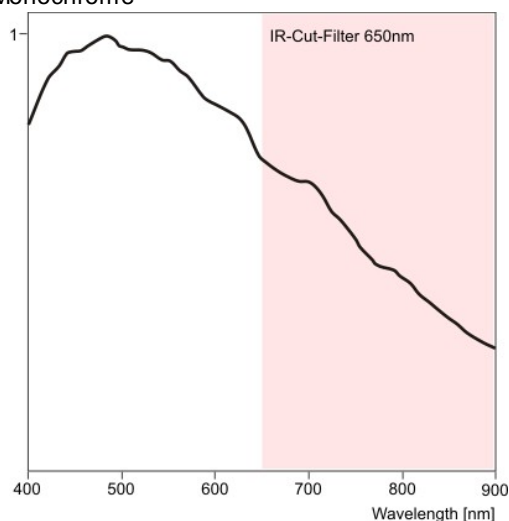
*2) Requires maximum pixel clock frequency.

*3) Requires minimum pixel clock frequency.

*4) Use of this function increases the frame rate.

*5) The power consumption depends on the sensor model and the pixel clock setting.

Relative sensor sensitivity
Monochrome



Color

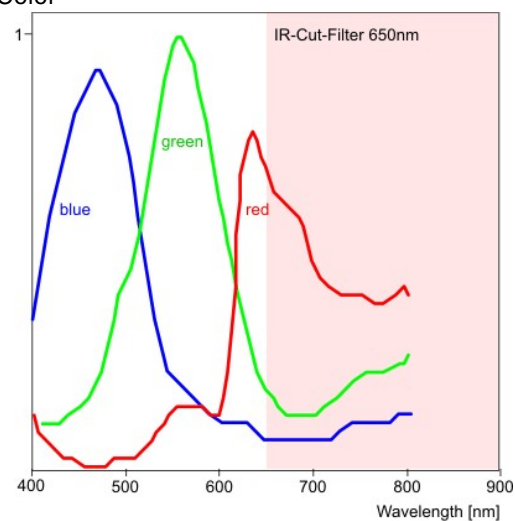


Figure 124: Sensor sensitivity of the UI-148x / UI-548x

**Notes on using the UI-148x / UI-548x**

- It is recommended to use a high-resolution (megapixel) lens.
- Use of the Global Start function slightly reduces the maximum possible frame rate because the sensor needs to be run in trigger mode.
- At very long exposure times and minimum gain, the white level may not be reached. The gain should be increased by one step in this case.
- Color version only: Live color display with color correction and 5x5 de-Bayering results in high CPU load (see [Color Filter \(Bayer Filter\)](#)).
- For hardware reasons, the sensor can only perform 3x vertical binning. When 3x horizontal binning is activated in the uEye software, the driver uses 3x subsampling instead. Therefore, the image will not become brighter when 3x horizontal binning is activated.
- Monochrome version only: No gain boost (factor) available. Use 30x master gain instead.
- Monochrome version: sensor internally works like the color version. This might lead to artefacts when binning and subsampling are used. Therefore, the monochrome sensor does not support binning factors higher than 2x.
- Monochrome version: Gain settings 0...49 use analog signal gain; from 50 up, the stronger digital gain is used. High gain settings may cause visible noise.

6.2.4 UI-149x

Sensor specification				
Sensor type	CMOS			
Shutter system	Electronic rolling shutter / global start shutter			
Readout mode	Progressive scan			
Resolution class	QHDTV			
Resolution	3840 x 2748 pixels (10.55 MP)			
Aspect ratio	4:3			
Bit depth	12 bits			
Optical sensor class	1/2 inch			
Exact sensitive area	6.41 x 4.59 mm			
Exact optical sensor diagonal	7.89 mm (1/2.03 inch)			
Pixel size	1.67 µm, square			
Sensor name, monochrome	Micron MT9J003			
Sensor name, color	Micron MT9J003			
Special features	Sensor internal image scaler, downscaling by factor 1...8			
Gain				
Monochrome model (master gain)	*)			
Color model (master/RGB)	11.2x/5.29x			
Gain boost	-			
Camera timing		USB uEye	GigE uEye SE	GigE uEye HE
Pixel clock range	MHz	5-36 ^{*1)}	5-68 ^{*1)}	5-90 ^{*1)}
Max. pixel clock with subsampling/binning	MHz	36 ^{*1)}	68 ^{*1)}	100 ^{*1)}
Frame rate (freerun mode)	fps	3.20 ^{*2)}	6.05 ^{*2)}	8.00 ^{*2)}
Frame rate (trigger mode, 1 ms exposure)	fps	3.19 ^{*2)}	6.00 ^{*2)}	7.92 ^{*2)}
Exposure time in freerun mode	ms	0.340 ^{*2)} - 14582 ^{*3)}	0.180 ^{*2)} - 14582 ^{*3)}	0.136 ^{*2)} - 14582 ^{*3)}
Exposure time in trigger mode	ms	0.340 ^{*2)} - 14582 ^{*3)}	0.180 ^{*2)} - 14582 ^{*3)}	0.136 ^{*2)} - 14582 ^{*3)}
AOI				
Mode		Horizontal ^{*4)} + Vertical ^{*4)}		
Minimum AOI	Pixels	448 x 4		
Position grid	Pixels	H: 4, V: 2		
AOI frame rate, 2560 x 1920 (4.92 MPixel)	fps	6.7	11.9	11.9
Binning				
Mode		Horizontal ^{*4)} + Vertical ^{*4)}		
Method		H + V: Color binning, additive		
Factors		2x, color model: 4x		
Frame rate with 2x binning, 1920 x 1080 pixels	fps	14.9	28.2	37.0

Color model: Frame rate with 4x binning, 800 x 600 pixels	fps	47.4	89.0	118.0
Subsampling				
Mode		Horizontal ^{*4)} + Vertical ^{*4)}		
Method		H + V: Color subsampling		
Factors		2x, 4x		
Hardware trigger				
Mode		Asynchronous	Asynchronous	Asynchronous
Rolling Shutter mode (no Global Start)				
Trigger delay with rising edge	μs	180.8 ±0.25	234.8 ±0.25	232.0 ±0.25
Trigger delay with falling edge	μs	199.1 ±0.25	244.1 ±0.25	232.1 ±0.25
Global Start mode				
Trigger delay with rising edge	μs	16.9 ±0.25	43.1 ±0.25	40.3 ±0.25
Trigger delay with falling edge	μs	35.3 ±0.25	51.9 ±0.25	40.2 ±0.25
Additive trigger delay (optional)	μs	15 μs...4 s	15 μs...4 s	15 μs...4 s
Sensor delay to exposure start	μs	< 200 ^{*2)}	< 200 ^{*2)}	< 200 ^{*2)}
Power consumption ^{*5)}		USB uEye	GigE uEye SE	GigE uEye HE
	W	0.5-1.3	2.4-3.3	3.0-5.0

^{*}) To be defined

^{*1)} The maximum possible pixel clock frequency depends on the PC hardware used.

^{*2)} Requires maximum pixel clock frequency.

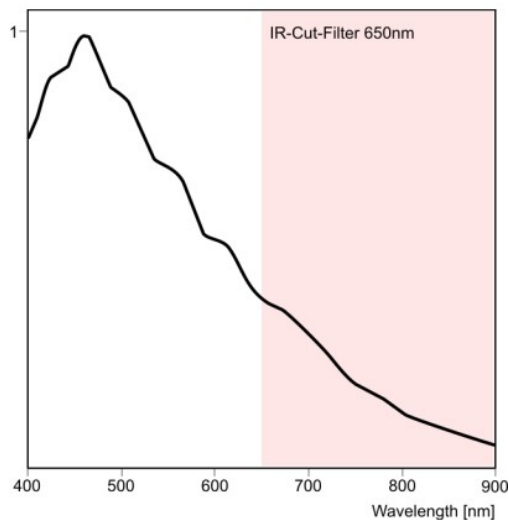
^{*3)} Requires minimum pixel clock frequency.

^{*4)} Use of this function increases the frame rate.

^{*5)} The power consumption depends on the sensor model and the pixel clock setting.

Relative sensor sensitivity

Monochrome



Color

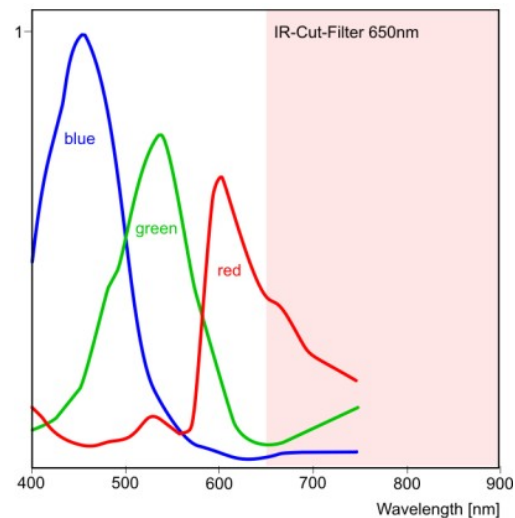


Figure 125: Sensor sensitivity of the UI-149x / UI-549x



Notes on using the UI-149x / UI-549x

- It is recommended to use a high-resolution (megapixel) lens.
- Use of the sensor's internal image scaler does not increase the maximum possible frame rate.
- It is recommended to always enable hot pixel correction.
- Use of the Global Start function slightly reduces the maximum possible frame rate because the sensor needs to be run in trigger mode.
- At very long exposure times and minimum gain, the white level may not be reached. The gain should be increased by one step in this case.
- Color version only: Live color display with color correction and 5x5 de-Bayering results in high CPU load (see [Color Filter \(Bayer Filter\)](#)).
- Monochrome version: The sensor has a reduced sensitivity in the NIR range. It is suitable for IR imaging only to a limited extent.
- Monochrome version: The sensor internally works like the color version. This might lead to artefacts when binning and subsampling are used. Therefore, the monochrome sensor does not support binning factors higher than 2x.

6.2.5 UI-154x / UI-554x

Sensor specification				
Sensor type	CMOS			
Shutter system	Electronic rolling shutter			
Readout mode	Progressive scan			
Resolution class	SXGA			
Resolution	1280 x 1024 pixels (1.3 Mpixels)			
Aspect ratio	5:4			
Bit depth	10 bits			
Optical sensor class	1/2 inch			
Exact sensitive area	6.66 x 5.32 mm			
Exact optical sensor diagonal	8.5 mm (1/1.9 inch)			
Pixel size	5.2 μm, square			
Sensor name, monochrome	Micron MT9M001			
Sensor name, color	-			
Gain				
Monochrome model (master gain)	13x			
Gain boost	1.5x			
Camera timing		USB uEye	GigE uEye SE	GigE uEye HE
Pixel clock range	MHz	5-43 ^{*1)}	2-61 ^{*1)}	2-61 ^{*1)}
Max. pixel clock with subsampling/binning	MHz	50 ^{*1)}	61 ^{*1)}	61 ^{*1)}
Frame rate (freerun mode)	fps	25.0 ^{*2)}	35.5 ^{*2)}	35.5 ^{*2)}
Frame rate (trigger mode, 1 ms exposure)	fps	25.0 ^{*2)}	35.2 ^{*2)}	35.2 ^{*2)}
Exposure time in freerun mode	ms	0.037 ^{*2)} -983 ^{*3)}	0.026 ^{*2)} -2459 ^{*3)}	0.026 ^{*2)} -2459 ^{*3)}
Exposure time in trigger mode	ms	0.037 ^{*2)} -983 ^{*3)}	0.026 ^{*2)} -2459 ^{*3)}	0.026 ^{*2)} -2459 ^{*3)}
AOI				
Mode		Horizontal ^{*4)} + Vertical ^{*4)}		
AOI image width, step width	Pixels	32 - 1280, 4	32 - 1280, 4	32 - 1280, 4
AOI image height, step width	Pixels	4 - 1024, 2	4 - 1024, 2	4 - 1024, 2
AOI position grid horizontal, vertical	Pixels	4, 2	4, 2	4, 2
AOI frame rate, 640 x 480 pixels (VGA)	fps	84	119	119
Binning				
Mode		-	-	-
Subsampling				
Mode		Horizontal ^{*4)} + Vertical ^{*4)}		
Method		H + V: Color subsampling		
Factor		2x, 4x, 8x		
Frame rate w/ 2x subsampling, 640 x 512	fps	98	119	119

pixels				
Frame rate w/ 4x subsampling, 320 x 256 pixels	fps	269	328	328
Hardware trigger				
Mode		Asynchronous	Asynchronous	Asynchronous
Trigger delay with rising edge	µs	22.0 ±0.25	3.1 ±0.25	0.3 ±0.25
Trigger delay with falling edge	µs	40.3 ±0.25	11.9 ±0.25	0.2 ±0.25
Additive trigger delay (optional)	µs	15 µs...4 s	15 µs...4 s	15 µs...4 s
Sensor delay to exposure start	µs	< 200 ^{*2)}	< 200 ^{*2)}	< 200 ^{*2)}
Power consumption ^{*5)}				
	W	USB uEye	GigE uEye SE	GigE uEye HE
		0.5-1.0	2.6-3.2	3.0-4.3

*1) The maximum possible pixel clock frequency depends on the PC hardware used.

*2) Requires maximum pixel clock frequency.

*3) Requires minimum pixel clock frequency.

*4) Use of this function increases the frame rate.

*5) The power consumption depends on the sensor model and the pixel clock setting.

Relative sensor sensitivity
Monochrome

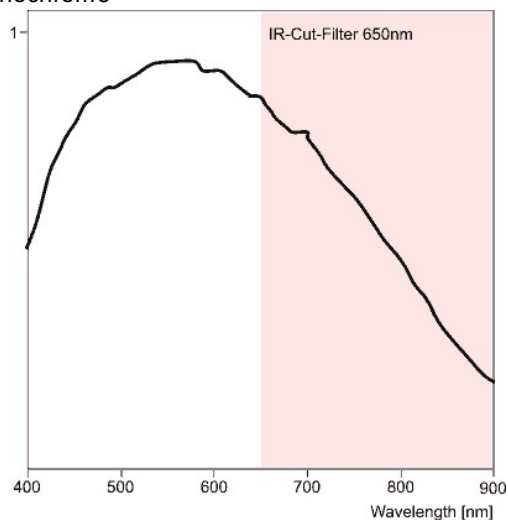


Figure 126: Sensor sensitivity of the UI-154x / UI-554x



Notes on using the UI-154x / UI-554x

- Sensor speed does not increase for AOI width <240 pixels.
- Extreme overexposure shifts the black level. Please deactivate the Auto Offset function in this case.
- At very long exposure times and minimum gain, the white level may not be reached. The gain should be increased by one step in this case.
- Monochrome version only: The sensor internally works like the color version. This might lead to artefacts when subsampling is used.
- The brightness of the first and last line might deviate due to the sensor.
- Gain values between 59 and 99 may lead to image inhomogeneity.
- Cameras with a date of manufacture after Dec. 9, 2008: The offset control has been calibrated internally. The calibration corrects offset errors when gain is used. In calibrated cameras, automatic black level correction is disabled by default. The calibration can only be used with uEye driver version 3.31 or higher.
- Cameras with a date of manufacture before Dec. 9, 2008: If manual offset control is used, fixed pattern noise and horizontal lines may become visible. High gain values may shift the black level and therefore should be avoided.
Offset increases the black level every 7th step. The steps in-between change the appearance of fixed pattern noise.

6.2.6 UI-155x / UI-555x

Sensor specification				
Sensor type	CMOS			
Shutter system	Electronic rolling shutter			
Readout mode	Progressive scan			
Resolution class	UXGA			
Resolution	1600 x 1200 pixels (1.92 Mpixels)			
Aspect ratio	4:3			
Bit depth	10 bits			
Optical sensor class	1/3 inch			
Exact sensitive area	4.48 x 3.36 mm			
Exact optical sensor diagonal	5.6 mm (1/2.9 inch)			
Pixel size	2.8 µm, square			
Sensor name, monochrome	-			
Sensor name, color	Micron MT9D131			
Gain				
Color model (master/RGB)	3.5x/3.1x			
Gain boost	2.0x			
Camera timing		USB uEye	GigE uEye SE	GigE uEye HE
Pixel clock range	MHz	3-56 ^{*1)}	3-60 ^{*1)}	3-60 ^{*1)}
Max. pixel clock with subsampling/binning	MHz	48 ^{*1)}	56 ^{*1)}	60 ^{*1)}
Frame rate (freerun mode)	fps	18.3 ^{*2)}	23.8 ^{*2)}	25.5 ^{*2)}
Frame rate (trigger mode, 1 ms exposure)	fps	18.2 ^{*2)}	23.5 ^{*2)}	25.5 ^{*2)}
Exposure time in freerun mode	ms	0.038 ^{*2)} - 12.8s ^{*3)}	0.029 ^{*2)} - 21.4s ^{*3)}	0.029 ^{*2)} - 21.4s ^{*3)}
Exposure time in trigger mode	ms	0.038 ^{*2)} - 12.8s ^{*3)}	0.029 ^{*2)} - 21.4s ^{*3)}	0.027 ^{*2)} - 21.4s ^{*3)}
AOI				
Mode		Horizontal ^{*4)} + Vertical ^{*4)}		
AOI image width, step width	Pixels	32 - 1600, 4	32 - 1600, 4	32 - 1600, 4
AOI image height, step width	Pixels	4 - 1200, 2	4 - 1200, 2	4 - 1200, 2
AOI position grid horizontal, vertical	Pixels	4, 2	4, 2	4, 2
AOI frame rate, 1280 x 720 pixels (HD 720)	fps	35.7	46.6	49.95
Binning				
Mode		Horizontal ^{*4)} + Vertical ^{*4)}		
Method		H + V: Color binning, averaging		
Factor		2x		
Subsampling				
Mode		Horizontal ^{*4)} + Vertical ^{*4)}		

Method		H + V: Color subsampling		
Factor		2x, 4x, 8x, 16x		
Frame rate w/ 2x subsampling, 800 x 600 pixels	fps	71	79,6	85
Frame rate w/ 4x subsampling, 400 x 300 pixels	fps	252	235	252
Hardware trigger				
Mode		Asynchronous	Asynchronous	Asynchronous
Trigger delay with rising edge	µs	180.9 ±0.25	233.1 ±0.25	229.6 ±0.25
Trigger delay with falling edge	µs	199.3 ±0.25	242.8 ±0.25	230.0 ±0.25
Additive trigger delay (optional)	µs	15 µs...4 s	15 µs...4 s	15 µs...4 s
Sensor delay to exposure start	µs	< 200 ^{*2)}	< 200 ^{*2)}	< 200 ^{*2)}
Power consumption ^{*5)}		USB uEye	GigE uEye SE	GigE uEye HE
	W	0.5-1.1	2.4-3.4	3.0-4.3

*1) The maximum possible pixel clock frequency depends on the PC hardware used.

*2) Requires maximum pixel clock frequency.

*3) Requires minimum pixel clock frequency.

*4) Use of this function increases the frame rate.

*5) The power consumption depends on the sensor model and the pixel clock setting.

Relative sensor sensitivity

Color

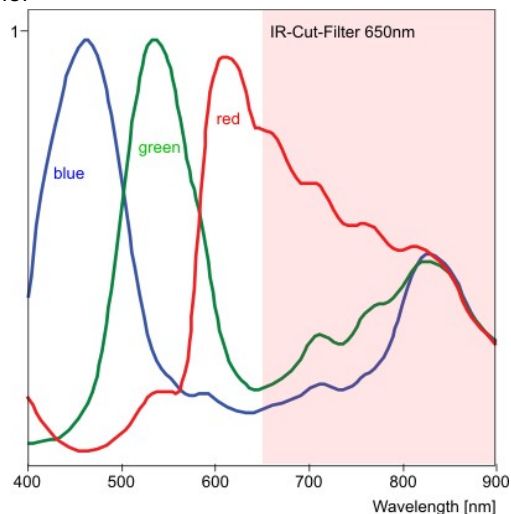


Figure 127: Sensor sensitivity of the UI-155x / UI-555x



Notes on using the UI-155x/ UI-555x

- At very long exposure times and minimum gain, the white level may not be reached. Increase the gain by one step.
- For AOI width <160 pixels, the sensor gets slower.
- Horizontal and vertical binning can only be used together.
- The sensor binning works by averaging pixels, so the image will not become brighter when binning is activated.
- When you are using a master gain setting of 72 or higher, we recommend the following RGB gain settings to avoid color errors: Set at least one RGB gain to 0. Set each of the other two RGB gains to a value less than 50.
- Homogeneous images may show color aberrations in the corner areas. This is caused by the way the micro lenses are placed on this sensor. The effect can be minimized by using a large aperture on the lens (small F number).

6.2.7 UI-164x / UI-564x

Sensor specification				
Sensor type	CMOS			
Shutter system	Electronic rolling shutter			
Readout mode	Progressive scan			
Resolution class	SXGA			
Resolution	1280 x 1024 pixels (1.3 Mpixels)			
Aspect ratio	5:4			
Bit depth	10 bits			
Optical sensor class	1/3 inch			
Exact sensitive area	4.61 x 3.69 mm			
Exact optical sensor diagonal	5.9 mm (1/2.7 inch)			
Pixel size	3.6 µm, square			
Sensor name, monochrome	-			
Sensor name, color	Micron MT9M131			
Gain				
Color model (master/RGB)	3.0x/3.1x			
Gain boost	2.0x			
Camera timing		USB uEye	GigE uEye SE	GigE uEye HE
Pixel clock range	MHz	5-40 ^{*1)}	5-40 ^{*1)}	5-40 ^{*1)}
Max. pixel clock with subsampling/binning	MHz	5-40 ^{*1)}	5-40 ^{*1)}	5-40 ^{*1)}
Frame rate (freerun mode)	fps	25.0 ^{*2)}	25.0 ^{*2)}	25.0 ^{*2)}
Frame rate (trigger mode, 1 ms exposure)	fps	24.9 ^{*2)}	24.9 ^{*2)}	25.0 ^{*2)}
Exposure time in freerun mode	ms	0.037 ^{*2)} - 10.1s ^{*3)}	0.038 ^{*2)} - 10.1s ^{*3)}	0.037 ^{*2)} - 10.1s ^{*3)}
Exposure time in trigger mode	ms	0.037 ^{*2)} - 10.1s ^{*3)}	0.038 ^{*2)} - 10.1s ^{*3)}	0.037 ^{*2)} - 10.1s ^{*3)}
AOI				
Mode		Horizontal ^{*4)} + Vertical ^{*4)}		
AOI image width, step width	Pixels	32 - 1280, 4	32 - 1280, 4	32 - 1280, 4
AOI image height, step width	Pixels	4 - 1024, 2	4 - 1024, 2	4 - 1024, 2
AOI position grid horizontal, vertical	Pixels	4, 2	4, 2	4, 2
AOI frame rate, 1280 x 720 pixels (HD 720)	fps	34	34	34
AOI frame rate, 800 x 600 pixels (SVGA)	fps	61	61	61
Binning				
Mode		-	-	-
Subsampling				
Mode		Horizontal ^{*4)} + Vertical ^{*4)}		
Method		H + V: Color subsampling		

Factor		2x, 4x		
Frame rate w/ 2x subsampling, 640 x 480 pixels	fps	89	89	89
Frame rate w/ 4x subsampling, 320 x 240 pixels	fps	263	263	263
Hardware trigger				
Mode		Asynchronous	Asynchronous	Asynchronous
Trigger delay with rising edge	µs	180.9 ±0.25	233.8 ±0.25	230.3 ±0.25
Trigger delay with falling edge	µs	199.3 ±0.25	242.2 ±0.25	231.5 ±0.25
Additive trigger delay (optional)	µs	15 µs...4 s	15 µs...4 s	15 µs...4 s
Sensor delay to exposure start	µs	< 200 ^{*2)}	< 200 ^{*2)}	< 200 ^{*2)}
Power consumption ^{*5)}				
	W	USB uEye	GigE uEye SE	GigE uEye HE
		0.3-0.8	2.3-3.2	2.7-4.1

*1) The maximum possible pixel clock frequency depends on the PC hardware used.

*2) Requires maximum pixel clock frequency.

*3) Requires minimum pixel clock frequency.

*4) Use of this function increases the frame rate.

*5) The power consumption depends on the sensor model and the pixel clock setting.

Relative sensor sensitivity

Color

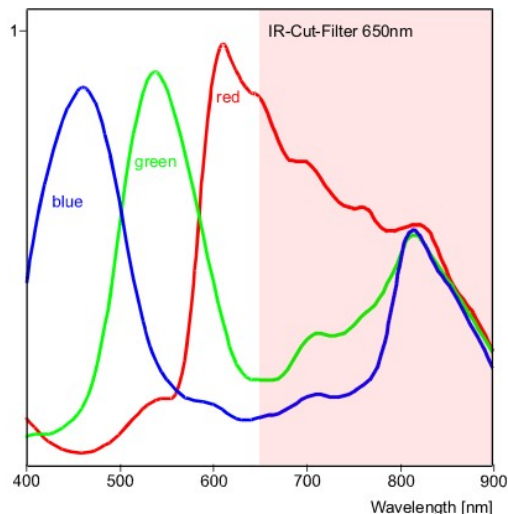


Figure 128: Sensor sensitivity of the UI-164x / UI-564x



Notes on using the UI-164x / UI-564x-X

- At very long exposure times and minimum gain, the white level may not be reached. The gain should be increased by one step in this case.
- The RGB gain controls have no effect for values >90.

6.2.8 UI-221x / UI-621x

Sensor specification				
Sensor type	CCD			
Shutter system	Electronic global shutter			
Readout mode	Progressive scan			
Resolution class	VGA			
Resolution	640 x 480 pixels (0.31 Mpixels)			
Aspect ratio	4:3			
Bit depth	12 bits			
Optical sensor class	1/2 inch			
Exact sensitive area	6.34 x 4.75 mm			
Exact optical sensor diagonal	7.9 mm (1/2.0 inch)			
Pixel size	9.9 μm, square			
Sensor name, monochrome	Sony ICX414AL			
Sensor name, color	Sony ICX414AQ			
Gain				
Monochrome model (master gain)	20.78x			
Color model (master/RGB)	12.0x/4.0x			
Gain boost	2.0x (monochrome model only)			
Camera timing		USB uEye	GigE uEye SE	GigE uEye HE
Pixel clock range	MHz	5-30 ^{*1)}	5-30 ^{*1)}	5-30 ^{*1)}
Max. pixel clock with subsampling/binning	MHz	30 ^{*1)}	30 ^{*1)}	30 ^{*1)}
Frame rate (freerun mode)	fps	75.3 ^{*2)}	75.3 ^{*2)}	75.3 ^{*2)}
Frame rate (trigger mode, 1 ms exposure)	fps	69.0 ^{*2)}	69.0 ^{*2)}	69.0 ^{*2)}
Exposure time in freerun mode	ms	0.04 ^{*2)} -630 ^{*3)}	0.04 ^{*2)} -630 ^{*3)}	0.04 ^{*2)} -630 ^{*3)}
Exposure time in trigger mode	ms	0.04 ^{*2)} -10 min ^{*3)}	0.04 ^{*2)} -10 min ^{*3)}	0.04 ^{*2)} -10 min ^{*3)}
AOI				
Mode		Horizontal + Vertical ^{*4)}		
AOI image width, step width	Pixels	16 - 640, 4	16 - 640, 4	16 - 640, 4
Mono: AOI image height, step width	Pixels	120 - 480, 1	120 - 480, 1	120 - 480, 1
Color: AOI image height, step width	Pixels	120 - 480, 2	120 - 480, 2	120 - 480, 2
Mono: AOI position grid horizontal, vertical	Pixels	1, 1	1, 1	1, 1
Color: AOI position grid horizontal, vertical	Pixels	2, 2	2, 2	2, 2
AOI frame rate, 320 x 240 pixels (CIF)	fps	122	122	122
Binning				
Mode		Vertical ^{*4)}	Horizontal + Vertical ^{*4)}	Horizontal + Vertical ^{*4)}
Method		V: Monochrome binning, additive	H + V: Monochrome	H + V: Monochrome

			binning, additive	binning, additive
Frame rate with 2x binning, 640 x 240 pixels	fps	135	135	135
Frame rate with 3x binning, 640 x 160 pixels	fps	183	183	183
Frame rate with 4x binning, 640 x 120 pixels	fps	221	221	221
Subsampling				
Mode		-	-	-
Hardware trigger				
Mode		Asynchronous	Asynchronous	Asynchronous
Trigger delay with rising edge	µs	39.6 ±2.5	28.1 ±19.0	22.4 ±19.0
Trigger delay with falling edge	µs	58.1 ±2.5	39.9 ±19.0	21.7 ±19.0
Additive trigger delay (optional)	µs	15 µs...4 s	15 µs...4 s	15 µs...4 s
Sensor delay to exposure start	µs	< 100 ^{*2)}	< 100 ^{*2)}	< 100 ^{*2)}
Power consumption ^{*5)}				
	W	USB uEye	GigE uEye SE	GigE uEye HE
		1.0-1.9	3.1-4.1	3.4-5.2

*1) The maximum possible pixel clock frequency depends on the PC hardware used.

*2) Requires maximum pixel clock frequency.

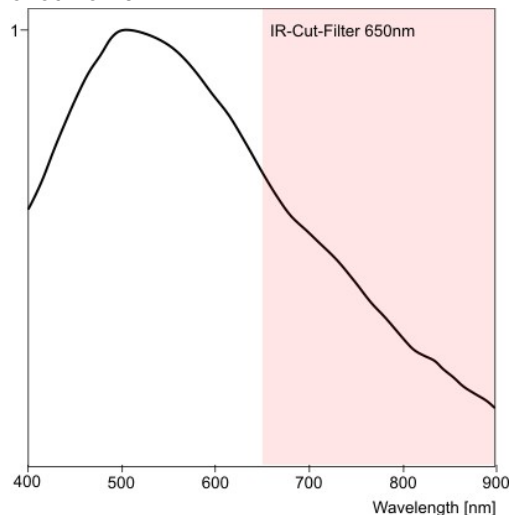
*3) Requires minimum pixel clock frequency.

*4) Use of this function increases the frame rate for monochrome models.

*5) The power consumption depends on the sensor model and the pixel clock setting.

Relative sensor sensitivity

Monochrome



Color

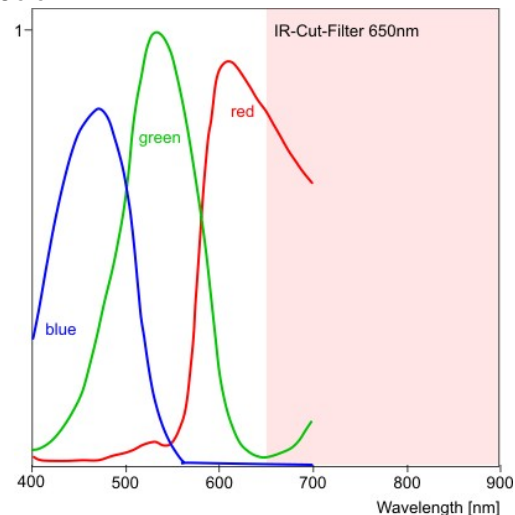


Figure 129: Sensor sensitivity of the UI-221x / UI-621x



Notes on using the UI-221x / UI-621x

- Optimum pixel clock frequency is 24 MHz.
- Recommended pixel clock range 16 - 26 MHz.

6.2.9 UI-222x / UI-622x

Sensor specification				
Sensor type	CCD			
Shutter system	Electronic global shutter			
Readout mode	Progressive scan			
Resolution class	CCIR/PAL			
Resolution	768 x 576 pixels (0.44 Mpixels)			
Aspect ratio	4:3			
Bit depth	12 bits			
Optical sensor class	1/2 inch			
Exact sensitive area	6.37 x 4.78 mm			
Exact optical sensor diagonal	8.0 mm (1/2.0 inch)			
Pixel size	8.3 μm, square			
Sensor name, monochrome	Sony ICX415AL			
Sensor name, color	Sony ICX415AQ			
Gain				
Monochrome model (master gain)	14.1x			
Color model (master/RGB)	8.9x/4.0x			
Gain boost	2.0x (monochrome model only)			
Camera timing		USB uEye	GigE uEye SE	GigE uEye HE
Pixel clock range	MHz	5-30 ^{*1)}	5-30 ^{*1)}	5-30 ^{*1)}
Max. pixel clock with subsampling/binning	MHz	30 ^{*1)}	30 ^{*1)}	30 ^{*1)}
Frame rate (freerun mode)	fps	52.2 ^{*2)}	52.2 ^{*2)}	52.2 ^{*2)}
Frame rate (trigger mode, 1 ms exposure)	fps	49.3 ^{*2)}	49.3 ^{*2)}	49.3 ^{*2)}
Exposure time in freerun mode	ms	0.05 ^{*2)} -770 ^{*3)}	0.05 ^{*2)} -770 ^{*3)}	0.05 ^{*2)} -770 ^{*3)}
Exposure time in trigger mode	ms	0.05 ^{*2)} -10 min ^{*3)}	0.05 ^{*2)} -10 min ^{*3)}	0.05 ^{*2)} -10 min ^{*3)}
AOI				
Mode		Horizontal + Vertical ^{*4)}		
AOI image width, step width	Pixels	16 - 768, 4	16 - 768, 4	16 - 768, 4
Mono: AOI image height, step width	Pixels	120 - 576, 1	120 - 576, 1	120 - 576, 1
Color: AOI image height, step width	Pixels	120 - 576, 2	120 - 576, 2	120 - 576, 2
Mono: AOI position grid horizontal, vertical	Pixels	1, 1	1, 1	1, 1
Color: AOI position grid horizontal, vertical	Pixels	2, 2	2, 2	2, 2
AOI frame rate, 640 x 480 pixels (VGA)	fps	60	60	60
AOI frame rate, 320 x 240 pixels (CIF)	fps	97	97	97
Binning				
Mode		Vertical ^{*4)}	Horizontal + Vertical ^{*4)}	Horizontal + Vertical ^{*4)}

Method		V: Monochrome binning, additive	H + V: Monochrome binning, additive	H + V: Monochrome binning, additive
Frame rate with 2x binning, 768 x 288 pixels	fps	90	90	90
Frame rate with 3x binning, 768 x 192 pixels	fps	121	121	121
Frame rate with 4x binning, 768 x 144 pixels	fps	143	143	143
Subsampling				
Mode		-	-	-
Hardware trigger				
Mode		Asynchronous	Asynchronous	Asynchronous
Trigger delay with rising edge	µs	40.5 ±2.5	28.3 ±19.0	22.2 ±19.0
Trigger delay with falling edge	µs	57.5 ±2.5	34.7 ±19.0	21.1 ±19.0
Additive trigger delay (optional)	µs	15 µs...4 s	15 µs...4 s	15 µs...4 s
Sensor delay to exposure start	µs	< 100 ^{*2)}	< 100 ^{*2)}	< 100 ^{*2)}
Power consumption ^{*5)}				
	W	1.0-1.9	3.0-4.0	3.5-5.4

*1) The maximum possible pixel clock frequency depends on the PC hardware used.

*2) Requires maximum pixel clock frequency.

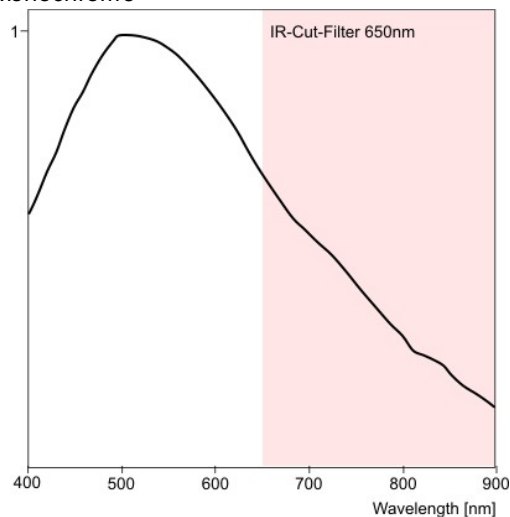
*3) Requires minimum pixel clock frequency.

*4) Use of the function increases the frame rate for monochrome models.

*5) The power consumption depends on the sensor model and the pixel clock setting.

Relative sensor sensitivity

Monochrome



Color

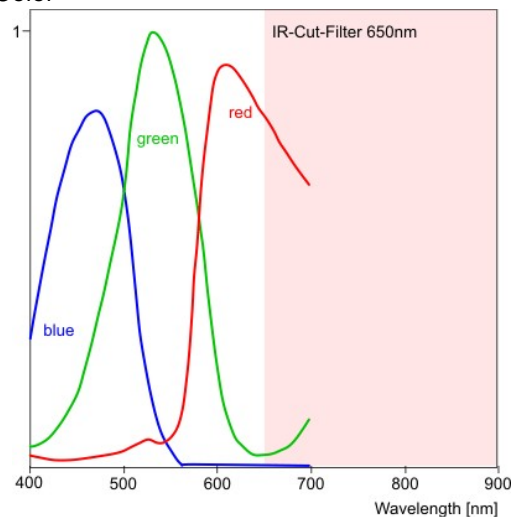


Figure 130: Sensor sensitivity of the UI-222x / UI-622x



Notes on using the UI-222x / UI-622x

- Optimum pixel clock frequency is 28 MHz.

6.2.10 UI-223x / UI-623x

Sensor specification				
Sensor type	CCD			
Shutter system	Electronic global shutter			
Readout mode	Progressive scan			
Resolution class	XGA			
Resolution	1024 x 768 pixels (0.79 Mpixels)			
Aspect ratio	4:3			
Bit depth	12 bits			
Optical sensor class	1/3 inch			
Exact sensitive area	4.76 x 3.57 mm			
Exact optical sensor diagonal	6.0 mm (1/2.7 inch)			
Pixel size	4.65 μm, square			
Sensor name, monochrome	Sony ICX204AL			
Sensor name, color	Sony ICX204AK			
Gain				
Monochrome model (master gain)	10.47x			
Color model (master/RGB)	7.59x/4.0x			
Gain boost	2.0x (monochrome model only)			
Camera timing		USB uEye	GigE uEye SE	GigE uEye HE
Pixel clock range	MHz	5-30 ^{*1)}	5-30 ^{*1)}	5-30 ^{*1)}
Max. pixel clock with subsampling/binning	MHz	30 ^{*1)}	30 ^{*1)}	30 ^{*1)}
Frame rate (freerun mode)	fps	30.0 ^{*2)}	30.0 ^{*2)}	30.0 ^{*2)}
Frame rate (trigger mode, 1 ms exposure)	fps	28.7 ^{*2)}	28.7 ^{*2)}	28.7 ^{*2)}
Exposure time in freerun mode	ms	0.066 ^{*2)} -1000 ^{*3)}	0.066 ^{*2)} -1000 ^{*3)}	0.066 ^{*2)} -1000 ^{*3)}
Exposure time in trigger mode	ms	0.066 ^{*2)} -10 min ^{*3)}	0.066 ^{*2)} -10 min ^{*3)}	0.066 ^{*2)} -10 min ^{*3)}
AOI				
Mode		Horizontal + Vertical ^{*4)}		
AOI image width, step width	Pixels	16 - 1024, 4	16 - 1024, 4	16 - 1024, 4
Mono: AOI image height, step width	Pixels	120 - 768, 1	120 - 768, 1	120 - 768, 1
Color: AOI image height, step width	Pixels	120 - 768, 2	120 - 768, 2	120 - 768, 2
Mono: AOI position grid horizontal, vertical	Pixels	1, 1	1, 1	1, 1
Color: AOI position grid horizontal, vertical	Pixels	2, 2	2, 2	2, 2
AOI frame rate, 800 x 600 pixels (SVGA)	fps	37	37	37
AOI frame rate, 640 x 480 pixels (VGA)	fps	45	45	45
AOI frame rate, 320 x 240 pixels (CIF)	fps	78	78	78
Binning				
Mode		Vertical ^{*4)}	Horizontal +	Horizontal +

			Vertical ^{*4)}	Vertical ^{*4)}
Method		V: Monochrome binning, additive	H + V: Monochrome binning, additive	H + V: Monochrome binning, additive
Frame rate with 2x binning, 1024 x 384 pixels	fps	53	53	53
Frame rate with 3x binning, 1024 x 256 pixels	fps	71	71	71
Frame rate with 4x binning, 1024 x 192 pixels	fps	85	85	85
Subsampling				
Mode		-	-	-
Hardware trigger				
Mode		Asynchronous	Asynchronous	Asynchronous
Trigger delay with rising edge	µs	39.5 ±2.6	25.5 ±19.0	22.6 ±19.0
Trigger delay with falling edge	µs	57.9 ±2.6	38.6 ±19.0	22.1 ±19.0
Additive trigger delay (optional)	µs	15 µs...4 s	15 µs...4 s	15 µs...4 s
Sensor delay to exposure start	µs	< 100 ^{*2)}	< 100 ^{*2)}	< 100 ^{*2)}
Power consumption ^{*5)}				
	W	1.0-1.7	3.0-3.8	3.5-5.0

*1) The maximum possible pixel clock frequency depends on the PC hardware used.

*2) Requires maximum pixel clock frequency.

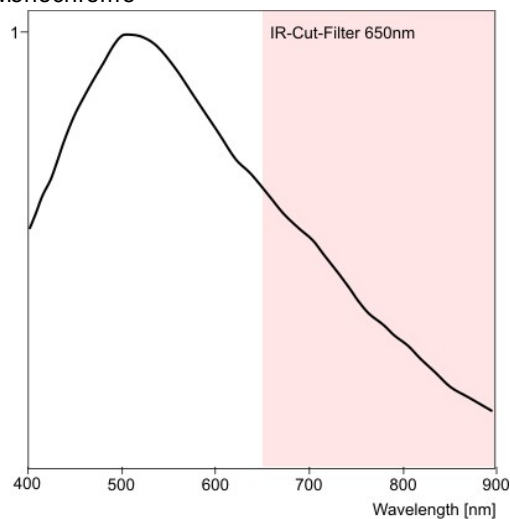
*3) Requires minimum pixel clock frequency.

*4) Use of the function increases the frame rate for monochrome models.

*5) The power consumption depends on the sensor model and the pixel clock setting.

Relative sensor sensitivity

Monochrome



Color

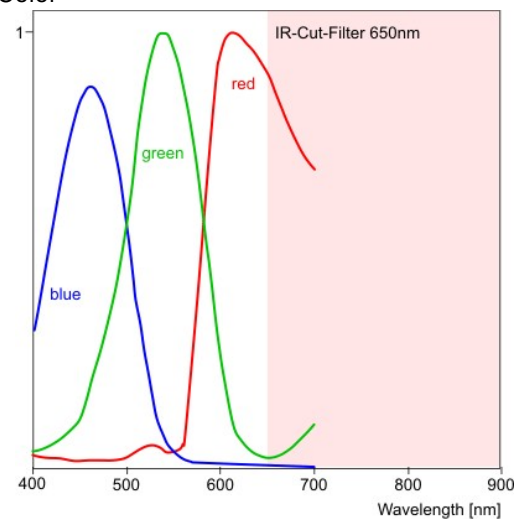


Figure 131: Sensor sensitivity of the UI-223x / UI-623x



Notes on using the UI-223x/ UI-623x

- Optimum pixel clock frequency is 15 MHz.
- Recommended pixel clock range 10 - 20 MHz.
- Long exposure times will increase the number of hotpixels.
- High temperatures will increase the black level of individual pixels.

6.2.11 UI-224x / UI-624x

Sensor specification				
Sensor type	CCD			
Shutter system	Electronic global shutter			
Readout mode	Progressive scan			
Resolution class	SXGA			
Resolution	1280 x 1024 pixels (1.3 Mpixels)			
Aspect ratio	4:3			
Bit depth	12 bits			
Optical sensor class	1/2 inch			
Exact sensitive area	5.95 x 4.76 mm			
Exact optical sensor diagonal	7.6 mm (1/2.1 inch)			
Pixel size	4.65 μm, square			
Sensor name, monochrome	Sony ICX205AL			
Sensor name, color	Sony ICX205AK			
Gain				
Monochrome model (master gain)	13.66x			
Color model (master/RGB)	4.0x/8.9x			
Gain boost	2.0x (monochrome model only)			
Camera timing		USB uEye	GigE uEye SE	GigE uEye HE
Pixel clock range	MHz	5-30 ^{*1)}	5-30 ^{*1)}	5-30 ^{*1)}
Max. pixel clock with subsampling/binning	MHz	30 ^{*1)}	30 ^{*1)}	30 ^{*1)}
Frame rate (freerun mode)	fps	15.0 ^{*2)}	15.0 ^{*2)}	15.0 ^{*2)}
Frame rate (trigger mode, 1 ms exposure)	fps	17.0 ^{*2)}	17.0 ^{*2)}	17.0 ^{*2)}
Exposure time in freerun mode	ms	0.083 ^{*2)} -1460 ^{*3)}	0.083 ^{*2)} -1460 ^{*3)}	0.083 ^{*2)} -1460 ^{*3)}
Exposure time in trigger mode	ms	0.083 ^{*2)} -10 min ^{*3)}	0.083 ^{*2)} -10 min ^{*3)}	0.083 ^{*2)} -10 min ^{*3)}
AOI				
Mode		Horizontal + Vertical ^{*4)}		
AOI image width, step width	Pixels	16 - 1280, 4	16 - 1280, 4	16 - 1280, 4
Mono: AOI image height, step width	Pixels	120 - 1024, 1	120 - 1024, 1	120 - 1024, 1
Color: AOI image height, step width	Pixels	120 - 1024, 2	120 - 1024, 2	120 - 1024, 2
Mono: AOI position grid horizontal, vertical	Pixels	1, 1	1, 1	1, 1
Color: AOI position grid horizontal, vertical	Pixels	2, 2	2, 2	2, 2
AOI frame rate, 1024 x 768 pixels (XGA)	fps	18	18	18
AOI frame rate, 640 x 480 pixels (VGA)	fps	28	28	28
AOI frame rate, 320 x 240 pixels (CIF)	fps	38	38	38
Binning				
Mode		Vertical ^{*4)}	Horizontal +	Horizontal +

			Vertical ^{*4)}	Vertical ^{*4)}
Method		V: Monochrome binning, additive	H + V: Monochrome binning, additive	H + V: Monochrome binning, additive
Frame rate with 2x binning, 1280 x 512 pixels	fps	23	23	23
Frame rate with 3x binning, 1280 x 340 pixels	fps	28	28	28
Frame rate with 4x binning, 1280 x 256 pixels	fps	31	31	31
Subsampling				
Mode		Vertical ^{*4)}	Vertical ^{*4)}	Horizontal + Vertical ^{*4)}
Method		V: Color subsampling	V: Color subsampling	H + V: Color subsampling
Frame rate w/ 4x subsampling, 1280 x 256 pixels	fps	31	31	31
Hardware trigger				
Mode		Asynchronous	Asynchronous	Asynchronous
Trigger delay with rising edge	µs	39.9 ±2.5	31.3 ±19.0	22.3 ±19.0
Trigger delay with falling edge	µs	57.7 ±2.5	36.2 ±19.0	23.2 ±19.0
Additive trigger delay (optional)	µs	15 µs...4 s	15 µs...4 s	15 µs...4 s
Sensor delay to exposure start	µs	< 100 ^{*2)}	< 100 ^{*2)}	< 100 ^{*2)}
Power consumption ^{*5)}				
	W	USB uEye	GigE uEye SE	GigE uEye HE
		1.1-2.1	3.1-4.2	3.7-5.6

*1) The maximum possible pixel clock frequency depends on the PC hardware used.

*2) Requires maximum pixel clock frequency.

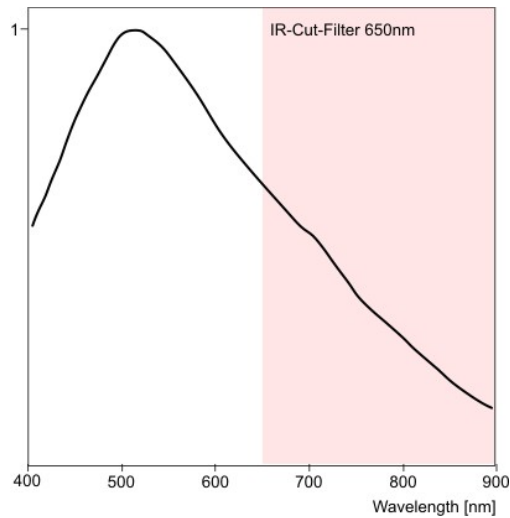
*3) Requires minimum pixel clock frequency.

*4) Use of the function increases the frame rate for monochrome models.

*5) The power consumption depends on the sensor model and the pixel clock setting.

Relative sensor sensitivity

Monochrome



Color

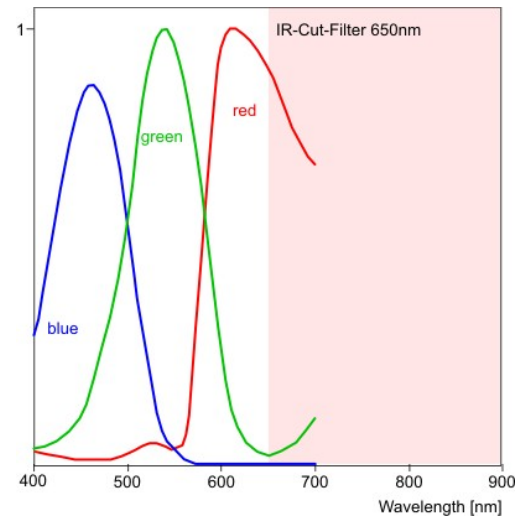


Figure 132: Sensor sensitivity of the UI-224x / UI-624x



Notes on using the UI-224x / UI-624x

- Optimum pixel clock frequency is 14 MHz.
- Recommended pixel clock range 10 - 20 MHz.
- Long exposure times will increase the number of hotpixels.
- High temperatures will increase the black level of individual pixels.
- When vertical 4x binning is activated, the minimum image width increases to 640 pixels.

6.2.12 UI-225x / UI-625x

Sensor specification				
Sensor type	CCD			
Shutter system	Electronic global shutter			
Readout mode	Progressive scan			
Resolution class	UXGA			
Resolution	1600 x 1200 pixels (1.92 Mpixels)			
Aspect ratio	4:3			
Bit depth	12 bits			
Optical sensor class	1/1.8 inch			
Exact sensitive area	7.04 x 5.28 mm			
Exact optical sensor diagonal	8.8 mm (1/1.8 inch)			
Pixel size	4.4 μm, square			
Sensor name, monochrome	Sony ICX274AL			
Sensor name, color	Sony ICX274AK			
Gain				
Monochrome model (master gain)	13.78x			
Color model (master/RGB)	8.9x/4.0x			
Gain boost	2.0x (monochrome model only)			
Camera timing		USB uEye	GigE uEye SE	GigE uEye HE
Pixel clock range	MHz	5-30 ^{*1)}	5-30 ^{*1)}	5-30 ^{*1)}
Max. pixel clock with subsampling/binning	MHz	30 ^{*1)}	30 ^{*1)}	30 ^{*1)}
Frame rate (freerun mode)	fps	12.5 ^{*2)}	12.5 ^{*2)}	12.5 ^{*2)}
Frame rate (trigger mode, 1 ms exposure)	fps	12.3 ^{*2)}	12.3 ^{*2)}	12.3 ^{*2)}
Exposure time in freerun mode	ms	0.094 ^{*2)} -1570 ^{*3)}	0.094 ^{*2)} -1570 ^{*3)}	0.094 ^{*2)} -1570 ^{*3)}
Exposure time in trigger mode	ms	0.094 ^{*2)} -5000 ^{*3)}	0.094 ^{*2)} -5000 ^{*3)}	0.094 ^{*2)} -5000 ^{*3)}
AOI				
Mode		Horizontal + Vertical ^{*4)}		
AOI image width, step width	Pixels	320 - 1600, 4	320 - 1600, 4	320 - 1600, 4
Mono: AOI image height, step width	Pixels	240 - 1200, 1	240 - 1200, 1	240 - 1200, 1
Color: AOI image height, step width	Pixels	240 - 1200, 2	240 - 1200, 2	240 - 1200, 2
Mono: AOI position grid horizontal, vertical	Pixels	1, 1	1, 1	1, 1
Color: AOI position grid horizontal, vertical	Pixels	2, 2	2, 2	2, 2
AOI frame rate, 1024 x 768 pixels (XGA)	fps	18	18	18
AOI frame rate, 640 x 480 pixels (VGA)	fps	28	28	28
AOI frame rate, 320 x 240 pixels (CIF)	fps	47	47	47
Binning				
Mode		Vertical ^{*4)}	Horizontal + Vertical ^{*4)}	Horizontal + Vertical ^{*4)}

Method		V: Monochrome binning, additive	H + V: Monochrome binning, additive	H + V: Monochrome binning, additive
Frame rate with 2x binning, 1600 x 600 pixels	fps	24	24	24
Frame rate with 3x binning, 1600 x 400 pixels	fps	34	34	34
Frame rate with 4x binning, 1600 x 300 pixels	fps	43	43	43
Subsampling				
Mode		Vertical ^{*4)}	Vertical ^{*4)}	Horizontal + Vertical ^{*4)}
Method		V: Color subsampling	V: Color subsampling	H + V: Color subsampling
Frame rate w/ 2x subsampling, 1600 x 600 pixels	fps	24	24	24
Frame rate w/ 4x subsampling, 1600 x 300 pixels	fps	43	43	43
Hardware trigger				
Mode		Asynchronous	Asynchronous	Asynchronous
Trigger delay with rising edge	µs	39.6 ±2.8	29.1 ±18.0	20.4 ±18.0
Trigger delay with falling edge	µs	57.8 ±2.8	31.1 ±18.0	23.3 ±18.0
Additive trigger delay (optional)	µs	15 µs...4 s	15 µs...4 s	15 µs...4 s
Sensor delay to exposure start	µs	< 100 ^{*2)}	< 100 ^{*2)}	< 100 ^{*2)}
Power consumption ^{*5)}				
	W	1.0-2.4	3.0-4.4	3.8-6.2

*1) The maximum possible pixel clock frequency depends on the PC hardware used.

*2) Requires maximum pixel clock frequency.

*3) Requires minimum pixel clock frequency.

*4) Use of the function increases the frame rate for monochrome models.

*5) The power consumption depends on the sensor model and the pixel clock setting.

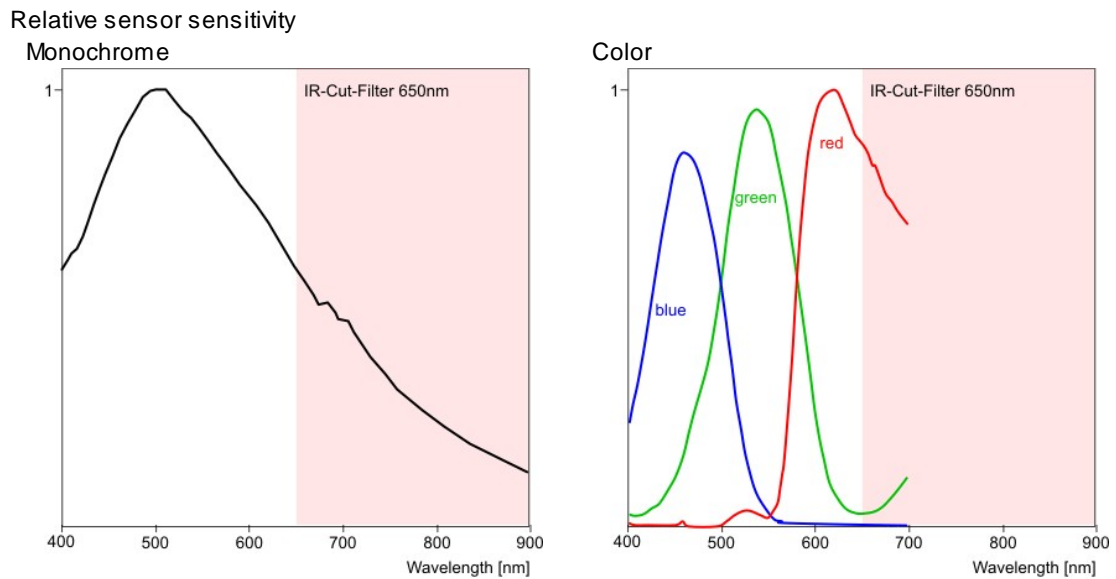


Figure 133: Sensor sensitivity of the UI-225x / UI-625x



Driver versions earlier than 3.30 allow setting an AOI height of less than 256 pixels for UI-225x cameras. An AOI height <256 pixels should be avoided, however, as it increases the camera's current consumption.



Notes on using the UI-225x / UI-625x

- Optimum pixel clock frequency is 29 MHz.
- Recommended pixel clock range 15 - 29 MHz.
- Long exposure times will increase the number of hotpixels.
- High temperatures will increase the black level of individual pixels.
- Vertical subsampling should not be used because it can cause corrupted images. The reason it is still supported by the driver is to ensure downward compatibility.

6.2.13 UI-241x / UI-641x

Sensor specification				
Sensor type	CCD			
Shutter system	Electronic global shutter			
Readout mode	Progressive scan			
Resolution class	VGA			
Resolution	640 x 480 pixels (0.31 Mpixels)			
Aspect ratio	4:3			
Bit depth	12 bits			
Optical sensor class	1/3 inch			
Exact sensitive area	4.74 x 3.55 mm			
Exact optical sensor diagonal	5.9 mm (1/2.7 inch)			
Pixel size	7.4 μm, square			
Sensor name, monochrome	Sony ICX424AL			
Sensor name, color	Sony ICX424AQ			
Gain				
Monochrome model (master gain)	18.0x			
Color model (master/RGB)	12.0x/4.0x			
Gain boost	2.0x (monochrome model only)			
Camera timing		USB uEye	GigE uEye SE	GigE uEye HE
Pixel clock range	MHz	5-30 ^{*1)}	5-30 ^{*1)}	5-30 ^{*1)}
Max. pixel clock with subsampling/binning	MHz	30 ^{*1)}	30 ^{*1)}	30 ^{*1)}
Frame rate (freerun mode)	fps	75.1 ^{*2)}	75.1 ^{*2)}	75.1 ^{*2)}
Frame rate (trigger mode, 1 ms exposure)	fps	69.0 ^{*2)}	69.0 ^{*2)}	69.0 ^{*2)}
Exposure time in freerun mode	ms	0.04 ^{*2)} -640 ^{*3)}	0.04 ^{*2)} -640 ^{*3)}	0.04 ^{*2)} -640 ^{*3)}
Exposure time in trigger mode	ms	0.04 ^{*2)} -10 min ^{*3)}	0.04 ^{*2)} -10 min ^{*3)}	0.04 ^{*2)} -10 min ^{*3)}
AOI				
Mode		Horizontal + Vertical ^{*4)}		
AOI image width, step width	Pixels	16 - 640, 4	16 - 640, 4	16 - 640, 4
Mono: AOI image height, step width	Pixels	120 - 480, 1	120 - 480, 1	120 - 480, 1
Color: AOI image height, step width	Pixels	120 - 480, 2	120 - 480, 2	120 - 480, 2
Mono: AOI position grid horizontal, vertical	Pixels	1, 1	1, 1	1, 1
Color: AOI position grid horizontal, vertical	Pixels	2, 2	2, 2	2, 2
AOI frame rate, 320 x 240 pixels (CIF)	fps	111	111	111
Binning				
Mode		Vertical ^{*4)}	Horizontal + Vertical ^{*4)}	Horizontal + Vertical ^{*4)}
Method		V: Monochrome binning, additive	H + V: Monochrome	H + V: Monochrome

			binning, additive	binning, additive
Frame rate with 2x binning, 640 x 240 pixels	fps	133	133	133
Frame rate with 3x binning, 640 x 160 pixels	fps	178	178	178
Frame rate with 4x binning, 640 x 120 pixels	fps	215	215	215
Subsampling				
Mode		-	-	-
Hardware trigger				
Mode		Asynchronous	Asynchronous	Asynchronous
Trigger delay with rising edge	μs	39.6 ± 2.5	30.2 ± 19.0	20.6 ± 19.0
Trigger delay with falling edge	μs	58.1 ± 2.5	39.2 ± 19.0	22.4 ± 19.0
Additive trigger delay (optional)	μs	$15 \mu\text{s} \dots 4 \text{ s}$	$15 \mu\text{s} \dots 4 \text{ s}$	$15 \mu\text{s} \dots 4 \text{ s}$
Sensor delay to exposure start	μs	$< 100^{*2)}$	$< 100^{*2)}$	$< 100^{*2)}$
Power consumption ^{*5)}				
	W	USB uEye	GigE uEye SE	GigE uEye HE
		1.0-1.9	3.0-4.0	3.5-5.3

*1) The maximum possible pixel clock frequency depends on the PC hardware used.

*2) Requires maximum pixel clock frequency.

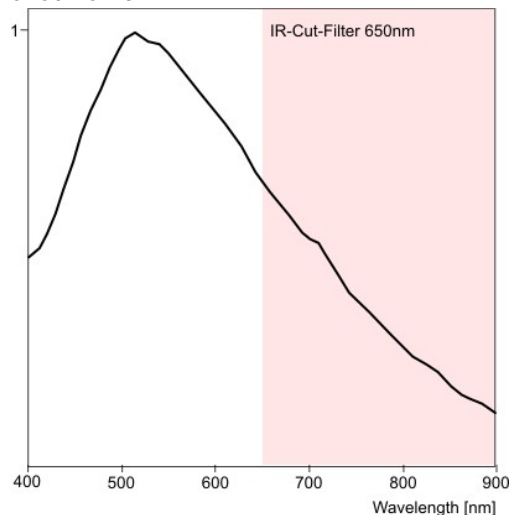
*3) Requires minimum pixel clock frequency.

*4) Use of the function increases the frame rate for monochrome models.

*5) The power consumption depends on the sensor model and the pixel clock setting.

Relative sensor sensitivity

Monochrome



Color

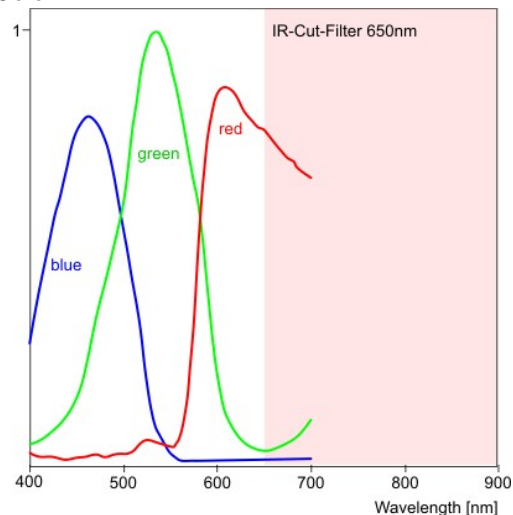


Figure 134: Sensor sensitivity of the UI-241x / UI-641x



Notes on using the UI-241x / UI-641x

- Optimum pixel clock frequency is 25 MHz.
- Recommended pixel clock range 15 - 27 MHz.
- Recommended pixel clock frequency for long term exposure is >10 MHz.

6.3 Mechanical Specifications

6.3.1 USB uEye SE

6.3.1.1 Housing Version

Lens mount	Enclosure protection class	Weight
C-mount	IP 30	with camera housing 62 g (CMOS), 74 g (CCD)
		C-mount, without housing 32 g (CMOS), 44 g (CCD)
		Board level version 18 g (CMOS), 30 g (CCD)



For the dimensions of the *USB uEye SE* accessories, please refer to the [USB uEye SE Accessories](#) chapter.

CMOS/CCD cameras

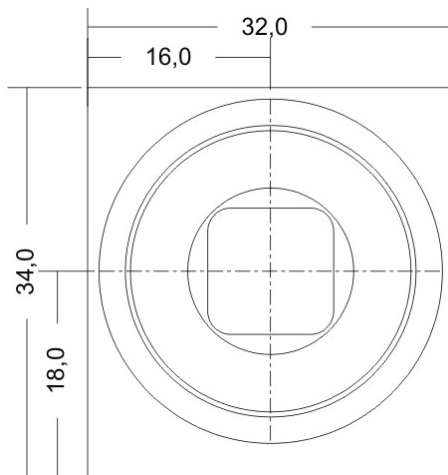


Figure 135: Front view

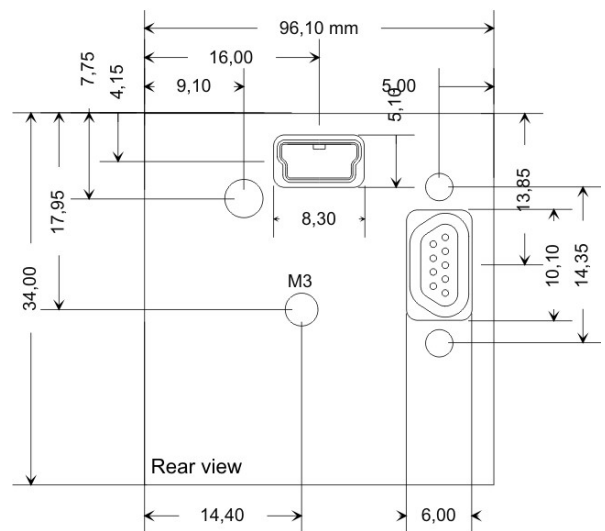
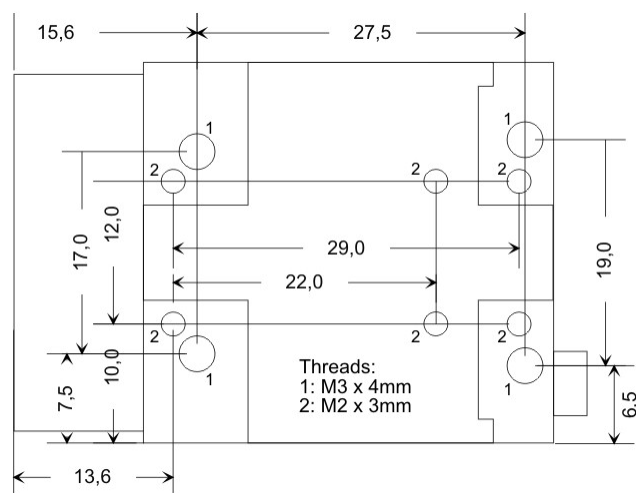
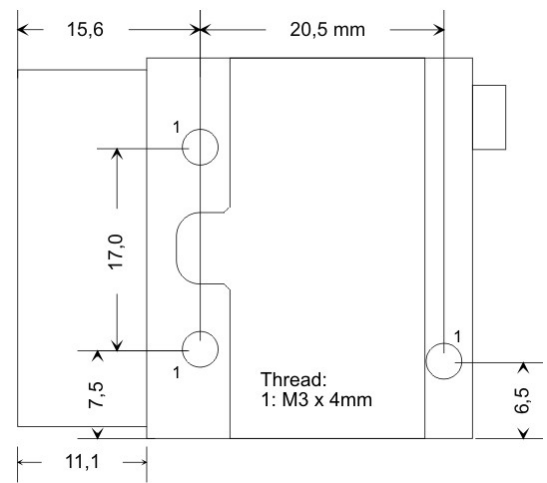
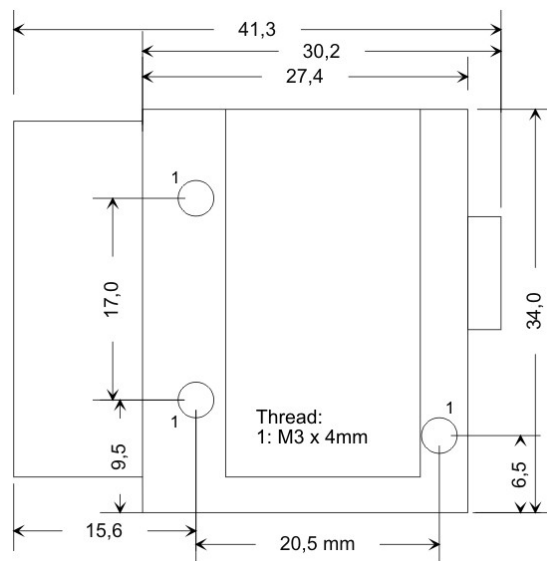


Figure 136: Rear view

CMOS cameras



CCD cameras



USB uEye SE CCD cameras have a 3 mm thick base. Do not use mounting screws that protrude more than 3 mm into the base. Otherwise, the camera could be damaged.

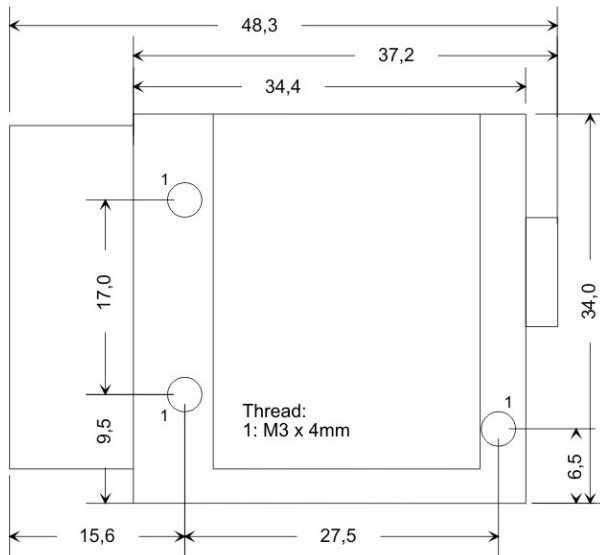


Figure 140: Side view

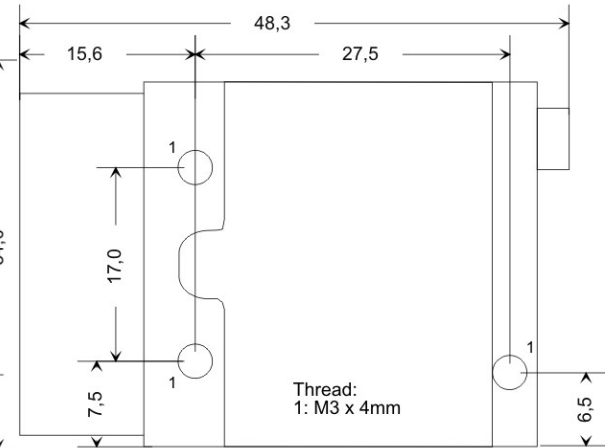


Figure 141: Top view

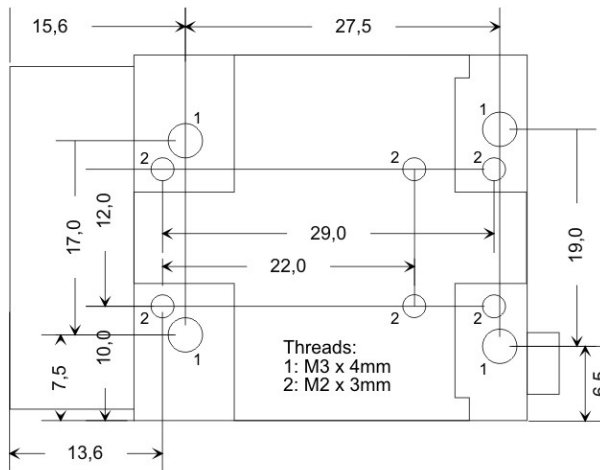


Figure 142: Bottom view

6.3.1.2 OEM Version 1 (C-Mount without Housing)

CMOS/CCD cameras

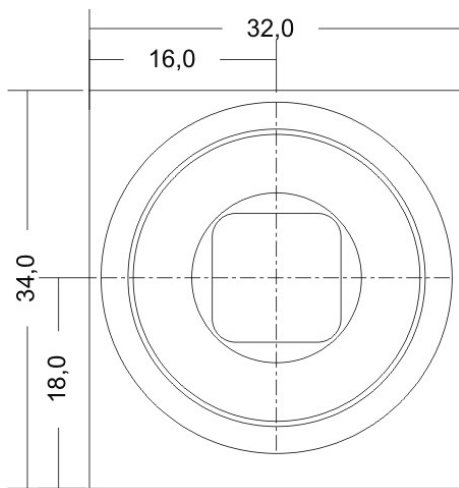


Figure 143: Front view

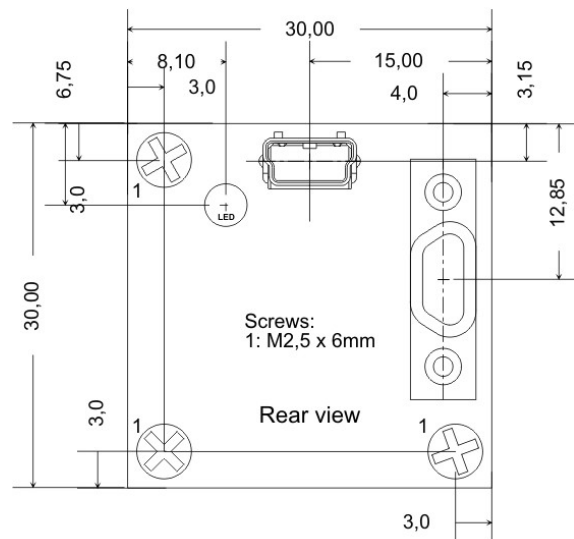


Figure 144: Rear view

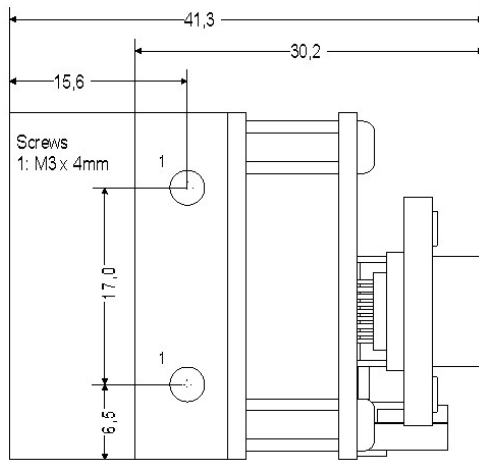
CMOS cameras

Figure 145: Side view

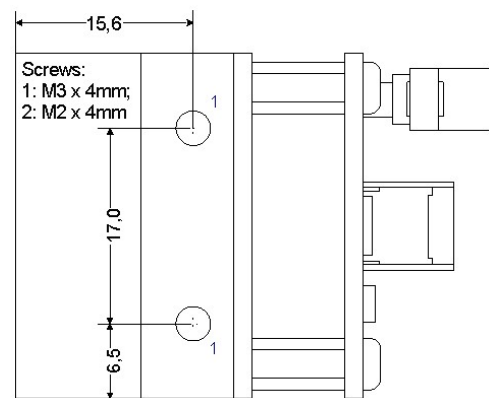


Figure 146: Top view

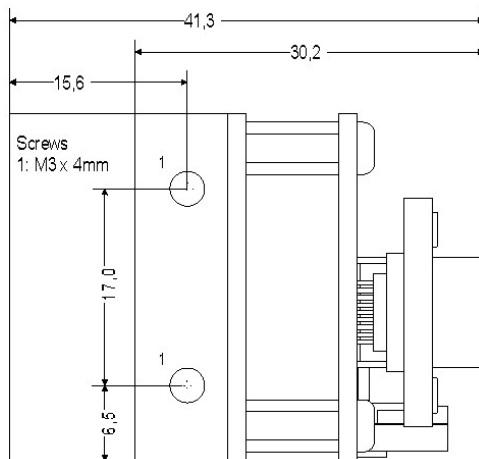


Figure 147: Bottom view

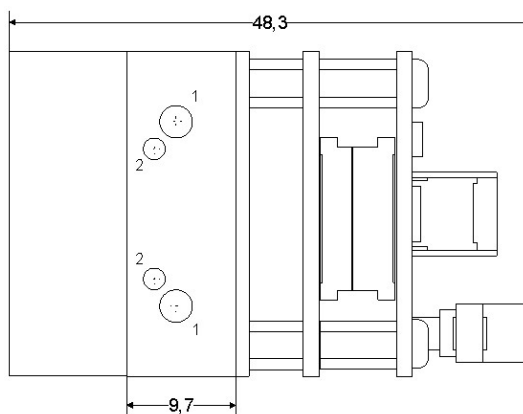
CCD cameras

Figure 148: CMOS - bottom view

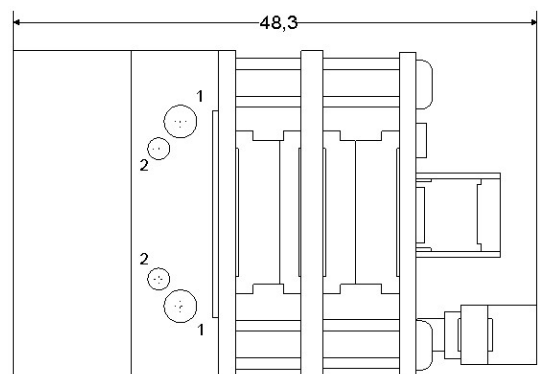


Figure 149: CCD - bottom view

6.3.1.3 OEM Version 2 (PCB Stack)

CMOS/CCD cameras

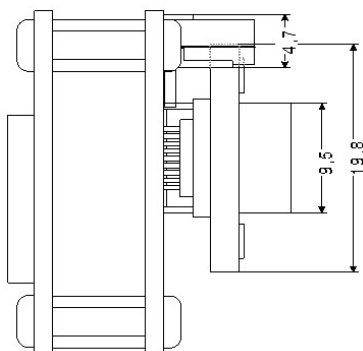


Figure 150: Side view

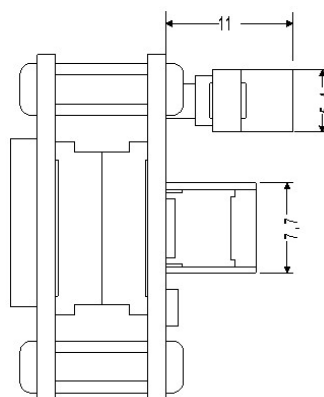


Figure 151: Top view

Value of distance X (model-dependent)

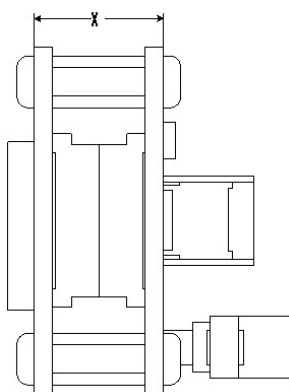


Figure 152: Bottom view

Camera model	Distance X
UI-1222 (sensor version V022)	11.4
UI-1222 (sensor version V032)	11.0
UI-1542	10.9
UI-1642	10.6
UI-1552	10.6
UI-1462	10.9
UI-1482	11.4
UI-1482	11.4

CCD cameras

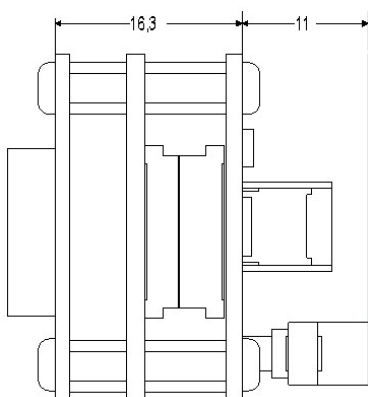


Figure 153: Bottom view

6.3.2 USB uEye ME

6.3.2.1 Housing Version

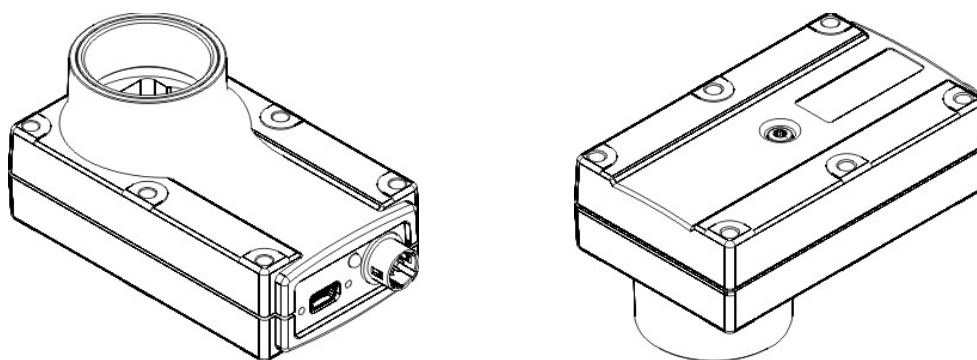


Figure 154: USB uEye ME

Lens mount	Enclosure protection class	Weight
C-mount	IP 30	164 g (CMOS), 174 g (CCD)



For the dimensions of the *USB uEye ME* accessories, please refer to the [USB uEye ME Accessories](#) chapter.

Dimensions of the housing version

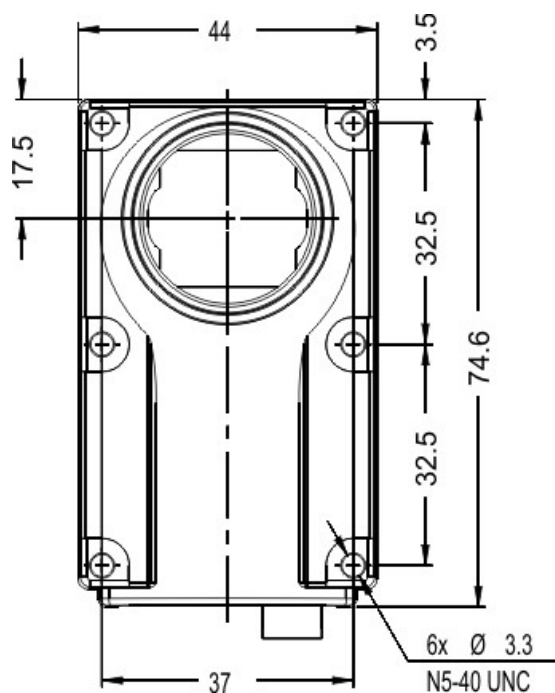


Figure 155: USB uEye ME dimensions - Front view

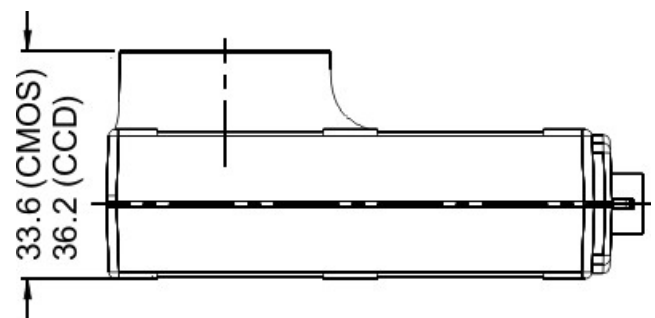


Figure 156: USB uEye ME dimensions - Side view

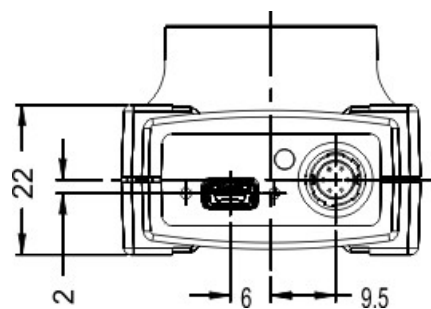


Figure 157: USB uEye ME dimensions
- Bottom view

6.3.2.2 OEM Version 1 (C-Mount without Housing)

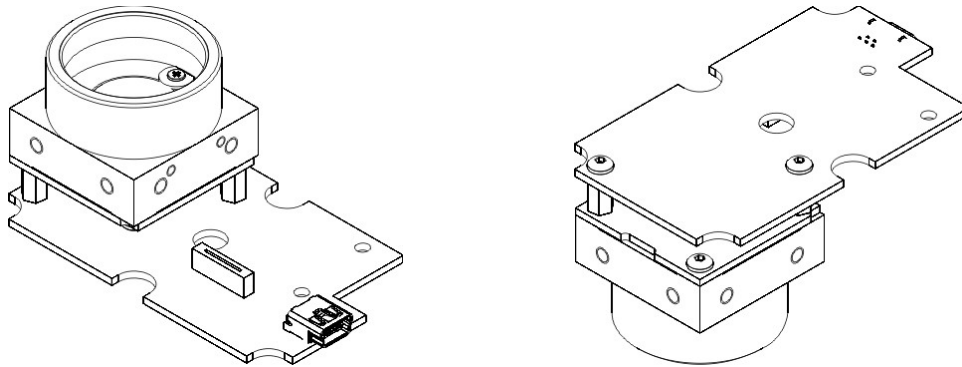


Figure 161: USB uEye ME - 3D view

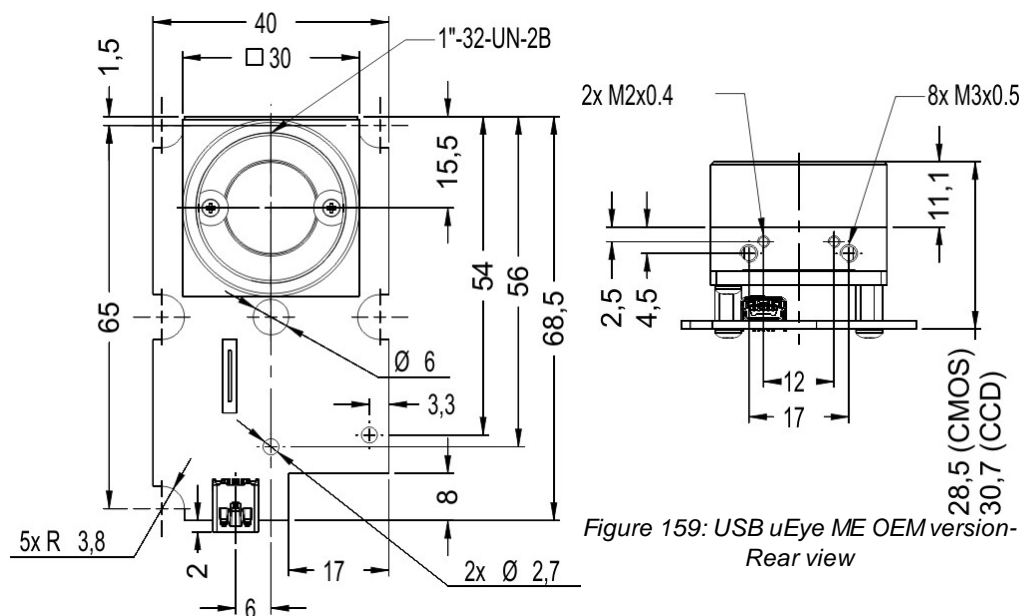


Figure 158: USB uEye ME OEM version - Top view

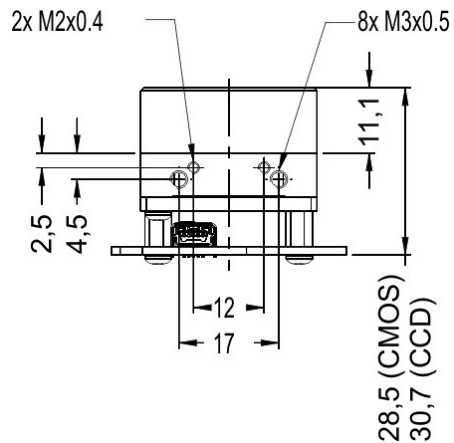


Figure 159: USB uEye ME OEM version - Rear view

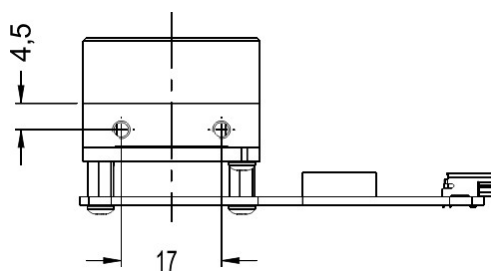


Figure 160: USB uEye ME OEM version - Side view

6.3.2.3 OEM Version 2 dimensions (PCB Stack)

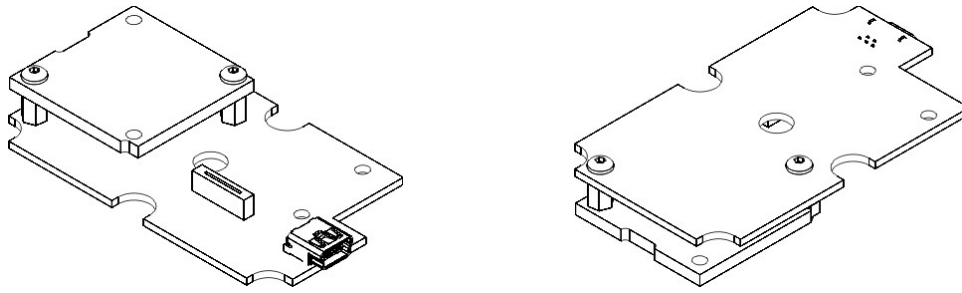


Abbildung 166: USB uEye ME OEM version 2 - 3D view

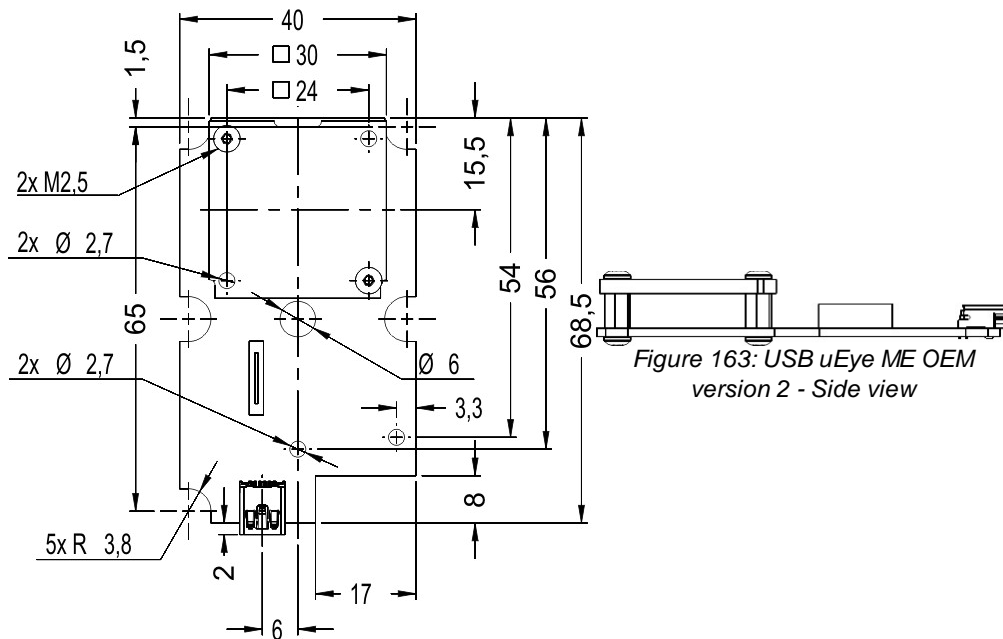


Figure 162: USB uEye ME OEM version 2 - Top view

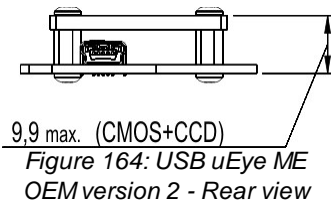


Figure 164: USB uEye ME OEM version 2 - Rear view

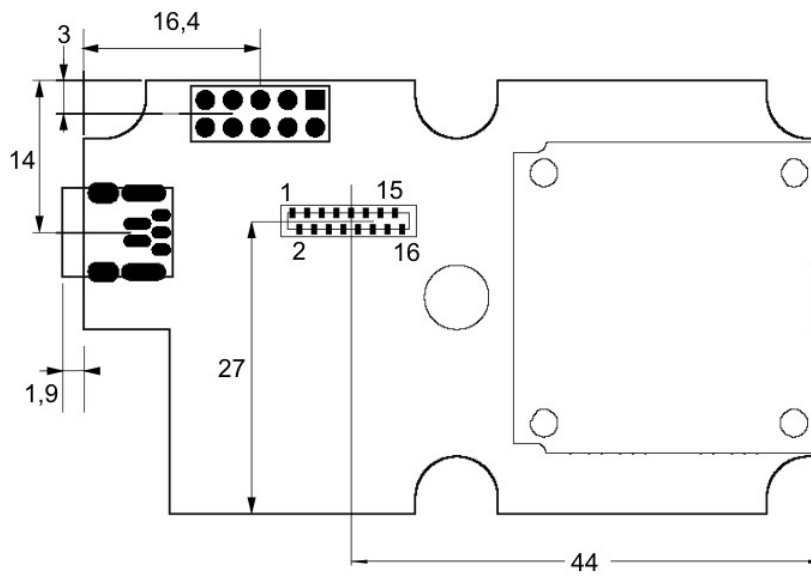
Connector positions on PCB version

Figure 165: USB uEye ME OEM version - Connector position

6.3.3 Dimensions

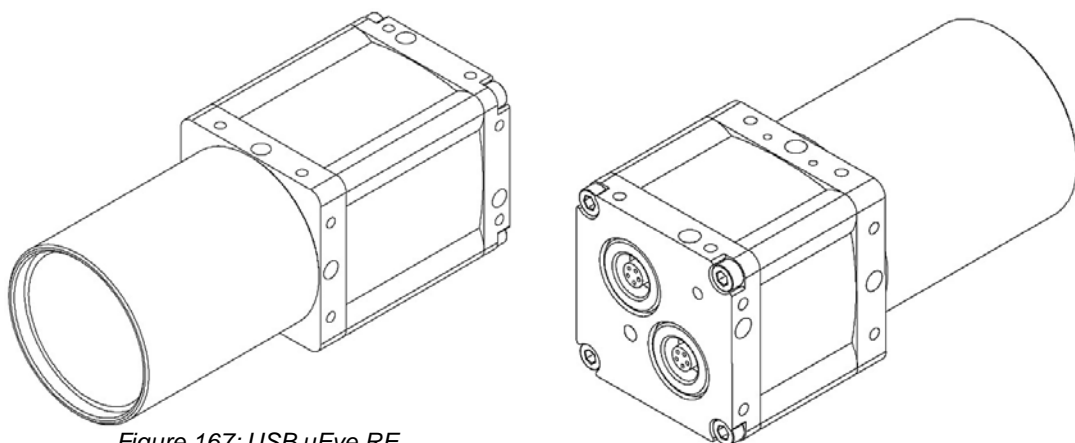


Figure 167: USB uEye RE

Lens mount	Enclosure protection class	Weight	
C-mount	IP 65/IP 67	Camera without tube	145 g (CMOS), 180 g (CCD)
		Tube	max. 70 g



For the dimensions of the *USB uEye RE* accessories, please refer to the [USB uEye RE Accessories](#) chapter.



The *USB uEye RE* housing variant is IP 65/67 compliant. The following requirements must be met for compliance with the IP 65/67 standards:

- Cables must be attached to both connectors (USB and trigger) during operation. If the trigger connector is not used, it must be closed with the cover.
- A tube must be connected.

Attention: The *USB uEye RE* has not been approved for underwater use.

Housing dimensions

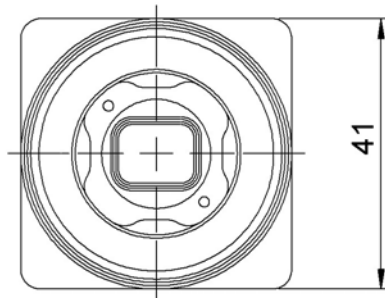


Figure 168: USB uEye RE - front view

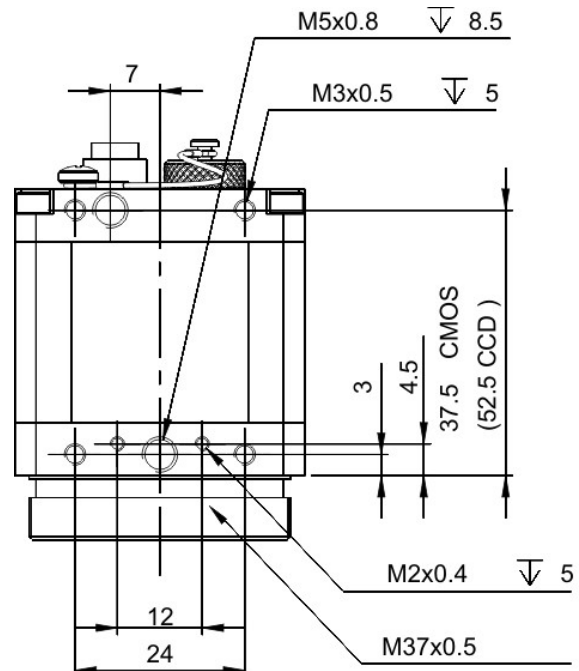


Figure 169: USB uEye RE - bottom view

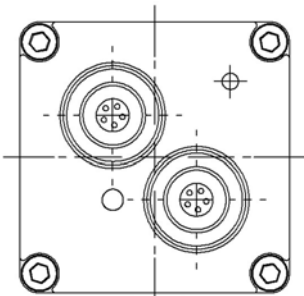


Figure 170: USB uEye RE - rear view

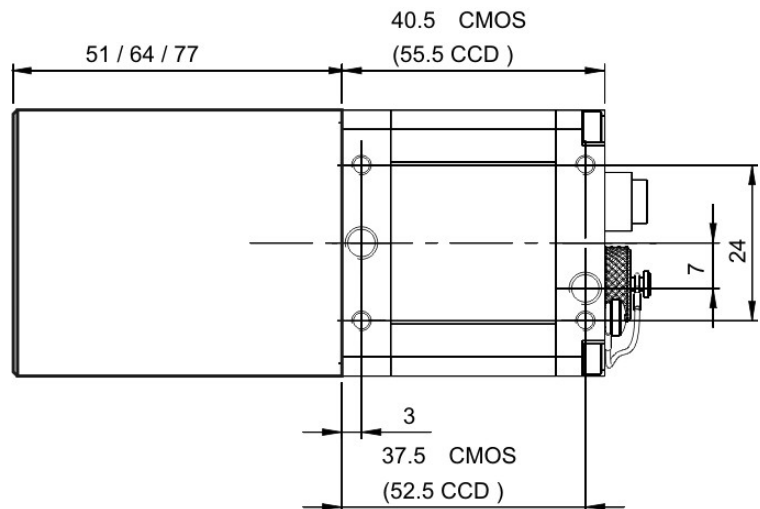


Figure 171: USB uEye RE - side view

6.3.4 USB uEye LE

6.3.4.1 Housing Version

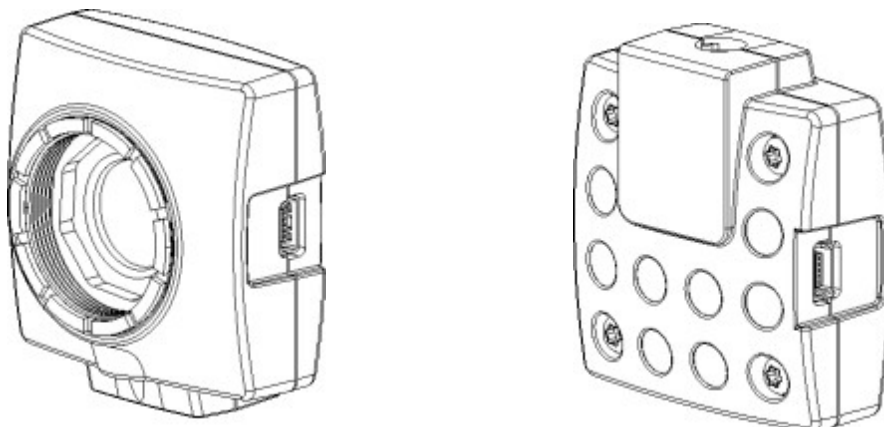


Figure 172: USB uEye LE

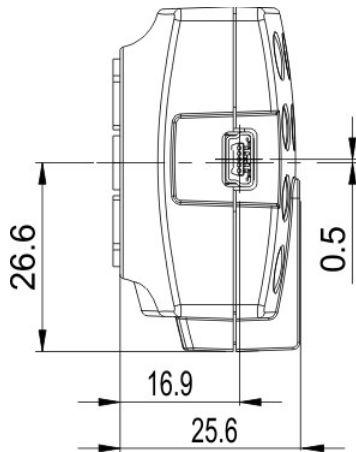
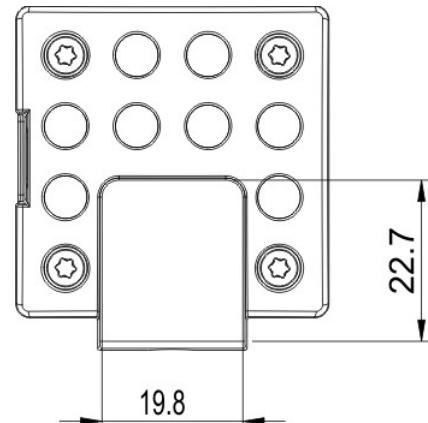
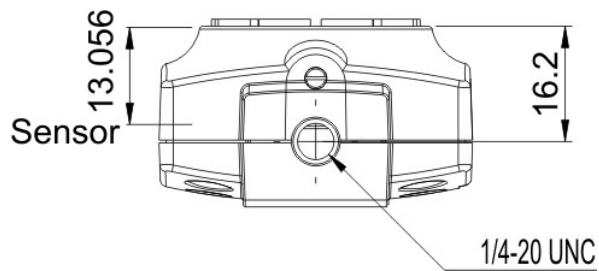
Lens mount	Enclosure protection class	Weight
CS-/S-mount	IP30 (with housing)	32 g (with housing) 12 g (S-mount) 8 g (PCB only)



For the dimensions of the *USB uEye LE* accessories, please refer to the [USB uEye LE Accessories](#) chapter.

For the *uEye LE* series the following sensor / lens mount combinations are available:

	CS-mount with housing: UI-1xx 5LE -x	S-mount M12: UI-1xx 6LE -x	S-mount M14: UI-1xx 7LE -x	without housing without lens holder: UI-1xx 8LE -x
UI-122xLE-C	X	X		X
UI-122xLE-M	X	X		X
UI-154xLE-M	X	X	X	X
UI-164xLE-C	X	X		X
UI-155xLE-C	X	X		X
UI-146xLE-C	X	X	X	X
UI-148xLE-C	X	X	X	X

Dimensions of the housing version*Figure 173: USB uEye LE - side view**Figure 174: USB uEye LE - rear view**Figure 175: USB uEye LE - bottom view*

6.3.4.2 PCB Version and Lens Holder



The applicable tolerances for the overall dimensions of PCBs are higher than the tolerances for housing dimensions.



The board-level version of the UI-1490x-LE has not been prepared for use of a vertical USB connector. The board provides five additional connection points to which a USB cable can be soldered if required (see [USB uEye LE: Pin Assignment of the USB Connector](#)).

Dimensions of the board level version



Figure 176: USB uEye LE USB connector
- horizontal position (default)

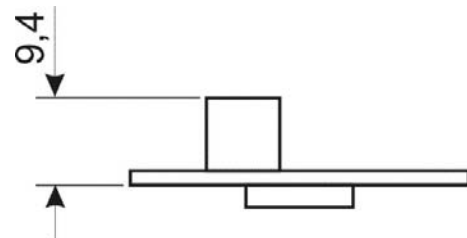


Figure 177: USB uEye LE USB connector
- vertical position (project-related option)

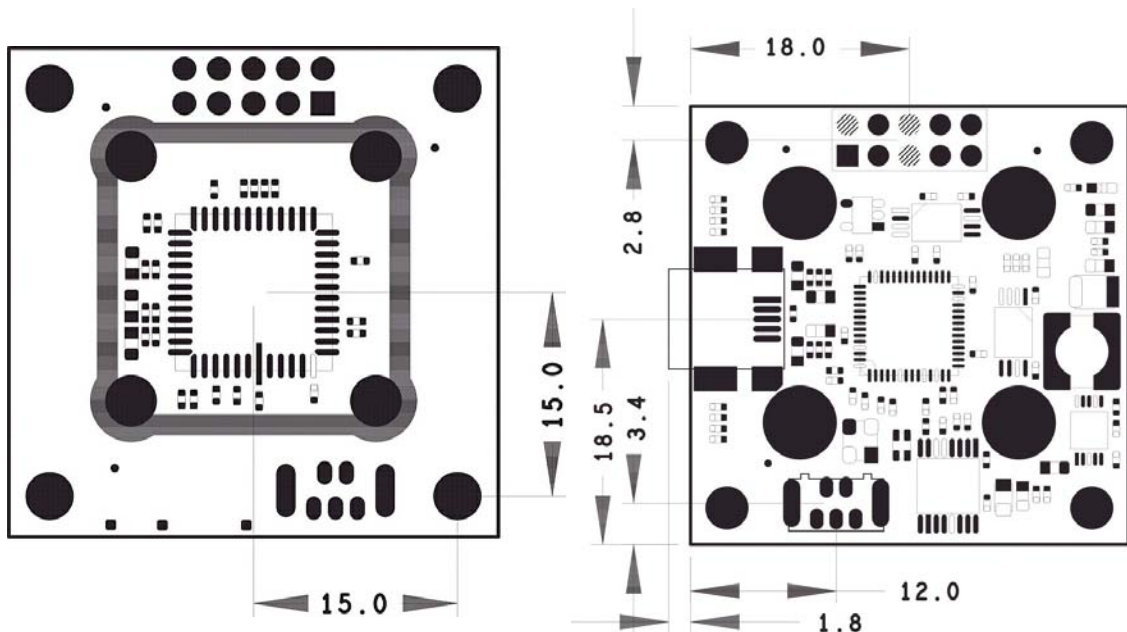


Figure 178: USB uEye LE PCB version - top view

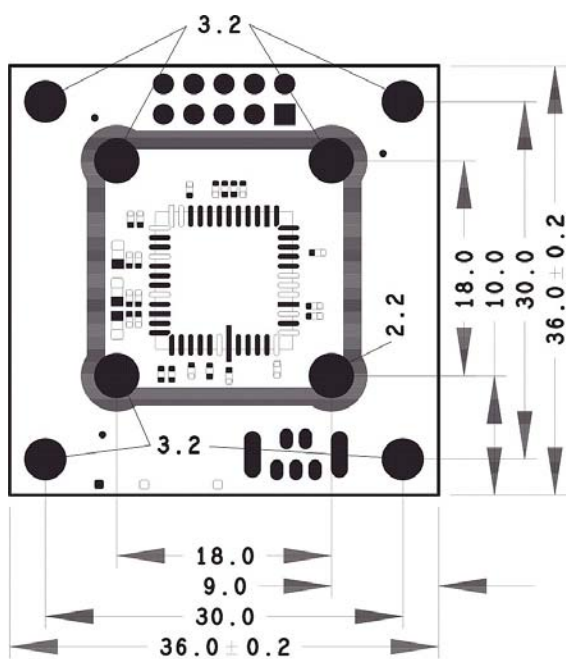


Figure 179: USB uEye LE PCB version - bottom view

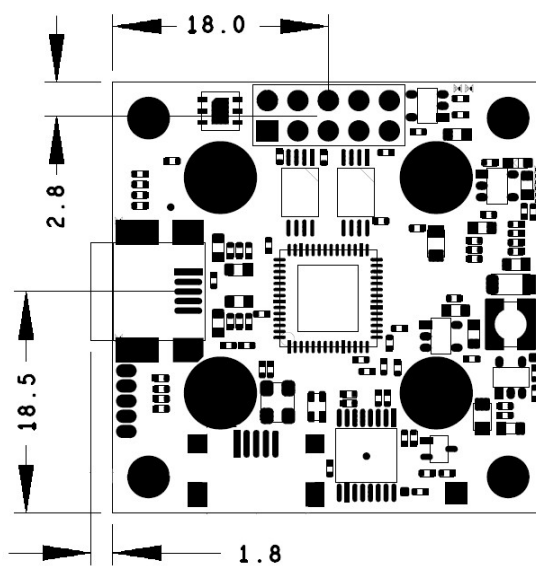


Figure 180: UI-149x-LE PCB version - bottom view

Dimensions of the S-mount lens holder (only USB uEye LE board level version)

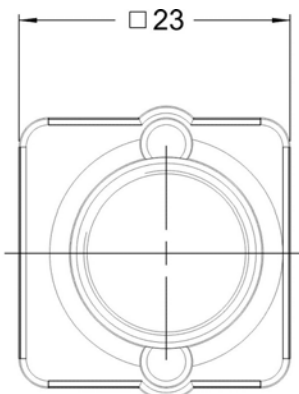
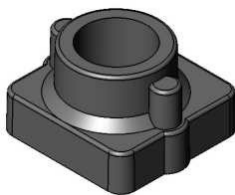


Figure 181: USB uEye LE lens holder - top view

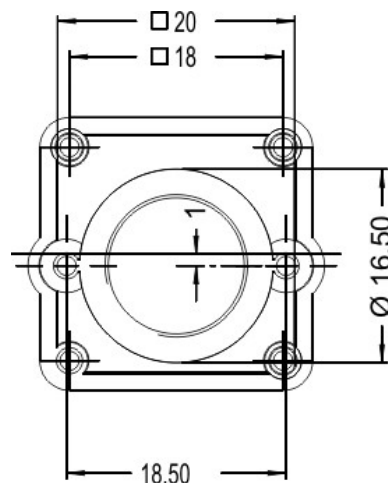
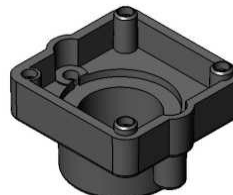


Figure 182: USB uEye LE lens holder - bottom view

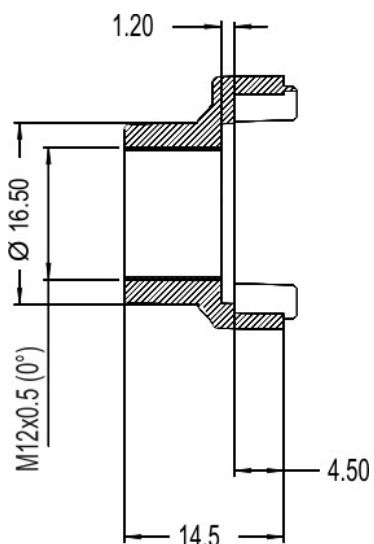


Figure 183: USB uEye LE lens holder M12

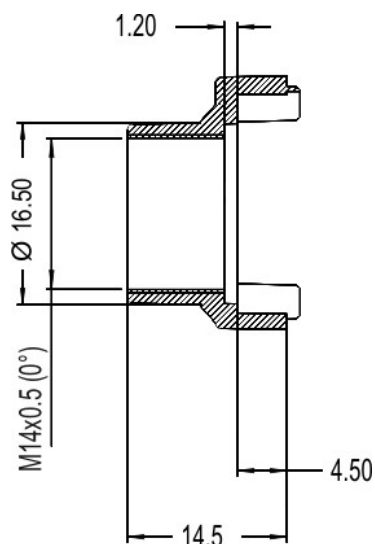


Figure 184: USB uEye LE lens holder M14

6.3.5 GigE uEye SE

6.3.5.1 Housing Version

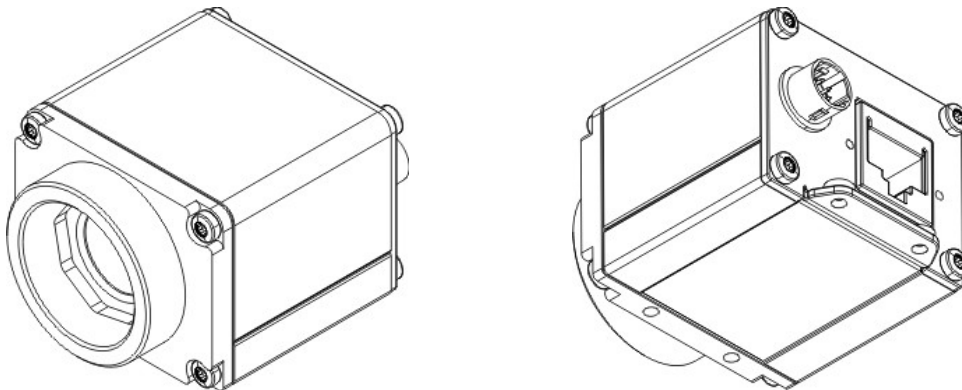


Figure 185: GigE uEye SE

Lens mount	Enclosure protection class	Weight
C-mount	IP 30	approx. 95 g (CMOS) approx. 112 g (CCD)



For the dimensions of the *GigE uEye SE* accessories, please refer to the [GigE uEye SE Accessories](#) chapter.

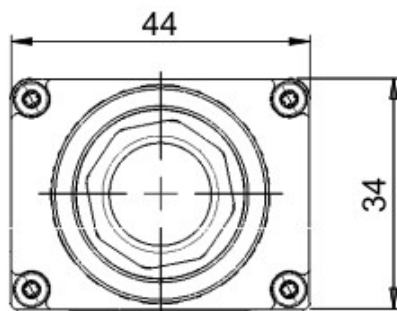


Figure 186: Front view

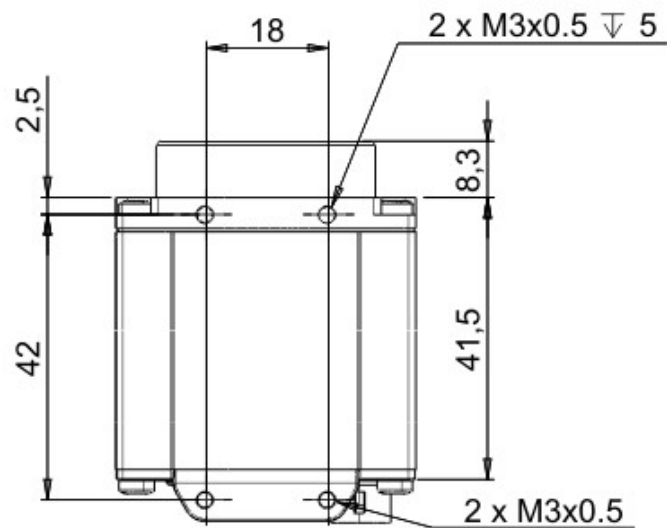


Figure 187: GigE uEye SE - bottom view (CMOS)

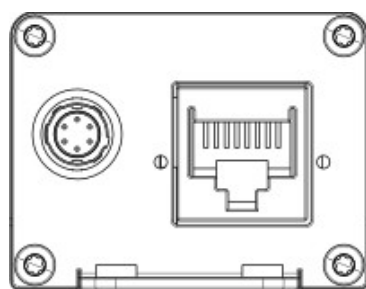


Figure 188: Rear view

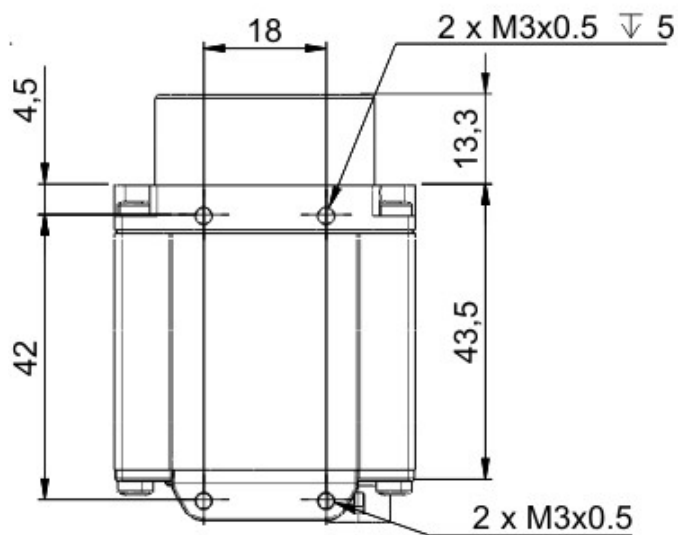


Figure 189: Bottom view

6.3.5.2 GigE uEye SE OEM version 1 (without housing)

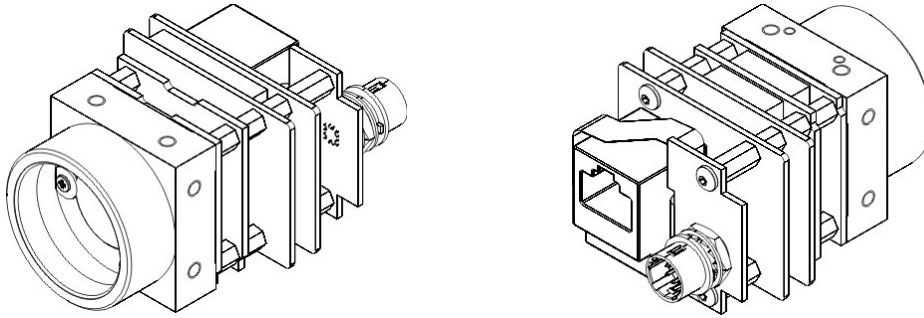


Figure 196: GigE uEye SE OEM version with CCD - 3D view

CCD and CMOS cameras

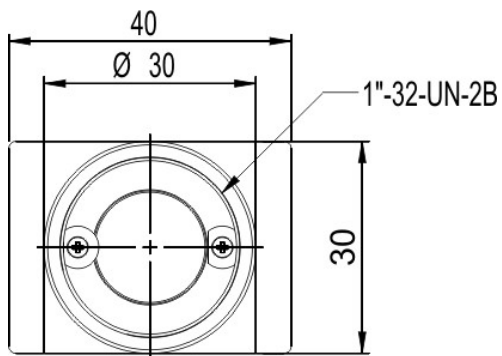


Figure 190: GigE uEye SE OEM version - Front view (CCD, CMOS)

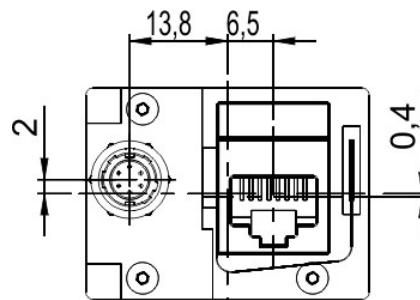


Figure 191: GigE uEye SE OEM version - Rear view (CCD, CMOS)

CCD cameras

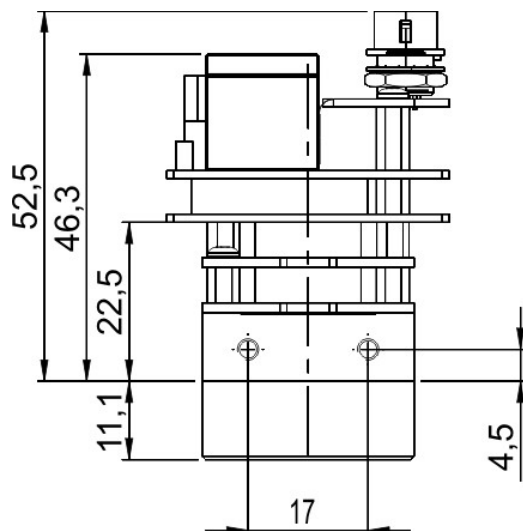


Figure 192: GigE uEye SE OEM version - Top view (CCD)

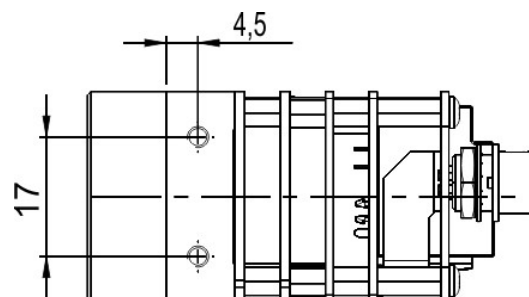


Figure 193: GigE uEye SE OEM version - Side view (CCD, CMOS)

CMOS cameras

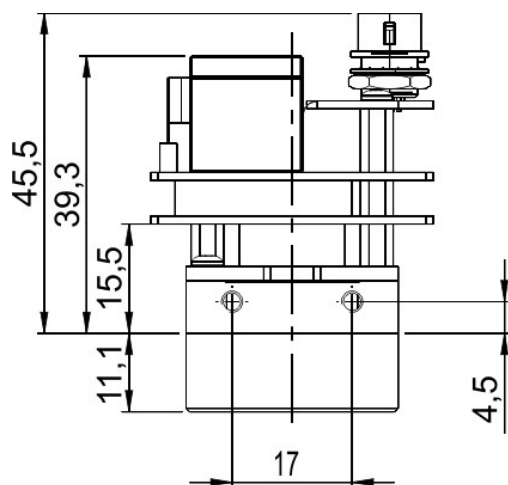


Figure 194: GigE uEye SE OEM version - Top view (CMOS)

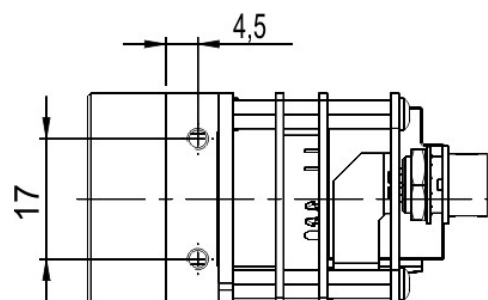


Figure 195: GigE uEye SE OEM version - Side view (CMOS)

6.3.5.3 GigE uEye SE OEM version 2 (PCB stack)

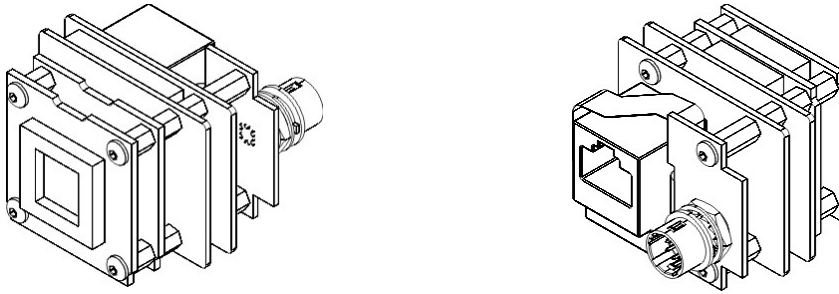


Figure 203: GigE uEye SE OEM version 2 (PCB stack) with CCD - 3D view

CCD and CMOS cameras

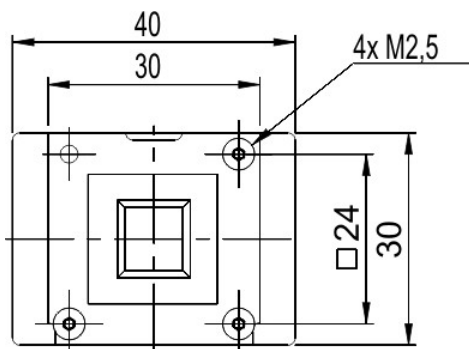


Figure 197: GigE uEye SE OEM version 2 - Front view (CCD, CMOS)

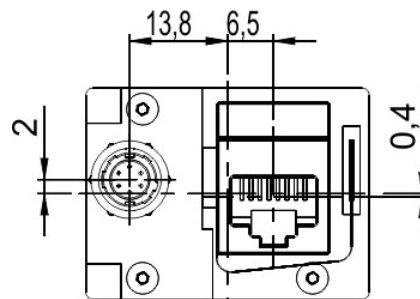


Figure 198: GigE uEye SE OEM version 2 - Rear view (CCD, CMOS)

CCD cameras

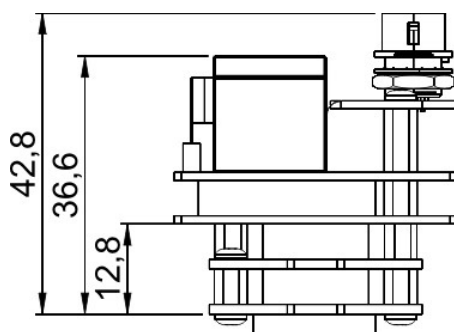


Figure 199: GigE uEye SE OEM version 2 - Top view (CCD)

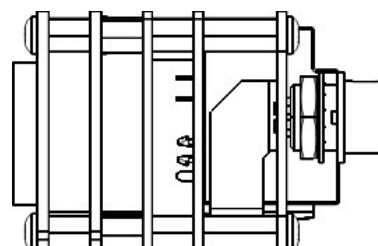


Figure 200: GigE uEye SE OEM version - Side view (CCD)

CMOS version

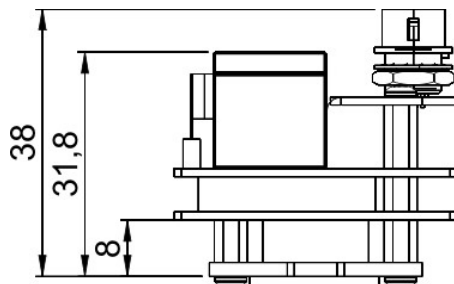


Figure 201: GigE uEye SE OEM version 2
- Top view (CMOS)

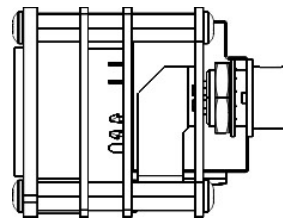


Figure 202: GigE uEye
SE OEM version 2 - Side
view (CMOS)

6.3.6 GigE uEye RE

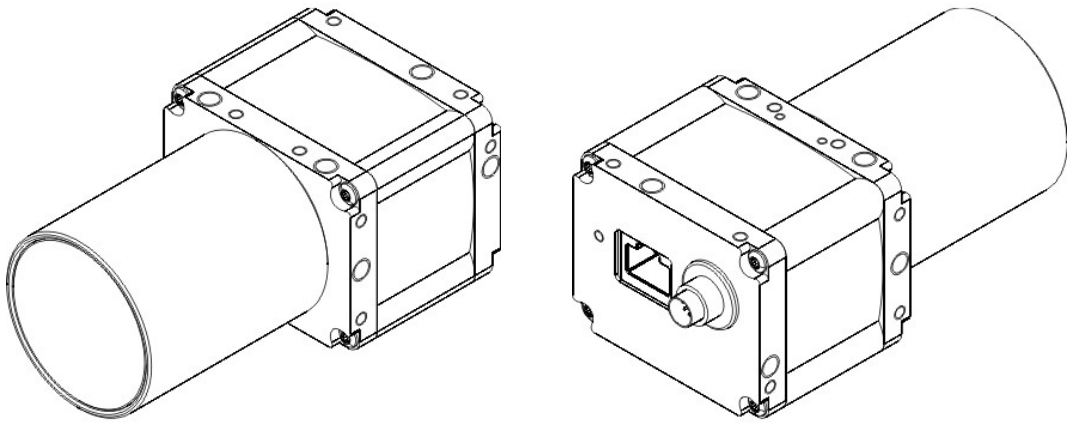


Figure 213: GigE uEye RE

Lens mount	Enclosure protection class	Weight
C-mount	IP 65 / IP 67	approx. 168 g (CMOS) approx. 188 g (CCD)



For the dimensions of the *GigE uEye RE* accessories, please refer to the [GigE uEye RE Accessories](#).

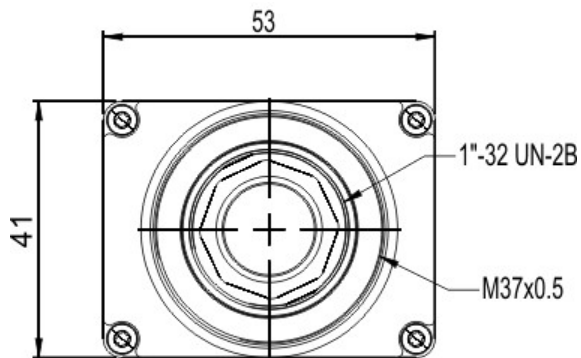


Figure 204: GigE uEye RE - Front view (CMOS / CCD)

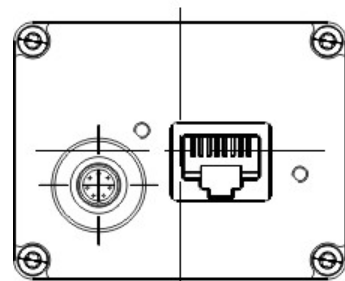
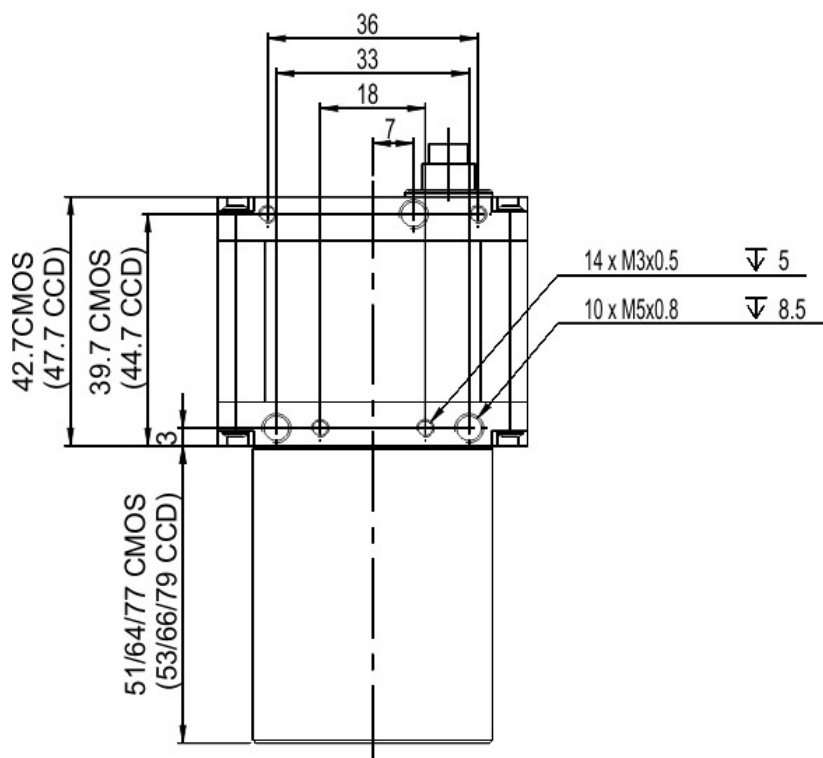
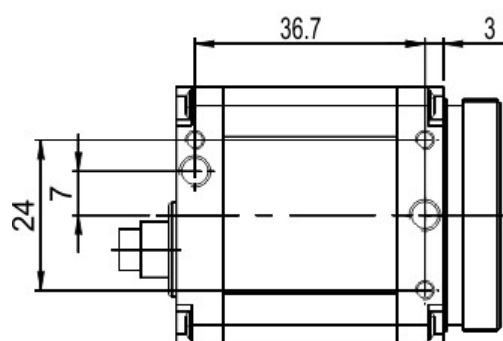
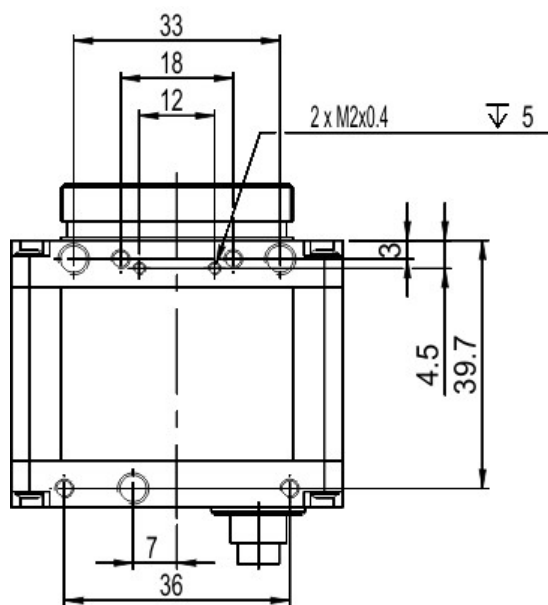


Figure 205: GigE uEye RE - Rear view (CMOS / CCD)



CMOS cameras



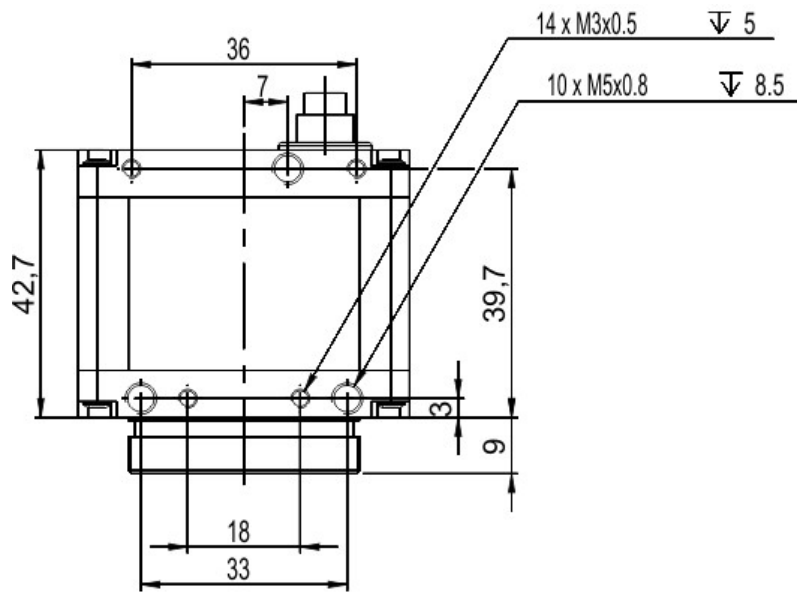


Figure 209: GigE uEye RE - Top view (CMOS)

CCD cameras

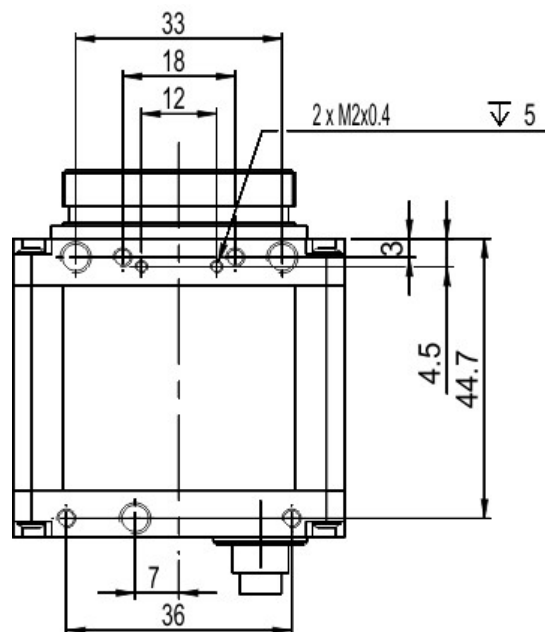


Figure 210: GigE uEye RE - Bottom view (CCD)

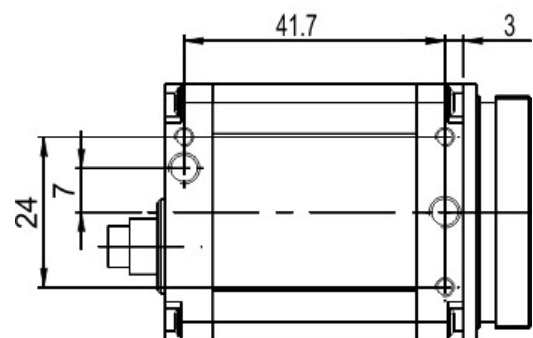


Figure 211: GigE uEye RE - Side view (CCD)

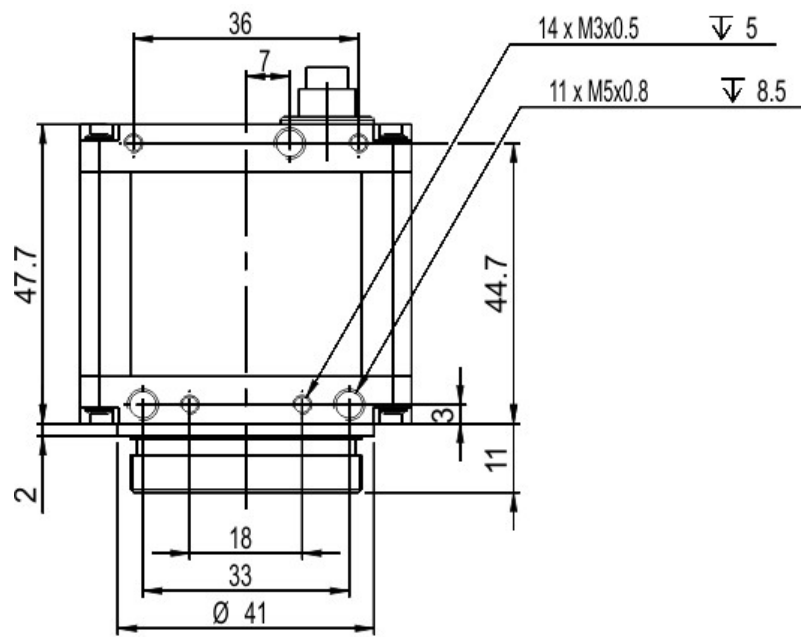


Figure 212: GigE uEye RE - Top view (CCD)

6.3.7 GigE uEye HE

6.3.7.1 Housing Version

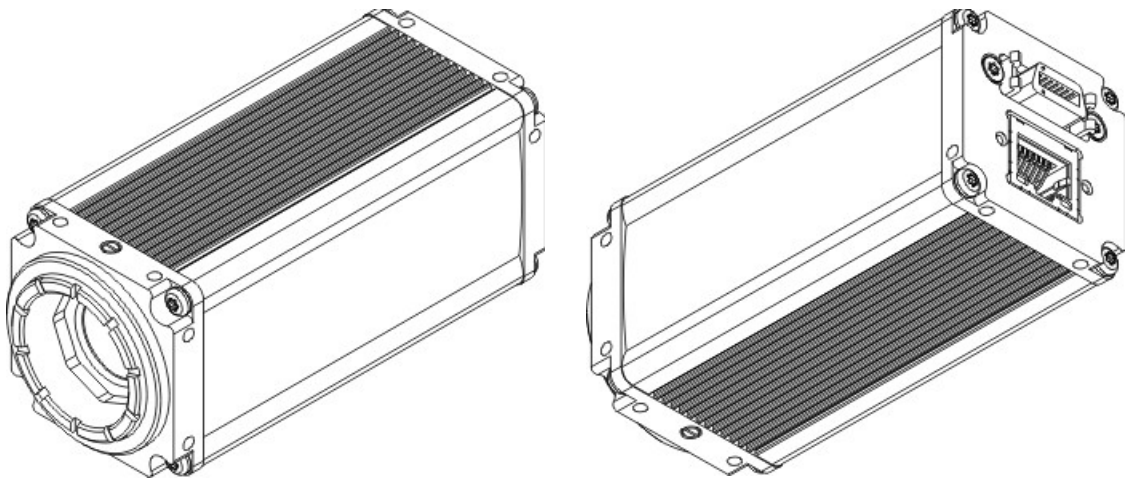


Figure 214: GigE uEye HE

Lens mount	Enclosure protection class	Weight
C-mount	IP 65/IP 67	170 g (CMOS/CCD)



For the dimensions of the *GigE uEye HE* accessories, please refer to the [GigE uEye HE Accessories](#) chapter.

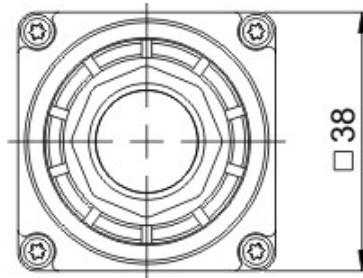


Figure 215: Front view

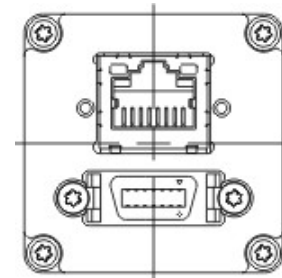


Figure 216: Rear view

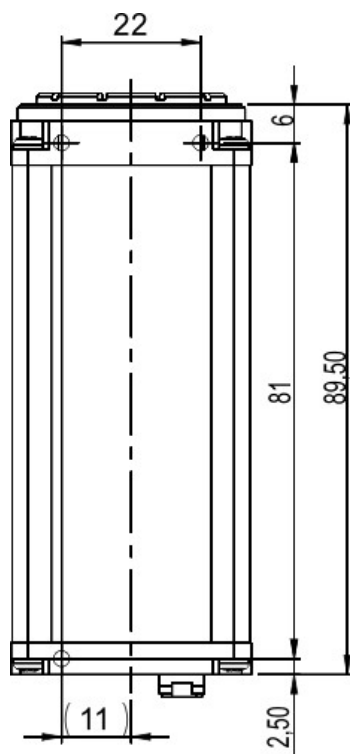


Figure 217: Side view

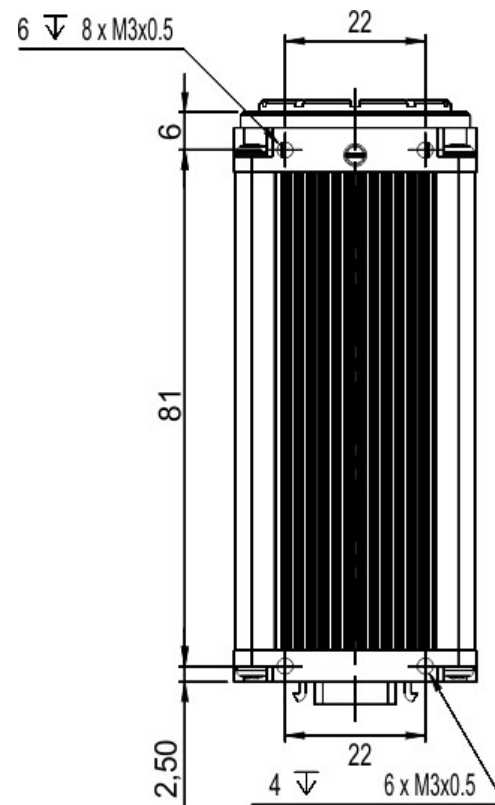


Figure 218: Bottom view

6.3.7.2 Angled Housing Version

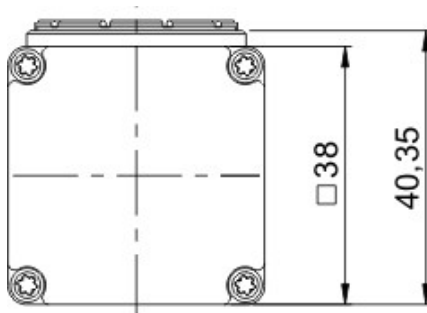


Figure 219: GigE uEye HE 90° - Top view

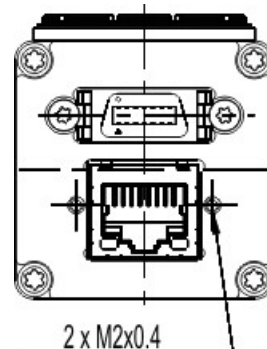


Figure 220: GigE uEye HE 90° - Bottom view

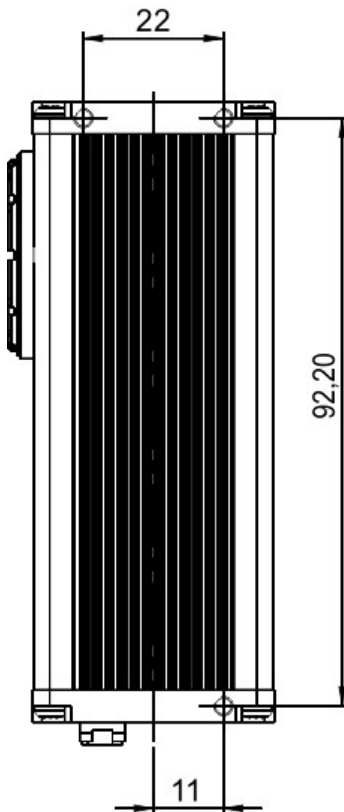


Figure 221: GigE uEye HE 90° - Side view

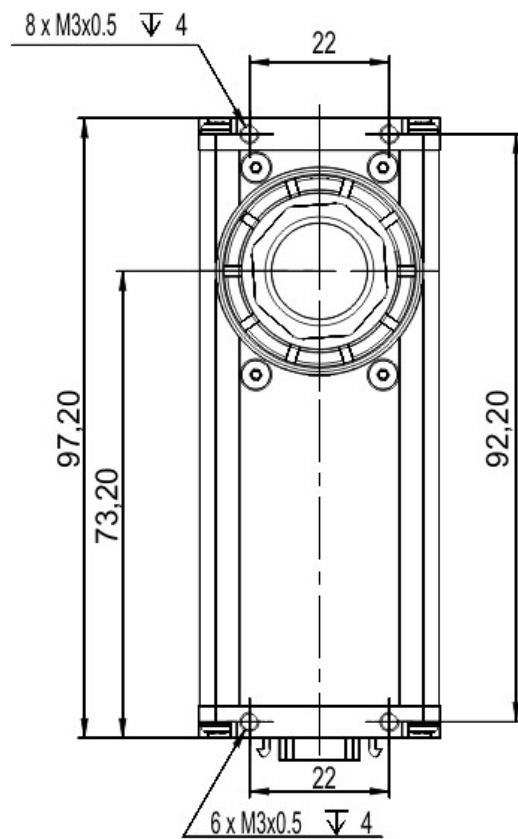


Figure 222: GigE uEye HE 90° - Front view

6.3.8 Flange Back Distance

6.3.8.1 Calculating the Flange Back Distance

To correctly determine the flange back distance of a *uEye* camera, you need to consider the distance between the lens flange and the active area of the sensor and, additionally, the type and thickness of any materials inserted into the optical path.

The *distance in air* between the threaded flange and the active area is 17.526 mm with C-mount lenses and 12.526 mm with CS-mount lenses.

This *mechanical distance* can change due to the material-specific refractive index of the inserted materials. The glass cover of the sensor and all filters inserted into the optical path (see [Filter Types](#) table) must be taken into account in the calculation.

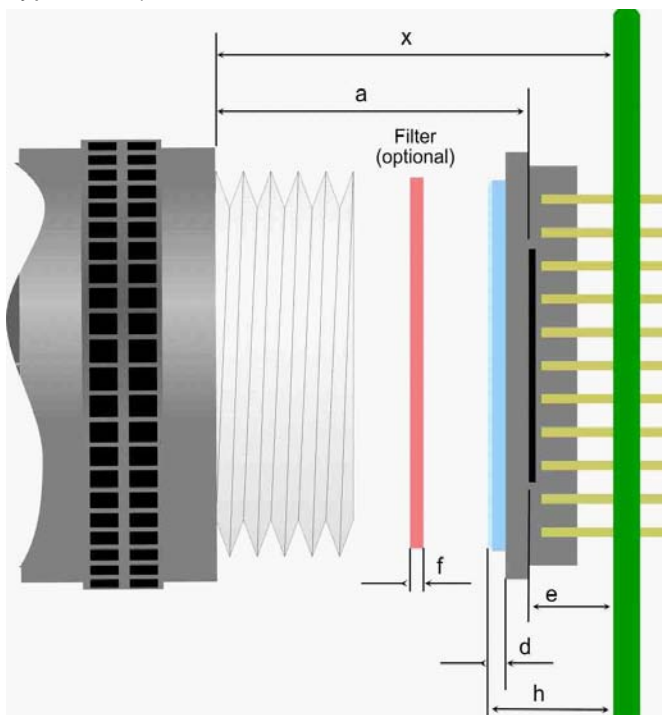


Figure 223: Calculating the flange back distance (schematic illustration)

Designation	Description
a	Distance from threaded flange to active sensor area (flange back distance) 17.526 mm ^{*1)} for C-mount 12.526 mm ^{*1)} for CS-mount
x	Distance from threaded flange to PCB
e	Distance from active sensor area to PCB
d	Thickness of the glass cover of the sensor
f	Filter thickness (optional)
n	Refractive index
h	Maximum sensor height above the PCB

^{*1)} This distance describes the equivalent in air (see introduction above)



You can use the following formula to calculate the mechanical flange back distance:

$$x = a + \frac{d \times (n_{\text{glass}} - 1)}{n_{\text{glass}}} + \frac{f \times (n_{\text{filter}} - 1)}{n_{\text{filter}}} + e$$

The tolerances for the position accuracy of uEye camera sensors are given in the [Position Accuracy](#) chapter.

Calculating the flange back distance for uEye cameras with C-mount

uEye sensor	Sensor glass d [mm]	Refractive index (n glass)	Distance e [mm]	x without filter glass [mm]	x with filter glass [mm]	Sensor height h [mm]
CMOS						
UI-122x/522x ^{*V022)}	0.400	1.520	0.642	18.040	18.780	1.450
UI-122x/522x ^{*V032)}	0.550	1.520	1.400	19.110	19.460	2.500
UI-154x/554x	0.525	1.500	1.270	18.970	19.320	2.480
UI-164x/564x	0.550	1.500	1.400	19.110	19.460	2.500
UI-145x/545x	0.525	1.500	1.270	18.970	19.320	2.480
UI-155x/555x	0.550	1.500	1.400	19.110	19.460	2.500
UI-146x/546x	0.550	1.490	1.250	18.960	19.300	2.375
UI-148x/548x	0.400	1.490	0.725	18.380	18.790	1.375
CCD						
UI-221x/621x	0.750	1.500	2.810	20.590	20.930	4.690
UI-231x	0.750	1.500	2.810	20.590	20.930	4.780
UI-241x/641x	0.750	1.500	2.810	20.590	20.930	4.780
UI-222x/622x	0.750	1.500	2.810	20.590	20.930	4.690
UI-223x/623x	0.750	1.500	2.810	20.590	20.930	4.780
UI-224x/624x	0.750	1.500	2.810	20.590	20.930	4.830
UI-225x/625x	0.500	1.500	2.910	20.590	20.950	4.430

^{*V022)} Sensor model MT9V022, identified by black housing (BGA package)

^{*V032)} Sensor model MT9V032, identified by brown housing (technically identical with V022)

Calculation example: UI-154x-xx with IR-cut filter

(a = 17.526 mm, d = 0.525 mm, nGlass = 1.50, f = 1mm, nFilter = 1.53; see [Filter Types](#) table)

$$x = 17.526 \text{ mm} + \frac{0.525 \text{ mm} \times (1.50 - 1)}{1.50} + \frac{1.00 \text{ mm} \times (1.53 - 1)}{1.53} + 1.27 \text{ mm} = 19.32 \text{ mm}$$

Calculating the flange back distance for USB uEye LE cameras with CS-mount



For *USB uEye LE* cameras with CS-mount, the flange back distance is only 12.526 mm!

USB uEye LE sensor	Sensor glass d [mm]	Distance e [mm]	Sensor height h [mm]
CMOS			
122xLE	0.550	1.400	2.500
164xLE	0.550	1.400	2.500
148xLE	0.400	0.725	1.375
154xLE	0.525	1.270	2.480
155xLE	0.550	1.400	2.500

6.3.8.2 Adjusting the Flange Back Distance

Some *uEye* models feature an adjusting ring that allows changing the flange back distance. Please follow the information below to avoid damage to the camera.

USB uEye LE with fix filter glass (production until April 2009)

The adjusting ring of the *USB uEye LE* has 10 adjusting positions. For each adjusting position, the flange back distance is altered by +/- 50 μm .



To change the position of the adjusting ring, please proceed exactly in the following order:

1. Loosen the locking screw (see figure below).
2. Adjust the CS-mount ring (IDS special tool required, see [USB uEye LE Accessories](#)).
3. Hold the CS-mount ring and screw in the locking screw. Do not use excessive force.

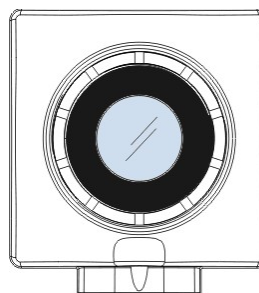


Figure 224: USB uEye LE with fix filter glass

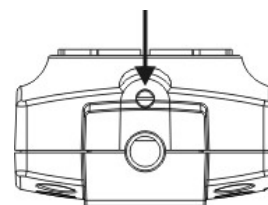


Figure 225: USB uEye LE - Locking screw for the adjusting ring

USB uEye LE with inserted filter glass (production since May 2009)

The adjusting ring of the *USB uEye LE* has 10 adjusting positions. For each adjusting position, the flange back distance is altered by +/- 50 μm .



To change the position of the adjusting ring, please proceed exactly in the following order:

1. Loosen the locking screw on the bottom of the camera (see illustration below).
2. Release the filter holder by turning it two revolutions counterclockwise (IDS special tool required, see also [USB uEye LE Accessories](#)).
3. Adjust the CS-mount ring. Ensure that the notch of the CS-mount ring is precisely aligned with the locking screw (when viewed from the camera front).
4. Hold the CS-mount ring and screw in the locking screw. Do not use excessive force.
5. Turn the filter holder clockwise until tight.

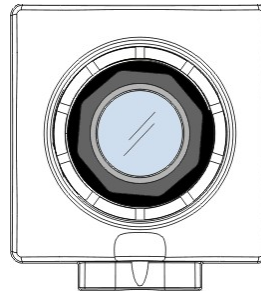


Figure 226: USB uEye LE with inserted filter glass

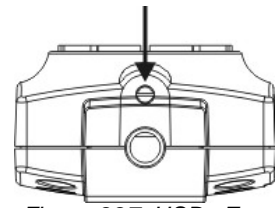


Figure 227: USB uEye LE - Locking screw for the adjusting ring

GigE uEye HE

The adjusting ring of the *GigE uEye HE* camera has 10 adjusting positions. For each adjusting position, the flange back distance is altered by +/- 50 µm.



To change the position of the adjusting ring, please proceed exactly in the following order:

1. Loosen the locking screw on the bottom of the camera (see illustration below).
2. Release the filter holder by turning it two revolutions counterclockwise (IDS special tool required, see also [GigE uEye HE Accessories](#)).
3. Adjust the C-mount ring. Ensure that the notch of the C-mount ring is precisely aligned with the locking screw (when viewed from the camera front).
4. Hold the C-mount ring and screw in the locking screw. Do not use excessive force.
5. Turn the filter holder clockwise until tight.

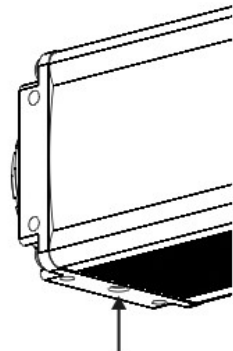


Figure 228: GigE uEye HE - Locking screw for the adjusting ring

Drawing of the uEye adjusting ring

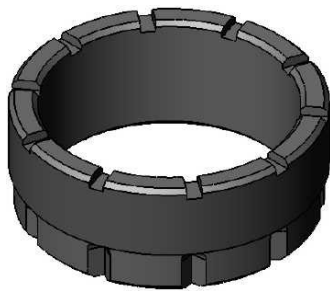


Figure 229: uEye flange adjusting ring (top view)

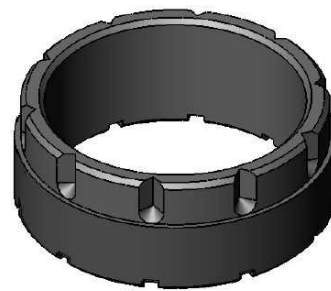


Figure 230: uEye flange adjusting ring (bottom view)

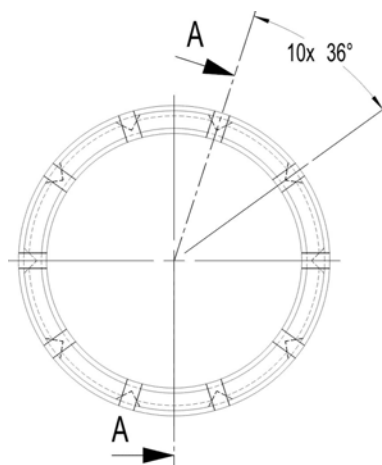


Figure 231: Rotation angle of adjustment ring

6.3.8.3 Maximum Immersion Depth for Lenses

Some C-mount lenses reach deep into the camera flange. This may cause the lens to push against the back of the filter glass inside the camera or even make it impossible to screw in the lens.

The table below indicates the maximum possible immersion depth for each *uEye* model. The actual immersion depth of a lens is given in the relevant data sheet. As lens parts with a small diameter are allowed to reach deeper into the camera flange, the immersion depths are specified based on the diameter.

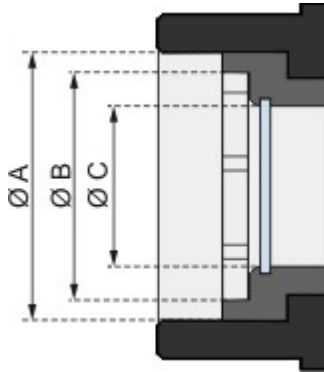


Figure 232: Camera front
(sectional view)

Camera	Type	Thread depth	Diameter at lens end (mm)	Max. immersion depth CMOS (mm)	Max. immersion depth CCD (mm)
USB uEye SE	C-mount	5 mm	24.0	9.4	8.4
			17.1	10.2	9.2
			14.1	10.7	9.7
USB uEye RE	C-mount	5 mm	24.0	8.9	7.9
			19.6	9.4	8.4
			17.1	10.2	9.2
			14.1	10.7	9.7
USB uEye LE	CS-mount	4 mm	24.0	6.1	-
			14.6	7.6	-
	C-mount with extension ring	4 mm *)	22.0 *)	11.1	-
			14.6 *)	12.6	-
	S-mount M12/M14	8.8 mm	M12/M14	with filter glass: 8.6 mm without filter glass: 12.0 mm	-
USB uEye ME	C-mount	5 mm	24.0	7.2	6.2
GigE uEye SE			20.4	9.7	8.7
GigE uEye RE			14.6	10.7	9.7
GigE uEye HE	C-mount	5 mm	24.0	6.9	5.9
			20.4	9.4	8.4
			14.6	10.4	9.4

*) May vary depending on the inside diameter of the extension ring used



The data given in the table include the following tolerances as a safety clearance:

- Immersion depth: 0.2 mm / 0.5 mm for GigE uEye HE
- Diameter: 0.2 mm

6.3.8.4 Position Accuracy of the Sensor

The following illustration shows the tolerance margins of the sensor position relative to the camera housing. A maximum error in all directions (rotation, translation) cannot occur simultaneously.

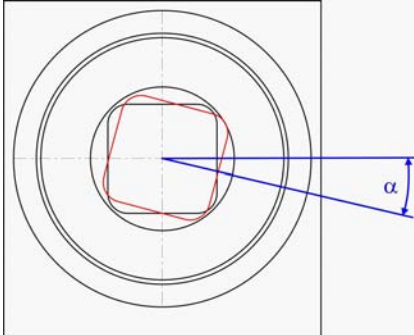
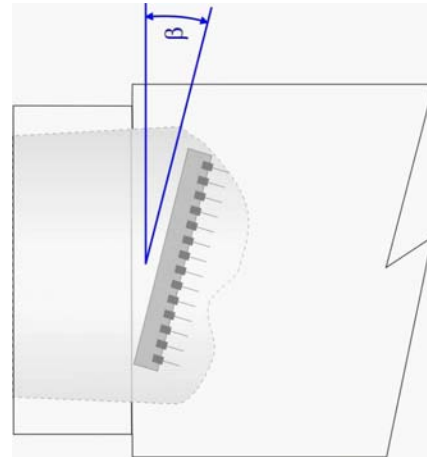


Figure 233: Position accuracy of the sensor



Position accuracy inside the camera housing, in each direction	±0.3	mm
Horizontal/vertical rotation (α)	±1.0	°
Translational rotation (β)	±1.0	°
Flange back distance	±0.05	mm



C-mount lenses can also be subject to inaccuracies of the flange back distance. The tolerance usually is ±0.05 mm. In some cases, however, the inaccuracies of camera and lens might add up, resulting in a total error > 0.05 mm.

6.3.9 Filter Glasses

6.3.9.1 Filter Types

Every *uEye* camera has a filter glass in the front flange to prevent the entry of dust particles. Color cameras by default use an IR cut filter, which is required to ensure correct color rendering. The default filter glass in monochrome cameras has no filter effect. Every camera model is available with different filter variants such as daylight cut filters (type DL). The filter type is given at the end of the *uEye model name*.

The following table shows an overview of the different optical filters used in the *uEye* cameras:

Filter type	Name	Refractive index (n_{Filter})	Glass type	Thickness (f)	Cut-off frequency	Non-reflective
IR cut filter (old)	BG	1,53	BG40	1 mm	650 nm	-
IRcut filter (new)	HQ	1,53	D263	1 mm	650 nm	On one side
Daylight cut filter	DL	1,53	RG665	1 mm	665 nm	-
Glass	GL	1,53	D263	1 mm	380 nm	On both sides



All *uEye* sensors have a D263 type cover glass. This glass is opaque to wavelengths below 330 nm.



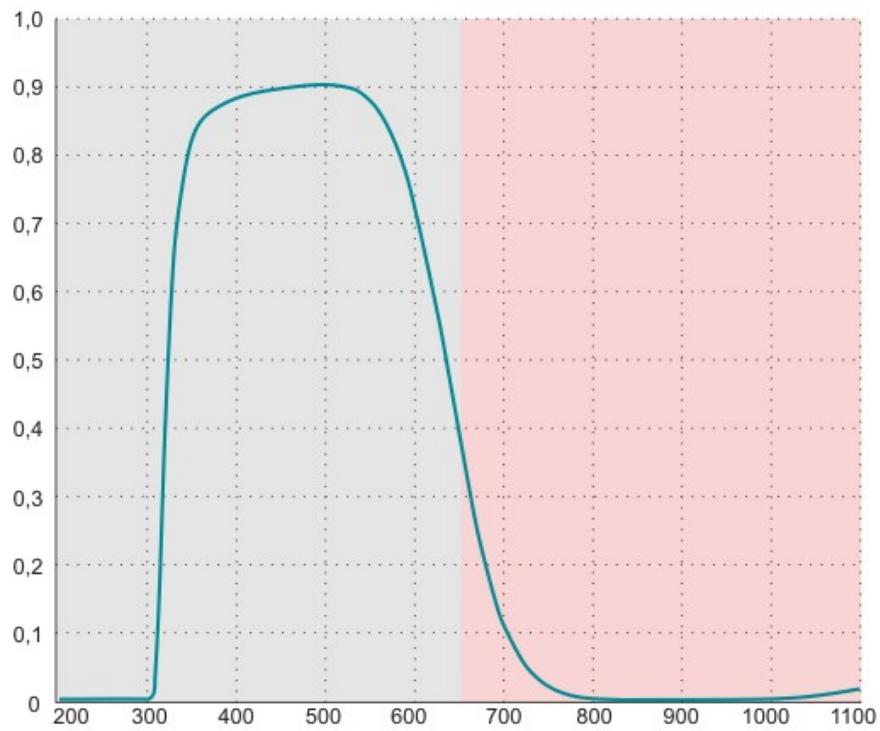
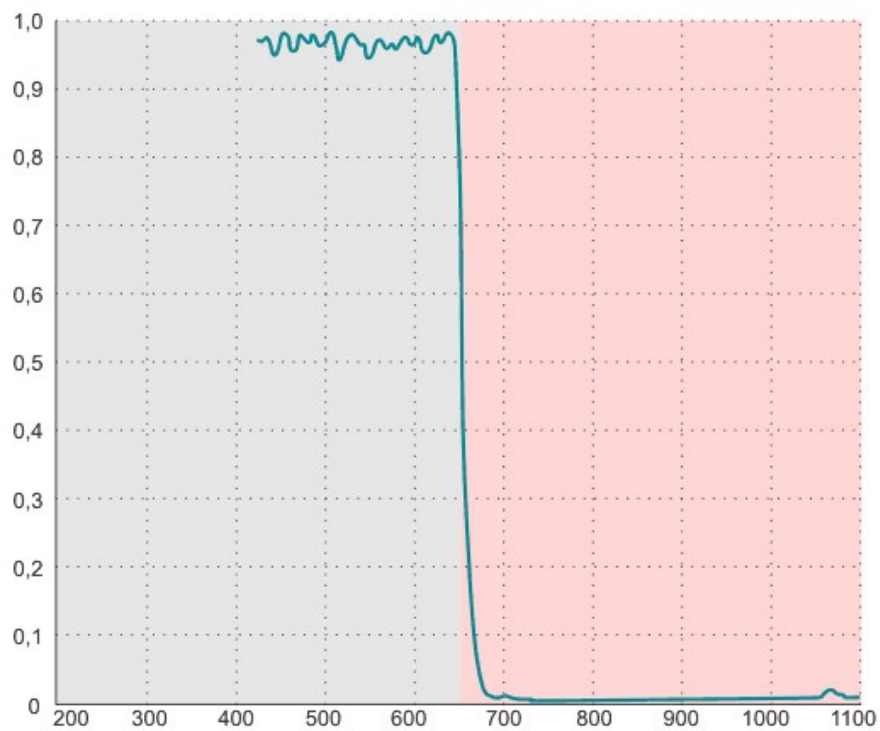
You can tell the filter type from the outside by its coloration:

- Reddish glass: HQ filter
- Bluish glass: BG filter
- Opaque glass: DL filter
- Plain glass: GL filter



New *uEye* color cameras use an IR cut filter of the type *HQ* by default. This filter offers an improved accuracy of the infrared content. *HQ* filters achieve a higher image brightness and better color rendering.

uEye drivers of version V3.24 and higher determine automatically which the IR filter is used in a camera. The corresponding color correction is selected automatically.

Infrared cut filter (type BG)*Figure 234: BG filter***Infrared cut filter (type HQ)***Figure 235: HQ filter*

Daylight cut filter (type DL)

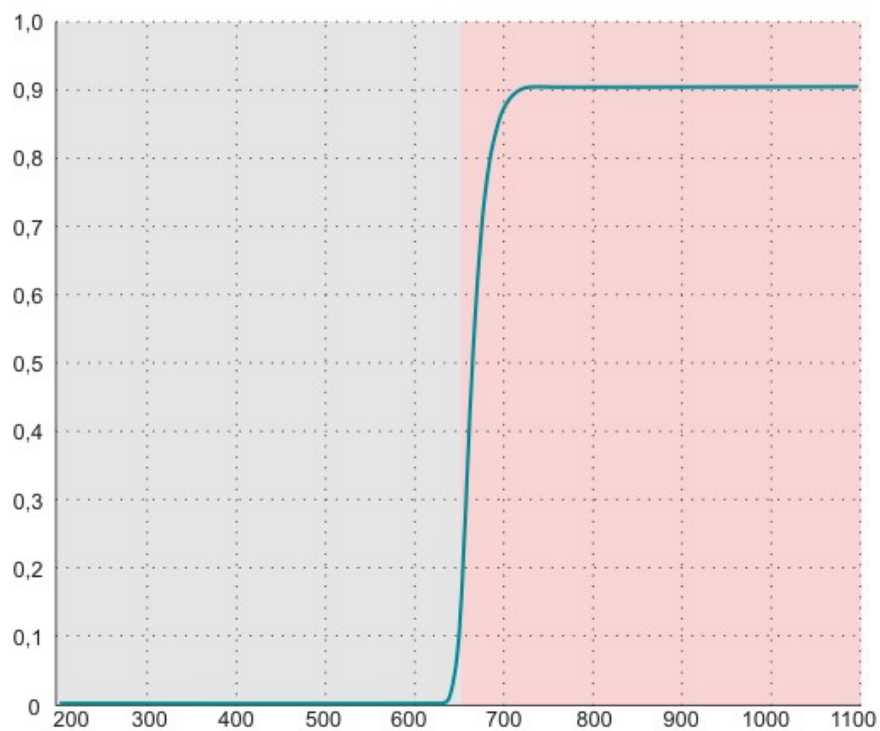


Figure 236: DL cut filter

Plain glass filter (type GL)

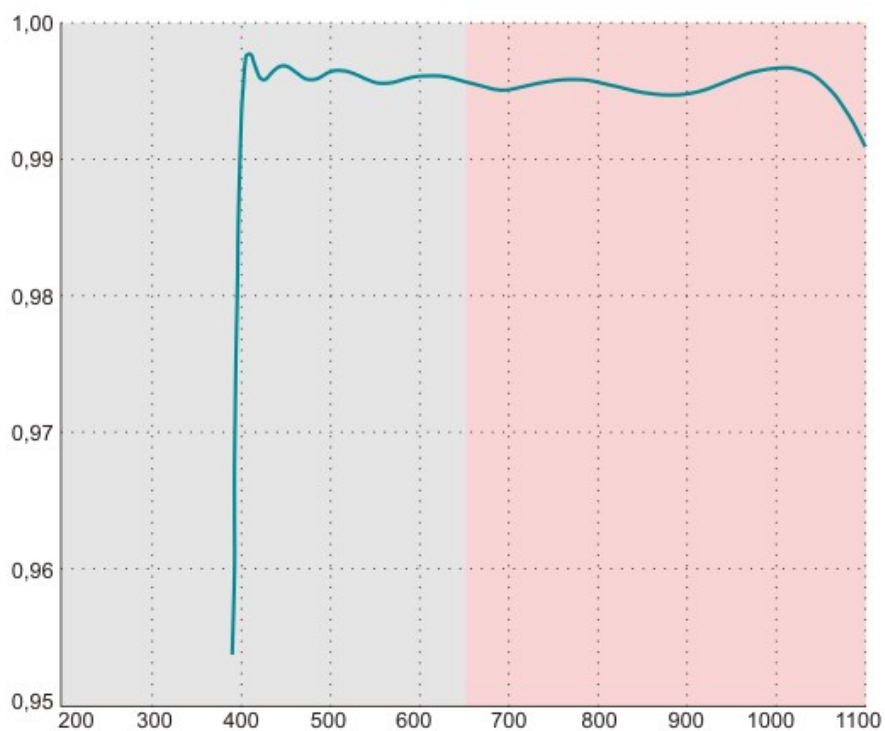


Figure 237: Glass filter

6.3.9.2 Mounting the Filter

The IR-cut filter of the *USB uEye ME* and *GigE uEye SE / RE / HE* is welded onto the threaded ring and mounted with it.



Figure 238: Threaded ring - top view



Figure 239: Threaded ring - bottom view

The threaded ring is screwed into the adjusting ring from the front with a torque of 0.2 Nm. A properly mounted threaded ring will seal off the sensor.



A special IDS tool is required for adjusting the threaded ring (see [GigE uEye HE Accessories](#)).

6.3.9.3 Cleaning the Filter Glasses

When handling the uEye camera with its lens removed, the filter glass can get soiled from the outside. This might be visible in the images that are captured. The filter glass should therefore be cleaned in that case.



It is strongly recommended to return the cameras to *Imaging Development Systems GmbH* for professional cleaning.

Imaging Development Systems GmbH is not liable for any damage resulting from cleaning the filter glasses. This even applies if the following instructions have been observed.

Instructions for cleaning uEye filter glasses

- The filter glasses may only be cleaned from the outside. If you remove the glasses, the sensor might get soiled. *Imaging Development Systems GmbH* is not liable for any damage to the sensor resulting from removal of the filter glasses.
- First, remove dirt particles on the glass using compressed air. Do not use compressed air from compressors or spray cans since it often contains oil droplets or droplets of other liquids. For best results, use purified nitrogen from nitrogen bottles.
- Only use lint-free wipes or cotton-free swabs for cleaning. Never touch the filter glasses with your bare fingers because often, fingerprints cannot be removed completely afterwards.
- We recommend to use pure alcohol for cleaning. 100% isopropyl alcohol evaporates without leaving any residues. Only add small quantities of alcohol to the wipe. Never pour alcohol directly onto the camera.



Never use cleaning agents containing acetone for cleaning the filter glasses!
Acetone may damage the filter glass coating and may deteriorate the optical quality of the glasses.

Cameras with fixed filter glass (USB uEye SE / RE)

Use a wipe to wipe off dirt particles in a single sweep beyond the edge of the filter glass (see figure below).



Figure 240: Cleaning fixed uEye filter glasses

Cameras with replaceable filter glass (USB uEye ME / LE, GigE uEye SE / RE / HE)

Use a wipe to wipe off dirt particles in a circular sweep (see figure below).



Figure 241: Cleaning interchangeable uEye filter glasses

6.3.10 Ambient Conditions



Avoid high air humidity levels and rapid temperature changes when using uEye cameras. Temperatures below +4 °C (39 °F) combined with excessive relative air humidity levels can cause icing.



At ambient temperatures above 45 °C (113 °F), the image quality could be reduced due to increased thermal noise. It is recommended to mount the camera to a heat-dissipating unit when high ambient temperatures prevail.



The temperature values given above refer to the ambient temperature. The camera temperature that can be queried for some uEye models is usually higher than the ambient temperature and can reach up to 70 °C (158 °F).

USB uEye SE/ME/LE, GigE uEye SE/HE

	Min.	Max.	°
Ambient temperature	0 32	45 113	°C °F
Storage temperature	-20 -4	60 140	°C °F
Relative humidity ^{*1)}	20	80	%

^{*1)} Non-condensing

GigE/USB uEye RE

	Min.	Max.	°
Ambient temperature	0 32	45 113	°C °F
Storage temperature	-20 -4	60 140	°C °F
Relative humidity for <i>uEye RE</i> ^{*2)}	0	100	%

^{*2)} Only if *uEye RE* lens tube and IP65/67 cables are used

Non-condensing means that the relative air humidity must be below 100%. Otherwise, moisture will form on the camera surface. If, for example, air has a relative humidity of 40% at 35°C, the relative humidity will increase to over 100% if the air cools down to 19.5°C; condensation begins to form.



Avoid high air humidity levels and rapid temperature changes when using *uEye* cameras.

Vibration and shock resistance

Vibration and shock resistance of the *USB uEye* and *GigE uEye* cameras were tested as specified in DIN EN 60068-2-6(1996-05), DIN EN 60068-2-27(1995-03) and DIN EN 60068-2-29 (1995-03). The mechanical shock was at 80 g; the vibration testing was performed with sinusoidal vibration at a frequency between 30 Hz-500 Hz and an amplitude of 10 g.

6.3.10.1 Definition of IP Protection Classes

The housing of the *USB uEye RE* models complies with the IP (*Ingress Protection*) 65/67 standard. The housings of the other *uEye* models comply with IP 30. These standards are defined as follows:

First digit

3	Protection against the ingress of small particles (diameter > 2.5 mm)
6	Dust-proof No ingress of dust at a vacuum of 20 mbar in the housing

Second digit

0	<i>No special protection against liquids</i>
5	<i>Protected from jets of water</i> Test conditions: Using a jet nozzle with an inside diameter of 6.3 mm, a jet of water at a volume flow of 12.5 liters per minute is applied to the housing from all directions at a distance of 2.5 - 3 meters. The testing period is at least 3 minutes.
7	<i>Protected against the effects of temporary immersion in water</i> Test conditions: The housing is dipped completely under water in a dip tank. The submerged depth is 30 cm and the testing period is 20 seconds. Water may not intrude in a quantity which causes harmful effects while the housing is dipped in water under standardized pressure and time conditions.

6.4 Electrical Specifications

6.4.1 Digital Inputs/Outputs

Depending on the model, *uEye* cameras have one or more digital inputs and outputs designed for different purposes.

Camera model	Digital inputs	Digital outputs	General purpose I/Os (GPIOs), Other
<i>USB uEye SE</i>	1 (opto coupler)	1 (opto coupler)	-
<i>USB uEye RE</i>	1 (opto coupler)	1 (opto coupler)	-
<i>USB uEye LE</i> housing version	-	-	-
<i>USB uEye LE</i> board level version	1 (TTL)	1 (TTL)	2 (TTL)
<i>GigE uEye SE</i>	1 (opto coupler)	1 (opto coupler)	-
<i>GigE uEye RE</i>	1 (opto coupler)	1 (opto coupler)	-
<i>GigE uEye HE</i>	1 (opto coupler)	1 (opto coupler)	2 (TTL) 1 RS-232

6.4.2 Camera EEPROM Specifications

uEye cameras have an EEPROM memory where the camera manufacturer, type, and serial number are stored. A 64-byte memory space can be used freely by the user.

EEPROM Specifications	
Data retention	10 years
Read/write cycles	100,000
Size of user data space	64 bytes

6.4.3 USB uEye SE

6.4.3.1 Pin Assignment

9-pin micro D-Sub socket

Pin	Description
1	Digital output (-)
2	Digital input (+)
3	Shielding
4	USB power supply (VCC) 5 V
5	USB ground (GND)
6	Digital output (+)
7	Digital input (-)
8	USB data (+)
9	USB data (-)

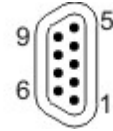


Figure 242: USB uEye SE - Micro D-Sub socket male, camera rear view

Pin assignment of the uEye special cable for USB 2.0, trigger and flash

Pin	Description	Cable color
1	Digital output (-)	green
2	Digital input (+)	white
6	Digital output (+)	yellow
7	Digital input (-)	brown

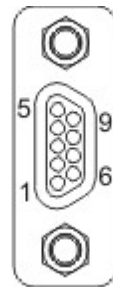


Figure 243: USB uEye SE - Micro D-Sub connector female, connecting side view

For a comprehensive list of all cables and connectors available for *USB uEye SE* cameras, please refer to the [USB uEye SE Accessories](#) section.

6.4.3.2 Digital Input Wiring (Trigger)

Digital input specifications

USB board revision *)	1.2		2.0 or higher		
	Min.	Max.	Min.	Max.	
Level low	0	2	0	2	V
Level high	9	24	5	24	V
Voltage range	0	30	0	30	V
Trigger pulse width (edge)	100	-	100	-	µs
Trigger edge steepness	35		35		V/ms
Breakdown voltage		50		50	V
Input current	10	-	10	-	mA



*) For information on how to determine the USB board revision, please refer to the [USB uEye SE Driver Compatibility](#) chapter.

For interpreting the trigger signal, either the positive or the negative edge can be used. The digital input is galvanically isolated using an opto coupler to protect the camera and the PC against surges. Only DC voltages may be applied to the digital input.

Digital input wiring

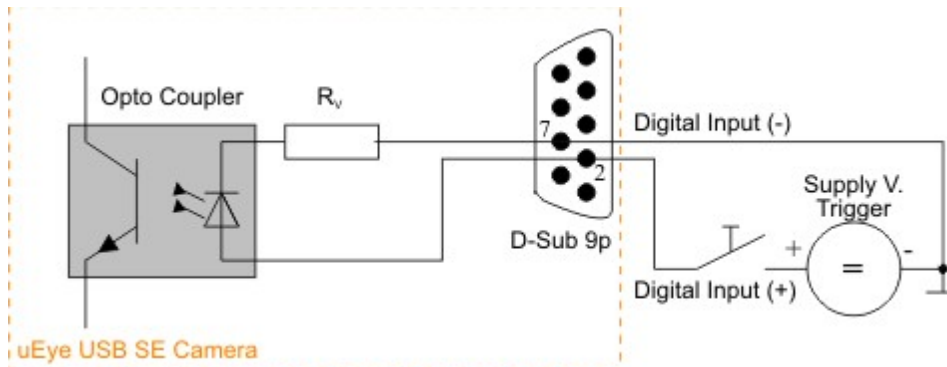


Figure 244: Wiring of the trigger connector

6.4.3.3 Digital Output Wiring (Flash)

Digital output specifications

USB board revision *)	1.2	2.0 or higher	
	Max.	Max.	
Output current (short-time)	50	500	mA
Output current (permanent)	15	150	mA
Output voltage	30	30	V
Breakdown voltage	50	50	V
Collector power dissipation	100	125	mW



*) For information on how to determine the USB board revision, please refer to the [USB uEye SE Driver Compatibility](#) chapter.

The digital output is galvanically isolated using an opto coupler to protect the camera and the PC against surges. Only DC voltages may be applied to the digital input.

The output of the opto coupler can be used as an open collector or open emitter output. This means that the output signal can be connected to ground or to the supply voltage. The output signal is active if the collector-emitter switch is closed (software setting: *Flash high active*, see also the [Camera Properties: Input/Output](#) section).

Digital output wiring

The following figures show examples of how the digital output is wired.

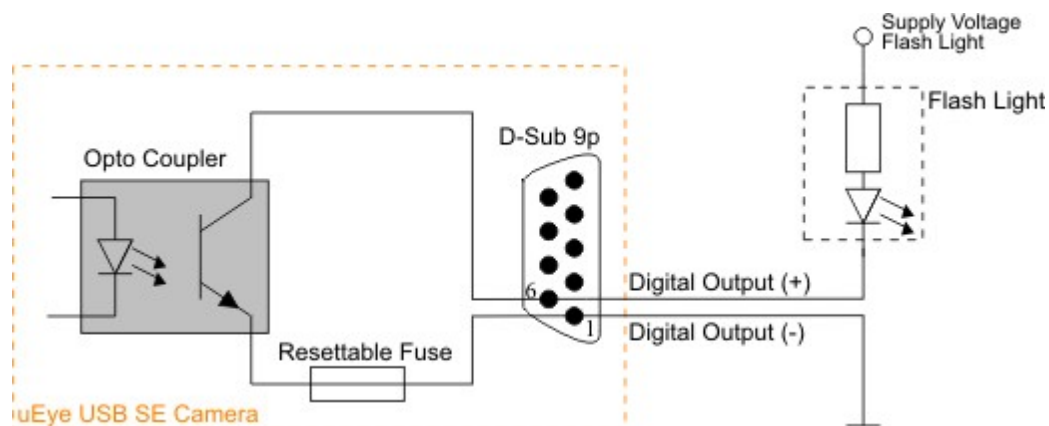


Figure 245: Wiring of the digital output as an open collector output (rev. 1.2)

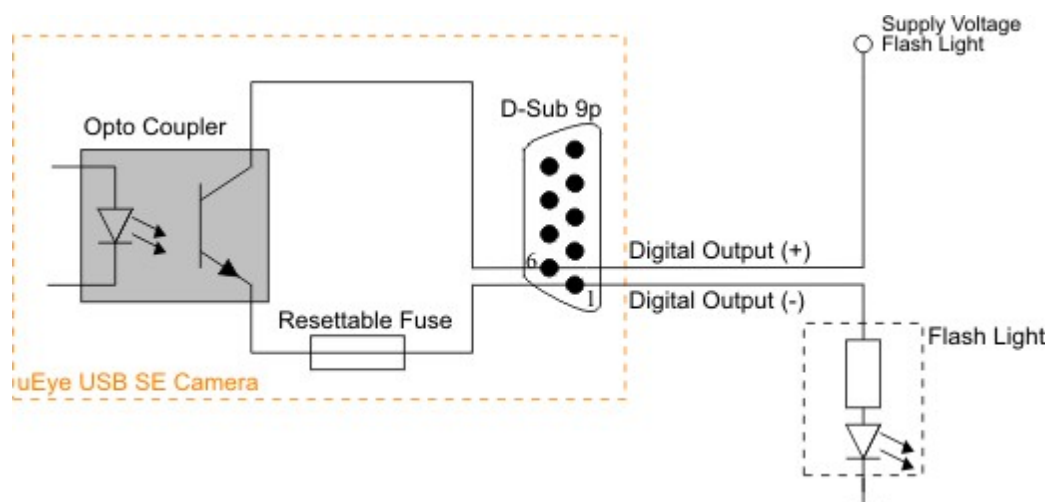


Figure 246: Wiring of the digital output as an open emitter output (rev. 1.2)

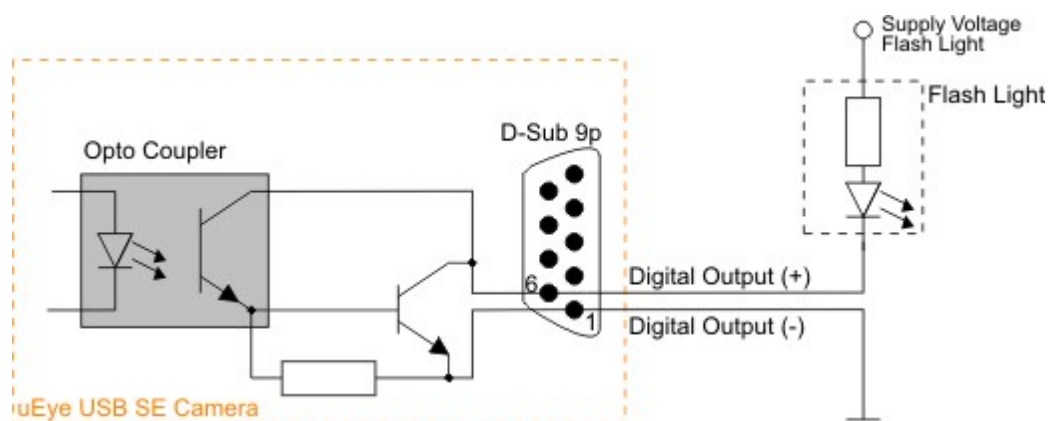


Figure 247: Wiring of the digital output as an open collector output (rev. 2.0)

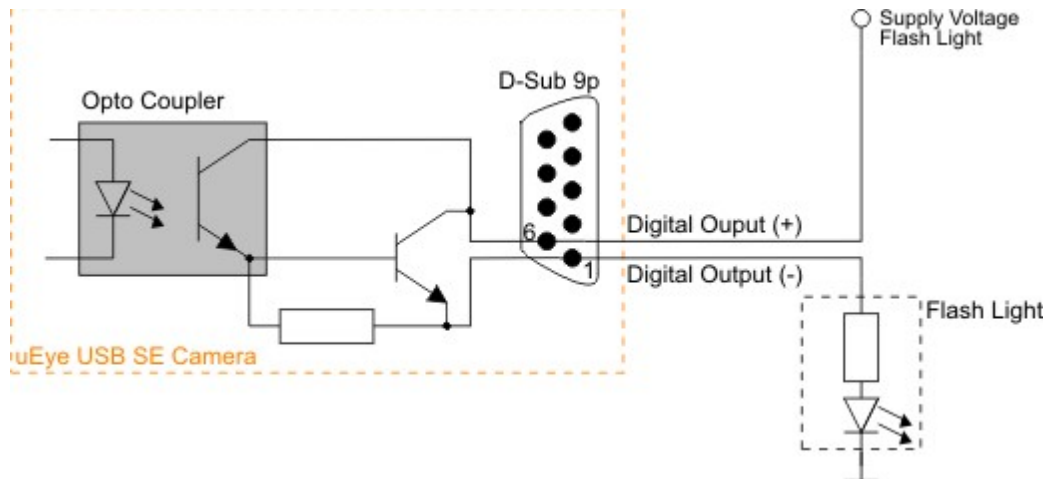


Figure 248: Wiring of the digital output as an open emitter output (rev. 2.0)

6.4.4 USB uEye ME

6.4.4.1 Pin Assignment of the I/O Connector

6-pin Hirose connector

Pin	Description
1	Ground (GND)
2	USB supply voltage (VCC) 5 V
3	Digital input with opto coupler (-)
4	Digital input with opto coupler (+)
5	Digital output with opto coupler (+)
6	Digital output with opto coupler (-)



Figure 249: USB uEye ME - Hirose connector, camera rear view

Pin assignment of the 6-wire connecting cable (6-pin Hirose connector)

Pin	Description	Cable color
1	Ground (GND)	gray
2	USB supply voltage (VCC) 5 V	pink
3	Digital input (-)	brown
4	Digital input (+)	white
5	Digital output (+)	yellow
6	Digital output (-)	green

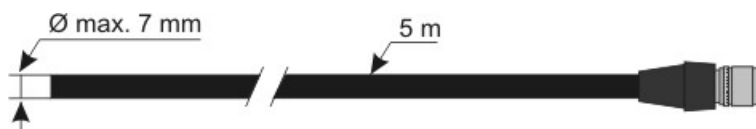


Figure 250: USB uEye ME 6-wire cable

6.4.4.2 Digital Input Wiring (Trigger)

Digital input specifications

	Min.	Max.	
Level low	0	2	V
Level high	5	24	V
Voltage range	0	30	V
Trigger pulse width (edge)	100	-	µs
Trigger edge steepness	35		V/ms
Breakdown voltage		50	V
Input current	10	-	mA

For interpreting the trigger signal, either the positive or the negative edge can be used. The digital input is galvanically isolated using an opto coupler to protect the camera and the PC against surges. Only DC voltages may be applied to the digital input.

Digital input wiring

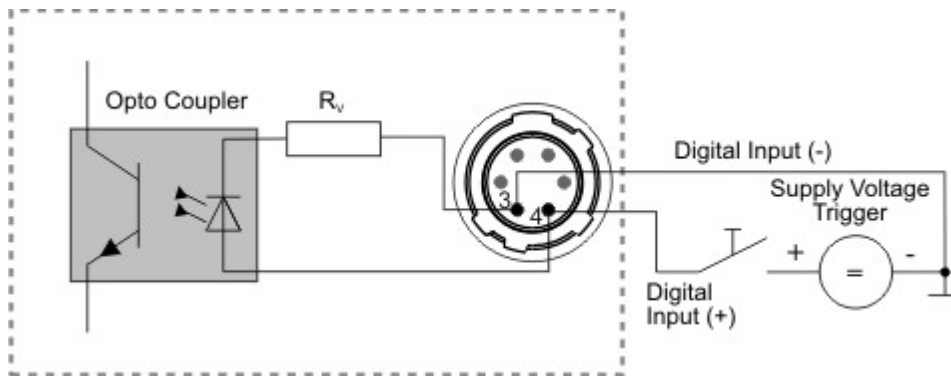


Figure 251: USB uEye ME - Wiring of the trigger connector

6.4.4.3 Digital Output Wiring (Flash)

Digital output specifications

	Max.	
Output current (short-time)	500	mA
Output current (permanent)	150	mA
Output voltage	30	V
Breakdown voltage	50	V
Collector power dissipation	125	mW

The digital output is galvanically isolated using an opto coupler to protect the camera and the PC against surges. Only DC voltages may be applied to the digital input.

The output of the opto coupler can be used as an open collector or open emitter output. This means that the output signal can be connected to ground or to the supply voltage. The output signal is active if the collector-emitter switch is closed (software setting: *Flash high active*, see also the [Camera Properties: Input/Output](#) section).

Digital output wiring

The following figures show examples of how the digital output is wired.

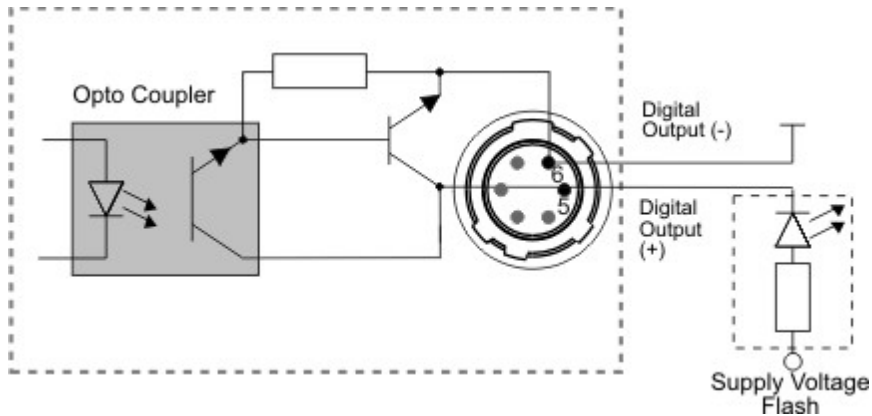


Figure 252: USB uEye ME - Wiring of the digital output as an open collector output

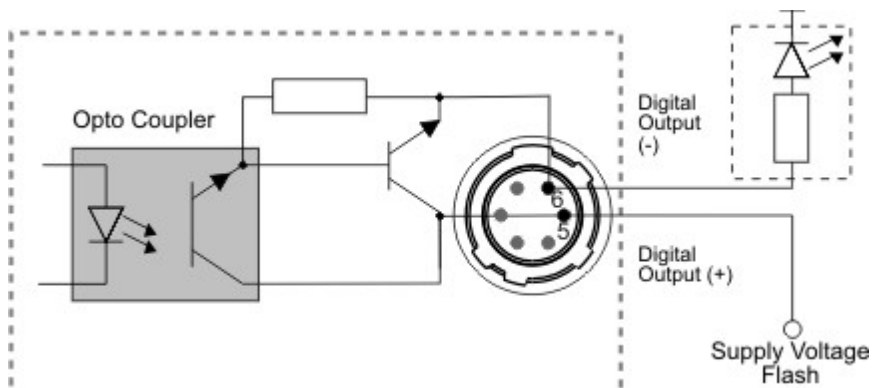


Figure 253: USB uEye ME - Wiring of the digital output as an open emitter output

6.4.4.4 PCB Version Wiring

10 pin connector (PCB version only)

Pin	Beschreibung
1	USB power supply (VCC) 5 V
2	USB ground (GND)
3	Digital input without opto coupler (+)
4	Digital output without opto coupler (+)
5	Power supply of the internal voltage transformer 3.3 V
6	USB ground (GND)
7	Programmable input/output (General Purpose I/O) 1
8	Programmable input/output (General Purpose I/O) 2
9	I ² C bus clock signal
10	I ² C bus data signal

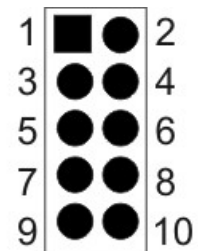


Figure 254: USB uEye LE PCB version - I/O connector



The 10 pin I/O connector of the *USB uEye ME* can be provided with a 2 x 5-pin connecting plug with a 2.54 mm (0.1") lead spacing.
IDS shall not be liable for any damage to the camera or connected devices arising from installation of the connecting plug.



The General Purpose I/Os are not potential-free and have no protective circuits.



If the 3.3 V and the 5 V power supplies are used simultaneously, please observe that the maximum power available from the USB bus is 2.5 W. Choose the overall current in such a way that the maximum power of 2.5 W will not be exceeded.

Max. admissible current with only one power output used	
USB power supply 5 V	150 mA
Power supply of the internal voltage transformer 3.3 V	150 mA

The following figures illustrate wiring examples for the *USB uEye ME* 10 pin connector.

Digital input wiring

	Min.	Max.	
Level Low	0	0.8	V
Level High	2.4	5.25 V	V

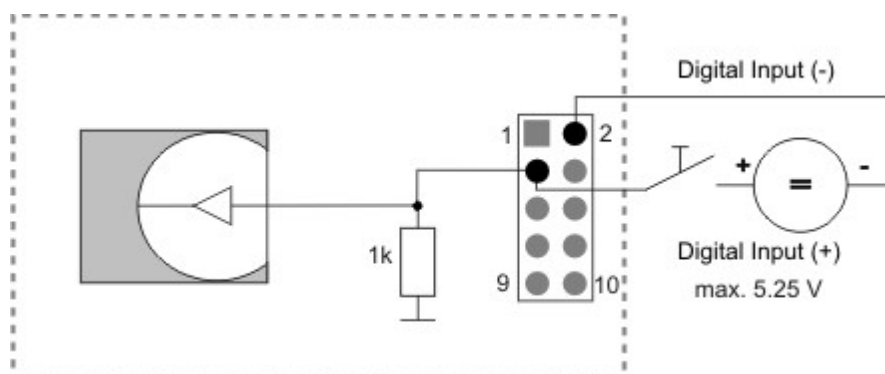


Figure 255: USB uEye ME PCB version - Wiring of the digital input

Digital output wiring

	Min.	Max.	
Level Low	0	0.4	V
Level High	2.4	3.3 V	V
Output current	0	4	mA

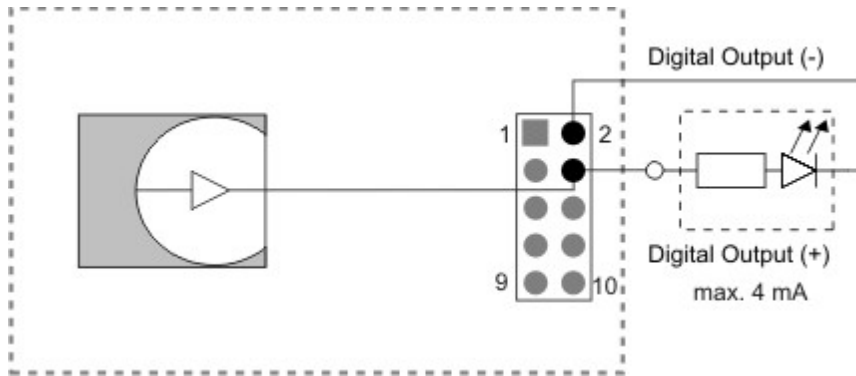


Figure 256: USB uEye ME PCB version - Wiring of the digital output

Programmable input/output (GPIO) wiring

The two GPIOs (General Purpose I/O) can be used as inputs or outputs. This selection is made by software using the corresponding SDK API functions. Please observe the following criteria:

	Min.	Max.	
Output Level <i>Low</i>	0	0.4	V
Output Level <i>High</i>	2.4	3.3 V	V
Input Level <i>Low</i>	0	0.8	V
Input Level <i>High</i>	2.4	5.25	V
Output current	0	4	mA

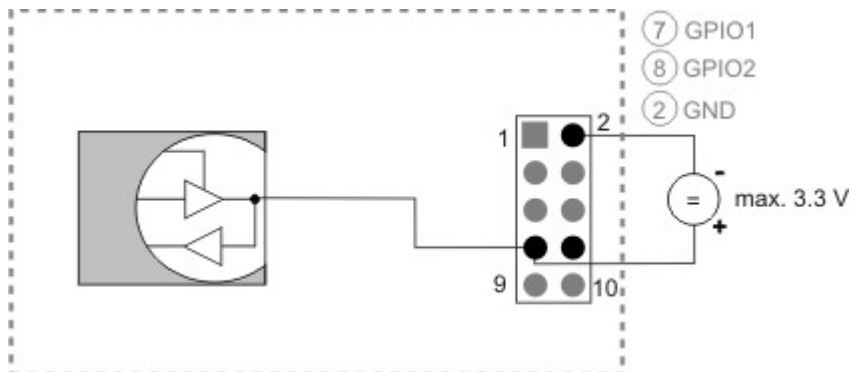


Figure 257: USB uEye ME - GPIO wired as an input

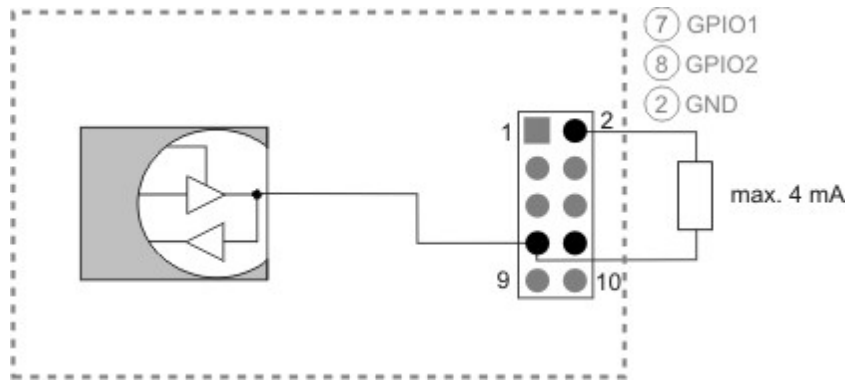
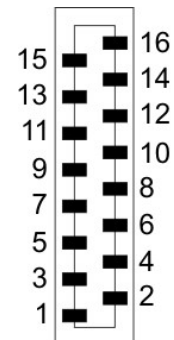


Figure 258: USB uEye ME - GPIO wired as an output

16 pin ZIF socket connector (Type Omron XF2J-1624-11A, PCB version only)

The drawing in the [Mechanical Data: uEye ME](#) chapter shows the position of pin 1 on the PCB.

Pin	Beschreibung
1	Control signal green LED
2	Control signal red LED
3	I ² C bus data signal
4	I ² C bus clock signal
5	USB power supply (VCC) 5 V
6	Ground (GND)
7	USB bus data signal D-
8	(not used)
9	USB bus data signal D+
10	Ground (GND)
11	Programmable input/output (General Purpose I/O) 1
12	Power supply of the internal voltage transformer 3.3 V
13	Digital output with opto coupler (-)
14	Digital output with opto coupler (+)
15	Digital input with opto coupler (+)
16	Digital input with opto coupler (-)

Figure 259:
Pinbelegung
ZIF-Sockel
(16-polig)

6.4.5 USB uEye RE

6.4.5.1 USB Connector

5 pin Binder plug (USB connector)

Pin	Description
1	USB data (+)
2	USB ground (GND)
3	Shielding
4	USB power supply (VCC)
5	USB data (-)

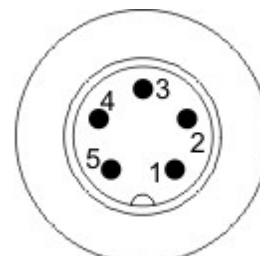


Figure 260: USB uEye RE - Binder USB socket male, camera rear view

6.4.5.2 I/O Connector

4 pin Binder socket (trigger connector)

Pin	Description
1	Digital output (+)
2	Digital output (-)
3	Digital input (+)
4	Digital input (-)

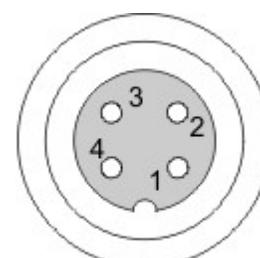


Figure 261: USB uEye RE - Binder I/O connector female, camera rear view

Color coding for USB uEye RE trigger cable

Color	Assignment
white	Digital input (+)
brown	Digital input (-)
green	Digital output (-)
yellow	Digital output (+)

For a comprehensive list of all cables and connectors available for *USB uEye RE* cameras, please refer to the [USB uEye RE Accessories](#) section.

6.4.5.3 Digital Input Wiring (Trigger)

Digital input specifications

	Min.	Max.	
Level low	0	2	V
Level high	5	24	V
Voltage range	0	30	V
Trigger pulse width (edge)	100	-	μ s
Trigger edge steepness	35		V/ms
Breakdown voltage		50	V
Input current	10	-	mA

For interpreting the trigger signal, either the positive or the negative edge can be used. The digital input is galvanically isolated using an opto coupler to protect the camera and the PC against surges. Only DC voltages may be applied to the digital input.

Digital input wiring

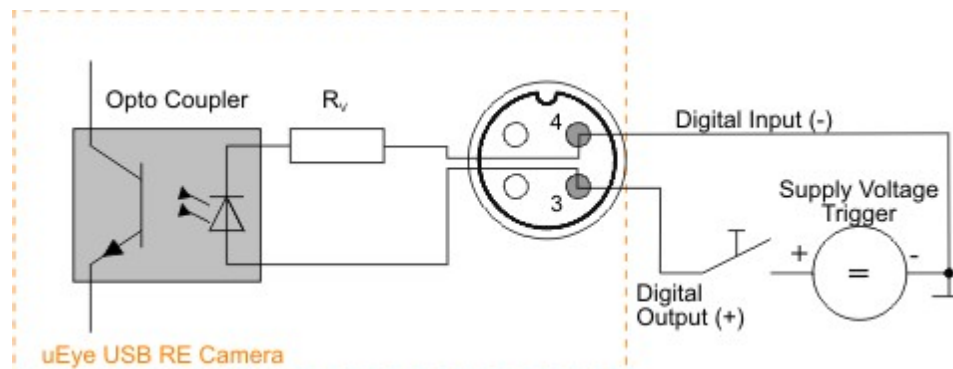


Figure 262: Wiring of the trigger connector

6.4.5.4 Digital Output Wiring (Flash)

Digital output specifications

	Max.	
Output current (short-time)	500	mA
Output current (permanent)	150	mA
Output voltage	30	V
Breakdown voltage	50	V
Collector power dissipation	125	mW

The digital output is galvanically isolated using an opto coupler to protect the camera and the PC against surges. Only DC voltages may be applied to the digital input.

The output of the opto coupler can be used as an open collector or open emitter output. This means that the output signal can be connected to ground or to the supply voltage. The output signal is active if the collector-emitter switch is closed (software setting: *Flash high active*, see also the [Camera Properties: Input/Output](#) section).

Digital output wiring

The following figures show examples of how the digital output is wired.

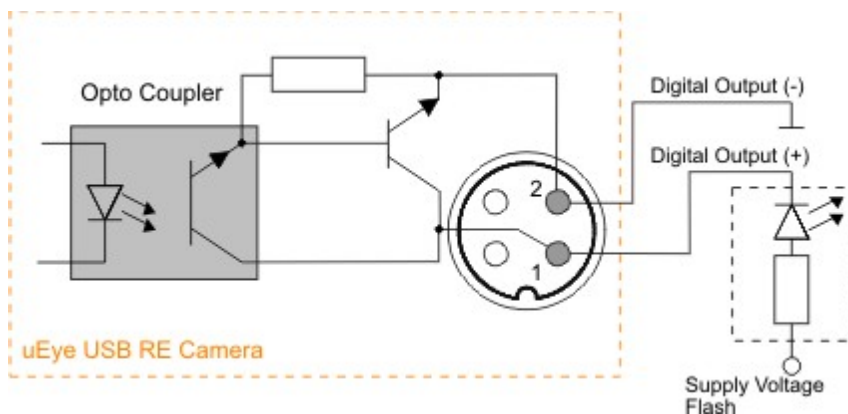


Figure 263: Wiring of the digital output as an open collector output

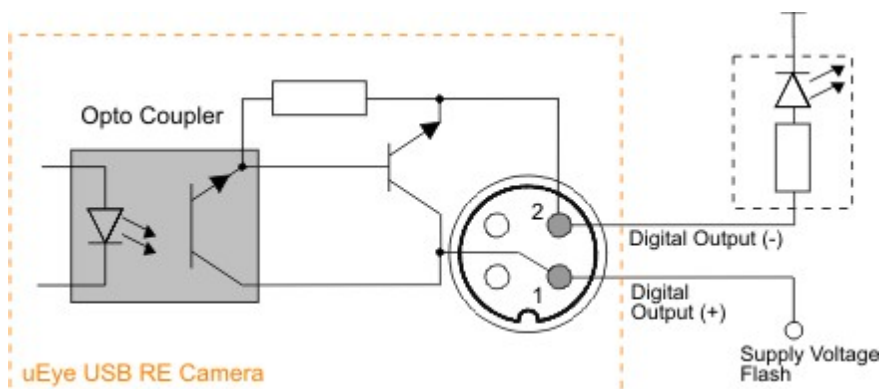


Figure 264: Wiring of the digital output as an open emitter output

6.4.6 USB uEye LE

6.4.6.1 Pin Assignment of the USB Connector

Additional USB connector on the USB uEye LE PCB version



The following specifications apply to the board-level versions of the *USB uEye LE*. The inputs/outputs mentioned are not accessible in the housing versions.

Pin	Description	USB cable (standard color)
1	Shielding	-
2	Power supply (VCC) 5 V	red
3	Data (+)	green
4	Ground (GND)	black
5	Shield	-
6	Not connected	-
7	Data (-)	white

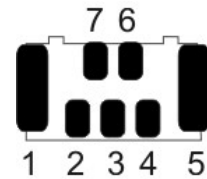


Figure 265: USB uEye LE - Pin assignment of the USB connector

Additional USB connector on the UI-149xLE PCB version



The board-level version of the UI-1490x-LE has not been prepared for use of a vertical USB connector. The board provides five additional connection points to which a USB cable can be soldered if required.

Pin	Description	USB cable (standard color)
1	Shielding	-
2	Data (+)	green
3	Data (-)	white
4	Power supply (VCC) 5 V	red
5	Ground (GND)	black

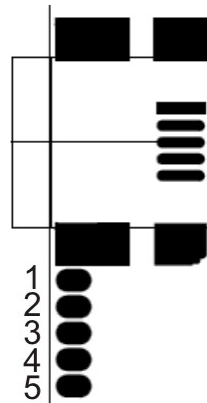


Figure 266: UI-149xLE PCB version - additional USB connector

6.4.6.2 Pin Assignment of the I/O Connector

10 pin connector (PCB version only)

Pin	Beschreibung
1	USB power supply (VCC) 5 V
2	USB ground (GND)
3	Digital input without opto coupler (+)
4	Digital output without opto coupler (+)
5	Power supply of the internal voltage transformer 3.3 V or 3.0 V ^{*1)}
6	USB ground (GND)
7	Programmable input/output (General Purpose I/O) 1
8	Programmable input/output (General Purpose I/O) 2
9	I ² C bus clock signal
10	I ² C bus data signal

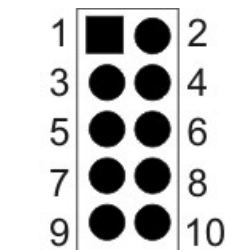


Figure 267: USB uEye
LE PCB version - I/O
connector

^{*1)} This voltage depends on the supply voltage required for the sensor used (see [table below](#))



The 10 pin I/O connector of the *USB uEye LE* can be provided with a 2 x 5-pin connecting plug with a 2.54 mm (0.1") lead spacing.
IDS shall not be liable for any damage to the camera or connected devices arising from installation of the connecting plug.



If the 3.3 V and the 5 V power supplies are used simultaneously, please observe that the maximum power available from the USB bus is 2.5 W. Choose the overall current in such a way that the maximum power of 2.5 W will not be exceeded.

Max. admissible current with only one power output used	
USB power supply 5 V	150 mA
Power supply of the internal voltage transformer (3.0 V or 3.3 V) ^{*1)}	200 mA

^{*1)} This voltage depends on the supply voltage required for the sensor used (see [table below](#))

GPIO specifications

The two GPIOs (General Purpose I/O) can be used as inputs or outputs. This selection is made by software using the corresponding SDK API functions. Please observe the following criteria:

- Input: 3.3 V LVTTTL, max. 3.3 V input voltage
- Output: 3.3 V LVTTTL, max. 4 mA

	Min.	Max.	
Signal level <i>Low</i>	0	0.4	V
Signal level <i>High</i>	2.4	3.0 V or 3.3 V ^{*1)}	V
Output current	0	4	mA

^{*1)} This voltage depends on the supply voltage required for the sensor used (see [table below](#))

Internal supply voltage by sensor type

3.0 V	3.3 V
164xLE	122xLE
155xLE	154xLE
148xLE	146xLE

I2C operation

From driver version 3.20, the I2C bus is operated with a clock frequency of approx. 300 kHz. For earlier versions, the clock frequency is 100 kHz.

Multi master mode is not allowed on the I2C bus while the *USB uEye LE* camera is used.

6.4.6.3 Digital Input Wiring (Trigger)**Digital input specifications**

	Min.	Max.	
Level low	0	0.8	V
Level high	2.0	5.25	V



The digital input of the *USB uEye LE* is not potential-free and has no protective circuits. Due to hardware limitations, the *USB uEye LE* can only be triggered on the falling edge.

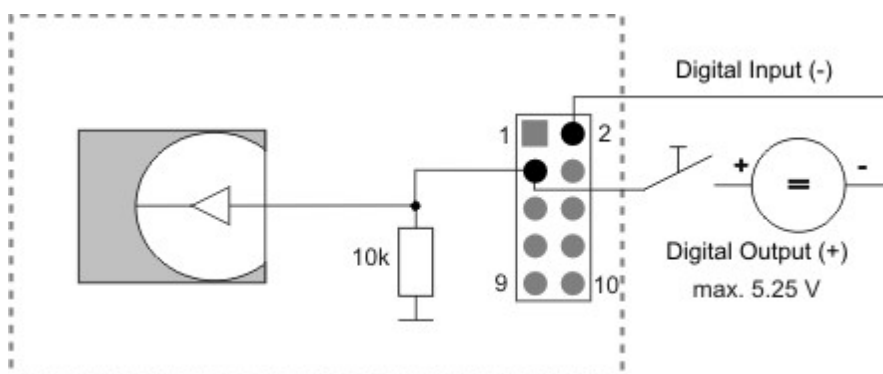
Digital input wiring

Figure 268: USB uEye LE PCB version - Wiring of the digital input

6.4.6.4 Digital Output Wiring (Flash)

Digital output specifications

	Min.	Max.	
Signal level <i>Low</i>	0	0.4	V
Signal level <i>High</i>	2.4	3.0 V or 3.3 V ^{*1)}	V
Output current	0	4	mA

^{*1)} This voltage depends on the supply voltage required for the sensor used (see table below)



The digital output of the *USB uEye LE* is not potential-free and has no protective circuits.

Internal supply voltage by sensor type

3.0 V	3.3 V
164xLE	122xLE
155xLE	154xLE
148xLE	146xLE

Digital output wiring

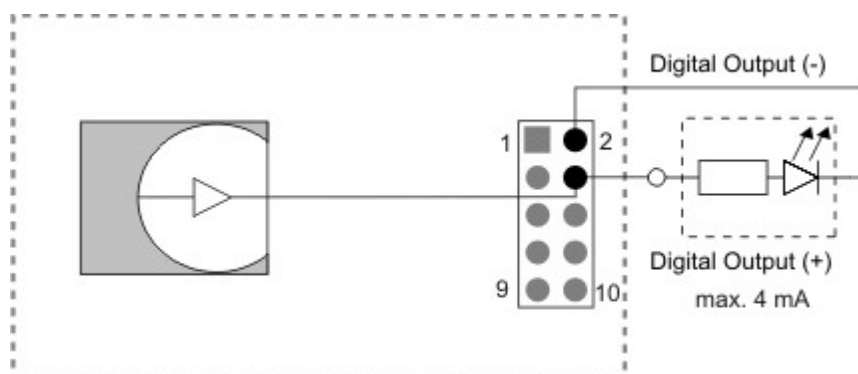


Figure 269: USB uEye LE PCB version - Wiring of the digital output

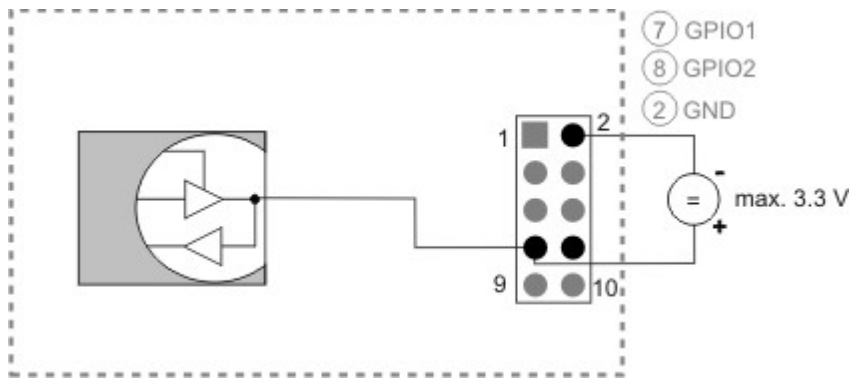
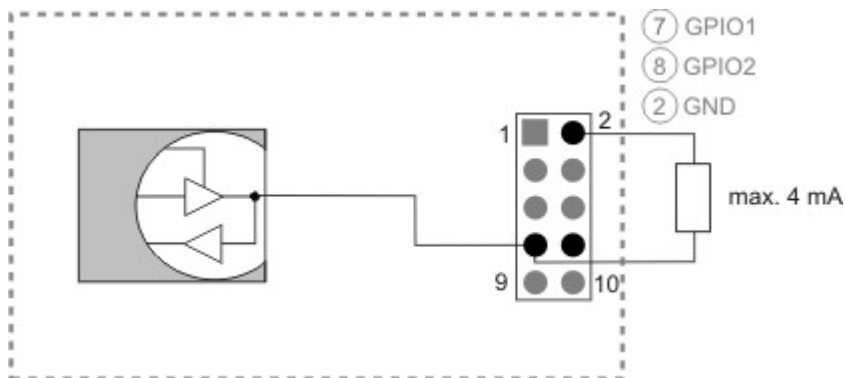
6.4.6.5 General Purpose I/O Wiring



The General Purpose I/Os are not potential-free and have no protective circuits.

GPIO wiring

The following figures illustrate GPIO wiring examples.

*Figure 270: GPIO wired as an input**Figure 271: GPIO wired as an output*

6.4.7 GigE uEye SE

6.4.7.1 Pin Assignment of the GigE Connector (RJ45)

8-pin RJ45 socket

Pin	Designation 100BASE-TX	Designation 1000BASE-TX
1	Tx+	BI_DA+
2	Tx-	BI_DA-
3	Rx+	BI_DB+
4		BI_DC+
5		BI_DC-
6	Rx-	BI_DB-
7		BI_DD+
8		BI_DD-

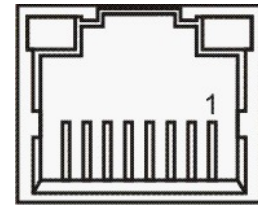


Figure 272: GigE uEye SE - RJ45 socket, camera rear view



The RJ45 socket of the *GigE uEye SE* complies with the IEC 60603-7 standard.

6.4.7.2 Pin Assignment of the I/O Connector

6-pin Hirose connector

Pin	Description
1	Ground (GND)
2	Power supply (VCC) 12 V
3	Digital input (-)
4	Digital input (+)
5	Digital output (+)
6	Digital output (-)



Figure 273: GigE uEye SE - Hirose connector male, camera rear view

Pin assignment of the 6-wire connecting cable (6-pin Hirose connector)

Pin	Description	Cable color
1	Ground (GND)	gray
2	Power supply (VCC) 12 V	pink
3	Digital input (-)	brown
4	Digital input (+)	white
5	Digital output (+)	yellow
6	Digital output (-)	green

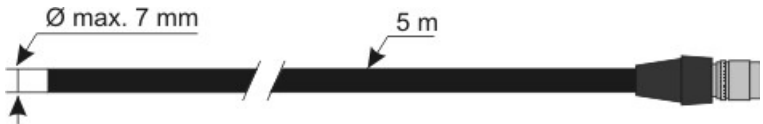


Figure 274: GigE uEye SE 6-wire cable without AC adapter
(AD.0040.2.18300.00)

Pin assignment of the 2+4-wire connecting cable (6-pin Hirose connector)

Pin	Cable	Description	Cable color
1	A1	Ground (GND)	white
2	A2	Power supply (VCC) 12 V	brown
3	B1	Digital input (-)	brown
4	B2	Digital input (+)	white
5	B3	Digital output (+)	yellow
6	B4	Digital output (-)	green

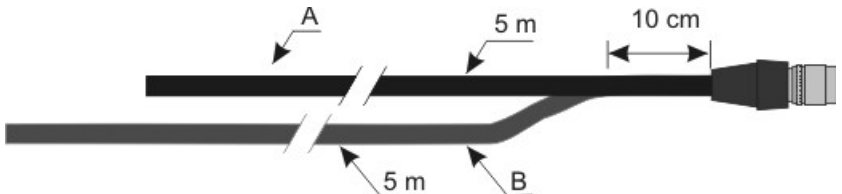


Figure 275: GigE uEye SE 2+4-wire cable without AC adapter
(AD.0040.2.18400.00)

Power supply

Voltage	Tolerance	Residual ripple
12 V	+/- 10%	max. 1%

For information on the camera's power consumption, see [Specifications: Sensor Data](#) chapter.



Please keep in mind that a voltage drop will occur when you use long cables for power supply to the camera. Choose the size of the cable in such a way that the supply voltage available at the input of the *GigE uEye SE* is 12 V.



The inrush current of the *GigE uEye SE* cameras may temporarily increase to up to 2 A.

6.4.7.3 Digital Input Wiring (Trigger)

Digital input specifications

	Min.	Max.	
Level low	0	2	V
Level high	5	24	V
Voltage range	0	30	V
Trigger pulse width (edge)	100	-	µs
Trigger edge steepness	35		V/ms
Breakdown voltage		50	V
Input current	10	-	mA

For interpreting the trigger signal, either the positive or the negative edge can be used. The digital input is galvanically isolated using an opto coupler to protect the camera and the PC against surges. Only DC voltages may be applied to the digital input.

Digital input wiring

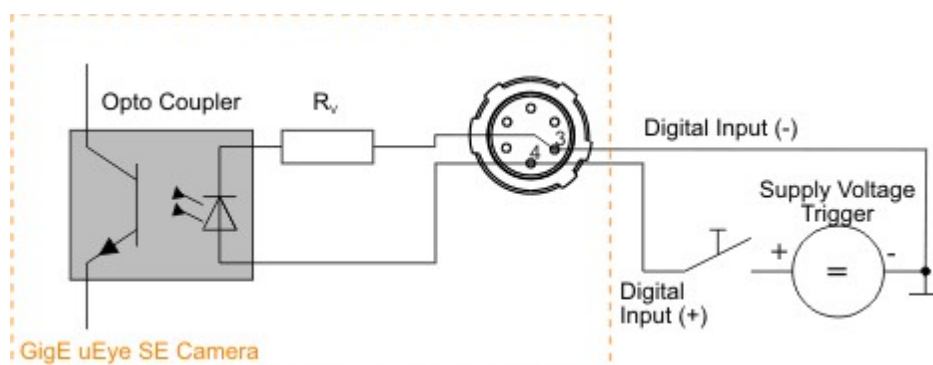


Figure 276: Wiring of the trigger connector

6.4.7.4 Digital Output Wiring (Flash)

Digital output specifications

	Max.	
Output current (short-time)	500	mA
Output current (permanent)	150	mA
Output voltage	30	V
Breakdown voltage	50	V
Collector power dissipation	125	mW

The digital output is galvanically isolated using an opto coupler to protect the camera and the PC against surges. Only DC voltages may be applied to the digital input.

The output of the opto coupler can be used as an open collector or open emitter output. This means that the output signal can be connected to ground or to the supply voltage. The output signal is active if the collector-emitter switch is closed (software setting: *Flash high active*, see also the [Camera Properties: Input/Output](#) section).

Digital output wiring

The following figures show examples of how the digital output is wired.

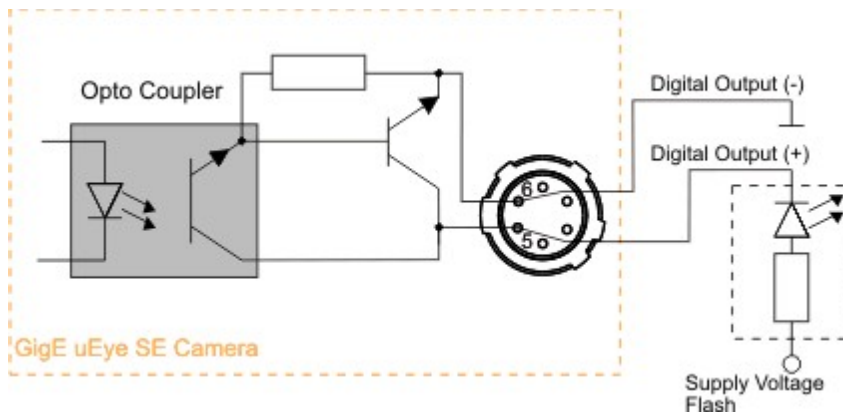


Figure 277: Wiring of the digital output as an open collector output

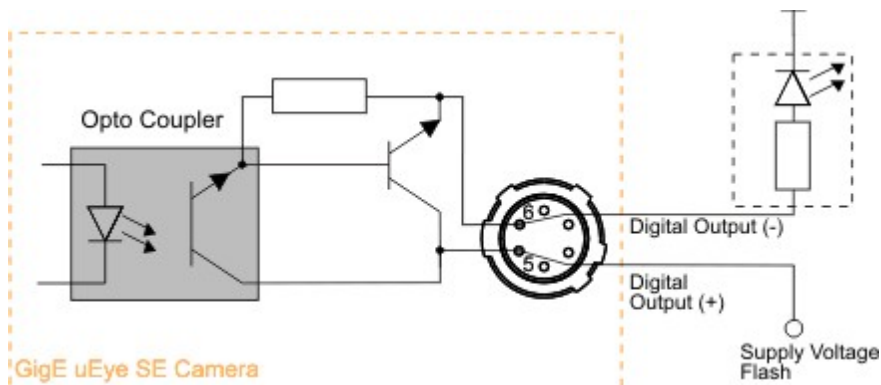


Figure 278: Wiring of the digital output as an open emitter output

6.4.8 GigE uEye RE

6.4.8.1 Pin Assignment of the GigE Connector (RJ45)

8-pin RJ45 socket

Pin	Designation 100BASE-TX	Designation 1000BASE-TX
1	Tx+	BI_DA+
2	Tx-	BI_DA-
3	Rx+	BI_DB+
4		BI_DC+
5		BI_DC-
6	Rx-	BI_DB-
7		BI_DD+
8		BI_DD-

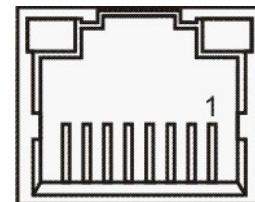


Figure 279: GigE uEye RE - RJ45 socket, camera rear view



The RJ45 socket of the *GigE uEye RE* complies with the IEC 60603-7 standard.

6.4.8.2 Pin Assignment of the I/O Connector

7-pin Binder connector

Pin	Description
1	Digital input (+)
2	Power supply (VCC) 12 V
3	Cable shield
4	Ground (GND)
5	Digital output (+)
6	Digital output (-)
7	Digital input (-)

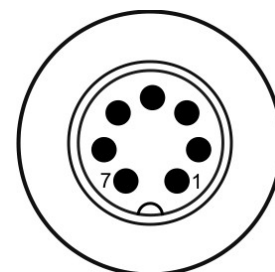


Figure 280: GigE uEye RE, I/O connector male, camera rear view

Pin assignment of the 6-wire connecting cable (7-pin Binder connector)

Pin	Description	Cable color
1	Digital input (+)	white
2	Power supply (VCC) 12 V	pink
3	Cable shield	Shield
4	Ground (GND)	gray
5	Digital output (+)	yellow
6	Digital output (-)	green
7	Digital input (-)	brown

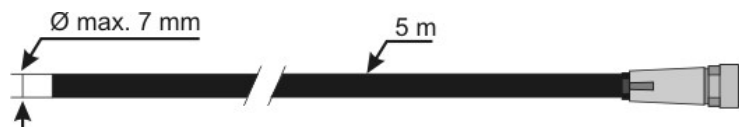


Figure 281: GigE uEye RE 6-wire cable without AC adapter (CK00079)

Pin assignment of the 2+4-wire connecting cable (7-pin Binder connector)

Pin	Ader	Beschreibung	Kabelfarbe
1	B1	Digital input (+)	white
2	A2	Power supply (VCC) 12 V	brown
3	-	Cable shield	Shield
4	A1	Ground (GND)	white
5	B4	Digital output (+)	yellow
6	B3	Digital output (-)	green
7	B2	Digital input (-)	brown

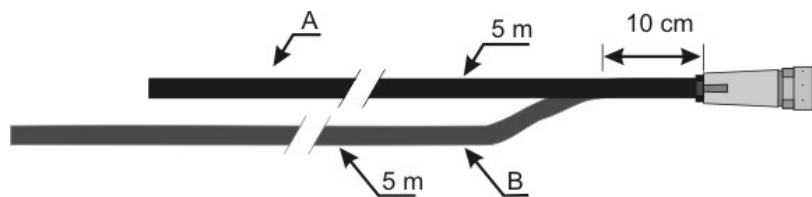


Figure 282: GigE uEye RE 2+4-wire cable without AC adapter (CK00080)

Power supply

Voltage	Tolerance	Residual ripple
12 V	+/- 10%	max. 1%

For information on the camera's power consumption, see [Specifications: Sensor Data](#) chapter.



Please keep in mind that a voltage drop will occur when you use long cables for power supply to the camera. Choose the size of the cable in such a way that the supply voltage available at the input of the *GigE uEye RE* is 12 V.



The inrush current of the *GigE uEye RE* cameras may temporarily increase to up to 2 A.

6.4.8.3 Digital Input Wiring (Trigger)

Digital input specifications

	Min.	Max.	
Level low	0	2	V
Level high	5	24	V
Voltage range	0	30	V
Trigger pulse width (edge)	100	-	µs
Trigger edge steepness	35		V/ms
Breakdown voltage		50	V
Input current	10	-	mA

For interpreting the trigger signal, either the positive or the negative edge can be used. The digital input is galvanically isolated using an opto coupler to protect the camera and the PC against surges. Only DC voltages may be applied to the digital input.

Digital input wiring

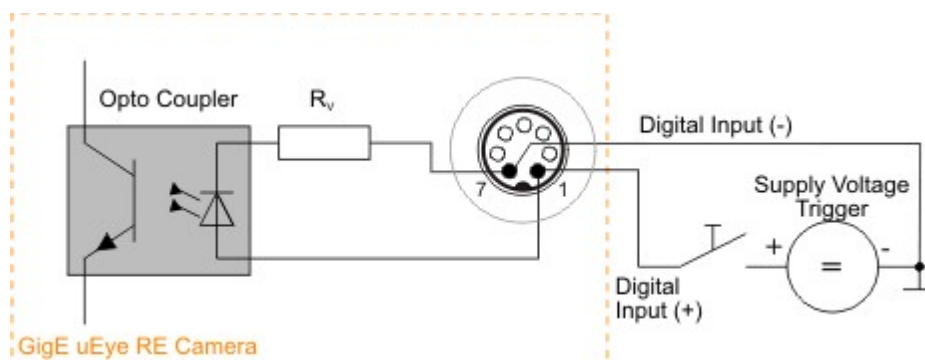


Figure 283: GigE uEye RE - Wiring of the trigger connector

6.4.8.4 Digital Output Wiring (Flash)

Digital output specifications

	Max.	
Output current (short-time)	500	mA
Output current (permanent)	150	mA
Output voltage	30	V
Breakdown voltage	50	V
Collector power dissipation	125	mW

The digital output is galvanically isolated using an opto coupler to protect the camera and the PC against surges. Only DC voltages may be applied to the digital input.

The output of the opto coupler can be used as an open collector or open emitter output. This means that the output signal can be connected to ground or to the supply voltage. The output signal is active if the collector-emitter switch is closed (software setting: *Flash high active*, see also the [Camera Properties: Input/Output](#) section).

Digital output wiring

The following figures show examples of how the digital output is wired.

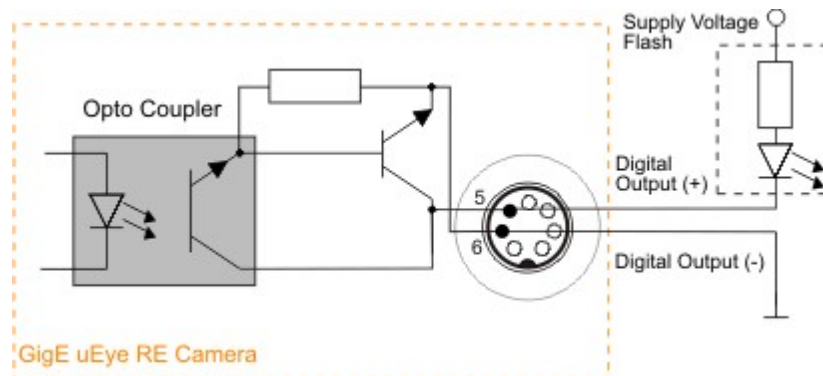


Figure 284: GigE uEye RE - Wiring of the digital output as an open collector output

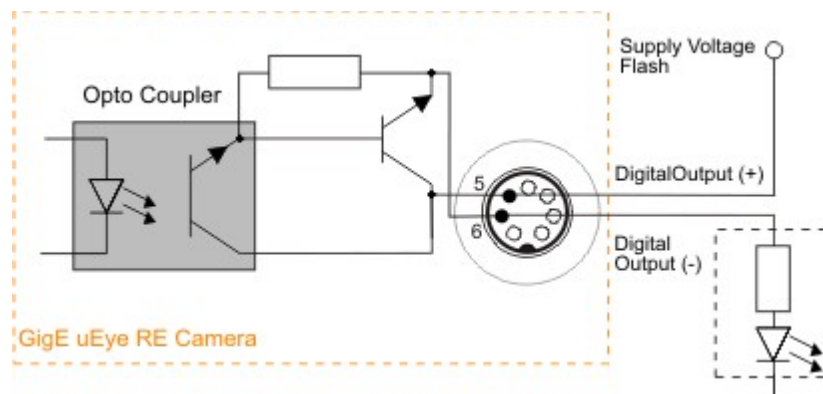


Figure 285: GigE uEye RE - Wiring of the digital output as an open emitter output

6.4.9 GigE uEye HE

6.4.9.1 Pin Assignment of the GigE Connector (RJ45)

8-pin RJ45 socket

Pin	Designation 100BASE-TX	Designation 1000BASE-TX
1	Tx+	BI_DA+
2	Tx-	BI_DA-
3	Rx+	BI_DB+
4		BI_DC+
5		BI_DC-
6	Rx-	BI_DB-
7		BI_DD+
8		BI_DD-

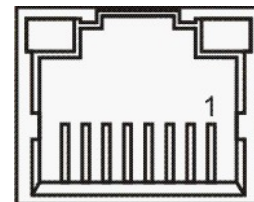


Figure 286: GigE uEye HE - RJ45 socket, camera rear view



The RJ45 socket of the *GigE uEye HE* complies with the IEC 60603-7 standard.

6.4.9.2 Pin Assignment of the I/O Connector

14-pin MDR 14 socket

Pin	Description
1	Ground (GND)
2	Power supply (VCC)
3	Digital input (-)
4	Digital input (+)
5	Digital output (-)
6	Digital output (+)
7	Ground (GND)
8	Ground (GND)
9	Power supply (VCC)
10	General Purpose I/O 1 (not potential-free)
11	General Purpose I/O 2 (not potential-free)
12	RS232 RxD (not potential-free)
13	RS232 TxD (not potential-free)
14	Ground (GND)

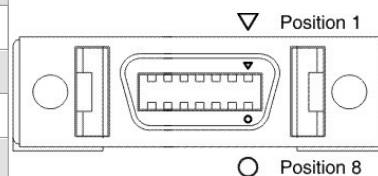


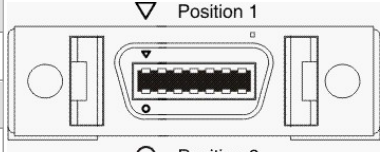
Figure 287: GigE uEye HE - I/O socket, camera rear view



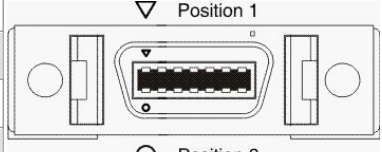
The following pins are internally connected and can be used identically:

- Pins 2 and 9: Power supply (VCC)
- Pins 1, 7, 8, and 14: Ground (GND)

Pin assignment of the 12-wire connecting cable (14-pin MDR 14 plug)

Pin	Designation	Cable color	 Figure 288: GigE uEye HE cable - I/O socket, connecting side view
1, 7, 8, 14	Ground (GND)	black	
2, 9	Power supply (VCC)	red	
3	Digital input (-)	brown	
4	Digital input (+)	white	
5	Digital output (-)	green	
6	Digital output (+)	yellow	
10	General Purpose I/O 1	blue	
11	General Purpose I/O 2	pink	
1, 7, 8, 14	Ground (GND)	red/blue	
12	RS232 RxD	gray	
13	RS232 TxD	purple	
1, 7, 8, 14	Ground (GND)	gray/pink	

Pin assignment of the I/O and power cable without AC adapter

Pin	Cable	Designation	Cable color	 Figure 289: GigE uEye HE cable - I/O socket, connecting side view
1, 7, 8, 14	A	Ground (GND)	white	
2, 9	A	Power supply (VCC)	brown	
3	B	Digital input (-)	brown	
4	B	Digital input (+)	white	
5	B	Digital output (-)	green	
6	B	Digital output (+)	yellow	



The power supply (VCC) must be connected to pins 2 and 9. In addition, the power supply ground wire must be connected to all 4 GND pins (pins 1, 7, 8, and 14). If this is not possible due to insufficient space, connect at least pins 1 and 8 to the power supply ground wire. For EMC reasons, the cable shield must not be connected to the GND wire.

Power supply

Voltage		Tolerance	Residual ripple
Minimum (at camera)	6 V	-10%	max. 1%
Maximum	24 V	+10%	max. 1%
Recommended	12 V	-	-

For information on the camera's power consumption, see [Specifications: Sensor Data](#) chapter.



Please keep in mind that a voltage drop will occur when you use long cables for power supply to the camera. Choose the size of the cable in such a way that the supply voltage available at the input of the *GigE uEye HE* is 12 V.

To ensure a sufficient voltage (6...24 V) at the camera input, we recommend the following AC adapter voltages:

Power cable length	AC adapter voltage
up to 5 m	9-24 V
5-10 m	12-24 V
10-20 m	15-24 V
20-30 m	18-24 V
30-50 m	24 V



The inrush current of the *GigE uEye HE* may temporarily increase to up to 2 A.

6.4.9.3 Digital Input Wiring (Trigger)

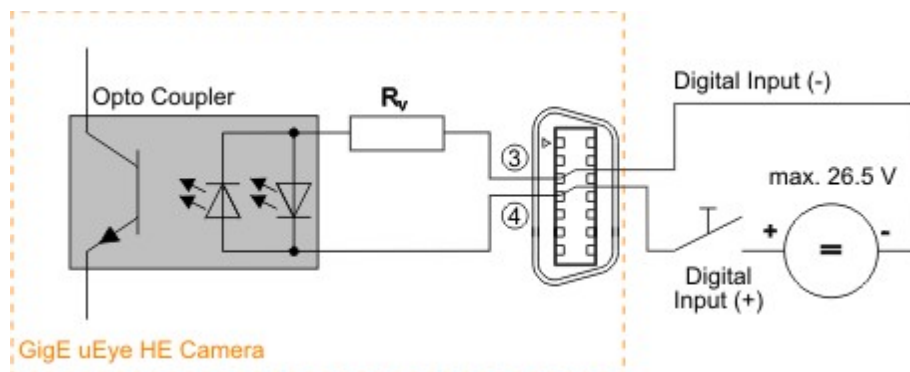
Digital input specifications

	Min.	Max.	
Level low	0	2	V
Level high	5,0	26,5	V
Trigger pulse width	1	-	µs
Trigger edge steepness	35		V/ms
Breakdown voltage		50	V
Input current	10	-	mA

For interpreting the trigger signal, either the positive or the negative edge can be used. The digital input is galvanically isolated using an opto coupler to protect the camera and the PC against surges. Only DC voltages may be applied to the digital input.

Digital input wiring

The following figures show examples of how the digital input is wired.



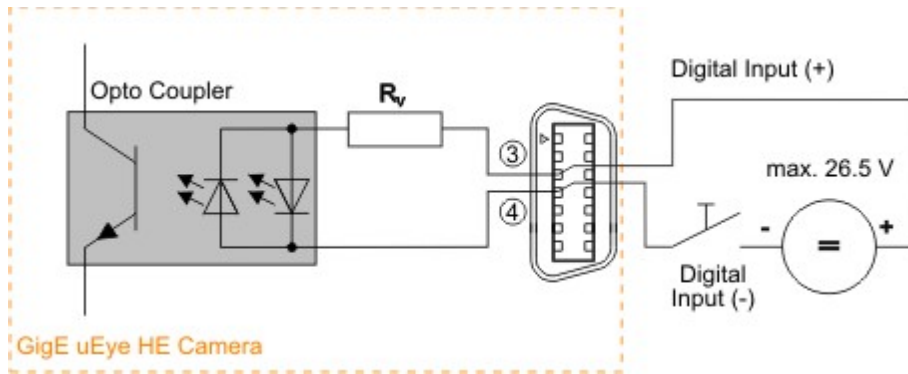


Figure 290: Trigger connector (schematic)



The opto isolated digital input has two LEDs which are not activated in parallel. This way, you can use either positive or negative signals for triggering. The input polarity can be selected as desired. The Trigger+ and Trigger- labeling in the figures above is only used for schematic illustration.

6.4.9.4 Digital Output Wiring (Flash)

Digital output specifications

	Max.	
Output current (short-time)	500	mA
Output current (permanent)	150	mA
Output voltage	30	V
Breakdown voltage	50	V
Collector power dissipation	125	mW

The digital output is galvanically isolated using an opto coupler to protect the camera and the PC against surges. Only DC voltages may be applied to the digital input.

The output of the opto coupler can be used as an open collector or open emitter output. This means that the output signal can be connected to ground or to the supply voltage. The output signal is active if the collector-emitter switch is closed (software setting: *Flash high active*, see also the [Camera Properties: Input/Output](#) section).

Digital output wiring

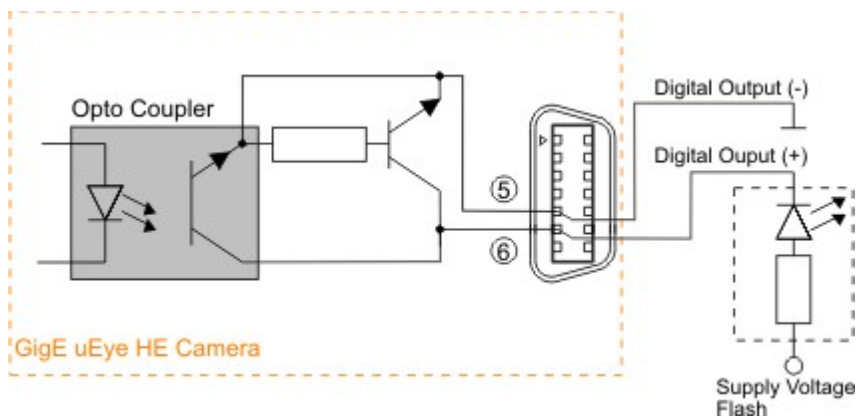


Figure 291: Flash connector (schematic)

6.4.9.5 General Purpose I/O Wiring

GPIO specifications

The two GPIOs (General Purpose I/O) can be used as inputs or outputs. This selection is made by software using the corresponding SDK API functions. Please observe the following criteria:

- Input: 3.3 V LVTTTL, max. input voltage 4.0 V
- Output: 3.3 V LVCMOS, max. 10 mA



The General Purpose I/Os are not potential-free and have no protective circuits.

GPIO wiring

The following figures illustrate GPIO wiring examples.

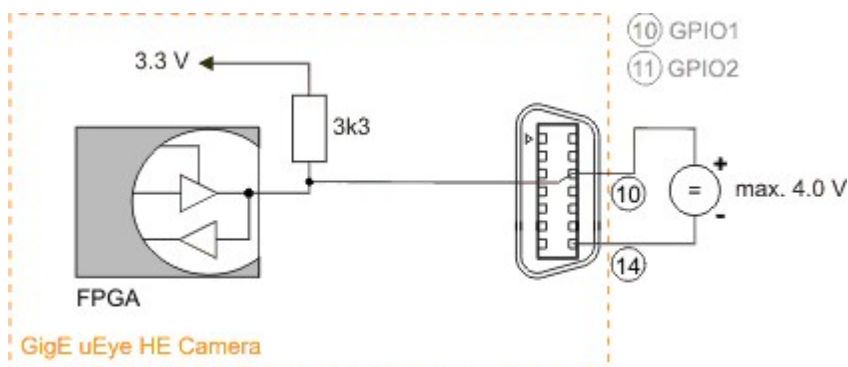


Figure 292: GPIO input

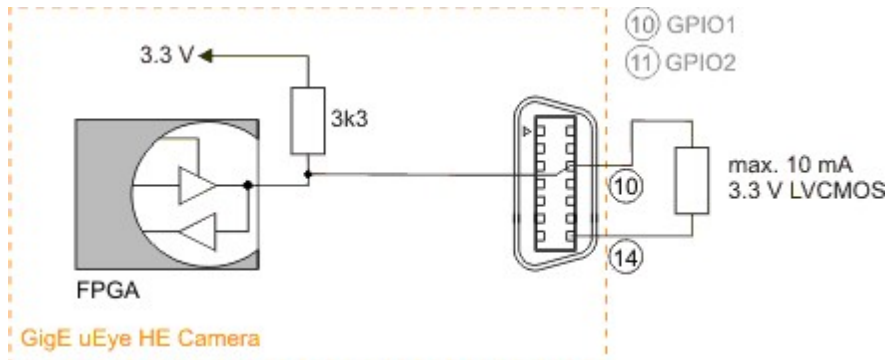


Figure 293: GPIO output

6.4.9.6 Serial Interface Wiring (RS232)

Serial interface specifications

Minimum output voltage	±33.5	V
Maximum input voltage	±315	V
Supported baud rates	1,200 2,400 4,800 9,600 19,200 38,400 57,600 115,200	baud
Transmission mode	Full duplex, 8N1	
Data bits	8	
Stop bits	1	
Parity	None	



With the 8N1 mode, the maximum payload data rate achievable is 80% of the selected baud rate.

Serial interface wiring

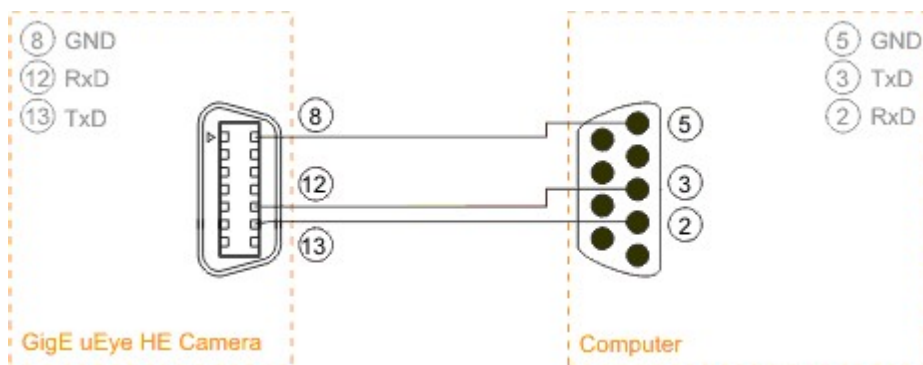


Figure 294: Serial interface connector (schematic)

6.5 Accessories

Lenses

IDS also supplies a wide variety of lenses from leading manufacturers. Contact our sales department for a detailed quote tailored to your needs.

USB cables

Purchase Order No.	Length	Cable Type	Connector Camera Side	Connector PC Side
AD.0040.2.08400.00	3 m	AWG 28, single shielded	USB 2.0 Mini B type	USB 2.0 Type A
AD.0040.2.08500.00	5 m			
AD.0040.2.11400.00	25 cm			
CK00092	5 m		USB 2.0 Mini B type, angled to the right	
CK00093			USB 2.0 Mini B type, angled to the left	
CK00096			USB 2.0 Mini B type, angled forwards with integrated adapter	
CK00097			USB 2.0 Mini B type, angled backwards with integrated adapter	

USB Hubs and Extensions

Purchase Order No.	Description
AL.0094.2.01900.00	EX-1200 USB 2.0 high speed PCMCIA card, 2 ports, NEC chipset
AL.0094.2.02500.00	EX-1074 USB 2.0 high speed PCI card, 4 ports, NEC chipset
AL.0094.2.02400.00	EX-1171 USB 2.0 hub, metal housing, 7 ports
AL.0094.2.02100.00	EX-1171 USB 2.0 hub, 7 ports
AL.0094.2.02200.00	EX-1163 USB 2.0 hub, 4 ports
AL.0094.2.02300.00	USB 2.0 active expansion cable, 5 m (single port), USB 2.0 connector A type to A type

GigE Cables

Purchase Order No.	Length	Cable Type	Connector Camera Side	Connector PC Side
AD.0122.2.17400.00 *)	5 m	Cat5e patch cable, UTP	RJ 45, straight, lockable	RJ 45, straight, lockable
AD.0122.2.17500.00 *)	10 m			
CK00091	5m	Cat5e patch cable, double shielded	RJ 45, straight, lockable with screws	RJ 45, straight, lockable

*) Item not available from stock

GigE Network Interface Cards (NIC)

Purchase Order No.	Description
AL.0122.2.06700.00	Intel PRO/1000 GT, Gigabit Ethernet NIC, PCI interface
AL.0122.2.06800.00	Intel PRO/1000 PT, Gigabit Ethernet NIC, PCIexpress interface

6.5.1 USB uEye SE

USB cables

Purchase Order No.	Length	Cable Type	Connector Camera Side	Connector PC Side
AD.0040.2.08600.00	3 m	AWG 28, single shielded	Mikro D-Sub for screw-mounting, <u>straight</u>	USB 2.0 Type A
AD.0040.2.08700.00	5 m			
AD.0040.2.11200.00	3 m		Mikro D-Sub for screw-mounting, <u>angled</u>	
AD.0040.2.11300.00	5 m			
AD.0040.2.14800.00	1.7 m			
AD.0040.2.16200.00	5 m	IGUS Chainflex CFBUS.065, ø 7.5 mm		

For more USB cables and accessories see also [Accessories for all uEye cameras](#).

For information on the pin assignment of the cables and connectors see chapter [USB uEye SE: Pin Assignment](#).

USB Cables with Cables for digital I/Os

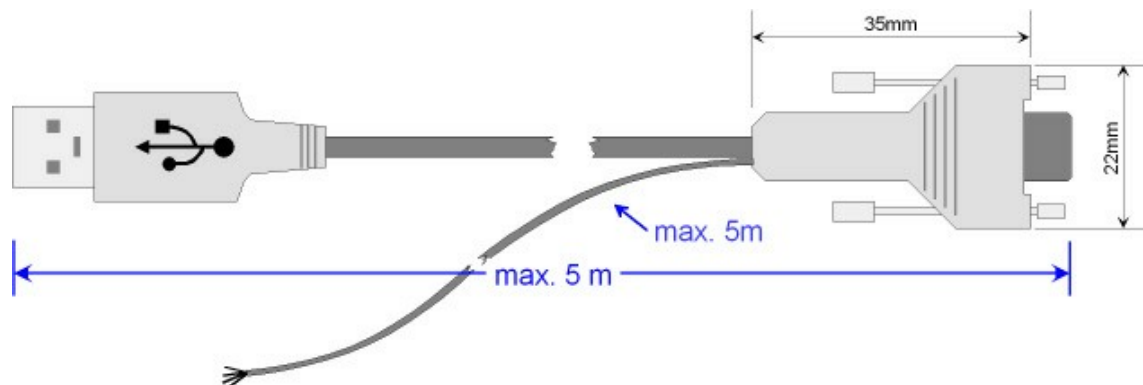


Figure 295: USB uEye SE special cable with trigger input (AD.0040.2.08700.00)

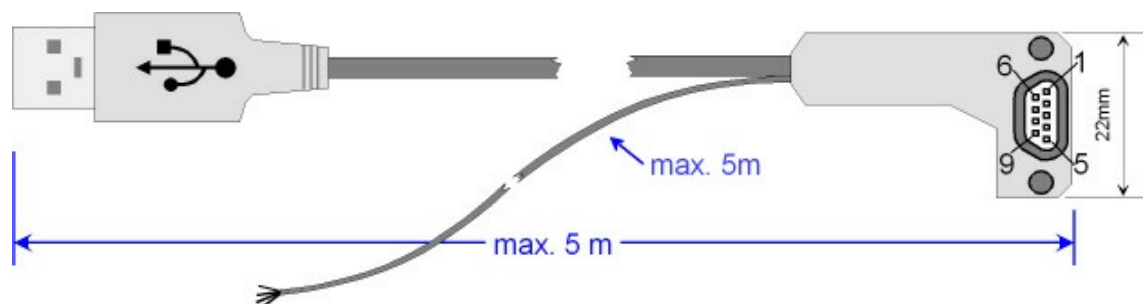


Figure 296: USB uEye SE special cable, angled, with trigger input and digital output (AD.0040.2.10100.00)

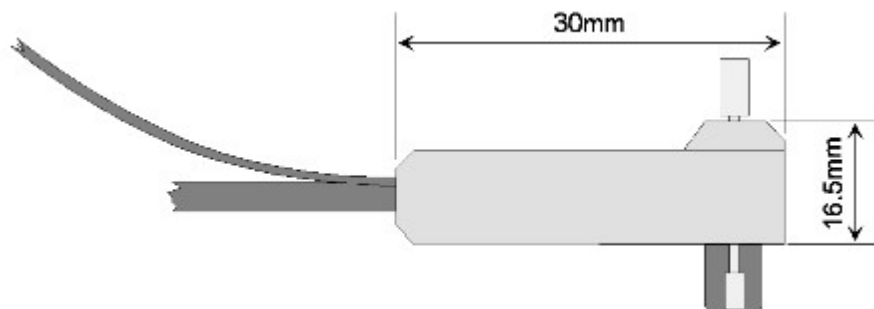


Figure 297: USB uEye SE angled D-Sub connector

Purchase Order No.	Length	Cable Type	Connector Camera Side	Connector PC Side
AD.0040.2.10300.00	3 m	USB cable, AWG 28, single shielded, additional cable for <u>digital I/Os</u> , 4-wire, open wires	Mikro D-Sub for screw-mounting, <u>straight</u>	USB 2.0 Type A
AD.0040.2.10400.00	5 m			
AD.0040.2.10000.00	3 m			
AD.0040.2.10100.00	5 m			
AD.0040.2.16600.00	5 m	Drag-chain compatible, IGUS Chainflex CFBUS.065, ø 7.5 mm, additional cable for <u>digital I/Os</u> , 4-wire, open wires	Mikro D-Sub for screw-mounting, <u>angled</u>	

Tripod adapter for USB uEye SE

Purchase Order No.	Description
AL.0012.2.01300.00	Tripod adapter for <i>USB uEye SE</i> series (4 screws included)

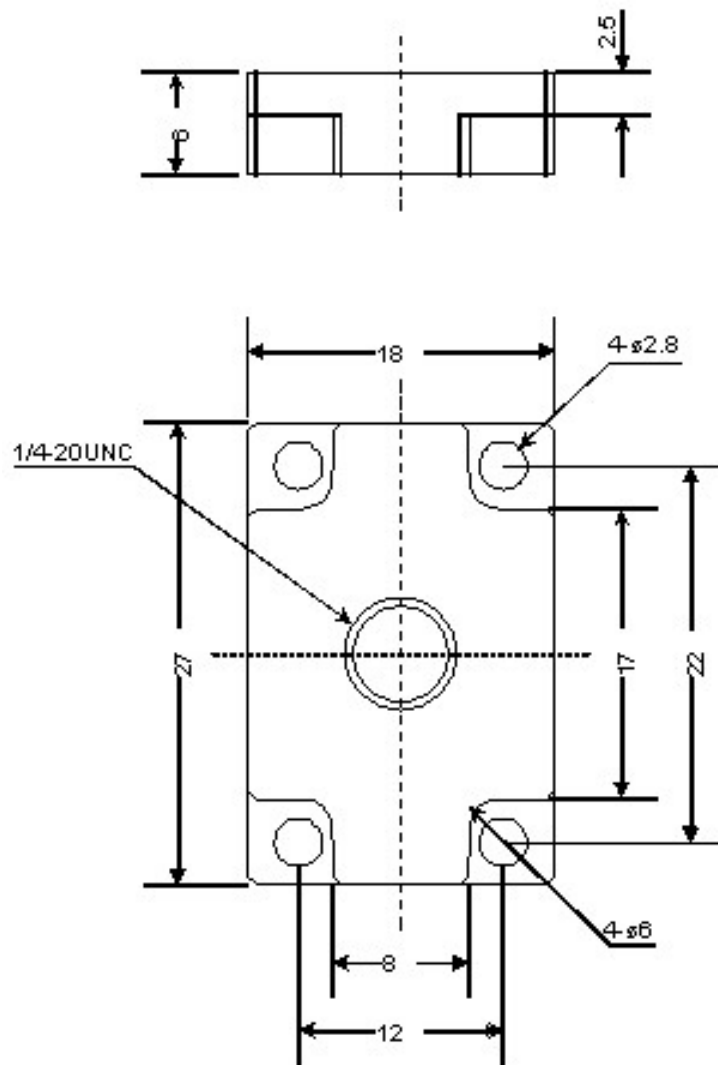


Figure 298: USB uEye SE tripod adapter

6.5.2 USB uEye ME

USB uEye ME USB cables

Purchase Order No.	Length	Cable Type	Connector Camera Side	Connector PC Side
CK00036	3 m	AWG 24, double shielded	USB 2.0 Mini B type, straight, for screw-mounting	USB 2.0 Type A
CK00036	5 m			

For more USB cables and accessories see also [Accessories for all uEye cameras](#).

For information on the pin assignment of the cables and connectors see chapter [USB uEye ME: I/O Connector](#).

Cables for digital I/Os

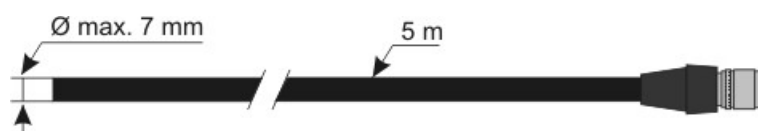


Figure 299: USB uEye ME 6-wire cable

Purchase Order No.	Length	Cable Type	Connector Camera Side	Connector PC Side
CK00040	5 m	4-wire cable, open wires	Hirose HR10A-7P-6S, 6-pin, lockable	-

Tripod adapter for *USB uEye ME*

Purchase Order No.	Description
CK00033	Tripod adapter for <i>USB uEye ME</i> (4 screws included)

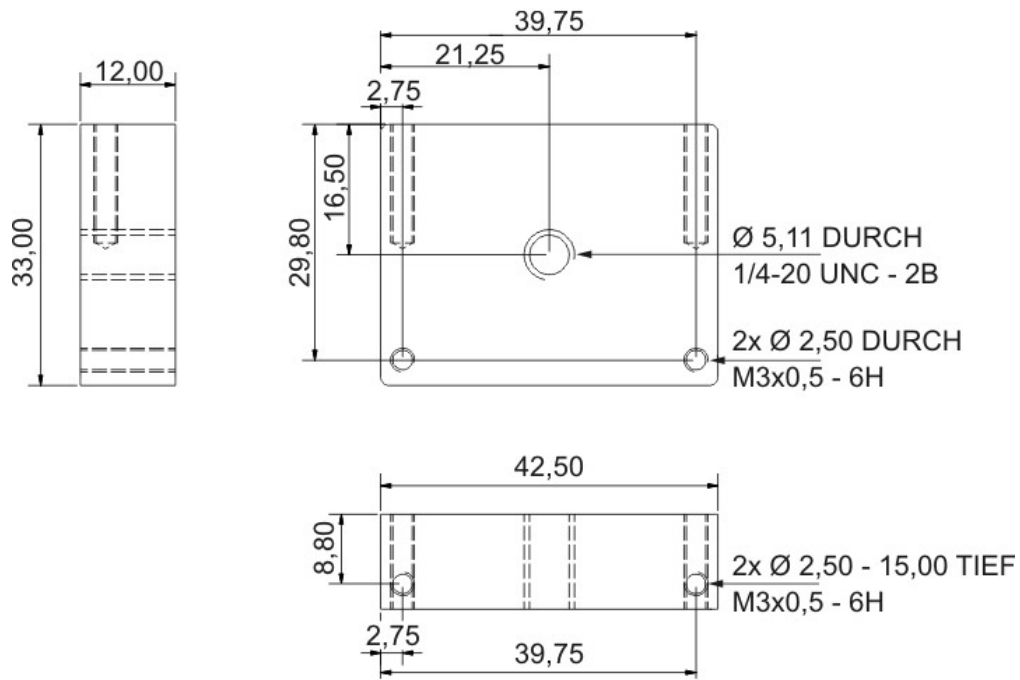


Figure 300: USB uEye ME - Tripod adapter dimensions

Special tool for filter glass replacement

Purchase Order No.	Description
CK.0121.2.26900.00	Octagonal Allen key-type tool for filter glasses

6.5.3 USB uEye RE

USB cables

Purchase Order No.	Length	Cable Type	Connector Camera Side	Connector PC Side
AD.0040.2.12100.00	3 m	AWG 28, single shielded	Binder type 712, 5-pin, <u>straight</u> , IP 65/67, lockable	USB 2.0 type A
AD.0040.2.12200.00	5 m			
AD.0040.2.12300.00 *)	3 m		Binder type 712, 5-pin, <u>angled</u> , IP 65/67, lockable	
AD.0040.2.12400.00 *)	5 m			

*) Item not available from stock

For more USB cables and accessories see also [Accessories for all uEye cameras](#).

For information on the pin assignment of the cables and connectors see chapter [USB uEye RE: I/O Connector](#).

USB cables, drag-chain compatible

Purchase Order No.	Length	Cable Type	Connector Camera Side	Connector PC Side
AD.0040.2.16300.00	5 m	IGUS Chainflex CFBUS.065 ø 7.5 mm	Binder type 712, 5-pin, <u>straight</u> , IP 65/67, lockable	USB 2.0 type A
AD.0040.2.13500.00	6 m			
AD.0040.2.13600.00	8 m			
AD.0040.2.13700.00	10 m			
AD.0040.2.16500.00 *)	5 m		Binder type 712, 5-pin, <u>angled</u> , IP 65/67, lockable	

*) Item not available from stock

Cables for digital I/Os

Purchase Order No.	Length	Cable Type	Connector Camera Side	Connector PC Side
AD.0040.2.12500.00	5 m	4-wire cable, open wires	Binder type 712, 4-pin, <u>straight</u> , IP 65/67, lockable	-
AD.0040.2.12600.00	5 m		Binder type 712, 4-pin, <u>angled</u> , IP 65/67, lockable	

USB RE Hub and Connecting Cables

Purchase Order No.	Length	Cable Type	Connector Camera Side	Connector Hub Side
SL.0028.2.06200.00		USB 2.0 hub, 4 ports for lockable <i>USB uEye RE</i> cables, metal housing, power supply included		
AD.0040.2.14900.00 *)	3 m	Connecting cable to RE hub	Binder type 712, 5-pin, <u>straight</u> , IP 65/67, lockable	Binder type 712, 5-pin, <u>straight</u> , IP 65/67, lockable
AD.0040.2.15000.00 *)	5 m			

*) Item not available from stock

Socket, not fitted

BK.0068.2.01500.00	Binder type 712, 5-pin, straight, IP 65/67, lockable, not fitted
BK.0068.2.01600.00	Binder type 712, 5-pin, angled, IP 65/67, lockable, not fitted
BK.0068.2.01700.00	Binder type 712, 4-pin, straight, IP 65/67, lockable, not fitted
BK.0068.2.01800.00	Binder type 712, 4-pin, angled, IP 65/67, lockable, not fitted

Lens tubes

Purchase Order No	Length	Max. usable diameter	Max. usable lens length	Protective glass type
CK.0010.1.12100.00	51 mm	35 mm	38 mm	IMPAdur clear glass, heat-strengthened, AR-coating on the inside, thickness 3±0.3 mm
CK.0010.1.12000.00	64 mm	35 mm	51 mm	
CK.0010.1.12200.00	77 mm	35 mm	64 mm	

6.5.4 USB uEye LE

For USB cables and accessories see also [Accessories for all uEye cameras](#).

CS-/C-mount adapter

Purchase Order No.	Description
BE.0050.2.01300.00	5 mm CS-/C-mount adapter for <i>USB uEye LE</i> housing version with CS-mount

Dimensions of the extension ring (only USB uEye LE housing version with CS-mount)

The extension ring increases the flange back distance of the *USB uEye LE* by 5 mm. This way, you can also use lenses with a C-mount thread for the *USB uEye LE*.



Figure 301: USB uEye LE extension ring (top view)

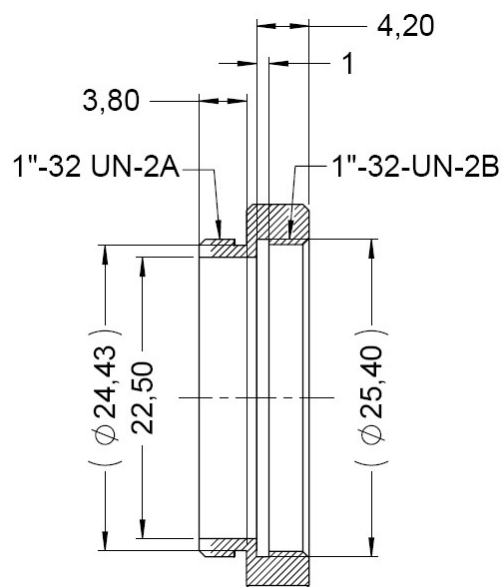


Figure 302: USB uEye LE extension ring dimensions

Special tool for adjusting the flange back distance

Purchase Order No.	Description
CK.0121.2.26900.00	Octagonal Allen key-type tool for filter glasses
CK.0124.1.28700.00	Shim for the flange back distance

6.5.5 GigE uEye SE

For GigE cables and accessories see also [Accessories for all uEye cameras](#).

For information on the pin assignment of the cables and connectors see chapter [GigE uEye SE: Pin Assignment of the I/O Connector](#).

Cables for Power and Digital I/Os, with Power Supply

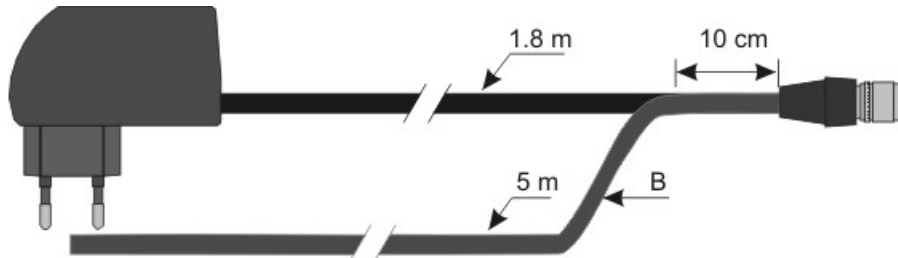


Figure 303: GigE uEye SE 2+4-wire Y-cable with AC adapter (AD.0040.2.18600.00)

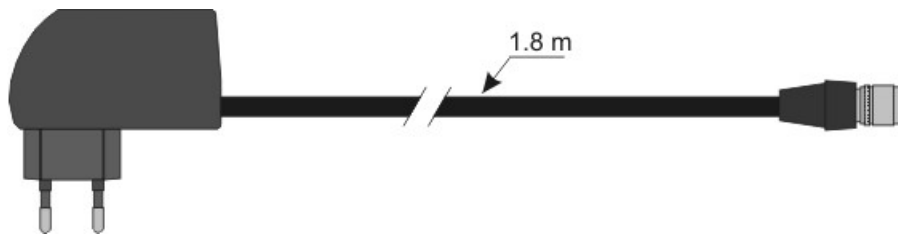


Figure 304: GigE uEye SE power cable with AC adapter (CK.0040.2.18500.00)

Purchase Order No.	Length	Cable Type	Connector Camera Side	Power Supply
CK.0040.2.18800.00	1.8 m	Cable for <u>power supply</u> , 2-wire	Hirose HR10A-7P-6S, 6-pin, straight, lockable	Worldwide power supply, incl. 5 adapters
AD.0040.2.18600.00	1.8m/ 5 m	Y-cable for <u>power supply</u> and <u>digital I/Os</u> , 2+4-wire, open wires		EU power supply
AD.0040.2.18600.10 *)	open	Cable for AD.0040.2.18600.10, available by the meter		
AD.0040.2.18600.11 *)				

*) Item not available from stock

Cables for Power and Digital I/Os, without Power Supply

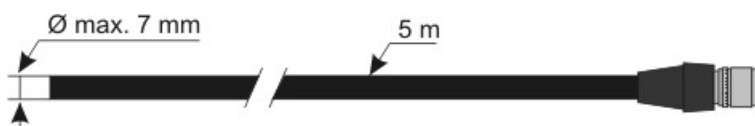


Figure 305: GigE uEye SE 6-wire cable without AC adapter (AD.0040.2.18300.00)

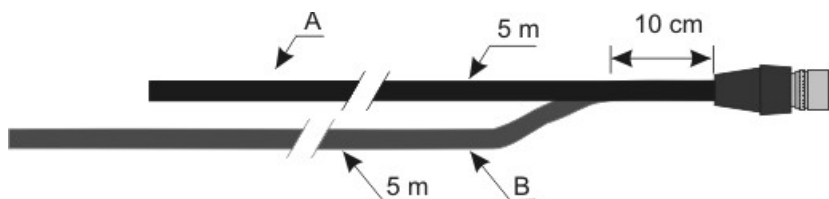


Figure 306: GigE uEye SE 2+4-wire Y-cable without AC adapter
(AD.0040.2.18400.00)

Purchase Order No.	Length	Cable Type	Connector Camera Side	Power Supply
AD.0040.2.18300.00	5 m	Cable for <u>power supply</u> and <u>digital I/Os</u> , 6-wire, open wires	Hirose HR10A-7P-6S, 6-pin, straight, lockable	
AD.0040.2.18700.00	10 m			
AD.0040.2.18400.00	5 m	Y-cable for <u>power supply</u> and <u>digital I/Os</u> , 2+4-wire, open wires		
AD.0040.2.18400.10 *)	open			
AD.0040.2.18400.11 *)		Cable for AD.0040.2.18400.10, available by the meter		

*) Item not available from stock

Tripod adapter for GigE uEye SE

Purchase Order No.	Description
AL.0113.2.07400.00	Tripod adapter for GigE uEye SE (4 screws included)

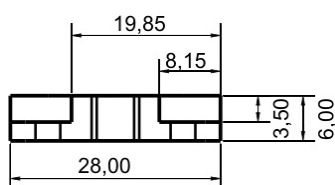


Figure 307: GigE uEye SE stand plate - front view

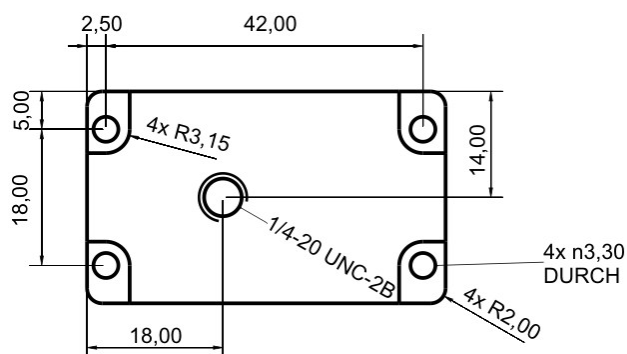


Figure 308: GigE uEye SE stand plate - top view

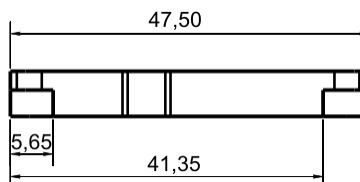


Figure 309: GigE uEye SE stand plate - side view

Special tool for filter glass replacement

Purchase Order No.	Description
CK.0121.2.26900.00	Octagonal Allen key-type tool for filter glasses

6.5.6 GigE uEye RE

GigE Cables

Purchase Order No.	Length	Cable Type	Connector Camera Side	Connector PC Side
CK00086	5 m	Cat5e patch cable, UTP	RJ 45 Harting push/pull housing, straight, lockable, including IP 65/67 hood for screw-mounting	RJ 45, straight, lockable
CK00087	5 m	Cat5e patch cable, drag-chain compatible, AWG 26, STP		
CK00157 *)	open	Cat5e patch cable, UTP		
CK00158 *)		Cable for CK00157, available by the meter		

*) Item not available from stock

For GigE cables and accessories see also [Accessories for all uEye cameras](#).

For information on the pin assignment of the cables and connectors see chapter [GigE uEye RE: Pin Assignment of the I/O Connector](#).

Cables for Power and Digital I/Os, with Power Supply

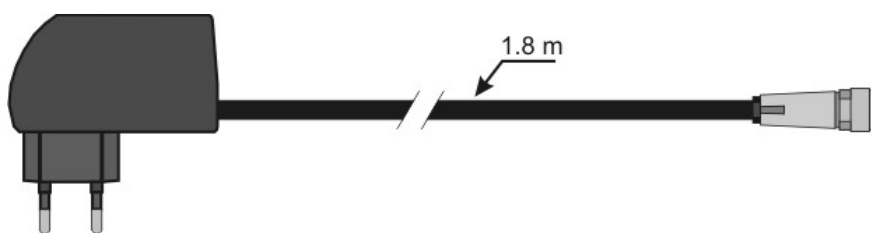


Figure 310: GigE uEye RE power cable with power supply (CK00090)

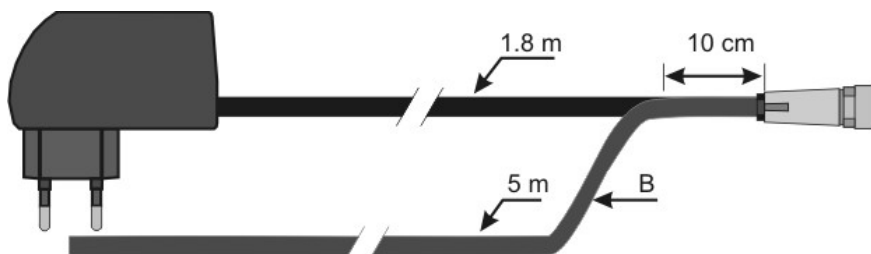


Figure 311: GigE uEye RE Y-cable with power supply (CK00081)

Purchase Order No.	Length	Cable Type	Connector Camera Side	Power Supply
CK00090	1.8 m	Cable for <u>power supply</u> , 2-wire	Binder SUBCIR712 HDRx7, 7-pin, straight, lockable	Worldwide power supply, incl. 5 adapters
CK00078				
CK00081	1.8 m/ 5 m	Y-cable for <u>power supply</u> and <u>digital I/Os</u> , 2+4-wire, open wires		EU power supply
CK00085 *)				
CK00112 *)	1.8 m	Cable for <u>power supply</u> , 2-wire	Binder SUBCIR712 HDRx7, 7-pin, angled, lockable	Worldwide power supply, incl. 5 adapters
CK00082 *)				EU power supply

*) Item not available from stock

Cables for Power and Digital I/Os, without Power Supply

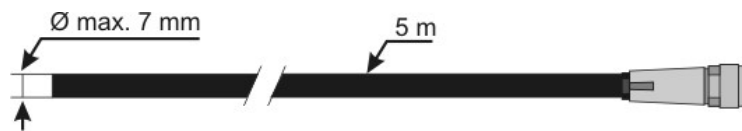


Figure 312: GigE uEye RE cable 6-wire with power supply (CK00079)

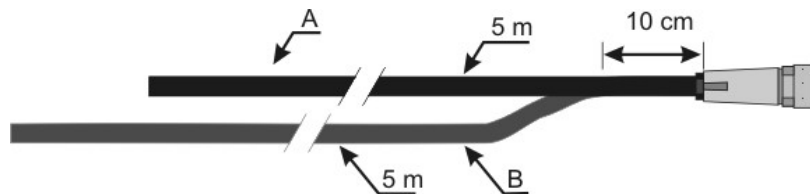


Figure 313: GigE uEye RE Y-cable without power supply (CK00080)

Purchase Order No.	Length	Cable Type	Connector Camera Side	Power Supply
CK00079	5 m	Cable for <u>power supply</u> and <u>digital I/Os</u> , 6-wire, open wires	Binder SUBCIR712 HDRx7, 7-pin, straight, lockable	-
CK00080		Y-cable for <u>power supply</u> and <u>digital I/Os</u> , 2+4-wire, open wires		
CK00083 *)		Cable for <u>power supply</u> and <u>digital I/Os</u> , 6-wire, open wires	Binder SUBCIR712 HDRx7, 7-pin, angled, lockable	
CK00084 *)		Y-cable for <u>power supply</u> and <u>digital I/Os</u> , 2+4-wire, open wires		

*) Item not available from stock

Stativplatte

Purchase Order No.	Description
AL.0113.2.07400.00	<u>Tripod adapter for GigE uEye SE</u> (4 screws included, <i>GigE uEye RE</i> uses only 2 mounting screws)

Special tool for filter glass replacement

Purchase Order No.	Description
CK.0121.2.26900.00	Octagonal Allen key-type tool for filter glasses

6.5.7 GigE uEye HE

For GigE cables and accessories see also [Accessories for all uEye cameras](#).

For information on the pin assignment of the cables and connectors see chapter [GigE uEye HE: Pin Assignment of the I/O Connector](#).

Cables for Power and Digital I/Os, with Power Supply

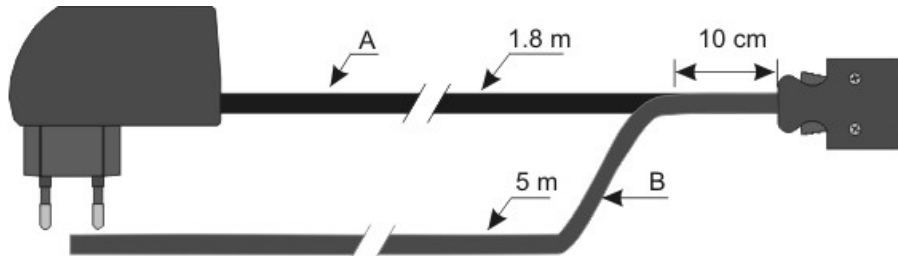


Figure 314: GigE uEye HE trigger and flash cable with AC adapter
(AD.0040.2.17000.00)

Purchase Order No.	Length	Cable Type	Connector Camera Side	Power Supply
CK.0040.2.16900.00	1.8 m	Cable for <u>power supply</u> , 2-wire	3M MDR, 14-pin, straight, lockable	EU power supply
CK00071				Worldwide power supply, incl. 5 adapters
AD.0040.2.17000.00	1.8 m/ 5 m	Y-cable for <u>power supply</u> and <u>digital I/Os</u> , 2+4-wire, open wires		EU power supply
CK00024	1.8 m/ 10 m			

Cables for Power and Digital I/Os, without Power Supply

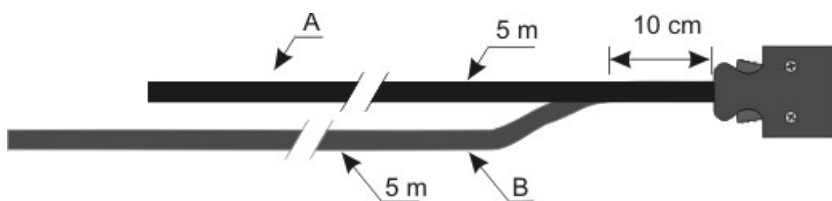


Figure 315: GigE uEye HE trigger and flash cable without AC adapter
(AD.0040.2.17100.00)

Purchase Order No.	Length	Cable Type	Connector Camera Side	Power Supply
AD.0040.2.17100.00 *)	5 m	Y-cable for <u>power supply</u> and <u>digital I/Os</u> , 2+4-wire, open wires	3M MDR, 14-pin, straight, lockable	-
AD.0040.2.17100.10 *)	open	Y-cable for <u>power supply</u> and <u>digital I/Os</u> , 2+4-wire, open wires		
AD.0040.2.17100.11 *)		Cable for AD.0040.2.17100.10, available by the meter		

*) Item not available from stock

Multi-I/O-Cables for Power and Digital I/Os, without Power Supply

Purchase Order No.	Length	Cable Type	Connector Camera Side	Power Supply
AD.0040.2.17700.00 *)	5 m	Y-cable for <u>power supply</u> and <u>digital I/Os</u> , and <u>RS-232</u> , 12-wire, open wires	3M MDR, 14-pin, straight, lockable	-
AD.0040.2.17700.10 *)	open	Y-cable for <u>power supply</u> and <u>digital I/Os</u> , and <u>RS-232</u> , 12-wire, open wires		
AD.0040.2.17700.11 *)		Cable for AD.0040.2.17700.10, available by the meter		

*) Item not available from stock

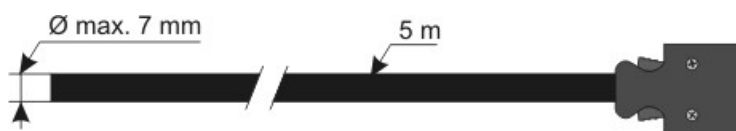


Figure 316: GigE uEye HE connecting cable, 12 wires (AD.0040.2.17700.00)

MDR-14 connector, not fitted

Purchase Order No.	Description
BK.0040.2.01900.00	3M MDR, 14-pin, straight, lockable, not fitted

Tripod adapter for GigE uEye HE cameras

Purchase Order No.	Description
AL.0125.2.07100.00	Tripod adapter for GigE uEye HE cameras (2 screws included)

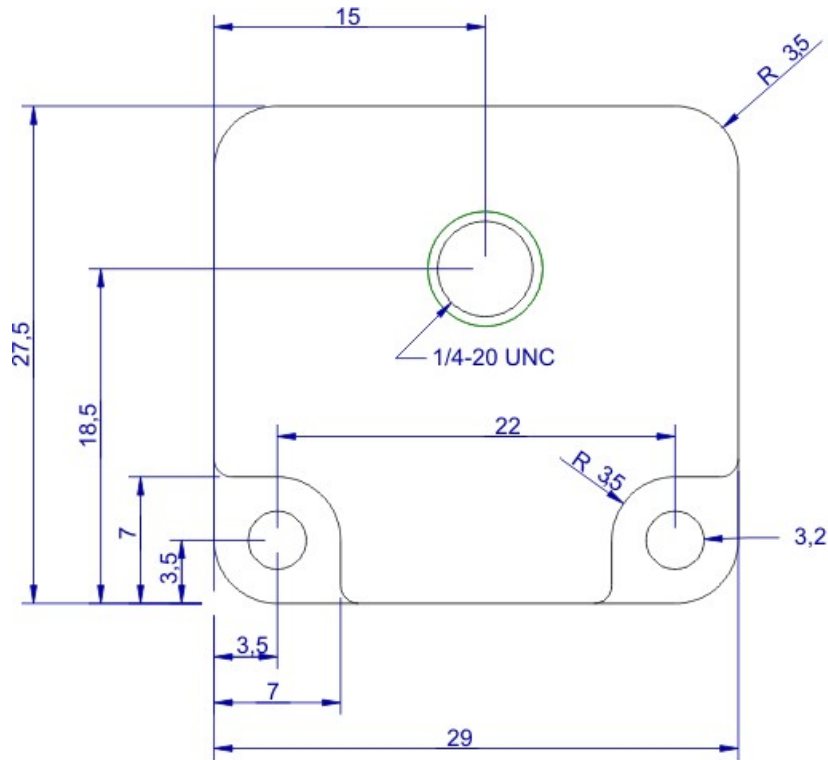


Figure 317: Tripod adapter top view

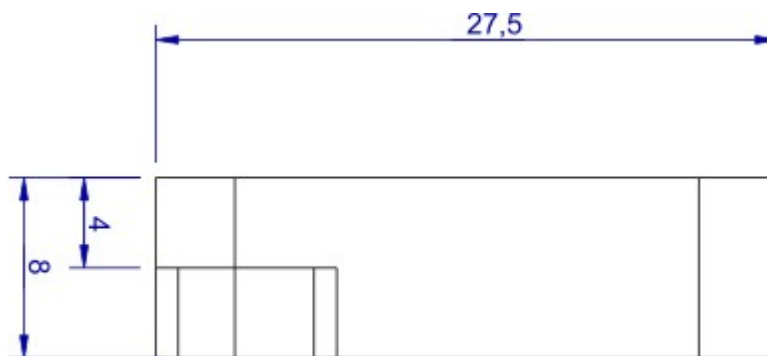


Figure 318: Tripod adapter side view

Special tools for filter change and for adjusting the flange back distance

Purchase Order No.	Description
CK.0121.2.26900.00	Octagonal Allen key-type tool for filter glasses
CK.0124.1.28700.00	Shim for the flange back distance

6.6 Components of uEye Cameras

Components of the USB uEye SE/RE cameras

The *USB uEye SE* and *RE* cameras have a modular structure consisting of the following components:

- USB board, including:
 - a USB 2.0 interface which controls data traffic between the camera and the host PC
 - a micro-controller which controls the digital inputs and outputs, the pixel clock and the image size
 - an EEPROM where the camera manufacturer, type, and serial number are stored
a 64-byte memory area can be used freely by the user
- Sensor board. This board includes:
 - the sensor
 - an EEPROM where the camera type is stored
- Timing board (CCD cameras only)
 - The timing board digitizes the analog output signals of the CCD sensor.

Components of the USB uEye LE camera

USB uEye LE cameras are equipped with a PCB containing the following components:

- CMOS sensor
- Sensor EEPROM where the camera type is stored.
- USB 2.0 interface which controls data traffic between the camera and the host PC.
- Micro-controller which controls the digital inputs and outputs, the pixel clock and the image size.
- EEPROM where the camera manufacturer, type, and serial number are stored.
A 64-byte memory area can be used freely by the user.

Components of the GigE uEye HE camera

The *GigE uEye HE* cameras have a modular structure consisting of the following components:

- Gigabit Ethernet board, including:
 - a Gigabit Ethernet interface which controls data traffic between the camera and the host PC
 - an FPGA which controls the camera functions and performs basic image preprocessing
 - a 64-Mbyte memory used for processing image data
- Sensor board. This board includes:
 - the sensor
 - an EEPROM where the camera type is stored
- Timing board (CCD cameras only)
 - The timing board digitizes the analog output signals of the CCD sensor.

7 Appendix

7.1 Troubleshooting and FAQ

Installation and connection

[Installation of the uEye software fails.](#)

You need administrator privileges to install the software. Operating the cameras, however, does not require administrator privileges.

[The camera is connected to the PC, but cannot be opened in uEye Demo.](#)

Check the [LED on the camera](#) (except *USB uEye LE*):

- LED is **red**: Camera detection failed. Check whether the uEye driver software has been installed. Disconnect and reconnect the camera to the USB cable. The camera should then be correctly recognized.
If the camera is still not listed in the *uEye Camera Manager*, open the Windows Device Manager to check whether the camera has been correctly recognized. If recognition was successful, you will find an entry in the format "uEye UI-xxxx-xx Series" under "Universal Serial Bus Controllers." A question mark or exclamation mark before the entry indicates that camera was not correctly recognized. You can remove the entry using the shortcut menu (right-click). Disconnect and reconnect the camera. The Found New Hardware Wizard will detect it as a new device and install the appropriate drivers. The camera should then be correctly recognized.
- LED is **green**: The camera is fully operational. Check whether the camera has been opened in a different application.
- LED is **off**: No power supply to the camera. Check the cable, the connectors and, if applicable, the power supply to the hubs. With *USB uEye SE* cameras, check whether any pins of the [micro D-sub connector](#) have been bent.
- LED **flashes**: A fault has occurred in the camera hardware. Please contact the [uEye Support](#).

USB uEye camera operation

[The camera can be opened in the software, but captures images sporadically or not at all.](#)

Check the [status bar](#) in the *uEye Demo* software. If the status bar indicates transfer errors, the camera speed settings are too high for the system you are using. Check the following:

- Use only USB 2.0 certified cables and hubs.
- Do not use any passive extension cables.
- Do not connect the camera to the USB ports on the front of the PC, but to the ones directly on the mainboard. You will find those USB ports at the back of the PC.

In addition, check the following camera settings in the software:

- Pixel clock frequency: Reduce the [pixel clock](#) if data transfer errors occur. When you are operating more than one USB camera on one port, the pixel clock of all the cameras added together should not exceed about 50 MHz.

[The Hardware LUT and Hardware de-Bayering functions cannot be enabled for the USB uEye.](#)

These functions allow [hardware preprocessing](#) in the camera and are only available for the models of the *GigE uEye SE* (LUT only) and *GigE uEye HE* series (LUT and De-Bayering).

GigE uEye camera operation

[The camera is connected to the PC, but cannot be opened in uEye Demo.](#)

See if the camera is displayed in the *uEye Camera Manager*. In particular, check the entries for *Free* and *Avail*. (Available):

- Free = Yes, Avail. = Yes: The camera can be used.
- Free = Yes, Avail. = No: The camera has not been properly configured. Check the IP settings of the camera and the network card (see below).
- Free = No, Avail. = Yes: The camera has been correctly configured, but is currently paired with a different PC. The camera can only be paired with one PC.

Camera detection failed: The camera is connected to the PC, but not displayed in the uEye Camera Manager.

Check the LED on the camera:

- LED flashes **green**: The camera is fully operational. Check whether the camera has been opened in a different application.
- LED is **off**: No power supply to the camera. Check the AC adapter and the power supply cabling. Information on the power supply requirements for the camera is provided in the *Specifications: Electrical Specifications* chapter.

[Can I use the camera on a 100 Mbps network \(FastLAN\)?](#)

GigE uEye cameras can be used on standard GigE networks and on 100 Mbps networks. Please note, however, that the camera will only deliver very low frame rates in that case. You may need to adjust the Pixel Clock and Frame Rate parameters.

Using the *uEye Demo* software

[I have added comments and drawings to a camera image. How can I save the image with this data?](#)

To save a camera image with all the included text and drawings, select *Save window* from the Draw/Measure menu. The menu also provides an option for saving only the drawings, so you can load them again later.

7.2 Colour and Memory Formats

Each colour format supported by the *uEye* camera defines a different memory format. The following table shows the byte arrangement in memory:

API constant	Description	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
		0	7 8	15 16	23 24 31	32	39 40	47 48	55 56 63
IS_CM_BAYER_RG16	Raw Bayer (16)								
	Odd row	16 *		16 *		16 *		16 *	
	Even row	16 *		16 *		16 *		16 *	
IS_CM_BAYER_RG12	Raw Bayer (12)								
	Odd row	12		12		12		12	
	Even row	12		12		12		12	
IS_CM_BAYER_RG8	Raw Bayer (8)								
	Odd row	8	8	8	8	8	8	8	8
	Even row	8	8	8	8	8	8	8	8
IS_CM_MONO16	Grey value (16)	16 *		16 *		16 *		16 *	
IS_CM_MONO12	Grey value (12)	12		12		12		12	
IS_CM_MONO8	Grey value (8)	8	8	8	8	8	8	8	8
IS_CM_RGB10V2_PACKED	RGB30 (10 10 10)	10	10	10		10	10	10	
IS_CM_RGBA8_PACKED	RGB32 (8 8 8)	8	8	8		8	8	8	
IS_CM_RGBY8_PACKED	RGBY (8 8 8 8)	8	8	8	8	8	8	8	8
IS_CM_RGB8_PACKED	RGB24 (8 8 8)	8	8	8	8	8	8	8	8
IS_CM_BGR10V2_PACKED	BGR30 (10 10 10)	10	10	10		10	10	10	
IS_CM_BGRA8_PACKED	BGR32 (8 8 8)	8	8	8		8	8	8	
IS_CM_BGR8_PACKED	BGR24 (8 8 8)	8	8	8	8	8	8	8	8
IS_CM_BGRY8_PACKED	BGRY (8 8 8 8)	8	8	8	8	8	8	8	8
IS_CM_BGR565_PACKED	BGR16 (5 6 5)	5	6	5	5	6	5	5	6
IS_CM_BGR555_PACKED	BGR15 (5 5 5)	5	5	5	5	5	5	5	5
IS_CM_UYVY_PACKED	UYVY (8 8 8 8)	8	8	8	8	8	8	8	8
IS_CM_UYVY_MONO_PACKED	UYVY (8 8 8 8)	8	8	8	8	8	8	8	8
IS_CM_UYVY_BAYER_PACKED	UYVY (8 8 8 8)	8	8	8	8	8	8	8	8
IS_CM_CbYCrY_PACKED	CbYCrY (8 8 8 8)	8	8	8	8	8	8	8	8

Colour codes

- Red channel
- Green channel
- Blue channel
- Y / grey channel
- U component
- V component
- Cb component
- Cr component
- Unused (0)

Figure 319: Colour and memory formats



An asterisk (*) identifies formats which are filled starting with the most significant bit (MSB) but which may have less than the indicated number of payload bits, depending on the camera model.
For the RGB16 and RGB15 data formats, the MSBs of the internal 8-bit R, G and B colours are used.

7.3 Structure of the uEye Parameter File (INI File)

Using the `is_SaveParameters()` function, you can save the currently set *uEye* camera parameters to a file in INI format (*.ini). To load the saved settings, use `is_LoadParameters()`.



Only camera-specific ini files can be loaded.
The ini file you want to load has to match the paired camera model.
When loading an ini file, make sure that the image size (AOI) and colour depth parameters in the ini file match those in the allocated memory. Otherwise, display errors may occur.

uEye parameter files can also be created and edited manually. The following table shows the structure of the parameter file. The entries in square brackets [] indicate sections. If a section does not exist in the ini file, the corresponding camera parameters will not be modified when you load the file.

Structure of a uEye parameter file

Parameter	Description	Value range	Example
[Versions]			
ueye_api.dll	File version of the <i>uEye</i> API	-	3.32.0000
ueye_eth.sys	File version of the <i>GigE uEye</i> driver	-	3.32.0000
ueye_usb.sys	File version of the <i>USB uEye</i> driver	-	3.32.0000
ueye_boot.sys	File version of the <i>USB uEye</i> boot loader	-	3.32.0000
[Sensor]			
Sensor	Full name of the camera model	-	UI548xHE-C
[Image size]			
Image size settings			
Start X	Start point (X coordinate) in AOI mode	0...(max. width ¹). Width)	100
Start Y	Start point (Y coordinate) in AOI mode	0...(max. height ¹). Height)	100
Start X absolute	Activate absolute AOI positioning in the memory (see <code>is_SetImagePos()</code>)	0, 1	1
Start Y absolute	Activate absolute AOI positioning in the memory (see <code>is_SetImagePos()</code>)	0, 1	1
Width	Width of the AOI	Sensor-dependent 1)	2460
Height	Height of the AOI	Sensor-dependent 1)	1820
Binning	Activate binning mode and select factor	Sensor-dependent 2)	0

Parameter	Description	Value range	Example
Subsampling	Activate subsampling mode and select factor	Sensor-dependent 2)	0
[Timing]	Timing parameter settings		
Pixelclock	Current pixel clock of the camera	Sensor-dependent 1)	103
Framerate	Current frame rate	Depends on Pixelclock and image geometry	15.104458
Exposure	Current exposure time	Depends on Framerate	0.334059
[Selected Converter]	Sets the type of Bayer conversion for the specified colour format when using colour cameras (see is_SetColorConverter()). For a description of all colour formats, see the Colour and Memory Formats section.		
IS_SET_CM_RGB32	Colour format	0, 1, 2, 4, (8 ³)	2
IS_SET_CM_RGB24	Colour format	0, 1, 2, 4, (8 ³)	2
IS_SET_CM_RGB16	Colour format	0, 1, 2, 4, (8 ³)	2
IS_SET_CM_RGB15	Colour format	0, 1, 2, 4, (8 ³)	2
IS_SET_CM_Y8	Colour format	0, 1, 2, 4, (8 ³)	2
IS_SET_CM_RGB8	Colour format	0, 1, 2, 4, (8 ³)	2
IS_SET_CM_BAYER	Colour format	0, 1, 2, 4, (8 ³)	8
IS_SET_CM_UYVY	Colour format	0, 1, 2, 4, (8 ³)	2
IS_SET_CM_UYVY_MONO	Colour format	0, 1, 2, 4, (8 ³)	2
IS_SET_CM_UYVY_BAYER	Colour format	0, 1, 2, 4, (8 ³)	2
IS_CM_CBYCRY_PACKED	Colour format	0, 1, 2, 4, (8 ³)	8
IS_SET_CM_RGBY	Colour format	0, 1, 2, 4, (8 ³)	8
IS_SET_CM_RGB30	Colour format	0, 1, 2, 4, (8 ³)	8
IS_SET_CM_Y12	Colour format	0, 1, 2, 4, (8 ³)	8
IS_SET_CM_BAYER12	Colour format	0, 1, 2, 4, (8 ³)	8
IS_SET_CM_Y16	Colour format	0, 1, 2, 4, (8 ³)	8
IS_SET_CM_BAYER16	Colour format	0, 1, 2, 4, (8 ³)	8
IS_CM_RGBA8_PACKED	Colour format	0, 1, 2, 4, (8 ³)	2
IS_CM_RGB8_PACKED	Colour format	0, 1, 2, 4, (8 ³)	2
IS_CM_RGBY8_PACKED	Colour format	0, 1, 2, 4, (8 ³)	8
IS_CM_RGB10V2_PACKED	Colour format	0, 1, 2, 4, (8 ³)	8

Parameter	Description	Value range	Example
[Parameters]			
Additional image parameter settings			
Colormode	Sets the current colour mode	see Colour and Memory Formats	11
Brightness	Software correction of image brightness ^{x)}	0...255	100
Contrast	Software correction of image contrast ^{x)}	0...511	215
Gamma	Software correction of the gamma value	0.01...10.0	1.000000
Hardware Gamma	Sensor-based hardware correction of the gamma value	0, 1	0
Blacklevel Mode	Mode for black level correction of the sensor	0, 1, 32 ²⁾	1
Blacklevel Offset	Manual offset for black level correction of the sensor	0...255	0
Hotpixel Mode	Mode for hot pixel correction	0, 1, 2, 4 ²⁾	2
Hotpixel Threshold	Not used	-	0
GlobalShutter	Enables the Global Start shutter of the sensor	0, 1	0
[Gain]			
Sets the sensor gain control for image brightness			
Master	Master gain	0...100	0
Red	Red gain	0...100	6
Green	Green gain	0...100	0
Blue	Blue gain	0...100	6
GainBoost	Activate gain boost	0, 1	0
[Processing]			
Parameters for image pre-processing in the driver			
EdgeEnhancement	Enable edge enhancement	0...2	0
RopEffect	Image geometry change (Rop = raster operation), e.g. mirroring	0, 8, 16, 32, 64 ²⁾	0
Whitebalance	Enable software white balance	0, 1, 2, 4 ²⁾	0
Whitebalance Red	Red factor for software white balance	double value	1.000000
Whitebalance Green	Green factor for software white balance	double value	1.000000
Whitebalance Blue	Blue factor for software white balance	double value	1.000000
Color correction	Enable colour correction	0, 1, 2, 4, 80 ²⁾	1
Color_correction_factor	Set the colour correction factor	0.0...1.0	1.000000

Parameter	Description	Value range	Example
Bayer Conversion	Sets the size of the Bayer conversion mask for the current colour format when using colour cameras	1, 2 ²⁾	1
[Auto features]	Sets the parameters for automatic image control		
Auto Framerate control	Enable frame rate control	0, 1	0
Brightness exposure control	Enable exposure time control	0, 1	0
Brightness gain control	Enable sensor gain control	0, 1	0
Brightness control once	Carry out brightness control once	0, 1	0
Brightness reference	Reference value for brightness control	0...255	128
Brightness speed	Brightness control speed	0...100	50
Brightness max gain	Maximum gain for brightness control	0...100	100
Brightness max exposure	Maximum exposure time for brightness control	Depends on Pixelclock and image geometry	66.082816
Brightness Aoi Left	X start point of reference AOI for brightness control	0...(max. width ¹⁾ -Aoi Width)	0
Brightness Aoi Top	Y start point of reference AOI for brightness control	0...(max. height ¹⁾ -Aoi Height)	0
Brightness Aoi Width	Width of reference AOI for brightness control	Sensor-dependent 1)	2560
Brightness Aoi Height	Height of reference AOI for brightness control	Sensor-dependent 1)	1920
Auto WB control	Enable white balance control	0, 1	0
Auto WB offsetR	Red offset for white balance control	0...100	0
Auto WB offsetB	Blue offset for white balance control	0...100	0
Auto WB gainMin	Minimum gain for white balance control	0 <= gainMin <= gainMax <= 100	0
Auto WB gainMax	Maximum gain for white balance control	0 <= gainMin <= gainMax <= 100	100
Auto WB speed	White balance control speed	0...100	50
Auto WB Aoi Left	X start point of reference AOI for white balance control	0...(max. width ¹⁾ -Aoi Width)	0
Auto WB Aoi Top	Y start point of reference AOI for white balance control	0...(max. height ¹⁾ -Aoi Height)	0
Auto WB Aoi Width	Width of reference AOI for white balance control	Sensor-dependent 1)	2560
Auto WB Aoi Height	Height of reference AOI for white balance control	Sensor-dependent 1)	1920

Parameter	Description	Value range	Example
Auto WB Once	Carry out white balance control once	0, 1	0
[Trigger and Flash]		Sets the digital inputs/outputs	
Trigger delay	Delay of triggered image capture in μ s	Sensor-dependent 1)	15
Flash strobe	Activate flash output	0...6 2)	0
Flash delay	Delay of the flash signal in μ s	Depends on sensor setting, can be queried using <code>is_SetFlashDelay()</code>	0
Flash duration	Duration of the flash signal in μ s	Depends on sensor setting, can be queried using <code>is_SetFlashDelay()</code>	200

¹⁾ This information is provided in [Specifications: Sensors](#) chapter.

²⁾ For the parameters, please refer to the *uEye.h* header file provided in the `Develop\include` folder of the uEye installation directory (see also [Programmierhinweise](#)).

³⁾ Only for *GigE uEye HE* cameras.

^{x)} Function ist obsolete, see chapter [Obsolete Functions](#).

7.4 History of uEye Software Versions

New in Version 3.40

Hardware	Described in chapter
Support of new camera series <i>GigE uEye RE</i>	Welcome: GigE uEye RE Mechanical Specifications: GigE uEye RE Electrical Specifications: GigE uEye RE Accessories: GigE uEye RE
Support of new 10 Megapixel sensor of 149x / 549x cameras	Specifications: UI-149x / UI-549x
Higher frame rates with <i>GigE uEye SE</i> cameras	Sensor Data

Software	Described in chapter
Support of Direct3D graphics functions with Overlay The DirectDraw functions have been completely replaced by the new API function. This function allows image scaling and inserting overlay data into the camera's live image without flicker.	Camera properties: Format System Requirements
Firmware update on camera initialization The <i>GigE uEye SE</i> camera firmware is automatically updated on camera initialization in the sample programs.	Firmware and Camera Start-up
Operating <i>GigE uEye</i> cameras in DHCP environment	Important Notes on connecting GigE uEye Cameras
Extended Automatic Image Control: <ul style="list-style-type: none"> • Configurable hysteresis control • Internal image controls of UI-122x/522x sensor supported 	Automatic Image Control AES/AGC (Automatic Brightness Control)
Extended image information on <i>GigE uEye</i> cameras The <i>uEye Demo</i> shows the load of the camera's internal image buffers and the number of resent data packages.	uEye Demo: Image Infos

Information in this Manual	Described in chapter
New chapter: <i>Quick-start</i> This chapter explains in a nutshell how to configure your uEye camera and capture images.	Quick-start
<i>New chapter:</i> Firmware and Camera Start-up	Firmware and Camera Start-up
<i>New chapter:</i> All uEye models at a glance A table shows the most important features of each uEye series at a glance.	Model comparison
Exact measuring results for the trigger delay of all uEye models.	Specifications
Mechanical Specifications for OEM version of the <i>USB uEye ME</i> and <i>GigE uEye SE</i> series	GigE uEye SE OEM version USB uEye ME OEM version

New in Version 3.33

New features	Described in chapter
Support of new camera series <i>USB uEye ME</i>	Welcome: USB uEye ME Mechanical Specifications: USB uEye ME Mechanical Specifications: USB uEye ME Accessories: USB uEye ME
New information in the manual	Described in chapter
Detailed presentation of the <i>GigE uEye's</i> pixel preprocessing	Pixel Preprocessing in GigE uEye cameras

New in Version 3.32

New features	Described in chapter
Extended trigger mode The continuous trigger mode allows triggering the <i>uEye</i> repeatedly. The camera no longer has to be made ready for the next trigger before each image capture.	Operating Modes: Trigger Mode
New information in the manual	Described in chapter
Detailed presentation of all <i>uEye</i> operating modes	Operating Modes
Updated connected load data on every camera model	Specifications: Sensor Data
Wiring diagrams for the inputs/outputs of the <i>USB uEye LE</i>	USB uEye LE: Wiring

New in Version 3.31

New features	Described in chapter
Support of <i>GigE uEye SE</i> The <i>uEye</i> driver version 3.31 introduces the new camera series <i>GigE uEye SE</i> . This manual provides all the information you need to integrate and use the new camera.	Welcome: GigE uEye SE GigE uEye SE Specifications GigE uEye SE Camera Dimensions
Improved <i>uEye Camera Manager</i> features	uEye Camera Manager

New in Version 3.30

New features	Described in chapter
Serial interface of the <i>GigE uEye HE</i> The serial interface (RS232) on the <i>GigE uEye HE</i> allows the control of peripherals.	Serial Interface (RS232)
Test image function The camera transmits a selectable test image that you can use for testing the data transmission.	Camera properties: Test image
Color calculation in the camera (<i>GigE uEye HE</i> only). The <i>GigE uEye HE</i> can optionally calculate the color data from raw Bayer format directly in the camera. This reduces the load on the host computer's CPU. Color correction and	Camera properties: Color

color saturation are continuously adjustable.	
LUT/gamma curves in the camera (<i>GigE uEye HE</i> only). In addition, the <i>GigE uEye HE</i> can apply LUT and gamma curves to the image directly in the camera in order to adjust brightness, contrast and color distribution.	Camera properties: LUT/Gamma
Support of 10 and 12 bit sensor data Some sensors can output images with a color depth of 10 to 12 bits. This data can now be processed by the <i>uEye</i> software.	Specifications: Sensors
New color formats <i>uEye</i> driver version 3.30 supports a wide range of new color formats for all <i>uEye</i> cameras. These include: <ul style="list-style-type: none"> ○ RGB/BGR 30 ○ RGBY ○ Y12 ○ YCbCr ○ Enhanced YUV 	Camera properties: Color
Subsampling for GigE uEye HE cameras with CCD sensors The <i>GigE uEye HE</i> can also perform binning and subsampling for CCD sensors. Binning and subsampling are supported both in the horizontal and vertical direction, and allow higher frame rates. For CMOS sensors, subsampling takes place directly in the sensor and is supported by all <i>uEye</i> models.	Specifications: Sensors
Full support of <i>Windows Vista</i> (32 Bit) From driver version 3.30 onwards, all <i>uEye</i> cameras will run under <i>Windows Vista</i> 32.	System Requirements
Localization of the <i>uEye Camera Manager</i> The <i>uEye Camera Manager</i> offers new features and now also supports over 10 languages that can be switched anytime.	uEye Camera Manager

7.5 History of API Functions

New functions in driver version 3.40.000

<code>is_DirectRenderer()</code>
<code>is_GetImageInfo()</code>
<code>is_GetDuration()</code>
<code>is_GetSensorScalerInfo()</code>
<code>is_SetSensorScaler()</code>

New functions in driver version 3.32.000

<code>is_GetTimeout()</code>
<code>is_SetTimeout()</code>
<code>is_SetTriggerCounter()</code>

New functions in driver version 3.30.000

<code>is_GetCameraLUT()</code>
<code>is_GetCaptureErrorInfo()</code>
<code>is_GetColorConverter()</code>
<code>is_GetComportNumber()</code>
<code>is_GetSupportedTestImages()</code>
<code>is_GetTestImageValueRange()</code>
<code>is_ResetCaptureErrorInfo()</code>
<code>is_SetCameraLUT()</code>
<code>is_SetColorConverter()</code>
<code>is_SetSensorTestImage()</code>

New functions in driver version 3.20.000

<code>is_SetOptimalCameraTiming()</code>
--

New functions in driver version 3.10.000

<code>is_EnableHdr()</code>
<code>is_GetHdrKneepointInfo()</code>
<code>is_GetHdrKneepoints()</code>
<code>is_GetHdrMode()</code>
<code>is_SetHdrKneepoints()</code>

New functions in driver version 3.00.000

<code>is_GetEthDeviceInfo()</code>
<code>is_SetAutoCfgIpSetup()</code>
<code>is_SetPacketFilter()</code>
<code>is_SetPersistentIpCfg()</code>
<code>is_SetStarterFirmware()</code>

Index

- A -

ActiveX	411
Ambient conditions	502
AOI (Area of Interest)	75, 115
Auto Exposure (AES)	51, 55, 70, 89, 119
Auto Frame Rate (AFR)	89, 119
Auto Gain (AGC)	51, 55, 73, 89, 118, 119
Auto White Balance (AWB)	51, 55, 119
Hysteresis	91
Automatic Image Control	55
Hysteresis	119
Programming	160, 283
Set reference area	51
AVI Recording	60, 365
Playback	96

- B -

Bayer Filter	
Bayer conversion	77, 80, 109
Binning	75, 116
Black level correction	73, 118
Broadcast	132

- C -

Camera ID	39
Camera Manager	35
Camera list	36
Capture	
Freerun synchronisation	143
Live mode (freerun)	102
Single snap mode	102
Trigger mode	83, 103, 127
C-mount / CS-mount	490
Color correction	80
Color formats	77, 559
COM port -> Serial interface	128

- D -

Deinstallation	25
DHCP	33, 43, 132
Digital in-/output	83, 85, 104, 127, 128, 505
Direct3D	
Overlay display	178
DirectShow	39
DirectX	23, 178
Display	
Bitmap mode (DIB)	106

Colour formats	559
Direct3D	23, 106
DirectDraw	386
modes (uEye Demo)	77
Display overlay	142
Driver version	38

- E -

EEPROM of the camera	39, 505
Error	
Troubleshooting	557
Errors	
Error messages	413
Transfer failed	51, 67, 68, 118
Exposure time	70, 118

- F -

FALCON functions	419
Filter glasses	498
Firmware	120
Flange back distance	490
Adjusting	492
Flash	103, 111, 127
Global Flash	85
Frame rate	70, 118
Freerun -> Capture	102

- G -

Gain	73, 118
Gamma	73, 81
GigE	
Cable	131
Connection	131
GigE uEye network service	40
Global Shutter	111
Global Start Shutter	111
GPIO	85, 128
Graphics card	23, 106

- H -

HDR mode (UI-122x/522x)	87
Heartbeat	44, 122, 132
Hot pixel correction	77, 93

- I -

Image display	
DIB mode	142
Direct3D	142
Overlay	142
Image display -> Display	106
Image scaling (UI-149x/549x)	95

INI file -> Parameter file	560	Serial interface (GigE uEye HE)	128, 141
Installation	25, 26	Serial interface (on GigE uEye HE cameras)	40
IP 65 / 67 norm	503	Shock resistance	502
IP address	122	Shutter	
of the Network interface card	40	Global Shutter	111
persistent camera IP	43	Global Start Shutter	111
IR cut filter	498	Rolling Shutter	111
- L -		Standby mode of the camera	104
Linux	23, 26, 416	Starter firmware	43, 120, 122
LUT	81	uploading	43
- M -		Status LED	29
Memory board	29, 386	GigE uEye	32
Memory formats	559	USB uEye	28
Micro lenses	108	Subnet	132
- N -		Subsampling	75, 116
Network card	33	System requirements	23
Network interface card	23	- T -	
- O -		Temperature range	502
Obsolete Functions	386	Test image	93
Operating system	23	Thread	412
Overlay -> Display	106	Transfer failed -> Errors	51
- P -		Trigger -> Image capture	103
Pairing	132, 133	- U -	
Parameter file (INI file)	53, 560	UDP	132
Pixel clock	70	uEye Camera Models	17, 421, 422
Pixel pre-processing (GigE uEye)	81, 125	uEye Demo	48
GigE uEye HE	126	uEye support	557
GigE uEye SE	125	uEye.h	410, 411
Position Accuracy of the Sensor	497	uEye_Api.dll	410, 411
- Q -		USB	544
Quick-Start		Bandwidth	130
Connection	20	Cable	130
Image capture	20	Connection	130
Programming	134	Hub	27
- R -		Standard	129
Raw Bayer -> Bayer Filter	109	Topology	129
Return values	413	- V -	
Rolling Shutter	111	Vibration resistance	502
RS-232 -> Serial interface	128	- W -	
- S -		Windows	
Sensor formats	108	Installation	25