
THORLABS

CCD and CMOS Cameras

DCU223x, DCU224x

DCC1240x

DCC1545M, DCC1645C

DCC3240X

.NET Programming Interface



2015

Version: 4.40
Date: 4/13/2015

Contents

Foreword	5
1 General Information	6
1.1 What is new in this version?	6
1.2 About this manual	7
2 Installation and requirements	8
2.1 System requirements	8
2.2 DCx Camera Software Installation	9
2.3 Connecting a DCx Camera	9
2.4 Getting started quickly	10
3 .NET library	11
3.1 uc480	11
3.1.1 Camera	11
3.1.1.1 Acquisition	14
3.1.1.2 AutoFeatures	17
3.1.1.3 BlackLevel	58
3.1.1.4 Color	62
3.1.1.5 Device	76
3.1.1.6 DirectRenderer	120
3.1.1.7 Display	133
3.1.1.8 EdgeEnhancement	141
3.1.1.9 EEPROM	143
3.1.1.10 FaceDetection	145
3.1.1.11 Focus	154
3.1.1.12 Gain	164
3.1.1.13 Gamma	176
3.1.1.14 GigE	180
3.1.1.15 Hdr	185
3.1.1.16 Hotpixel	187
3.1.1.17 I2C	197
3.1.1.18 Image	199
3.1.1.19 ImageStabilization	204
3.1.1.20 Information	206
3.1.1.21 IO	214
3.1.1.22 Lut	228
3.1.1.23 Memory	241
3.1.1.24 Messaging	260
3.1.1.25 Parameter	263
3.1.1.26 PixelFormat	265
3.1.1.27 RopEffect	269

3.1.1.28	Saturation	270
3.1.1.29	ScenePreset	273
3.1.1.30	Sharpness	276
3.1.1.31	Size	278
3.1.1.32	TestImage	302
3.1.1.33	Timeout	306
3.1.1.34	Timing	307
3.1.1.35	Trigger	322
3.1.1.36	Video	334
3.1.1.37	Zoom	339
3.1.1.38	Camera	342
3.1.1.39	Exit	344
3.1.1.40	Init	344
3.2	uc480.Configuration	345
3.2.1	BootBoost	346
3.2.1.1	AddId	347
3.2.1.2	ClearIdList	347
3.2.1.3	GetEnable	348
3.2.1.4	GetIdList	348
3.2.1.5	RemoveId	348
3.2.1.6	SetEnable	349
3.2.1.7	SetIdList	349
3.2.2	CPUIdleState	350
3.2.2.1	GetDisableOnOpen	350
3.2.2.2	GetEnable	350
3.2.2.3	GetSupported	351
3.2.2.4	SetDisableOnOpen	351
3.2.3	InitialParameterset	352
3.2.3.1	GetEnable	352
3.2.3.2	GetSupported	352
3.2.3.3	SetEnable	353
3.2.4	Ipo	353
3.2.4.1	GetEnable	354
3.2.4.2	GetSupported	354
3.2.4.3	SetEnable	354
3.2.5	OpenMP	355
3.2.5.1	GetEnable	355
3.2.5.2	GetEnableDefault	355
3.2.5.3	GetSupported	356
3.2.5.4	SetEnable	356
3.3	uc480.Configuration.GigE	356
3.3.1	AutoConfigIp	357
3.3.1.1	Get	357

3.3.1.2	GetSupported	358
3.3.1.3	Set	358
3.3.2	PacketFilter	359
3.3.2.1	Set	359
3.3.3	Persistently	359
3.3.3.1	Get	360
3.3.3.2	GetSupported	360
3.3.3.3	Set	361
3.3.4	StarterFirmware	361
3.3.4.1	GetUploadDuration	362
3.3.4.2	GetUploadDurationCP	362
3.3.4.3	GetUploadDurationSE	363
3.3.4.4	Upload	363
3.4	uc480.Info	363
3.4.1	Camera	364
3.4.1.1	GetCameraList	364
3.4.1.2	GetDeviceInfo	365
3.4.1.3	GetNumberOfDevices	365
3.4.2	System	365
3.4.2.1	GetNetVersion	366
3.4.2.2	GetApiVersion	366
3.5	uc480.Tools	366
3.5.1	Video	367
3.5.1.1	AddFrame	367
3.5.1.2	Close	368
3.5.1.3	Exit	368
3.5.1.4	GetFrameCount	368
3.5.1.5	GetLostCount	369
3.5.1.6	GetSize	369
3.5.1.7	Init	370
3.5.1.8	Open	370
3.5.1.9	ResetCounter	370
3.5.1.10	SetFramerate	371
3.5.1.11	SetImageSize	371
3.5.1.12	SetQuality	372
3.5.1.13	Start	372
3.5.1.14	Stop	372
3.6	uc480.Types	373
3.6.1	AoiSequenceParameter	374
3.6.2	CameralInfo	375
3.6.3	CameralInformation	376
3.6.4	CaptureStatus	377
3.6.5	CaptureStatus.CaptureStatusApi	378

3.6.6	CaptureStatus.CaptureStatusDriver	378
3.6.7	CaptureStatus.CaptureStatusEth	379
3.6.8	CaptureStatus.CaptureStatusUsb	379
3.6.9	ConversionParameter	380
3.6.10	DeviceInfoControl	380
3.6.11	DeviceInfoHeartbeat	380
3.6.12	DeviceInformation	381
3.6.13	FaceDetectionInformation	381
3.6.14	ImageFormatInfo	383
3.6.15	ImageInfo	389
3.6.16	KneePointInformation	391
3.6.17	LutState	391
3.6.18	LutSupportInfo	392
3.6.19	MultilIntegrationCycles	392
3.6.20	MultilIntegrationScope	392
3.6.21	SensorInfo	393
3.6.22	SensorScalerInformation	394
3.6.23	TimestampConfiguration	395
3.7	uc480.Types.AutoFeature	395
3.7.1	BrightStatus	396
3.7.2	Information	396
3.7.3	WhitebalanceChannelStatus	399
3.7.4	WhitebalanceStatus	399
3.8	uc480.Types.ETH	399
3.8.1	AdapterInformation	400
3.8.2	AutoConfigIPSetup	400
3.8.3	DeviceHeartbeat	400
3.8.4	DeviceInfoControl	402
3.8.5	DeviceInformation	404
3.8.6	DriverInformation	404
3.8.7	EthernetConfiguration	404
3.8.8	IpConfiguration	405
3.8.9	MacAddr	405
3.9	Complete list of all returns values	405
4	.NET version history	410
5	Appendix	414
5.1	PCs with Energy Saving CPU Technology	414
5.2	Exclusion of Liability and Copyright	415
5.3	Thorlabs Worldwide Contacts	416

The product development team at Thorlabs Scientific Imaging is continually looking for new challenges to push the boundaries of scientific cameras. Our team of senior research and development engineers welcomes your input on our products and how we can help meet your advanced imaging needs.

Thorlabs Scientific Imaging

Warning

Sections marked by this symbol explain dangers that might result in personal injury or death. Always read the associated information carefully, before performing the indicated procedure.

Attention

Paragraphs preceded by this symbol explain hazards that could damage the instrument and the connected equipment or may cause loss of data.

1 General Information

The uc480 .NET manual contains all information that you need for programming applications with uc480 cameras and the .NET interface. The uc480 .NET interface is part of the comprehensive software package for DCx cameras. In addition to the drivers, this software package features the DCx Camera Manager, the uc480 Cockpit and a Software Development Kit (SDK) for creating your own uc480 programs under Windows. Numerous demo applications make it easy for you to get started with uc480 programming.



For detailed information about your DCx camera, refer to the DCx_User_and_SDK_Manual.

In the [What is new in this version?](#) chapter you find information about the changes in this version.

The .NET interface offers a user-friendly and object-oriented programming interface (at least version 3.5 or higher is required).

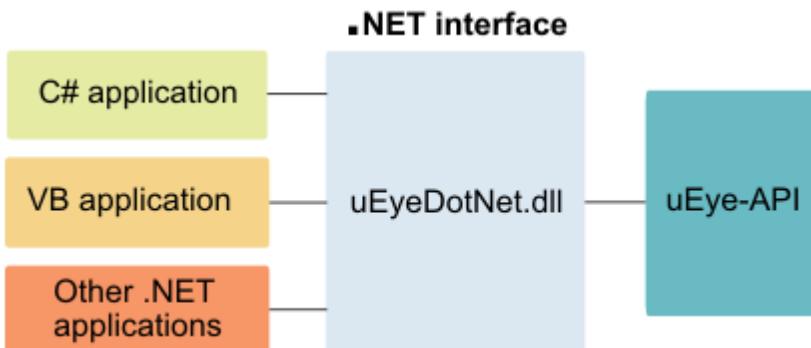


Fig. 1: .NET interface

Via the uc480DotNet.dll the code is encapsulated so that you can use different programming languages for the .NET interface e.g. C#, C++ or VB.

Examples for .NET programming

- **C#:** uc480.NET C# SimpleLive
- **Visual Basic:** uc480.NET VB SimpleLive

1.1 What is new in this version?

The .NET interface has been reworked and methods have been renamed to get the relationships more clearly. For the same reasons classes were newly grouped. Note also, that there are no more unsafe methods.



Note that version 4.01 is not downward compatible to version 4.00.

Also the message and event handling has been changed, please refer to the corresponding chapters ([Messaging](#) or [Camera](#)) or see the uc480.NET C# SimpleLive demo.

Some methods are overloaded, so they can be called with different parameters. So a new instance can be created in various ways (e.g. [Allocate\(\)](#)).

New in version 4.40

New classes and methods:

- [Feature](#) class: [MultiIntegration](#)
- [Flash](#) class: [GetAutoFreerunDefault](#), [GetAutoFreerunEnable](#) and [SetAutoFreerunEnable](#)
- New methods in the [Lut](#) class.

Added new features:

- Crosshairs effect for XS camera: [ImageEffect](#) class
- New auto Log mode for camera models UI-124x/UI-324x/UI-524x and UI-125x/UI-325x/UI-525x

Changes classes and methods:

- [Messaging](#) class: changed [Enable](#) and [Disable](#) methods and removed other methods

For information on changes in former version refer to [.NET version history](#).

1.2 About this manual

The DCx .NET Manual contains all the information you need for programming your own applications with your DCx Camera in the .NET environment. The DCx .NET interface is part of the comprehensive software package included with every DCx Camera. In addition to the drivers, the software package includes the uc480 Camera Manager, the uc480 Viewer and a Software Development Kit (SDK) for creating your own uc480 programs in Windows. Demo applications make it easier to start uc480 programming.

2 Installation and requirements



- [System requirements](#)
- [Configuring .NET](#)

2.1 System requirements

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

	Recommended
CPU speed	>2.0 GHz Intel Core i5 or Core i7
Memory (RAM)	8 GByte
For USB DCx cameras: USB host controller	USB 3.0 Super Speed Intel® motherboard chipset
Graphics card	Dedicated AGP/PCIe graphics card Latest version of Microsoft DirectX Runtime 9.0c
Operating system	Windows 8.1 32 or 64 bit Windows 7 32 or 64 bit

Drivers for network cards

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

USB interface

- Onboard USB 2.0 ports usually provide significantly better performance than PCI and PCMCIA USB adapters.
- Current generation CPUs with energy saving technologies can cause bandwidth problems on the USB bus. See section on PCs With Energy Saving CPU Technology.

Large multi-camera systems

Connecting a large number of cameras to a single PC may require a large working memory (RAM). This is especially the case when many cameras with high sensor resolution are used.

If you want to set up such a system we recommend to use PCs with 64 bit operating systems and more than 4 GB of RAM.



Note on color cameras with high frame rates

For uc480 color cameras, the color conversion is done by software in the PC. When you use a color camera with a high frame rate, the conversion might lead to a high CPU load. Depending on the PC hardware used you might not be able to reach the camera's maximum frame rate.

Direct3D graphics functions

The uc480 driver can use Direct3D to display the camera image with overlay information (Microsoft

DirectX Runtime had to be installed). 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.

OpenGL graphics functions

For OpenGL version 1.4 or higher must be installed. The OpenGL graphics functions do not work with QT under Linux.

2.2 DCx Camera Software Installation

Attention

1. You need administrator privileges to install the software.
2. Please install the software prior to connect a DCx camera!

The software for DCx camera is delivered on a CD. Alternatively, or if the CD is lost, the software can be downloaded from [Thorlabs' website](#). Please insert the delivered with the DCx camera CD to the drive of your PC and start the software installation as shown in the Quick Start Guide.

2.3 Connecting a DCx Camera

Please install the software first as described in the Quick Start Guide. Connect the DCx camera to the PC, using the USB cable. The camera will be recognized automatically and the necessary driver software is being installed.

When the camera has been correctly installed, the LED on the back of the camera lights up green.

Note

The first time you connect a USB DCx camera to a USB port under Windows, two driver files will be registered. The first file (uc480 boot) contains the generic driver, the second file the model-specific driver.

The model will be immediately recognized whenever you connect the camera to this port again. If you use a different port, the registration will be repeated. Under Windows the camera will show up in the uc480 Camera Manager's camera list.

The DCx 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 commercially. 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 DCx Cameras.

Note

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

Attention

USB cables with non-standard connectors must be connected to the camera first and then to the PC. Otherwise the camera might not be recognized correctly.

2.4 Getting started quickly



Note: You should be familiar with the use of Windows Forms.

Setting-up the IDE

1. Copy the `uc480DotNet.dll` file into your project directory.
2. Open your project in Visual Studio.
3. In the "Solution Explorer" right-click on the "References" entry.
4. Select "Add reference". A dialog for adding references opens.
5. Open the "Browse" tab.
6. Select the `uc480DotNet.dll`.
7. Click on "OK".

Using the .NET interface

1. Create a `uc480` .NET object, e.g.
`uc480.Camera cam = new uc480.Camera();`
2. Now you can start working with the .NET interface.

Capturing the first image

1. Initialize the camera: `cam.Init();`
2. Allocate an default image memory: `cam.Memory.Allocate(out s32MemId);`
 You can also allocate an image memory by using: `cam.Memory.Allocate();`
3. Capture a live image with `cam.Acquisition.Capture(s32Wait);` or a single image with `cam.Acquisition.Freeze(s32Wait);`
 With `s32Wait = uc480.Defines.DeviceParameter.Wait` the image acquisition waits until an image is captured and returns afterwards. With `s32Wait = uc480.Defines.DeviceParameter.DontWait` the image acquisition returns immediately.
4. Display the image on the screen: `cam.DisplayImage.Set(s32MemId, s32DisplayHandle, s32Mode)`
`s32MemId` is the ID of the image memory to be displayed. `s32DisplayHandle` is required to show the image e.g. in a picture box of Window Forms. `s32Mode` selects different rendering modes, e.g. `s32Mode = uc480.Defines.DisplayRenderMode.FitToWindow.`

Example Code

Example programs can be found in

```
C:\Program Files\Thorlabs\Scientific Imaging\DCx Camera Support
\Develop\Source
```

3 .NET library

The classes of the .NET interface are grouped into classes. All method calls are automatically available via the IntelliSense support in Visual Studio.

Namespace

Contains classes (`uc480`).

Class

These classes contain a set of methods or classes. A class represents the function calls of the API (e.g. `uc480.Exposure`).

Methods

The single methods in the classes are structured logically by `uc480.Acquisition.Capture()`, `uc480.Acquisition.Stop()` etc.

See also:

- [uc480](#)
- [uc480.Configuration](#)
- [uc480.Configuration.GigE](#)
- [uc480.Info](#)
- [uc480.Tools](#)
- [uc480.Types](#)
- [uc480.Types.AutoFeature](#)
- [uc480.Types.ETH](#)
- [Complete list of all returns values](#)

3.1 uc480

The `uc480` class provides one class for using `uc480` cameras (capturing images, setting camera parameter, etc). The following class exists:

- [Camera](#)

3.1.1 Camera

The `Camera` class provides methods for controlling `uc480` cameras and capturing images. The following classes and methods exist:

- | | | |
|-----------------------------------|--------------------------------------|-------------------------------|
| • Acquisition | • GigE | • PixelFormat |
| • AutoFeatures | • Hdr | • RopEffect |
| • BlackLevel | • Hotpixel | • Saturation |
| • Color | • I2C | • ScenePreset |
| • Device | • Image | • Sharpness |
| • DirectRenderer | • ImageStabilization | • Size |
| • Display | • Information | • TestImage |
| • EdgeEnhancement | • IO | • Timeout |
| • EEPROM | • Lut | • Timing |
| • FaceDetection | • Memory | • Trigger |
| • Focus | • Messaging | • Video |
| • Gain | • Parameter | • Zoom |
| • Gamma | | |

Methods

<u>Camera</u>	Starts the driver and establishes the connection to the camera. When using Direct3D for image display, you can pass a handle to the output window. The method uses internally the <code>Init()</code> method. Only when no parameters are passed, <u>Init()</u> must be called.
<u>Exit</u>	Disables the camera handle and releases the data structures and memory areas taken up by the uc480 camera.
<u>Init</u>	Starts the driver and establishes the connection to the camera.

Events

The `Camera` class also provides special camera events:

Event	Description
uc480.Camera.EventAutoBrightnessFinished	The automatic brightness control in the run-once mode is completed.
uc480.Camera.EventAutoFocusFinished	Automatic focus control is finished (XS only)
uc480.Camera.EventCameraMemory	In the camera memory mode an image acquisition iteration is finished.
uc480.Camera.EventCaptureStatus	There is an information about image capturing available. This information can be requested by Information.GetCaptureStatus() .
uc480.Camera.EventConectionSpeedChanged	The connection speed of a USB 3 uc480 camera changed from USB 2.0 to USB 3.0 or from USB 3.0 to USB 2.0.
uc480.Camera.EventDeviceReconnect	A camera initialized with Init() and disconnected afterwards was reconnected.
uc480.Camera.EventDeviceRemove	A camera was removed. This is independent of the device handle.
uc480.Camera.EventExtTrigger	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 uc480.Camera.EventFrame processing event.
uc480.Camera.EventFirstPacket	The first data packet of the image was transferred to the PC. This is the earliest time for determining if the image exposure is finished.
uc480.Camera.EventFrame	A new image is available.
uc480.Camera.EventMemoryModeFinished	In the camera memory mode an image acquisition iteration is finished.
uc480.Camera.EventSequence	The sequence is completed.
uc480.Camera.EventSteal	An image extracted from the overlay is available.
uc480.Camera.EventWhitebalanceFinished	The automatic white balance control is completed.

Example

Example for registering the frame event

```

private void InitCamera()
{
    uc480.Camera Cam = new uc480.Camera();
    ....
    ....
    // Connect to FrameEvent
    Cam.EventFrame += onFrameEvent;
}

private void onFrameEvent(object sender, EventArgs e)
{
    // sender is our camera object
    uc480.Camera Cam = sender as uc480.Camera;
    ...
}

```

3.1.1.1 Acquisition

The `Acquisition` class provides methods for image acquisition.

Methods

Method	Description
Capture	Activates the camera's live video mode (free run mode). The driver transfers the images to an allocated image memory or, if Direct3D is used, to the graphics card.
Freeze	Acquires a single image from the camera.
HasStarted	Checks whether the image digitizing process has started.
IsFinished	Checks whether an image has been captured and stored completely in the image memory.
Stop	Stops live mode or cancels a hardware triggered image capture in case the exposure has not yet started.

3.1.1.1.1 Capture

Class

[uc480.Acquisition](#)

Accessible

Camera.Acquisition

Syntax

```

uc480.Acquisition.Capture()
uc480.Acquisition.Capture(int wait)
uc480.Acquisition.Capture(uc480.Defines.DeviceParameter param)

```

Description

Activates the camera's live video mode (free run mode). The driver 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 [Allocate\(\)](#) and designated as active image memory using [SetActive\(\)](#). Using [GetActive\(\)](#), 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 [LockSequenceId\(\)](#) 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.



Note: `uc480.Acquisition.Capture()` uses the default setting "do not wait".

For further information on the image capture modes, see the "How to proceed: Image capture" section in the uc480 manual.

Parameter

none	Default: do not wait
param/wait	<code>uc480.Defines.DeviceParameter.DontWait</code> <code>uc480.Defines.DeviceParameter.Wait</code>
	Time t (Value range: 4...429496729): Timeout value for image acquisition

3.1.1.1.2 Freeze

Class

[uc480.Acquisition](#)

Accessible

Camera.Acquisition

Syntax

```
uc480.Acquisition.Freeze()
uc480.Acquisition.Freeze(int wait)
uc480.Acquisition.Freeze(uc480.Defines.DeviceParameter param)
```

Description

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 [Set\(\)](#). A hardware triggered image acquisition can be cancelled using [Stop\(\)](#) if exposure has not started yet.



Note: `uc480.Acquisition.Freeze()` uses the default setting "do not wait".

For further information on the image capture modes, see the "How to proceed: Image capture" section in the uc480 manual.

Parameter

none	Default: do not wait
param/wait	<code>uc480.Defines.DeviceParameter.DontWait</code> <code>uc480.Defines.DeviceParameter.Wait</code>
	Time t (Value range: 4...429496729): Timeout value for image acquisition

3.1.1.3 HasStarted

Class

[uc480.Acquisition](#)

Accessible

Camera.Acquisition

Syntax

uc480.Acquisition.HasStarted(out bool started)

Description

Checks whether the image digitizing process has started. This method is helpful when [Freeze\(\)](#) was called with the uc480.Defines.DeviceParameter.DontWait parameter.

Parameter

started	1 = Image capturing has started. 0 = Image capturing has not started yet.
---------	--

3.1.1.4 IsFinished

Class

[uc480.Acquisition](#)

Accessible

Camera.Acquisition

Syntax

uc480.Acquisition.IsFinished(out bool finished)

Description

Checks whether an image has been captured and stored completely in the image memory. This method is helpful if [Freeze\(\)](#) was called with the uc480.Defines.DeviceParameter.DontWait parameter.

Parameter

finished	1 = Digitizing of the image is completed. 0 = Digitizing of the image is not completed yet.
----------	--

3.1.1.5 Stop

Class

[uc480.Acquisition](#)

Accessible

Camera.Acquisition

Syntax

uc480.Acquisition.Stop()
uc480.Acquisition.Stop(int wait)
uc480.Acquisition.Stop(uc480.Defines.DeviceParameter param)

Description

Stops live mode or cancels a hardware triggered image capture in case the exposure

has not yet started.



Note: This method uses the default setting "do not wait".

Parameter

none	The method returns immediately.
param/wait	<ul style="list-style-type: none"> • uc480.Defines.DeviceParameter.DontWait: The method returns immediately. Digitizing the image is completed in the background. • uc480.Defines.DeviceParameter.Wait: The method waits until the image save is complete. • uc480.Defines.DeviceParameter.Force: Digitizing is stopped immediately.

Example

```
public void StopCamera()
{
    uc480.Camera cam = new uc480.Camera(1);
    uc480.Defines.Status statusRet;
    statusRet = uc480.Acquisition.Stop(uc480.Defines.DeviceParameter.Wait);
}
```

3.1.1.2 AutoFeatures

The `AutoFeatures` class provides classes for querying and setting status information on the automatic image control features. The following classes and method exist:

- [Sensor](#)
- [Software](#)

Methods

Method	Description
GetInfo	Returns status information on the automatic image control features.

3.1.1.2.1 Sensor

The `Sensor` class provides methods for controlling the sensor-specific automatic image control features.

- 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 exposure 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 control cannot be used simultaneously.
 - The `auto gain` function can only be used for cameras with master gain control. Auto white balance is only available for cameras with hardware RGB gain control.
 - The `sensor's internal auto features` are only supported by the sensors of the following camera models.
- Please also read the notes on using these sensors.

- UI-112x/UI-512x (HDR sensor)
- UI-122x/UI-322x/UI-522x
- UI-1008XS
- XS
- The following functions are currently only supported by the UI-1008XS:
Sensor's internal white balance, photometry, auto contrast correction, backlight compensation, and anti flicker mode.
- The following functions are currently only supported by the XS:
Sensor's internal white balance and backlight compensation.



Notes on models with HDR sensor

In the UI-112x/UI-512x models with HDR sensor, some auto features have a different functionality or are not available:

- The HDR sensor has no gain. The gain methods control the white level adjustment.
- The shutter, frame rate and white balance methods are not available for this sensor.



Notes on the sensor's internal control functionality

Automatic control by the sensor and the software is not possible simultaneously. To use the sensor's internal control functionality, disable [software](#) control, and vice versa.



Notes on the UI-1008XS

- The auto contrast correction and backlight compensation functions cannot be used simultaneously.
- When the anti flicker function is active, the auto exposure shutter settings must not be changed.

The following classes exist:

- [AntiFlicker](#)
- [BacklightCompensation](#)
- [Contrast](#)
- [Framerate](#)
- [Gain](#)
- [GainShutter](#)
- [Shutter](#)
- [Whitebalance](#)

The [AntiFlicker](#) class provides methods for setting the anti flicker mode of the UI-1008XS camera model. With the anti flicker option, the frame rate and exposure time settings are adjusted during automatic image control so that flicker from the lighting is reduced in the image.

Methods

Method	Description
Get	Returns the anti flicker mode.
GetDefault	Returns the default anti flicker mode.
GetSupported	Returns the supported anti flicker mode.
Set	Sets the anti flicker mode.

Class

[uc480.AutoFeaturesSensorAntiFlicker](#)

Accessible

Camera.AutoFeatures.Sensor.AntiFlicker

Syntax

```
uc480.AutoFeaturesSensorAntiFlicker.Get(out uc480.Defines.Whitebalance.AntiFlickerMode mode)
```

Description

Returns the anti flicker mode.

Parameter

mode	Returns the mode: <ul style="list-style-type: none"> • uc480.Defines.Whitebalance.AntiFlickerMode.Disable: Anti flicker mode is disabled. • uc480.Defines.Whitebalance.AntiFlickerMode.SensorAuto: The anti flicker mode is selected automatically (50 or 60 Hz). • uc480.Defines.Whitebalance.AntiFlickerMode.Sensor50Fixed: The anti flicker mode is set to a fixed value of 50 Hz. • uc480.Defines.Whitebalance.AntiFlickerMode.Sensor60Fixed: The anti flicker mode is set to a fixed value of 60 Hz.
------	--

Class

[uc480.AutoFeaturesSensorAntiFlicker](#)

Accessible

Camera.AutoFeatures.Sensor.AntiFlicker

Syntax

```
uc480.AutoFeaturesSensorAntiFlicker.GetDefault(out uc480.Defines.Whitebalance.AntiFlickerMode mode)
```

Description

Returns the default anti flicker mode.

Parameter

- mode: Returns the default mode (see [Get\(\)](#)).

Class

[uc480.AutoFeaturesSensorAntiFlicker](#)

Accessible

Camera.AutoFeatures.Sensor.AntiFlicker

Syntax

```
uc480.AutoFeaturesSensorAntiFlicker.GetSupported(out uc480.Defines.Whitebalance.AntiFlickerMode mode)
```

Description

Returns the supported anti flicker mode.

Parameter

- `mode`: Returns the supported modes (see also [Get\(\)](#)).

Class

[uc480.AutoFeaturesSensorAntiFlicker](#)

Accessible

Camera.AutoFeatures.Sensor.AntiFlicker

Syntax

`uc480.AutoFeaturesSensorAntiFlicker.Set(uc480.Defines.Whitebalance.AntiFlickerMode mode)`

Description

Sets the anti flicker mode.

Parameter

mode	<p>Anti flicker mode to be set:</p> <ul style="list-style-type: none"> • <code>uc480.Defines.Whitebalance.AntiFlickerMode.Disable</code>: Anti flicker mode is disabled. • <code>uc480.Defines.Whitebalance.AntiFlickerMode.SensorAuto</code>: The anti flicker mode is selected automatically (50 or 60 Hz). • <code>uc480.Defines.Whitebalance.AntiFlickerMode.Sensor50Fixed</code>: The anti flicker mode is set to a fixed value of 50 Hz. • <code>uc480.Defines.Whitebalance.AntiFlickerMode.Sensor60Fixed</code>: The anti flicker mode is set to a fixed value of 60 Hz.
------	---

The `BacklightCompensation` class provides methods for adjusting automatic brightness control in the USB uc480 XS so that the influence of a bright backlight on the control function is reduced. This way, you can capture a clearer image of dark objects against bright background.

Methods

Method	Description
GetEnable	Returns the backlight compensation status.
GetSupported	Returns if backlight compensation is supported.
SetEnable	Enables/disables backlight compensation for automatic brightness control.

Class

[uc480.AutoFeaturesSensorBacklightCompensation](#)

Accessible

Camera.AutoFeatures.Sensor.BacklightCompensation

Syntax

```
uc480.AutoFeaturesSensorBacklightCompensation.GetEnable(out bool enable)
```

Description

Returns the backlight compensation status.

Parameter

enable	1 = Backlight compensation is enabled. 0 = Backlight compensation is disabled.
--------	---

Class

[uc480.AutoFeaturesSensorBacklightCompensation](#)

Accessible

Camera.AutoFeatures.Sensor.BacklightCompensation

Syntax

```
uc480.AutoFeaturesSensorBacklightCompensation.GetSupported(out bool supported)
```

Description

Returns if backlight compensation is supported.

Parameter

supported	1 = Backlight compensation is supported. 0 = Backlight compensation is not supported.
-----------	--

Class

[uc480.AutoFeaturesSensorBacklightCompensation](#)

Accessible

Camera.AutoFeatures.Sensor.BacklightCompensation

Syntax

```
uc480.AutoFeaturesSensorBacklightCompensation.SetEnable(bool enable)
```

Description

Enables/disables backlight compensation for automatic brightness control.

Parameter

enable	1 = Enables backlight compensation 0 = Disables backlight compensation
--------	---

The `Contrast` class provides methods for controlling the auto contrast correction and the use of the face detection as field of view for automatic brightness control.



This class is currently only supported by the USB uc480 XS camera family.

The following classes exist:

- [Correction](#)
- [FaceDetectionAoi](#)

The `Correction` class provides methods for controlling the auto contrast correction.



This class is currently only supported by the USB uc480 XS camera family.

Methods

Method	Description
Get	Returns the value for auto contrast correction.
GetDefault	Returns the default value for auto contrast correction.
GetRange	Returns the permissible range for auto contrast correction.
GetSupported	Returns if auto contrast correction is supported.
Set	Sets the value for auto contrast correction.

Class

[uc480.AutoFeaturesSensorContrastCorrection](#)

Accessible

Camera.AutoFeatures.Sensor.ContrastCorrection

Syntax

```
uc480.AutoFeaturesSensorContrastCorrection.Get(out double correction)
```

Description

Returns the value for auto contrast correction.

Parameter

- `correction`: Returns the current setting.

Class

[uc480.AutoFeaturesSensorContrastCorrection](#)

Accessible

Camera.AutoFeatures.Sensor.ContrastCorrection

Syntax

```
uc480.AutoFeaturesSensorContrastCorrection.GetDefault(out double correction)
```

Description

Returns the default value for auto contrast correction.

Parameter

- `correction`: Returns the default value.

Class

[uc480.AutoFeaturesSensorContrastCorrection](#)

Accessible

Camera.AutoFeatures.Sensor.ContrastCorrection

Syntax

```
uc480.AutoFeaturesSensorContrastCorrection.GetRange(out uc480.Types.Range<double> range)
uc480.AutoFeaturesSensorContrastCorrection.GetRange(out double min, out double max, out double inc)
```

Description

Returns the permissible range for auto contrast correction.

Parameter

range	Minimum: Returns the minimum permitted value Maximum: Returns the maximum permitted value Increment: Returns the increment
min	Returns the minimum permitted value.
max	Returns the maximum permitted value.
inc	Returns the increment.

Class

[uc480.AutoFeaturesSensorContrastCorrection](#)

Accessible

Camera.AutoFeatures.Sensor.ContrastCorrection

Syntax

```
uc480.AutoFeaturesSensorContrastCorrection.GetSupported(out bool supported)
```

Description

Returns if auto contrast correction is supported.

Parameter

supported	0 = Not supported 1 = Supported
-----------	------------------------------------

Class

[uc480.AutoFeaturesSensorContrastCorrection](#)

Accessible

Camera.AutoFeatures.Sensor.Contrast.Corrector

Syntax

`uc480.AutoFeaturesSensorContrastCorrection.Set(double correction)`

Description

Sets the value for auto contrast correction.

Parameter

- `correction`: Value to be set.

The `FaceDetectionAoi` class provides methods for using the face detection as field of view for automatic brightness control.



This class is currently only supported by the UI-1008XS models.

Methods

Method	Description
GetEnable	Returns the status of using face detection as field of view for automatic brightness control.
GetSupported	Returns if using of face detection as field of view for automatic brightness control is supported.
SetEnable	Enables/disables the use of face detection as field of view for automatic brightness control.

Class

[uc480.AutoFeaturesSensorContrastFaceDetectionAoi](#)

Accessible

Camera.AutoFeatures.Sensor.Contrast.FaceDetectionAoi

Syntax

`uc480.AutoFeaturesSensorContrastFaceDetectionAoi.GetEnable(out bool enable)`

Description

Returns the status of using face detection as field of view for automatic brightness control.

Parameter

<code>enable</code>	1 = Using face detection is enabled. 0 = Using face detection is disabled.
---------------------	---

Class

[uc480.AutoFeaturesSensorContrastFaceDetectionAoi](#)

Accessible

Camera.AutoFeatures.Sensor.Contrast.FaceDetectionAoi

Syntax

```
uc480.AutoFeaturesSensorContrastFaceDetectionAoi.GetSupported(out bool supported)
```

Description

Returns if using of face detection as field of view for automatic brightness control is supported.

Parameter

supported	0 = Not supported
	1 = Supported

Class

[uc480.AutoFeaturesSensorContrastFaceDetectionAoi](#)

Accessible

Camera.AutoFeatures.Sensor.Contrast.FaceDetectionAoi

Syntax

```
uc480.AutoFeaturesSensorContrastFaceDetection.SetEnable(bool enable)
```

Description

Enables/disables the use of face detection as field of view for automatic brightness control. If face detection is disabled, this setting will not become effective until face detection is enabled. The auto exposure shutter and auto gain control functions have to be enabled.

Parameter

enable	1 = Enables the control
	0 = Disables the control

The `Framerate` class provides methods for enabling/disabling the auto frame rate. The auto frame rate is only available when the auto shutter control is on. Auto frame rate and auto gain control cannot be used simultaneously.

Methods

Method	Description
GetEnable	Returns the sensor's current auto frame rate setting. (Not all sensors support this feature.)
GetSupported	Returns if setting the auto frame rate is supported.
SetEnable	Enables/disables the sensor's internal auto frame rate function. (Not all sensors support this feature.)

Class

[uc480.AutoFeaturesSensorFramerate](#)

Accessible

Camera.AutoFeatures.Sensor.Framerate

Syntax

`uc480.AutoFeaturesSensorFramerate.GetEnable(out bool enable)`

Description

Returns the sensor's current auto frame rate setting. (Not all sensors support this feature.)

Parameter

enable	1 = Auto frame rate is enabled.
	0 = Auto frame rate is disabled.

Class

[uc480.AutoFeaturesSensorFramerate](#)

Accessible

Camera.AutoFeatures.Sensor.Framerate

Syntax

`uc480.AutoFeaturesSensorFramerate.GetSupported(out bool supported)`

Description

Returns if setting the auto frame rate is supported.

Parameter

supported	0 = Not supported.
	1 = Supported.

Class

[uc480.AutoFeaturesSensorFramerate](#)

Accessible

Camera.AutoFeatures.Sensor.Framerate

Syntax

`uc480.AutoFeaturesSensorFramerate.SetEnable(bool enable)`

Description

Enables/disables the sensor's internal auto frame rate function. (Not all sensors support this feature.)

Parameter

enable	1 = Enables auto frame rate control
	0 = Disables auto frame rate control

The `Gain` class provides methods for controlling auto gain and the photometry mode for auto gain.

Methods

Method	Description
GetDefaultPhotom	Returns the default photometry mode for auto gain control.
GetEnable	Returns the current auto gain setting or white level adjustment of the sensor. (Not all sensors support this feature.)
GetPhotom	Returns the photometry mode for auto gain control.
GetSupported	Returns if auto gain control is supported.
SetEnable	Enables/disables the internal auto gain control function or, in case of HDR sensors, the white level adjustment of the sensor. (Not all sensors support this feature.)
SetPhotom	Sets the photometry mode for auto gain control.

Class

[uc480.AutoFeaturesSensorGain](#)

Accessible

Camera.AutoFeatures.Sensor.Gain

Syntax

```
uc480.AutoFeaturesSensorGain.GetDefaultPhotom(out uc480.Defines.Whitebalance.GainPhotomMode mode)
```

Description

Returns the default photometry mode for auto gain control.

Parameter

- `mode`: Returns the default mode (see [SetPhotom\(\)](#)).

Class

[uc480.AutoFeaturesSensorGain](#)

Accessible

Camera.AutoFeatures.Sensor.Gain

Syntax

```
uc480.AutoFeaturesSensorGain.GetEnable(out bool enable)
```

Description

Returns the current auto gain setting or white level adjustment of the sensor. (Not all sensors support this feature.)

Parameter

enable	1 = Auto gain is enabled. 0 = Auto gain is disabled.
--------	---

Class[uc480.AutoFeaturesSensorGain](#)**Accessible**

Camera.AutoFeatures.Sensor.Gain

Syntax

```
uc480.AutoFeaturesSensorGain.GetPhotom(out uc480.Defines.Whitebalance.GainPhotomMode mode)
```

Description

Returns the photometry mode for auto gain control.

Parameter

- mode: Returns the current setting (see [SetPhotom\(\)](#)).

Class[uc480.AutoFeaturesSensorGain](#)**Accessible**

Camera.AutoFeatures.Sensor.Gain

Syntax

```
uc480.AutoFeaturesSensorGain.GetSupported(out bool supported)
```

Description

Returns if auto gain control is supported.

Parameter

supported	0 = Not supported 1 = Supported
-----------	------------------------------------

Class[uc480.AutoFeaturesSensorGain](#)**Accessible**

Camera.AutoFeatures.Sensor.Gain

Syntax

```
uc480.AutoFeaturesSensorGain.SetEnable(bool enable)
```

Description

Enables/disables the internal auto gain control function or, in case of HDR sensors, the white level adjustment of the sensor. (Not all sensors support this feature.)

Parameter

enable	1 = Enables auto gain control 0 = Disables auto gain control
--------	---

Class

[uc480.AutoFeaturesSensorGain](#)

Accessible

Camera.AutoFeatures.Sensor.Gain

Syntax

```
uc480.AutoFeaturesSensorGain.SetPhotom(uc480.Defines.Whitebalance.GainPhotomMode mode)
```

Description

Sets the photometry mode for auto gain control.

Parameter

mode	Defines which fields of view are used for auto gain control: <ul style="list-style-type: none"> • uc480.Defines.Whitebalance.GainPhotomMode.None • uc480.Defines.Whitebalance.GainPhotomMode.CenterWeighted • uc480.Defines.Whitebalance.GainPhotomMode.CenterSpot • uc480.Defines.Whitebalance.GainPhotomMode.Portrait • uc480.Defines.Whitebalance.GainPhotomMode.Landscape
------	---

The `GainShutter` class provides methods for controlling the auto exposure time/auto gain feature.



This class is currently only supported by the XS model.

For this model the automatic control of the exposure time and gain can only be enabled/disabled together.

Methods

Method	Description
GetEnable	Returns the state of the auto exposure time/ auto gain.
GetSupported	Returns if setting the auto exposure time/ auto gain is supported.
SetEnable	Sets the state of the auto exposure time/ auto gain.

Class

[uc480.AutoFeaturesSensorGainShutter](#)

Accessible

Camera.AutoFeatures.Sensor.GainShutter

Syntax

```
uc480.AutoFeaturesSensorGainShutter.GetEnable(out bool enable)
```

Description

Returns the state of the auto exposure time/auto gain.

Parameter

enable	1 = Auto exposure time/auto gain is enabled. 0 = Autoexposure time/auto gain is disabled.
--------	--

Class

[uc480.AutoFeaturesSensorGainShutter](#)

Accessible

Camera.AutoFeatures.Sensor.GainShutter

Syntax

```
uc480.AutoFeaturesSensorGainShutter.GetSupported(out bool supported)
```

Description

Returns if setting the auto exposure time/auto gain is supported.

Parameter

supported	0 = Not supported. 1 = Supported.
-----------	--------------------------------------

Class

[uc480.AutoFeaturesSensorGainShutter](#)

Accessible

Camera.AutoFeatures.Sensor.GainShutter

Syntax

```
uc480.AutoFeaturesSensorGainShutter.SetEnable(bool enable)
```

Description

Sets the state of the auto exposure time/auto gain.

Parameter

enable	1 = Enable auto exposure time/auto gain 0 = Disable time/auto gain
--------	---

The `Shutter` class provides methods for controlling the default photometry mode for auto exposure shutter (not all sensors support this feature).

Methods

Method	Description
GetDefaultPhotom	Returns the default photometry mode for auto exposure shutter.
GetEnable	Returns the sensor's current auto exposure shutter setting. (Not all sensors support this feature.)
GetPhotom	Returns the photometry mode for auto exposure shutter.
GetSupported	Returns is the setting of the default photometry mode for auto exposure shutter is supported.
SetEnable	Enables/disables the sensor's internal auto exposure shutter function. (Not all sensors support this feature.)
SetPhotom	Sets the photometry mode for auto exposure shutter.

Class

[uc480.AutoFeaturesSensorShutter](#)

Accessible

Camera.AutoFeatures.Sensor.Shutter

Syntax

`uc480.AutoFeaturesSensorShutter.GetDefaultPhotom(out uc480.Defines.Whitebalance.ShutterPhotomMode mode)`

Description

Returns the default photometry mode for auto exposure shutter.

Parameter

- `mode`: Returns the default mode (see [SetPhotom\(\)](#)).

Class

[uc480.AutoFeaturesSensorShutter](#)

Accessible

Camera.AutoFeatures.Sensor.Shutter

Syntax

`uc480.AutoFeaturesSensorShutter.GetEnable(out bool enable)`

Description

Returns the sensor's current auto exposure shutter setting. (Not all sensors support this feature.)

Parameter

enable	1 = Auto shutter is enabled.
	0 = Auto shutter is disabled.

Class[uc480.AutoFeaturesSensorShutter](#)**Accessible**

Camera.AutoFeatures.Sensor.Shutter

Syntax

```
uc480.AutoFeaturesSensorShutter.GetPhotom(out  
uc480.Defines.Whitebalance.ShutterPhotomMode mode)
```

Description

Returns the photometry mode for auto exposure shutter.

Parameter

- mode: Returns the current setting (see [SetPhotom\(\)](#)).

Class[uc480.AutoFeaturesSensorShutter](#)**Accessible**

Camera.AutoFeatures.Sensor.Shutter

Syntax

```
uc480.AutoFeaturesSensorShutter.GetSupported(out bool supported)
```

Description

Returns is the setting of the default photometry mode for auto exposure shutter is supported.

Parameter

supported	0 = Not supported.
	1 = Supported.

Class[uc480.AutoFeaturesSensorShutter](#)**Accessible**

Camera.AutoFeatures.Sensor.Shutter

Syntax

```
uc480.AutoFeaturesSensorShutter.SetEnable(bool enable)
```

Description

Enables/disables the sensor's internal auto exposure shutter function. (Not all sensors support this feature.)

Parameter

enable	1 = Enables auto shutter control 0 = Disables auto shutter control
--------	---

Class[uc480.AutoFeaturesSensorShutter](#)**Accessible**

Camera.AutoFeatures.Sensor.Shutter

Syntax

uc480.AutoFeaturesSensorShutter.SetPhotom(uc480.Defines.Whitebalance.ShutterPhotomMode mode)

Description

Sets the photometry mode for auto exposure shutter.

Parameter

mode	Defines which fields of view are used for auto exposure shutter: <ul style="list-style-type: none"> • uc480.Defines.Whitebalance.ShutterPhotomMode.None • uc480.Defines.Whitebalance.ShutterPhotomMode.CenterWeighted • uc480.Defines.Whitebalance.ShutterPhotomMode.CenterSpot • uc480.Defines.Whitebalance.ShutterPhotomMode.Portrait • uc480.Defines.Whitebalance.ShutterPhotomMode.Landscape
------	--

The `Whitebalance` class provides methods for controlling the sensor's auto white balance feature (not all sensors support this feature).

Methods

Method	Description
GetEnable	Returns the sensor's current auto white balance setting. (Not all sensors support this feature.)
GetSupported	Returns if the sensor supports the auto white balance.
SetEnable	Enables/disables the sensor's internal auto white balance feature. (Not all sensors support this feature.)

Class[uc480.AutoFeaturesSensorWhitebalance](#)**Accessible**

Camera.AutoFeatures.Sensor.Whitebalance

Syntax

uc480.AutoFeaturesSensorWhitebalance.GetEnable(out bool enable)

Description

Returns the sensor's current auto white balance setting. (Not all sensors support this feature.)

Parameter

enable	1 = Auto white balance is enabled. 0 = Auto white balance is disabled.
--------	---

Class

[uc480.AutoFeaturesSensorWhitebalance](#)

Accessible

Camera.AutoFeatures.Sensor.Whitebalance

Syntax

uc480.AutoFeaturesSensorWhitebalance.GetSupported(out bool supported)

Description

Returns if the sensor supports the auto white balance.

Parameter

supported	1 = Auto white balance is supported. 0 = Auto white balance is not supported.
-----------	--

Class

[uc480.AutoFeaturesSensorWhitebalance](#)

Accessible

Camera.AutoFeatures.Sensor.Whitebalance

Syntax

uc480.AutoFeaturesSensorWhitebalance.SetEnable(bool enable)

Description

Enables/disables the sensor's internal auto white balance function. (Not all sensors support this feature.)

Parameter

enable	1 = Enables auto white balance control 0 = Disables auto white balance control
--------	---

3.1.1.2.2 Software

The `Software` class provides methods for controlling the automatic image control features.

- 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 exposure 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 control cannot be used simultaneously.
- The **auto gain** function can only be used for cameras with master gain control. Auto white balance is only available for cameras with hardware RGB gain control.



Notes on models with HDR sensor

In the UI-112x/UI-512x models with HDR sensor, some auto features have a different functionality or are not available:

- The HDR sensor has no gain. The gain methods control the white level adjustment.
- The shutter, frame rate and white balance methods are not available for this sensor.



Notes on the sensor's internal control functionality

Automatic control by the [sensor](#) and the software is not possible simultaneously. To use the sensor's internal control functionality, disable [software](#) control, and vice versa.



Note on automatic controls when using very high frame rates

Using very high frame rates can cause that too many control commands are sent to the camera. When using frame rates higher than 100 fps you should increase the value for the skipped frames via [Set\(\)](#). Thus, less image will be used for the automatic controls which takes load off the camera.

The following classes exist:

- [Framerate](#)
- [FrameSkip](#)
- [Gain](#)
- [Hysteresis](#)
- [Reference](#)
- [Shutter](#)
- [Speed](#)
- [Whitebalanc](#)

The [Framerate](#) class provides methods for controlling the auto frame feature.

Methods

Method	Description
GetEnable	Returns the current auto frame rate setting.
GetSupported	Returns if setting the auto frame rate is supported.
SetEnable	Enables/disables the auto frame rate function.

Class

[uc480.AutoFeaturesSoftwareFramerate](#)

Accessible

Camera.AutoFeatures.Software.Framerate

Syntax

`uc480.AutoFeaturesSoftwareFramerate.GetEnable(out bool enable)`

Description

Returns the current auto frame rate setting.

Parameter

enable	1 = Auto frame rate control is enabled
	0 = Auto frame rate control is disabled.

Class

[uc480.AutoFeaturesSoftwareFramerate](#)

Accessible

Camera.AutoFeatures.Software.Framerate

Syntax

`uc480.AutoFeaturesSoftwareFramerate.GetSupported(out bool supported)`

Description

Returns if setting the auto frame rate is supported.

Parameter

supported	0 = Not supported.
	1 = Supported.

Class

[uc480.AutoFeaturesSoftwareFramerate](#)

Accessible

Camera.AutoFeatures.Software.Framerate

Syntax

`uc480.AutoFeaturesSoftwareFramerate.SetEnable(bool enable)`

Description

Enables/disables the auto frame rate function.

Parameter

enable	1 = Enables auto frame rate control
	0 = Disables auto frame rate control

The `FrameSkip` class provides methods for controlling the number of frames to be skipped during automatic control.

Methods

Method	Description
Get	Returns the number of frames to be skipped during automatic control.
GetRange	Returns the permissible range for the number of frames to be skipped.
Set	Sets the number of frames to be skipped during automatic control.

Class

[uc480.AutoFeaturesSoftwareFrameSkip](#)

Accessible

Camera.AutoFeatures.Software.FrameSkip

Syntax

```
uc480.AutoFeaturesSoftwareFrameSkip.Get(out uint frameSkip)
```

Description

Returns the number of frames to be skipped during automatic control.

Parameter

- `frameSkip`: Returns the current setting.

Class

[uc480.AutoFeaturesSoftwareFrameSkip](#)

Accessible

Camera.AutoFeatures.Software.FrameSkip

Syntax

```
uc480.AutoFeaturesSoftwareFrameSkip.GetRange(out uc480.Types.Range<uint> range)
uc480.AutoFeaturesSoftwareFrameSkip.GetRange(out uint min, out uint max, out uint inc)
```

Description

Returns the permissible range for the number of frames to be skipped.

Parameter

<code>range</code>	<code>Minimum</code> : Returns the minimum value
	<code>Maximum</code> : Returns the maximum value
	<code>Increment</code> : Returns the increment
<code>min</code>	Returns the minimum number of frames to be skipped.
<code>max</code>	Returns the maximum number of frames to be skipped.
<code>inc</code>	Returns the increment.

Class

[uc480.AutoFeaturesSoftwareFrameSkip](#)

Accessible

Camera.AutoFeatures.Software.FrameSkip

Syntax

`uc480.AutoFeaturesSoftwareFrameSkip.Set(uint frameSkip)`

Description

Sets the number of frames to be skipped during automatic control.

Parameter

- `frameSkip`: Defines the number of frames to be skipped during automatic control (default: 4)

The `Gain` class provides methods for controlling the auto gain or, in case of HDR sensors, the white level adjustment.

Methods

Method	Description
GetEnable	Returns the current auto gain setting or white level adjustment.
GetMax	Returns the upper limit for auto gain control.
GetSupported	Returns if setting the auto gain is supported.
SetEnable	Enables/disables the auto gain control feature or, in case of HDR sensors, the white level adjustment.
SetMax	Sets the upper limit for auto gain control.

Class

[uc480.AutoFeaturesSoftwareGain](#)

Accessible

Camera.AutoFeatures.Software.Gain

Syntax

`uc480.AutoFeaturesSoftwareGain.GetEnable(out bool enable)`

Description

Returns the current auto gain setting or white level adjustment.

Parameter

- `enable`: Returns the current setting.

Class

[uc480.AutoFeaturesSoftwareGain](#)

Accessible

Camera.AutoFeatures.Software.Gain

Syntax

```
uc480.AutoFeaturesSoftwareGain.GetMax(out int max)
```

Description

Returns the upper limit for auto gain control.

Parameter

- `max`: Returns the current setting.

Class

[uc480.AutoFeaturesSoftwareGain](#)

Accessible

Camera.AutoFeatures.Software.Gain

Syntax

```
uc480.AutoFeaturesSoftwareGain.GetSupported(out bool supported)
```

Description

Returns if setting the auto gain is supported.

Parameter

supported	0 = Not supported. 1 = Supported.
-----------	--------------------------------------

Class

[uc480.AutoFeaturesSoftwareGain](#)

Accessible

Camera.AutoFeatures.Software.Gain

Syntax

```
uc480.AutoFeaturesSoftwareGain.SetEnable(bool enable)
```

Description

Enables/disables the auto gain control function or, in case of HDR sensors, the white level adjustment.

Parameter

enable	1 = Enables auto gain control 0 = Disables auto gain control
--------	---

Class

[uc480.AutoFeaturesSoftwareGain](#)

Accessible

Camera.AutoFeatures.Software.Gain

Syntax

```
uc480.AutoFeaturesSoftwareGain.SetMax(int max)
```

Description

Sets the upper limit for auto gain control.

Parameter

- `max`: Valid value for gain (0...100)

The `Hysteresis` class provides methods for controlling the hysteresis value for auto exposure and auto gain control.

Methods

Method	Description
<u>Get</u>	Returns the hysteresis value for auto exposure shutter and auto gain control.
<u>GetRange</u>	Returns the permissible range for the hysteresis value.
<u>Set</u>	Sets the hysteresis value for auto exposure shutter and auto gain control.

Class

[uc480.AutoFeaturesSoftwareHysteresis](#)

Accessible

`Camera.AutoFeatures.Software.Hysteresis`

Syntax

`uc480.AutoFeaturesSoftwareHysteresis.Get(out uint hysteresis)`

Description

Returns the hysteresis value for auto exposure shutter and auto gain control.

Parameter

- `hysteresis`: Returns the current setting.

Class

[uc480.AutoFeaturesSoftwareHysteresis](#)

Accessible

`Camera.AutoFeatures.Software.Hysteresis`

Syntax

`uc480.AutoFeaturesSoftwareHysteresis.GetRange(out uc480.Types.Range<uint> range)`
`uc480.AutoFeaturesSoftwareHysteresis.GetRange(out uint min, out uint max, out uint inc)`

Description

Returns the permissible range for the hysteresis value.

Parameter

range	Minimum: Returns the minimum value Maximum: Returns the maximum value Increment: Returns the increment
min	Returns the minimum hysteresis value.
max	Returns the maximum hysteresis value.
inc	Returns the increment.

Class

[uc480.AutoFeaturesSoftwareHysteresis](#)

Accessible

Camera.AutoFeatures.Software.Hysteresis

Syntax

`uc480.AutoFeaturesSoftwareHysteresis.Set(uint hysteresis)`

Description

Sets the hysteresis value for auto exposure shutter and auto gain control.

Parameter

- `hysteresis`: defines the hysteresis value (default: 2)

The `Reference` class provides methods for controlling the auto gain/auto exposure shutter.

Methods

Method	Description
Get	Returns the setpoint for auto gain control/ auto exposure shutter.
GetDefault	Returns the default setpoint for auto gain control and auto exposure shutter.
GetRange	Returns the range for auto gain control and cuto exposure shutter.
Set	Sets the setpoint for auto gain control/auto exposure shutter.

Class

[uc480.AutoFeaturesSoftwareReference](#)

Accessible

Camera.AutoFeatures.Software.Reference

Syntax

`uc480.AutoFeaturesSoftwareReference.Get(out uint reference)`

Description

Returns the setpoint for auto gain control/auto exposure shutter.

Parameter

- `reference`: Returns the current setting.

Class[uc480.AutoFeaturesSoftwareReference](#)**Accessible**

Camera.AutoFeatures.Software.Reference

Syntax`uc480.AutoFeaturesSoftwareReference.GetDefault(out int speed)`**Description**

Returns the default setpoint for auto gain control and auto exposure shutter.

Parameter

- `speed`: Returns the default setting.

Class[uc480.AutoFeaturesSoftwareReference](#)**Accessible**

Camera.AutoFeatures.Software.Reference

Syntax`uc480.AutoFeaturesSoftwareReference.GetRange(out uc480.Types.Range<uint> range)
uc480.AutoFeaturesSoftwareReference.GetRange(out uint min, out uint max, out uint inc)`**Description**

Returns the range for auto gain control and cuto exposure shutter.

Parameter

range	Minimum: Returns the minimum value Maximum: Returns the maximum value Increment: Returns the increment
min	Returns the minimum value.
max	Returns the maximum value.
inc	Returns the increment.

Class[uc480.AutoFeaturesSoftwareReference](#)**Accessible**

Camera.AutoFeatures.Software.Reference

Syntax`uc480.AutoFeaturesSoftwareReference.Set(uint reference)`**Description**

Sets the setpoint for auto gain control/auto exposure shutter.



When using the sensor's internal control functionality, you can only use values in a range between [44...235]. The increment in this range is 4. Smaller values are automatically set to 44, larger values to 235.

Parameter

reference	Defines the setpoint (average brightness of the image); the following rule applies independently of the image bit depth: <ul style="list-style-type: none">• 0 = black• 128 = 50% gray (default)• 255 = white
-----------	---

The `Shutter` class provides methods for controlling the auto exposure shutter feature.

Methods

Method	Description
GetEnable	Returns the current auto exposure shutter setting.
GetMax	Returns the upper limit for auto exposure shutter.
GetSupported	Returns if setting the auto exposure shutter is supported.
SetEnable	Enables/disables the auto exposure shutter control.
SetMax	Sets the upper limit for auto exposure shutter.

Class

[uc480.AutoFeaturesSoftwareShutter](#)

Accessible

Camera.AutoFeatures.Software.Shutter

Syntax

```
uc480.AutoFeaturesSoftwareShutter.GetEnable(out uc480.Defines.ActivateMode activateMode)
uc480.AutoFeaturesSoftwareShutter.GetEnable(out bool enable)
```

Description

Returns the current auto exposure shutter setting.

Parameter

- `activateMode/enable`: Returns the current setting.

Class

[uc480.AutoFeaturesSoftwareShutter](#)

Accessible

Camera.AutoFeatures.Software.Shutter

Syntax

```
uc480.AutoFeaturesSoftwareShutter.GetMax(out double max)
```

Description

Returns the upper limit for auto exposure shutter.

Parameter

- `max`: Returns the current setting.

Class

[uc480.AutoFeaturesSoftwareShutter](#)

Accessible

Camera.AutoFeatures.Software.Shutter

Syntax

```
uc480.AutoFeaturesSoftwareShutter.GetSupported(out bool supported)
```

Description

Returns if setting the auto exposure shutter is supported.

Parameter

<code>supported</code>	<code>0</code> = Not supported. <code>1</code> = Supported.
------------------------	--

Class

[uc480.AutoFeaturesSoftwareShutter](#)

Accessible

Camera.AutoFeatures.Software.Shutter

Syntax

```
uc480.AutoFeaturesSoftwareShutter.SetEnable(uc480.Defines.ActivateMode activateMode)  
uc480.AutoFeaturesSoftwareShutter.SetEnable(bool enable)
```

Description

Enables/disables the auto exposure shutter control.

Parameter

<code>activateMode/enable</code>	<code>uc480.Defines.ActivateMode.Enable/1:</code> Enables auto shutter control
	<code>uc480.Defines.ActivateMode.Disable/0:</code> Disables auto shutter control

Class

[uc480.AutoFeaturesSoftwareShutter](#)

Accessible

Camera.AutoFeatures.Software.Shutter

Syntax

```
uc480.AutoFeaturesSoftwareShutter.SetMax(double max)
```

Description

Sets the upper limit for auto exposure shutter.

Parameter

- `max`: Valid exposure value (0 sets the value continuously to max. exposure)

The `Speed` class provides methods for controlling the speed of the auto functions.

Methods

Method	Description
<u>Get</u>	Returns the speed value for the auto function.
<u>GetDefault</u>	Returns the default value for auto speed.
<u>GetRange</u>	Returns the range for auto speed.
<u>Set</u>	Sets the speed value for the auto function.

Class

[uc480.AutoFeaturesSoftwareSpeed](#)

Accessible

Camera.AutoFeatures.Software.Speed

Syntax

```
uc480.AutoFeaturesSoftwareSpeed.Get(out uint speed)
```

Description

Returns the speed value for the auto function.

Parameter

- `speed`: Returns the current setting.

Class

[uc480.AutoFeaturesSoftwareSpeed](#)

Accessible

Camera.AutoFeatures.Software.Speed

Syntax

```
uc480.AutoFeaturesSoftwareSpeed.GetDefault(out int speed)
```

Description

Returns the default value for auto speed.

Parameter

- `speed`: Returns the default value.

Class

[uc480.AutoFeaturesSoftwareSpeed](#)

Accessible

Camera.AutoFeatures.Software.Speed

Syntax

```
uc480.AutoFeaturesSoftwareSpeed.GetRange(out uc480.Types.Range<uint> range)
uc480.AutoFeaturesSoftwareSpeed.GetRange(out uint min, out uint max, out uint inc)
```

Description

Returns the range for auto speed.

Parameter

range	Minimum: Returns the minimum value Maximum: Returns the maximum value Increment: Returns the increment
min	Returns the minimum value.
max	Returns the maximum value.
inc	Returns the increment.

Class

[uc480.AutoFeaturesSoftwareSpeed](#)

Accessible

Camera.AutoFeatures.Software.Speed

Syntax

```
uc480.AutoFeaturesSoftwareSpeed.Set(uint speed)
```

Description

Sets the speed value for the auto function.

Parameter

- speed: Defines the control speed (0...100)

Using the `Whitebalance` class, you can control the auto white balance. With this class, you can require all supported types for white balance. In addition to the older white balance with the Gray-World algorithm, there is also a color temperature control according to Kelvin. Also the supported color spaces are queried and set.

The following classes and methods exist:

- [Frameskip](#)
- [Gain](#)
- [Hysteresis](#)
- [Offset](#)
- [Speed](#)

Methods

Method	Description
GetColorModel	Returns the current color space for the auto white balance.
GetEnable	Returns the current auto white balance setting.
GetReferenceRange	Returns the range for white balance for auto gain control/auto exposure shutter.
GetSupported	Returns if setting the auto white balance is supported.
GetSupportedColorModel	Returns the supported color spaces for auto white balance.
GetSupportedType	Returns the supported types for auto white balance.
GetType	Returns the current set type for auto white balance.
SetColorModel	Sets the color space for auto white balance.
SetEnable	Enables/disables auto white balance.
SetType	Sets the type for auto white balance.

The `FrameSkip` class provides methods for controlling the frames to be skipped during automatic control.

Methods

Method	Description
Get	Returns the number of frames to be skipped during automatic control.
GetRange	Returns the permissible range for the number of frames to be skipped.
Set	Sets the number of frames to be skipped during automatic control.

Class

[uc480.AutoFeaturesSoftwareWhitebalanceFrameSkip](#)

Accessible

`Camera.AutoFeatures.Software.Whitebalance.FrameSkip`

Syntax

```
uc480.AutoFeaturesSoftwareWhitebalanceFrameSkip.Get(out uint frameSkip)
```

Description

Returns the number of frames to be skipped during automatic control.

Parameter

- `frameSkip`: Returns the current setting.

Class

[uc480.AutoFeaturesSoftwareWhitebalanceFrameSkip](#)

Accessible

Camera.AutoFeatures.Software.Whitebalance.FrameSkip

Syntax

```
uc480.AutoFeaturesSoftwareWhitebalanceFrameSkip.GetRange(out uc480.Types.Range<uint> range)
uc480.AutoFeaturesSoftwareWhitebalanceFrameSkip.GetRange(out uint min, out uint max, out uint inc)
```

Description

Returns the permissible range for the number of frames to be skipped.

Parameter

range	Minimum: Returns the minimum number of frames to be skipped. Maximum: Returns the maximum number of frames to be skipped. Increment: Returns the increment.
min	Returns the minimum number of frames to be skipped.
max	Returns the maximum number of frames to be skipped.
inc	Returns the increment.

Class

[uc480.AutoFeaturesSoftwareWhitebalanceFrameSkip](#)

Accessible

Camera.AutoFeatures.Software.Whitebalance.FrameSkip

Syntax

```
uc480.AutoFeaturesSoftwareWhitebalanceFrameSkip.Set(uint frameSkip)
```

Description

Sets the number of frames to be skipped during automatic control.

Parameter

- `frameSkip`: Defines the number (default: 4)

The `Gain` class provides methods for controlling the color gain limits for auto white balance.

Methods

Method	Description
GetRange	Returns the color gain limits for auto white balance.
SetRange	Sets the color gain limits for auto white balance.

Class

[uc480.AutoFeaturesSoftwareWhitebalanceGain](#)

Accessible

Camera.AutoFeatures.Software.Whitebalance.Gain

Syntax

```
uc480.AutoFeaturesSoftwareWhitebalanceGain.GetRange(out uc480.Types.Range<int> range)
uc480.AutoFeaturesSoftwareWhitebalanceGain.GetRange(out int min, out int max, out int inc)
```

Description

Returns the color gain limits for auto white balance.

Parameter

range	Minimum: Returns the minimum value Maximum: Returns the maximum value Increment: Returns the increment
min	Returns the minimum value.
max	Returns the maximum value.
inc	Returns the increment.

Class

[uc480.AutoFeaturesSoftwareWhitebalanceGain](#)

Accessible

Camera.AutoFeatures.Software.Whitebalance.Gain

Syntax

```
uc480.AutoFeaturesSoftwareWhitebalanceGain.SetRange(uc480.Types.Range<int> range)
uc480.AutoFeaturesSoftwareWhitebalanceGain.SetRange(int min, int max)
```

Description

Returns the color gain limits for auto white balance.

Parameter

range	Minimum: Sets the minimum value Maximum: Sets the maximum value Increment: Sets the increment
min	Sets the minimum value.
max	Sets the maximum value.

The `Hysteresis` class provides methods for controlling hysteresis for auto white balance.

Methods

Method	Description
<code>Get</code>	Returns the hysteresis value for auto white balance.
<code>GetRange</code>	Returns the permissible range for the hysteresis value.
<code>Set</code>	Sets the hysteresis value for auto white balance.

Class

[uc480.AutoFeaturesSoftwareWhitebalanceHysteresis](#)

Accessible

Camera.AutoFeatures.Software.Whitebalance.Hysteresis

Syntax

```
uc480.AutoFeaturesSoftwareWhitebalanceHysteresis.Get(out uint hysteresis)
```

Description

Returns the hysteresis value for auto white balance.

Parameter

- `hysteresis`: Returns the current setting.

Class

[uc480.AutoFeaturesSoftwareWhitebalanceHysteresis](#)

Accessible

Camera.AutoFeatures.Software.Whitebalance.Hysteresis

Syntax

```
uc480.AutoFeaturesSoftwareWhitebalanceHysteresis.GetRange(out uc480.Types.Range<uint> range)
uc480.AutoFeaturesSoftwareWhitebalanceHysteresis.GetRange(out uint min, out uint max, out uint inc)
```

Description

Returns the permissible range for the hysteresis value.

Parameter

<code>range</code>	Minimum: returns the minimum value
	Maximum: returns the maximum value
	Increment: returns the increment
<code>min</code>	Returns the minimum permitted hysteresis value.
<code>max</code>	Returns the maximum permitted hysteresis value.
<code>inc</code>	Returns the increment.

Class

[uc480.AutoFeaturesSoftwareWhitebalanceHysteresis](#)

Accessible

Camera.AutoFeatures.Software.Whitebalance.Hysteresis

Syntax

`uc480.AutoFeaturesSoftwareWhitebalanceHysteresis.Set(uint hysteresis)`

Description

Sets the hysteresis value for auto white balance.

Parameter

- `hysteresis`: Defines the hysteresis value (default: 2)

The `Offset` class provides methods for controlling the white balance offset values for the red and blue channels.

Methods

Method	Description
Get	Returns the offset values for the red and blue channels.
GetDefault	Returns the default offset value.
GetRange	Returns the range for offset.
Set	Sets the offset values for the red and blue channels.

Class

[uc480.AutoFeaturesSoftwareWhitebalanceOffset](#)

Accessible

Camera.AutoFeatures.Software.Whitebalance.Offset

Syntax

`uc480.AutoFeaturesSoftwareWhitebalanceOffset.Get(out int redOffset, out int blueOffset)`

Description

Returns the offset values for the red and blue channels.

Parameter

<code>redOffset</code>	Returns the red level offset (-50...50)
<code>blueOffset</code>	Returns the blue level offset (-50...50)

Class

[uc480.AutoFeaturesSoftwareWhitebalanceOffset](#)

Accessible

Camera.AutoFeatures.Software.Whitebalance.Offset

Syntax

`uc480.AutoFeaturesSoftwareWhitebalanceOffset.GetDefault(out int offset)`

Description

Returns the default offset value.

Parameter

- **offset**: Returns the default value.

Class

[uc480.AutoFeaturesSoftwareWhitebalanceOffset](#)

Accessible

Camera.AutoFeatures.Software.Whitebalance.Offset

Syntax

```
uc480.AutoFeaturesSoftwareWhitebalanceOffset.GetRange(out uc480.Types.Range<int> range)
uc480.AutoFeaturesSoftwareWhitebalanceOffset.GetRange(out int min, out int max, out int inc)
```

Description

Returns the range for offset.

Parameter

range	Minimum: returns the minimum value Maximum: returns the maximum value Increment: returns the increment
min	Returns the minimum value.
max	Returns the maximum value.
inc	Returns the increment.

Class

[uc480.AutoFeaturesSoftwareWhitebalanceOffset](#)

Accessible

Camera.AutoFeatures.Software.Whitebalance.Offset

Syntax

```
uc480.AutoFeaturesSoftwareWhitebalanceOffset.Set(int redOffset, int blueOffset)
```

Description

Sets the offset values for the red and blue channels.

Parameter

redOffset	Defines the red level offset (-50...50)
blueOffset	Defines the blue level offset (-50...50)

The `Speed` class provides methods for controlling the speed for auto white balance.

Methods

Method	Description
Get	Returns the speed for auto white balance.
GetDefault	Returns the default value for auto white balance speed.
GetRange	Returns the range for auto white balance speed.
Set	Sets the speed for auto white balance.

Class

[uc480.AutoFeaturesSoftwareWhitebalanceSpeed](#)

Accessible

`Camera.AutoFeatures.Software.Whitebalance.Speed`

Syntax

```
uc480.AutoFeaturesSoftwareWhitebalanceSpeed.Get(out uint speed)
```

Description

Returns the speed for auto white balance.

Parameter

- `speed`: Returns the current setting.

Class

[uc480.AutoFeaturesSoftwareWhitebalanceSpeed](#)

Accessible

`Camera.AutoFeatures.Software.Whitebalance.Speed`

Syntax

```
uc480.AutoFeaturesSoftwareWhitebalanceSpeed.GetDefault(out int speed)
```

Description

Returns the default value for auto white balance speed.

Parameter

- `speed`: Returns the default value.

Class

[uc480.AutoFeaturesSoftwareWhitebalanceSpeed](#)

Accessible

`Camera.AutoFeatures.Software.Whitebalance.Speed`

Syntax

```
uc480.AutoFeaturesSoftwareWhitebalanceSpeed.GetRange(out uc480.Types.Range<uint> range)
uc480.AutoFeaturesSoftwareWhitebalanceSpeed.GetRange(out uint min, out uint max, out uint inc)
```

Description

Returns the range for auto white balance speed.

Parameter

range	Minimum: returns the minimum value Maximum: returns the maximum value Increment: returns the increment
min	Returns the minimum value.
max	Returns the maximum value.
inc	Returns the increment.

Class

[uc480.AutoFeaturesSoftwareWhitebalanceSpeed](#)

Accessible

Camera.AutoFeatures.Software.Whitebalance.Speed

Syntax

`uc480.AutoFeaturesSoftwareWhitebalanceSpeed.Set(uint speed)`

Description

Sets the speed for auto white balance.

Parameter

- `speed`: Defines the control speed (0...100)

Class

[uc480.AutoFeaturesSoftwareWhitebalance](#)

Accessible

Camera.AutoFeatures.Software.Whitebalance

Syntax

`uc480.AutoFeaturesSoftwareWhitebalance.GetColorModel(out uc480.Defines.ColorTemperatureRgbMode colorMode)`

Description

Returns the current color space for auto white balance.

Parameter

- `colorMode`: Returns the color space, see [GetSupportedColorModel\(\)](#).

Class

[uc480.AutoFeaturesSoftwareWhitebalance](#)

Accessible

Camera.AutoFeatures.Software.Whitebalance

Syntax

`uc480.AutoFeaturesSoftwareWhitebalance.GetEnable(out bool bEnable)`
`uc480.AutoFeaturesSoftwareWhitebalance.GetEnable(out uc480.Defines.ActivateMode activateMode)`

Description

Returns the current auto white balance setting.

Parameter

bEnable	1 = Auto white balance is enabled. 0 = Auto white balance is disabled.
activateMode	Returns the supported white balance modes: <ul style="list-style-type: none">• uc480.Defines.ActivateMode.Disable• uc480.Defines.ActivateMode.Enable• uc480.Defines.ActivateMode.Once

Class

[uc480.AutoFeaturesSoftwareWhitebalance](#)

Accessible

Camera.AutoFeatures.Software.Whitebalance

Syntax

```
uc480.AutoFeaturesSoftwareWhitebalance.GetReferenceRange(out int min, out int max, out int inc)
```

Description

Returns the range for white balance for auto gain control/auto exposure shutter.

Parameter

min	Returns the minimum value.
max	Returns the maximum value.
inc	Returns the increment.

Class

[uc480.AutoFeaturesSoftwareWhitebalance](#)

Accessible

Camera.AutoFeatures.Software.Whitebalance

Syntax

```
uc480.AutoFeaturesSoftwareWhitebalance.GetSupported(out bool supported)
```

Description

Returns if setting the auto white balance is supported.

Parameter

supported	0 = Not supported. 1 = Supported.
-----------	--

Class

[uc480.AutoFeaturesSoftwareWhitebalance](#)

Accessible

Camera.AutoFeatures.Software.Whitebalance

Syntax

```
uc480.AutoFeaturesSoftwareWhitebalance.GetSupportedColorModel(out uc480.Defines.ColorTemperatureRgbMode color
```

Description

Returns the supported color spaces for auto white balance.

Parameter

colorModeSupported	Returns the supported color spaces: <ul style="list-style-type: none">• uc480.Defines.ColorTemperatureRgbMode .ADOBE_RGB_D65• uc480.Defines.ColorTemperatureRgbMode .CIE_RGB_E• uc480.Defines.ColorTemperatureRgbMode .ECI_RGB_D50• uc480.Defines.ColorTemperatureRgbMode .SRGB_D50• uc480.Defines.ColorTemperatureRgbMode .SRGB_D65
--------------------	---

Class

[uc480.AutoFeaturesSoftwareWhitebalance](#)

Accessible

Camera.AutoFeatures.Software.Whitebalance

Syntax

```
uc480.AutoFeaturesSoftwareWhitebalance.GetSupportedType(out uc480.Defines.Whitebalance.Type supported)
```

Description

Returns the supported types for auto white balance.

Parameter

supported	Returns the supported type: <ul style="list-style-type: none">• uc480.Defines.Whitebalance.Type.ColorTemperature• uc480.Defines.Whitebalance.Type.GreyWorld
-----------	---

Class

[uc480.AutoFeaturesSoftwareWhitebalance](#)

Accessible

Camera.AutoFeatures.Software.Whitebalance

Syntax

```
uc480.AutoFeaturesSoftwareWhitebalance.GetType(out uc480.Defines.Whitebalance.Type type)
```

Description

Returns the current set type of the auto white balance control.

Parameter

- **type:** Returns the type for auto white balance, see [GetSupportedType\(\)](#).

Class[uc480.AutoFeaturesSoftwareWhitebalance](#)**Accessible**

Camera.AutoFeatures.Software.Whitebalance

Syntax

uc480.AutoFeaturesSoftwareWhitebalance.SetColorModel(uc480.Defines.ColorTemperatureRgbMode colorMode)

Description

Sets the color space for auto white balance.

Parameter

- **colorMode:** Sets the color space, see [GetSupportedColorModel\(\)](#).

Class[uc480.AutoFeaturesSoftwareWhitebalance](#)**Accessible**

Camera.AutoFeatures.Software.Whitebalance

Syntax

```
uc480.AutoFeaturesSoftwareWhitebalance.SetEnable(bool bEnable)
uc480.AutoFeaturesSoftwareWhitebalance.SetEnable(uc480.Defines.ActivateMode activateMode)
```

Description

Enables/disables auto white balance.

Parameter

bEnable	1 = Enable auto white balance
	0 = Disable auto white balance
activateMode	Sets the white balance mode: <ul style="list-style-type: none"> • uc480.Defines.ActivateMode.Disable • uc480.Defines.ActivateMode.Enable • uc480.Defines.ActivateMode.Once

Class[uc480.AutoFeaturesSoftwareWhitebalance](#)**Accessible**

Camera.AutoFeatures.Software.Whitebalance

Syntax

uc480.AutoFeaturesSoftwareWhitebalance.GetType(uc480.Defines.Whitebalance.Type type)

Description

Sets the type for auto white balance.

Parameter

- `type`: Sets the type for auto white balance, see [GetSupportedType\(\)](#).

3.1.1.2.3 GetInfo**Class**[uc480.AutoFeatures](#)**Accessible**

Camera.AutoFeatures

Syntax`uc480.AutoFeatures.GetInfo(out uc480.Types.AutoFeature.Information information)`**Description**

Returns status information on the automatic image control features. This information is written to the `uc480.Types.AutoFeature.Information` structure.



The status information returned is only valid if at least one of the auto control features has been enabled using [Sensor](#) or [Software](#).

Parameter

- `information`: Returns [uc480.Types.AutoFeature.Information](#)

3.1.1.3 BlackLevel

The `BlackLevel` class provides methods for controlling the black level correction 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 by the `offset` parameter.



UI-112x/UI-512x only: On models with HDR sensor `BlackLevel` controls the black level when a contrast adjustment is performed.

Methods

Method	Description
Get	Returns the current mode of black level correction.
GetDefault	Returns the default mode for the black level correction.
GetDefaultOffset	Returns the default value for offset.
GetOffset	Returns the currently set value for offset.
GetOffsetRange	Returns the range for the offset.
GetSupported	Returns if the black level correction is supported.
GetSupportedOffset	Returns if offset is supported.
Set	Sets the mode for black level correction.
SetOffset	Sets the offset for black level correction.

3.1.1.3.1 Get

Class

[uc480.BlackLevel](#)

Accessible

Camera.BlackLevel

Syntax

```
uc480.BlackLevel.Get(out uc480.Defines.BlackLevelMode mode)
```

Description

Returns the current mode of black level correction.

Parameter

mode	Returns the current mode: <ul style="list-style-type: none"> • uc480.Defines.BlackLevelMode.Enable • uc480.Defines.BlackLevelMode.Invalid • uc480.Defines.BlackLevelMode.Offset
------	---

3.1.1.3.2 GetDefault

Class

[uc480.BlackLevel](#)

Accessible

Camera.BlackLevel

Syntax

```
uc480.BlackLevel.GetDefault(out uc480.Defines.BlackLevelMode mode)
```

Description

Returns the default mode.

Parameter

- mode: Returns the standard mode (see [Get\(\)](#)).

3.1.1.3.3 GetDefaultOffset

Class

[uc480.BlackLevel](#)

Accessible

Camera.BlackLevel

Syntax

```
uc480.BlackLevel.GetDefaultOffset(out int s32Offset)
```

Description

Returns the default value for offset.

Parameter

- s32Offset: Returns the standard value for the offset.

3.1.1.3.4 GetOffset

Class

[uc480.BlackLevel](#)

Accessible

Camera.BlackLevel

Syntax

`uc480.BlackLevel.GetOffset(out int s32Offset)`

Description

Returns the currently set value for offset, which is used in addition to the black level correction. Valid values are between 0 and 255.

Parameter

- `s32Offset`: Returns the current value for the offset.

3.1.1.3.5 GetOffsetRange

Class

[uc480.BlackLevel](#)

Accessible

Camera.BlackLevel

Syntax

`uc480.BlackLevel.GetOffsetRange(out uc480.Types.Range<int> range)`
`uc480.BlackLevel.GetOffsetRange(out int s32Min, out int s32Max, out int s32Inc)`

Description

Returns the minimum and maximum value and the increment for the offset. Valid values are between 0 and 255.

Parameter

range	<code>Minimum</code> : Returns the minimum value for the offset.
	<code>Maximum</code> : Returns the maximum value for the offset.
	<code>Increment</code> : Returns the increment for the offset.
s32Min	Returns the minimum value for the offset.
s32Max	Returns the maximum value for the offset.
s32Inc	Returns the increment for the offset.

3.1.1.3.6 GetSupported

Class

[uc480.BlackLevel](#)

Accessible

Camera.BlackLevel

Syntax

`uc480.BlackLevel.GetSupported(out bool bSupported)`

```
uc480.BlackLevel.GetSupported(out uc480.Defines.BlackLevelMode mode)
```

Description

Returns if the black level correction is supported.

Parameter

bSupported	1 = Black level correction is supported 0 = Black level correction is not supported
mode	Returns if black level correction is supported. Additionally it is ORed if offset is supported.

3.1.1.3.7 GetSupportedOffset

Class

[uc480.BlackLevel](#)

Accessible

Camera.BlackLevel

Syntax

```
uc480.BlackLevel.GetSupportedOffset(out bool bSupported)
```

Description

Returns if offset is supported.

Parameter

bSupported	1 = Offset is supported 0 = Offset is not supported
------------	--

3.1.1.3.8 Set

Class

[uc480.BlackLevel](#)

Accessible

Camera.BlackLevel

Syntax

```
uc480.BlackLevel.Set(uc480.Defines.BlackLevelMode mode)
```

Description

Sets the mode for black level correction.

Parameter

mode	Mode to be set for black level correction: <ul style="list-style-type: none"> • uc480.Defines.BlackLevelMode.Enable • uc480.Defines.BlackLevelMode.Invalid • uc480.Defines.BlackLevelMode.Offset
------	--

3.1.1.3.9 SetOffset

Class

[uc480.BlackLevel](#)

Accessible

Camera.BlackLevel

Syntax

```
uc480.BlackLevel.SetOffset(int s32Offset)
```

Description

Sets the offset for black level correction.

Parameter

- `s32Offset`: Value to be set for offset.

3.1.1.4 Color

The `Color` class provides classes for classes for setting the color conversion and color correction. It also provides a class for setting the bit depth per pixel or the color temperature. The following classes and method exist:

- [Converter](#)
- [Correction](#)
- [Depth](#)
- [Temperature](#)

3.1.1.4.1 Converter

The `Converter` class provides methods for setting the Bayer conversion mode for a specified color mode. Software conversion is done on the PC, while hardware conversion (GigE uc480 HE and USB 3 uc480 CP) 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: Color filters" chapter in the uc480 manual.



Hardware conversion is only supported by GigE uc480 HE and USB 3 uc480 CP cameras.



Software conversion with the large filter mask should only be used for sensors whose green pixels have the same sensitivity. This applies to the following sensors:

- All uc480 CCD sensors
- CMOS sensors of the UI-122x/UI-322x/UI-522x

For all other sensors, we recommend using the standard filter mask.



While free run mode is active, you cannot change the color conversion type. To do so, you must first stop the capturing process using [Stop\(\)](#) or set the camera to trigger mode (see [Trigger](#)).

Methods

Method	Description
Get	Returns for color cameras the set mode for the specified color mode.
GetDefault	Returns the default converter for a color mode.
GetSupported	Returns for color cameras all available Bayer conversion modes for the specified color mode.
Set	Sets the converter for a color mode.

Class

[uc480.ColorConverter](#)

Accessible

Camera.Color.Converter

Syntax

```
uc480.ColorConverter.Get(uc480.Defines.ColorMode colorMode, out uc480.Defines.ColorConvertMode convertM
```

Description

Returns for color cameras the set mode for the specified color mode. The return value depends on the selected color mode.

Parameter

colorMode	Color mode for which the converter is to be returned (see GetSupported()).
convertMode	Currently selected converter (see GetSupported()).

Class

[uc480.ColorConverter](#)

Accessible

Camera.Color.Converter

Syntax

```
uc480.ColorConverter.GetDefault(uc480.Defines.ColorMode colorMode, out uc480.Defines.ColorConvertMode convertM
```

Description

Returns the default converter for a color mode.

Parameter

colorMode	Color mode for which the converter is to be returned (see GetSupported()).
convertMode	Default converter (see GetSupported()).

Class

[uc480.ColorConverter](#)

Accessible

Camera.Color.Converter

Syntax

`uc480.ColorConverter.GetSupported(uc480.Defines.ColorMode colorMode, out uc480.Defines.ColorConvertMode convertMode)`

Description

Returns for color cameras all available Bayer conversion modes for the specified color mode. The return value depends on the selected color mode.

Parameter

colorMode	<p>Color mode for which the converter is to be returned:</p> <ul style="list-style-type: none"> • uc480.Defines.ColorMode.BGR10Packed • uc480.Defines.ColorMode.BGR10Unpacked • uc480.Defines.ColorMode.BGR12Unpacked • uc480.Defines.ColorMode.BGR565Packed • uc480.Defines.ColorMode.BGR5Packed • uc480.Defines.ColorMode.BGR8Packed • uc480.Defines.ColorMode.BGRA12Unpacked • uc480.Defines.ColorMode.BGRA8Packed • uc480.Defines.ColorMode.BGRY8Packed • uc480.Defines.ColorMode.CBYCRYPacked • uc480.Defines.ColorMode.Jpeg • uc480.Defines.ColorMode.Mono12 • uc480.Defines.ColorMode.Mono16 • uc480.Defines.ColorMode.Mono8 • uc480.Defines.ColorMode.RGB10Packed • uc480.Defines.ColorMode.RGB10Unpacked • uc480.Defines.ColorMode.RGB12Unpacked • uc480.Defines.ColorMode.RGB8Packed • uc480.Defines.ColorMode.RGB8Planar • uc480.Defines.ColorMode.RGBA12Unpacked • uc480.Defines.ColorMode.RGBA8Packed • uc480.Defines.ColorMode.RGY8Packed • uc480.Defines.ColorMode.SensorRaw12 • uc480.Defines.ColorMode.SensorRaw16 • uc480.Defines.ColorMode.SensorRaw8 • uc480.Defines.ColorMode.UYVYBayerPacked • uc480.Defines.ColorMode.UYVYMonoPacked • uc480.Defines.ColorMode.UYVYPacked • uc480.Defines.ColorMode.PreferPackedSourceFormat <p>For more Information, see also the Appendix "Color and memory formats" in the uc480 manual.</p>
convertMode	<p>All converters supported for this color mode. Possible converters are:</p> <ul style="list-style-type: none"> • uc480.Defines.ColorConvertMode.None: No conversion • uc480.Defines.ColorConvertMode.Software: Only for monochrome cameras, if you want to add a gamma • uc480.Defines.ColorConvertMode.Software3X3: Software conversion using the standard filter mask (default) • uc480.Defines.ColorConvertMode.Software5X5: Software conversion using a large filter mask • uc480.Defines.ColorConvertMode.Hardware3X3: Hardware conversion using the standard filter mask (GigE uc480 only) • uc480.Defines.ColorConvertMode.OpenCL3X3: Software conversion using the standard filter mask, but conversion is done on the graphic board

Class

[uc480.ColorConverter](#)

Accessible

Camera.Color.Converter

Syntax

```
uc480.ColorConverter.Set(uc480.Defines.ColorMode colorMode, uc480.Defines.ColorConvertMode convertMode)
```

Description

Sets the converter for a specific color mode.

Parameter

colorMode	Color mode for which the converter is to be set (see GetSupported()).
convertMode	Convert mode to be set (see GetSupported()).

3.1.1.4.2 Correction

The `Correction` class provides methods for setting the color correction for color cameras. This enhances the rendering of colors for cameras with color sensors. Color correction is a digital correction based on a color matrix which is adjusted individually for each sensor.

If you perform Bayer conversion for GigE uc480 HE color cameras in the camera itself, [color conversion](#) will automatically also take place in the camera.



Color correction is enabled by default for the camera models UI-149x/UI-549x.



After changing this parameter, perform manual or automatic white balancing in order to obtain correct color rendering (see also [Software](#)).

Methods

Method	Description
Get	Returns the current color correction mode.
GetDefault	Returns the default color correction mode.
GetSupported	Returns all supported color correction modes.
Set	Sets the color correction mode.

Class

[uc480.ColorCorrection](#)

Accessible

Camera.Color.Correction

Syntax

```
uc480.ColorCorrection.Get(out uc480.Defines.ColorCorrectionMode correctionMode, out double f64Factor)
```

Description

Returns the current color correction mode.

Parameter

correctionMode	Returns the current set color correction mode (see GetSupported()).
f64Factor	Returns the strength of the color correction between 0.0 (no correction) and 1.0 (strong correction).

Class

[uc480.ColorCorrection](#)

Accessible

Camera.Color.Correction

Syntax

```
uc480.ColorCorrection.GetDefault(out uc480.Defines.ColorCorrectionMode correctionMode)
```

Description

Returns the default color correction mode.

Parameter

- `correctionMode`: Returns the default color correction mode.

Class

[uc480.ColorCorrection](#)

Accessible

Camera.Color.Correction

Syntax

```
uc480.ColorCorrection.GetSupported(out uc480.Defines.ColorCorrectionMode correctionMode)
```

Description

Returns all supported color correction modes.

Parameter

correctionMode	<p>Returns the supported values linked by a logical OR:</p> <ul style="list-style-type: none"> • uc480.Defines.ColorCorrectionMode.Disable: Disable color correction. • uc480.Defines.ColorCorrectionMode.Enable: Enable color correction. • uc480.Defines.ColorCorrectionMode.EnableNormal: Simple color correction. • uc480.Defines.ColorCorrectionMode.EnableBG40Enhanced: Color correction for cameras with optical IR filter glasses of the BG40 type. • uc480.Defines.ColorCorrectionMode.EnableHqEnhanced: Color correction for cameras with optical IR filter glasses of the HQ type. • uc480.Defines.ColorCorrectionMode.IrAutomatic: When used for monochrome cameras, the system returns 0.
----------------	--

Class

[uc480.ColorCorrection](#)

Accessible

Camera.Color.Correction

Syntax

```
uc480.ColorCorrection.Set(uc480.Defines.ColorCorrectionMode correctionMode, double f64Factor)
```

Description

Sets the color correction mode.

Parameter

correctionMode	Sets the color correction mode (see GetSupported()).
f64Factors	Sets the strength of the color correction between 0.0 (no correction) and 1.0 (strong correction).

3.1.1.4.3 Depth

The `Depth` class provides a method for retrieving the current Windows Desktop color setting and returns the bit depth per pixel and the matching uc480 color mode. The color mode can be passed directly to the set pixel format ([Set\(\)](#)). You need to pass the bit depth when allocating an image memory.

Methods

Method	Description
Get	Returns the bit depth of the color setting.

Class

[uc480.ColorDepth](#)

Accessible

Camera.Color.Depth

Syntax

```
uc480.ColorDepth.Get(out int colorDepth, out uc480.Defines.ColorMode colorMode)
```

Description

Retrieves the current Windows Desktop color setting and returns the bit depth of the color setting and the matching uc480 color mode.

Parameter

colorDepth	Returns the bit depth of the color setting.
colorMode	Returns the uc480 color mode that corresponds to <code>s32Col</code> . For a list of all available color formats and the associated input parameters, see the Appendix "Color and memory formats" in the uc480 manual.

3.1.1.4.4 Temperature

The `Temperature` class provides methods for setting the color temperature (in Kelvin) of an image when you are using a color camera. The class will use the sensor's hardware gain controls for the setting, as far as possible. In addition, you can choose between different color spaces. A specific color temperature will result in slightly differing RGB values, depending on the selected color space.

The color temperature is the temperature to which a black body radiator has to be heated to glow and give off light in the corresponding color. Warm light (reddish) has a lower color temperature than cold light (bluish). The following table lists a few example values:

Light source	Color temperature
Light bulb (100 W)	2800
Halogen lamp	3200
Fluorescent lamp (cold white)	4000
Morning and evening sunlight	5000
Noon sunlight	5500-5800
Flash strobe	6000
Overcast daylight	6500-7500
Fog	8000





`ColorTemperature` cannot be used simultaneously with the automatic white balance methods in [Software](#).

The following class and methods exist:

- [LscModel](#)
- [RgbModel](#)

Methods

Method	Description
Get	Returns the set color temperature.
GetDefault	Returns the default value for the color temperature.
GetRange	Returns the range of the color temperature.
Set	Sets a color temperature.

The `LscModel` class provides methods for setting a lens shading model. A specific color model will result in slightly differing RGB values, depending on the selected lens shading model.



This class is currently only supported by the XS model.

Methods

Method	Description
Get	Returns the current lens shading model.
GetDefault	Returns the default lens shading model.
GetSupported	Returns the supported lens shading models.
Set	Sets the lens shading model.

Class

[uc480.ColorTemperatureLscModel](#)

Accessible

Camera.Color.Temperature.LscModel

Syntax

`uc480.ColorTemperatureLscModel.Get(out uc480.Defines.ColorTemperatureLscMode modelMode)`

Description

Returns the current lens shading model.

Parameter

- `modelMode`: Returns the current value (see [GetSupported\(\)](#)).

Class

[uc480.ColorTemperatureLscModel](#)

Accessible

Camera.Color.Temperature.LscModel

Syntax

`uc480.ColorTemperatureLscModel.GetDefault(out uc480.Defines.ColorTemperatureLscMode modelMode)`

Description

Returns the default lens shading model.

Parameter

- `modelMode`: Returns the default value (see [GetSupported\(\)](#)).

Class

[uc480.ColorTemperatureLscModel](#)

Accessible

Camera.Color.Temperature.LscModel

Syntax

`uc480.ColorTemperatureLscModel.GetSupported(out uc480.Defines.ColorTemperatureLscMode modelMode)`

Description

Returns the supported lens shading models.

Parameter

modelMode	<p>Returns the supported lens shading models.</p> <ul style="list-style-type: none">• <code>uc480.Defines.ColorTemperatureLscMode.AGL</code>: lens shading model with a white point of 3000 kelvins• <code>uc480.Defines.ColorTemperatureLscMode.D50</code>: lens shading model with a white point of 5000 kelvins• <code>uc480.Defines.ColorTemperatureLscMode.D65</code>: lens shading model with a white point of 6500 kelvins• <code>uc480.Defines.ColorTemperatureLscMode.TL84</code>: lens shading model with a white point of 4000 kelvins
-----------	--

Class

[uc480.ColorTemperatureLscModel](#)

Accessible

Camera.Color.Temperature.LscModel

Syntax

`uc480.ColorTemperatureLscModel.Set(uc480.Defines.ColorTemperatureLscMode modelMode)`

Description

Sets the lens shading model.

Parameter

- `modelMode`: Lens shading model to be set.

The `RgbModel` class provides methods for setting the color space. A specific color space will result in slightly differing RGB values, depending on the selected color space.

Methods

Method	Description
Get	Returns the set color space.
GetDefault	Returns the default color space.
GetSupported	Returns the supported color spaces.
Set	Sets a color space.

Class

[uc480.ColorTemperatureRgbModel](#)

Accessible

Camera.Color.Temperature.RgbModel

Syntax

```
uc480.ColorTemperatureRgbModel.Get(out uc480.Defines.ColorTemperatureRgbMode modelMode)
```

Description

Returns the set color space.

Parameter

- `modelMode`: Returns the current value (see [GetSupported\(\)](#)).

Class

[uc480.ColorTemperatureRgbModel](#)

Accessible

Camera.Color.Temperature.RgbModel

Syntax

```
uc480.ColorTemperatureRgbModel.GetDefault(out uc480.Defines.ColorTemperatureRgbMode modelMode)
```

Description

Returns the default color space.

Parameter

- `modelMode`: Returns the default value (see [GetSupported\(\)](#)).

Class

[uc480.ColorTemperatureRgbModel](#)

Accessible

Camera.Color.Temperature.RgbModel

Syntax

`uc480.ColorTemperatureRgbModel.GetSupported(out uc480.Defines.ColorTemperatureRgbMode modelMode)`

Description

Returns the supported color spaces.

Parameter

modelMode	<p>Returns the supported color spaces.</p> <ul style="list-style-type: none">• <code>uc480.Defines.ColorTemperatureRgbMode.ADOBE_RGB_D65</code>: Adobe RGB color space with a white point of 6500 kelvins (mid daylight). The Adobe RGB color space is larger than the sRGB color space, but not all devices can render it.• <code>uc480.Defines.ColorTemperatureRgbMode.CIE_RGB_E</code>: CIE-RGB color space with standard illumination E• <code>uc480.Defines.ColorTemperatureRgbMode.ECI_RGB_D50</code>: ECI-RGB color space with a white point of 5000 kelvins (warm light)• <code>uc480.Defines.ColorTemperatureRgbMode.SRGB_D50</code>: sRGB (standard RGB) color space with a white point of 5000 kelvins (warm light)• <code>uc480.Defines.ColorTemperatureRgbMode.SRGB_D65</code>: sRGB (standard RGB) color space with a white point of 6500 kelvins (mid daylight)
-----------	--

Class

[uc480.ColorTemperatureRgbModel](#)

Accessible

Camera.Color.Temperature.RgbModel

Syntax

`uc480.ColorTemperatureRgbModel.Set(uc480.Defines.ColorTemperatureRgbMode modelMode)`

Description

Sets a color space.

Parameter

- `modelMode`: Color space to be set (see [GetSupported\(\)](#)).

Class

[uc480.ColorTemperature](#)

Accessible

Camera.Color.Temperature

Syntax

```
uc480.ColorTemperature.Get(out uint u32ColTemp)
```

Description

Returns the set color temperature.

Parameter

- **u32ColTemp:** Returns the current value.

Class

[uc480.ColorTemperature](#)

Accessible

Camera.Color.Temperature

Syntax

```
uc480.ColorTemperature.GetDefault(out uint u32DefColTemp)
```

Description

Returns the default value for the color temperature.

Parameter

- **u32DefColTemp:** Returns the default value.

Class

[uc480.ColorTemperature](#)

Accessible

Camera.Color.Temperature

Syntax

```
uc480.ColorTemperature.GetRange(out uc480.Types.Range<uint> range)
```

```
uc480.ColorTemperature.GetRange(out uint u32MinColTemp, out uint u32MaxColTemp, out uint u32IncColTemp)
```

Description

Returns the range of the color temperature.

Parameter

range	Minimum: Returns the minimum value for the color temperature. Maximum: Returns the maximum value for the color temperature. Increment: Returns the increment for setting the color temperature.
u32MinColTemp	Returns the minimum value for the color temperature.
u32MaxColTemp	Returns the maximum value for the color temperature.
u32IncColTemp	Returns the increment for setting the color temperature.

Class

[uc480.ColorTemperature](#)

Accessible

Camera.Color.Temperature

Syntax

`uc480.ColorTemperature.Set(uint u32ColTemp)`

Description

Sets a color temperature.

Parameter

- `u32ColTemp`: Color temperature to be set.

3.1.1.5 Device

The `Device` class provides methods for setting the camera ID and initializing the camera. The following class and methods exist:

- [Feature](#)

Methods

Method	Description
GetCameraID	Returns the current camera ID.
GetDeviceID	Returns the current device ID.
GetEnableAutoExit	Returns the state of the automatic closing of the camera handle.
SetCameraID	Sets a unique camera ID to a camera.
SetEnableAutoExit	Enables/disables automatic closing of the camera handle.

3.1.1.5.1 Feature

The `Feature` class provides classes for configuring special camera functions provided by specific uc480 models:

- On UI-124x/UI-324x/UI-524x and UI-125x/UI-325x/UI-525x models: Set line scan mode.
- On UI-124x/UI-324x/UI-524x and UI-125x/UI-325x/UI-525x models: Toggle between rolling and global shutter mode.
- On UI-124x/UI-324x/UI-524x and UI-125x/UI-325x/UI-525x models: Control the Log mode
- On UI-1008XS model: Choose the HS mode for triggered image capture.
- On XS model: "Reduce noise" function, JPEG compression, image effects and temperature
- On UI-324x, UI-336x/UI-536x and UI-337x/UI-537x models: Set AOI merge mode
- On UI-336x/UI-536x and UI-337x/UI-537x models: Enable/disable FPN correction, enable/disable black reference columns and set the sensor bit depth

The following classes exist:

- [AOIMerge](#)
- [BlackReference](#)

- [FpnCorrection](#)
- [ImageEffect](#)
- [JpegCompression](#)
- [Linescan](#)
- [Log](#)
- [MultiIntegration](#)
- [NoiseReduction](#)
- [SensorBitDepth](#)
- [SensorSourceGain](#)
- [ShutterMode](#)
- [Temperature](#)
- [Timestamp](#)
- [XSHSMode](#)

The `AOIMerge` class provides classes for controlling the special AOI merge mode which combines the lines of an AOI to a new image.



Note: This feature is currently only supported by the UI-324x, UI-336x/UI-536x and UI-337x/UI-537x models.

Depending on the current AOI height each 2 double lines are combined into one image. X is half of the AOI height. In the default (full screen) is x = 512. The resulting image is 1024 pixels high, and thus re-fit the sensor size in full-screen.

The following class exists:

- [Vertical](#)

The `Vertical` class provides methods for setting the vertical AOI merge in the special AOI merge mode which combines the lines of an AOI to a new image. For setting the position and height of the AOI use the classes `Position` and `Height`.



Note: This feature is currently only supported by the UI-324x, UI-336x/UI-536x and UI-337x/UI-537x models.

The following classes exist:

- [AdditionalPosition](#)
- [Height](#)
- [Position](#)

Methods

Method	Description
Get	Returns the current AOI merge mode.
GetDefault	Returns the default AOI merge mode.
GetEnable	Returns if the AOI merge mode is enabled.
GetSupported	Returns if the AOI merge mode is supported.
Set	Sets a new AOI merge mode.
SetEnable	Enables/disabled the AOI merge mode.

The `AdditionalPosition` class provides methods for setting the additional position of the special AOI merge mode.



Note: This feature is currently only supported by the UI-324x, UI-336x/UI-536x and UI-337x/UI-537x models.

Methods

Method	Description
<code>Get</code>	Returns the position of the additional used sensor line(s).
<code>GetDefault</code>	Returns the default value for the additional AOI merge position.
<code>GetRange</code>	Returns the range for the additional AOI merge position.
<code>Set</code>	Sets the position of the additional used sensor line(s).

Class

[uc480.DeviceFeatureAOIMerge.AoiMergeAdditionalPosition](#)

Accessible

Camera.Device.Feature.AoiMerge.Vertical.AdditionalPosition

Syntax

```
uc480.DeviceFeatureAOIMerge.AoiMergeAdditionalPosition.Get(out uint position)
```

Description

Returns the position of the additional used sensor line(s).

Parameter

- `position`: Returns the number of the additional sensor lines (i.e. 0 means the two top lines)

Class

[uc480.DeviceFeatureAOIMerge.AoiMergeAdditionalPosition](#)

Accessible

Camera.Device.Feature.AoiMerge.Vertical.AdditionalPosition

Syntax

```
uc480.DeviceFeatureAOIMerge.AoiMergeAdditionalPosition.GetDefault(out uint position)
```

Description

Returns the default value for the additional AOI merge position.

Parameter

- `position`: Returns the default number of the sensor lines for the additional AOI merge position.

Class

[uc480.DeviceFeatureAOIMerge.AoiMergeAdditionalPosition](#)

Accessible

Camera.Device.Feature.AoiMerge.Vertical.AdditionalPosition

Syntax

```
uc480.DeviceFeatureAOIMerge.AoiMergeAdditionalPosition.GetRange(out uc480.Types.Range<uint> range)
uc480.DeviceFeatureAOIMerge.AoiMergeAdditionalPosition.GetRange(out uint min, out uint max, out uint i
```

Description

Returns the range for the additional AOI merge position.

Parameter

range	Minimum: returns the minimum value Maximum: returns the maximum value Increment: returns the increment
min	Returns the minimum value.
max	Returns the maximum value.
inc	Returns the increment.

Class

[uc480.DeviceFeatureAOIMerge.AoiMergeAdditionalPosition](#)

Accessible

Camera.Device.Feature.AoiMerge.Vertical.AdditionalPosition

Syntax

```
uc480.DeviceFeatureAOIMerge.AoiMergeAdditionalPosition.Set(uint position)
```

Description

Sets the position of the additional used sensor line(s).

Parameter

- position: Value to be set (i.e. 0 means the two top lines)

The Height class provides methods for setting the AOI merge height. The minimum AOI merge height is 1



Note: This feature is currently only supported by the UI-324x, UI-336x/UI-536x and UI-337x/UI-537x models.

Methods

Method	Description
Get	Returns the current set AOI merge height.
GetDefault	Returns the default value of the AOI merge height (2 for color, 1 for monochrome).
GetList	Returns the list with the allowed AOI merge heights.
GetNumber	Returns the number of list elements with the allowed AOI merge heights. The image height has to be a multiple of the AOI merge height. Therefore, the list of the current image height changes.
Set	Sets the AOI merge height.

Class

[uc480.DeviceFeatureAOIMerge.AoiMergeHeight](#)

Accessible

Camera.Device.Feature.AoiMerge.Vertical.Height

Syntax

`uc480.DeviceFeatureAOIMerge.AoiMergeHeight.Get(out uint height)`

Description

Returns the current set AOI merge height.

Parameter

- `height`: Returns the current AOI merge height.

Class

[uc480.DeviceFeatureAOIMerge.AoiMergeHeight](#)

Accessible

Camera.Device.Feature.AoiMerge.Vertical.Height

Syntax

`uc480.DeviceFeatureAOIMerge.AoiMergeHeight.GetDefault(out uint height)`

Description

Returns the default value of the AOI merge height (2 for color, 1 for monochrome).

Parameter

- `height`: Returns the default AOI merge height.

Class

[uc480.DeviceFeatureAOIMerge.AoiMergeHeight](#)

Accessible

Camera.Device.Feature.AoiMerge.Vertical.Height

Syntax

```
uc480.DeviceFeatureAOIMerge.AoiMergeHeight.GetList(out uint[] height)
```

Description

Returns the list with the allowed AOI merge heights.

Parameter

- `height`: Returns a list with possible AOI merge heights.

Class

[uc480.DeviceFeatureAOIMerge.AoiMergeHeight](#)

Accessible

Camera.Device.Feature.AoiMerge.Vertical.Height

Syntax

```
uc480.DeviceFeatureAOIMerge.AoiMergeHeight.GetNumber(out uint number)
```

Description

Returns the number of list elements with the allowed AOI merge heights. The image height has to be a multiple of the AOI merge height. Therefore, the list of the current image height changes.

Parameter

- `number`: Returns the number of list elements.

Class

[uc480.DeviceFeatureAOIMerge.AoiMergeHeight](#)

Accessible

Camera.Device.Feature.AoiMerge.Vertical.Height

Syntax

```
uc480.DeviceFeatureAOIMerge.AoiMergeHeight.Set(uint height)
```

Description

Sets the AOI merge height.

Parameter

- `height`: Value to be set (for color sensors at least 2).

The `Position` class provides methods for setting the position in the special AOI merge mode which combines the lines of an AOI to a new image. For enabling/disabling the AOI merge mode see [Vertical](#).



Note: This feature is currently only supported by the UI-324x, UI-336x/UI-536x and UI-337x/UI-537x models.

Methods

Method	Description
Get	Returns the position of the two sensor lines.
GetDefault	Returns the default value for the AOI merge position (default = 0, i.e. the two top lines).
GetRange	Returns the setting range for the AOI merge position.
GetSet	Sets the position of the two sensor lines.

Class

[uc480.DeviceFeatureAOIMerge.AoiMergePosition](#)

Accessible

Camera.Device.Feature.AoiMerge.Vertical.Position

Syntax

`uc480.DeviceFeatureAOIMerge.AoiMergePosition.Get(out uint position)`

Description

Returns the position of the two sensor lines for AOI merge mode.

Parameter

- `position`: Returns the number of the sensor lines (i.e. 0 means the two top lines)

Class

[uc480.DeviceFeatureAOIMerge.AoiMergePosition](#)

Accessible

Camera.Device.Feature.AoiMerge.Vertical.Position

Syntax

`uc480.DeviceFeatureAOIMerge.AoiMergePosition.GetDefault(out uint position)`

Description

Returns the default value for the AOI merge position (default = 0, i.e. the two top lines).

Parameter

- `position`: Returns the default number of the sensor lines for the AOI merge position.

Class

[uc480.DeviceFeatureAOIMerge.AoiMergePosition](#)

Accessible

Camera.Device.Feature.AoiMerge.Vertical.Position

Syntax

`uc480.DeviceFeatureAOIMerge.AoiMergePosition.GetRange(out uc480.Types.Range<uint> range)`
`uc480.DeviceFeatureAOIMerge.AoiMergePosition.GetRange(out uint min, out uint max, out uint inc)`

Description

Returns the range for the AOI merge position.

Parameter

range	Minimum: returns the minimum value Maximum: returns the maximum value Increment: returns the increment
min	Returns the minimum value.
max	Returns the maximum value.
inc	Returns the increment.

Class

[uc480.DeviceFeatureAOIMerge.AoiMergePosition](#)

Accessible

Camera.Device.Feature.AoiMerge.Vertical.Position

Syntax

`uc480.DeviceFeatureAOIMerge.AoiMergePosition.Set(uint position)`

Description

Sets the position of the two sensor lines for AOI merge mode.

Parameter

- `position`: Value to be set (i.e. 0 means the two top lines)

Class

[uc480.DeviceFeatureAOIMerge.VerticalMerge](#)

Accessible

Camera.Device.Feature.AoiMerge.Vertical

Syntax

`uc480.DeviceFeatureAOIMerge.VerticalMerge.Get(out uc480.Defines.VerticalAoiMergeMode mode)`

Description

Returns the current AOI merge mode.

Parameter

- `mode`: current AOI merge mode (see [GetDefault\(\)](#))

Class

[uc480.DeviceFeatureAOIMerge.VerticalMerge](#)

Accessible

Camera.Device.Feature.AoiMerge.Vertical

Syntax

`uc480.DeviceFeatureAOIMerge.VerticalMerge.GetDefault(out uc480.Defines.VerticalAoiMergeMode mode)`

Description

Returns the default AOI merge mode.

Parameter

mode	uc480.Defines.VerticalAoiMergeMode.Of: AOI merge mode is off
	uc480.Defines.VerticalAoiMergeMode.FallingGpio1: Sensor is triggered by falling edge on GPIO 1
	uc480.Defines.VerticalAoiMergeMode.FallingGpio2: Sensor is triggered by falling edge on GPIO 2
	uc480.Defines.VerticalAoiMergeMode.RisingGpio1: Sensor is triggered by rising edge on GPIO 1
	uc480.Defines.VerticalAoiMergeMode.RisingGpio2: Sensor is triggered by rising edge on GPIO 2
	uc480.Defines.VerticalAoiMergeMode.Freerun: Sensor is operated with maximum speed.
	uc480.Defines.VerticalAoiMergeMode.Software: Sensor is triggered via software

Class[uc480.DeviceFeatureAOIMerge.VerticalMerge](#)**Accessible**

Camera.Device.Feature.AoiMerge.Vertical

Syntax

uc480.DeviceFeatureAOIMerge.VerticalMerge.GetEnable(out bool enable)

Description

Returns if the AOI merge mode is enabled.

Parameter

enable	1 = AOI merge mode is enabled. 0 = AOI merge mode is disabled.
--------	---

Class[uc480.DeviceFeatureAOIMerge.VerticalMerge](#)**Accessible**

Camera.Device.Feature.AoiMerge.Vertical

Syntax

uc480.DeviceFeatureAOIMerge.VerticalMerge.GetSupported(out bool supported)

Description

Returns if the AOI merge mode is supported.

Parameter

supported	0 = Not supported 1 = Supported
-----------	------------------------------------

Class[uc480.DeviceFeatureAOIMerge.VerticalMerge](#)**Accessible**

Camera.Device.Feature.AoiMerge.Vertical

Syntax

uc480.DeviceFeatureAOIMerge.VerticalMerge.Set(uc480.Defines.VerticalAoiMergeMode mode)

Description

Sets a new AOI merge mode.

Parameter

- mode: mode to be set (see [GetDefault\(\)](#))

Class[uc480.DeviceFeatureAOIMerge.VerticalMerge](#)**Accessible**

Camera.Device.Feature.AoiMerge.Vertical

Syntax

uc480.DeviceFeatureAOIMerge.VerticalMerge.SetEnable(bool enable)

Description

Enables/disabled the AOI merge mode.

Parameter

enable	1 = Enables the AOI merge mode 0 = Disables the AOI merge mode
--------	---

The `BlackReference` class provides methods for activating the display of black reference columns in the image. These can be used for calibrating the black level.



Note: This feature is currently only supported by the models UI-336x/UI-536x and UI-337x/UI-537x.

Methods

Method	Description
Get	Returns the current setting of the black reference columns.
GetDefault	Returns the default setting for the black reference columns.
GetSupported	Returns if setting black reference columns is supported.
Set	Sets the display of black reference columns.

Class

[uc480.DeviceFeatureBlackReference](#)

Accessible

Camera.Device.Feature.BlackReference

Syntax

`uc480.DeviceFeatureBlackReference.Get(out uc480.Defines.BlackReferenceMode mode)`

Description

Returns the current setting of the black reference columns.

Parameter

mode	<code>uc480.Defines.BlackReferenceMode.ColumnsLeft</code> : Black reference columns are displayed on the left image side <code>uc480.Defines.BlackReferenceMode.Off</code> : Black reference columns are not displayed.
------	--

Class

[uc480.DeviceFeatureBlackReference](#)

Accessible

Camera.Device.Feature.BlackReference

Syntax

`uc480.DeviceFeatureBlackReference.GetDefault(out uc480.Defines.BlackReferenceMode mode)`

Description

Returns the default setting for the black reference columns.

Parameter

- mode: default mode for the black reference columns

Class

[uc480.DeviceFeatureBlackReference](#)

Accessible

Camera.Device.Feature.BlackReference

Syntax

```
uc480.DeviceFeatureBlackReference.GetSupported(out bool supported)
```

Description

Returns if setting black reference columns is supported.

Parameter

supported	0 = Not supported
	1 = Supported

Class

[uc480.DeviceFeatureBlackReference](#)

Accessible

Camera.Device.Feature.BlackReference

Syntax

```
uc480.DeviceFeatureBlackReference.Set(uc480.Defines.BlackReferenceMode mode)
```

Description

Sets the display of black reference columns.

Parameter

mode	uc480.Defines.BlackReferenceMode.ColumnsLeft: Display black reference columns on the left image side uc480.Defines.BlackReferenceMode.Off: No display
------	--

The `FpnCorrection` class provides methods for setting the FPN (fixed pattern noise) correction.



Note: This feature is currently only supported by the models UI-336x/UI-536x and UI-337x/UI-537x.

Methods

Method	Description
Get	Returns the current setting of the FPN correction
GetDefault	Returns the default setting of the FPN (fixed pattern noise) correction.
GetSupported	Returns if setting the FPN correction is supported.
Set	Sets the settings of the FPN correction

Class

[uc480.DeviceFeatureFpnCorrection](#)

Accessible

Camera.Device.Feature.FpnCorrection

Syntax

uc480.DeviceFeatureFpnCorrection.Get(out uc480.Defines.FPNCorrectionMode mode)

Description

Returns the current setting of the FPN correction.

Parameter

mode	uc480.Defines.FPNCorrectionMode.Hardware: Enables the FPN correction uc480.Defines.FPNCorrectionMode.Off: Disables the FPN correction in the camera
------	--

Class

[uc480.DeviceFeatureFpnCorrection](#)

Accessible

Camera.Device.Feature.FpnCorrection

Syntax

uc480.DeviceFeatureFpnCorrection.GetDefault(out uc480.Defines.FPNCorrectionMode mode)

Description

Returns the default setting of the FPN (fixed pattern noise) correction (enabled by default).

Parameter

- mode: Default mode for FPN correction

Class

[uc480.DeviceFeatureFpnCorrection](#)

Accessible

Camera.Device.Feature.FpnCorrection

Syntax

uc480.DeviceFeatureFpnCorrection.GetSupported(out bool supported)

Description

Returns if setting the FPN correction is supported.

Parameter

supported	0 = Not supported. 1 = Supported.
-----------	--------------------------------------

Class

[uc480.DeviceFeatureFpnCorrection](#)

Accessible

Camera.Device.Feature.FpnCorrection

Syntax

`uc480.DeviceFeatureFpnCorrection.Set(uc480.Defines.FPNCorrectionMode mode)`

Description

Sets the settings of the FPN correction.

Parameter

- mode: Mode to be set (see [Get\(\)](#))

The `ImageEffect` class provides methods for setting an image effect on some uc480 models.



This class is currently only supported by the XS model.

Methods

Method	Description
Get	Returns the current setting for the image effect.
GetDefault	Returns the default setting for the image effect.
GetSupported	Returns if setting an image effect is supported.
Set	Sets an image effect.

Class

[uc480.DeviceFeatureImageEffect](#)

Accessible

Camera.Device.Feature.ImageEffect

Syntax

`uc480.DeviceFeatureImageEffect.Get(out uc480.Defines.ImageEffectMode effect)`

Description

Returns the current setting for the image effect.

Parameter

- effect: Returns the current setting (see [Set\(\)](#))

Class

[uc480.DeviceFeatureImageEffect](#)

Accessible

Camera.Device.Feature.ImageEffect

Syntax

```
uc480.DeviceFeatureImageEffect.GetDefault(out uc480.Defines.ImageEffectMode effect)
```

Description

Returns the default setting for the image effect (disabled by default).

Parameter

- `effect`: Default value for image effect

Class

[uc480.DeviceFeatureImageEffect](#)

Accessible

Camera.Device.Feature.ImageEffect

Syntax

```
uc480.DeviceFeatureImageEffect.GetSupported(out bool Supported)
```

Description

Returns if setting an image effect is supported.

Parameter

<code>supported</code>	<code>0 = Not supported.</code>
	<code>1 = Supported.</code>

Class

[uc480.DeviceFeatureImageEffect](#)

Accessible

Camera.Device.Feature.ImageEffect

Syntax

```
uc480.DeviceFeatureImageEffect.Set(uc480.Defines.ImageEffectMode effect)
```

Description

Sets an image effect.

Parameter

effect	uc480.Defines.ImageEffectMode.Disable s: Image effects are disabled
	uc480.Defines.ImageEffectMode.Sepia: Sepia effect is enabled
	uc480.Defines.ImageEffectMode.Monochr ome: Monochrome effect is enabled
	uc480.Defines.ImageEffectMode.Negativ e: Negative effect is enabled
	uc480.Defines.ImageEffectMode.Crossha ir: Crosshairs effect is enabled

The `JpegCompression` class provides methods for setting the compression in the JPEG mode.



This class is currently only supported by the XS model.
Only in JPEG mode the maximum frame rate is achieved.

Methods

Method	Description
Get	Returns the current value of the JPEG compression.
GetDefault	Returns the default value of the JPEG compression.
GetRange	Returns the range for the compression in the JPEG mode
GetSupported	Returns if JPEG compression is supported.
Set	Sets the compression for the JPEG mode.

Class

[uc480.DeviceFeatureJpegCompression](#)

Accessible

Camera.Device.Feature.JpegCompression

Syntax

`uc480.DeviceFeatureJpegCompression.Get(out uint value)`

Description

Returns the current value of the JPEG compression.

Parameter

- `value`: Current compression value

Class

[uc480.DeviceFeatureJpegCompression](#)

Accessible

Camera.Device.Feature.JpegCompression

Syntax

`uc480.DeviceFeatureJpegCompression.GetDefault(out uint value)`

Description

Returns the default compression value.

Parameter

- `value`: default compression value (4)

Class

[uc480.DeviceFeatureJpegCompression](#)

Accessible

Camera.Device.Feature.JpegCompression

Syntax

`uc480.DeviceFeatureJpegCompression.GetRange(out uc480.Types.Range<uint> range)`
`uc480.DeviceFeatureJpegCompression.GetRange(out uint min, out uint max, out uint inc)`

Description

Returns the range for the compression in the JPEG mode (1...9). The higher the value, the more the image data is compressed.

Parameter

<code>range</code>	<code>Minimum</code> : returns the minimum value <code>Maximum</code> : returns the maximum value <code>Increment</code> : returns the increment
<code>min</code>	Returns the minimum value.
<code>max</code>	Returns the maximum value.
<code>inc</code>	Returns the increment.

Class

[uc480.DeviceFeatureJpegCompression](#)

Accessible

Camera.Device.Feature.JpegCompression

Syntax

`uc480.DeviceFeatureJpegCompression.GetSupported(out bool supported)`

Description

Returns if JPEG compression is supported.

Parameter

supported	0 = Not supported. 1 = Supported.
-----------	--------------------------------------

Class[uc480.DeviceFeatureJpegCompression](#)**Accessible**

Camera.Device.Feature.JpegCompression

Syntax

uc480.DeviceFeatureJpegCompression.Set(uint value)

Description

Sets the compression for the JPEG mode. To enable/disable the JPEG mode use [Converter.Set\(\)](#).

Parameter

- **value:** Compression value to be set (see also [GetRange\(\)](#))

The `Linescan` class provides methods for configuring special camera functions provided by specific uc480 models:

- On UI-124x/UI-324x/UI-524x models: Set line scan mode.

Methods

Method	Description
Get	Returns the currently set line scan mode.
GetNumber	Returns the scan line used for the line scan mode.
GetSupported	Returns the supported line scan modes.
IsSupported	Returns if the passed line scan mode is supported.
Set	Sets the line scan mode.
SetNumber	Sets the scan line used for the line scan mode.

Class[uc480.DeviceFeatureLinescan](#)**Accessible**

Camera.Device.Feature.Linescan

Syntax

uc480.DeviceFeatureLinescan.Get(out uc480.Defines.LinescanMode mode)

Description

Returns the currently set line scan mode.

Parameter

- mode: Returns the current line scan mode (see [GetSupported\(\)](#)).

Class[uc480.DeviceFeatureLinescan](#)**Accessible**

Camera.Device.Feature.Linescan

Syntax

uc480.DeviceFeatureLinescan.GetNumber(out uint u32Number)

Description

Returns the scan line used for the line scan mode.

Parameter

- u32Number: Returns the currently set line.

Class[uc480.DeviceFeatureLinescan](#)**Accessible**

Camera.Device.Feature.Linescan

Syntax

uc480.DeviceFeatureLinescan.GetSupported(out uc480.Defines.LinescanMode mode)

Description

Returns the supported line scan modes.

Parameter

mode	uc480.Defines.LinescanMode.Fast: Fast line scan mode is supported
	uc480.Defines.LinescanMode.Number: Line scan mode with selectable line number is supported

Class[uc480.DeviceFeatureLinescan](#)**Accessible**

Camera.Device.Feature.Linescan

Syntax

uc480.DeviceFeatureLinescan.IsSupported(uc480.Defines.LinescanMode mode)

Description

Returns if the passed line scan mode is supported.

Parameter

- mode: Line scan mode to be queried to be supported (see [GetSupported\(\)](#)).

Class

[uc480.DeviceFeatureLinescan](#)

Accessible

Camera.Device.Feature.Linescan

Syntax

```
uc480.DeviceFeatureLinescan.Set(uc480.Defines.LinescanMode mode)
```

Description

Sets the line scan mode.

Parameter

mode	uc480.Defines.LinescanMode.Fast: Set fast line scan mode uc480.Defines.LinescanMode.Number: Set line scan mode with selectable line number
------	---

Class

[uc480.DeviceFeatureLinescan](#)

Accessible

Camera.Device.Feature.Linescan

Syntax

```
uc480.DeviceFeatureLinescan.SetNumber(uint u32Number)
```

Description

Sets the scan line used for the line scan mode.

Parameter

- `u32Number`: Line to be set.

The `Log` class provides methods for controlling the special Log mode. The Log mode is a special mode of the camera models UI-124x/UI-324x/UI-524x and UI-125x/UI-325x/UI-525x in which a threshold defines at which point the linear sensitivity pass over into a logarithmic characteristic.

At very short exposure times (less than 0.1 ms) there may occur e.g. so-called crosstalk effects in the global shutter mode, which have the effect that the image content appears brighter in the vertical from top to bottom. This effect can be avoided by the Log mode.



Note: This feature is currently only supported by the UI-124x/UI-324x/UI-524x and UI-125x/UI-325x/UI-525x models.

The following classes and methods exist:

- [LogManualGain](#)
- [LogManualValue](#)

Methods

Method	Description
Get	Returns the current Log mode.
GetDefault	Returns the default settings for the Log mode.
GetSupported	Returns if the Log mode is supported.
Set	Sets the Log mode.

The `LogManualGain` class provides methods for setting the gain. In the manual Log mode it is possible to display the information in the overexposed image areas. This mode is effective for exposure times below 5 ms.

When using the manual Log mode no master gain is possible. The gain can be adjusted with `LogManualGain`. The gain can be adjusted in 5 levels. The level "3" corresponds to a gain factor of 1 for monochrome or color sensors. For the NIR sensor the level "1" corresponds to a gain factor of 1.

Example: A low level results in a low gain and may display more details in the overexposed image areas. A higher level gives a higher gain, thereby a darker image can be brightened



Note: This feature is currently only supported by the UI-124x/UI-324x/UI-524x and UI-125x/UI-325x/UI-525x models.

Methods

Method	Description
Get	Returns the current manual gain of the Log mode.
GetDefault	Returns the default settings for the manual gain for the Log mode.
GetRange	Returns the range for the manual gain of the Log mode.
Set	Sets the manual gain of the Log mode.

Class

[uc480.DeviceFeatureLog.LogManualGain](#)

Accessible

Camera.Device.Feature.Log.ManualGain

Syntax

```
uc480.DeviceFeatureLog.LogManualGain.Get(out uint gain)
```

Description

Returns the current manual gain of the Log mode.

Parameter

- `gain`: Value of the gain in the manual Log mode.

Class

[uc480.DeviceFeatureLog.LogManualGain](#)

Accessible

Camera.Device.Feature.Log.ManualGain

Syntax

`uc480.DeviceFeatureLog.LogManualGain.GetDefault(out uint gain)`

Description

Returns the default settings for the manual gain for the Log mode.

Parameter

- `gain`: Default value for the gain in manual Log mode

Class

[uc480.DeviceFeatureLog.LogManualGain](#)

Accessible

Camera.Device.Feature.Log.ManualGain

Syntax

`uc480.DeviceFeatureLog.LogManualGain.GetRange(out uc480.Types.Range<uint> range)`
`uc480.DeviceFeatureLog.LogManualGain.GetRange(out uint min, out uint max, out uint inc)`

Description

Returns the range for the manual gain of the Log mode.

Parameter

<code>range</code>	Minimum: returns the minimum value Maximum: returns the maximum value Increment: returns the increment
<code>min</code>	Returns the minimum value.
<code>max</code>	Returns the maximum value.
<code>inc</code>	Returns the increment.

Class

[uc480.DeviceFeatureLog.LogManualGain](#)

Accessible

Camera.Device.Feature.Log.ManualGain

Syntax

`uc480.DeviceFeatureLog.LogManualGain.Set(uint gain)`

Description

Sets the manual gain of the Log mode.

Parameter

- `gain`: Value to be set

The `LogManualValue` class provides methods for setting the gain. In the manual Log mode it is possible to display the information in the overexposed image areas. This mode is effective for exposure times below 5 ms.

You can adjust the threshold (0...12) at which the linear sensitivity pass over into a logarithmic characteristic. Here, the value "0" represents the lowest active level and "12" corresponds to the highest level.



Note: This feature is currently only supported by the UI-124x/UI-324x/UI-524x and UI-125x/UI-325x/UI-525x models.

Methods

Method	Description
<u>Get</u>	Returns the current manual value of the Log mode.
<u>GetDefault</u>	Returns the default settings for the manual value of the Log mode.
<u>GetRange</u>	Returns the range of the manual value of the Log mode.
<u>Set</u>	Sets the manual value of the Log mode.

Class

[uc480.DeviceFeatureLog.LogManualValue](#)

Accessible

Camera.Device.Feature.Log.ManualValue

Syntax

`uc480.DeviceFeatureLog.LogManualValue.Get(out uint value)`

Description

Returns the current manual value of the Log mode.

Parameter

- `value`: Manual value of the Log mode.

Class

[uc480.DeviceFeatureLog.LogManualValue](#)

Accessible

Camera.Device.Feature.Log.ManualValue

Syntax

`uc480.DeviceFeatureLog.LogManualValue.GetDefault(out uint value)`

Description

Returns the default settings for the manual value of the Log mode.

Parameter

- `value`: Default manual value of the Log mode

Class

[uc480.DeviceFeatureLog.LogManualValue](#)

Accessible

Camera.Device.Feature.Log.ManualValue

Syntax

```
uc480.DeviceFeatureLog.LogManualValue.GetRange(out uc480.Types.Range<uint> range)
uc480.DeviceFeatureLog.LogManualValue.GetRange(out uint min, out uint max, out uint inc)
```

Description

Returns the range of the manual value of the Log mode.

Parameter

range	Minimum: returns the minimum value Maximum: returns the maximum value Increment: returns the increment
min	Returns the minimum value.
max	Returns the maximum value.
inc	Returns the increment.

Class

[uc480.DeviceFeatureLog.LogManualValue](#)

Accessible

Camera.Device.Feature.Log.ManualValue

Syntax

```
uc480.DeviceFeatureLog.LogManualValue.Set(uint value)
```

Description

Sets the manual value of the Log mode.

Parameter

- value: Value to be set.

Class

[uc480.DeviceFeatureLog](#)

Accessible

Camera.Device.Feature.Log

Syntax

```
uc480.DeviceFeatureLog.Get(out uc480.Defines.LogMode mode)
```

Description

Returns the current Log mode.

Parameter

mode	uc480.Defines.LogMode.Factory: Default setting for the Log mode
	uc480.Defines.LogMode.Manual: Manual Log mode. In this case the classes LogManualGain and LogManualValue are used for further setting.
	uc480.Defines.LogMode.Off: Log mode off

Class[uc480.DeviceFeatureLog](#)**Accessible**

Camera.Device.Feature.Log

Syntax

```
uc480.DeviceFeatureLog.GetDefault(out uc480.Defines.LogMode mode)
```

Description

Returns the default settings for the Log mode.

Parameter

- mode: Default Log mode (see also [Get\(\)](#))

Class[uc480.DeviceFeatureLog](#)**Accessible**

Camera.Device.Feature.Log

Syntax

```
uc480.DeviceFeatureLog.GetSupported(out bool supported)
```

Description

Returns if the Log mode is supported.

Parameter

supported	0 = Not supported.
	1 = Supported.

Class[uc480.DeviceFeatureLog](#)**Accessible**

Camera.Device.Feature.Log

Syntax

```
uc480.DeviceFeatureLog.Set(uc480.Defines.LogMode mode)
```

Description

Sets the Log mode.

Parameter

mode	<code>uc480.Defines.LogMode.Factory</code> : Factory setting for the Log mode
	<code>uc480.Defines.LogMode.Manual</code> : Manual Log mode. In this case the classes <code>LogManualGain</code> and <code>LogManualValue</code> are used for further setting.
	<code>uc480.Defines.LogMode.Off</code> : Log mode off
	<code>uc480.Defines.LogMode.Auto</code> : Automatic Log mode (default)

The `MultiIntegration` class provides methods for controlling the multi integration mode of some uc480 cameras.



Note: This feature is currently only supported by the UI-325x model.

Methods

Method	Description
<code>GetMode</code>	Returns the current setting for multi integration mode.
<code>GetModeDefault</code>	Returns the default setting for multi integration mode.
<code>GetParams</code>	Returns the current parameter for multi integration mode.
<code>GetScope</code>	Returns the minimum and maximum ranges in multi integration mode.
<code>GetSupported</code>	Returns if the multi integration mode is supported.
<code>SetMode</code>	Sets the multi integration mode.
<code>SetParams</code>	Sets the parameter for multi integration mode.

Class

[`uc480.DeviceFeatureMultiIntegration`](#)

Accessible

`Camera.Device.Feature.MultiIntegration`

Syntax

`uc480.DeviceFeatureMultiIntegration.GetMode(out uc480.Defines.MultiIntegrationMode mode)`

Description

Returns the current setting for multi integration mode.

Parameter

- mode: current mode, see [SetMode \(\)](#)

Class

[uc480.DeviceFeatureMultiIntegration](#)

Accessible

Camera.Device.Feature.MultiIntegration

Syntax

```
uc480.DeviceFeatureMultiIntegration.GetModeDefault(out uc480.Defines.MultiIntegrationMode mode)
```

Description

Returns the default setting for multi integration mode.

Parameter

- mode: Returns the default mode

Class

[uc480.DeviceFeatureMultiIntegration](#)

Accessible

Camera.Device.Feature.MultiIntegration

Syntax

```
uc480.DeviceFeatureMultiIntegration.GetParams(out uc480.Types.MultiIntegrationCycles[] cycles)
```

Description

Returns the current parameter for multi integration mode.

Parameter

- cycles: current cycles for multi integration mode, see [MultiIntegrationCycles](#)

Class

[uc480.DeviceFeatureMultiIntegration](#)

Accessible

Camera.Device.Feature.MultiIntegration

Syntax

```
uc480.DeviceFeatureMultiIntegration.GetScope(out uc480.Types.MultiIntegrationScope scope)
```

Description

Returns the minimum and maximum ranges in multi integration mode.

Parameter

- scope: ranges for multi integration mode, see [uc480.Types.MultiIntegrationScope](#)

Class

[uc480.DeviceFeatureMultiIntegration](#)

Accessible

Camera.Device.Feature.MultiIntegration

Syntax

```
uc480.DeviceFeatureMultiIntegration.GetSupported(out bool supported)
```

Description

Returns if the multi integration mode is supported.

Parameter

supported	0 = Not supported 1 = Supported
-----------	------------------------------------

Class

[uc480.DeviceFeatureMultiIntegration](#)

Accessible

Camera.Device.Feature.MultiIntegration

Syntax

```
uc480.DeviceFeatureMultiIntegration.SetMode(uc480.Defines.MultiIntegrationMode mode)
```

Description

Sets the multi integration mode.

Parameter

mode	uc480.Defines.MultiIntegrationMode.Gp io1: Sets the multi integration mode on GPIO 1. The exposure sequence is generated via the GPIO 1 input by the user. In this mode, the parameters are not effective. uc480.Defines.MultiIntegrationMode.Gp io2: Sets the multi integration mode on GPIO 2. The exposure sequence is generated via the GPIO 2 input by the user. In this mode, the parameters are not effective. uc480.Defines.MultiIntegrationMode.So ftware: The multi integration mode is controlled by software. In this case, the exposure sequence is generated by the camera-internal pulse-width modulation (PWM). In this mode, the parameters are effective. uc480.Defines.MultiIntegrationMode.O f: Disables the multi integration mode
------	--

Class

[uc480.DeviceFeatureMultiIntegration](#)

Accessible

Camera.Device.Feature.MultiIntegration

Syntax

`uc480.DeviceFeatureMultiIntegration.SetParams(uc480.Types.MultiIntegrationCycles[] cycles)`

Description

Sets the parameter for multi integration mode.

Parameter

- `cycles`: cycles for multi integration mode, see [MultiIntegrationCycles](#)

The `NoiseReduction` class provides methods for setting the "Reduce noise" option.



This class is currently only supported by the XS model.

Methods

Method	Description
Get	Returns the current noise reduction mode.
GetDefault	Returns the default noise reduction mode.
GetSupported	Returns if noise reduction is supported.
Set	Sets the mode for noise reduction.

Class

[uc480.DeviceFeatureNoiseReduction](#)

Accessible

Camera.Device.Feature.NoiseReduction

Syntax

`uc480.DeviceFeatureNoiseReduction.Get(out uc480.Defines.NoiseReductionMode mode)`

Description

Returns the current noise reduction mode.

Parameter

<code>mode</code>	<code>uc480.Defines.NoiseReductionMode.Adaptive</code> : Noise reduction is enabled.
	<code>uc480.Defines.NoiseReductionMode.Off</code> : Noise reduction is disabled.

Class

[uc480.DeviceFeatureNoiseReduction](#)

Accessible

Camera.Device.Feature.NoiseReduction

Syntax

```
uc480.DeviceFeatureNoiseReduction.GetDefault(out uc480.Defines.NoiseReductionMode mode)
```

Description

Returns the default noise reduction mode.

Parameter

- `mode`: Default mode for noise reduction.

Class

[uc480.DeviceFeatureNoiseReduction](#)

Accessible

Camera.Device.Feature.NoiseReduction

Syntax

```
uc480.DeviceFeatureNoiseReduction.GetSupported(out bool supported)
```

Description

Returns if noise reduction is supported.

Parameter

supported	0 = Not supported. 1 = Supported.
-----------	--------------------------------------

Class

[uc480.DeviceFeatureNoiseReduction](#)

Accessible

Camera.Device.Feature.NoiseReduction

Syntax

```
uc480.DeviceFeatureNoiseReduction.Set(uc480.Defines.NoiseReductionMode mode)
```

Description

Sets the mode for noise reduction.

Parameter

mode	uc480.Defines.NoiseReductionMode.Adaptive: Noise reduction is enabled. uc480.Defines.NoiseReductionMode.Off: Noise reduction is disabled.
------	--

The `SensorBitDepth` class provides methods for setting the sensor bit depth. With the auto setting the software selects the appropriate sensor bit depth to the chosen image format. The bit depth can also be selected independent from the image format.



Attention: As you can choose combinations that does not fit, this function should be used by experts only. In most cases the auto control are sufficient.



Note: This feature is currently only supported by the models UI-336x/UI-536x and UI-337x/UI-537x.

Methods

Method	Description
<u>Get</u>	Returns the current mode for the sensor bit depth.
<u>GetDefault</u>	Returns the default mode for the sensor bit depth.
<u>GetSupported</u>	Returns if setting the sensor bit depth is supported.
<u>GetSupportedBitDepth</u>	Returns the supported modes.
<u>Set</u>	Sets the mode of the sensor bit depth.

Class

[uc480.DeviceFeatureSensorBitDepth](#)

Accessible

Camera.Device.Feature.SensorBitDepth

Syntax

```
uc480.DeviceFeatureSensorBitDepth.Get(out uc480.Defines.SensorBitDepth mode)
```

Description

Returns the current mode for the sensor bit depth.

Parameter

- `mode`: Returns the default mode of the sensor bit depth (see [GetSupportedBitDepth\(\)](#))

Class

[uc480.DeviceFeatureSensorBitDepth](#)

Accessible

Camera.Device.Feature.SensorBitDepth

Syntax

```
uc480.DeviceFeatureSensorBitDepth.GetDefault(out uc480.Defines.SensorBitDepth mode)
```

Description

Returns the default mode for the sensor bit depth.

Parameter

- `mode`: Returns the default mode of the sensor bit depth (auto)

Class[uc480.DeviceFeatureSensorBitDepth](#)**Accessible**

Camera.Device.Feature.SensorBitDepth

Syntax

uc480.DeviceFeatureSensorBitDepth.GetSupported(out bool supported)

Description

Returns if setting the sensor bit depth is supported.

Parameter

<code>supported</code>	0 = Not supported. 1 = Supported.
------------------------	--------------------------------------

Class[uc480.DeviceFeatureSensorBitDepth](#)**Accessible**

Camera.Device.Feature.SensorBitDepth

Syntax

uc480.DeviceFeatureSensorBitDepth.GetSupportedBitDepth(out uc480.Defines.SensorBitDepth mode)

Description

Returns the supported modes.

Parameter

<code>mode</code>	<code>uc480.Defines.SensorBitDepth.Auto</code> : Sets the sensor bit depth to auto
	<code>uc480.Defines.SensorBitDepth.Bit12</code> : Sets the sensor bit depth to 12 bit
	<code>uc480.Defines.SensorBitDepth.Bit10</code> : Sets the sensor bit depth to 10 bit

Class[uc480.DeviceFeatureSensorBitDepth](#)**Accessible**

Camera.Device.Feature.SensorBitDepth

Syntax

uc480.DeviceFeatureSensorBitDepth.Set(uc480.Defines.SensorBitDepth mode)

Description

Sets the mode of the sensor bit depth.

Parameter

- mode: Mode to be set (see [GetSupportedBitDepth\(\)](#))

The `SensorSourceGain` class provides methods for setting the sensor source gain (analog master gain).



Note: This feature is currently only supported by the models UI-336x/UI-536x and UI-337x/UI-537x.

Methods

Method	Description
Get	Returns the current sensor source gain.
GetDefault	Returns the default sensor source gain.
GetRange	Returns the range for the sensor source gain.
GetSupported	Returns if setting the sensor source gain is supported.
Set	Sets the sensor source gain.

Class

[uc480.DeviceFeatureSensorSourceGain](#)

Accessible

`Camera.Device.Feature.SensorSourceGain`

Syntax

```
uc480.DeviceFeatureSensorSourceGain.Get(out int gain)
```

Description

Returns the current sensor source gain.

Parameter

- gain: Returns the value of the sensor source gain.

Class

[uc480.DeviceFeatureSensorSourceGain](#)

Accessible

`Camera.Device.Feature.SensorSourceGain`

Syntax

```
uc480.DeviceFeatureSensorSourceGain.GetDefault(out int gain)
```

Description

Returns the default sensor source gain.

Parameter

- gain: Returns the default value of the sensor source gain.

Class

[uc480.DeviceFeatureSensorSourceGain](#)

Accessible

Camera.Device.Feature.SensorSourceGain

Syntax

```
uc480.DeviceFeatureSensorSourceGain.GetRange(out uc480.Types.Range<int> range)
uc480.DeviceFeatureSensorSourceGain.GetRange(out int min, out int max, out int inc)
```

Description

Returns the range for the sensor source gain.

Parameter

range	Minimum: returns the minimum value Maximum: returns the maximum value Increment: returns the increment
min	Returns the minimum value.
max	Returns the maximum value.
inc	Returns the increment.

Class

[uc480.DeviceFeatureSensorSourceGain](#)

Accessible

Camera.Device.Feature.SensorSourceGain

Syntax

```
uc480.DeviceFeatureSensorSourceGain.GetSupported(out bool supported)
```

Description

Returns if setting the sensor source gain is supported.

Parameter

supported	0 = Not supported. 1 = Supported.
-----------	--------------------------------------

Class

[uc480.DeviceFeatureSensorSourceGain](#)

Accessible

Camera.Device.Feature.SensorSourceGain

Syntax

```
uc480.DeviceFeatureSensorSourceGain.Set(int gain)
```

Description

Sets the sensor source gain.

Parameter

- gain: Value to be set.

The `ShutterMode` class provides methods for configuring special camera functions provided by specific uc480 models. The class does not enable the global shutter.



The global start shutter function is only supported by the sensors UI-148x/UI-348x/UI-548x and UI-158x/UI-358x/UI-558x.



The global start shutter function is only supported in trigger mode (see also [Trigger](#)).

Methods

Method	Description
Get	Returns the shutter mode.
GetSupported	Returns the supported shutter modes.
IsSupported	Returns if the passed shutter mode is supported.
Set	Sets the shutter mode.

Class

[uc480.DeviceFeatureShutter](#)

Accessible

Camera.Device.Feature.ShutterMode

Syntax

```
uc480.DeviceFeatureShutter.Get(out uc480.Defines.Shuttermode mode)
```

Description

Returns the shutter mode.

Parameter

- mode: Returns the current shutter mode (see [GetSupported\(\)](#)).

Class

[uc480.DeviceFeatureShutter](#)

Accessible

Camera.Device.Feature.ShutterMode

Syntax

```
uc480.DeviceFeatureShutter.GetSupported(out uc480.Defines.Shuttermode mode)
```

Description

Returns the supported shutter modes.

Parameter

mode	<p>uc480.Defines.Shuttermode.Rolling: Rolling shutter mode</p> <p>uc480.Defines.Shuttermode.RollingGlobalStart: Rolling shutter with global start</p> <p>uc480.Defines.Shuttermode.Global: Global shutter mode</p> <p>uc480.Defines.Shuttermode.GlobalAlternativeTiming: Global shutter mode with alternative timing (UI-124x/UI-324x/UI-524x only)</p>
------	---

Class

[uc480.DeviceFeatureShutter](#)

Accessible

Camera.Device.Feature.ShutterMode

Syntax

uc480.DeviceFeatureShutter.Issupported(uc480.Defines.Shuttermode mode)

Description

Returns if the passed shutter mode is supported.

Parameter

- mode: Shutter mode to be queried to be supported (see [GetSupported\(\)](#)).

Class

[uc480.DeviceFeatureShutter](#)

Accessible

Camera.Device.Feature.ShutterMode

Syntax

uc480.DeviceFeatureShutter.Set(uc480.Defines.Shuttermode mode)

Description

Sets the shutter mode.

Parameter

- mode: shutter mode to be set (see [GetSupported\(\)](#))

The `Temperature` class provides methods for getting the internal camera temperature of some uc480 models.

Methods

Method	Description
<u>Get</u>	Returns the internal camera temperature (XS only).
<u>GetNumerical</u>	Returns the content of the sensor register. This value is not the absolute temperature (UI-336x/UI-536x and UI-337x/UI-537x only).
<u>GetSupported</u>	Returns if the display of the internal camera temperature is supported.



This class is currently only supported by the XS model.

Class

[uc480.DeviceFeatureTemperature](#)

Accessible

Camera.Device.Feature.Temperature

Syntax

```
uc480.DeviceFeatureTemperature.Get(out int temperature)
```

Description

Returns the internal camera temperature. This is always above the ambient temperature.

Parameter

- `temperature`: Returns the internal camera temperature.



Note: This feature is currently only supported by the models UI-336x/UI-536x and UI-337x/UI-537x.

Class

[uc480.DeviceFeatureTemperature](#)

Accessible

Camera.Device.Feature.Temperature

Syntax

```
uc480.DeviceFeatureTemperature.GetNumerical(out int numerical)
```

Description

Returns the content of the sensor register. This value is not the absolute temperature.

Parameter

- `numerical`: Returns the content of the sensor register.

Class

[uc480.DeviceFeatureTemperature](#)

Accessible

Camera.Device.Feature.Temperature

Syntax

```
uc480.DeviceFeatureTemperature.GetSupported(out bool supported)
```

Description

Returns if the display of the internal camera temperature is supported.

Parameter

supported	0 = Internal camera temperature is not supported. 1 = Internal camera temperature is supported.
-----------	--

The `Timestamp` class provides methods for configuring the time stamp of the camera.



Note: This function is only supported by the camera families GigE uc480 LE and GigE uc480 CP.

Methods

Method	Description
Get	Returns the current time stamp configuration.
GetSupported	Returns if setting the time stamp is supported.
Reset	Resets the time stamp.
Set	Sets the time stamp configuration.

Class

[uc480.DeviceFeatureTimestamp](#)

Accessible

Camera.Device.Feature.Timestamp

Syntax

```
uc480.DeviceFeatureTimestamp.Get(out uc480.Types.TimestampConfiguration timestampConfiguration)
uc480.DeviceFeatureTimestamp.Get(out uc480.Defines.TimestampMode mode, out uc480.Defines.IO.Pin pin, ou
```

Description

Returns the current time stamp configuration.

Parameter

timestampConfiguration	<p>Mode: Mode for the time stamp configuration</p> <p>Pin: Pin which is used for time stamp event.</p> <p>Edge: Signal on the rising/falling edge which is used for time stamp event</p>
mode	Mode for the time stamp configuration: <ul style="list-style-type: none">• uc480.Defines.TimestampMode.Once: Resets once the time stamp of the camera to 0.
pin	Pin which is used for time stamp event: <ul style="list-style-type: none">• uc480.Defines.IO.Pin.None: No time stamp event.• uc480.Defines.IO.Pin.Trigger: A signal on the trigger pin fires the time stamp event.• uc480.Defines.IO.Pin.Gpio0: A signal on the GPIO 1 fires the time stamp event.• uc480.Defines.IO.Pin.Gpio1: A signal on the GPIO 2 fires the time stamp event.
edge	Signal on the rising/falling edge which is used for time stamp event (when uc480.Defines.IO.Pin.Trigger is used): <ul style="list-style-type: none">• uc480.Defines.IO.Edge.Falling: The falling edge signal fires the time stamp event.• uc480.Defines.IO.Edge.Rising: The rising edge signal fires the time stamp event.

Class[uc480.DeviceFeatureTimestamp](#)**Accessible**

Camera.Device.Feature.Timestamp

Syntax`uc480.DeviceFeatureTimestamp.GetSupported(out bool supported)`**Description**

Returns if setting the time stamp is supported.

Parameter

supported	0 = Not supported. 1 = Supported.
-----------	--------------------------------------

Class

[uc480.DeviceFeatureTimestamp](#)

Accessible

Camera.Device.Feature.Timestamp

Syntax

```
uc480.DeviceFeatureTimestamp.Reset(uc480.Defines.IO.Pin pin, uc480.Defines.IO.Edge edge)
```

Description

Resets the time stamp.

Parameter

pin	Pin which is used for time stamp event: <ul style="list-style-type: none"> • uc480.Defines.IO.Pin.None: No time stamp event. • uc480.Defines.IO.Pin.Trigger: A signal on the trigger pin fires the time stamp event. • uc480.Defines.IO.Pin.Gpio0: A signal on the GPIO 1 fires the time stamp event. • uc480.Defines.IO.Pin.Gpio1: A signal on the GPIO 2 fires the time stamp event.
edge	Signal on the rising/falling edge which is used for time stamp event (when uc480.Defines.IO.Pin.Trigger is used): <ul style="list-style-type: none"> • uc480.Defines.IO.Edge.Falling: The falling edge signal fires the time stamp event. • uc480.Defines.IO.Edge.Rising: The rising edge signal fires the time stamp event.

Class

[uc480.DeviceFeatureTimestamp](#)

Accessible

Camera.Device.Feature.Timestamp

Syntax

```
uc480.DeviceFeatureTimestamp.Set(uc480.Types.TimestampConfiguration timestampConfiguration)
uc480.DeviceFeatureTimestamp.Set(uc480.Defines.TimestampMode mode, uc480.Defines.IO.Pin pin, uc480.Defi
```

Description

Sets the time stamp configuration.

Parameter

timestampConfiguration	<p>Mode: Mode for the time stamp configuration</p> <p>Pin: Pin which is used for time stamp event.</p> <p>Edge: Signal on the rising/falling edge which is used for time stamp event</p>
mode	<p>Mode for the time stamp configuration:</p> <ul style="list-style-type: none"> • uc480.Defines.TimestampMode.Once: Resets once the time stamp of the camera to 0.
pin	<p>Pin which is used for time stamp event:</p> <ul style="list-style-type: none"> • uc480.Defines.IO.Pin.None: No time stamp event. • uc480.Defines.IO.Pin.Trigger: A signal on the trigger pin fires the time stamp event. • uc480.Defines.IO.Pin.Gpio0: A signal on the GPIO 1 fires the time stamp event. • uc480.Defines.IO.Pin.Gpio1: A signal on the GPIO 2 fires the time stamp event.
edge	<p>Signal on the rising/falling edge which is used for time stamp event (when uc480.Defines.IO.Pin.Trigger is used):</p> <ul style="list-style-type: none"> • uc480.Defines.IO.Edge.Falling: The falling edge signal fires the time stamp event. • uc480.Defines.IO.Edge.Rising: The rising edge signal fires the time stamp event.

The `XSHSMode` class provides methods for configuring special camera functions provided by specific uc480 models:

- On UI-1008XS model: Choose the HS mode for triggered image capture.

Methods

Method	Description
<u>Get</u>	Returns if the HS mode is enabled.
<u>GetDefault</u>	Returns the default setting for the HS mode
<u>GetSupported</u>	Returns the supported HS modes.
<u>IsSupported</u>	Returns if the passed HS mode is supported.
<u>Set</u>	Sets the HS mode on the UI-1008XS.

Class

[uc480.DeviceFeatureXSHSMode](#)

Accessible

Camera.Device.Feature.XSHSMode

Syntax

```
uc480.DeviceFeatureXSHSMode.Get(out uc480.Defines.XSMode mode)
```

Description

Returns if the HS mode is enabled.

Parameter

- mode : Returns the current HS mode.

Class

[uc480.DeviceFeatureXSHSMode](#)

Accessible

Camera.Device.Feature.XSHSMode

Syntax

```
uc480.DeviceFeatureXSHSMode.GetDefault(out uc480.Defines.XSMode mode)
```

Description

Returns the default setting for the HS mode

Parameter

- mode : Returns the default HS mode.

Class

[uc480.DeviceFeatureXSHSMode](#)

Accessible

Camera.Device.Feature.XSHSMode

Syntax

```
uc480.DeviceFeatureXSHSMode.GetSupported(out uc480.Defines.XSMode mode)
```

Description

Returns the supported HS modes.

Parameter

mode	uc480.Defines.XSMode.Hs : HS mode
------	-----------------------------------

Class

[uc480.DeviceFeatureXSHSMode](#)

Accessible

Camera.Device.Feature.XSHSMode

Syntax

```
uc480.DeviceFeatureXSHSMode.IsSupported(uc480.Defines.XSMode mode)
```

Description

Returns if the passed HS mode is supported.

Parameter

- mode: HS mode to be queried to be supported (see [GetSupported\(\)](#)).

Class

[uc480.DeviceFeatureXSHSMode](#)

Accessible

Camera.Device.Feature.XSHSMode

Syntax

```
uc480.DeviceFeatureXSHSMode.Set(uc480.Defines.XSMode mode)
```

Description

Sets the HS mode on the UI-1008XS.

Parameter

mode	uc480.Defines.XSMode.Hs: HS mode
------	----------------------------------

3.1.1.5.2 GetCameraID

Class

[uc480.Device](#)

Accessible

Camera.Device

Syntax

```
uc480.Device.GetCameraID(out int s32Value)
```

Description

Returns the current camera ID.

Parameter

- s32Value: Returns the current ID.

3.1.1.5.3 GetDeviceID

Class

[uc480.Device](#)

Accessible

Camera.Device

Syntax

```
uc480.Device.GetDeviceID(out int s32Value)
```

Description

Returns the current device ID.

Parameter

- `s32Value`: Returns the device ID.

3.1.1.5.4 GetEnableAutoExit**Class**

[uc480.Device](#)

Accessible

Camera.Device

Syntax

```
uc480.Device.GetEnableAutoExit(out bool bEnable)
```

Description

Returns the state of the automatic closing of the camera handle.

Parameter

<code>bEnable</code>	1 = Automatic closing is enabled
	0 = Automatic closing is disabled

3.1.1.5.5 SetCameraID**Class**

[uc480.Device](#)

Accessible

Camera.Device

Syntax

```
uc480.Device.SetCameraID(int s32Value)
```

Description

Sets a unique camera ID to a camera. Thus, it is possible to access the camera directly with [Init\(\)](#).

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 IDS Camera Manager.

Parameter

- `s32Value`: New camera ID (1...254)

3.1.1.5.6 SetEnableAutoExit**Class**

[uc480.Device](#)

Accessible

Camera.Device

Syntax

```
uc480.Device.SetEnableAutoExit(bool bEnable)
```

Description

Enables/disables 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.

Parameter

bEnable	1 = Enables automatic closing 0 = Disables automatic closing
---------	---

3.1.1.6 DirectRenderer

The `DirectRenderer` class 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 OpenGL library are supported under Windows and Linux. The Direct3D library are supported only under Windows.



Note on system requirements

- 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 [GetMaxSize\(\)](#).
- 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 in the uc480 manual.



Note on displaying monochrome or raw data formats

To display monochrome or Bayer raw data in Direct3D, please set the appropriate display mode constants using [set\(\)](#).

The following class and methods exist:

- [Overlay](#)

Methods

Method	Description
DisableScaling	Disables real-time scaling.
EnableImageScaling	Enables real-time scaling of the image to the size of the display window. The overlay is not scaled.
EnableScaling	Enables real-time scaling of the image to the size of the display window. The overlay is scaled together with the camera image.
GetStealFormat	Returns the color format setting for the steal function.
GetSupported	Returns the supported display modes.
 GetUserSyncPosRange	Returns the minimum and maximum row position for SetUserSync() .
SetScaling	Enables/disables real-time scaling of the image to the size of the display window.
SetStealFormat	Defines the color format for the steal function.
SetUserSync	Enables synchronization of the image display with a monitor pixel row specified by the user.
SetWindowHandle	Sets a new window handle for image output in Direct3D.
StealNextFrame	Copies the next image to the active user memory (steal function).
Update	Redraws the current image and overlay.
VsyncAuto	Enables synchronization of the image display with the monitor's image rendering.
VsyncOff	Disables image display synchronization.

3.1.1.6.1 Overlay

The `Overlay` class provides methods for controlling the overlay data of the camera's live image without flicker. The graphics card functions of the OpenGL library are supported under Windows and Linux. The Direct3D library are supported only under Windows.

Note on system requirements

- 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 [GetMaxSize\(\)](#).
- 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 in the uc480 manual.



Note on displaying monochrome or raw data formats

To display monochrome or Bayer raw data in Direct3D, please set the appropriate display mode constants using [Set\(\)](#).



Methods

Method	Description
Clear	Deletes the data of the overlay area by filling it with black color.
DisableSemiTransparent	Disables the semi-transparent display of the overlay area.
EnableSemiTransparent	Enables a semi-transparent display of the overlay area.
GetDC	Returns the device context (DC) handle to the overlay area of the graphics card.
GetGraphics	Returns the overlay graphics.
GetKeyColor	Returns the RGB values of the current key color (default: black).
GetMaxSize	Returns the width and height of the maximum overlay area supported by the graphics card.
GetSize	Returns the size of the overlay area.
Hide	Disables overlay display.
LoadImage	Loads a bitmap image (*.BMP file) into the overlay area.
ReleaseDC	Releases the device context (DC) handle and updates the overlay data.
SetGraphics	Releases the device context (DC) handle.
SetColor	Defines the RGB values of the key color.
SetPosition	Defines the position of the overlay area.
SetSemiTransparent	Enables/disabled the semi-transparent display of the overlay area.
SetSize	Defines the size of the overlay area (default: current camera image size).
SetVisible	Sets the overlay to visible.
Show	Enables overlay display on top of the current camera image.

Class

[uc480.DirectRendererOverlay](#)

Accessible

Camera.DirectRenderer.Overlay

Syntax

`uc480.DirectRendererOverlay.Clear()`

Description

Deletes the data of the overlay area by filling it with black color.

Parameter

None

Class

[uc480.DirectRendererOverlay](#)

Accessible

Camera.DirectRenderer.Overlay

Syntax

`uc480.DirectRendererOverlay.DisableSemiTransparent()`

Description

Disables the semi-transparent display of the overlay area.

Parameter

None

Class

[uc480.DirectRendererOverlay](#)

Accessible

Camera.DirectRenderer.Overlay

Syntax

`uc480.DirectRendererOverlay.EnableSemiTransparent()`

Description

Enables a semi-transparent display of the overlay area. 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!

Parameter

None

Class

[uc480.DirectRendererOverlay](#)

Accessible

Camera.DirectRenderer.Overlay

Syntax

`uc480.DirectRendererOverlay.GetDC(out int hDC)`

Description

Returns the device context (DC) handle to the overlay area of the graphics card.

In Direct3D mode, the method 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 [ReleaseDC\(\)](#).

Parameter

- `hDC`: Device context handle

Class[uc480.DirectRendererOverlay](#)**Accessible**

Camera.DirectRenderer.Overlay

Syntax`uc480.DirectRendererOverlay.GetGraphics(out System.Drawing.Graphics graphics)`**Description**

Returns the overlay graphics.

Parameter

- `graphics`: Overlay data

Class[uc480.DirectRendererOverlay](#)**Accessible**

Camera.DirectRenderer.Overlay

Syntax`uc480.DirectRendererOverlay.GetKeyColor(out int s32Red, out int s32Green, out int s32Blue)`
`uc480.DirectRendererOverlay.GetKeyColor(out System.Drawing.Color color)`**Description**

Returns the RGB values of the current key color (default: black).

Parameter

<code>s32Red</code>	Returns the red value
<code>s32Green</code>	Returns the green value
<code>s32Blue</code>	Returns the blue value

Class[uc480.DirectRendererOverlay](#)**Accessible**

Camera.DirectRenderer.Overlay

Syntax`uc480.DirectRendererOverlay.GetMaxSize(out uc480.Types.Size<uint> size)`
`uc480.DirectRendererOverlay.GetMaxSize(out uint u32MaxWidth, out uint u32MaxHeight)`**Description**

Returns the width and height of the maximum overlay area supported by the graphics card.

Parameter

size	Width: Maximum width of the overlay area Height: Maximum height of the overlay area
u32MaxWidth	Maximum width of the overlay area
u32MaxHeight	Maximum height of the overlay area

Class[uc480.DirectRendererOverlay](#)**Accessible**

Camera.DirectRenderer.Overlay

Syntax

```
uc480.DirectRendererOverlay.GetSize(out uc480.Types.Size<uint> size)
uc480.DirectRendererOverlay.GetSize(out uint u32Width, out uint u32Height)
```

Description

Returns the size of the overlay area.

Parameter

size	Width: Returns the width of the overlay area Height: Returns the height of the overlay area
u32Width	Returns the width of the overlay area
u32Height	Returns the height of the overlay area

Class[uc480.DirectRendererOverlay](#)**Accessible**

Camera.DirectRenderer.Overlay

Syntax

```
uc480.DirectRendererOverlay.Hide()
```

Description

Disables overlay display.

Parameter

None

Class[uc480.DirectRendererOverlay](#)**Accessible**

Camera.DirectRenderer.Overlay

Syntax

```
uc480.DirectRendererOverlay.LoadImage(string strFilename)
```

Description

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.

Parameter

- strFilename: Bitmap file to be loaded

Class

[uc480.DirectRendererOverlay](#)

Accessible

Camera.DirectRenderer.Overlay

Syntax

```
uc480.DirectRendererOverlay.ReleaseDC()
```

Description

Releases the device context (DC) handle and updates the overlay data.

Parameter

None

Class

[uc480.DirectRendererOverlay](#)

Accessible

Camera.DirectRenderer.Overlay

Syntax

```
uc480.DirectRendererOverlay.SetGraphics(ref System.Drawing.Graphics graphics)
```

Description

Releases the device context (DC) handle.

Parameter

- graphics: Overlay data

Class

[uc480.DirectRendererOverlay](#)

Accessible

Camera.DirectRenderer.Overlay

Syntax

```
uc480.DirectRendererOverlay.SetKeyColor(int s32Red, int s32Green, int s32Blue)  
uc480.DirectRendererOverlay.SetKeyColor(System.Drawing.Color color)
```

Description

Defines the RGB values of the key color.

Parameter

s32Red	Red value to be set
s32Green	Green value to be set
s32Blue	Blue value to be set

Class

[uc480.DirectRendererOverlay](#)

Accessible

Camera.DirectRenderer.Overlay

Syntax

```
uc480.DirectRendererOverlay.SetPosition(uc480.Types.Point<uint> position)
uc480.DirectRendererOverlay.SetPosition(uint u32PosX, uint u32PosY)
```

Description

Defines the position of the overlay area.

Parameter

position	x: X value of the left upper corner of the overlay area y: Y value of the left upper corner of the overlay area
u32PosX	X value of the left upper corner of the overlay area
u32PosY	Y value of the left upper corner of the overlay area

Class

[uc480.DirectRendererOverlay](#)

Accessible

Camera.DirectRenderer.Overlay

Syntax

```
uc480.DirectRendererOverlay.SetSemiTransparent(bool bEnable)
```

Description

Enables/disabled the semi-transparent display of the overlay area.

Parameter

bEnable	1 = Enables semi-transparent display. 0 = Disables semi-transparent display.
---------	---

Class

[uc480.DirectRendererOverlay](#)

Accessible

Camera.DirectRenderer.Overlay

Syntax

```
uc480.DirectRendererOverlay.SetSize(uc480.Types.Size<uint> size)
uc480.DirectRendererOverlay.SetSize(uint u32Width, uint u32Height)
```

Description

Defines the size of the overlay area (default: current camera image size).

Parameter

size	Width: Width of the overlay area Height: Height of the overlay area
u32Width	Width of the overlay area
u32Height	Height of the overlay area

Class

[uc480.DirectRendererOverlay](#)

Accessible

Camera.DirectRenderer.Overlay

Syntax

`uc480.DirectRendererOverlay.SetVisible(bool bVisible)`

Description

Sets the overlay to visible.

Parameter

bVisible	1 = Set to visible. 0 = Set to invisible.
----------	--

Class

[uc480.DirectRendererOverlay](#)

Accessible

Camera.DirectRenderer.Overlay

Syntax

`uc480.DirectRendererOverlay.Show()`

Description

Enables overlay display on top of the current camera image.

Parameter

None

3.1.1.6.2 DisableScaling

Class

[uc480.DirectRenderer](#)

Accessible

Camera.DirectRenderer

Syntax

`uc480.DirectRenderer.DisableScaling()`

Description

Disables real-time scaling.

Parameter

None

3.1.1.6.3 EnableImageScaling**Class**

[uc480.DirectRenderer](#)

Accessible

Camera.DirectRenderer

Syntax

```
uc480.DirectRenderer.EnableImageScaling()
```

Description

Enables real-time scaling of the image to the size of the display window. The overlay is not scaled.

Parameter

None

3.1.1.6.4 EnableScaling**Class**

[uc480.DirectRenderer](#)

Accessible

Camera.DirectRenderer

Syntax

```
uc480.DirectRenderer.EnableScaling()
```

Description

Enables real-time scaling of the image to the size of the display window. The overlay is scaled together with the camera image.

Parameter

None

3.1.1.6.5 GetStealFormat**Class**

[uc480.DirectRenderer](#)

Accessible

Camera.DirectRenderer

Syntax

```
uc480.DirectRenderer.GetStealFormat(out uc480.Defines.ColorMode mode)
```

Description

Returns the color format setting for the steal function.

Parameter

- mode: Returns the set color format

3.1.1.6.6 GetSupported

Class

[uc480.DirectRenderer](#)

Accessible

Camera.DirectRenderer

Syntax

`uc480.DirectRenderer.GetSupported(out uc480.Defines.DisplayMode mode)`

Description

Returns the supported display modes.

Parameter

- mode: Supported display modes (see also [Set\(\)](#))

3.1.1.6.7 GetUserSyncPosRange

Class

[uc480.DirectRenderer](#)

Accessible

Camera.DirectRenderer

Syntax

`uc480.DirectRenderer.GetUserSyncPosRange(out uint u32PosMin, out uint u32PosMax)`

Description

Returns the minimum and maximum row position for [SetUserSync\(\)](#).

Parameter

u32PosMin	Minimum row position
u32PosMax	Maximum row position

3.1.1.6.8 SetScaling

Class

[uc480.DirectRenderer](#)

Accessible

Camera.DirectRenderer

Syntax

`uc480.DirectRenderer.SetScaling(bool bEnable)`

Description

Enables/disables real-time scaling of the image to the size of the display window. The overlay is scaled together with the camera image (see also [EnableScaling\(\)](#)).

Parameter

bEnable	1 = Enable scaling 0 = Disable scaling
---------	---

3.1.1.6.9 SetStealFormat**Class**[uc480.DirectRenderer](#)**Accessible**

Camera.DirectRenderer

Syntax`uc480.DirectRenderer.SetStealFormat(uc480.Defines.ColorMode mode)`**Description**

Defines the color format for the steal function. For a list of all available color formats, see the description for the pixel format ([set\(\)](#)). The default is `uc480.Defines.ColorMode.BGRA8Packed` (RGB 32).

Parameter

- `mode`: Color format to be set

3.1.1.6.10 SetUserSync**Class**[uc480.DirectRenderer](#)**Accessible**

Camera.DirectRenderer

Syntax`uc480.DirectRenderer.SetUserSync(uint u32SyncPosition)`**Description**

Enables synchronization of the image display with a monitor pixel row specified by the user. 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.

Parameter

- `u32SyncPosition`: Number of image row

3.1.1.6.11 SetWindowHandle**Class**[uc480.DirectRenderer](#)**Accessible**

Camera.DirectRenderer

Syntax`uc480.DirectRenderer.SetWindowHandle(int hWnd)`

Description

Sets a new window handle for image output in Direct3D.

Parameter

- hWnd: Window handle

3.1.1.6.12 StealNextFrame**Class**

[uc480.DirectRenderer](#)

Accessible

Camera.DirectRenderer

Syntax

```
uc480.DirectRenderer.StealNextFrame(uc480.Defines.DeviceParameter mode)  
uc480.DirectRenderer.StealNextFrame(int s32Wait)
```

Description

Copies the next image to the active user [memory](#) (steal function).

Parameter

mode/s32Wait	uc480.Defines.DeviceParameter.Wait: The method waits until the image save is complete. uc480.Defines.DeviceParameter.DontWait: The method returns immediately.
--------------	---

3.1.1.6.13 Update**Class**

[uc480.DirectRenderer](#)

Accessible

Camera.DirectRenderer

Syntax

```
uc480.DirectRenderer.Update()
```

Description

Redraws the current image and overlay.

Parameter

None

3.1.1.6.14 VsyncAuto**Class**

[uc480.DirectRenderer](#)

Accessible

Camera.DirectRenderer

Syntax

```
uc480.DirectRenderer.VsyncAuto()
```

Description

Enables synchronization of the image display with the monitor's image rendering. The image is displayed upon the monitor's next VSYNC signal.

Parameter

None

3.1.1.6.15 VsyncOff

Class

[uc480.DirectRenderer](#)

Accessible

Camera.DirectRenderer

Syntax

```
uc480.DirectRenderer.VsyncOff()
```

Description

Disables image display synchronization. The image is displayed immediately.

Parameter

None

3.1.1.7 Display

The `Display` class provides classes for displaying and rendering images on the screen. The following classes and method exist:

- [AutoRender](#)
- [Mode](#)
- [Position](#)

Methods

Method	Description
Render	Outputs an image from an image memory in the specified window.

3.1.1.7.1 AutoRender

The `AutoRender` class provides methods for rendering an image automatically after it is transferred.

Methods

Method	Description
GetEnable	Returns if the auto rendering is enabled.
GetMode	Returns the current render mode.
GetModeDefault	Returns the default render mode.
GetWindow	Returns a pointer to a window handle.
SetEnable	Enables/disables the auto render.
SetMode	Sets the render mode.
SetWindow	Sets the window handle.

Class

[uc480.DisplayAutoRender](#)

Accessible

Camera.Display.AutoRender

Syntax

`uc480.DisplayAutoRender.GetEnable(out bool enable)`

Description

Returns if the auto rendering is enabled.

Parameter

<code>enable</code>	1 = Auto render is enabled. 0 = Auto render is disabled.
---------------------	---

Class

[uc480.DisplayAutoRender](#)

Accessible

Camera.Display.AutoRender

Syntax

`uc480.DisplayAutoRender.GetMode(out uc480.Defines.DisplayRenderMode mode)`

Description

Returns the current render mode.

Parameter

mode	<p>uc480.Defines.DisplayRenderMode.Normal: The image is rendered normally. It will be displayed in 1:1 scale as stored in the image memory.</p> <p>uc480.Defines.DisplayRenderMode.FitToWindow: The image size is adjusted to fit the output window.</p> <p>uc480.Defines.DisplayRenderMode.DownScale_1_2: Displays the image at 50 % of its original size.</p> <p>uc480.Defines.DisplayRenderMode.PlanarColorBlue: Displays only the blue channel of planar color format.</p> <p>uc480.Defines.DisplayRenderMode.PlanarColorGreen: Displays only the green channel of planar color format.</p> <p>uc480.Defines.DisplayRenderMode.PlanarColorRed: Displays only the red channel of planar color format.</p> <p>uc480.Defines.DisplayRenderMode.PlanarMonoBlue: Displays only the blue channel as monochrome image.</p> <p>uc480.Defines.DisplayRenderMode.PlanarMonoGreen: Displays only the green channel as monochrome image.</p> <p>uc480.Defines.DisplayRenderMode.PlanarMonoRed: Displays only the red channel as monochrome image.</p> <p>uc480.Defines.DisplayRenderMode.MirrorUpDown: Mirrors the displayed image along the horizontal axis.</p>
------	---

Class

[uc480.DisplayAutoRender](#)

Accessible

Camera.Display.AutoRender

Syntax

```
uc480.DisplayAutoRender.GetModeDefault(out uc480.Defines.DisplayRenderMode mode)
```

Description

Returns the default render mode.

Parameter

- mode: Default render mode

Class

[uc480.DisplayAutoRender](#)

Accessible

Camera.Display.AutoRender

Syntax

`uc480.DisplayAutoRender.GetWindow(out System.IntPtr handle)`

Description

Returns a pointer to a window handle.

Parameter

- `handle`: Window handle

Class

[uc480.DisplayAutoRender](#)

Accessible

Camera.Display.AutoRender

Syntax

`uc480.DisplayAutoRender.SetEnable(bool enable)`

Description

Enables/disables the auto render.

Parameter

<code>enable</code>	<code>1</code> = Enable auto render. <code>0</code> = Disable auto render.
---------------------	---

Class

[uc480.DisplayAutoRender](#)

Accessible

Camera.Display.AutoRender

Syntax

`uc480.DisplayAutoRender.SetMode(uc480.Defines.DisplayRenderMode mode)`

Description

Sets the render mode.

Parameter

mode	<p>uc480.Defines.DisplayRenderMode.Normal: The image is rendered normally. It will be displayed in 1:1 scale as stored in the image memory.</p> <p>uc480.Defines.DisplayRenderMode.FitToWindow: The image size is adjusted to fit the output window.</p> <p>uc480.Defines.DisplayRenderMode.DownScale_1_2: Displays the image at 50 % of its original size.</p> <p>uc480.Defines.DisplayRenderMode.PlanarColorBlue: Displays only the blue channel of planar color format.</p> <p>uc480.Defines.DisplayRenderMode.PlanarColorGreen: Displays only the green channel of planar color format.</p> <p>uc480.Defines.DisplayRenderMode.PlanarColorRed: Displays only the red channel of planar color format.</p> <p>uc480.Defines.DisplayRenderMode.PlanarMonoBlue: Displays only the blue channel as monochrome image.</p> <p>uc480.Defines.DisplayRenderMode.PlanarMonoGreen: Displays only the green channel as monochrome image.</p> <p>uc480.Defines.DisplayRenderMode.PlanarMonoRed: Displays only the red channel as monochrome image.</p> <p>uc480.Defines.DisplayRenderMode.MirrorUpDown: Mirrors the displayed image along the horizontal axis.</p>
------	---

Class[uc480.DisplayAutoRender](#)**Accessible**

Camera.Display.AutoRender

Syntax

uc480.DisplayAutoRender.SetWindow(System.IntPtr handle)

Description

Sets the window handle.

Parameter

- **handle:** Pointer to be set

3.1.1.7.2 Mode

The `Mode` class provides methods for setting the way in which images will be displayed on the screen.

For live videos including overlays, you can use the Direct3D or OpenGL mode. This mode is not supported by all graphics cards. The graphics card must have sufficient extended memory because Direct3D or OpenGL mode requires additional memory up to the size needed for the current screen resolution.

For further information on the display modes of the uc480 camera, see the uc480 manual.



The Direct3D display mode is not available on Linux operating systems.



We recommend that you call the following methods exclusively from a single thread in order to avoid unpredictable behavior of the application.

- [Init\(\)](#)
- `DisplayMode`
- [Exit\(\)](#)

Methods

Method	Description
Get	Returns the current display mode.
Set	Sets the display mode.

Class

[uc480.DisplayMode](#)

Accessible

Camera.Display.Mode

Syntax

```
uc480.DisplayMode.Get(out uc480.Defines.DisplayMode mode)
```

Description

Returns the current display mode.

Parameter

- `mode`: Returns the current mode (see [set\(\)](#)).

Class

[uc480.DisplayMode](#)

Accessible

Camera.Display.Mode

Syntax

```
uc480.DisplayMode.Set(uc480.Defines.DisplayMode mode)
```

Description

Sets the display mode.

Parameter

mode	<p>Mode to be set:</p> <ul style="list-style-type: none"> • uc480.Defines.DisplayMode.DIB: Captures an image in system memory (RAM). Using Render(), you can define the image display (default). • uc480.Defines.DisplayMode.Direct3D: Image display in Direct3D mode • uc480.Defines.DisplayMode.Direct3D uc480.Defines.DisplayMode.Mono: Monochrome image display in Direct3D mode • uc480.Defines.DisplayMode.Direct3D uc480.Defines.DisplayMode.Bayer: Raw Bayer format image display in Direct3D mode • uc480.Defines.DisplayMode.OpenGL: Image display in OpenGL mode • uc480.Defines.DisplayMode.OpenGL uc480.Defines.DisplayMode.Mono: Monochrome image display in OpenGL mode • uc480.Defines.DisplayMode.OpenGL uc480.Defines.DisplayMode.Bayer: Raw Bayer format image display in OpenGL mode
------	---

3.1.1.7.3 Position

The `Position` class provides a method for moving an area of interest when an image is rendered.

Methods

Method	Description
Set	Moves an area of interest when rendering images.

Class

[uc480.DisplayPosition](#)

Accessible

Camera.Display.Position

Syntax

```
uc480.DisplayPosition.Set(uc480.Types.Point<int> position)
uc480.DisplayPosition.Set(int s32X, int s32Y)
```

Description

Moves an area of interest when rendering images using [Render\(\)](#). The method moves the camera image by the selected offset within the output window. The image memory remains unchanged.



To set the size and position of an area of interest in memory, use [AOI](#).

Parameter

position	x: Offset in x direction, measured from the top left corner of the output window.
	y: Offset in y direction, measured from the top left corner of the output window.
s32X	Offset in x direction, measured from the top left corner of the output window.
s32Y	Offset in y direction, measured from the top left corner of the output window.

3.1.1.7.4 Render

Class

[uc480.Display](#)

Accessible

Camera.Display

Syntax

```
uc480.Display.Render(uc480.Defines.DisplayRenderMode mode)
uc480.Display.Render(System.IntPtr windowHandle)
uc480.Display.Render(System.IntPtr windowHandle, uc480.Defines.DisplayRenderMode mode)
uc480.Display.Render(int memID, uc480.Defines.DisplayRenderMode mode)
uc480.Display.Render(int memID, System.IntPtr windowHandle)
uc480.Display.Render(int memID)
uc480.Display.Render()
uc480.Display.Render(int memID, System.IntPtr windowHandle, uc480.Defines.DisplayRenderMode mode)
```

Description

Using this method, 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.

[Allocate\(\)](#) defines in its parameters the color depth and display type. RGB16 and RGB15 require the same amount of memory but can be distinguished by the parameter.



Display.Render() can render Y8 and RGB formats. For displaying YUV/YCbCr formats please use [DirectRenderer](#).

Parameter

memID	ID of the image memory whose contents is to be displayed.
windowHandle	Output window handle
mode	<p>Output mode:</p> <ul style="list-style-type: none"> • uc480.Defines.DisplayRenderMode.Normal: The image is rendered normally. It will be displayed in 1:1 scale as stored in the image memory. • uc480.Defines.DisplayRenderMode.FitToWindow: The image size is adjusted to fit the output window. • uc480.Defines.DisplayRenderMode.DownScale_1_2: Displays the image at 50 % of its original size. • uc480.Defines.DisplayRenderMode.PlanarColorBlue: Displays only the blue channel of planar color format. • uc480.Defines.DisplayRenderMode.PlanarColorGreen: Displays only the green channel of planar color format. • uc480.Defines.DisplayRenderMode.PlanarColorRed: Displays only the red channel of planar color format. • uc480.Defines.DisplayRenderMode.PlanarMonoBlue: Displays only the blue channel as monochrome image. • uc480.Defines.DisplayRenderMode.PlanarMonoGreen: Displays only the green channel as monochrome image. • uc480.Defines.DisplayRenderMode.PlanarMonoRed: Displays only the red channel as monochrome image. <p>The following option can be linked by a logical OR using the <code>s32Mode</code> parameter:</p> <ul style="list-style-type: none"> • uc480.Defines.DisplayRenderMode.MirrorUpDown: Mirrors the displayed image along the horizontal axis.

3.1.1.8 EdgeEnhancement

The `EdgeEnhancement` class provides methods for the software edge filter in a uc480 camera. Due to Bayer format color conversion, the original edges of a color image may easily become blurred. By enabling the digital edge filter, you can optimize edge representation. This function causes a higher CPU load.

If you perform Bayer conversion for GigE uc480 HE color cameras in the camera itself, edge enhancement will automatically also take place in the camera. In this case, the CPU load will not increase.



For USB uc480 XS cameras please use [Sharpness](#) instead.

Methods

Method	Description
Get	Returns the current set edge enhancement.
GetDefault	Returns the default value of the edge enhancement.
GetRange	Returns the range (minimum, maximum and increment) of the edge enhancement.
Set	Sets the edge enhancement.

3.1.1.8.1 Get

Class

[uc480.EdgeEnhancement](#)

Accessible

Camera.EdgeEnhancement

Syntax

```
uc480.EdgeEnhancement.Get(out int s32Value)
```

Description

Returns the current set edge enhancement.

Parameter

- `s32Value`: Returns the current edge enhancement.

3.1.1.8.2 GetDefault

Class

[uc480.EdgeEnhancement](#)

Accessible

Camera.EdgeEnhancement

Syntax

```
uc480.EdgeEnhancement.GetDefault(out int s32DefaultValue)
```

Description

Returns the default value of the edge enhancement.

Parameter

- `s32DefaultValue`: Returns the default value.

3.1.1.8.3 GetRange

Class

[uc480.EdgeEnhancement](#)

Accessible

Camera.EdgeEnhancement

Syntax

```
uc480.EdgeEnhancement.GetRange(out uc480.Types.Range<int> range)
uc480.EdgeEnhancement.GetRange(out int u32Min, out int u32Max, out int u32Inc)
```

Description

Returns the range (minimum, maximum and increment) of the edge enhancement.

Parameter

range	Minimum: Returns the minimum value.
	Maximum: Returns the maximum value.
	Increment: Returns the increment.
u32Min	Returns the minimum value.
u32Max	Returns the maximum value.
u32Inc	Returns the increment.

3.1.1.8.4 Set

Class

[uc480.EdgeEnhancement](#)

Accessible

Camera.EdgeEnhancement

Syntax

```
uc480.EdgeEnhancement.Set(int s32Value)
```

Description

Sets the edge enhancement.

Parameter

- `s32Value`: Value to be set.

3.1.1.9 EEPROM

The `EEPROM` class provides methods for writing and reading the EEPROM memory of a uc480 camera.

Methods

Method	Description
Read	Reads the content of the EEPROM memory.
Write	Writes into the EEPROM memory of a camera.

3.1.1.9.1 Read

Class

[uc480.EEPROM](#)

Accessible

Camera.EEPROM

Syntax

```
uc480.EEPROM.Read(int s32Addr, string strData)
uc480.EEPROM.Read(out string strData)
uc480.EEPROM.Read(out byte[] data, int len)
uc480.EEPROM.Read(int s32Addr, out byte[] data, int len)
```

Description

Using this method, you can read the contents of the camera EEPROM. Besides the hard-coded factory information, the EEPROM of the uc480 camera can hold 64 bytes of user data.

Parameter

s32Addr	Starting address for data reads. Value range: 0...63
strData	String for the data to read.
data	Bytes to be read
len	Length of data to be read

3.1.1.9.2 Write

Class

[uc480.EEPROM](#)

Accessible

Camera.EEPROM

Syntax

```
uc480.EEPROM.Write(string strData)
uc480.EEPROM.Write(int s32Addr, string strData)
uc480.EEPROM.Write(byte[] data)
uc480.EEPROM.Write(int s32Addr, byte[] data)
```

Description

Using this method, you can write data to the EEPROM of the camera. Besides the hard-coded factory information, the EEPROM of the uc480 camera can hold 64 bytes of user data.

Parameter

s32Addr	Starting address for data writes. Value range 0...63
strData	String containing the data to be written.
data	Bytes to be written
len	Length of data to be written

3.1.1.10 FaceDetection

The `FaceDetection` class provides methods for the built-in face detection of some uc480 cameras.



This class is currently only supported by the UI-1008XS models.

When you are using a UI-1008XS, you can only call the face detection in live mode. A call in software trigger mode will cause an error message. If you enable the software trigger mode after starting face detection, face detection will continue running as normal.

To ensure that the number of detected faces and their positions do not change during readout of the list, you should suspend with `SetSuspend()` before starting the readout. After the face detection results have been read out, you can resume again with `SetSuspend()`.

Note on camera orientation

To increase the detection reliability when the camera is turned upright, you can preset a facial orientation angle (search angle).

Upside-down faces are not detected. You have to turn the camera or enable the mirror function with `RopEffect` in order to detect upside-down faces.

Note on resolution for face detection

The position and size coordinates of the detected faces are independent of the image size setting. You can query the horizontal and vertical resolution of the face detection. For the UI-1008XS the coordinates are 0...320 in X direction and 0...240 in Y direction. These coordinates may need to be converted to fit the image size. The coordinate (0,0) is at the top left corner of the image.

Methods

Method	Description
<u>GetEnable</u>	Returns the face detection status (enabled/disabled).
<u>GetFaceList</u>	Returns a list of information on the detected faces.
<u>GetHorizontalResolution</u>	Returns the horizontal resolution of the face detection function.
<u>GetMaxNumOfFaces</u>	Returns the maximum number of faces that can be detected by the camera.
<u>GetMaxNumOverlay</u>	Returns the maximum number of faces that will be marked with an overlay box in the live image.
<u>GetNumOfFaces</u>	Returns the current number of detected faces.
<u>GetOverlayLineWidth</u>	Returns the line width set for the overlay boxes that indicate the detected faces in the live image.
<u>GetSearchAngle</u>	Returns the search angle that is currently set for face detection.
<u>GetSearchAngleEnable</u>	Returns the status of the search angle for face detection.
<u>GetSupported</u>	Returns the face detection functions supported by the camera.
<u>GetSuspend</u>	Returns the face detection status (suspended/running).
<u>GetVerticalResolution</u>	Returns the vertical resolution of the face detection function.
<u>SetEnable</u>	Enables/disables face detection.
<u>SetMaxNumOverlay</u>	Sets the maximum number of faces that will be marked with an overlay box in the live image.
<u>SetOverlayLineWidth</u>	Sets the line width you want to use for the overlay boxes around the detected faces in the live image.
<u>SetSearchAngle</u>	Sets a search angle for face detection.
<u>SetSearchAngleEnable</u>	Enables/disables the search angle for face detection.
<u>SetSuspend</u>	Temporarily suspends the face detection function./Resumes face detection.

3.1.1.10.1 GetEnable

Class

[uc480.FaceDetection](#)

Accessible

Camera.FaceDetection

Syntax

`uc480.FaceDetection.GetEnable(out bool bEnable)`



This class is currently only supported by the USB uc480 XS camera family.

Description

Returns the face detection status (enabled/disabled).

Parameter

bEnable	1 = Face detection is enabled 0 = Face detection is disabled
---------	---

3.1.1.10.2 GetFaceList

Class

[uc480.FaceDetection](#)

Accessible

Camera.FaceDetection

Syntax

```
uc480.FaceDetection.GetFaceList(out uc480.Types.FaceDetectionInformation[] FaceInfoList)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Returns a list of information on the detected faces.

Parameter

- FaceInfoList: Returns [uc480.Types.FaceDetectionInformation\[\]](#)

3.1.1.10.3 GetHorizontalResolution

Class

[uc480.FaceDetection](#)

Accessible

Camera.FaceDetection

Syntax

```
uc480.FaceDetection.GetHorizontalResolution(out uint u32Value)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Returns the horizontal resolution of the face detection function.

Parameter

- u32Value: Returns the horizontal resolution.

3.1.1.10.4 GetMaxNumOfFaces

Class

[uc480.FaceDetection](#)

Accessible

Camera.FaceDetection

Syntax

```
uc480.FaceDetection.GetMaxNumOfFaces(out uint u32MaxNumber)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Returns the maximum number of faces that can be detected by the camera.

Parameter

- `u32MaxNumber`: Returns the maximum number of faces.

3.1.1.10.5 GetMaxNumOverlay

Class

[uc480.FaceDetection](#)

Accessible

Camera.FaceDetection

Syntax

```
uc480.FaceDetection.GetMaxNumOverlay(out uint u32MaxNumOvl)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Returns the maximum number of faces that will be marked with an overlay box in the live image.

Parameter

- `u32MaxNumOvl`: Returns the maximum number of faces to be marked.

3.1.1.10.6 GetNumOfFaces

Class

[uc480.FaceDetection](#)

Accessible

Camera.FaceDetection

Syntax

```
uc480.FaceDetection.GetNumOfFaces(out uint u32Number)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Returns the current number of detected faces.

Parameter

- `u32MaxNumber`: Returns the number of detected faces.

3.1.1.10.7 GetOverlayLineWidth

Class

[uc480.FaceDetection](#)

Accessible

Camera.FaceDetection

Syntax

```
uc480.FaceDetection.GetOverlayLineWidth(out uint u32LineWidth)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Returns the line width set for the overlay boxes that indicate the detected faces in the live image.

Parameter

- `u32LineWidth`: Returns the line width.

3.1.1.10.8 GetSearchAngle

Class

[uc480.FaceDetection](#)

Accessible

Camera.FaceDetection

Syntax

```
uc480.FaceDetection.GetSearchAngle(out uint u32Angle)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Returns the search angle (facial orientation angle) that is currently set for face detection.

Parameter

- `u32Angle`: Returns the angle of 0, 90 or 270 degree.

3.1.1.10.9 GetSearchAngleEnable

Class

[uc480.FaceDetection](#)

Accessible

Camera.FaceDetection

Syntax

```
uc480.FaceDetection.GetSearchAngleEnable(out bool bEnable)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Returns the status of the search angle (facial orientation angle) for face detection.

Parameter

bEnable	1 = Search angle for face detection is enabled
	0 = Search angle for face detection is disabled

3.1.1.10.10 GetSupported

Class

[uc480.FaceDetection](#)

Accessible

Camera.FaceDetection

Syntax

```
uc480.FaceDetection.GetSupported(out uc480.Defines.FaceDetectionMode mode)
```



This class is currently only supported by the UI-1008XS models.

Description

Returns the face detection functions supported by the camera.

Parameter

- mode

uc480.Defines.FaceDetectionMode.Supported	The camera supports face detection
uc480.Defines.FaceDetectionMode.SearchAngle	Face detection supports presetting a search angle
uc480.Defines.FaceDetectionMode.SearchAoi	Face detection supports presetting a search area
uc480.Defines.FaceDetectionMode.InfoPositionX	Face detection supports output of the X position of a face
uc480.Defines.FaceDetectionMode.InfoPositionY	Face detection supports output of the Y position of a face
uc480.Defines.FaceDetectionMode.InfoWidth	Face detection supports output of the width of a face
uc480.Defines.FaceDetectionMode.InfoHeight	Face detection supports output of the height of a face
uc480.Defines.FaceDetectionMode.InfoOrientationAngle	Face detection supports output of the orientation angle of a face
uc480.Defines.FaceDetectionMode.InfoPosture	Face detection supports output of the posture of a face
uc480.Defines.FaceDetectionMode.InfoFaceNumber	Face detection supports output of the number of faces detected
uc480.Defines.FaceDetectionMode.InfoOverlay	Face detection supports marking detected faces with an overlay box
uc480.Defines.FaceDetectionMode.InfoNumberOfOverlay	Face detection supports setting the maximum number of faces for marking
uc480.Defines.FaceDetectionMode.InfoOverlayLineWidth	Face detection supports setting the line width for the overlay boxes

3.1.1.10.11 GetSuspend

Class

[uc480.FaceDetection](#)

Accessible

Camera.FaceDetection

Syntax

```
uc480.FaceDetection.GetSuspend(out bool bSuspend)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Returns the face detection status (suspended/running).

Parameter

bSuspend	1= Face detection is suspended 0 = Face detection is running
----------	---

3.1.1.10.12 GetVerticalResolution

Class

[uc480.FaceDetection](#)

Accessible

Camera.FaceDetection

Syntax

```
uc480.FaceDetection.GetVerticalResolution(out uint u32Value)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Returns the vertical resolution of the face detection function.

Parameter

- `u32Value`: Returns the vertical resolution.

3.1.1.10.13 SetEnable

Class

[uc480.FaceDetection](#)

Accessible

Camera.FaceDetection

Syntax

```
uc480.FaceDetection.SetEnable(bool bEnable)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Enables/disables face detection.

Parameter

<code>bEnable</code>	1 = Enable face detection
	0 = Disable face detection

3.1.1.10.14 SetMaxNumOverlay

Class

[uc480.FaceDetection](#)

Accessible

Camera.FaceDetection

Syntax

```
uc480.FaceDetection.SetMaxNumOverlay(uint u32MaxNumOvl)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Sets the maximum number of faces that will be marked with an overlay box in the live image.

Parameter

- `u32MaxNumOvl`: Maximum number of faces to be marked

3.1.10.15 SetOverlayLineWidth

Class

[uc480.FaceDetection](#)

Accessible

Camera.FaceDetection

Syntax

```
uc480.FaceDetection.SetOverlayLineWidth(uint u32LineWidth)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Sets the line width you want to use for the overlay boxes around the detected faces in the live image.

Parameter

- `u32LineWidth`: Border width of the boxes around the detected faces

3.1.10.16 SetSearchAngle

Class

[uc480.FaceDetection](#)

Accessible

Camera.FaceDetection

Syntax

```
uc480.FaceDetection.SetSearchAngle(uint u32Angle)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Sets a search angle for face detection.

Parameter

- `u32Angle`: Search angle for face detection

3.1.1.10.17 SetSearchAngleEnable

Class

[uc480.FaceDetection](#)

Accessible

Camera.FaceDetection

Syntax

```
uc480.FaceDetection.SetSearchAngleEnable(bool bEnable)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Enables/disables the search angle for face detection.

Parameter

bEnable	1 = Enable search angle for face detection
	0 = Disable search angle for face detection

3.1.1.10.18 SetSuspend

Class

[uc480.FaceDetection](#)

Accessible

Camera.FaceDetection

Syntax

```
uc480.FaceDetection.SetSuspend(bool bSuspend)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Temporarily suspends the face detection function/resumes face detection.

Parameter

bSuspend	1= Suspend face detection
	0 = Resume face detection

3.1.1.11 Focus

The `Focus` class provides methods for controlling the focus of the objective lens of some uc480 cameras.



This class is currently only supported by the USB uc480 XS camera family.

According to the sensor manufacturer, there is no guarantee that a focal point set manually for the USB uc480 XS will lead to a reproducible focal distance. A focal value determined with one USB uc480 XS model will not necessarily result in the same focal point for a different model.

The following classes and method exist:

- [Auto](#)
- [Manual](#)
- [Zone](#)

Methods

Method	Description
Trigger	Triggers once the adjustment of the auto focus.

3.1.11.1 Auto

The `Auto` class provides methods for the automatic control of the objective lens of some uc480 cameras.



This class is currently only supported by the USB uc480 XS camera family.

According to the sensor manufacturer, there is no guarantee that a focal point set manually for the USB uc480 XS will lead to a reproducible focal distance. A focal value determined with one USB uc480 XS model will not necessarily result in the same focal point for a different model.

Methods

Method	Description
GetEnable	Returns the auto focus status.
GetObjectDistance	Returns the approximate distance in millimeters to the focused object when auto focus is enabled.
GetStatus	Returns the auto focus status.
GetSupported	Returns the focus functions supported by the camera.
SetEnable	Enables/disables auto focus.

Class

[uc480.FocusAuto](#)

Accessible

Camera.Focus.Auto

Syntax

```
uc480.FocusAuto.GetEnable(out bool bEnable)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Returns if the auto focus is enabled/disabled.

Parameter

bEnable	1 = Auto focus enabled
	0 = Auto focus disabled

Class[uc480.FocusAuto](#)**Accessible**

Camera.Focus.Auto

Syntax`uc480.FocusAuto.GetObjectDistance(out uc480.Defines.FocusDistance distance)`

This class is currently only supported by the UI-1008XS models.

Description

Returns the approximate distance in millimeters to the focused object when auto focus is enabled. The USB uc480 XS distinguishes three distances:

- macro = 100 mm
- near = 500 mm
- far = 20,000 mm

Parameter

distance	Returns the approximate distance to the focused object: • uc480.Defines.FocusDistance.Macro • uc480.Defines.FocusDistance.Near • uc480.Defines.FocusDistance.Far • uc480.Defines.FocusDistance.None
----------	---

Class[uc480.FocusAuto](#)**Accessible**

Camera.Focus.Auto

Syntax`uc480.FocusAuto.GetStatus(out uc480.Defines.FocusStatus status)`

This class is currently only supported by the USB uc480 XS camera family.

Description

Returns the auto focus status.

Parameter

status	<p>Returns the state of the auto focus:</p> <ul style="list-style-type: none"> • uc480.Defines.FocusStatus.Error: An error occurred. • uc480.Defines.FocusStatus.Focused: The lens is focused. • uc480.Defines.FocusStatus.Focusing: The lens is being focused. • uc480.Defines.FocusStatus.Timeout: A timeout occurred. • uc480.Defines.FocusStatus.Cancel: Lens focusing was canceled
--------	--

Class

[uc480.FocusAuto](#)

Accessible

Camera.Focus.Auto

Syntax

```
uc480.FocusAuto.GetSupported(out bool bSupported)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Returns the focus functions supported by the camera.

Parameter

bSupported	1 = Auto focus is supported 0 = Auto focus is not supported
------------	--

Class

[uc480.FocusAuto](#)

Accessible

Camera.Focus.Auto

Syntax

```
uc480.FocusAuto.SetEnable(bool bEnable)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Enables/disables auto focus.

Parameter

bEnable	1 = Enable auto focus
	0 = Disable auto focus

3.1.11.2 Manual

The `Manual` class provides methods for the manual control of the objective lens of some uc480 cameras.



This class is currently only supported by the USB uc480 XS camera family.

According to the sensor manufacturer, there is no guarantee that a focal point set manually for the USB uc480 XS will lead to a reproducible focal distance. A focal value determined with one USB uc480 XS model will not necessarily result in the same focal point for a different model.

Methods

Method	Description
Get	Returns the current set manual focus.
GetRange	Returns the range for manually setting the focal point.
Set	Sets a manual focal point.

Class

[uc480.FocusManual](#)

Accessible

Camera.Focus.Manual

Syntax

```
uc480.FocusManual.Get(out uint u32Value)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Returns the current set manual focus.

Parameter

- `u32Value`: Returns the manual focus.
0 = No valid manual focus has been set.

Class

[uc480.FocusManual](#)

Accessible

Camera.Focus.Manual

Syntax

```
uc480.FocusManual.GetRange(out uc480.Types.Range<uint> range)
uc480.FocusManual.GetRange(out uint u32Min, out uint u32Max, out uint u32Inc)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Returns the range for manually setting the focal point.

Parameter

range	Minimum: Returns the minimum value for the color temperature.
	Maximum: Returns the maximum value for the color temperature.
	Increment: Returns the increment for setting the color temperature.
u32Min	Returns the minimum value for the color temperature.
u32Max	Returns the maximum value for the color temperature.
u32Inc	Returns the increment.

Class

[uc480.FocusManual](#)

Accessible

Camera.Focus.Manual

Syntax

`uc480.FocusManual.Set(uint u32Value)`



This class is currently only supported by the USB uc480 XS camera family.

Description

Sets a manual focal point. The range for the manual focus can be queried with [GetRange\(\)](#).

Parameter

- **u32Value:** Value for the manual focus.

3.1.1.11.3 Zone

The `Zone` class provides methods for setting a measuring window. Via the measuring window you define the area which should be focused. The measuring window is divided in multiple zones. You can set a different weight for each zone.



This class is currently only supported by the XS model.

Methods

Method	Description
GetAOI	Returns the current measuring window for the auto focus.
GetAOIDefault	Returns the default measuring window for the auto focus.
GetCount	Returns the number of zones in the measuring window.
GetPosRange	Returns the range of the measuring window position.
GetSizeRange	Returns the range of the measuring window size.
GetSupported	Returns if the focus measuring window is supported by the camera.
GetWeight	Returns the weight of the zones in the measuring window.
GetWeightDefault	Returns the standard weight of the zones in the measuring window.
SetAOI	Sets the measuring window for the auto focus.
SetWeight	Sets the weight of the zones in the measuring window.

Class

[uc480.FocusZone](#)

Accessible

Camera.Focus.Zone

Syntax

`uc480.FocusZone.GetAOI(out System.Drawing.Rectangle rect)`

Description

Returns the current measuring window for the auto focus.

Parameter

- `rect`: Returns the position and size of the measuring window

Class

[uc480.FocusZone](#)

Accessible

Camera.Focus.Zone

Syntax

`uc480.FocusZone.GetAOIDefault(out System.Drawing.Rectangle rect)`

Description

Returns the default measuring window for the auto focus.

Parameter

- `rect`: Returns the default position and size of the measuring window

Class

[uc480.FocusZone](#)

Accessible

Camera.Focus.Zone

Syntax

```
uc480.FocusZone.GetCount(out int count)
```

Description

Returns the number of zones in the measuring window.

Parameter

- `count`: Returns the number of zones

Class

[uc480.FocusZone](#)

Accessible

Camera.Focus.Zone

Syntax

```
uc480.FocusZone.GetPosRange(out uc480.Types.Range<int> rangePosX, out uc480.Types.Range<int> rangePosY)
```

Description

Returns the range of the measuring window position.

Parameter

rangePosX	<ul style="list-style-type: none"> • <code>Minimum</code>: Returns the minimum value. • <code>Maximum</code>: Returns the maximum value. • <code>Increment</code>: Returns the increment.
rangePosY	<ul style="list-style-type: none"> • <code>Minimum</code>: Returns the minimum value. • <code>Maximum</code>: Returns the maximum value. • <code>Increment</code>: Returns the increment.

Class

[uc480.FocusZone](#)

Accessible

Camera.Focus.Zone

Syntax

```
uc480.FocusZone.GetSizeRange(out uc480.Types.Range<int> rangeWidth, out uc480.Types.Range<int> rangeHeight)
```

Description

Returns the range of the measuring window size.

Parameter

rangeWidth	<ul style="list-style-type: none">• Minimum: Returns the minimum value.• Maximum: Returns the maximum value.• Increment: Returns the increment.
rangeHeight	<ul style="list-style-type: none">• Minimum: Returns the minimum value.• Maximum: Returns the maximum value.• Increment: Returns the increment.

Class

[uc480.FocusZone](#)

Accessible

Camera.Focus.Zone

Syntax

```
uc480.FocusZone.GetSupported(out bool supported)
```

Description

Returns if the focus measuring window is supported by the camera.

Parameter

supported	1 = Measuring window is supported
	0 = Measuring window is not supported

Class

[uc480.FocusZone](#)

Accessible

Camera.Focus.Zone

Syntax

```
uc480.FocusZone.GetWeight(out int[] weight)
```

Description

Returns the weight of the zones in the measuring window.

Parameter

- weight: Returns the current weight of the zones, see [SetWeight\(\)](#)

Class

[uc480.FocusZone](#)

Accessible

Camera.Focus.Zone

Syntax

```
uc480.FocusZone.GetWeightDefault(out int[] weight)
```

Description

Returns the standard weight of the zones in the measuring window.

Parameter

- `weight`: Returns the default value.

Class

[uc480.FocusZone](#)

Accessible

Camera.Focus.Zone

Syntax

```
uc480.FocusZone.SetAOI(System.Drawing.Rectangle rect)
```

Description

Sets the measuring window for the auto focus.

Parameter

- `rect`: Sets the position and size of the measuring window

Class

[uc480.FocusZone](#)

Accessible

Camera.Focus.Zone

Syntax

```
uc480.FocusZone.SetWeight(int[] weight)
```

Description

Sets the weight of the zones in the measuring window.

Parameter

<code>weight</code>	Value to be set: <ul style="list-style-type: none"> • 0 = disabled • 33 = weak • 66 = middle • 100 = strong
---------------------	--

3.1.1.11.4 Trigger

Class

[uc480.Focus](#)

Accessible

Camera.Focus

Syntax

```
uc480.Focus.Trigger()
```

Description

Triggers once the adjustment of the auto focus.

Parameter

none

3.1.1.12 Gain

The `Gain` class provides methods for controlling the gain factor and the sensor gain channels. The following class exists:

- [Hardware](#)

3.1.1.12.1 Hardware

The `Hardware` class provides methods for controlling the sensor gain channels or an additional analog hardware gain boost feature on the sensor. The following classes and methods exist:

- [Boost](#)
- [ConvertScaledToFactor](#)
- [Factor](#)
- [Scaled](#)

Methods

Method	Description
GetSupported	Returns if red, blue, green or master gain is supported.

The `Boost` class provides methods for enabling an additional analog hardware gain boost feature on the sensor.

Methods

Method	Description
GetEnable	Returns the current state of the gain boost function.
GetSupported	Indicates whether the camera supports a gain boost feature or not.
SetEnable	Enables/disables the gain boost function.

Class

[uc480.GainBoost](#)

Accessible

`Camera.Gain.Hardware.Boost`

Syntax

`uc480.GainBoost.GetEnable(out bool bEnable)`

Description

Returns the current state of the gain boost function.

Parameter

- `bEnable`: Returns the current setting.

Class

[uc480.GainBoost](#)

Accessible

Camera.Gain.Hardware.Boost

Syntax

```
uc480.GainBoost.GetSupported(out bool bSupported)
```

Description

Indicates whether the camera supports a gain boost feature or not.

Parameter

bSupported	1 = Hardware gain boost is supported 0 = Hardware gain boost is not supported
------------	--

Class

[uc480.GainBoost](#)

Accessible

Camera.Gain.Hardware.Boost

Syntax

```
uc480.GainBoost.SetEnable(bool bEnable)
```

Description

Enables/disables the gain boost function.

Parameter

bEnable	1 = Enable hardware gain boost 0 = Disable hardware gain boost
---------	---

The `ConvertScaledToFactor` class provides methods for using gain factors to control sensor gain channels. These channels can be set independently of each other. The `Factor` class does not use factors for setting the gain channels, but standardized values between 0 and 100. The actual gain factor is sensor-dependent and can be found in "Camera and sensor data" chapter in the uc480 manual.

You can use [GetSensorInfo\(\)](#) 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.

Methods

Method	Description
Blue	Converts the index value for the blue gain factor.
Green	Converts the index value for the green gain factor.
Master	Converts the index value for the master gain factor.
Red	Converts the index value for the red gain factor.

Class[uc480.GainConvertScaledToFactor](#)**Accessible**

Camera.Gain.Hardware.ConvertScaledToFactor

Syntax

uc480.GainConvertScaledToFactor.Blue(int s32Blue, out int s32Factor)

Description

Converts the index value for the blue gain factor.

Parameter

s32Blue	Index value to be converted
s32Factor	Gain value (100 = gain factor 1, i. e. no effect)

Class[uc480.GainConvertScaledToFactor](#)**Accessible**

Camera.Gain.Hardware.ConvertScaledToFactor

Syntax

uc480.GainConvertScaledToFactor.Green(int s32Green, out int s32Factor)

Description

Converts the index value for the green gain factor.

Parameter

s32Green	Index value to be converted
s32Factor	Gain value (100 = gain factor 1, i. e. no effect)

Class[uc480.GainConvertScaledToFactor](#)**Accessible**

Camera.Gain.Hardware.ConvertScaledToFactor

Syntax

uc480.GainConvertScaledToFactor.Master(int s32Master, out int s32Factor)

Description

Converts the index value for the master gain factor.

Parameter

s32Master	Index value to be converted
s32Factor	Gain value (100 = gain factor 1, i. e. no effect)

Class

[uc480.GainConvertScaledToFactor](#)

Accessible

Camera.Gain.Hardware.ConvertScaledToFactor

Syntax

```
uc480.GainConvertScaledToFactor.Red(int s32Red, out int s32Factor)
```

Description

Converts the index value for the red gain factor.

Parameter

s32Red	Index value to be converted
s32Factor	Gain value (100 = gain factor 1, i. e. no effect)

The `Factor` class provides methods for controlling 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 "Camera and sensor data" chapter in the uc480 manual.

For HDR camera models UI-112x/UI-512x, this class changes the white level of the image.

You can use [GetSensorInfo\(\)](#) to query the available gain controls.



Note on using sensor gain

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: [SetEnable\(\)](#).
2. If required, adjust the gain setting with `Factor`.

New gain settings might only become effective when the next image is captured. This depends on the time when the gain settings are changed.



Note on the linearity of sensor gain

On uc480 cameras, you can set the gain factor in increments from 0 to 100. These increments are not graduated linearly throughout the range due to the sensor. The increments will typically be greater in the upper range than in the lower range.

The maximum gain factor settings also vary from sensor to sensor.



Note on default settings for RGB gains

The default setting values for the red, green and blue channel gain factors depend on the color correction matrix that has been set. If you select a different color correction matrix, the returned default values might change (see also [set\(\)](#)).

Methods

Method	Description
GetDefaultBlue	Returns the default blue gain factor.
GetDefaultGreen	Returns the default green gain factor.
GetDefaultMaster	Returns the default master gain factor.
GetDefaultRed	Returns the default red gain factor.
GetBlue	Returns the blue channel gain factor.
GetGreen	Returns the green channel gain factor.
GetMaster	Returns the master channel gain factor.
GetRed	Returns the red channel gain factor.
SetBlue	Sets the blue channel gain factor (0...100).
SetGreen	Sets the green channel gain factor (0...100).
SetMaster	Sets the master channel gain factor (0...100).
SetRed	Sets the red channel gain factor (0...100).

Class

[uc480.GainFactor](#)

Accessible

Camera.Gain.Hardware.Factor

Syntax

`uc480.GainFactor.GetDefaultBlue(out int s32Factor)`

Description

Returns the default blue gain factor.

Parameter

- `s32Factor`: Returns the default factor.

Class

[uc480.GainFactor](#)

Accessible

Camera.Gain.Hardware.Factor

Syntax

`uc480.GainFactor.GetDefaultGreen(out int s32Factor)`

Description

Returns the default green gain factor.

Parameter

- `s32Factor`: Returns the default factor.

Class

[uc480.GainFactor](#)

Accessible

Camera.Gain.Hardware.Factor

Syntax

uc480.GainFactor.GetDefaultMaster(out int s32Factor)

Description

Returns the default master gain factor.

Parameter

- `s32Factor`: Returns the default factor.

Class

[uc480.GainFactor](#)

Accessible

Camera.Gain.Hardware.Factor

Syntax

uc480.GainFactor.GetDefaultRed(out int s32Factor)

Description

Returns the default red gain factor.

Parameter

- `s32Factor`: Returns the default factor.

Class

[uc480.GainFactor](#)

Accessible

Camera.Gain.Hardware.Factor

Syntax

uc480.GainFactor.GetBlue(out int s32Factor)

Description

Returns the blue channel gain factor.

Parameter

- `s32Factor`: Returns the current factor.

Class

[uc480.GainFactor](#)

Accessible

Camera.Gain.Hardware.Factor

Syntax

uc480.GainFactor.GetGreen(out int s32Factor)

Description

Returns the green channel gain factor.

Parameter

- `s32Factor`: Returns the current factor.

Class

[uc480.GainFactor](#)

Accessible

Camera.Gain.Hardware.Factor

Syntax

```
uc480.GainFactor.GetMaster(out int s32Factor)
```

Description

Returns the master channel gain factor.

Parameter

- `s32Factor`: Returns the current factor.

Class

[uc480.GainFactor](#)

Accessible

Camera.Gain.Hardware.Factor

Syntax

```
uc480.GainFactor.GetRed(out int s32Factor)
```

Description

Returns the red channel gain factor.

Parameter

- `s32Factor`: Returns the current factor.

Class

[uc480.GainFactor](#)

Accessible

Camera.Gain.Hardware.Factor

Syntax

```
uc480.GainFactor.SetBlue(int s32Factor)
```

Description

Sets the blue channel gain factor (0...100).

Parameter

- `s32Factor`: Factor to be set.

Class

[uc480.GainFactor](#)

Accessible

Camera.Gain.Hardware.Factor

Syntax

`uc480.GainFactor.SetGreen(int s32Factor)`

Description

Sets the green channel gain factor (0...100).

Parameter

- `s32Factor`: Factor to be set.

Class

[uc480.GainFactor](#)

Accessible

Camera.Gain.Hardware.Factor

Syntax

`uc480.GainFactor.SetMaster(int s32Factor)`

Description

Sets the master channel gain factor (0...100).

Parameter

- `s32Factor`: Factor to be set.

Class

[uc480.GainFactor](#)

Accessible

Camera.Gain.Hardware.Factor

Syntax

`uc480.GainFactor.SetRed(int s32Factor)`

Description

Sets the red channel gain factor (0...100).

Parameter

- `s32Factor`: Factor to be set.

The `Scaled` class provides methods for controlling 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 "Camera and sensor data" chapter in the uc480 manual.

For HDR camera models UI-112x/UI-512x, this class changes the white level of the image.

You can use [GetInfo\(\)](#) to query the available gain controls.

Note on using sensor gain

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: [SetEnable\(\)](#) .
2. If required, adjust the gain setting with `Scaled`.

New gain settings might only become effective when the next image is captured. This depends on the time when the gain settings are changed.

Note on the linearity of sensor gain

On uc480 cameras, you can set the gain factor in increments from 0 to 100. These increments are not graduated linearly throughout the range due to the sensor. The increments will typically be greater in the upper range than in the lower range.

The maximum gain factor settings also vary from sensor to sensor.

Note on default settings for RGB gains

The default setting values for the red, green and blue channel gain factors depend on the color correction matrix that has been set. If you select a different color correction matrix, the returned default values might change (see also [Set\(\)](#)).

Methods

Method	Description
GetDefaultBlue	Returns the default blue channel gain factor.
GetDefaultGreen	Returns the default green channel gain factor.
GetDefaultMaster	Returns the default master channel gain factor.
GetDefaultRed	Returns the default red channel gain factor.
GetBlue	Returns the blue channel gain factor.
GetGreen	Returns the green channel gain factor.
GetMaster	Returns the master channel gain factor.
GetRed	Returns the red channel gain factor.
SetBlue	Sets the blue channel gain factor (0...100).
SetGreen	Sets the green channel gain factor (0...100).
SetMaster	Sets the master channel gain factor (0...100).
SetRed	Sets the red channel gain factor (0...100).

Class

[uc480.GainHardwareScaled](#)

Accessible

Camera.Gain.Hardware.Scaled

Syntax

`uc480.GainHardwareScaled.GetDefaultBlue(out int s32Value)`

Description

Returns the default blue channel gain factor.

Parameter

- `s32Value`: Returns the default factor.

Class

[uc480.GainHardwareScaled](#)

Accessible

Camera.Gain.Hardware.Scaled

Syntax

`uc480.GainHardwareScaled.GetDefaultGreen(out int s32Value)`

Description

Returns the default green channel gain factor.

Parameter

- `s32Value`: Returns the default factor.

Class

[uc480.GainHardwareScaled](#)

Accessible

Camera.Gain.Hardware.Scaled

Syntax

`uc480.GainHardwareScaled.GetDefaultMaster(out int s32Value)`

Description

Returns the default master channel gain factor.

Parameter

- `s32Value`: Returns the default factor.

Class

[uc480.GainHardwareScaled](#)

Accessible

Camera.Gain.Hardware.Scaled

Syntax

`uc480.GainHardwareScaled.GetDefaultRed(out int s32Value)`

Description

Returns the default red channel gain factor.

Parameter

- `s32Value`: Returns the default factor.

Class

[uc480.GainHardwareScaled](#)

Accessible

Camera.Gain.Hardware.Scaled

Syntax

`uc480.GainHardwareScaled.GetBlue(out int s32Value)`

Description

Returns the blue channel gain factor.

Parameter

- `s32Value`: Returns the default factor.

Class

[uc480.GainHardwareScaled](#)

Accessible

Camera.Gain.Hardware.Scaled

Syntax

`uc480.GainHardwareScaled.GetGreen(out int s32Value)`

Description

Returns the green channel gain factor.

Parameter

- `s32Value`: Returns the default factor.

Class

[uc480.GainHardwareScaled](#)

Accessible

Camera.Gain.Hardware.Scaled

Syntax

`uc480.GainHardwareScaled.GetMaster(out int s32Value)`

Description

Returns the master channel gain factor.

Parameter

- `s32Value`: Returns the default factor.

Class

[uc480.GainHardwareScaled](#)

Accessible

Camera.Gain.Hardware.Scaled

Syntax

`uc480.GainHardwareScaled.GetRed(out int s32Value)`

Description

Returns the red channel gain factor.

Parameter

- `s32Value`: Returns the default factor.

Class

[uc480.GainHardwareScaled](#)

Accessible

Camera.Gain.Hardware.Scaled

Syntax

`uc480.GainHardwareScaled.SetBlue(int s32Value)`

Description

Sets the blue channel gain factor (0...100).

Parameter

- `s32Value`: Gain factor to be set.

Class

[uc480.GainHardwareScaled](#)

Accessible

Camera.Gain.Hardware.Scaled

Syntax

`uc480.GainHardwareScaled.SetGreen(int s32Value)`

Description

Sets the green channel gain factor (0...100).

Parameter

- `s32Value`: Gain factor to be set.

Class

[uc480.GainHardwareScaled](#)

Accessible

Camera.Gain.Hardware.Scaled

Syntax

`uc480.GainHardwareScaled.SetMaster(int s32Value)`

Description

Sets the master channel gain factor (0...100).

Parameter

- `s32Value`: Gain factor to be set.

Class

[uc480.GainHardwareScaled](#)

Accessible

Camera.Gain.Hardware.Scaled

Syntax

`uc480.GainHardwareScaled.SetRed(int s32Value)`

Description

Sets the red channel gain factor (0...100).

Parameter

- `s32Value`: Gain factor to be set.

Class

[uc480.GainHardware](#)

Accessible

Camera.Gain.Hardware

Syntax

`uc480.GainHardware.GetSupported(out bool bMasterGain, out bool bRed, out bool bGreen, out bool bBlue)`

Description

Returns if red, blue, green or master gain is supported.

Parameter

<code>bMasterGain</code>	1 = Supported 0 = Not supported
<code>bRed</code>	1 = Supported 0 = Not supported
<code>bGreen</code>	1 = Supported 0 = Not supported
<code>bBlue</code>	1 = Supported 0 = Not supported

3.1.13 Gamma

The `Gamma` class provides methods for controlling the hardware gamma or the digital software gamma correction. The following class and methods exist:

- [Hardware](#)
- [Software](#)

3.1.13.1 Hardware

The `Hardware` class provides methods for controlling the hardware gamma correction with a fixed gamma value.



Hardware gamma correction is only supported by sensors of the UI-122x/UI-322x/UI-522x camera models.



After enabling/disabling hardware sensor gamma correction, the white balance needs to be readjusted for color sensors (see [Software](#)).

Methods

Method	Description
GetEnable	Returns the current setting of hardware gamma correction.
GetSupported	Returns if the camera supports hardware gamma correction.
SetEnable	Enables/disables the hardware gamma correction.

Class

[uc480.GammaHardware](#)

Accessible

Camera.Gamma.Hardware

Syntax

```
uc480.GammaHardware.GetEnable(out bool bEnable)
```

Description

Returns the current setting of hardware gamma correction.

Parameter

bEnable	1 = Hardware gamma is enabled 0 = Hardware gamma is disabled
---------	---

Class

[uc480.GammaHardware](#)

Accessible

Camera.Gamma.Hardware

Syntax

```
uc480.GammaHardware.GetSupported(out bool bSupported)
```

Description

Returns if the camera supports hardware gamma correction.

Parameter

bSupported	1 = Hardware gamma correction is supported 0 = Hardware gamma correction is not supported
------------	--

Class

[uc480.GammaHardware](#)

Accessible

Camera.Gamma.Hardware

Syntax

```
uc480.GammaHardware.SetEnable(bool bEnable)
```

Description

Enables/disables the hardware gamma correction.

Parameter

bEnable	1 = Enable hardware gamma correction 0 = Disable hardware gamma correction
---------	---

3.1.1.13.2 Software

The `Software` class provides method for controlling the digital gamma correction which applies a gamma characteristic to the image. When hardware color conversion is used on GigE uc480 HE cameras the gamma correction is performed in the camera hardware as well. When the color conversion is performed in the PC (software conversion) the gamma correction is performed in software.



When the color format is set to Raw Bayer the gamma correction can not be used.



Typical values for gamma range between 1.6 and 2.2.

Methods

Method	Description
<u>Get</u>	Returns the current set value for the digital gamma correction.
<u>GetDefault</u>	Returns the default value for the digital gamma correction.
<u>GetRange</u>	Returns the range of the digital gamma correction.
<u>Set</u>	Sets the value of the digital gamma correction.

Class

[uc480.GammaSoftware](#)

Accessible

Camera.Gamma.Software

Syntax

```
uc480.GammaSoftware.Get(out int value)
```

Description

Returns the current set value for the digital gamma correction.

Parameter

- `value`: Returns the current value.

Class

[uc480.GammaSoftware](#)

Accessible

Camera.Gamma.Software

Syntax

```
uc480.GammaSoftware.GetDefault(out int value)
```

Description

Returns the default value for the digital gamma correction.

Parameter

- **value**: Returns the default value.

Class

[uc480.GammaSoftware](#)

Accessible

Camera.Gamma.Software

Syntax

```
uc480.GammaSoftware.GetRange(out uc480.Types.Range<int> range)
uc480.GammaSoftware.GetRange(out int s32Min, out int s32Max, out int s32Inc)
```

Description

Returns the minimum and maximum value and the increment of the digital gamma correction.

Parameter

range	Minimum: Returns the minimum value. Maximum: Returns the maximum value. Increment: Returns the increment.
s32Min	Returns the minimum value.
s32Max	Returns the maximum value.
s32Inc	Returns the increment.

Class

[uc480.GammaSoftware](#)

Accessible

Camera.Gamma.Software

Syntax

```
uc480.GammaSoftware.Set(int value)
```

Description

Sets the value of the digital gamma correction.

Parameter

- `value`: Gamma value to be set, multiplied with 100 (range: 1...1000, default value = 100, corresponds to a gamma of 1.0)

3.1.1.14 GigE

The `GigE` class provides methods for configuring the IP of GigE uc480 cameras, setting-up the transfer or uploading the starter firmware. The following classes exist:

- [Comport](#)
- [Transfer](#)

3.1.1.14.1 Comport

The `Comport` class provides a method for returning the current COM port number of a uc480 camera. The default port number is 100. You can change the port number in the IDS Camera Manager.



The `Comport` class is only supported by cameras of the USB 3 uc480 CP and GigE uc480 HE series.

Methods

Method	Description
GetNumber	Returns the COM port number.

Class

[uc480.Comport](#)

Accessible

Camera.GigE.Comport

Syntax

```
uc480.Comport.GetNumber(out uint u32ComNumber)
```

Description

Returns the COM port number of a uc480 camera.



The `Comport` class is only supported by cameras of the USB 3 uc480 CP and GigE uc480 HE series.

Parameter

- `u32ComNumber`: Returns the COM port number.

3.1.1.14.2 Transfer

The `Transfer` class provide methods for adjusting the latency of the image data transfer of GigE uc480 cameras. You can adjust the interval between capturing and transferring an image for each individual camera that is connected. This may be useful to divide available bandwidth amongst cameras running simultaneously on slow networks (e.g. transfer in a wireless or 100 Mbit network). Currently, you can adjust the following parameters:

- `Image delay`

This value determines the internal camera delay of the image transfer. When the

sensor begins outputting data, the camera waits for the specified duration before starting to transfer the image data. In this way you can coordinate the data transfer for several simultaneously triggered cameras.

- **Packet interval**

GigE uc480 cameras transfer image data in several data packets of the same size. The packet interval determines the interval between the transfer of two successive packets, improving the data transfer of one or several cameras on slow networks.



The usual value for the "Packet interval" in Gigabit Ethernet networks is around 20 µs. Higher values can reduce the transfer speed of the GigE uc480 considerably.

The following classes exist:

- [ImageDelay](#)
- [PacketInterval](#)

The `ImageDelay` class provides methods for determining the internal camera delay of the image transfer. When the sensor begins outputting data, the camera waits for the specified duration before starting to transfer the image data. In this way you can coordinate the data transfer for several simultaneously triggered cameras.

Methods

Method	Description
Get	Returns the internal camera delay of the image transfer in microseconds (µs).
GetDefault	Returns the default value for image delay.
GetRange	Returns the value range for the internal camera delay of the image transfer in microseconds (µs).
GetSupported	Returns if setting the image delay is supported.
Set	Sets the internal camera delay of the image transfer in microseconds (µs).

Class

[uc480.GigEImageDelay](#)

Accessible

Camera.GigE.Transfer.ImageDelay

Syntax

```
uc480.GigEImageDelay.Get(out uint u32Value)
```

Description

Returns the internal camera delay of the image transfer in microseconds (µs).

Parameter

- `u32Value`: Returns the current delay.

Class

[uc480.GigEImageDelay](#)

Accessible

Camera.GigE.Transfer.ImageDelay

Syntax

```
uc480.GigEImageDelay.GetDefault(out uint u32DefaultValue)
```

Description

Returns the default value for image delay.

Parameter

- `u32DefaultValue`: Returns the default delay.

Class

[uc480.GigEImageDelay](#)

Accessible

Camera.GigE.Transfer.ImageDelay

Syntax

```
uc480.GigEImageDelay.GetRange(out uc480.Types.Range<uint> range)
uc480.GigEImageDelay.GetRange(out uint u32Min, out uint u32Max, out uint u32Inc)
```

Description

Returns the value range for the internal camera delay of the image transfer in microseconds (μ s).

Parameter

range	Minimum: Returns the minimum value.
	Maximum: Returns the maximum value.
	Increment: Returns the increment.
u32Min	Returns the minimum value.
u32Max	Returns the maximum value.
u32Inc	Returns the increment.

Class

[uc480.GigEImageDelay](#)

Accessible

Camera.GigE.Transfer.ImageDelay

Syntax

```
uc480.GigEImageDelay.GetSupported(out bool bSupported)
```

Description

Returns if setting the image delay is supported.

Parameter

bSupported	1 = Image delay is supported. 0 = Image delay is not supported.
------------	--

Class

[uc480.GigEImageDelay](#)

Accessible

Camera.GigE.Transfer.ImageDelay

Syntax

```
uc480.GigEImageDelay.Set(uint u32Value)
```

Description

Sets the internal camera delay of the image transfer in microseconds (μs).

Parameter

- `u32Value`: Image delay to be set.

The `PacketInterval` class provides methods for transferring image data in several data packets of the same size. The packet interval value determines the interval between the transfer of two successive packets, improving the data transfer of one or several cameras on slow networks.



The usual value for the "Packet interval" in Gigabit Ethernet networks is around 20 μs . Higher values can reduce the transfer speed of the GigE uc480 considerably.

Methods

Method	Description
Get	Returns the packet interval of the image transfer in microseconds (μs).
GetDefault	Returns the default value for packet interval.
GetRange	Returns the value range for the packet interval of image transfers in microseconds (μs).
GetSupported	Returns if setting the packet interval is supported.
Set	Sets the packet interval for the image transfer in microseconds (μs).

Class

[uc480.GigEPacketInterval](#)

Accessible

Camera.GigE.Transfer.PacketInterval

Syntax

```
uc480.GigEPacketInterval.Get(out uint u32Value)
```

Description

Returns the packet interval of the image transfer in microseconds (μs).

Parameter

- `u32Value`: Returns the current packet interval.

Class

[uc480.GigEPacketInterval](#)

Accessible

Camera.GigE.Transfer.PacketInterval

Syntax

`uc480.GigEPacketInterval.GetDefault(out uint u32DefaultValue)`

Description

Returns the default value for packet interval.

Parameter

- `u32Value`: Returns the default packet interval.

Class

[uc480.GigEPacketInterval](#)

Accessible

Camera.GigE.Transfer.PacketInterval

Syntax

`uc480.GigEPacketInterval.GetRange(out uc480.Types.Range<uint> range)`
`uc480.GigEPacketInterval.GetRange(out uint u32Min, out uint u32Max, out uint u32Inc)`

Description

Returns the value range for the packet interval of image transfers in microseconds (μ s).

Parameter

range	Minimum: Returns the minimum value.
	Maximum: Returns the maximum value.
	Increment: Returns the increment.
u32Min	Returns the minimum value.
u32Max	Returns the maximum value.
u32Inc	Returns the increment.

Class

[uc480.GigEPacketInterval](#)

Accessible

Camera.GigE.Transfer.PacketInterval

Syntax

`uc480.GigEPacketInterval.GetSupported(out bool bSupported)`

Description

Returns if setting the packet interval is supported.

Parameter

bSupported	1 = Packet interval is supported. 0 = Packet interval is not supported.
------------	--

Class[uc480.GigEPacketInterval](#)**Accessible**

Camera.GigE.Transfer.PacketInterval

Syntax

uc480.GigEPacketInterval.Set(uint u32Value)

Description

Sets the packet interval for the image transfer in microseconds (μ s).

Parameter

- u32Value: Packet interval to be set.

3.1.1.15 Hdr

The `Hdr` class provides methods for controlling the HDR mode (HDR = High Dynamic Range) of some uc480 cameras. The following class and methods exist:

- [Kneepoints](#)

Methods

Method	Description
GetSupported	Returns if the HDR mode is supported.
SetEnable	Enables/disables the HDR mode of some sensors.

3.1.1.15.1 Kneepoints

The `Kneepoints` class provides methods for controlling the keepoint of the HDR mode. The HDR mode is enabled/disabled with [SetEnable\(\)](#).



HDR mode is only supported by sensors of the UI-122x/UI-322x/UI-522x camera models.

Methods

Method	Description
Get	Returns the current set kneepoints.
GetInfo	Returns information about the kneepoints.
Set	Sets the kneepoints.

Class[uc480.HdrKneepoints](#)**Accessible**

Camera.Hdr.Kneepoints

Syntax

```
uc480.HdrKneepoints.Get(out uc480.Types.Point<double>[ ] KneePointList)
```

Description

Some sensors support HDR mode (high dynamic range). You can use [SetEnable\(\)](#) to enable/disable it. Using `Get()`, you can query the currently set knee points.



HDR mode is only supported by sensors of the UI-122x/UI-322x/UI-522x camera models.

Parameter

KneePointList	x: Kneepoint X value
	y: Kneepoint Y value

Class

[uc480.HdrKneepoints](#)

Accessible

Camera.Hdr.Kneepoints

Syntax

```
uc480.HdrKneepoints.GetInfo(out uc480.Types.KneePointInformation KneePointInfo)
```

Description

Some sensors support HDR mode (high dynamic range). You can use [SetEnable\(\)](#) to enable/disable it. Using `Kneepoints.GetInfo()`, you can query information on the knee points.



HDR mode is only supported by sensors of the UI-122x/UI-322x/UI-522x camera models.

Parameter

- `KneepointInfo`: Returns [uc480.Types.KneePointInformation](#)

Class

[uc480.HdrKneepoints](#)

Accessible

Camera.Hdr.Kneepoints

Syntax

```
uc480.HdrKneepoints.Set(uc480.Types.Point<double>[ ] KneePointList)
```

Description

Sets the kneepoints of the supported HDR mode (high dynamic range) for some sensors. The HDR mode is enabled/disabled with [SetEnable\(\)](#).



HDR mode is only supported by sensors of the UI-122x/UI-322x/UI-522x camera models.

Parameter

KneePointList	x: Kneepoint X value
	y: Kneepoint Y value

3.1.1.15.2 GetSupported**Class**[uc480.Hdr](#)**Accessible**

Camera.Hdr

Syntax`uc480.Hdr.GetSupported(out bool bSupported)`**Description**

Some sensors support HDR (high dynamic range) mode. You can use [SetEnable\(\)](#) to enable/disable it. Using `GetSupported()`, you can query the HDR mode supported by the sensor.

Parameter

bSupported	1 = HDR mode is supported. 0 = HDR mode is not supported.
------------	--

3.1.1.15.3 SetEnable**Class**[uc480.Hdr](#)**Accessible**

Camera.Hdr

Syntax`uc480.Hdr.SetEnable(bool bEnable)`**Description**

Enables/disables the HDR mode of some sensors.



HDR mode is only supported by sensors of the UI-122x/UI-322x/UI-522x camera models.

Parameter

bEnable	1 = Enable HDR mode 0 = Disable HDR mode
---------	---

3.1.1.16 Hotpixel

The `Hotpixel` class provides methods for the configuration of the sensor hot pixel correction. The correction is performed by the software. The hot pixel list is stored in the camera's non-volatile EEPROM. Some sensor models can also correct hot pixels directly in the sensor. More information for the hot pixel correction can be found in the

uc480 manual.



This correction will not work with subsampling or with binning factors greater than 2.

The following class and methods exist:

- [Camera](#)
- [Factory](#)
- [Software](#)

Methods

Method	Description
DisableSensorCorrection	Disables hot pixel correction.
EnableSensorCorrection	Enables hot pixel correction.
GetCorrectionMode	Returns the currently set hot pixel correction mode.
GetMergedList	Returns the number of hot pixels in a merged list.
GetSupportedCorrectionMode	Returns the supported hot pixel correction modes.
SetEnableSensorCorrection	Enables/disables the sensor's own hot pixel correction function.

3.1.1.16.1 Camera

The `Camera` class provides methods for the configuration of the user-defined hot pixel correction. The hot pixel list is stored in the camera's non-volatile EEPROM. More information for the hot pixel correction can be found in the uc480 manual.



This correction will not work with subsampling or with binning factors greater than 2.

Methods

Method	Description
ClearList	Deletes the user-defined hot pixel list from the camera EEPROM.
GetList	Returns the user-defined hot pixel list stored in the camera EEPROM.
GetListExists	Indicates whether the user-defined hot pixel list exists in the camera EEPROM.
GetListMaxNumber	Returns the maximum number of hot pixels that the user can store in the camera EEPROM.
GetListNumber	Returns the number of hot pixels in the user-defined hot pixel list stored in the camera EEPROM.
SetEnable	Enables hot pixel correction using the hot pixel list(s) stored in the camera EEPROM.
SetList	Sets the user-defined hot pixel list stored in the camera EEPROM.

Class

[uc480.HotpixelsCameraList](#)

Accessible

Camera.Hotpixels.Camera

Syntax

`uc480.HotpixelsCameraList.ClearList()`

Description

Deletes the user-defined hot pixel list from the camera EEPROM.

Parameter

None

Class

[uc480.HotpixelsCameraList](#)

Accessible

Camera.Hotpixels.Camera

Syntax

```
uc480.HotpixelsCameraList.GetList(out uc480.Types.Point<short>[] List)
uc480.HotpixelsCameraList.GetList(out short[] s16List)
```

Description

Returns the user-defined hot pixel list stored in the camera EEPROM.

Parameter

- `List/s16List`: Returns an array with the user-defined hot pixel list.

Class[uc480.HotpixelsCameraList](#)**Accessible**

Camera.Hotpixels.Camera

Syntax

uc480.HotpixelsCameraList.GetListExists(out bool bExist)

Description

Indicates whether the user-defined hot pixel list exists in the camera EEPROM.

Parameter

bExists	1 = User-defined hot pixel list exists
	0 = User-defined hot pixel list does not exist

Class[uc480.HotpixelsCameraList](#)**Accessible**

Camera.Hotpixels.Camera

Syntax

uc480.HotpixelsCameraList.GetListMaxNumber(out int s32Number)

Description

Returns the maximum number of hot pixels that the user can store in the camera EEPROM.

Parameter

- **s32Number:** Returns the maximum number of user-defined hot pixels.

Class[uc480.HotpixelsCameraList](#)**Accessible**

Camera.Hotpixels.Camera

Syntax

uc480.HotpixelsCameraList.GetListNumber(out int s32Number)

Description

Returns the number of hot pixels in the user-defined hot pixel list stored in the camera EEPROM.

Parameter

- **s32Number:** Returns the number.

Class

[uc480.HotpixelsCameraList](#)

Accessible

Camera.Hotpixels.Camera

Syntax

`uc480.HotpixelsCameraList.SetEnable(bool bEnable)`

Description

Enables hot pixel correction using the hot pixel list(s) stored in the camera EEPROM.

Parameter

bEnable	1 = Enable hot pixel correction 0 = Disable hot pixel correction
---------	---

Class

[uc480.HotpixelsCameraList](#)

Accessible

Camera.Hotpixels.Camera

Syntax

`uc480.HotpixelsCameraList.SetList(uc480.Types.Point<short>[] List)
uc480.HotpixelsCameraList.SetList(short[] List)`

Description

Sets the user-defined hot pixel list stored in the camera EEPROM.

Parameter

- `List`: Array which sets the user-defined hot pixel list.

3.1.1.16.2 Factory

The `Factory` class provides methods for getting the factory-set hot pixel list. More information for the hot pixel correction can be found in the uc480 manual.



This correction will not work with subsampling or with binning factors greater than 2.

Methods

Method	Description
GetList	Returns the factory-set hot pixel list.
GetListExists	Indicates whether the factory-set hot pixel list exists.

Class

[uc480.HotpixelsFactoryList](#)

Accessible

Camera.Hotpixels.Factory

Syntax

```
uc480.HotpixelFactoryList.GetList(out uc480.Types.Point<short>[] List)
uc480.HotpixelFactoryList.GetList(out short[] s16List)
```

Description

Returns the factory-set hot pixel list.

Parameter

- List/s16List: Returns an array with the factory-set hot pixel list.

Class

[uc480.HotpixelFactoryList](#)

Accessible

Camera.Hotpixel.Factory

Syntax

```
uc480.HotpixelFactoryList.GetListExists(out bool bExist)
```

Description

Indicates whether the factory-set hot pixel list exists.

Parameter

bExists	1 = Factory-set hot pixel list exists
	0 = Factory-set hot pixel list does not exist

3.1.1.16.3 Software

The `Software` class provides methods for the configuration of the user's hot pixel list. More information for the hot pixel correction can be found in the uc480 manual.



This correction will not work with subsampling or with binning factors greater than 2.

Methods

Method	Description
GetList	Returns the user-defined hot pixel list stored in the computer.
GetListExists	Indicates whether the user-defined hot pixel list exists in the computer.
LoadList	Loads the user-defined hot pixel list from a file.
SaveList	Saves the user-defined hot pixel list to a file.
SetEnable	Enables hot pixel correction using the user's hot pixel list stored in the computer.
SetList	Sets the user-defined hot pixel list that is stored in the computer.

Class

[uc480.HotpixelsoftwareList](#)

Accessible

Camera.Hotpixelsoftware

Syntax

```
uc480.HotpixelsoftwareList.GetList(out uc480.Types.Point<short>[] List)
uc480.HotpixelsoftwareList.GetList(out short[] s16List)
```

Description

Returns the user-defined hot pixel list stored in the computer.

Parameter

- `List/s16List`: Returns an array with the user-defined hot pixel list.

Class

[uc480.HotpixelsoftwareList](#)

Accessible

Camera.Hotpixelsoftware

Syntax

```
uc480.HotpixelsoftwareList.GetListExists(out bool bExist)
```

Description

Indicates whether the user-defined hot pixel list exists in the computer.

Parameter

bExists	1 = User-defined hot pixel list exists
	0 = User-defined hot pixel list does not exist

Class

[uc480.HotpixelsoftwareList](#)

Accessible

Camera.Hotpixelsoftware

Syntax

```
uc480.HotpixelsoftwareList.LoadList(string strFile)
```

Description

Loads the user-defined hot pixel list from a file. The method can also be used with Unicode file names.

Parameter

- `strFile`: File with the user-defined list.

Class[uc480.HotpixelsoftwareList](#)**Accessible**

Camera.Hotpixel.Software

Syntax

uc480.HotpixelsoftwareList.SaveList(string strFile)

Description

Saves the user-defined hot pixel list to a file. The method can also be used with Unicode file names

Parameter

- strFile: Path to the file for saving the user-defined hot pixel list.

Class[uc480.HotpixelsoftwareList](#)**Accessible**

Camera.Hotpixel.Software

Syntax

uc480.HotpixelsoftwareList.SetEnable(bool bEnable)

Description

Enables hot pixel correction using the user's hot pixel list stored in the computer. This requires the user hot pixel list to be set (see [SetList\(\)](#))

Parameter

bEnable	1 = Enable hot pixel correction
	0 = Disable hot pixel correction

Class[uc480.HotpixelsoftwareList](#)**Accessible**

Camera.Hotpixel.Software

Syntax

```
uc480.HotpixelsoftwareList.SetList(uc480.Types.Point<short>[] List)  
uc480.HotpixelsoftwareList.SetList(short[] List)
```

Description

Sets the user-defined hot pixel list that is stored in the computer.

Parameter

- List: Array which passes the user-defined hot pixel list.

3.1.1.16.4 DisableSensorCorrection

Class

[uc480.Hotpixel](#)

Accessible

Camera.Hotpixel

Syntax

`uc480.Hotpixel.DisableSensorCorrection()`

Description

Disables hot pixel correction.

Parameter

None

3.1.1.16.5 EnableSensorCorrection

Class

[uc480.Hotpixel](#)

Accessible

Camera.Hotpixel

Syntax

`uc480.Hotpixel.EnableSensorCorrection()`

Description

Enables hot pixel correction.

Parameter

None

3.1.1.16.6 GetCorrectionMode

Class

[uc480.Hotpixel](#)

Accessible

Camera.Hotpixel

Syntax

`uc480.Hotpixel.GetCorrectionMode(out uc480.Defines.CORRECTIONMODE mode)`

Description

Returns the currently set hot pixel correction mode.

Parameter

mode	Returns the current hot pixel correction mode: <ul style="list-style-type: none">• uc480.Defines.CorrectionMode.CameraCorrection: Hot pixel correction is possible via the hot pixel list in the camera EEPROM.• uc480.Defines.CorrectionMode.SoftwareCorrection: Hot pixel correction is possible via the user-defined hot pixel list.• uc480.Defines.CorrectionMode.SensorCorrection: Hot pixel correction is possible via the sensor-internal hot pixel correction.• uc480.Defines.CorrectionMode.Disabled: Hot pixel correction is disabled.
------	---

3.1.1.16.7 GetMergedList**Class**[uc480.Hotpixel](#)**Accessible**

Camera.Hotpixel

Syntax`uc480.Hotpixel.GetMergedList(out short[] s16List)`**Description**

Returns the combined (factory-set and user-defined) hot pixel list.

Parameter

- s16List: Returns an array with the combined hot pixel list.

3.1.1.16.8 GetSupportedCorrectionMode**Class**[uc480.Hotpixel](#)**Accessible**

Camera.Hotpixel

Syntax`uc480.Hotpixel.GetSupportedCorrectionMode(out uc480.Defines.CorrectionMode mode)`**Description**

Returns the supported hot pixel correction modes.

Parameter

mode	Returns the supported hot pixel correction modes. The return value is a bitmask with the following constants (combined by OR): <ul style="list-style-type: none">• uc480.Defines.CorrectMode.CameraCorrection: Hot pixel correction is possible via the hot pixel list in the camera EEPROM.• uc480.Defines.CorrectMode.SoftwareCorrection: Hot pixel correction is possible via the user-defined hot pixel list.• uc480.Defines.CorrectMode.SensorCorrection: Hot pixel correction is possible via the sensor-internal hot pixel correction.• uc480.Defines.CorrectMode.Disabled: Hot pixel correction is disabled.
------	---

3.1.16.9 SetEnableSensorCorrection

Class

[uc480.Hotpixel](#)

Accessible

Camera.Hotpixel

Syntax

`uc480.Hotpixel.SetEnableSensorCorrection(bool bEnable)`

Description

Enables sensor's own hot pixel correction function (if available).

Parameter

bEnable	1 = Enable sensor hot pixel correction 0 = Disable sensor hot pixel correction
---------	---

3.1.17 I2C

The `I2C` class provides methods for writing and reading over the I²C bus of some uc480 cameras.

Methods

Method	Description
Read	Reads data over the I ² C bus of some uc480 board level cameras.
Write	Writes data over the I ² C bus of some uc480 board level cameras.

3.1.17.1 Read



`I2C.Read()` is only supported by PCB versions of the USB uc480 ME/LE camera series.

Class

[uc480.I2C](#)

Accessible

Camera.I2C

Syntax

```
uc480.I2C.Read(int s32DeviceAddr, int s32RegisterAddr, ref byte[] byData)
```

Description

Reads data over the I^C bus of some uc480 board level cameras.

The uc480 processes I^C 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 `nDeviceAddr` are assigned to the uc480 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

Additional for model UI-336x/UI-536x and UI-337x/UI-537x: 0x82

For information on the signals applied to the I^C bus, refer to uc480 manual.

Parameter

<code>s32DeviceAddr</code>	Slave device address (7 bit format)
<code>s32RegisterAddr</code>	Address of a 8 bit register (only 8-bit addresses are valid)
<code>byData</code>	Data to be read

3.1.1.17.2 Write



`uc480.I2C.Write()` is only supported by PCB versions of the USB uc480 ME/LE camera series.

Class

[uc480.I2C](#)

Accessible

Camera.I2C

Syntax

```
uc480.I2C.Write(int s32DeviceAddr, int s32RegisterAddr, byte[] byData)
```

Description

Writes data via the I^C bus of a board level camera.

The uc480 processes I^C 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 `nDeviceAddr` are assigned to the uc480 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

Additional for model UI-336x/UI-536x and UI-337x/UI-537x: 0x82

For information on the signals applied to the I2C bus, refer to the uc480 manual.



If you write data on a micro controller via the uc480 camera, make sure that the micro controller confirms the transfer with an ACK message.

Parameter

s32DeviceAddr	Device address (7 bit format)
s32RegisterAddr	Address of a 8 bit register (only 8-bit addresses are valid)
byData	Data to be written

3.1.1.18 Image

The `Image` class provides methods for returning color values of an image, loading an image from a file or saving to a file.

Methods

Method	Description
Convert	Converts a raw Bayer image in the desired format.
GetHistogram	Computes the histogram of the submitted image.
GetValues	Returns the RGB values of a specific position of the image.
Load	Loads an image file (BMP, JPG, PNG).
Save	Save an image to a file.

3.1.1.18.1 Measure

The `Measure` class allows the measurement of the sharpness in a defined AOI of the current image. To get a sharpness value the edges in the image are evaluated. The sharpness can only be indicated as a relative value as it depends on the edges in the current image. An image with less edges will not reach the sharpness value of an image with a lot of edges.

The higher the value, the better the sharpness. The value can be used in comparative measurements to detect changes in the image acquisition of the same object, e.g. caused by readjusted lenses.

Methods

Method	Description
Inquire	Returns information of the set AOI, e.g. the sharpness.
SetAOI	Sets an AOI in which the sharpness is measured. In the image are up to 5 AOIs possible. These AOIs can also overlap.
SetPreset	Sets different predefined AOIs in the image.

Class

[uc480.ImageMeasure](#)

Accessible

Camera.Image.Measure

Syntax

```
uc480.ImageMeasure.Inquire(uint s32AoiNum, out System.Drawing.Rectangle rectAoi, out uint u32Sharp)
```

Description

Returns information of the set AOI, e.g. the sharpness.

Parameter

s32AoiNum	ID of the AOI
rectAoi	Position and size of the AOI
u32Sharp	Relative sharpness value in the defined AOI

Class

[uc480.ImageMeasure](#)

Accessible

Camera.Image.Measure

Syntax

```
uc480.ImageMeasure.SetAOI(uint u32AoiNum, System.Drawing.Rectangle rectAoi)
```

Description

Sets an AOI in which the sharpness is measured. In the image are up to 5 AOIs possible. These AOIs can also overlap.

Parameter

s32AoiNum	ID of the AOI
rectAoi	Position and size of the AOI

Class

[uc480.ImageMeasure](#)

Accessible

Camera.Image.Measure

Syntax

```
uc480.ImageMeasure.SetPreset(uc480.Defines.Measure ePreset)
```

Description

Sets different predefined AOIs in the image.

Parameter

- **ePreset:** Predefined AOI for the sharpness measurement (in each of the four image corners and in the center)

3.1.1.18.2 Convert

Class

[uc480.Image](#)

Accessible

Camera.Image

Syntax

```
uc480.Image.Convert(uc480.Types.ConversionParameter ConversionParam)
```

Description

Converts a raw Bayer image in the desired format. You can set all parameters, which are important for software conversion:

- Pixel format
- Pixel converter (3×3, 5×5)
- Color correction
- Gamma
- Saturation
- Edge enhancement

The target buffer must be allocated with [Allocate\(\)](#) and must have the right size.

Parameter

- ConversionParam: Returns [uc480.Types.ConversionParameter](#)

3.1.1.18.3 GetHistogram

Class

[uc480.Image](#)

Accessible

Camera.Image

Syntax

```
uc480.Image.GetHistogram(int s32MemID, uc480.Defines.ColorMode colorMode, out uint[] u32HistoMem)
```

Description

Computes the histogram of the submitted image. The histogram always contains 256 values per channel. For color modes with a bit depth of more than 8 bits, the system evaluates the 8 most significant bits (MSBs).

Note: The method `GetHistogram` supports the following color formats:

- uc480.Defines.ColorMode.SensorRaw8
- uc480.Defines.ColorMode.Mono8
- uc480.Defines.ColorMode.RGBY8Packed
- uc480.Defines.ColorMode.BGRY8Packed
- uc480.Defines.ColorMode.RGBA8Packed
- uc480.Defines.ColorMode.BGRA8Packed
- uc480.Defines.ColorMode.BGR565Packed
- uc480.Defines.ColorMode.BGR5Packed



- uc480.Defines.ColorMode.RGB8Packed
- uc480.Defines.ColorMode.BGR8Packed

Parameter

s32MemID	Image memory ID
colorMode	Color mode of the image with the s32MemID memory ID For a list of all available color formats and the associated input parameters, see the PixelFormat class or the Appendix "Color and memory formats" in the uc480 manual.
u32HistoMem	Returns an array: The array must be allocated in such a way that it can accommodate 3*256 values for color formats and in raw Bayer mode. In monochrome mode, the array must be able to accommodate 1*256 values.

3.1.1.18.4 GetValues

Class

[uc480.Image](#)

Accessible

Camera.Image

Syntax

```
uc480.Image.GetValues(int32 s32MemID, uc480.Defines.ColorMode colorMode, int s32X, int s32Y,
                      out int s32Red, out int s32Green, out int s32Blue)
uc480.Image.GetValues(int32 s32MemID, uc480.Defines.ColorMode colorMode, int s32X, int s32Y,
                      out int s32Mono)
```

Description

Returns the RGB values of a specific position of the image.

Parameter

s32MemID	Image memory ID
colorMode	Sets the color mode (see GetSupported()).
s32X	X position
s32Y	Y position
s32Red	Returns the red value.
s32Green	Returns the green value.
s32Blue	Returns the blue value.
s32Mono	Only monochrome formats: returns the gray value

3.1.1.18.5 Load

Class

[uc480.Image](#)

Accessible

Camera.Image

Syntax

```
uc480.Image.Load(string strFilename)
uc480.Image.Load(string strFilename, out int s32MemID)
```

Description

Loads an image file. The image must be BMP, JPEG or PNG format. The image is loaded into the active image memory or in a newly allocated image memory.

Parameter

strFilename	Name of the file to be loaded (Unicode). If <code>NULL</code> is passed, the "Open file" dialog opens.
s32MemID	The image is loaded into a new allocated image memory. This memory must be released using Free() .

3.1.18.6 Save

Class

[uc480.Image](#)

Accessible

Camera.Image

Syntax

```
uc480.Image.Save(string strFilename)
uc480.Image.Save(string strFilename, System.Drawing.Imaging.ImageFormat format)
uc480.Image.Save(string strFilename, uint u32Format)
uc480.Image.Save(string strFilename, System.Drawing.Imaging.ImageFormat format, int s32Quality)
uc480.Image.Save(string strFilename, uint u32Format, int s32Quality)
uc480.Image.Save(string strFilename, int s32MemID)
uc480.Image.Save(string strFilename, int s32MemID, System.Drawing.Imaging.ImageFormat format)
uc480.Image.Save(string strFilename, int s32MemID, uint u32Format)
uc480.Image.Save(string strFilename, int s32MemID, System.Drawing.Imaging.ImageFormat format, int s32Quality)
uc480.Image.Save(string strFilename, int s32MemID, uint u32Format, int s32Quality)
```

Description

Save an image to a file. The image must be BMP, JPEG or PNG format. The image is read-out from the active image memory or the specified image memory.



When saving an image [Freeze\(\)](#) should not be called before with `uc480.Defines.DeviceParameter.DontWait`, because the image acquisition might not be completed.

The bitmap is stored with the color depth that was used when allocating the image memory (in DIB mode) or that was set for the current color mode (in Direct3D mode). You can save images with a bit depth of more than 8 bit in the PNG format. 12 bit formats are converted into 16 bit. JPEG files are always saved with a color depth of 8 or 24 bits.



In Direct3D mode, overlay data is not saved.

Parameter

strFilename	Name of the file to be saved (Unicode). If <code>NULL</code> is passed, the "Save as" dialog opens.
format/u32Format	File format to be saved: <ul style="list-style-type: none"> • <code>System.Drawing.Imaging.ImageFormat.Bmp</code> • <code>System.Drawing.Imaging.ImageFormat.Jpeg</code> • <code>System.Drawing.Imaging.ImageFormat.Png</code>
s32MemID	Image memory ID
s32Quality	Sets the image quality for JPEG and PNG (and therefore the compression). The higher the value, the better the quality is: <ul style="list-style-type: none"> • 100 = maximum quality, that means no compression • If the parameter is set to 0, the default value of 75 is used. For BMP the parameter is ignored.

3.1.1.19 ImageStabilization

The `ImageStabilization` class provides methods for controlling the electronic image stabilization function of some uc480 cameras.



This class is currently only supported by the UI-1008XS models.

- Enable automatic exposure time control for best image stabilization results (see [Software](#)).
- The image is stabilized electronically using part of the field of view for movement compensation. When this function is enabled, the visible field of view is reduced by about 10 %.
- The image stabilization feature at the same time activates the UI-1008XS camera's anti flicker mode.
- If you have mounted the camera (e.g. on a stand) so that it is vibration-free, you should disable the image stabilization function. Otherwise, slight image background movement could occur.

Methods

Method	Description
<u>GetEnable</u>	Returns the image stabilization status.
<u>GetSupported</u>	Returns the image stabilization function is supported by the camera.
<u>SetEnable</u>	Enables/disables the image stabilization.

3.1.1.19.1 GetEnable

Class

[uc480.ImageStabilization](#)

Accessible

`Camera.ImageStabilization`

Syntax

```
uc480.ImageStabilization.GetEnable(out bool bEnable)
```

Description

Returns the image stabilization status.



This class is currently only supported by the USB uc480 XS camera family.

Parameter

bEnable	1 = Image stabilization enabled
	0 = Image stabilization disabled

3.1.1.19.2 GetSupported

Class

[uc480.ImageStabilization](#)

Accessible

Camera.ImageStabilization

Syntax

```
uc480.ImageStabilization.GetSupported(out bool bSupported)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Returns the image stabilization function is supported by the camera.

Parameter

bSupported	1 = Camera supports image stabilization
	0 = Camera does not support image stabilization

3.1.1.19.3 SetEnable

Class

[uc480.ImageStabilization](#)

Accessible

Camera.ImageStabilization

Syntax

```
uc480.ImageStabilization.SetEnable(bool bEnable)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Enables/disables image stabilization.

Parameter

bEnable	1 = Enable image stabilization
	0 = Disable image stabilization

3.1.1.20 Information

The `Information` class provides methods for querying information about uc480 cameras and other hardware or errors while image acquisition etc.

Methods

Method	Description
GetAvgBandwidth	Returns the average bandwidth.
GetBusSpeed	Queries whether a camera is connected to a USB 2.0 or USB 3.0 host controller.
GetCameraInfo	Returns the data hard-coded in the EEPROM.
GetCameraStatus	Returns various status information and settings.
GetCaptureStatus	Returns information on errors that occurred during an image capture.
GetDeviceInfo	Returns information about connected USB 3 and GigE uc480 cameras.
GetEnableErrorReport	Get the status (enabled/disabled) for error event logging.
GetImageInfo	Provides additional information on the images you take.
GetLastError	Returns the last error that occurred and returns the associated error code and message.
GetPeakBandwidth	Returns the peak bandwidth.
GetSensorInfo	Returns information about the sensor type used in the camera.
GetUsedBandwidth	Returns the bus bandwidth (in MByte/s) currently used by all initialized or selected cameras.
ResetCaptureStatus	Resets the capture status infomation.
SetCameraStatus	Sets various status information and camera settings.
SetEnableErrorReport	Enables/disables error event logging.

3.1.1.20.1 GetAvgBandwidth

Class

[uc480.Information](#)

Accessible

Camera.Information

Syntax

```
uc480.Information.GetAvgBandwidth(out double f64Value)
```

Description

Returns the average bandwidth.

Parameter

- `f64Value`: Returns the average bandwidth.

3.1.1.20.2 GetBusSpeed

Class

[uc480.Information](#)

Accessible

Camera.Information

Syntax

```
uc480.Information.GetBusSpeed(out int s32Value)
```

Description

Queries whether a camera is connected to a USB 2.0 or USB 3.0 host controller. You can see in the IDS Camera Manager below "General Information" which kind of USB host controller are available on your PC.

Parameter

- `s32Value`: Returns the USB host controller.

3.1.1.20.3 GetCameraInfo

Class

[uc480.Information](#)

Accessible

Camera.Information

Syntax

```
uc480.Information.GetCameraInfo(out uc480.Types.CameraInfo cameraInfo)
```

Description

Returns the data hard-coded in the EEPROM.



The serial number or model name should not be used to find a specific camera (e.g. in order to control this specific camera). If you use the serial number, the software may not find the serial number after exchanging the camera. The model name can be changed when updating the camera driver.

Instead, we recommend identifying a camera by a fixed camera ID, the camera type

or by the sensor ID ([GetCameraList\(\)](#)). The advantage of the camera ID is that you can set it manually. That means if you exchange a camera, you can set the same camera ID for the new camera.

For technical reasons, the following values for `CAMINFO::Type` are internally redirected to the same value:



`uc480.Defines.BoardType.Usb_SE` and `uc480.Defines.BoardType.Usb_RE`
as well as

`uc480.Defines.BoardType.Eth_SE` and `uc480.Defines.BoardType.Eth_RE`

You can use [GetSensorInfo\(\)](#) to discern the camera models USB uc480 SE and RE.

Parameter

- `cameraInfo`: Returns [uc480.Types.CameraInfo](#)

3.1.1.20.4 GetCameraStatus

Class

[uc480.Information](#)

Accessible

`Camera.Information`

Syntax

```
uc480.Information.GetCameraStatus(uc480.Defines.CameraStatus cameraStatus, out int s32Status)
```

Description

Returns various status information and settings.

Parameter

cameraStatus	<ul style="list-style-type: none"> uc480.Defines.CameraStatus.FIFOOverflowCount: Number of FIFO overruns. Is increased if image data gets lost because the USB bus is congested. uc480.Defines.CameraStatus.SequenceCount: Returns the sequence count. For Capture(), this parameter is set to 0. Each time the sequence buffer (image counter) changes, the counter is increased by 1. uc480.Defines.CameraStatus.SequenceSize: Returns the number of sequence buffers. uc480.Defines.CameraStatus.ExternalTriggerEventCount: Returns the camera's internal count of external trigger events. uc480.Defines.CameraStatus.TriggerMissed: Returns the number of unprocessed trigger signals. Is reset to 0 after each call. uc480.Defines.CameraStatus.LastCaptureError: Returns the last image capture error. For a list of all possible error events, see GetCaptureStatus(). uc480.Defines.CameraStatus.ParameterExists: Indicates whether a parameter set is present on the camera EEPROM (read-only). See also Parameter. uc480.Defines.CameraStatus.ParametersSet1: Indicates whether parameter set 1 including camera settings is present on the camera (read-only). See also Parameter. uc480.Defines.CameraStatus.ParametersSet2: Indicates whether parameter set 2 including camera settings is present on the camera (read-only). See also Parameter. uc480.Defines.CameraStatus.Standby: Sets the camera to standby mode. uc480.Defines.CameraStatus.StandbySupported: Queries whether the camera supports standby mode (read-only).
s32Status	Returns the information specified by cameraStatus.

3.1.1.20.5 GetCaptureStatus

Class

[uc480.Information](#)

Accessible

Camera.Information

Syntax

```
uc480.Information.GetCaptureStatus(out uc480.Types.CaptureStatus CaptureStatus)
```

Description

Returns information on errors that occurred during an image capture. All errors are listed that occurred since the last reset of the method.

Parameter

- **CaptureStatus:** Returns [uc480.Types.CaptureStatus](#)

3.1.1.20.6 GetDeviceInfo

Class

[uc480.Information](#)

Accessible

Camera.Information

Syntax

```
uc480.Information.GetDeviceInfo(out uc480.Types.ETH.DeviceInformation EthInfo)  
uc480.Information.GetDeviceInfo(out uc480.Types.DeviceInformation Info)
```

Description

Returns information about connected USB 3 and GigE uc480 cameras. The resulting information is written to the `ETH_DEVICE_INFO` structure. For this purpose, the cameras need not be opened.

Parameter

- **EthInfo/Info:** Returns information about connected USB 3 and GigE uc480 cameras, see [uc480.Types.ETH...](#)

3.1.1.20.7 GetEnableErrorReport

Class

[uc480.Information](#)

Accessible

Camera.Information

Syntax

```
uc480.Information.GetEnableErrorReport(out bool bEnable)
```

Description

Get the status (enabled/disabled) for error event logging. If error reporting is enabled, errors will automatically be displayed in a dialog box. Canceling the dialog box disables the error report. Even with disabled error reporting, you can still query errors using [GetLastError\(\)](#).

Parameter

bEnable	1 = Error report is enabled. 0 = Error report is disabled.
---------	---

3.1.1.20.8 GetImageInfo

Class

[uc480.Information](#)

Accessible

Camera.Information

Syntax

```
uc480.Information.GetImageInfo(int s32MemId, out uc480.Types.ImageInfo imageInfo)
```

Description

Provides additional information on the images you take. The method 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 `GetImageInfo()` directly after receiving the `uc480.Camera.EventFrame` event.

Using with GigE uc480 cameras

`TimestampTick` returns the camera timestamp, which indicates the time of image capture with an accuracy of 0.1 µs. 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 "Camera and sensor data" chapter in uc480 manual.
- The time when the sensor starts to output image data in freerun mode. 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.

`TimestampSystem` 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 `TimestampSystem`.

To determine the exact interval between two image captures, it is therefore recommended to read out the camera timestamp.

Using with USB uc480 cameras

The timestamp returns the time when image data transfer to the PC was completed. The `TimestampSystem` 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 `Lock()`. 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.

Parameter

s32MemId	Image memory ID
imageInfo	See uc480.Types.ImageInfo

3.1.1.20.9 GetLastError

Class

[uc480.Information](#)

Accessible

Camera.Information

Syntax

```
uc480.Information.GetLastError(out uc480.Defines.Status status, out string strText)  
uc480.Information.GetLastError(out int s32Error, out string strText)
```

Description

Returns the last error that occurred and returns the associated error code and message. We recommend to use this method after an error has occurred that returned `uc480.Defines.Status.NO_SUCCESS`. Each error message will be overwritten when a new error occurs.

Parameter

status/s32Error	Returns the error code.
strText	Returns the error text.

3.1.1.20.10 GetPeakBandwidth

Class

[uc480.Information](#)

Accessible

Camera.Information

Syntax

```
uc480.Information.GetPeakBandwidth(out double f64Value)
```

Description

Returns the peak bandwidth.

Parameter

- `f64Value`: Returns the peak bandwidth.

3.1.1.20.11 GetSensorInfo

Class

[uc480.Information](#)

Accessible

Camera.Information

Syntax

```
uc480.Information.GetSensorInfo(out uc480.Types.SensorInfo sensorInfo)
```

Description

Returns information about the sensor type used in the camera. `uc480.Defines.Sensor` provides a complete up-to-date list of all supported sensor types.

Parameter

- `sensorInfo`: Returns [uc480.Types.SensorInfo](#)

3.1.1.20.12 GetUsedBandwidth**Class**[uc480.Information](#)**Accessible**

Camera.Information

Syntax`uc480.Information.GetUsedBandwidth(out int s32Value)`**Description**

Returns the bus bandwidth (in MByte/s) currently used by all initialized 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.

Parameter

- `s32Value`: Returns the bus bandwidth.

3.1.1.20.13 ResetCaptureStatus**Class**[uc480.Information](#)**Accessible**

Camera.Information

Syntax`uc480.Information.ResetCaptureStatus()`**Description**

Resets the capture status infomation.

Parameter

None

3.1.1.20.14 SetCameraStatus**Class**[uc480.Information](#)**Accessible**

Camera.Information

Syntax`uc480.Information.SetCameraStatus(uc480.Defines.CameraStatus cameraStatus, int s32Status)`**Description**

Sets various status information and camera settings.

Parameter

cameraStatus	Camera status information, see GetCameraStatus()
s32Status	Sets the information specified by cameraStatus.

3.1.1.20.15 SetEnableErrorReport**Class**[uc480.Information](#)**Accessible**

Camera.Information

Syntax`uc480.Information.SetEnableErrorReport(bool bEnable)`**Description**

Enables/disables error event logging. If error reporting is enabled, errors will automatically be displayed in a dialog box. Canceling the dialog box disables the error report. Even with disabled error reporting, you can still query errors using [GetLastError\(\)](#).



`SetEnableErrorReport()` can be called before initializing the camera by calling [Init\(\)](#).

You only need to enable `SetEnableErrorReport()` once for all cameras in the application.

Parameter

bEnable	1 = Enable error report
	0 = Disable error report

3.1.1.21 IO

The `IO` class provides classes for controlling the flash function of uc480 cameras. For some uc480 cameras also the general purpose I/O (GPIO), pulse-width modulation (PWM) or LED functions can be controlled. The following classes exist:

- [Flash](#)
- [Gpio](#)
- [Led](#)
- [Pwm](#)

3.1.1.21.1 Flash

The `Flash` class provides methods for controlling all flash functions.

- Rolling shutter cameras:

Using `Flash`, 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 method returns `uc480.Defines.Status.NO_SUCCESS`.



To use a rolling shutter camera with the global start function, first call [Get\(\)](#). 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. Flash determines the required delay in order to synchronize 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 chapters Camera basics: "Shutter methods", "Digital input/output (trigger/flash)" and "Operating modes" in the uc480 manual.

Note on the accuracy of flash synchronization

The following parameters have an influence on the camera's internal timing:



- Image geometry (CMOS and CCD sensors)
- Pixel clock (CMOS and CCD sensors)
- Exposure time (CCD sensors)

If you change any of these parameters, you will have to set the flash duration and flash delay parameters once again.

Methods

Method	Description
ApplyGlobalParams	Returns the parameters for the global exposure window and sets them as flash parameters.
GetAutoFreerunDefault	Returns the default auto flash setting in freerun mode.
GetAutoFreerunEnable	Returns the current auto flash setting in freerun mode.
GetDelayRange	Returns the range for flash delay.
GetDurationRange	Returns the range for flash duration.
GetGlobalParams	Returns the parameters for the global exposure window.
GetMode	Returns the current flash mode.
GetParams	Returns the current values for flash delay and duration.
GetSupported	Returns the GPIOs which can be used for flash output.
SetAutoFreerunEnable	Enables/disables the auto flash in freerun mode.
SetMode	Sets the flash mode.
SetParams	Sets the values for flash delay and duration.

Class

[uc480.IOFlash](#)

Accessible

Camera.IO.Flash

Syntax

```
uc480.IOFlash.ApplyGlobalParams()
```

Description

Returns the parameters for the global exposure window and sets them as flash parameters.

Parameter

None

Class

[uc480.IOFlash](#)

Accessible

Camera.IO.Flash

Syntax

```
uc480.IOFlash.GetAutoFreerunDefault(bool freerun)
```

Description

Returns the default auto flash setting in freerun mode.

Parameter

- `freerun`: Returns the default setting.

Class

[uc480.IOFlash](#)

Accessible

Camera.IO.Flash

Syntax

```
uc480.IOFlash.GetAutoFreerunEnable(out bool enable)
```

Description

Returns the current auto flash setting in freerun mode.

Parameter

<code>enable</code>	1 = Automatic flash in freerun mode is enabled
	0 = Automatic flash in freerun mode is disabled

Class

[uc480.IOFlash](#)

Accessible

Camera.IO.Flash

Syntax

```
uc480.IOFlash.GetDelayRange(out Types.Range<uint> range)
uc480.IOFlash.GetDelayRange(out uint u32Min, out uint u32Max, out uint u32Inc)
```

Description

Returns the range for flash delay.

Parameter

range	Minimum: returns the minimum value Maximum: returns the maximum value Increment: returns the increment
u32Min	Returns the minimum value.
u32Max	Returns the maximum value.
u32Inc	Returns the increment.

Class

[uc480.IOFlash](#)

Accessible

Camera.IO.Flash

Syntax

```
uc480.IOFlash.GetDurationRange(out Types.Range<uint> range)
uc480.IOFlash.GetDurationRange(out uint u32Min, out uint u32Max, out uint u32Inc)
```

Description

Returns the range for flash duration.

Parameter

range	Minimum: returns the minimum value Maximum: returns the maximum value Increment: returns the increment
u32Min	Returns the minimum value.
u32Max	Returns the maximum value.
u32Inc	Returns the increment.

Class

[uc480.IOFlash](#)

Accessible

Camera.IO.Flash

Syntax

```
uc480.IOFlash.GetGlobalParams(out uint u32GlobalDelay, out uint u32GlobalDuration)
```

Description

Returns the parameters for the global exposure window.

Parameter

u32GlobalDelay	Returns the global delay.
u32GlobalDuration	Returns the global duration.

Class[uc480.IOFlash](#)**Accessible**

Camera.IO.Flash

Syntax`uc480.IOFlash.GetMode(out uc480.Defines.IO.FlashMode mode)`**Description**

Returns the current flash mode.

Parameter

mode	<p>Returns the current mode:</p> <ul style="list-style-type: none"> • <code>uc480.Defines.IO.FlashMode.Off</code>: Disables the digital output. • <code>uc480.Defines.IO.FlashMode.TriggerLowActive</code>: Enables the flash strobe in trigger mode. The digital output is set to low level for the flash duration. • <code>uc480.Defines.IO.FlashMode.TriggerHighActive</code>: Enables the flash strobe in trigger mode. The digital output is set to high level for the flash duration. • <code>uc480.Defines.IO.FlashMode.ConstantHigh</code>: Statically sets the digital output to high level (HIGH). • <code>uc480.Defines.IO.FlashMode.ConstantLow</code>: Statically sets the digital output to low level (LOW). • <code>uc480.Defines.IO.FlashMode.FreerunLowActive</code>: Enables the flash strobe in freerun mode. The digital output is set to low level for the flash duration. • <code>uc480.Defines.IO.FlashMode.FreerunHighActive</code>: Enables the flash strobe in freerun mode. The digital output is set to high level for the flash duration.
------	---

Class[uc480.IOFlash](#)**Accessible**

Camera.IO.Flash

Syntax`uc480.IOFlash.GetParams(out uint u32Delay, out uint u32Duration)`**Description**

Returns the current values for flash delay and duration (in μ s).

Parameter

u32Delay	Returns the delay.
u32Duration	Returns the duration.

Class[uc480.IOFlash](#)**Accessible**

Camera.IO.Flash

Syntax

```
uc480.IOFlash.GetSupported(out uc480.Defines.IO.GPIO supported)
```

Description

Returns the GPIOs which can be used for flash output.

Parameter

supported	Returns the GPIO which can be used for flash output: <ul style="list-style-type: none"> • uc480.Defines.IO.GPIO.One • uc480.Defines.IO.GPIO.Two
-----------	--

Class[uc480.IOFlash](#)**Accessible**

Camera.IO.Flash

Syntax

```
uc480.IOFlash.SetAutoFreerunEnable(bool enable)
```

Description

Enables/disables the auto flash in freerun mode. It may take a few images until the flash timing is adjusted.

Parameter

enable	1 = Enable automatic flash in freerun mode. 0 = Disable automatic flash in freerun mode.
--------	---

Class[uc480.IOFlash](#)**Accessible**

Camera.IO.Flash

Syntax

```
uc480.IOFlash.SetMode(uc480.Defines.IO.FlashMode mode)
```

Description

Sets the flash mode.

Parameter

mode/u32Value	Mode to be set: <ul style="list-style-type: none"> • uc480.Defines.IO.FlashMode.Off: Disables the digital output. • uc480.Defines.IO.FlashMode.TriggerLowActive: Enables the flash strobe in trigger mode. The digital output is set to low level for the flash duration. • uc480.Defines.IO.FlashMode.TriggerHighActive: Enables the flash strobe in trigger mode. The digital output is set to high level for the flash duration. • uc480.Defines.IO.FlashMode.ConstantHigh: Statically sets the digital output to high level (HIGH). • uc480.Defines.IO.FlashMode.ConstantLow: Statically sets the digital output to low level (LOW). • uc480.Defines.IO.FlashMode.FreerunLowActive: Enables the flash strobe in freerun mode. The digital output is set to low level for the flash duration. • uc480.Defines.IO.FlashMode.FreerunHighActive: Enables the flash strobe in freerun mode. The digital output is set to high level for the flash duration.
---------------	--

Class[uc480.IOFlash](#)**Accessible**

Camera.IO.Flash

Syntax`uc480.IOFlash.SetParams(uint u32Delay, uint u32Duration)`**Description**

Sets the values for flash delay and duration (in μ s).

Parameter

u32Delay	Sets the delay.
u32Duration	Sets the duration.

3.1.21.2 Gpio

The `Gpio` class provides methods for controlling the general purpose I/O (GPIO) of some uc480 cameras.



The GPIOs are only available on some uc480 camera series. The GPIOs are not provided with optocouplers and use TTL/LVCMS voltages. For information on GPIO wiring, please refer to the electrical specifications in the uc480 manual.

Methods

Method	Description
GetDirection	Returns the input/output state of the GPIOs.
GetState	Returns the state of the GPIO.
GetSupported	Returns the supported GPIOs.
SetDirection	Set the GPIO on input/output.
SetState	Sets the state of the GPIOs if they are defined as output.

Class

[uc480.IOGpio](#)

Accessible

Camera.IO.Gpio

Syntax

```
uc480.IOGpio.GetDirection(uc480.Defines.IO.GPIO gpio, out uc480.Defines.IO.Direction direction)
```

Description

Returns the input/output direction of the GPIOs.

Parameter

gpio	GPIO which direction is to be returned: <ul style="list-style-type: none"> • uc480.Defines.IO.GPIO.One • uc480.Defines.IO.GPIO.Two
direction	Returns the direction of the GPIO: <ul style="list-style-type: none"> • uc480.Defines.IO.Direction.In • uc480.Defines.IO.Direction.Out

Example

```
// set directionif gpio1 to out
statusRet = Camera.IO.IOGpio.SetDirection(uc480.Defines.IO.GPIO.One, uc480.Defines.IO.Direction.Out);

// set direction of gpio1 and gpio2 to in
statusRet = Camera.IO.IOGpio.SetDirection(uc480.Defines.IO.GPIO.One | uc480.Defines.IO.GPIO.Two, uc480.Defines.IO.Direction.In);

// get direction of gpio1
uc480.Defines.IO.Direction currentDirection;
statusRet = Camera.IO.IOGpio.GetDirection(uc480.Defines.IO.GPIO.One, out currentDirection);
```

Class

[uc480.IOGpio](#)

Accessible

Camera.IO.Gpio

Syntax

```
uc480.IOGpio.GetState(uc480.Defines.IO.GPIO gpio, out uc480.Defines.IO.State state)
```

Description

Returns the state of the GPIO (high, low).

Parameter

gpio	GPIO which state is to be returned: • uc480.Defines.IO.GPIO.One • uc480.Defines.IO.GPIO.Two
state	Returns the state of the GPIO: • uc480.Defines.IO.State.High • uc480.Defines.IO.State.Low

Example

```
// set state if gpio1 to high
statusRet = Camera.IO.IOGpio.SetState(uc480.Defines.IO.GPIO.One, uc480.Defines.IO.State.High);

// set state of gpio1 and gpio2 to low
statusRet = Camera.IO.IOGpio.SetState(uc480.Defines.IO.GPIO.One | uc480.Defines.IO.GPIO.Two, uc480.Defines.IO.State.Low);

// get state of gpio1
uc480.Defines.IO.State currentState;
statusRet = Camera.IO.IOGpio.GetState(uc480.Defines.IO.GPIO.One, out currentState);
```

Class[uc480.IOGpio](#)**Accessible**

Camera.IO.Gpio

Syntax

uc480.IOGpio.GetSupported(uc480.Defines.IO.GPIO Gpio, out bool bSupported, out bool bInput, out bool bOutput)

Description

Returns the supported GPIOs.

Parameter

Gpio	GPIO to be queried: • uc480.Defines.IO.GPIO.One • uc480.Defines.IO.GPIO.Two
bSupported	1 = Supported 0 = Not supported
bInput	Returns if input is supported: 1 = Supported 0 = Not supported
bOutput	Returns if output is supported: 1 = Supported 0 = Not supported

Example

```
Boolean bSupported, bIn, bOut;
statusRet = Camera.IO.IOGpio.GetSupported(uc480.Defines.IO.GPIO.One, out bSupported, out bIn, out bOut);
```

Class

[uc480.IOGpio](#)

Accessible

Camera.IO.Gpio

Syntax

```
uc480.IOGpio.SetDirection(uc480.Defines.IO.GPIO gpio, uc480.Defines.IO.Direction direction)
```

Description

Set the GPIO on input/output.

Parameter

gpio	GPIO which direction is to be set: <ul style="list-style-type: none"> • uc480.Defines.IO.GPIO.One • uc480.Defines.IO.GPIO.Two
direction	Sets the direction of the GPIO: <ul style="list-style-type: none"> • uc480.Defines.IO.Direction.In • uc480.Defines.IO.Direction.Out

Example

```
// set direction if gpio1 to out
statusRet = Camera.IO.IOGpio.SetDirection(uc480.Defines.IO.GPIO.One, uc480.Defines.IO.Direction.Out);

// set direction of gpio1 and gpio2 to in
statusRet = Camera.IO.IOGpio.SetDirection(uc480.Defines.IO.GPIO.One | uc480.Defines.IO.GPIO.Two, uc480.Defines.IO.Direction.In);

// get direction of gpio1
uc480.Defines.IO.Direction currentDirection;
statusRet = Camera.IO.IOGpio.GetDirection(uc480.Defines.IO.GPIO.One, out currentDirection);
```

Class

[uc480.IOGpio](#)

Accessible

Camera.IO.Gpio

Syntax

```
uc480.IOGpio.SetState(uc480.Defines.IO.GPIO gpio, uc480.Defines.IO.State state)
```

Description

Sets the state of the GPIOs if they are defined as output (high, low). Before setting the state you should define the direction of the GPIO via [SetDirection\(\)](#).

Parameter

gpio	GPIO which state is to be set: <ul style="list-style-type: none"> • uc480.Defines.IO.GPIO.One • uc480.Defines.IO.GPIO.Two
state	Sets the state of the GPIO: <ul style="list-style-type: none"> • uc480.Defines.IO.State.High • uc480.Defines.IO.State.Low

Example

```
// set state of gpio1 to high  
statusRet = Camera.IO.IOGpio.SetState(uc480.Defines.IO.GPIO.One, uc480.Defines.IO.State.High);  
  
// set state of gpio1 and gpio2 to low  
statusRet = Camera.IO.IOGpio.SetState(uc480.Defines.IO.GPIO.One | uc480.Defines.IO.GPIO.Two, uc480.Defines.IO.State.Low);  
  
// get state of gpio1  
uc480.Defines.IO.State currentState;  
statusRet = Camera.IO.IOGpio.GetState(uc480.Defines.IO.GPIO.One, out currentState);
```

3.1.1.21.3 Led

The `Led` class provides methods for toggling the color of the LED on the back of the USB uc480 SE/RE camera housing.

Methods

Method	Description
Get	Returns the state of the LED.
Set	Sets the state of the LED.
Toggle	Toggles between the LED states.

Class

[uc480.IOLed](#)

Accessible

Camera.IO.Led

Syntax

```
uc480.IOLed.Get(out uc480.Defines.IO.LedState state)
```

Description

Returns the state of the LED, see [Set\(\)](#).

Parameter

- `state`: Returns the state.

Class

[uc480.IOLed](#)

Accessible

Camera.IO.Led

Syntax

```
uc480.IOLed.Set(uc480.Defines.IO.LedState state)
```

Description

Sets the state of the LED.

Parameter

<code>state</code>	State to be set: <ul style="list-style-type: none">• <code>uc480.Defines.IO.LedState.One</code>: Sets LED to orange.• <code>uc480.Defines.IO.LedState.Two</code>: Sets LED to green.
--------------------	---

Class

[uc480.IOLed](#)

Accessible

Camera.IO.Led

Syntax

`uc480.IOLed.Toggle()`

Description

Toggles between the LED states.

Parameter

None

3.1.1.21.4 Pwm

The `Pwm` class provides methods for controlling the pulse-width modulation (PWM). PWM is a way of altering a digital signal. A pulse width modulated signal is a rectangular signal with a fixed frequency (i.e. distance between pulses) but varying pulse duration ("duty cycle"). For example, you can control the brightness of a light source with a fast PWM signal: When using a short duty cycle, the light source is dim. When using a longer duty cycle, the light source gets brighter.

Methods

Method	Description
GetDutyCycleRange	Returns the range for the duty cycle.
GetFrequencyRange	Returns the range for the frequency.
GetMode	Returns the current PWM mode.
GetParams	Returns the current values of the PWM parameters.
GetSupported	Returns the GPIOs which can be used for PWM.
SetMode	Sets the current PWM mode.
SetParams	Sets the current values of the PWM parameters.

Class

[uc480.IOPwm](#)

Accessible

Camera.IO.Pwm

Syntax

```
uc480.IOPwm.GetDutyCycleRange(out uc480.Types.Range<double> range)
uc480.IOPwm.GetDutyCycleRange(out double f64Min, out double f64Max, out double f64Inc)
```

Description

Returns the range for the duty cycle.

Parameter

range	Minimum: returns the minimum value
	Maximum: returns the maximum value
	Increment: returns the increment
f64Min	Returns the minimum value for the duty cycle.
f64Max	Returns the maximum value for the duty cycle.
f64Inc	Returns the increment.

Class[uc480.IOPwm](#)**Accessible**

Camera.IO.Pwm

Syntax

```
uc480.IOPwm.GetFrequencyRange(out uc480.Types.Range<double> range)
uc480.IOPwm.GetFrequencyRange(out double f64Min, out double f64Max, out double f64Inc)
```

Description

Returns the range for the frequency.

Parameter

range	Minimum: returns the minimum value
	Maximum: returns the maximum value
	Increment: returns the increment
f64Min	Returns the minimum value for the frequency.
f64Max	Returns the maximum value for the frequency.
f64Inc	Returns the increment.

Class[uc480.IOPwm](#)**Accessible**

Camera.IO.Pwm

Syntax

```
uc480.IOPwm.GetMode(out uc480.Defines.IO.PwmMode mode)
```

Description

Returns the current PWM mode.

Parameter

- mode: Returns the current mode, see [SetMode\(\)](#).

Class

[uc480.IOPwm](#)

Accessible

Camera.IO.Pwm

Syntax

```
uc480.IOPwm.GetParams(out double f64Frequency, out double f64DutyCycle)
```

Description

Returns the current values of the PWM parameters.

Parameter

f64Frequency	Returns the current frequency.
f64DutyCycle	Returns the current duty cycle.

Class

[uc480.IOPwm](#)

Accessible

Camera.IO.Pwm

Syntax

```
uc480.IOPwm.GetSupported(out uc480.Defines.IO.GPIO supported)
```

Description

Returns the GPIOs which can be used for PWM.

Parameter

- supported: Returns the supported GPIOs (uc480.Defines.IO.GPIO.One / uc480.Defines.IO.GPIO.Two)

Class

[uc480.IOPwm](#)

Accessible

Camera.IO.Pwm

Syntax

```
uc480.IOPwm.SetMode(uc480.Defines.IO.PwmMode mode)
```

Description

Sets the current PWM mode.

Parameter

mode	Mode to be set: <ul style="list-style-type: none"> uc480.Defines.IO.PwmMode.Flash: Sets the flash output as output for PWM mode. uc480.Defines.IO.PwmMode.Gpio1: Sets GPIO 1 as output. uc480.Defines.IO.PwmMode.Gpio2: Sets GPIO 2 as output.
------	--

Class

[uc480.IOPwm](#)

Accessible

Camera.IO.Pwm

Syntax

`uc480.IOPwm.SetParams(double f64Frequency, double f64DutyCycle)`

Description

Sets the current values of the PWM parameters.

Parameter

<code>f64Frequency</code>	Sets the frequency value.
<code>f64DutyCycle</code>	Sets the duty cycle value.

3.1.1.22 Lut

The `Lut` class provides methods for setting the hardware or software LUT for uc480 cameras. This LUT 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.

Each lookup table (LUT) for the uc480 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.

The following class and methods exist:

- [Preset](#)
- [Raw](#)

Methods

Method	Description
GetEnable	Enables/disables the LUT.
GetMode	Returns the current set LUT mode.
GetState	Returns the current state of the LUT.
GetStateInfo	Returns current state information of the LUT.
GetSupportedInfo	Returns the current support information of the LUT
GetValuesComplete	Returns the LUT values set by the user after the gamma, contrast and brightness values have been taken into account.
GetValuesPreset	Returns the predefined LUT.
GetValuesUser	Returns the LUT values set by the user without modifications.
Load	Loads and sets the LUT from a file.
Save	Saves a set LUT into a file.
SetEnable	Enables/disables the LUT.
SetMode	Sets the LUT calculation mode.
SetPreset	Sets a predefined LUT.
SetValuesUser	Sets the LUT values of the user.

3.1.1.22.1 Preset

The `Preset` class provides methods for setting a predefined LUT.

Methods

Method	Description
Astrol	Predefined LUT, false-color display of the image
ColdHot	Predefined LUT, false-color display of the image
Glow1	Predefined LUT, false-color display of the image
Glow2	Predefined LUT, false-color display of the image
Hot	Predefined LUT, false-color display of the image
Identity	Predefined LUT, linear LUT, no image modifications
Map1	Predefined LUT, false-color display of the image
Negativ	Predefined LUT, inverts the image
OnlyBlue	Predefined LUT, shows only the blue channel of the image
OnlyGreen	Predefined LUT, shows only the green channel of the image
OnlyRed	Predefined LUT, shows only the red channel of the image
Rainbow1	Predefined LUT, false-color display of the image
Sepic	Predefined LUT, uses sepia toning for coloring the image

Class

[uc480.LutPreset](#)

Accessible

Camera.Lut.Preset

Syntax

`uc480.LutPreset.Astrol()`

Description

Sets a predefined LUT, false-color display of the image.

Parameter

None

Class

[uc480.LutPreset](#)

Accessible

`Camera.Lut.Preset`

Syntax

`uc480.LutPreset.ColdHot()`

Description

Sets a predefined LUT, false-color display of the image.

Parameter

None

Class

[uc480.LutPreset](#)

Accessible

`Camera.Lut.Preset`

Syntax

`uc480.LutPreset.Glow1()`

Description

Sets a predefined LUT, false-color display of the image.

Parameter

None

Class

[uc480.LutPreset](#)

Accessible

`Camera.Lut.Preset`

Syntax

`uc480.LutPreset.Glow2()`

Description

Sets a predefined LUT, false-color display of the image.

Parameter

None

Class

[uc480.LutPreset](#)

Accessible

Camera.Lut.Preset

Syntax

`uc480.LutPreset.Hot()`

Description

Sets a predefined LUT, false-color display of the image.

Parameter

None

Class

[uc480.LutPreset](#)

Accessible

Camera.Lut.Preset

Syntax

`uc480.LutPreset.Identity()`

Description

Sets a predefined LUT, linear LUT, no image modifications.

Parameter

None

Class

[uc480.LutPreset](#)

Accessible

Camera.Lut.Preset

Syntax

`uc480.LutPreset.Map1()`

Description

Sets a predefined LUT, false-color display of the image.

Parameter

None

Class

[uc480.LutPreset](#)

Accessible

Camera.Lut.Preset

Syntax

`uc480.LutPreset.Negativ()`

Description

Sets a predefined LUT, inverts the image.

Parameter

None

Class

[uc480.LutPreset](#)

Accessible

Camera.Lut.Preset

Syntax

`uc480.LutPreset.OnlyBlue()`

Description

Sets a predefined LUT, shows only the blue channel of the image.

Parameter

None

Class

[uc480.LutPreset](#)

Accessible

Camera.Lut.Preset

Syntax

`uc480.LutPreset.OnlyGreen()`

Description

Sets a predefined LUT, shows only the green channel of the image.

Parameter

None

Class

[uc480.LutPreset](#)

Accessible

Camera.Lut.Preset

Syntax

`uc480.LutPreset.OnlyRed()`

Description

Sets a predefined LUT, shows only the red channel of the image.

Parameter

None

Class

[uc480.LutPreset](#)

Accessible

Camera.Lut.Preset

Syntax

```
uc480.LutPreset.Rainbow1()
```

Description

Sets a predefined LUT, false-color display of the image.

Parameter

None

Class

[uc480.LutPreset](#)

Accessible

Camera.Lut.Preset

Syntax

```
uc480.LutPreset.Sepic()
```

Description

Sets a predefined LUT, uses sepia toning for coloring the image.

Parameter

None

3.1.1.22.2 Raw

The `Raw` class provides methods for setting the use of the camera LUT in combination with RAW formats.

Methods

Method	Description
GetEnable	Returns if the camera LUT can be used in combination with RAW formats.
SetEnable	Enables the use of the camera LUT with RAW formats. If the value 1 is passed, the camera LUT can also be used with RAW formats. The default value is 0, i.e. the feature is disabled.

Class

[uc480.LutRaw](#)

Accessible

Camera.Lut.Raw

Syntax

`uc480.LutRaw.GetEnable(out bool enable)`

Description

Returns if the camera LUT can be used in combination with RAW formats.

Parameter

enable	1 = LUT is enabled with RAW formats
	0 = LUT is disabled with RAW formats

Class

[uc480.LutRaw](#)

Accessible

Camera.Lut.Raw

Syntax

`uc480.LutRaw.SetEnable(bool enable)`

Description

Enables the use of the camera LUT with RAW formats. If the value 1 is passed, the camera LUT can also be used with RAW formats. The default value is 0, i.e. the feature is disabled.

Parameter

enable	1 = Enable LUT
	0 = Disable LUT

3.1.1.22.3 GetEnable

Class

[uc480.Lut](#)

Accessible

Camera.Lut

Syntax

`uc480.Lut.GetEnable(out bool bEnable)`

Description

Enables/disables the LUT.

Parameter

bEnable	1 = LUT is enabled
	0 = LUT is disabled

3.1.1.22.4 GetMode

Class

[uc480.Lut](#)

Accessible

Camera.Lut

Syntax

```
uc480.Lut.GetMode(out uc480.Defines.LutMode mode)
```

Description

Returns the current set LUT mode.

Parameter

- mode: Returns the set LUT mode (see [SetMode\(\)](#))

3.1.1.22.5 GetState

Class

[uc480.Lut](#)

Accessible

Camera.Lut

Syntax

```
uc480.Lut.GetState(out uc480.Defines.LutState state)
```

Description

Returns the current state of the LUT.

Parameter

state	uc480.Defines.LutState.HardwareGamma: Gamma is calculated in hardware
	uc480.Defines.LutState.HardwareLut: LUT is calculated in hardware
	uc480.Defines.LutState.HardwareLutAndGamma: LUT and gamma are calculated in hardware
	uc480.Defines.LutState.SoftwareGamma: Gamma is calculated in software
	uc480.Defines.LutState.SoftwareLut: LUT is calculated in software
	uc480.Defines.LutState.SoftwareLutAndGamma: LUT and gamma are calculated in software
	uc480.Defines.LutState.Inactive: LUT is inactive (No gamma and no LUT is set)
	uc480.Defines.LutState.NotSupported: LUT is not supported with the current settings

3.1.1.22.6 GetStateInfo

Class

[uc480.Lut](#)

Accessible

Camera.Lut

Syntax

```
uc480.Lut.GetStateInfo(out uc480.Types.LutState stateInfo)
```

Description

Returns current state information of the LUT.

Parameter

- stateInfo: Returns the LUT state information, see [uc480.Types.LutState](#)

3.1.1.22.7 GetSupportedInfo

Class

[uc480.Lut](#)

Accessible

Camera.Lut

Syntax

```
uc480.Lut.GetSupportedInfo(out uc480.Types.LutSupportInfo supportInfo)
```

Description

Returns the current support information of the LUT.

Parameter

- supportInfo: LUT support information, see [uc480.Types.LutSupportInfo](#)

3.1.1.22.8 GetValuesComplete

Class

[uc480.Lut](#)

Accessible

Camera.Lut

Syntax

```
uc480.Lut.GetValuesComplete(out double[] f64Red, out double[] f64Green, out double[] f64Blue)
```

Description

Returns the LUT values set by the user after the gamma, contrast and brightness values have been taken into account. For an description of the array see [SetValuesUser\(\)](#).

Parameter

f64Red	Returns an array with the red channel values or the gray-scale values of the LUT.
f64Green	Returns an array with the green channel values of the LUT.
f64Blue	Returns an array with the blue channel values of the LUT.

3.1.1.22.9 GetValuesPreset**Class**[uc480.Lut](#)**Accessible**

Camera.Lut

Syntax`uc480.Lut.GetValuesPreset(uc480.Defines.LutPreset preset, out double[] f64Red, out double[] f64Green, out double[] f64Blue)`**Description**

Returns the predefined LUT.

Parameter

preset	A predefined LUT, see SetPreset()
f64Red	Returns an array with the red channel values or the gray-scale values of the predefined LUT.
f64Green	Returns an array with the green channel values of the predefined LUT.
f64Blue	Returns an array with the blue channel values of the predefined LUT.

3.1.1.22.10 GetValuesUser**Class**[uc480.Lut](#)**Accessible**

Camera.Lut

Syntax`uc480.Lut.GetValuesUser(out double[] f64Red, out double[] f64Green, out double[] f64Blue)`**Description**

Returns the LUT values set by the user without modifications. For an description of the array see [SetValuesUser\(\)](#).

Parameter

f64Red	Returns an array with the red channel values or the gray-scale value (GigE uc480 SE cameras) of the LUT are written.
f64Green	Returns an array with the green channel values of the LUT are written.
f64Blue	Returns an array with the blue channel values of the LUT are written.

3.1.1.22.11 Load**Class**[uc480.Lut](#)**Accessible**

Camera.Lut

Syntax

```
uc480.Lut.Load(string lutFile)
```

Description

Loads and sets the LUT from a file.

Parameter

- `lutFile`: LUT file to be set

3.1.1.22.12 Save**Class**[uc480.Lut](#)**Accessible**

Camera.Lut

Syntax

```
uc480.Lut.Save(string lutFile)
```

Description

Saves a set LUT into a file.

Parameter

- `lutFile`: LUT file to be saved

3.1.1.22.13 SetEnable**Class**[uc480.Lut](#)**Accessible**

Camera.Lut

Syntax

```
uc480.Lut.SetEnable(bool bEnable)
```

Description

Enables the LUT. If no other LUT has been defined, the system sets the linear LUT as specified by [Identity\(\)](#).

Parameter

bEnable	1 = Enable LUT 0 = Disable LUT
---------	-----------------------------------

3.1.1.22.14 SetMode

Class

[uc480.Lut](#)

Accessible

Camera.Lut

Syntax

```
uc480.Lut.SetMode(uc480.Defines.LutMode mode)
```

Description

Sets the LUT calculation mode.

Parameter

mode	uc480.Defines.LutMode.Default: Default mode: If supported the LUT/gamma is done on hardware otherwise in software uc480.Defines.LutMode.ForceHardware: LUT/gamma in hardware (if not possible: uc480.Defines.LutState.NotSupported) uc480.Defines.LutMode.ForceSoftware: IS_LUT_MODE_ID_FORCE_SOFTWARE: LUT/gamma in software (if not possible: uc480.Defines.LutState.NotSupported)
------	---

3.1.1.22.15 SetPreset

Class

[uc480.Lut](#)

Accessible

Camera.Lut

Syntax

```
uc480.Lut.SetPreset(uc480.Defines.LutPreset preset)
```

Description

Sets a predefined LUT.

Parameter

preset	uc480.Defines.LutPreset.Astro1: false-color display of the image
	uc480.Defines.LutPreset.Glow1: false-color display of the image
	uc480.Defines.LutPreset.Glow2: false-color display of the image
	uc480.Defines.LutPreset.Hot: false-color display of the image
	uc480.Defines.LutPreset.Identity: linear LUT, no image modifications
	uc480.Defines.LutPreset.Map1: false-color display of the image
	uc480.Defines.LutPreset.Negative: inverts the image
	uc480.Defines.LutPreset.OnlyBlue: shows only the blue channel of the image
	uc480.Defines.LutPreset.OnlyGreen: shows only the green channel of the image
	uc480.Defines.LutPreset.OnlyRed: shows only the red channel of the image

3.1.1.22.16 SetValuesUser**Class**[uc480.Lut](#)**Accessible**

Camera.Lut

Syntax

uc480.Lut.SetValuesUser(double[] f64Red, double[] f64Green, double[] f64Blue)

Description

Sets the LUT values. The `f64Red`, `f64Green` and `f64Blue` arrays contain double values between 0.0 and 1.0.

Parameter

<code>f64Red</code>	Array containing the red channel values or the gray-scale value of the LUT.
<code>f64Green</code>	Array containing the green channel values of the LUT.
<code>f64Blue</code>	Array containing the blue channel values of the LUT.

3.1.1.23 Memory

The `Memory` class provides methods for managing image memory. The following classes and methods exist:

- [ImageBuffer](#)
- [Sequence](#)

Methods

Method	Description
Allocate	Allocates an image memory for an image
CopyImageMem	Copies the contents of the image memory to a memory area.
CopyImageMemLines	Copies a number of lines indicated out of the image memory to a memory area.
CopyToArray	Copies the image of the passed image memory to an array.
CopyToBitmap	Copies the image of the passed image memory to a bitmap.
Free	Releases an image memory and removes it from the driver management.
GetActive	Returns the starting address and the ID number of the active image memory.
GetBitsPerPixel	Returns the bits per pixel for the passed image memory.
GetHeight	Returns the height of the image in the image memory.
GetLast	Returns the last ID in the image memory.
GetList	Returns the list of image memory IDs.
GetLocked	Returns if the passed image memory is locked.
GetPitch	Returns the line increment (in bytes).
GetSize	Returns the size of the image in the image memory.
GetWidth	Returns the width of the image in the image memory.
Inquire	Returns the properties of an allocated image memory.
Lock	Locks the image memory defined by the ID.
SetActive	Makes the specified image memory the active memory.
ToBitmap	Returns a bitmap which contains the image.
ToIntPtr	Returns a pointer to the image memory of the active/given image memory ID.
Unlock	Unlocks the image memory defined by the ID.

3.1.1.23.1 ImageBuffer

The `ImageBuffer` class allows accessing the captured images in the camera memory of a GigE uc480 camera.

Methods

Method	Description
<u>GetEnable</u>	Returns if the camera memory mode is supported.
<u>GetImageRange</u>	Returns the information about a specific iteration, e.g. the number of images in the iteration.
<u>GetIterationRange</u>	Returns the number of iterations which are currently in the camera memory.
<u>GetSupported</u>	Returns if memory mode is supported by the camera.
<u>ReleaseIteration</u>	Releases all iterations up to the given ID in the camera memory.
<u>SetEnable</u>	Enables the camera memory mode.
<u>TransferImage</u>	Transfers an image from an iteration in the camera memory to the user buffer on the PC.

Class

[uc480.MemoryImageBuffer](#)

Accessible

Camera.Memory.ImageBuffer

Syntax

```
uc480.MemoryImageBuffer.GetEnable(out bool bEnable)
```

Description

Returns if the camera memory mode is supported.

Parameter

bEnable	1 = Camera memory is enabled. 0 = Camera memory is disabled.
---------	---

Class

[uc480.MemoryImageBuffer](#)

Accessible

Camera.Memory.ImageBuffer

Syntax

```
uc480.MemoryImageBuffer.GetImageRange(uint u32Iteration, out int s32First, out int s32Last)  
uc480.MemoryImageBuffer.GetImageRange(uint u32Iteration, out uc480.Types.Range<int> imageRange)
```

Description

Returns the information about a specific iteration, e.g. the number of images in the iteration.

Parameter

u32Iteration	Iteration ID
s32First	First image ID
s32Last	Last image ID

Class[uc480.MemoryImageBuffer](#)**Accessible**

Camera.Memory.ImageBuffer

Syntax

```
uc480.MemoryImageBuffer.GetIterationRange(out int s32First, out int s32Last)
uc480.MemoryImageBuffer.GetIterationRange(out uc480.Types.Range<int> iterationRange)
```

Description

Returns the number of iterations which are currently in the camera memory.

Parameter

s32First	First iteration ID
s32Last	Last iteration ID

Class[uc480.MemoryImageBuffer](#)**Accessible**

Camera.Memory.ImageBuffer

Syntax

```
uc480.MemoryImageBuffer.GetSupported(out bool bSupported)
```

Description

Returns if memory mode is supported by the camera.

Parameter

bSupported	1 = Camera memory mode is supported
	0 = Camera memory mode is not supported

Class[uc480.MemoryImageBuffer](#)**Accessible**

Camera.Memory.ImageBuffer

Syntax

```
uc480.MemoryImageBuffer.ReleaseIteration(uint u32Iteration)
```

Description

Releases all iterations up to the given ID in the camera memory.

Parameter

- `u32Iteration`: All iterations below the given iteration ID are released.

Class[uc480.MemoryImageBuffer](#)**Accessible**

Camera.Memory.ImageBuffer

Syntax`uc480.MemoryImageBuffer.SetEnable(bool bEnable)`**Description**

Enables the camera memory mode.

Parameter

<code>bEnable</code>	1 = Enable camera memory
	0 = Disable camera memory

Class[uc480.MemoryImageBuffer](#)**Accessible**

Camera.Memory.ImageBuffer

Syntax`uc480.MemoryImageBuffer.TransferImage(uint u32Iteration, int s32Image)`**Description**

Transfers an image from an iteration in the camera memory to the user buffer on the PC.

Parameter

<code>u32Iteration</code>	Iteration ID
<code>s32Image</code>	Image ID

3.1.1.23.2 Sequence

The `Sequence` class provides methods for controlling the image memory in the ring buffering.

Methods

Method	Description
<u>Add</u>	Adds an image memory to the list of image memories used for ring buffering.
<u>Clear</u>	Removes all image memories from the sequence list.
<u>ExitImageQueue</u>	Deletes an image queue.
<u>GetActive</u>	Determines the image memory which is currently used for capturing an image.
<u>GetLast</u>	Returns the last used sequence memory ID.
<u>GetList</u>	Returns an array with all sequence memory IDs.
<u>GetLocked</u>	Returns if the passed sequence memory is locked.
<u>InitImageQueue</u>	Enables the queue mode for existing image memory sequences.
<u>Lock</u>	Locks write access to an image memory within a sequence.
<u>ToMemory</u>	Returns the image memory ID for the passed sequence memory ID.
<u>Unlock</u>	Unlocks a previously locked sequence memory.
<u>WaitForNextImage</u>	Returns the sequence ID of the first (i.e. oldest) image in a memory sequence.

Class

[uc480.MemorySequence](#)

Accessible

Camera.Memory.Sequence

Syntax

```
uc480.MemorySequence.Add(int s32MemId)
uc480.MemorySequence.Add(int[] s32MemIdList)
```

Description

Adds an image memory to the list of image memories used for ring buffering. The image memory must have been previously requested using [Allocate\(\)](#). Using [SetActive\(\)](#), 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 color depth (bits per pixel).

Parameter

- `s32MemId/s32MemIdList`: Image memory IDs

Class

[uc480.MemorySequence](#)

Accessible

Camera.Memory.Sequence

Syntax

`uc480.MemorySequence.Clear()`

Description

Removes all image memories from the sequence list that were added using [AddId\(\)](#). After a call of `MemorySequence.Clear()`, there is no more active image memory. To make an image memory the active memory, call [SetActive\(\)](#).

Parameter

None

Class

[uc480.MemorySequence](#)

Accessible

Camera.Memory.Sequence

Syntax

`uc480.MemorySequence.ExitImageQueue()`

Description

Deletes a queue which has been initialized with [InitImageQueue\(\)](#) and discards all information about the order of queued images. The image memories will be unlocked. The memory sequence itself persists and can be deleted with [Clear\(\)](#).

Parameter

None

Class

[uc480.MemorySequence](#)

Accessible

Camera.Memory.Sequence

Syntax

`uc480.MemorySequence.GetActive(out int s32SeqID)`

Description

Determines the image memory which is currently active. This method is only available if you have enabled ring buffering.



This number is not the ID of the image memory that was allocated using [Allocate\(\)](#), but the running number from the order in which memory was allocated by [AddId\(\)](#). You can use [ToMemory\(\)](#) to get the image memory ID.

Parameter

- `s32SeqID`: Returns the ID of the active sequence memory.

Class[uc480.MemorySequence](#)**Accessible**

Camera.Memory.Sequence

Syntax`uc480.MemorySequence.GetLast(out int s32SeqID)`**Description**

Returns the last used sequence memory ID.

Parameter

- `s32SeqID`: Returns the ID of the last sequence memory.

Class[uc480.MemorySequence](#)**Accessible**

Camera.Memory.Sequence

Syntax`uc480.MemorySequence.GetList(out int[] idList)`**Description**

Returns an array with all sequence memory IDs.

Parameter

- `idList`: List containing the sequence memory IDs.

Class[uc480.MemorySequence](#)**Accessible**

Camera.Memory.Sequence

Syntax`uc480.MemorySequence.GetLocked(int s32SeqID, out bool bLocked)`**Description**

Returns if the passed sequence memory is locked.

Parameter

<code>s32SeqID</code>	Sequence memory ID
<code>bLocked</code>	1 = Memory is locked. 0 = Memory is not locked.

Class

[uc480.MemorySequence](#)

Accessible

Camera.Memory.Sequence

Syntax

`uc480.MemorySequence.InitImageQueue()`

Description

Enables the queue mode for existing image memory sequences. New images will be added to the end of the queue on arrival (FIFO principle). The image memory sequence has to be created with [Add\(\)](#) prior to calling `InitImageQueue()`. With [WaitForNextImage\(\)](#) you can query the sequence ID of the first (i.e. oldest) image in the sequence.

Image memory sequences can also be used without queue mode. In this case the current image memory has to be queried with [GetActive\(\)](#) on every frame event. Disadvantage of this proceeding is that at very high frame rates it may happen that additional images arrive between the frame event and accessing/locking the memory. The images arriving in this period will be skipped when you query the current image. When the queue mode is used (`InitImageQueue()`), however, you can be sure to always receive the oldest image which has not yet been queried. In addition, image memories are automatically locked immediately after receiving the image. This prevents images from being overwritten when very high frame rates and few image memories are used.

Parameter

None

Class

[uc480.MemorySequence](#)

Accessible

Camera.Memory.Sequence

Syntax

`uc480.MemorySequence.Lock(int s32SeqID)`

Description

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 [Unlock\(\)](#), you can re-enable write access to the image memory.

Parameter

- `s32SeqID`: ID of the sequence memory to be locked (1...max).

Class

[uc480.MemorySequence](#)

Accessible

Camera.Memory.Sequence

Syntax

```
uc480.MemorySequence.ToMemoryID(int s32SeqID, out int s32MemID)
```

Description

Returns the image memory ID for the passed sequence number.

Parameter

s32SeqID	Sequence memory ID
s32MemID	Image memory ID

Class

[uc480.MemorySequence](#)

Accessible

Camera.Memory.Sequence

Syntax

```
uc480.MemorySequence.Unlock(int s32SeqID)
```

Description

Unlocks a previously locked sequence memory in order to make it available again for storing captured images. The sequence memory is re-inserted at its previous position in the sequence list.

Parameter

- s32SeqID: ID of the sequence memory to unlock.

Class

[uc480.MemorySequence](#)

Accessible

Camera.Memory.Sequence

Syntax

```
uc480.MemorySequence.WaitForNextImage(uint u32Timeout, out int s32MemId, out int s32SeqId)
```

Description

Returns the sequence memory ID of the first (i.e. oldest) image in a memory sequence. The queue mode has to be enabled for the sequence (see [InitImageQueue\(\)](#)). If the sequence does not contain images, `WaitForNextImage()` waits until a new image arrives or until the specified time has elapsed.



Note that also image capture errors are added to the image queue like images. If a call of `WaitForNextImage()` returns `uc480.Defines.Message.CaptureStatus` then you can check by a new call of the method, if any further images were enqueued into the image queue after the error.



Image memories in a sequence with queue mode are automatically locked. The sequence memories will have to be unlocked with [Unlock\(\)](#) in order to be re-used in the sequence.

Parameter

u32Timeout	Timeout in ms. Range 0...2 ³² -1 If no images are in the sequence and no image arrives during the timeout, the returns <code>uc480.Defines.Status.TIMED_OUT</code> .
s32MemId	Returns the image memory ID.
s32SeqId	Returns the sequence memory ID.

3.1.1.23.3 Allocate

Class

[uc480.Memory](#)

Accessible

Camera.Memory

Syntax

```
uc480.Memory.Allocate()
uc480.Memory.Allocate(bool bActive)
uc480.Memory.Allocate(out int s32MemId, bool bSetAsActiveMem)
uc480.Memory.Allocate(uc480.Types.Size<int> size, int s32BitsPerPixel)
uc480.Memory.Allocate(uc480.Types.Size<int> size)
uc480.Memory.Allocate(int s32Width, int s32Height)
uc480.Memory.Allocate(int s32Width, int s32Height, int s32BitsPerPixel)
uc480.Memory.Allocate(int s32Width, int s32Height, int s32BitsPerPixel, out int s32MemId)
uc480.Memory.Allocate(int s32Width, int s32Height, int s32BitsPerPixel, bool bSetActive, out int s32MemId)
uc480.Memory.Allocate(int s32Width, int s32Height, int s32BitsPerPixel, System.IntPtr allocated, out int s32MemId)
```

Description

Allocates an image memory for an image having its dimensions defined by `s32Width` and `s32height` and its color depth defined by `s32BitsPerPixel`. The memory size is at least:

`size = [s32Width * ((s32BitsPerPixel + 1) / 8) + adjust] * s32Height` (adjust see below)

The line increment is calculated as:

`line = s32Width * [(s32BitsPerPixel + 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

RGB16 and RGB15 require the same amount of memory, but can be distinguished by the `s32BitsPerPixel` parameter. For information on the bit depths of different color formats please refer to the Appendix: "Color and memory formats" in the uc480 manual.



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.

Parameter

<none>	The image memory is allocated automatically.
bActive/bSetActiveMem/ bSetActive	1 = Set image memory as active 0 = Do not set image memory as active
size	Height: Image height
	Width: Image width
s32Width	Image width
s32Height	Image height
s32BitsPerPixel	Image bit depth (bits per pixel).
s32MemId	Returns the image memory ID
allocated	Pointer to the starting address of the allocated memory

3.1.1.23.4 CopyImageMem

Class

[uc480.Memory](#)

Accessible

Camera.Memory

Syntax

```
uc480.Memory.CopyImageMem(System.IntPtr ptrSrc, int s32Id, System.IntPtr ptrDst)
```

Description

Copies the contents of the image memory described by `ptrSrc` and `s32Id` to the memory area to whose starting address `ptrDst` points.



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

Parameter

ptrSrc	Pointer to the image memory
s32Id	ID of this image memory
ptrDst	Pointer to the destination memory to copy the image to.

3.1.1.23.5 CopyImageMemLines

Class

[uc480.Memory](#)

Accessible

Camera.Memory

Syntax

```
uc480.Memory.CopyImageMemLines(System.IntPtr ptrSrc, int s32Id, int s32Lines, System.IntPtr ptrDst)
```

Description

Copies the contents of the image memory described by `ptrSrc` and `s32Id` to the memory area to whose starting address `ptrDst` points. The method only copies the number of lines indicated by `s32Lines`.



The allocated memory must be large enough to accommodate the in `s32Lines` given number of image lines considering the image width and format (Bits per Pixel).

Parameter

<code>ptrSrc</code>	Pointer to the image memory
<code>s32Id</code>	ID of this image memory
<code>s32Lines</code>	Number of lines to be copied
<code>ptrDst</code>	Pointer to the destination memory to copy the image to

3.1.1.23.6 CopyToArray

Class

[uc480.Memory](#)

Accessible

Camera.Memory

Syntax

```
uc480.Memory.CopyToArray(int s32MemId, out byte[] u8Image)
uc480.Memory.CopyToArray(int s32MemId, out int[] s32Image)
```

Description

Copies the image of the passed image memory to an array.

Parameter

<code>s32MemId</code>	Image memory ID
<code>s32Image/u8Image</code>	Returns the array

3.1.1.23.7 CopyToBitmap

Class

[uc480.Memory](#)

Accessible

Camera.Memory

Syntax

```
uc480.Memory.CopyToBitmap(int s32MemId, out System.Drawing.Bitmap BitmapFormat)
```

Description

Copies the image of the passed image memory to a bitmap.

Parameter

s32MemId	Image memory ID
BitmapFormat	Returns the bitmap

3.1.1.23.8 Free

Class

[uc480.Memory](#)

Accessible

Camera.Memory

Syntax

```
uc480.Memory.Free(int s32MemId)
uc480.Memory.Free(int[] memIdList)
```

Description

Releases an image memory that was allocated using [Allocate\(\)](#) and removes it from the driver management.



If the memory was not allocated using an SDK function, you need to call `Free()` 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.

Parameter

- s32MemId/memIdList: Image memory ID to be released

3.1.1.23.9 GetActive

Class

[uc480.Memory](#)

Accessible

Camera.Memory

Syntax

```
uc480.Memory.GetActive(out int s32MemId)
uc480.Memory.GetActive(out System.IntPtr ptr)
```

Description

Returns the ID number of the active image memory.

If a Direct3D mode is active and image memory was nevertheless allocated, the ID will be returned. However, in Direct3D mode, the image will not be copied automatically to this image memory.

Parameter

- s32MemId ptr: Returns the image memory ID

3.1.1.23.10 GetBitsPerPixel

Class

[uc480.Memory](#)

Accessible

Camera.Memory

Syntax

```
uc480.Memory.GetBitsPerPixel(int s32MemId, out int s32Bpp)
```

Description

Returns the bits per pixel for the passed image memory.

Parameter

s32MemId	Image memory ID
s32Bpp	Returns the bits per pixel

3.1.1.23.11 GetHeight

Class

[uc480.Memory](#)

Accessible

Camera.Memory

Syntax

```
uc480.Memory.GetHeight(int s32MemId, out int s32Height)
```

Description

Returns the height of the image in the image memory.

Parameter

s32MemId	Image memory ID
s32Height	Returns the image height.

3.1.1.23.12 GetLast

Class

[uc480.Memory](#)

Accessible

Camera.Memory

Syntax

```
uc480.Memory.GetLast(out int s32MemId)  
uc480.Memory.GetLast(out System.IntPtr ptr)
```

Description

Returns the last ID in the image memory.

Parameter

s32MemId	ID of the last image
ptr	Pointer to image memory

3.1.1.23.13 GetList**Class**[uc480.Memory](#)**Accessible**

Camera.Memory

Syntax`uc480.Memory.GetList(out int[] idList)`**Description**

Returns the list of image memory IDs.

Parameter

- `idList`: List of image memory IDs

3.1.1.23.14 GetLocked**Class**[uc480.Memory](#)**Accessible**

Camera.Memory

Syntax`uc480.Memory.GetLocked(int s32MemId, out bool bLocked)`**Description**

Returns if the passed image memory is locked.

Parameter

s32MemId	Image memory ID
bLocked	1 = Image memory is locked 0 = Image memory is not locked

3.1.1.23.15 GetPitch**Class**[uc480.Memory](#)**Accessible**

Camera.Memory

Syntax`uc480.Memory.GetPitch(int s32MemId, out int s32Pitch)`**Description**

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 [Allocate\(\)](#). The line increment is always a number that can be divided by 4.

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
```

Parameter

- `p32Pitch`: Pointer to the variable containing the line increment

3.1.1.23.16 GetSize

Class

[uc480.Memory](#)

Accessible

Camera.Memory

Syntax

```
uc480.Memory.GetSize(int s32MemId, out uc480.Types.Size<int> size)
uc480.Memory.GetSize(int s32MemId, out int s32Width, out int s32Height)
```

Description

Returns the size of the image in the image memory.

Parameter

<code>s32MemId</code>	Image memory ID
<code>size</code>	<code>Width</code> : Returns the image width.
	<code>Height</code> : Returns the image height.
<code>s32Width</code>	Returns the image width.
<code>s32Height</code>	Returns the image height.

3.1.1.23.17 GetWidth

Class

[uc480.Memory](#)

Accessible

Camera.Memory

Syntax

```
uc480.Memory.GetWidth(int s32MemId, out int s32Width)
```

Description

Returns the width of the image in the image memory.

Parameter

s32MemId	Image memory ID
s32Width	Returns the image width.

3.1.1.23.18 Inquire**Class**[uc480.Memory](#)**Accessible**

Camera.Memory

Syntax

```
uc480.Memory.Inquire(int s32MemId, out int s32X, out int s32Y,
                      out int s32BitsPerPixel, out int s32Pitch)
```

Description

Returns the properties of an allocated image memory.

Parameter

s32MemId	Image memory ID of Allocate()
s32X	Returns the width used to define the image memory. You can also pass <code>NULL</code> instead.
s32Y	Returns the height used to define the image memory. You can also pass <code>NULL</code> instead.
s32BitsPerPixel	Returns the bit width used to define the image memory. You can also pass <code>NULL</code> instead.
s32Pitch	Returns the line increment of the image memory. You can also pass <code>NULL</code> instead.

3.1.1.23.19 Lock**Class**[uc480.Memory](#)**Accessible**

Camera.Memory

Syntax

```
uc480.Memory.Lock(int s32MemID)
```

Description

Locks the image memory defined by the ID.

Parameter

- s32MemID: Image memory ID to be locked.

3.1.1.23.20 SetActive

Class

[uc480.Memory](#)

Accessible

Camera.Memory

Syntax

`uc480.Memory.SetActive(int s32MemId)`

Description

Makes the specified image memory the active memory. Only an active image memory can receive image data. When you call [Freeze\(\)](#), the captured image is stored in the image buffer.



In the Direct3D modes, there is no need to set an image memory.

Parameter

- `s32MemId`: Image memory ID

3.1.1.23.21 ToBitmap

Class

[uc480.Memory](#)

Accessible

Camera.Memory

Syntax

`uc480.Memory.ToBitmap(int s32MemId, out System.Drawing.Bitmap bitmap)`

Description

Returns a bitmap which contains the image.

Parameter

s32MemId	Image memory ID
bitmap	<p>Bitmap with the image. The following color modes are supported in the <code>Bitmap</code> class:</p> <ul style="list-style-type: none"> • <code>System.Drawing.Imaging.PixelFormat.Format32bppArgb</code> (<code>uc480.Defines.ColorMode.RGBA8Packed</code>/ <code>uc480.Defines.ColorMode.BGRA8Packed</code>) • <code>System.Drawing.Imaging.PixelFormat.Format24bppRgb</code> (<code>uc480.Defines.ColorMode.RGB8Packed</code>/ <code>uc480.Defines.ColorMode.BGR8Packed</code>) • <code>System.Drawing.Imaging.PixelFormat.Format16bppRgb565</code> (<code>uc480.Defines.ColorMode.BGR565Packed</code>) • <code>System.Drawing.Imaging.PixelFormat.Format16bppRgb555</code> (<code>uc480.Defines.ColorMode.BGR5Packed</code>) • <code>System.Drawing.Imaging.PixelFormat.Format8bppIndexed</code> (<code>uc480.Defines.ColorMode.Mono8</code>/ <code>uc480.Defines.ColorMode.SensorRaw8</code>)

3.1.1.23.22 ToIntPtr

Class

[uc480.Memory](#)

Accessible

Camera.Memory

Syntax

```
uc480.Memory.ToIntPtr(out System.IntPtr ptr)
uc480.Memory.ToIntPtr(int s32MemId, out System.IntPtr ptr)
```

Description

Returns a pointer to the image memory of the active/given image memory ID.

Parameter

s32MemId	Image memory ID
ptr	Pointer to the image memory

3.1.1.23.23 Unlock

Class

[uc480.Memory](#)

Accessible

Camera.Memory

Syntax

```
uc480.Memory.Unlock(int s32MemID)
```

Description

Unlocks the image memory defined by the ID.

Parameter

- `s32MemID`: Image memory ID to be unlocked.

3.1.1.24 Messaging

The `Messaging` class provides methods for enabling Windows messages. If a particular event occurs, the messages are sent to the application.



You have to deactivate Windows messages with `hWnd == NULL` before you free the uc480 API library. Otherwise the application may not close properly.

Methods

Method	Description
Disable	Disables the passed message.
Enable	Enables the passed message.



For the usage of events see also the uc480.NET C# SimpleLive and the uc480MultipleCameraScan demos.

3.1.1.24.1 Disable**Class**

[uc480.Messaging](#)

Accessible

Camera.Messaging

Syntax

```
uc480.Messaging.Disable(uc480.Defines.Message msg)
```

Description

Disables the passed message.

Parameter

- `msg`: Message to be disabled (see [Enable\(\)](#)).

3.1.1.24.2 Enable**Class**

[uc480.Messaging](#)

Accessible

Camera.Messaging

Syntax

```
uc480.Messaging.Enable(uc480.Defines.Message msg, System.IntPtr ptr)
```

Description

Enables the passed message.

Parameter

msg	<p>Message to be enabled:</p> <ul style="list-style-type: none"> • uc480.Defines.Message.AutoBrightnessFinished: The automatic brightness control in the run-once mode is completed. • uc480.Defines.Message.AutoFocusFinished: Automatic focus control is finished (XS only) • uc480.Defines.Message.CameraMemory: In the camera memory mode an image acquisition iteration is finished. • uc480.Defines.Message.CaptureStatus: There is an information about image capturing available. This information can be requested by GetCaptureStatus(). • uc480.Defines.Message.ConnectionSpeedChanged: The connection speed of a USB 3 uc480 camera changed from USB 2.0 to USB 3.0 or from USB 3.0 to USB 2.0. • uc480.Defines.Message.DeviceReconnected: A camera initialized with Init() and disconnected afterwards was reconnected. • uc480.Defines.Message.DeviceRemoved: A camera initialized with Init() was disconnected. • uc480.Defines.Message.FirstPacketReceived: The first data packet of the image was transferred to the PC. This is the earliest time for determining if the image exposure is finished. • uc480.Defines.Message.Frame: A new image is available. • uc480.Defines.Message.NewDevice: A new camera was connected. This is independent of the device handle. • uc480.Defines.Message.Sequence: The sequence is completed. • uc480.Defines.Message.Trigger: 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 uc480.Camera.EventFrame processing event. • uc480.Defines.Message.WhitebalanceFinished: The automatic white balance control is completed.
ptr	Pointer for receiving the message.

3.1.1.25 Parameter

The `Parameter` class provides methods for saving and loading camera parameter sets. Only camera-specific ini files can be loaded.



As the USB uc480 XS has only one image size you cannot save a separate resolution for trigger mode.



When loading an INI file, make sure that the image size (AOI) and color depth parameters in the INI file match those in the allocated memory. Otherwise, display errors may occur.

Methods

Method	Description
Clear	Deletes the camera parameter set in the EEPROM.
GetNumSupported	Returns the number of supported parameter sets in the camera EEPROM.
GetSupported	Returns if a camera parameter set in the EEPROM is supported.
Load	Loads a camera parameter set from the EEPROM or from a file.
ResetToDefault	Resets all parameters to the camera-specific defaults as specified by the driver.
Save	Saves a camera parameter set into the EEPROM or into a file.

3.1.1.25.1 Clear

Class

[uc480.Parameter](#)

Accessible

Camera.Parameter

Syntax

`uc480.Parameter.Clear()`

Description

Deletes the camera parameter set in the EEPROM.

Parameter

none

3.1.1.25.2 GetNumSupported

Class

[uc480.Parameter](#)

Accessible

Camera.Parameter

Syntax

```
uc480.Parameter.GetNumSupported(out uint num)
```

Description

Returns the number of supported parameter sets in the camera EEPROM. At the moment this is "1" for all cameras.

Parameter

- `num`: Returns the number of supported parameter sets.

3.1.1.25.3 GetSupported

Class

[uc480.Parameter](#)

Accessible

Camera.Parameter

Syntax

```
uc480.Parameter.GetSupported(out bool supported)
```

Description

Returns if a camera parameter set in the EEPROM is supported.

Parameter

<code>supported</code>	1 = Zoom function is supported
	0 = Zoom function is not supported

3.1.1.25.4 Load

Class

[uc480.Parameter](#)

Accessible

Camera.Parameter

Syntax

```
uc480.Parameter.Load()  
uc480.Parameter.Load(string strFilename)
```

Description

Loads a camera parameter set from the EEPROM or a file.

Parameter

<code><none></code>	Parameter set is loaded from the EEPROM
<code>strFilename</code>	Parameter set is loaded from the file specified in <code>strFilename</code> .

3.1.1.25.5 ResetToDefault

Class

[uc480.Parameter](#)

Accessible

Camera.Parameter

Syntax

`uc480.Parameter.ResetToDefault()`

Description

Resets all parameters to the camera-specific defaults as specified by the driver. By default, the camera uses full resolution, a medium speed and color level gain values adapted to daylight exposure.

Parameter

None

3.1.1.25.6 Save

Class

[uc480.Parameter](#)

Accessible

Camera.Parameter

Syntax

`uc480.Parameter.Save()`
`uc480.Parameter.Save(string strFilename)`

Description

Saves a camera parameter set into the EEPROM or a file.

Parameter

<none>	Parameter set is saved into the EEPROM
strFilename	Parameter set is saved into the file specified in <code>strFilename</code> .

3.1.1.26 PixelFormat

The `PixelFormat` class provides methods for setting the color 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 color mode. When images are transferred directly to the graphics card memory, make sure that the display settings match the color mode settings. Otherwise, the images will be displayed with altered colors or are not clearly visible.



Note on display modes

This class is only supported in the bitmap (DIB) display mode.

Use [Render\(\)](#) to display other color formats in Direct3D mode.



Note on USB uc480 XS

The USB uc480 XS performs color conversion internally and currently supports the following color formats:

- uc480.Defines.ColorMode.BGR8Packed (software conversion from YUV camera data)
- uc480.Defines.ColorMode.BGRA8Packed (software conversion from YUV camera data)
- uc480.Defines.ColorMode.CBYCRYPacked

Note on bit depth

 Color formats with a bit depth of more than 8 bits per channel are only supported by GigE uc480 and USB 3 uc480 camera models and USB cameras (UI-214x and UI-228x models). Using color formats with higher bit depth increases the bandwidth used by a camera.

Note on RGB15/16

 For the RGB16 and RGB15 data formats, the MSBs of the internal 8-bit R, G and B colors are used.

Methods

Method	Description
Get	Returns the current setting.
GetBitsPerPixel	Returns the bits per pixel for the current color mode.
GetBytesPerPixel	Returns the bytes per pixel for the current color mode.
Set	Sets the color mode.

3.1.1.26.1 Get**Class**[uc480.PixelFormat](#)**Accessible**

Camera.PixelFormat

Syntax

uc480.PixelFormat.Get(out uc480.Defines.ColorMode mode)

Description

Returns the current setting.

Parameter

- mode: Returns the current setting (see [set\(\)](#)).

3.1.1.26.2 GetBitsPerPixel**Class**[uc480.PixelFormat](#)**Accessible**

Camera.PixelFormat

Syntax

uc480.PixelFormat.GetBitsPerPixel()

```
uc480.PixelFormat.GetBitsPerPixel(out int bpp)
```

Description

Returns the bits per pixel for the current color mode.

Parameter

- `bpp`: Returns the bits per pixel.

3.1.1.26.3 GetBytesPerPixel

Class

[uc480.PixelFormat](#)

Accessible

Camera.PixelFormat

Syntax

```
uc480.PixelFormat.GetBytesPerPixel()
uc480.PixelFormat.GetBytesPerPixel(out int bpp)
```

Description

Returns the bytes per pixel for the current color mode.

Parameter

- `bpp`: Returns the bytes per pixel.

3.1.1.26.4 Set

Class

[uc480.PixelFormat](#)

Accessible

Camera.PixelFormat

Syntax

```
uc480.PixelFormat.Set(uc480.Defines.ColorMode mode)
```

Description

Sets the color mode.

Parameter

- `mode`: Color mode to be set.

<code>uc480.Defines.ColorMode.Mono16</code>	Grayscale (16), for monochrome and color cameras, LUT/gamma active
<code>uc480.Defines.ColorMode.Mono12</code>	Grayscale (12), for monochrome and color cameras, LUT/gamma active
<code>uc480.Defines.ColorMode.Mono8</code>	Grayscale (8), for monochrome and color cameras, LUT/gamma active
<code>uc480.Defines.ColorMode.SensorRaw16</code>	Raw sensor data (16), for monochrome and color cameras, LUT/gamma active. For color cameras same as <code>uc480.Defines.ColorMode.BAYER_RG16</code> .

uc480.Defines.ColorMode.SensorRaw12	Raw sensor data (12), for monochrome and color cameras, LUT/gamma active. For color cameras same as uc480.Defines.ColorMode.BAYER_RG12.
uc480.Defines.ColorMode.SensorRaw8	Raw sensor data (8), for monochrome and color cameras, LUT/gamma active. For color cameras same as uc480.Defines.ColorMode.BAYER_RG8.
uc480.Defines.ColorMode.RGBA12Unpacked	Unpacked RGB (12 12 12), for monochrome and color cameras, LUT/gamma active
uc480.Defines.ColorMode.RGB12Unpacked	Unpacked RGB (12 12 12), for monochrome and color cameras, LUT/gamma active
uc480.Defines.ColorMode.RGB10Unpacked	Unpacked RGB (10 10 10), for monochrome and color cameras, LUT/gamma active
uc480.Defines.ColorMode.RGB10Packed	RGB (10 10 10), for monochrome and color cameras, LUT/gamma active
uc480.Defines.ColorMode.RGBA8Packed	RGB (8 8 8), for monochrome and color cameras, LUT/gamma active
uc480.Defines.ColorMode.RGBY8Packed	RGBY (8 8 8), for monochrome and color cameras, LUT/gamma active
uc480.Defines.ColorMode.RGB8Packed	RGB (8 8 8), for monochrome and color cameras, LUT/gamma active
uc480.Defines.RGB8Planar	Planar RGB (8) for monochrome and color cameras, LUT/gamma active
uc480.Defines.ColorMode.BGRA12Unpacked	Unpacked BGRA (12 12 12), for monochrome and color cameras, LUT/gamma active
uc480.Defines.ColorMode.BGR12Unpacked	Unpacked BGR (12 12 12) for monochrome and color cameras, LUT/gamma active
uc480.Defines.ColorMode.BGR10Unpacked	Unpacked BGR (10 10 10) for monochrome and color cameras, LUT/gamma active
uc480.Defines.ColorMode.BGR10Packed	BGR (10 10 10), for monochrome and color cameras, LUT/gamma active
uc480.Defines.ColorMode.BGRA8Packed	BGR (8 8 8), for monochrome and color cameras, LUT/gamma active
uc480.Defines.ColorMode.BGR8Packed	BGR (8 8 8), for monochrome and color cameras, LUT/gamma active
uc480.Defines.ColorMode.BGRY8Packed	BGRY (8 8 8), for monochrome and color cameras, LUT/gamma active
uc480.Defines.ColorMode.BGR565Packed	BGR (5 6 5), for monochrome and color cameras, LUT/gamma active
uc480.Defines.ColorMode.BGR5Packed	BGR (5 5 5), for monochrome and color cameras, LUT/gamma active
uc480.Defines.ColorMode.UYVYPacked	YUV 4:2:2 (8 8), for monochrome and color cameras, LUT/gamma active
uc480.Defines.ColorMode.CBYCRYPacked	YC _b C _r 4:2:2 (8 8), for monochrome and color

d	cameras, LUT/gamma active
uc480.Defines.ColorMode.Jpeg	JPEG for XS cameras
uc480.Defines.ColorMode.PreferPackedSourceFormat	<p>USB 3 uc480 CP only: If IS_CM_PREFER_PACKED_SOURCE_FORMAT is ORed to the selected pixel format, packed pixel formats are preferred for transfer. The command can be combined with the following pixel formats (in combination with other formats it is ignored):</p> <ul style="list-style-type: none"> • IS_CM_SENSOR_RAW10 • IS_CM_MONO10 • IS_CM_BGR10_UNPACKED • IS_RGB10_UNPACKED <p>Because a packed format is used for transfer, less bandwidth is required. But the CPU load increases, because the image has to be unpacked in the PC again.</p>

3.1.1.27 RopEffect

The `RopEffect` class provides methods for controlling the real-time image geometry modification (Rop = raster operation).

Methods

Method	Description
Get	Returns the current settings.
Set	Sets the values for the image mirroring.

3.1.1.27.1 Get

Class

[uc480.RopEffect](#)

Accessible

Camera.RopEffect

Syntax

```
uc480.RopEffect.Get(out uc480.Defines.RopEffectMode mode)
```

Description

Returns the current settings.

Parameter

mode	<p>Returns the mode of the Rop effect:</p> <ul style="list-style-type: none"> • <code>uc480.Defines.RopEffectMode.UpDown</code>: Mirrors the image along the horizontal axis. • <code>uc480.Defines.RopEffectMode.LeftRight</code>: Mirrors the image along the vertical axis. • <code>uc480.Defines.RopEffectMode.MirrorNone</code>: No image mirroring.
------	--

3.1.1.27.2 Set

Class

[uc480.RopEffect](#)

Accessible

Camera.RopEffect

Syntax

```
uc480.RopEffect.Set(uc480.Defines.RopEffectMode mode, bool bEnable)
```

Description

Sets the values for the image mirroring.

Parameter

mode	Mode of the Rop effect: <ul style="list-style-type: none"> • uc480.Defines.RopEffectMode.UpDown: Mirrors the image along the horizontal axis. • uc480.Defines.RopEffectMode.LeftRight: Mirrors the image along the vertical axis. • uc480.Defines.RopEffectMode.MirrorNone: No image mirroring. Depending on the sensor, this operation is performed in the camera or in the PC software.
bEnable	Enables/disables Rop effect: 1 = Enable 0 = Disable

3.1.1.28 Saturation

The `Saturation` provides methods for controlling the software color saturation.



In the YUV format, color information (i.e. the color 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 color formats than YUV, U and V are converted using a driver matrix.

Methods

Method	Description
Get	Returns the current setting for color saturation.
GetDefault	Returns the default value for color saturation.
GetRange	Returns the range for color saturation.
GetSupported	Returns the supported modes for color saturation.
Set	Sets the values for color saturation.

3.1.1.28.1 Get

Class

[uc480.Saturation](#)

Accessible

Camera.Saturation

Syntax

```
uc480.Saturation.Get(out int s32Value)
uc480.Saturation.Get(out int s32ValueU, out int s32ValueV)
```

Description

Returns the current setting for color saturation.

Parameter

s32Value	Returns the current color saturation setting.
s32ValueU	Returns the current value for the U saturation.
s32ValueV	Returns the current value for the V saturation.

3.1.1.28.2 GetDefault

Class

[uc480.Saturation](#)

Accessible

Camera.Saturation

Syntax

```
uc480.Saturation.GetDefault(out int s32Default)
uc480.Saturation.GetDefault(out int s32DefValueU, out int s32DefValueV)
```

Description

Returns the default value for color saturation.

Parameter

s32Default	Returns the default color saturation setting.
s32DefValueU	Returns the default value for the U saturation.
s32DefValueV	Returns the default value for the V saturation.

3.1.1.28.3 GetRange

Class

[uc480.Saturation](#)

Accessible

Camera.Saturation

Syntax

```
uc480.Saturation.GetRange(out uc480.Types.Range<int> range)
uc480.Saturation.GetRange(out uc480.Types.Range<int> rangeU, out uc480.Types.Range<int> rangeV)
uc480.Saturation.GetRange(out int s32Min, out int s32Max, out int s32Inc)
```

Description

Returns the minimum and maximum value and the increment for color saturation.

Parameter

range	Minimum: Returns the minimum value.
rangeU	Maximum: Returns the maximum value.
rangeV	Increment: Returns the increment.
s32Min	Returns the minimum value.
s32Max	Returns the maximum value.
s32Inc	Returns the increment.

3.1.1.28.4 GetSupported

Class

[uc480.Saturation](#)

Accessible

Camera.Saturation

Syntax

```
uc480.Saturation.GetSupported(out bool bSupported)
```

Description

Returns the supported modes for color saturation.

Parameter

bSupported	1 = Color saturation is supported. 0 = Color saturation is not supported.
------------	--

3.1.1.28.5 Set

Class

[uc480.Saturation](#)

Accessible

Camera.Saturation

Syntax

```
uc480.Saturation.Set(int s32Value)
uc480.Saturation.Set(int s32ValueU, int s32ValueV)
```

Description

Sets the values for color saturation.

Parameter

s32Value	Saturation: value multiplied by 100.
s32ValueU	U saturation: value multiplied by 100.
s32ValueV	V saturation: value multiplied by 100.

3.1.1.29 ScenePreset

The `ScenePreset` class provides methods for controlling the camera parameters of some uc480 cameras to suit specific image acquisition situations. Depending on the selected mode, the feature changes the settings for several of the following parameters:

- Exposure time
- Gain
- Sharpness
- Color temperature
- Fields of view for measuring auto focus and exposure time



This class is currently only supported by the UI-1008XS models.

Additional methods that have an influence on the above parameters should be called after `ScenePreset`. Otherwise some of these settings might be overwritten by calling `ScenePreset`.

Methods

Method	Description
<u>Get</u>	Returns the set scene mode.
<u>GetDefault</u>	Returns the default setting for the scene mode.
<u>GetSupported</u>	Returns the scene modes supported by the camera.
<u>IsSupported</u>	Returns if the passed mode is supported.
<u>Set</u>	Enables a scene mode.

3.1.1.29.1 Get

Class

[uc480.ScenePreset](#)

Accessible

`Camera.ScenePreset`

Syntax

```
uc480.ScenePreset.Get(out uc480.Defines.SceneMode mode)
```



This class is currently only supported by the UI-1008XS models.

Description

Returns the set scene mode.

Parameter

- mode: Returns the mode.

3.1.1.29.2 GetDefault**Class**[uc480.ScenePreset](#)**Accessible**

Camera.ScenePreset

Syntax

```
uc480.ScenePreset.GetDefault(out uc480.Defines.SceneMode mode)
```



This class is currently only supported by the UI-1008XS models.

Description

Returns the default setting for the scene mode.

Parameter

- mode: Returns the default mode.

3.1.1.29.3 GetSupported**Class**[uc480.ScenePreset](#)**Accessible**

Camera.ScenePreset

Syntax

```
uc480.ScenePreset.GetSupported(out uc480.Defines.SceneMode mode)
```



This class is currently only supported by the UI-1008XS models.

Description

Returns the scene modes supported by the camera.

Parameter

- mode: Returns the supported modes.

<code>uc480.Defines.SceneMode.Invalid</code>	The camera does not support scene modes
<code>uc480.Defines.SceneMode.Automatic</code>	Mode for automatic adjustment
<code>uc480.Defines.SceneMode.Portion</code>	Mode for capturing portraits
<code>uc480.Defines.SceneMode.Sunny</code>	Mode for image capture in direct sunlight
<code>uc480.Defines.SceneMode.Entertainment</code>	Mode for image capture in artificial light environments
<code>uc480.Defines.SceneMode.Night</code>	Mode for image capture with little light
<code>uc480.Defines.SceneMode.Sports</code>	Mode for capturing fast moving objects
<code>uc480.Defines.SceneMode.Landscape</code>	Mode for capturing landscapes

3.1.1.29.4 IsSupported

Class

[uc480.ScenePreset](#)

Accessible

Camera.ScenePreset

Syntax

`uc480.ScenePreset.IsSupported(uc480.Defines.SceneMode mode)`

Description

Returns if the passed mode is supported.

Parameter

- `mode`: Scene mode to be queried to be supported (see [GetSupported\(\)](#)).

3.1.1.29.5 Set

Class

[uc480.ScenePreset](#)

Accessible

Camera.ScenePreset

Syntax

`uc480.ScenePreset.Set(uc480.Defines.SceneMode mode)`



This class is currently only supported by the UI-1008XS models.

Description

Enables a scene mode.

Parameter

- `mode`: Mode to be set (see [GetSupported\(\)](#)).

3.1.1.30 Sharpness

The `Sharpness` class provides methods for enhancing or reducing the image sharpness of some uc480 cameras.



This class is currently only supported by the USB uc480 XS camera family.

Methods

Method	Description
<u>Get</u>	Returns the current image sharpness setting.
<u>GetDefault</u>	Returns the default value for image sharpness.
<u>GetRange</u>	Returns the range for image sharpness.
<u>GetSupported</u>	Returns if the camera support the sharpness function.
<u>Set</u>	Sets a value for image sharpness.

3.1.1.30.1 Get

Class

[uc480.Sharpness](#)

Accessible

Camera.Sharpness

Syntax

```
uc480.Sharpness.Get(out uint u32Value)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Returns the current image sharpness setting.

Parameter

- `u32Value`: Returns the current value.

3.1.1.30.2 GetDefault

Class

[uc480.Sharpness](#)

Accessible

Camera.Sharpness

Syntax

```
uc480.Sharpness.GetDefault(out uint u32DefValue)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Returns the default value for image sharpness.

Parameter

- `u32DefValue`: Returns the default value.

3.1.1.30.3 GetRange

Class

[uc480.Sharpness](#)

Accessible

Camera.Sharpness

Syntax

```
uc480.Sharpness.GetRange(out uc480.Types.Range<uint> range)
uc480.Sharpness.GetRange(out uint u32Min, out uint u32Max, out uint u32Inc)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Returns the minimum and maximum value and the increment for image sharpness.

Parameter

<code>range</code>	<code>Minimum</code> : Returns the minimum value. <code>Maximum</code> : Returns the maximum value. <code>Increment</code> : Returns the increment.
<code>u32Min</code>	Returns the minimum value.
<code>u32Max</code>	Returns the maximum value.
<code>u32Inc</code>	Returns the increment.

3.1.1.30.4 GetSupported

Class

[uc480.Sharpness](#)

Accessible

Camera.Sharpness

Syntax

```
uc480.Sharpness.GetSupported(out bool bSupported)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Returns if the camera support the sharpness function.

Parameter

bSupported	1 = The camera supports setting the image sharpness 0 = The camera does not support setting the image sharpness
------------	--

3.1.1.30.5 Set**Class**

[uc480.Sharpness](#)

Accessible

Camera.Sharpness

Syntax

```
uc480.Sharpness.Set(uint u32Value)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Sets a value for image sharpness.

Parameter

- `u32Value`: Value to be set.

3.1.1.31 Size

The `Size` class provides classes for changing the image size or format. The following classes exist:

- [AOI](#)
- [Binning](#)
- [ImageFormat](#)
- [SensorScaler](#)
- [Subsampling](#)

3.1.1.31.1 AOI

The `AOI` class provides methods for setting 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 for automatic white balance control

AOI for automatic image control

The AOI for automatic brightness control (AES/AGC) and automatic white balance (AWB) defaults to the same size as the current image (i.e. the image AOI).

After changes to the image geometry (by resetting an image AOI, by binning or

subsampling), the AOIs for automatic image control will always be reset to the image AOI value. This means that it might be necessary to set the AOIs for auto brightness/auto white balance again manually.

Fast changes of AOI position

Using [SetPosFast\(\)](#), you can change the positions of AOIs very quickly. Executing this command takes just a few milliseconds. When using this command, a few special requirements have to be met:

- The command is currently not supported by all uc480 cameras. With [GetPosFastSupported\(\)](#), you can check whether your sensor supports fast position changes.
- Hot pixel correction has to be disabled (see [HotPixel](#)).
- Image capture is not suspended for fast AOI position changes. As a result, a number of images might still be transferred with the old AOI position if they were in the driver buffer at that moment.

Note on changing the image size

- When changing the size of the AOI, please make sure that the selected image memory is large enough. If it isn't, allocate a new image memory (see [Allocate\(\)](#)).
- Changes to the image size affect the value ranges of the frame rate and exposure time. After executing `AOI`, calling the following classes is recommended in order to keep the defined camera settings:
 - [Framerate](#)
 - [Exposure](#)
 - If you are using the flash function: [IO](#)



Note on step widths for AOI definition (position grid)

The available step widths for the position and size of image AOIs depend on the sensor. The values defining the position and size of an AOI have to be integer multiples of the allowed step widths.

For details on the AOI grids of the individual camera models, please see "Camera and sensor data" in the uc480 manual.



Note to AOI in combination with high frame rates

With very small AOI and therefore high frame rate and maximum possible frame rate set, it is possible that the USB camera transfers in freerun mode only half frame rates. This is a signal for a camera-internal overload. In this case it is recommended to set the frame rate to maximum of 98 %.



The following classes and methods exists:

- [Multi](#)
- [Sequences](#)

Methods

Method	Description
Get	Returns the AOI.
GetAbsX	Returns the X position.
GetAbsY	Returns the Y position.
GetAutoBrightness	Returns the AOI for automatic brightness control
GetOriginal	Returns the AOI without binning, sub-sampling or scaling.
GetPosFastSupported	Returns if fast AOI position changes are supported.
GetPosRange	Returns the range of possible positions.
GetSizeRange	Returns the range possible sizes.
GetWhiteBalance	Returns the AOI for automatic white balance
Set	Sets the AOI.
SetAutoBrightness	Sets the AOI for automatic brightness control
SetPosFast	Allows changing the AOI position very quickly.
SetWhiteBalance	Sets the AOI for automatic white balance



Note that only the camera models UI-124x/UI-324x/UI-524x have the multi AOI mode.

The `Multi` class provides methods for setting multiple AOIs in one image capture. The AOIs are transferred together as one image. In this mode you can create 2 or 4 AOIs, which have either the same X axis or the same Y axis. The sensor is faster in this mode.

It is possible to switch the AOI in the horizontal direction.

Methods

Method	Description
Disable	Disables Multi AOI.
GetAxes	Returns the set multi AOI mode.
GetSupported	Returns the supported multi AOI modes.
SetAxes	Sets the multi AOI mode.

Class

[uc480.SizeAOIMulti](#)

Accessible

Camera.Size.AOI.Multi

Syntax

`uc480.SizeAOIMulti.Disable()`

Description

Disables multi AOI.

Parameter

None

Class

[uc480.SizeAOIMulti](#)

Accessible

Camera.Size.AOI.Multi

Syntax

```
uc480.SizeAOIMulti.GetAxis(out uint[] u32Axes)
```

Description

Returns the set multi AOI mode.

Parameter

- `u32Axes`: Returns an array with the axes of the AOI

Class

[uc480.SizeAOIMulti](#)

Accessible

Camera.Size.AOI.Multi

Syntax

```
uc480.SizeAOIMulti.GetSupported(out uc480.Defines.AOIMultiMode mode)
```

Description

Returns the supported multi AOI modes.

Parameter

- `mode`: Returns the supported multi AOI modes.

Class

[uc480.SizeAOIMulti](#)

Accessible

Camera.Size.AOI.Multi

Syntax

```
uc480.SizeAOIMulti.SetAxis(uint[] u32Axes)
```

Description

Sets the multi AOI mode.

Parameter

- `u32Axes`: Array with the axes of the mutli AOI.



Note that only the camera models UI-124x/UI-324x/UI-524x and UI-125x/UI-325x/UI-525x have the sequence AOI mode.

The `Sequences` class provides methods for using the sequence AOI mode. In this mode you can define besides the normal AOI (AOI 1) up to 3 further AOI on the sensor. When activating the sequence mode, note that only the following combinations are possible:

1. All additional AOIs are off. AOI 1 is always active.
2. AOI 2 (+ AOI 1)
3. AOI 2 and 3 (+ AOI 1)
4. AOI 2, 3 and 4 (+ AOI 1)

It is not possible to have a combination e.g. of AOI 2 and AOI 4.

In the version 4.40 binning, subsampling and scaler are not supported.

Methods

Method	Description
GetEnable	Returns the bitmask.
GetParams	Returns the parameters of AOI 2, 3 or 4.
GetSupported	Returns a bitmask with the supported AOIs.
SetEnable	Set a bitmask defining which AOIs should be active.
SetParams	Sets the parameters of AOI 2, 3 or 4.

Class

[uc480.SizeAOISequences](#)

Accessible

Camera.Size.AOI.Sequences

Syntax

```
uc480.SizeAOISequences.GetEnable(out uc480.Defines.AOISequencemode mode)
```

Description

Returns the bitmask (only UI-124x/UI-324x/UI-524x and UI-125x/UI-325x/UI-525x camera models).

Parameter

- `mode`: Returns the bitmask.

Class

[uc480.SizeAOISequences](#)

Accessible

Camera.Size.AOI.Sequences

Syntax

```
uc480.SizeAOISequences.GetParams(out uc480.Types.AoiSequenceparameter parameter)
```

Description

Returns the parameters of AOI 2, 3 or 4 (only UI-124x/UI-324x/UI-524x and UI-125x/UI-325x/UI-525x camera models).

Parameter

- parameter: Returns [uc480.Types.AoiSequenceParameter](#)

Class

[uc480.SizeAOISquence](#)

Accessible

Camera.Size.AOI.Sequences

Syntax

```
uc480.SizeAOISquence.GetSupported(out uc480.Defines.AOISequenceMode mode)
uc480.SizeAOISquence.GetSupported(out int s32Mask)
```

Description

Returns a bitmask with the supported AOIs (only UI-124x/UI-324x/UI-524x and UI-125x/UI-325x/UI-525x camera models).

Parameter

- mode/s32Mask: Returns the supported AOIs.

Class

[uc480.SizeAOISquence](#)

Accessible

Camera.Size.AOI.Sequences

Syntax

```
uc480.SizeAOISquence.SetEnable(uc480.Defines.AOISequenceMode mode)
```

Description

Set a bitmask defining which AOIs should be active (only UI-124x/UI-324x/UI-524x and UI-125x/UI-325x/UI-525x camera models).



Note: [SetParams\(\)](#) must be called after `SetEnable()`, with enabling the sequence AOI mode all AOIs are set to the same value and therefore the parameters are lost.

Parameter

- mode: Bitmask with the defined AOIs

Class

[uc480.SizeAOISquence](#)

Accessible

Camera.Size.AOI.Sequences

Syntax

```
uc480.SizeAOISquence.SetParams(uc480.Types.AoiSequenceParameter parameter)
```

Description

Sets the parameters of AOI 2, 3 or 4 (only UI-124x/UI-324x/UI-524x and UI-125x/UI-325x/UI-525x camera models).

Parameter

- parameter: Sets the sequence AOI parameters (see [uc480.Types.AoiSequenceParameter](#))

Class

[uc480.SizeAOI](#)

Accessible

Camera.Size.AOI

Syntax

```
uc480.SizeAOI.Get(out int s32PosX, out int s32PosY, out int s32Width, out int s32Height)  
uc480.SizeAOI.Get(out System.Drawing.Rectangle rect)
```

Description

Returns the AOI.

Parameter

s32PosX	Returns the X position.
s32PosY	Returns the Y position.
s32Width	Returns the width.
s32Height	Returns the height.

Class

[uc480.SizeAOI](#)

Accessible

Camera.Size.AOI

Syntax

```
uc480.SizeAOI.GetAbsX(out bool bAbsPosX)
```

Description

Returns if the X position is set absolutely.

Parameter

bAbsPosX	1 = X position is set absolutely
	0 = X position is not set absolutely

Class

[uc480.SizeAOI](#)

Accessible

Camera.Size.AOI

Syntax

```
uc480.SizeAOI.GetAbsY(out bool bAbsPosY)
```

Description

Returns if the Y position is set absolutely.

Parameter

bAbsPosY	1 = Y position is set absolutely
	0 = Y position is not set absolutely

Class

[uc480.SizeAOI](#)

Accessible

Camera.Size.AOI

Syntax

```
uc480.SizeAOI.GetAutoBrightness(out int s32PosX, out int s32PosY, out int s32Width, out int s32Height)
uc480.SizeAOI.GetAutoBrightness(out System.Drawing.Rectangle rect)
```

Description

Returns the AOI for automatic brightness control.

Parameter

s32PosX	Returns the X position.
s32PosY	Returns the Y position.
s32Width	Returns the width.
s32Height	Returns the height.

Class

[uc480.SizeAOI](#)

Accessible

Camera.Size.AOI

Syntax

```
uc480.SizeAOI.GetOriginal(out int s32PosX, out int s32PosY, out int s32Width, out int s32Height)
uc480.SizeAOI.GetOriginal(out System.Drawing.Rectangle rect)
```

Description

Returns the AOI without binning, subsampling or scaling.

Parameter

s32PosX	Returns the X position.
s32PosY	Returns the Y position.
s32Width	Returns the width.
s32Height	Returns the height.

Class

[uc480.SizeAOI](#)

Accessible

Camera.Size.AOI

Syntax

`uc480.SizeAOI.GetPosFastSupported(out bool bSupported)`

Description

Returns if fast AOI position changes are supported. The variable returns 0 if the function is not supported by the sensor.

Parameter

bSupported	1 = Fast AOI position changes are supported
	0 = Fast AOI position changes are not supported

Class

[uc480.SizeAOI](#)

Accessible

Camera.Size.AOI

Syntax

`uc480.SizeAOI.GetPosRange(out uc480.Types.Range<int> rangePosX, out uc480.Types.Range<int> rangePosY)`
`uc480.SizeAOI.GetPosRange(out int s32MinPosX, out int s32MinPosY, out int s32MaxPosX, out int s32MaxPosY, ou`

Description

Returns the range of possible position.

Parameter

rangePosX	Minimum: Returns the minimum X position. Maximum: Returns the maximum X position. Increment: Returns the increment for X.
rangePosY	Minimum: Returns the minimum Y position. Maximum: Returns the maximum Y position. Increment: Returns the increment for Y.
s32MinPosX	Returns the minimum X position.
s32MinPosY	Returns the minimum Y position.
s32MaxPosX	Returns the maximum X position.
s32MaxPosY	Returns the maximum Y position.
s32IncPosX	Returns the increment for X.
s32IncPosY	Returns the increment for Y.

Class

[uc480.SizeAOI](#)

Accessible

Camera.Size.AOI

Syntax

```
uc480.SizeAOI.GetSizeRange(out uc480.Types.Range<int> rangeWidth, out uc480.Types.Range<int> rangeHeight)
uc480.SizeAOI.GetSizeRange(out int s32MinWidth, out int s32MinHeight, out int s32MaxWidth, out int s32MaxHeight)
```

Description

Returns the range for the size.

Parameter

rangeWidth	Minimum: Returns the minimum width. Maximum: Returns the maximum width. Increment: Returns the increment.
rangeHeight	Minimum: Returns the minimum height. Maximum: Returns the maximum height. Increment: Returns the increment.
s32MinWidth	Minimum width
s32MinHeight	Minimum height
s32MaxWidth	Maximum width
s32MaxHeight	Maximum height
s32IncWidth	Width increment
s32IncHeight	Height increment

Class

[uc480.SizeAOI](#)

Accessible

Camera.Size.AOI

Syntax

```
uc480.SizeAOI.GetWhiteBalance(out int s32PosX, out int s32PosY, out int s32Width, out int s32Height)
uc480.SizeAOI.GetWhiteBalance(out System.Drawing.Rectangle rect)
```

Description

Returns the AOI for automatic white balance.

Parameter

s32PosX	Returns the X position.
s32PosY	Returns the Y position.
s32Width	Returns the width.
s32Height	Returns the height.

Class

[uc480.SizeAOI](#)

Accessible

Camera.Size.AOI

Syntax

```
uc480.SizeAOI.Set(int s32PosX, int s32PosY, int s32Width, int s32Height)  
uc480.SizeAOI.Set(System.Drawing.Rectangle Rectangle)
```

Description

Sets the AOI.

Parameter

s32PosX	X position of the AOI
s32PosY	Y position of the AOI
s32Width	Width of the AOI
s32Height	Heighth of the AOI

Class

[uc480.SizeAOI](#)

Accessible

Camera.Size.AOI

Syntax

```
uc480.SizeAOI.SetAutoBrightness(int s32PosX, int s32PosY, int s32Width, int s32Height)  
uc480.SizeAOI.SetAutoBrightness(System.Drawing.Rectangle Rectangle)
```

Description

Sets the AOI for automatic brightness control.

Parameter

s32PosX	X position of the AOI
s32PosY	Y position of the AOI
s32Width	Width of the AOI
s32Height	Heighth of the AOI

Class

[uc480.SizeAOI](#)

Accessible

Camera.Size.AOI

Syntax

```
uc480.SizeAOI.SetPosFast(int s32PosX, int s32PosY)  
uc480.SizeAOI.SetPosFast(System.Drawing.Point Point)
```

Description

Allows changing the AOI position very quickly. [Hot pixel](#) correction has to be disabled.

Parameter

s32PosX	Sets X position
s32PosY	Sets Y position

Class

[uc480.SizeAOI](#)

Accessible

Camera.Size.AOI

Syntax

```
uc480.SizeAOI.SetWhiteBalance(int s32PosX, int s32PosY, int s32Width, int s32Height)
uc480.SizeAOI.SetWhiteBalance(System.Drawing.Rectangle Rectangle)
```

Description

Sets the AOI for automatic white balance.

Parameter

s32PosX	X position of the AOI
s32PosY	Y position of the AOI
s32Width	Width of the AOI
s32Height	Heigth of the AOI

3.1.1.31.2 Binning

The `Binning` class provides methods for using the binning mode. 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. In order to simultaneously enable horizontal and vertical binning, the horizontal and vertical binning parameters can by linked by a logical OR. Depending on the sensor used, the sensitivity or the frame rate can be increased while binning is enabled.

The adjustable binning factors of each sensor are listed in the "Camera and sensor data" chapter in the uc480 manual.



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 [Set\(\)](#), calling the following methods is recommended in order to keep the defined camera settings:

- Set the frame rate via [Set\(\)](#)
- [Exposure](#)
- If you are using the uc480's flash function: [IO](#)



Note on camera models UI-124x/UI-324x/UI-524x and UI-155x/UI-555x

For the models UI-124x/UI-524x and UI-155x/UI-555x, you can use binning only combined for the horizontal and the vertical direction. Please see also the information in chapters "UI-124x/UI-324x/UI-524x application notes" and "UI-155x/UI-555x application notes" in the uc480 manual.

Methods

Method	Description
Get	Returns the current setting.
GetFactorHorizontal	Returns the horizontal binning factor.
GetFactorVertical	Returns the vertical binning factor.
GetSupported	Returns the supported binning modes.
GetType	Indicates whether the camera uses color-proof binning or not.
Set	Enables/disables binning.

Class

[uc480.SizeBinning](#)

Accessible

Camera.Size.Binning

Syntax

uc480.SizeBinning.Get(out uc480.Defines.BinningMode mode)

Description

Returns the current setting.

Parameter

- mode: Returns the current setting.

Class

[uc480.SizeBinning](#)

Accessible

Camera.Size.Binning

Syntax

uc480.SizeBinning.GetFactorHorizontal(out int s32Factor)

Description

Returns the horizontal binning factor.

Parameter

- s32Factor: Returns the horizontal factor.

Class

[uc480.SizeBinning](#)

Accessible

Camera.Size.Binning

Syntax

uc480.SizeBinning.GetFactorVertical(out int s32Factor)

Description

Returns the vertical binning factor.

Parameter

- `s32Factor`: Returns the vertical factor.

Class

[uc480.SizeBinning](#)

Accessible

Camera.Size.Binning

Syntax

```
uc480.SizeBinning.GetSupported(out uc480.Defines.BinningMode mode)
```

Description

Returns the supported binning modes.

Parameter

- `mode`: Returns the supported binning modes.

<code>uc480.Defines.BinningMode.Vertical2X</code>	Enables vertical binning with factor 2.
<code>uc480.Defines.BinningMode.Horizontal2X</code>	Enables horizontal binning with factor 2.
<code>uc480.Defines.BinningMode.Vertical3X</code>	Enables vertical binning with factor 3.
<code>uc480.Defines.BinningMode.Horizontal3X</code>	Enables horizontal binning with factor 3.
<code>uc480.Defines.BinningMode.Vertical4X</code>	Enables vertical binning with factor 4.
<code>uc480.Defines.BinningMode.Horizontal4X</code>	Enables horizontal binning with factor 4.
<code>uc480.Defines.BinningMode.Vertical5X</code>	Enables vertical binning with factor 5.
<code>uc480.Defines.BinningMode.Horizontal5X</code>	Enables horizontal binning with factor 5.
<code>uc480.Defines.BinningMode.Vertical6X</code>	Enables vertical binning with factor 6.
<code>uc480.Defines.BinningMode.Horizontal6X</code>	Enables horizontal binning with factor 6.
<code>uc480.Defines.BinningMode.Vertical8X</code>	Enables vertical binning with factor 8.
<code>uc480.Defines.BinningMode.Horizontal8X</code>	Enables horizontal binning with factor 8.
<code>uc480.Defines.BinningMode.Vertical16X</code>	Enables vertical binning with factor 16.
<code>uc480.Defines.BinningMode.Horizontal16X</code>	Enables horizontal binning with factor 16.

Class[uc480.SizeBinning](#)**Accessible**

Camera.Size.Binning

Syntax`uc480.SizeBinning.GetType(out uc480.Defines.BinningMode mode)`**Description**

Indicates whether the camera uses color-proof binning or not.

Parameter

- `mode`: Returns whether the camera uses color-proof binning (`uc480.Defines.BinningMode.Color`) or not (`uc480.Defines.BinningMode.Mono`).

Class[uc480.SizeBinning](#)**Accessible**

Camera.Size.Binning

Syntax`uc480.SizeBinning.Set(uc480.Defines.BinningMode mode)`**Description**

Enables/disables binning. In order to simultaneously enable horizontal and vertical binning, the horizontal and vertical binning parameters can be linked by a logical OR.

Parameter

- `mode`: Mode to be set.

uc480.Defines.BinningMode.Vertical2X	Enables vertical binning with factor 2.
uc480.Defines.BinningMode.Horizontal2X	Enables horizontal binning with factor 2.
uc480.Defines.BinningMode.Vertical3X	Enables vertical binning with factor 3.
uc480.Defines.BinningMode.Horizontal3X	Enables horizontal binning with factor 3.
uc480.Defines.BinningMode.Vertical4X	Enables vertical binning with factor 4.
uc480.Defines.BinningMode.Horizontal4X	Enables horizontal binning with factor 4.
uc480.Defines.BinningMode.Vertical5X	Enables vertical binning with factor 5.
uc480.Defines.BinningMode.Horizontal5X	Enables horizontal binning with factor 5.
uc480.Defines.BinningMode.Vertical6X	Enables vertical binning with factor 6.
uc480.Defines.BinningMode.Horizontal6X	Enables horizontal binning with factor 6.
uc480.Defines.BinningMode.Vertical8X	Enables vertical binning with factor 8.
uc480.Defines.BinningMode.Horizontal8X	Enables horizontal binning with factor 8.
uc480.Defines.BinningMode.Vertical16X	Enables vertical binning with factor 16.
uc480.Defines.BinningMode.Horizontal16X	Enables horizontal binning with factor 16.

3.1.1.31.3 ImageFormat

The `ImageFormat` class provide methods for querying and setting image sizes of a uc480 camera. This is useful for sensors that do not support a free selection of the area of interest or image format. Using [AOI](#), [binning](#), [subsampling](#) or [scaling](#), the driver sets the selected image format to achieve the best possible image quality.



This class is currently only supported by camera models with CMOS sensors and the two CCD sensor models UI-214x and UI-228x.



Note on UI-1008XS image formats

- Image sizes exceeding 1280 x 720 pixels are only available in software trigger mode.
- To disable the software trigger mode, you need to set an image size that is available both in software trigger mode and in live mode. Which image sizes are available in a mode can be queried using `ImageFormat`. For a list of supported image formats see also the UI-1008XS sensor data chapter in the uc480 manual.

Methods

Method	Description
GetArbitraryAOISupported	Returns if the sensor supports a free selection of the area of interest (AOI).
GetList	Returns a list of all image formats supported by the sensor.
Set	Sets the desired image format.

Class

[uc480.SizeTypeFormat](#)

Accessible

Camera.SizeTypeFormat

Syntax

```
uc480.SizeTypeFormat.GetArbitraryAOISupported(out bool bSupported)
```

Description

Returns if the sensor supports a free selection of the area of interest (AOI).

Parameter

bSupported	1 = Free selection of AOI is supported 0 = Free selection of AOI is not supported
------------	--

Class

[uc480.SizeTypeFormat](#)

Accessible

Camera.SizeTypeFormat

Syntax

```
uc480.SizeTypeFormat.GetList(out uc480.Types.ImageFormatInfo[] FormatInfoList)
```

Description

Returns a list of all image formats supported by the sensor.

Parameter

- FormatList: Returns [uc480.Types.ImageFormatInfo](#)

Class

[uc480.SizeTypeFormat](#)

Accessible

Camera.SizeTypeFormat

Syntax

```
uc480.SizeTypeFormat.Set(uint u32Format)
```

Description

Sets the desired image format.

Parameter

- u32Format: [Format ID](#) to be set.

3.1.1.31.4 SensorScaler

The `SensorScaler` class provides methods for using the internal image scaling of some sensors. This allows to reduce the image resolution by adjustable factors. Thus, the amount of data from high resolution sensors can be reduced.

The use of the internal scaler has no effect on the attainable frame rate.



Internal image scaling is only supported by UI-149x/UI-549x and UI-124x/UI-324x/UI-524x series sensors.

Methods

Method	Description
Get	Returns the current mode and scaling factor.
GetFactor	Returns the current scaling factor.
GetInfo	Returns information for image scaling.
GetMode	Returns the current mode.
GetNumOfSteps	Returns the number of steps for the scaling factor.
GetRange	Returns the range of the scaling factor.
Set	Enables/disables image scaling.

Class

[uc480.SizeSensorScaler](#)

Accessible

Camera.Size.SensorScaler

Syntax

```
uc480.SizeSensorScaler.Get(out uc480.Defines.SensorScalerMode mode, out double factor)
```

Description

Returns the current mode and scaling factor.

Parameter

mode	Returns the current scaling mode, see GetMode()
factor	Returns the current scaling factor.

Class

[uc480.SizeSensorScaler](#)

Accessible

Camera.Size.SensorScaler

Syntax

```
uc480.SizeSensorScaler.GetFactor(out double f64Factor)
```

Description

Returns the current scaling factor.

Parameter

- `f64Factor`: Returns the current setting.

Class

[uc480.SizeSensorScaler](#)

Accessible

Camera.Size.SensorScaler

Syntax

```
uc480.SizeSensorScaler.GetInfo(out uc480.Types.SensorScalerInformation Info)
```

Description

Returns information about image scaling.

Parameter

- `Info`: Returns [uc480.Types.SensorScalerInformation](#)

Class

[uc480.SizeSensorScaler](#)

Accessible

Camera.Size.SensorScaler

Syntax

```
uc480.SizeSensorScaler.GetMode(out uc480.Defines.SensorScalerMode mode)
```

Description

Returns the current mode.

Parameter

mode	<code>uc480.Defines.SensorScalerMode.Enable</code> : Enable image scaling
	<code>uc480.Defines.SensorScalerMode.Enable</code> <code>uc480.Defines.SensorScalerMode.AntiAliasing</code> : Enable image scaling with smoothed edges (anti-aliasing effect)
	<code>uc480.Defines.SensorScalerMode.Available</code> : Image scaling is supported

Class

[uc480.SizeSensorScaler](#)

Accessible

Camera.Size.SensorScaler

Syntax

```
uc480.SizeSensorScaler.GetNumOfSteps(out int s32NumOfSteps)
```

Description

Returns the number of steps for the scaling factor.

Parameter

- `s32NumOfSteps`: Returns the number of steps.

Class

[uc480.SizeSensorScaler](#)

Accessible

Camera.Size.SensorScaler

Syntax

```
uc480.SizeSensorScaler.GetRange(out uc480.Types.Range<double> range)
```

```
uc480.SizeSensorScaler.GetRange(out double f64FactorMin, out double f64FactorMax, out double f64FactorInc)
```

Description

Returns the range of the scaling factor.

Parameter

range	Minimum: Returns the minimum value. Maximum: Returns the maximum value. Increment: Returns the increment.
f64FactorMin	Returns the minimum factor.
f64FactorMax	Returns the maximum factor.
f64FactorInc	Returns the increment.

Class

[uc480.SizeSensorScaler](#)

Accessible

Camera.Size.SensorScaler

Syntax

```
uc480.SizeSensorScaler.Set(uc480.Defines.SensorScalerMode mode, double factor)
```

Description

Enables/disables image scaling.

Parameter

mode	Scaling mode, see GetMode ()
factor	Scaling factor

3.1.1.31.5 Subsampling

The `Subsampling` class provides methods for controlling the sub-sampling mode. 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 color 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 "Camera and sensor data" chapter in the uc480 manual.



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 `Subsampling`, calling the following classes is recommended in order to keep the defined camera settings:

- [Framerate](#)
- [Exposure](#)
- If you are using the uc480's flash function: [IO](#)

Methods

Method	Description
Get	Returns the current setting.
GetFactorHorizontal	Returns the horizontal sub-sampling factor.
GetFactorVertical	Returns the vertical sub-sampling factor.
GetSupported	Returns the supported sub-sampling modes.
GetType	Indicates whether the camera uses color-proof sub-sampling.
IsSupported	Returns if the passed subsampling mode is supported.
Set	Enables/disables sub-sampling.

Class

[uc480.SizeSubsampling](#)

Accessible

Camera.Size.Subsampling

Syntax

```
uc480.SizeSubsampling.Get(out uc480.Defines.SubsamplingMode mode)
```

Description

Returns the current setting.

Parameter

- `mode`: Returns the current setting.

Class

[uc480.SizeSubsampling](#)

Accessible

Camera.Size.Subsampling

Syntax

```
uc480.SizeSubsampling.GetFactorHorizontal(out int s32Factor)
```

Description

Returns the horizontal sub-sampling factor.

Parameter

- `s32Factor`: Returns the horizontal factor.

Class

[uc480.SizeSubsampling](#)

Accessible

Camera.Size.Subsampling

Syntax

```
uc480.SizeSubsampling.GetFactorVertical(out int s32Factor)
```

Description

Returns the vertical sub-sampling factor.

Parameter

- `s32Factor`: Returns the vertical factor.

Class

[uc480.SizeSubsampling](#)

Accessible

Camera.Size.Subsampling

Syntax

```
uc480.SizeSubsampling.GetSupported(out uc480.Defines.SubsamplingMode mode)
```

Description

Returns the supported sub-sampling modes linked by logical ORs.

Parameter

- `mode`: Returns the supported sub-sampling modes.

<code>uc480.Defines.SubsamplingMode.Vertical2X</code>	Enables vertical sub-sampling with factor 2.
<code>uc480.Defines.SubsamplingMode.Horizontal2X</code>	Enables horizontal sub-sampling with factor 2.
<code>uc480.Defines.SubsamplingMode.Vertical3X</code>	Enables vertical sub-sampling with factor 3.
<code>uc480.Defines.SubsamplingMode.Horizontal3X</code>	Enables horizontal sub-sampling with factor 3.
<code>uc480.Defines.SubsamplingMode.Vertical4X</code>	Enables vertical sub-sampling with factor 4.
<code>uc480.Defines.SubsamplingMode.Horizontal4X</code>	Enables horizontal sub-sampling with factor 4.
<code>uc480.Defines.SubsamplingMode.Vertical5X</code>	Enables vertical sub-sampling with factor 5.
<code>uc480.Defines.SubsamplingMode.Horizontal5X</code>	Enables horizontal sub-sampling with factor 5.
<code>uc480.Defines.SubsamplingMode.Vertical6X</code>	Enables vertical sub-sampling with factor 6.
<code>uc480.Defines.SubsamplingMode.Horizontal6X</code>	Enables horizontal sub-sampling with factor 6.
<code>uc480.Defines.SubsamplingMode.Vertical8X</code>	Enables vertical sub-sampling with factor 8.
<code>uc480.Defines.SubsamplingMode.Horizontal8X</code>	Enables horizontal sub-sampling with factor 8.
<code>uc480.Defines.SubsamplingMode.Vertical16X</code>	Enables vertical sub-sampling with factor 16.
<code>uc480.Defines.SubsamplingMode.Horizontal16X</code>	Enables horizontal sub-sampling with factor 16.

Class

[uc480.SizeSubsampling](#)

Accessible

`Camera.Size.Subsampling`

Syntax

`uc480.SizeSubsampling.GetType(out uc480.Defines.SubsamplingMode mode)`

Description

Indicates whether the camera uses color-proof sub-sampling.

Parameter

- `mode`: Returns `uc480.Defines.SubsamplingMode.Color` if the camera uses color-proof sub-sampling, else `uc480.Defines.SubsamplingMode.Mono`.

Class

[uc480.SizeSubsampling](#)

Accessible

`Camera.Size.Subsampling`

Syntax

```
uc480.SizeSubsampling.IsEnabled(uc480.Defines.SubsamplingMode mode)
```

Description

Returns if the passed subsampling mode is supported.

Parameter

- `mode`: Mode to be checked if it is supported.

Class

[uc480.SizeSubsampling](#)

Accessible

Camera.Size.Subsampling

Syntax

```
uc480.SizeSubsampling.Set(uc480.Defines.SubsamplingMode mode)
```

Description

Enables/disables sub-sampling. In order to simultaneously enable horizontal and vertical sub-sampling, the horizontal and vertical sub-sampling parameters can be linked by a logical OR.

Parameter

- `mode`: Mode to be set for sub-sampling.

<code>uc480.Defines.SubsamplingMode.Vertical2X</code>	Enables vertical sub-sampling with factor 2.
<code>uc480.Defines.SubsamplingMode.Horizontal2X</code>	Enables horizontal sub-sampling with factor 2.
<code>uc480.Defines.SubsamplingMode.Vertical3X</code>	Enables vertical sub-sampling with factor 3.
<code>uc480.Defines.SubsamplingMode.Horizontal3X</code>	Enables horizontal sub-sampling with factor 3.
<code>uc480.Defines.SubsamplingMode.Vertical4X</code>	Enables vertical sub-sampling with factor 4.
<code>uc480.Defines.SubsamplingMode.Horizontal4X</code>	Enables horizontal sub-sampling with factor 4.
<code>uc480.Defines.SubsamplingMode.Vertical5X</code>	Enables vertical sub-sampling with factor 5.
<code>uc480.Defines.SubsamplingMode.Horizontal5X</code>	Enables horizontal sub-sampling with factor 5.
<code>uc480.Defines.SubsamplingMode.Vertical6X</code>	Enables vertical sub-sampling with factor 6.
<code>uc480.Defines.SubsamplingMode.Horizontal6X</code>	Enables horizontal sub-sampling with factor 6.
<code>Iuc480.Defines.SubsamplingMode.Vertical8X</code>	Enables vertical sub-sampling with factor 8.
<code>uc480.Defines.SubsamplingMode.Horizontal8X</code>	Enables horizontal sub-sampling with factor 8.
<code>uc480.Defines.SubsamplingMode.Vertical16X</code>	Enables vertical sub-sampling with factor 16.
<code>uc480.Defines.SubsamplingMode.Horizontal16X</code>	Enables horizontal sub-sampling with factor 16.

3.1.1.32 TestImage

The `TestImage` class provides methods for controlling the test image function.

Methods

Method	Description
<code>Get</code>	Returns the camera test images.
<code>GetRange</code>	Returns the range for the additional parameters of some camera test images.
<code>GetSupported</code>	Returns all test images supported by the camera.
<code>IsSupported</code>	Returns is the given test image is supported.
<code>Set</code>	Enables/duisables the test image function in the sensor.

3.1.1.32.1 Get

Class

[uc480.TestImage](#)

Accessible

`Camera.TestImage`

Syntax

```
uc480.TestImage.Get(int s32Image, out int s32Param)
```

Description

Returns the camera test images. You can enable the sensor test image feature using [Set\(\)](#).

Parameter

s32Image	Returns the set test image
s32Param	Returns the additional parameter for modifying the test image. Not available for all test images.

3.1.1.32.2 GetRange

Class

[uc480.TestImage](#)

Accessible

Camera.TestImage

Syntax

```
uc480.TestImage.GetRange(uc480.Defines.TestImageMode mode, out uc480.Types.Range<int> range)
uc480.TestImage.GetRange(uc480.Defines.TestImageMode mode, out int s32Min, out int s32Max)
```

Description

Returns the range for the additional parameters of some camera test images. You can enable the sensor test image feature using [Set\(\)](#).

Parameter

mode	Test image for which the range of the additional parameters is to be queried (see GetSupported())
range	Minimum: Returns the minimum value. Maximum: Returns the maximum value. Increment: Returns the increment.
s32Min	Returns the minimum value.
s32Max	Returns the maximum value.

3.1.1.32.3 GetSupported

Class

[uc480.TestImage](#)

Accessible

Camera.TestImage

Syntax

```
uc480.TestImage.GetSupported(out uc480.Defines.TestImageMode mode)
```

Description

Returns all test images supported by the camera. You can enable the sensor test image feature using [Set\(\)](#).

Parameter

- mode: Returns the test images supported by the camera.

uc480.Defines.TestImageMode.None	No test image
uc480.Defines.TestImageMode.White	White image
uc480.Defines.TestImageMode.Black	Black image
uc480.Defines.TestImageMode.HorizontalGreyscale	Horizontal grayscale
uc480.Defines.TestImageMode.VerticalGreyscale	Vertical grayscale
uc480.Defines.TestImageMode.DiagonalGreyscale	Diagonal grayscale
uc480.Defines.TestImageMode.WedgeGray	Gray wedges, generated by the GigE uc480
uc480.Defines.TestImageMode.WedgeGraySensor	Gray wedges, generated by the sensor
uc480.Defines.TestImageMode.WedgeColor	Color wedges
uc480.Defines.TestImageMode.AnimatedWedgeGray	Gray wedges, animated, generated by the GigE uc480
uc480.Defines.TestImageMode.AnimatedWedgeGraySensor	Gray wedges, animated, generated by the sensor
uc480.Defines.TestImageMode.AnimatedWedgeColor	Color wedges, animated
uc480.Defines.TestImageMode.ColorBars_1	Color bars
uc480.Defines.TestImageMode.GreyAndColorBars	Gray and color bars
uc480.Defines.TestImageMode.MovingGreyAndColorBars	Gray and color bars, animated
uc480.Defines.TestImageMode.AnimatedLine	Line, animated
uc480.Defines.TestImageMode.AlternatePattern	Alternating pattern (raw Bayer mode only)
uc480.Defines.TestImageMode.RampingPattern	Diagonal color pattern
uc480.Defines.TestImageMode.MonochromeHorizontalBars	Monochrome bars, horizontal
uc480.Defines.TestImageMode.MonochromeVerticalBars	Monochrome bars, vertical

uc480.Defines.TestImageMode.ColdpixelGrid	Camera image overlaid with a grid of blue dots
uc480.Defines.TestImageMode.HotpixelGrid	Camera image overlaid with a grid of red dots
uc480.Defines.TestImageMode.VariableGrey	Adjustable grayscale image
uc480.Defines.TestImageMode.VariableRedPart	Image with adjustable red content
uc480.Defines.TestImageMode.VariableGreenPart	Image with adjustable green content
uc480.Defines.TestImageMode.VariableBluePart	Image with adjustable blue content

3.1.1.32.4 IsSupported

Class

[uc480.TestImage](#)

Accessible

Camera.TestImage

Syntax

`uc480.TestImage.IsSupported(uc480.Defines.TestImageMode mode)`

Description

Returns is the given test image is supported.

Parameter

- `mode`: Test image to be queried to be supported (see [GetSupported\(\)](#)).

3.1.1.32.5 Set

Class

[uc480.TestImage](#)

Accessible

Camera.TestImage

Syntax

`uc480.TestImage.Set(uc480.Defines.TestImageMode mode, int s32Param)`

Description

Enables/duisables the test image function in the sensor. You can select different test images. The test images supported by a particular camera can be queried using [GetSupported\(\)](#). For some test images, the `s32Param` parameter provides additional options. If the test image does not support additional parameters, `s32Param` will be ignored.



Manually changing the pixel clock will disable the test image mode.

Parameter

mode	Test image to be set, see GetSupported()
s32Param	Additional parameter used for modifying the test image. Not available for all test images.

3.1.1.33 Timeout

The `Timeout` class provides methods for controlling the user-defined timeout values of the uc480 API.

Methods

Method	Description
Get	Returns the user-defined timeout values of the uc480 API.
Set	Sets the user-defined timeout values of the uc480 API.

3.1.1.33.1 Get**Class**[uc480.Timeout](#)**Accessible**

Camera.Timeout

Syntax

```
uc480.Timeout.Get(uc480.Defines.TimeoutMode timeout, out uint u32Timeout)
```

Description

Returns the user-defined timeout values from the uc480 API.

Parameter

timeout	Returns the mode of the timeout value: <ul style="list-style-type: none"> • <code>uc480.Defines.TimeoutMode.Trigger</code>: Returns the timeout value in steps of 10 ms for triggered image capture.
u32Timeout	Returns the timeout value. Returns 0 if the default value of the uc480 API is used.

3.1.1.33.2 Set**Class**[uc480.Timeout](#)**Accessible**

Camera.Timeout

Syntax

```
uc480.Timeout.Set(uc480.Defines.TimeoutMode timeout, uint u32Timeout)
```

Description

Sets the user-defined timeout values of the uc480 API. If no user-defined timeout is set, the default value of the uc480 API is used for the relevant timeout.



The user-defined timeout only applies to the specified camera at runtime of the program.

Parameter

timeout	Timeout value to be set • uc480.Defines.TimeoutMode.Trigger: Sets the timeout value for triggered image capture
u32Timeout	Timeout value in 10 ms. Value range [0; 4...429496729] (corresponds to 40 ms to approx. 1193 hours) 0 = use default value of the uc480 API For 1...3, the value 4 is used.

3.1.1.34 Timing

The `Timing` class provides methods for controlling the camera timing settings. The following classes and method exist:

- [Exposure](#)
- [Framerate](#)
- [PixelClock](#)
- [VsyncCount](#)

Methods

Method	Description
Optimal()	Sets the highest possible pixel clock.

3.1.1.34.1 Exposure

The `Exposure` class provides methods for controlling the exposure of a uc480 camera.

Note on dependencies on other settings

The use of the following classes will affect the exposure time:

- [PixelClock](#)
- [Optimal\(\)](#)
- [Framerate](#)
- [AOI](#) (if the image size is changed)
- [Subsampling](#)
- [Binning](#)



Changes made to the image size, the frame rate or the pixel clock frequency also affect the exposure time. For this reason, you need to call `Exposure` again after such changes.

Note on new driver versions

Newer driver versions sometimes allow an extended value range for the exposure time setting. We recommend querying the value range every time and set the exposure time explicitly.

Applying new settings

In freerun mode, any modification of the exposure time will only become effective when

the image after next is captured. In trigger mode, the modification will be applied to the next image.

Accuracy of the exposure time setting

The increments for setting the exposure time depend on the sensor's current timing settings (pixel clock, frame rate). The smallest increment usually corresponds to the duration of one pixel row, which is the time it takes the sensor to read out one pixel row.

You can query the actual exposure time setting with [Get\(\)](#).

Some sensors allow setting the exposure time in smaller increments. Using [GetSupported\(\)](#), you can check whether your sensor supports these methods.

For minimum and maximum exposure times as well as other sensor-based dependencies, please refer to the "Camera and sensor data" chapter in the uc480 manual.

Rounding errors from increments

When calculating a new exposure time based on the returned increment, note that calculations with floating point values in the PC will always be subject to rounding errors. Therefore, an addition or subtraction of the increment value might not always produce the exact desired result. In this case, the uc480 API rounds down the floating point value and sets the exposure time to the next lower value.

You can avoid this behavior by additionally adding the value `INCREMENT/2.f` (half increment) when calculating with the increment. This ensures that the desired value will be set even after rounding.

The following classes and methods exist:

- [Dual](#)
- [Fine](#)
- [Long](#)

Methods

Method	Description
Get	Returns the currently set exposure time (in ms).
GetDefault	Returns the default setting for the exposure time.
GetRange	Returns the exposure time range.
GetSupported	Returns the if the exposure time is supported.
Set	Sets the exposure time.

The `Dual` class provides methods for controlling the dual exposure of some uc480 cameras. For setting the exposure time use the [Exposure](#) class.



Note: This feature is currently only supported by the models UI-336x/UI-536x and UI-337x/UI-537x.

Methods

Method	Description
Get	Returns the ratio between exposure times for dual exposure.
GetDefault	Returns the default ratio for dual exposure.
GetRange	Returns the range of the dual exposure.
GetSupported	Returns is setting the dual exposure is supported.
Set	Sets the ration between exposure times for dual exposure.

Class

[uc480.TimingExposureDual](#)

Accessible

Camera.Timing.Exposure.Dual

Syntax

```
uc480.TimingExposureDual.Get(out uint value)
```

Description

Returns the ratio between exposure times for dual exposure (see also [Set\(\)](#)).

Parameter

- `value`: Returns the current ratio.

Class

[uc480.TimingExposureDual](#)

Accessible

Camera.Timing.Exposure.Dual

Syntax

```
uc480.TimingExposureDual.GetDefault(out uint value)
```

Description

Returns the default ratio for dual exposure.

Parameter

- `value`: Returns the default ratio for dual exposure.

Class

[uc480.TimingExposureDual](#)

Accessible

Camera.Timing.Exposure.Dual

Syntax

```
uc480.TimingExposureDual.GetRange(out uc480.Types.Range<uint> range)
```

Description

Returns the range of the dual exposure.

Parameter

range	Minimum: returns the minimum value Maximum: returns the maximum value Increment: returns the increment
min	Returns the minimum value.
max	Returns the maximum value.
inc	Returns the increment.

Class

[uc480.TimingExposureDual](#)

Accessible

Camera.Timing.Exposure.Dual

Syntax

```
uc480.TimingExposureDual.GetSupported(out bool supported)
```

Description

Returns is setting the dual exposure is supported.

Parameter

supported	0 = Not supported. 1 = Supported.
-----------	--------------------------------------

Class

[uc480.TimingExposureDual](#)

Accessible

Camera.Timing.Exposure.Dual

Syntax

```
uc480.TimingExposureDual.Set(uint value)
```

Description

Sets the ratio between exposure times for dual exposure. For models UI-336x/UI-536x and UI-337x/UI-537x select a percental value between 10 and 100. E.g. 50 means that odd lines are exposed at the selected exposure time and even lines are exposed with 50% of the selected exposure time.

Parameter

- value: Ratio to be set

The `Fine` class provides methods for controlling the exposure time of a uc480 camera in fine increments. For setting the exposure time use the [Exposure](#) class.

Methods

Method	Description
GetRange	Returns the exposure time range in fine increments for some sensors.
GetSupported	Returns if the sensor supports fine increments for adjusting the exposure time.

Class

[uc480.TimingExposureFine](#)

Accessible

Camera.Timing.Exposure.Fine

Syntax

```
uc480.TimingExposureFine.GetRange(out uc480.Types.Range<double> range)
uc480.TimingExposureFine.GetRange(out double f64Min, out double f64Max, out double f64Inc)
```

Description

Returns the exposure time range in fine increments for some sensors.

Parameter

range	Minimum: Returns the minimum value. Maximum: Returns the maximum value. Increment: Returns the increment.
f64Min	Returns the minimum value.
f64Max	Returns the maximum value.
f64Inc	Returns the increment.

Class

[uc480.TimingExposureFine](#)

Accessible

Camera.Timing.Exposure.Fine

Syntax

```
uc480.TimingExposureFine.GetSupported(out bool bSupported)
```

Description

Returns if the sensor supports fine increments for adjusting the exposure time.

Parameter

bSupported	1 = Fine increment is supported 0 = Fine increment is not supported
------------	--

The `Long` class provides methods for controlling the long exposure of a uc480 camera. For setting the exposure time use the [Exposure](#) class.

Methods

Method	Description
GetEnable	Returns the current settings for long exposure.
GetRange	Returns the value range for long exposure.
GetSupported	Returns is long time exposure is supported.
SetEnable	Enables/Disables long exposure.

Class

[uc480.TimingExposureLong](#)

Accessible

Camera.Timing.Exposure.Long

Syntax

```
uc480.TimingExposureLong.GetEnable(out bool bEnable)
```

Description

Returns the current settings for long exposure.

Parameter

bEnable	1 = Long time exposure is enabled 0 = Long time exposure is disabled
---------	---

Class

[uc480.TimingExposureLong](#)

Accessible

Camera.Timing.Exposure.Long

Syntax

```
uc480.TimingExposureLong.GetRange(out uc480.Types.Range<double> range)  
uc480.TimingExposureLong.GetRange(out double f64Min, out double f64Max, out double f64Inc)
```

Description

Returns the value range for long exposure.

Parameter

range	Minimum: Returns the minimum value. Maximum: Returns the maximum value. Increment: Returns the increment.
f64Min	Returns the minimum value.
f64Max	Returns the maximum value.
f64Inc	Returns the increment.

Class

[uc480.TimingExposureLong](#)

Accessible

Camera.Timing.Exposure.Long

Syntax

```
uc480.TimingExposureLong.GetSupported(out bool bSupported)
```

Description

Returns is long time exposure is supported.

Parameter

bSupported	1 = Long time exposure is supported
	0 = Long time exposure is not supported

Class

[uc480.TimingExposureLong](#)

Accessible

Camera.Timing.Exposure.Long

Syntax

```
uc480.TimingExposureLong.SetEnable(bool bEnable)
```

Description

Enables/Disables long exposure.

Parameter

bEnable	1 = Enable long time exposure
	0 = Disable long time exposure

Class

[uc480.TimingExposure](#)

Accessible

Camera.Timing.Exposure

Syntax

```
uc480.TimingExposure.Get(out double f64Value)
```

Description

Returns the currently set exposure time (in ms).

Parameter

- f64Value: Returns the current value.

Class

[uc480.TimingExposure](#)

Accessible

Camera.Timing.Exposure

Syntax

`uc480.TimingExposure.GetDefault(out double f64Value)`

Description

Returns the default setting for the exposure time.

Parameter

- `f64Value`: Returns the default value.

Class

[uc480.TimingExposure](#)

Accessible

Camera.Timing.Exposure

Syntax

`uc480.TimingExposure.GetRange(out uc480.Types.Range<double> range)`
`uc480.TimingExposure.GetRange(out double f64Min, out double f64Max, out double f64Inc)`

Description

Returns the exposure time range (minimum, maximum and increment).

Parameter

range	Minimum: Returns the minimum value.
	Maximum: Returns the maximum value.
	Increment: Returns the increment.
f64Min	Returns the minimum value.
f64Max	Returns the maximum value.
f64Inc	Returns the increment.

Class

[uc480.TimingExposure](#)

Accessible

Camera.Timing.Exposure

Syntax

`uc480.TimingExposure.GetSupported(out bool bSupported)`

Description

Returns if setting the exposure time is supported.

Parameter

bSupported	1 = Exposure time is supported 0 = Exposure time is not supported
------------	--

Class[uc480.TimingExposure](#)**Accessible**

Camera.Timing.Exposure

Syntax

uc480.TimingExposure.Set(double f64Value)

Description

Sets the exposure time. If 0 is passed, the exposure time is set to the maximum value of 1/frame rate.

Parameter

- `f64Value`: Exposure time in ms.

3.1.1.34.2 Framerate

The `Framerate` class provides methods for controlling the frame rate of a uc480 camera.

Methods

Method	Description
Get	Returns the current setting of the frame rate.
GetCurrentFps	In live capture mode, the method returns the number of frames actually captured per second.
GetDefault	Returns the default frame rate.
GetFrameRateRange	Returns the minimum and maximum frame rate and the increment.
GetFrameTimeRange	Returns the possible frame rate settings which are available for the current pixel clock setting.
Set	Sets the sensor frame rate in freerun mode.

Class[uc480.TimingFramerate](#)**Accessible**

Camera.Timing.Framerate

Syntax

uc480.TimingFramerate.Get(out double f64Value)

Description

Returns the currently set value of the frame rate.

Parameter

- `f64Value`: Returns the current value.

Class

[uc480.TimingFramerate](#)

Accessible

Camera.Timing.Framerate

Syntax

```
uc480.TimingFramerate.GetCurrentFps(out double f64Value)
```

Description

In live capture mode, the method returns the number of frames actually captured per second.

Parameter

- `f64Value`: Returns the current value reached.

Class

[uc480.TimingFramerate](#)

Accessible

Camera.Timing.Framerate

Syntax

```
uc480.TimingFramerate.GetDefault(out double f64Value)
```

Description

Returns the default frame rate.

Parameter

- `f64Value`: Returns the default frame rate.

Class

[uc480.TimingFramerate](#)

Accessible

Camera.Timing.Framerate

Syntax

```
uc480.TimingFramerate.GetFrameRateRange(out uc480.Types.Range<double> range)
uc480.TimingFramerate.GetFrameRateRange(out double f64MinFramerate,
                                         out double f64MaxFramerate,
                                         out double f64IncFramerate)
```

Description

Returns the minimum and maximum frame rate and the increment.

Parameter

range	Minimum: Returns the minimum value. Maximum: Returns the maximum value. Increment: Returns the increment.
f64MinFramerate	Returns the minimum frame rate.
f64MaxFramerate	Returns the maximum frame rate.
f64IncFramerate	Returns the increment.

Class

[uc480.TimingFramerate](#)

Accessible

Camera.Timing.Framerate

Syntax

```
uc480.TimingFramerate.GetFrameTimeRange(out uc480.Types.Range<double> range)
uc480.TimingFramerate.GetFrameTimeRange(out double f64MinFrametime,
                                         out double f64MaxFrametime,
                                         out double f64IncFrametime)
```

Description

Using this method, 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 a minimum and maximum frame time in increments defined by the increment parameter.

The following applies:

$$\begin{aligned}fps_{\min} &= \frac{1}{\max} \\fps_{\max} &= \frac{1}{\min} \\fps_n &= \frac{1}{(\min + n * \text{intervall})}\end{aligned}$$

The call of this method makes only sense in the freerun mode.

Parameter

range	Minimum: Returns the minimum value. Maximum: Returns the maximum value. Increment: Returns the increment.
f64MinFrametime	Returns the minimum available frame duration in seconds.
f64MaxFrametime	Returns the maximum available frame duration in seconds.
f64IncFrametime	Returns the increment.

Class

[uc480.TimingFramerate](#)

Accessible

Camera.Timing.Framerate

Syntax

```
uc480.TimingFramerate.Set(double f64Value, out double f64newValue)
uc480.TimingFramerate.Set(double f64Value)
```

Description

Using this method, 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 method, the actual frame rate is returned through the `f64newValue` 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 "Camera and sensor data" chapter in the uc480 manual.

 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 `Set()`, calling `Exposure` is recommended in order to keep the defined camera settings.

The use of the following classes/methods will affect the frame rate:

- [PixelClock](#)
- [Optimal\(\)](#)
- [AOI](#) (if the image size is changed)
- [Subsampling](#)
- [Binning](#)

 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 `Set()` again after such changes.

 To be able to set the default frame rate, you have to set a pixel clock equal to or higher than the default pixel clock.

Parameter

<code>f64Value</code>	Desired frame rate in frames per second (fps)
<code>f64newValue</code>	Returns the frame rate actually set.

3.1.1.34.3 PixelClock

The `Pixelclock` class provides methods for controlling the pixel clock of a uc480 camera. The pixel clock is the frequency at which the image data is read-out from the sensor.

Methods

Method	Description
<u>Get</u>	Returns the current set pixel clock in MHz.
<u>GetDefault</u>	Returns the default pixel clock.
<u>GetList</u>	Returns the list with discrete pixel clocks of the camera.
<u>GetNumber</u>	Returns the number of discrete pixel clock which are supported by the camera.
<u>GetRange</u>	Returns the range for the pixel clock.
<u>Set</u>	Sets the pixel clock in MHz.

Class

[uc480.TimingPixelClock](#)

Accessible

Camera.Timing.PixelClock

Syntax

```
uc480.TimingPixelClock.Get(out int s32Value)
```

Description

Returns the current set pixel clock in MHz.

Parameter

- `s32Value`: Returns the current value.

Class

[uc480.TimingPixelClock](#)

Accessible

Camera.Timing.PixelClock

Syntax

```
uc480.TimingPixelClock.GetDefault(out int s32Value)
```

Description

Returns the default pixel clock.

Parameter

- `s32Value`: Returns the default value.

Class[uc480.TimingPixelClock](#)**Accessible**

Camera.Timing.PixelClock

Syntax

uc480.TimingPixelClock.GetList(out int[] pixelclockList)

Description

Returns the list with discrete pixel clocks of the camera.

Parameter

- `pixelclockList`: Returns the discrete pixel clock list.

Class[uc480.TimingPixelClock](#)**Accessible**

Camera.Timing.PixelClock

Syntax

uc480.TimingPixelClock.GetNumber(out int number)

Description

Returns the number of discrete pixel clock which are supported by the camera.

Parameter

- `number`: Returns the number of discrete pixel clocks.

Class[uc480.TimingPixelClock](#)**Accessible**

Camera.Timing.PixelClock

Syntaxuc480.TimingPixelClock.GetRange(out uc480.Types.Range<int> range)
uc480.TimingPixelClock.GetRange(out int s32Min, out int s32Max, out int s32Inc)**Description**

Returns the range for the pixel clock. 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 uc480 manual.

Parameter

range	Minimum: Returns the minimum value. Maximum: Returns the maximum value. Increment: Returns the increment.
s32Min	Returns the minimum value.
s32Max	Returns the maximum value.
s32Inc	Returns the increment.

Class

[uc480.TimingPixelClock](#)

Accessible

Camera.Timing.PixelClock

Syntax

```
uc480.TimingPixelClock.Set(int s32Value)
```

Description

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.

Parameter

- s32Value: Value to be set for the pixel clock in MHz.

3.1.1.34.4 VsyncCount

The `VsyncCount` class provides methods for reading-out the VSYNC counter.

Methods

Get	Returns the VSYNC counter
---------------------	---------------------------

Class

[uc480.TimingVsyncCount](#)

Accessible

Camera.Timing.VsyncCount

Syntax

```
uc480.TimingVsyncCount.Get(out long s64Vsync, out long s64Fsync)
```

Description

Reads out the VSYNC counter. It will be incremented by 1 each time the sensor starts capturing an image.

Parameter

s64Vsync	Current VSYNC count
s64Fsync	Current Frame SYNC count

3.1.1.34.5 Optimal**Class**[uc480.Timing](#)**Accessible**

Camera.Timing

Syntax

```
uc480.Timing.Optimal(int s32Timeout, out int s32MaxPclk, out double f64MaxFramerate)
```

Description

Using this method, you can determine the highest possible pixel clock frequency (with full resolution) for the current configuration. This method 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. This method can only be executed in freerun mode.



The methods should be executed in a separate thread and run in the background to allow for the computational load caused by additional color conversions, etc. Otherwise, it will not able to return the optimum values.



Changes to the image geometry or pixel clock affect the value ranges of the frame rate and exposure time. After executing `Optimal()`, calling the following classes/methods is recommended in order to keep the defined camera settings:

- Set the frame rate via [Set\(\)](#)
- [Exposure](#)
- If you are using the uc480's flash function: [IO\(\)](#)
- For starting the image acquisition in freerun mode: [Capture\(\)](#)

Parameter

s32Timeout [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 method correspondingly.
s32MaxPclk	Returns the maximum pixel clock frequency (in MHz).
f64MaxFramerate	Returns the maximum frame rate (in fps).

3.1.1.35 Trigger

The `Trigger` class provides methods for activating 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 [Freeze\(\)](#) is called, or a continuous triggered capture is started when [Capture\(\)](#) is called. In hardware trigger

mode, you can use the [Force\(\)](#) command to trigger an image capture even if no electric signal is present.

When you disable the trigger functionality, you can query the signal level at the trigger input. This option causes the camera to change to freerun mode.



For hardware reasons, the board-level versions of the USB uc480 LE cameras can only be triggered on the falling edge.

Notes on using the UI-1008XS in trigger mode

- The UI-1008XS currently supports the following parameters:
 - uc480.Defines.TriggerMode.Off
 - uc480.Defines.TriggerMode.Software
- The continuous trigger mode with [Capture\(\)](#) is not supported.
- Triggered image capture using [Freeze\(\)](#) with the uc480.Defines.DeviceParameter.DontWait parameter is not supported. Please use the uc480.Defines.DeviceParameter.Wait parameter instead.
- In software trigger mode, the exposure time cannot be changed manually.
- The delay between software trigger and image capture is higher than with other cameras and can be up to 1 second due to the sensor. If you have enabled auto focus and auto exposure shutter/auto gain control, the image will be captured as soon as adjustment is complete.
- Image sizes exceeding 1280 x 720 pixels are only available in software trigger mode.
- To disable the software trigger mode, you first need to set an image size that is also available in live mode. Which image sizes are available in a mode can be queried using the [ImageFormat](#) class.



The following classes and methods exist:

- [Burst](#)
- [Counter](#)
- [Debounce](#)
- [Delay](#)

Methods

Method	Description
Force	Forces a software-controlled capture of an image while a capturing process triggered by hardware is in progress.
Get	Returns the current trigger mode.
GetStatus	Returns the current signal level at the trigger input.
GetSupported	Returns the supported trigger modes.
Set	Sets the trigger mode.

3.1.1.35.1 Burst

The `Burst` class provides methods for activating the burst trigger mode in GigE uc480 cameras. In burst trigger mode, the camera captures a series of images in rapid succession on receipt of a single trigger signal. The trigger signal can be generated by the software ([Freeze\(\)](#)) or transmitted via the digital input of the camera. The burst images are captured and transferred at maximum speed. The maximum speed depends on the pixel clock parameter (see [Pixelclock](#)) and the exposure time parameter (see [Exposure](#)). [Transfer](#) allows adjusting the latency of image data transfer.



This class is currently only supported by the GigE uc480 camera series.



Note on trigger delay in burst trigger mode

If you set a trigger delay with [Set\(\)](#), the delay will only apply to the first image after each trigger signal.

Methods

Method	Description
Get	Returns the currently set number of images in a burst.
GetDefault	Returns the default number of images in a burst.
GetRange	Returns the minimum and maximum value the increment for the number of images in a burst.
GetSupported	Returns if the camera supports the burst trigger mode.
Set	Sets the number of images in a burst.

Class

[uc480.TriggerBurst](#)

Accessible

Camera.Trigger.Burst

Syntax

```
uc480.TriggerBurst.Get(out uint u32BurstSize)
```

Description

Returns the currently set number of images in a burst.

Parameter

- `u32BurstSize`: Returns the current number.

Class

[uc480.TriggerBurst](#)

Accessible

Camera.Trigger.Burst

Syntax

```
uc480.TriggerBurst.GetDefault(out uint u32Value)
```

Description

Returns the default number of images in a burst.

Parameter

- `u32Value`: Returns the default number.

Class

[uc480.TriggerBurst](#)

Accessible

Camera.Trigger.Burst

Syntax

```
uc480.TriggerBurst.GetRange(out uc480.Types.Range<uint> range)
uc480.TriggerBurst.GetRange(out uint u32Min, out uint u32Max, out uint u32Inc)
```

Description

Returns the minimum and maximum value the increment for the number of images in a burst.

Parameter

range	Minimum: Returns the minimum value. Maximum: Returns the maximum value. Increment: Returns the increment.
u32Min	Returns the minimum value.
u32Max	Returns the maximum value.
u32Inc	Returns the increment.

Class

[uc480.TriggerBurst](#)

Accessible

Camera.Trigger.Burst

Syntax

```
uc480.TriggerBurst.GetSupported(out bool bSupported)
```

Description

Returns if the camera supports the burst trigger mode.

Parameter

bSupported	1 = Burst trigger mode is supported. 0 = Burst trigger mode is not supported.
------------	--

Class

[uc480.TriggerBurst](#)

Accessible

Camera.Trigger.Burst

Syntax

```
uc480.TriggerBurst.Set(uint u32BurstSize)
```

Description

Sets the number of images in a burst.

Parameter

- `u32BurstSize`: Number to be set.

3.1.1.35.2 Counter

The `Counter` class provides methods for getting the number of images captured in hardware or software trigger mode.



In freerun mode, the counter always returns 0 even when images were captured.



This class is currently only supported by USB 2.0 uc480 cameras.

Methods

Method	Description
<u>Get</u>	Returns the current count for triggered image captures.
<u>Reset</u>	Resets the counter for triggered image captures.

Class

[uc480.TriggerCounter](#)

Accessible

`Camera.Trigger.Counter`

Syntax

```
uc480.TriggerCounter.Get(out int s32Value)
```

Description

Returns the current count for triggered image captures.



This class is currently only supported by USB 2.0 uc480 cameras.

Parameter

- `s32Value`: Returns the current count.

Class

[uc480.TriggerCounter](#)

Accessible

Camera.Trigger.Counter

Syntax

`uc480.TriggerCounter.Reset()`

Description

Resets the counter for triggered image captures.



This class is currently only supported by USB 2.0 uc480 cameras.

Parameter

None

3.1.1.35.3 Debounce

The `Debounce` class provides methods for suppressing disturbances at the digital input when you are running a GigE uc480 camera in trigger mode. The signal at the digital input is only recognized as a trigger if the signal level remains constant at the target level for a user-selectable time. The signal edge and a delay can be set as parameters. It is recommended to use automatic signal edge selection.

Example: Mode set to "rising edge" (`uc480.Defines.TriggerDebounceMode.RisingEdge`) and delay set to 50 µs. The camera will not trigger the image capture on the rising edge until the digital signal has remained at the high level for longer than 50 µs without interruption. If this is not the case, the signal is regarded as a disturbance and ignored.



[SetDelayTime\(\)](#) delays the start of a triggered image capture by the selected time.



This class is currently only supported by the GigE uc480 camera series.

Methods

Method	Description
Get	Returns the set mode.
GetDefault	Returns the default mode.
GetDelayTime	Returns the set delay time.
GetDelayTimeDefault	Returns the default delay time.
GetDelayTimeRange	Returns the range for the delay time.
GetSupported	Returns the supported modes.
Set	Sets the mode.
SetDelayTime	Sets the delay time.

Class[uc480.TriggerDebounce](#)**Accessible**

Camera.Trigger.Debounce

Syntax

uc480.TriggerDebounce.Get(out uc480.Defines.TriggerDebounceMode mode)

Description

Returns the set mode.

Parameter

- mode: Returns the current set mode (see [Set\(\)](#)).

Class[uc480.TriggerDebounce](#)**Accessible**

Camera.Trigger.Debounce

Syntax

uc480.TriggerDebounce.GetDefault(out uc480.Defines.TriggerDebounceMode mode)

Description

Returns the default mode.

Parameter

- mode: Returns the default mode.

Class[uc480.TriggerDebounce](#)**Accessible**

Camera.Trigger.Debounce

Syntax

uc480.TriggerDebounce.GetDelayTime(out uint u32Value)

Description

Returns the set delay time (in μ s).

Parameter

- u32Value: Returns the current delay time.

Class[uc480.TriggerDebounce](#)**Accessible**

Camera.Trigger.Debounce

Syntax

uc480.TriggerDebounce.GetDelayTimeDefault(out uint u32Value)

Description

Returns the default delay time (in μ s).

Parameter

- `u32Value`: Returns the default delay time.

Class

[uc480.TriggerDebounce](#)

Accessible

Camera.Trigger.Debounce

Syntax

```
uc480.TriggerDebounce.GetDelayTimeRange(out uc480.Types.Range<uint> range)
uc480.TriggerDebounce.GetDelayTimeRange(out uint u32Min, out uint u32Max, out uint u32Inc)
```

Description

Returns the range for the delay time (in μ s).

Parameter

range	Minimum: Returns the minimum value. Maximum: Returns the maximum value. Increment: Returns the increment.
u32Min	Returns the minimum delay time.
u32Max	Returns the maximum delay time.
u32Inc	Returns the increment.

Class

[uc480.TriggerDebounce](#)

Accessible

Camera.Trigger.Debounce

Syntax

```
uc480.TriggerDebounce.GetSupported(out uc480.Defines.TriggerDebounceMode mode)
```

Description

Returns the supported modes.

Parameter

- `mode`: Supported mode (see [Set\(\)](#))

Class

[uc480.TriggerDebounce](#)

Accessible

Camera.Trigger.Debounce

Syntax

```
uc480.TriggerDebounce.Set(uc480.Defines.TriggerDebounceMode mode)
```

Description

Sets the mode.

Parameter

- mode: Mode to be set.

uc480.Defines.TriggerDebounceMode.None	Disables debouncing the digital input.
uc480.Defines.TriggerDebounceMode.Falling Edge	Debounces the digital input for falling edge signals.
uc480.Defines.TriggerDebounceMode.RisingEdge	Debounces the digital input for rising edge signals.
uc480.Defines.TriggerDebounceMode.BothEdges	Debounces the digital input for rising or falling edge signals.
uc480.Defines.TriggerDebounceMode.Automatic	Debounces the digital input with automatic edge selection (recommended). The edge is selected based on the set trigger edge (see Set()).

Class

[uc480.TriggerDebounce](#)

Accessible

Camera.Trigger.Debounce

Syntax

```
uc480.TriggerDebounce.SetDelayTime(uint u32Value)
```

Description

Sets the delay time (in μ s).

Parameter

- u32Value: Delay time to be set

3.1.1.35.4 Delay

The `Delay` class provides methods for setting 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 "Camera and sensor data" chapter in the uc480 manual.

Methods

Method	Description
Get	Returns the currently set delay time.
GetRange	Returns the range for the delay time.
Set	Sets the time by which the image capture is delayed (in μ s)

Class

[uc480.TriggerDelay](#)

Accessible

Camera.Trigger.Delay

Syntax

```
uc480.TriggerDelay.Get(out int s32Value)
```

Description

Returns the currently set delay time.

Parameter

- `s32Value`: Returns the current delay time.

Class

[uc480.TriggerDelay](#)

Accessible

Camera.Trigger.Delay

Syntax

```
uc480.TriggerDelay.GetRange(out uc480.Types.Range<int> range)
uc480.TriggerDelay.GetRange(out int s32Min, out int s32Max, out int s32Inc)
```

Description

Returns the range for the delay time.

Parameter

range	Minimum: Returns the minimum value. Maximum: Returns the maximum value. Increment: Returns the increment.
s32Min	Returns the minimum delay time.
s32Max	Returns the maximum delay time.
s32Inc	Returns the increment.

Class

[uc480.TriggerDelay](#)

Accessible

Camera.Trigger.Delay

Syntax

```
uc480.TriggerDelay.Set(int s32Value)
```

Description

Sets the time by which the image capture is delayed (in μ s). "0" deactivates the trigger delay.

Parameter

- `s32Value`: Delay time to be set.

3.1.1.35.5 Force**Class**

[uc480.Trigger](#)

Accessible

Camera.Trigger

Syntax

`uc480.Trigger.Force()`

Description

You can use `Force()` to force a software-controlled capture of an image while a capturing process triggered by hardware is in progress. This method can only be used if the triggered capturing process was started using the `uc480.Defines.DeviceParameter.DontWait` parameter.

Parameter

None

3.1.1.35.6 Get**Class**

[uc480.Trigger](#)

Accessible

Camera.Trigger

Syntax

`uc480.Trigger.Get(out uc480.Defines.TriggerMode mode)`

Description

Returns the current set trigger mode.

Parameter

- `mode`: Returns the current setting.

3.1.1.35.7 GetStatus**Class**

[uc480.Trigger](#)

Accessible

Camera.Trigger

Syntax

`uc480.Trigger.GetStatus(out int s32Value)`

Description

Returns the current signal level at the trigger input.

Parameter

- `s32Value`: Returns the signal level.

3.1.1.35.8 GetSupported**Class**

[uc480.Trigger](#)

Accessible

Camera.Trigger

Syntax

```
uc480.Trigger.GetSupported(out uc480.Defines.TriggerMode mode)
```

Description

Returns the supported trigger modes.

Parameter

- `mode`: Returns the supported modes (see [Set\(\)](#))

3.1.1.35.9 Set**Class**

[uc480.Trigger](#)

Accessible

Camera.Trigger

Syntax

```
uc480.Trigger.Set(uc480.Defines.TriggerMode mode)
```

Description

Sets the trigger mode.

Parameter

- `mode`: Mode to be set:

<code>uc480.Defines.TriggerMode.Continuous</code>	Software trigger	Call of Capture() (continuous mode)
<code>uc480.Defines.TriggerMode.Hi_Lo</code>	Hardware trigger	Falling signal edge
<code>uc480.Defines.TriggerMode.Hi_Lo_Pre</code>	Enables the pre-trigger in the memory mode	Falling signal edge
<code>uc480.Defines.TriggerMode.Hi_Lo_Sync</code>	Freerun sync./hardware trigger (The freerun synchronization mode is currently only supported by the sensors of the UI-146x/UI-546x series.)	Falling signal edge
<code>uc480.Defines.TriggerMode.Lo_Hi</code>	Hardware trigger	Rising signal edge
<code>uc480.Defines.TriggerMode.Lo_Hi_Pre</code>	Enables the pre-trigger in the memory mode	Rising signal edge
<code>uc480.Defines.TriggerMode.Lo_Hi_Sync</code>	Freerun sync./hardware trigger (The freerun synchronization mode is currently only supported by the sensors of the UI-146x/UI-546x series.)	Rising signal edge
<code>uc480.Defines.TriggerMode.Off</code>	Off	-
<code>uc480.Defines.TriggerMode.Software</code>	Software trigger	Call of Freeze() (single frame mode)

3.1.1.36 Video

The `Video` class provides methods for saving captured images in an AVI file. Images are added automatically by [Capture\(\)](#). All settings have to be done before starting the video with [Start\(\)](#). To reduce the file size, the single frames are stored in the AVI container using an adjustable JPEG compression.

In contrast to the `uc480.Tools.Video` class the `uc480.Camera.Video` class can be used more automatically. It can only be used with the [DIB display mode](#).



Attention: After starting the video do not change the image memory otherwise it may lead to undefined behavior.

Methods

Method	Description
GetFrameCount	Reads out the number of frames that have been saved.
GetFrameRate	Returns the set frame rate for AVI capturing.
GetLostCount	Reads out the number of frames that have been discarded.
GetQuality	Returns the set quality for the frame compression.
GetRunning	Returns the current status of the video.
GetSize	Use this method to retrieve the size of the frame sequence saved to the current AVI file.
GetVideoID	Returns the ID of the current AVI file.
ResetCount	Resets the counters for saved and discarded images.
SetFrameRate	Sets the frame rate for AVI capturing.
SetQuality	Indicates the quality for the frames to be compressed.
Start	Starts the image capture thread.
Stop	Stops the image capture thread.

3.1.1.36.1 GetFrameCount

Class

[uc480.Video](#)

Accessible

Camera.Video

Syntax

`uc480.Video.GetFrameCount(out uint u32Count)`

Description

Reads out the number of frames that have been saved.

Parameter

- `u32Count`: Number of saved frames

3.1.1.36.2 GetFrameRate

Class

[uc480.Video](#)

Accessible

Camera.Video

Syntax

```
uc480.Video.GetFrameRate(out double f64FrameRate)
```

Description

Returns the set frame rate for AVI capturing. This value might not be equal to the frame rate set for the uc480 camera.

Parameter

- `s64FrameRate`: Returns the set frame rate.

3.1.1.36.3 GetLostCount

Class

[uc480.Video](#)

Accessible

Camera.Video

Syntax

```
uc480.Video.GetLostCount(out uint u32Count)
```

Description

Reads 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.

Parameter

- `u32Count`: Number of discarded frames

3.1.1.36.4 GetQuality

Class

[uc480.Video](#)

Accessible

Camera.Video

Syntax

```
uc480.Video.GetQuality(out int s32Quality)
```

Description

Returns the set quality for the frame compression. For compression, the system uses the JPEG algorithm.

Parameter

- `s32Quality`: Returns the image quality [1 = lowest ... 100 = highest]

3.1.1.36.5 GetRunning

Class

[uc480.Video](#)

Accessible

Camera.Video

Syntax

```
uc480.Video.GetRunning(out bool bEnable)
```

Description

Returns the current status of the video.

Parameter

- `bEnable`: Returns the current status (1 = running, 0 = not running).

3.1.1.36.6 GetSize

Class

[uc480.Video](#)

Accessible

Camera.Video

Syntax

```
uc480.Video.GetSize(out float f64Size)
```

Description

Use this method to retrieve the size of the frame sequence saved to the current AVI file.

Parameter

- `f64Size`: the size in kBytes

3.1.1.36.7 GetVideoID

Class

[uc480.Video](#)

Accessible

Camera.Video

Syntax

```
uc480.Video.GetVideoID(out int s32ID)
```

Description

Returns the ID of the current AVI file.

Parameter

- `s32ID`: video ID

3.1.1.36.8 ResetCount

Class

[uc480.Video](#)

Accessible

Camera.Video

Syntax

```
uc480.Video.ResetCount()
```

Description

Resets the counters for saved and discarded images.

Parameter

None

3.1.1.36.9 SetFrameRate**Class**

[uc480.Video](#)

Accessible

Camera.Video

Syntax

`uc480.Video.SetFrameRate(double s64FrameRate)`

Description

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 uc480 camera.

Parameter

- `s64FrameRate`: The frame rate to be set. Default = 25.0

3.1.1.36.10 SetQuality**Class**

[uc480.Video](#)

Accessible

Camera.Video

Syntax

`uc480.Video.SetQuality(int s32Quality)`

Description

Indicates the quality for the frames to be compressed. All settings have to done before starting the video. For compression, the system uses the JPEG algorithm.

Parameter

- `s32Quality`: Image quality [1 = lowest ... 100 = highest]

3.1.1.36.11 Start**Class**

[uc480.Video](#)

Accessible

Camera.Video

Syntax

`uc480.Video.Start(string strFileName)`

Description

Starts the image capture thread.

Parameter

- `strFileName`: AVI file name

3.1.1.36.12 Stop

Class

[uc480.Video](#)

Accessible

Camera.Video

Syntax

`uc480.Video.Stop()`

Description

Stops the image capture thread.

Parameter

None

3.1.1.37 Zoom

The `Zoom` class provides methods for controlling the zoom function of the lens or sensor (digital zoom) of some uc480 cameras.



This class is currently only supported by the USB uc480 XS camera family.

This class is currently only supported by the XS in the JPEG mode.

This UI-1008XS provides a digital zoom function with zoom factors from 1 to 16. The function always zooms in on the image center.

Methods

Method	Description
Get	Returns the current zoom factor.
GetDefault	Returns the default value for the digital zoom.
GetList	Returns a list of all zoom factors supported by the camera.
GetRange	Returns the range for the digital zoom.
GetSupported	Returns if the zoom function is supported by the camera.
Set	Sets a zoom factor.

3.1.1.37.1 Get

Class

[uc480.Zoom](#)

Accessible

Camera.Zoom

Syntax

```
uc480.Zoom.Get(out double f64Factor)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Returns the current zoom factor.

Parameter

- `f64Factor`: Returns the zoom factor.

3.1.1.37.2 GetDefault

Class

[uc480.Zoom](#)

Accessible

Camera.Zoom

Syntax

```
uc480.Zoom.GetDefault(out double value)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Returns the default value for the digital zoom.

Parameter

- `value`: Returns the zoom value.

3.1.1.37.3 GetList

Class

[uc480.Zoom](#)

Accessible

Camera.Zoom

Syntax

```
uc480.Zoom.GetList(out double[] List)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Returns a list of all zoom factors supported by the camera.

Parameter

- **List:** A user-defined array, which returns the supported zoom factors.

3.1.1.37.4 GetRange**Class**[uc480.Zoom](#)**Accessible**

Camera.Zoom

Syntax

```
uc480.Zoom.GetRange(out uc480.Types.Range<double> range)
uc480.Zoom.GetRange(out double min, out double max, out double inc)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Returns the range for the digital zoom.

Parameter

range	Minimum: Returns the minimum value. Maximum: Returns the maximum value. Increment: Returns the increment.
s32Min	Returns the minimum value.
s32Max	Returns the maximum value.
s32Inc	Returns the increment.

3.1.1.37.5 GetSupported**Class**[uc480.Zoom](#)**Accessible**

Camera.Zoom

Syntax

```
uc480.Zoom.GetSupported(out bool bSupported)
```



This class is currently only supported by the USB uc480 XS camera family.

Description

Returns if the zoom function is supported by the camera.

Parameter

bSupported	1 = Zoom function is supported
	0 = Zoom function is not supported

3.1.1.37.6 Set**Class**[uc480.Zoom](#)**Accessible**

Camera.Zoom

Syntax`uc480.Zoom.Set(double f64Factor)`

This class is currently only supported by the USB uc480 XS camera family.

Description

Sets a zoom factor.

Parameter

- `f64Factor`: Value to be set for the zoom factor. For example, 4.0 means zoom with factor 4.

3.1.1.38 Camera**Class**[uc480.Camera](#)**Accessible**

Camera

Syntax

```
uc480.Camera.Camera(int s32CamID)
uc480.Camera.Camera(int s32CamID, int hWnd)
uc480.Camera.Camera(int s32CamID, System.IntPtr hWnd)
uc480.Camera.Camera()
```

Description

Starts the driver and establishes the connection to the camera. When using Direct3D for image display, you can pass a handle to the output window. The method uses internally the [Init\(\)](#) method. Only when no parameters are passed, [Init\(\)](#) must be called.

Starter firmware for camera models GigE uc480 SE/RE/CP

When you are using these camera models, 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 [Init\(\)](#).

Updating the 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.



Camera models GigE uc480 SE/RE only: If you have initialized the camera with a driver version earlier than 3.40, you need to update the camera firmware manually using the IDS 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 [Init\(\)](#), `s32Cam` must previously have been set to the desired camera ID.

To initialize or select the next available camera without specifying a camera ID, `s32Cam` has to be preset with 0.

Thread safety

We recommend that you call the following methods exclusively from a single thread in order to avoid unpredictable behavior of the application.

- [Init\(\)](#)
- [Render\(\)](#)
- [Exit\(\)](#)

Parameter

<code>s32CamID</code>	<ul style="list-style-type: none"> • Camera ID <ul style="list-style-type: none"> 0: The first available camera will be initialized or selected. 1-254: The camera with the specified camera ID will be initialized or selected. • <code>s32Cam</code> <ul style="list-style-type: none"> <code> uc480.Defines.DeviceEnumeration.UseDeviceID:</code> The camera is opened using the device ID instead of the camera ID. • <code>s32Cam</code> <ul style="list-style-type: none"> <code> uc480.Defines.DeviceEnumeration.AllowStarterFwUpload:</code> 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 GigE uc480 SE/RE/CP cameras). To ensure backward compatibility of applications, always call <code>Init()</code> without the <code>uc480.Defines.DeviceEnumeration.AllowStarterFwUpload</code> parameter first. Only if an error occurs, call the method with this parameter set.
<code>hWnd</code>	Window where the Direct3D image will be displayed If <code>hWnd = NULL</code> , DIB mode will be used for image display.

3.1.1.39 Exit

Class

[uc480.Camera](#)

Accessible

Camera

Syntax

`uc480.Camera.Exit()`

Description

Disables the camera handle and releases the data structures and memory areas taken up by the uc480 camera. Image memory allocated using [Allocate\(\)](#) which has not been released yet is automatically released.



We recommend that you call the following methods only from a single thread in order to avoid unpredictable behavior of the application.

- [Init\(\)](#)
- [Render\(\)](#)
- [Exit\(\)](#)

Parameter

None

3.1.1.40 Init

Class

[uc480.Camera](#)

Accessible

Camera

Syntax

```
uc480.Camera.Init(int s32Cam, int s32hWnd)
uc480.Camera.Init(int s32Cam, System.IntPtr WindowHandle)
uc480.Camera.Init(int s32Cam)
uc480.Camera.Init()
```

Description

Starts the driver and establishes the connection to the camera. When using Direct3D for image display, you can pass a handle to the output window.

Starter firmware for camera models GigE uc480 SE/RE/CP

When you are using these camera models, 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 [Init\(\)](#).



Updating the 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.

Camera models GigE uc480 SE/RE only: If you have initialized the camera with a driver version earlier than 3.40, you need to update the camera firmware manually using the IDS 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 `Init()`, `s32Cam` must previously have been set to the desired camera ID.

To initialize or select the next available camera without specifying a camera ID, `s32Cam` has to be preset with 0.

**Thread safety**

We recommend that you call the following methods exclusively from a single thread in order to avoid unpredictable behavior of the application.

- `Init()`
- [Render\(\)](#)
- `Exit()`

Parameter

<code>s32Cam</code>	<ul style="list-style-type: none"> • Camera ID 0: The first available camera will be initialized or selected. 1-254: The camera with the specified camera ID will be initialized or selected. • <code>s32Cam uc480.Defines.DeviceEnumeration.UseDeviceID</code>: The camera is opened using the device ID instead of the camera ID. • <code>s32Cam uc480.Defines.DeviceEnumeration.AllowStarterFwUpload</code>: 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 GigE uc480 SE/RE/CP cameras). To ensure backward compatibility of applications, always call <code>Init()</code> without the <code>uc480.Defines.DeviceEnumeration.AllowStarterFwUpload</code> parameter first. Only if an error occurs, call the method with this parameter set.
<code>s32hWnd/WindowHandle</code>	Window where the Direct3D image will be displayed <code>If s32hWnd = NULL</code> , DIB mode will be used for image display.

3.2 uc480.Configuration

The `Configuration` class provides classes for setting various system-wide options:

- **Processor operating states (idle states/C-states)**

Modern processors have various operating states, so-called C-states, that are characterized by different power requirements. When the operating system selects an operating state with low power consumption (unequal C0), the USB transmission efficiency may be affected. Please refer to the [Application Note on our website](#) for more information on this topic.

Use `CPUIdleState` to disable these low power consumption operating states and improve USB transmission efficiency. The uc480 driver changes the current energy settings of the operating system when the first USB uc480 camera is opened. After the last USB uc480 camera is closed, the uc480 driver restores the original settings. The settings are valid for the current user only.

- **IPO thread**

The IPO thread (thread for performance optimization at image acquisition) is a thread

that runs with lowest priority. The IPO thread prevents the PC to use the power saving mode. If you already changed the power scheme of the operating system via the idle states, you must not use the IPO thread.

If the IPO thread is allowed, the uc480 API optimizes the performance if a USB uc480 camera is connected, more than one active CPU core is detected and the CPU supports C-states.

i Note: The IPO thread seems to increase the CPU load and prevents the PC to use the power saving mode. However, the IPO thread runs with lowest priority. If another thread needs the CPU, it gets the CPU immediately.

- **Activate OpenMP (Open Multi-Processing)**

OpenMP is a programming interface that supports distributed computing on multi-core processors. When you activate OpenMP support, intensive computing operations, such as the Bayer conversion, are distributed across several processor cores to accelerate execution. The use of [OpenMP](#), however, increases CPU load.

- **Load camera parameters during installation**

Use [InitialParameterSet](#) to indicate whether to apply the parameters stored on the camera automatically when opening the camera. You must first store the camera parameters on the camera using [Parameter](#) or via the corresponding function in the uc480 Cockpit. This setting applies to all connected cameras. If no parameters are stored on the camera, the standard parameters of this camera model are applied.

i **Note on settings for processor operating states**

The settings for processor operating states are available only on Windows operating systems.

The following classes exist:

- [BootBoost](#)
- [CPUIdleState](#)
- [InitialParameterSet](#)
- [Ipo](#)
- [OpenMP](#)

3.2.1 BootBoost

The `BootBoost` class opens the camera at the system start and allows a faster access to the camera in the running application.

i Note: The change of the camera ID only has an effect on the boot boost mode after reconnecting the camera.

If you add a camera to the boot boost list, the camera must not be open.

Methods

Method	Description
AddId	Adds an ID to the boot boost list.
ClearIdList	Deletes the boot boost list.
GetEnable	Returns if the boot boost mode is enabled or disabled.
GetIdList	Returns the boot boost list.
RemoveId	Deletes an ID from the boot boost list.
SetEnable	Enables/disables the boot boost mode.
SetIdList	Resets the complete boot boost list. Old entries are deleted.

3.2.1.1 AddId

Class

[uc480.Configuration.BootBoost](#)

Accessible

Configuration.BootBoost

Syntax

```
uc480.Configuration.BootBoost.AddId(uint u32Id)
```

Description

Adds an ID to the boot boost list.



Note: If you add a camera to the boot boost list, the camera must not be open.

Parameter

- `u32Id`: ID of the camera to be added to the boot boost list.

3.2.1.2 ClearIdList

Class

[uc480.Configuration.BootBoost](#)

Accessible

Configuration.BootBoost

Syntax

```
uc480.Configuration.BootBoost.ClearIdList()
```

Description

Deletes the boot boost list.

Parameter

None

3.2.1.3 GetEnable

Class

[uc480.Configuration.BootBoost](#)

Accessible

Configuration.BootBoost

Syntax

`uc480.Configuration.BootBoost.GetEnable(out bool bEnable)`

Description

Returns if the boot boost mode is enabled or disabled.

Parameter

bEnable	1 = Boot boost mode is enabled
	0 = Boot boost mode is disabled

3.2.1.4 GetIdList

Class

[uc480.Configuration.BootBoost](#)

Accessible

Configuration.BootBoost

Syntax

`uc480.Configuration.BootBoost.GetIdList(out int[] IdList)`
`uc480.Configuration.BootBoost.GetIdList(out System.Collections.Generic.List<int> IdList)`

Description

Returns the boot boost list.

Parameter

- `IdList`: Returns the list of camera IDs which are in boot boost mode.

3.2.1.5 Removeld

Class

[uc480.Configuration.BootBoost](#)

Accessible

Configuration.BootBoost

Syntax

`uc480.Configuration.BootBoost.RemoveId(uint u32Id)`

Description

Deletes an ID from the boot boost list.

Parameter

- `u32Id`: ID of the camera to be removed from the boot boost list.

3.2.1.6 SetEnable

Class

[uc480.Configuration.BootBoost](#)

Accessible

Configuration.BootBoost

Syntax

```
uc480.Configuration.BootBoost.SetEnable(bool bEnable)
```

Description

Enables/disables the boot boost mode.

Parameter

bEnable	1 = Enable boot boost
	0 = Disable boot boost

3.2.1.7 SetIdList

Class

[uc480.Configuration.BootBoost](#)

Accessible

Configuration.BootBoost

Syntax

```
uc480.Configuration.BootBoost.SetIdList(int[] IdList)
uc480.Configuration.BootBoost.SetIdList(System.Collections.Generic.List<int> IdList)
```

Description

Resets the complete boot boost list. Old entries are deleted.

Parameter

- `IdList`: List of camera IDs for boot boost mode

3.2.2 CPUIidleState

The `CPUIidleState` class provides methods for setting the processor operating states.

Methods

Method	Description
<u>GetDisableOnOpen</u>	Returns the processor operating states for power supply unit operation when a uc480 camera is opened.
<u>GetEnable</u>	Returns the state of processor operating states for power supply unit operation and battery operation.
<u>GetSupported</u>	Returns if function parameters for setting the processor operating states are supported.
<u>SetDisableOnOpen</u>	Disables processor operating states for power supply unit operation when a uc480 camera is opened.

3.2.2.1 GetDisableOnOpen

Class

[uc480.Configuration.CPUIidleState](#)

Accessible

Configuration.CPUIidleState

Syntax

```
uc480.Configuration.CPUIidleState.GetDisableOnOpen(out uc480.Defines.IdleState state)
```

Description

Returns the processor operating states for power supply unit operation when a uc480 camera is opened.

Parameter

state	uc480.Defines.IdleState.AC: Recover processor operating states for power supply unit operation
	uc480.Defines.IdleState.DC: Recover processor operating states for battery operation
	uc480.Defines.IdleState.None

3.2.2.2 GetEnable

Class

[uc480.Configuration.CPUIidleState](#)

Accessible

Configuration.CPUIidleState

Syntax

```
uc480.Configuration.CPUIidleState.GetEnable(out uc480.Defines.IdleState state)
```

Description

Returns the state of processor operating states for power supply unit operation and battery operation.

Parameter

state	uc480.Defines.IdleState.AC: Set/recover processor operating states for power supply unit operation
	uc480.Defines.IdleState.DC: Set/recover processor operating states for battery operation
	uc480.Defines.IdleState.None

3.2.2.3 GetSupported

Class

[uc480.Configuration.CPUIidleState](#)

Accessible

Configuration.CPUIidleState

Syntax

```
uc480.Configuration.CPUIidleState.GetSupported(out bool supported)
```

Description

Returns if function parameters for setting the processor operating states are supported.

Parameter

supported	1 = Setting the processor operating states is supported
	0 = Setting the processor operating states is not supported

3.2.2.4 SetDisableOnOpen

Class

[uc480.Configuration.CPUIidleState](#)

Accessible

Configuration.CPUIidleState

Syntax

```
uc480.Configuration.CPUIidleState.SetDisableOnOpen(uc480.Defines.IdleState state)
```

Description

Disables processor operating states for power supply unit operation when a uc480 camera is opened.

Parameter

state	uc480.Defines.IdleState.AC: Set/recover processor operating states for power supply unit operation
	uc480.Defines.IdleState.DC: Set/recover processor operating states for battery operation
	uc480.Defines.IdleState.None

3.2.3 InitialParameterset

The `InitialParameterset` class provides methods for initializing the parameter set. Within these methods a camera parameter set can be loaded during initialization.

Methods

Method	Description
<u>GetEnable</u>	Returns the state of the load of camera parameters during initialization.
<u>GetSupported</u>	Returns if function parameters to load camera parameters during initialization are supported.
<u>SetEnable</u>	Enables/disables the load of camera parameters during initialization.

3.2.3.1 GetEnable**Class**

[uc480.Configuration.InitialParameterSet](#)

Accessible

Configuration.InitialParameterset

Syntax

```
uc480.Configuration.InitialParameterset.GetEnable(out bool bEnable)
```

Description

Returns the state of the load of camera parameters during initialization.

Parameter

bEnable	1 = Loading of a parameter set is enabled. 0 = Loading of a parameter set is disabled.
---------	---

3.2.3.2 GetSupported**Class**

[uc480.Configuration.InitialParameterSet](#)

Accessible

Configuration.InitialParameterset

Syntax

```
uc480.Configuration.InitialParameterset.GetSupported(out bool b32Supported)
```

Description

Returns if function parameters to load camera parameters during initialization are supported.

Parameter

b32Supported	1 = Loading camera parameters during initialization is supported 0 = Loading camera parameters during initialization is not supported
--------------	--

3.2.3.3 SetEnable

Class

[uc480.Configuration.InitialParameterSet](#)

Accessible

Configuration.InitialParametersSet

Syntax

```
uc480.Configuration.InitialParametersSet.SetEnable(bool bEnable)
```

Description

Enables/disables the load of camera parameters during initialization.

Parameter

bEnable	1 = Enable loading of a parameter set. 0 = Disable loading of a parameter set.
---------	---

3.2.4 Ipo

The `Ipo` class provides methods for enabling/disabling the IPO thread. The IPO thread is a thread for performance optimization at image acquisition and runs with lowest priority. The IPO thread prevents the PC to use the power saving mode. If you already changed the power scheme of the operating system via the idle states, you must not use the IPO thread.

If the IPO thread is allowed, the uc480 API optimizes the performance if a USB uc480 camera is connected, more than one active CPU core is detected and the CPU supports C-states.



Note: The IPO thread seems to increase the CPU load and prevents the PC to use the power saving mode. However, the IPO thread runs with lowest priority. If another thread needs the CPU, it gets the CPU immediately.

Methods

Method	Description
GetEnable	Returns if the IPO thread is enabled.
GetSupported	Returns if setting the IPO thread is supported.
SetEnable	Disables/enables the IPO thread.

3.2.4.1 GetEnable

Class

[uc480.Configuration.Ipo](#)

Accessible

Configuration.Ipo

Syntax

```
uc480.Configuration.Ipo.GetEnable(out bool enable)
```

Description

Returns if the IPO thread (thread for performance optimization at image acquisition) is enabled.

Parameter

enable	1 = IPO thread is enabled. 0 = IPO thread is disabled.
--------	---

3.2.4.2 GetSupported

Class

[uc480.Configuration.Ipo](#)

Accessible

Configuration.Ipo

Syntax

```
uc480.Configuration.Ipo.GetSupported(out bool supported)
```

Description

Returns if setting the IPO thread (thread for performance optimization at image acquisition) is supported.

Parameter

supported	1 = is supported 0 = is not supported
-----------	--

3.2.4.3 SetEnable

Class

[uc480.Configuration.Ipo](#)

Accessible

Configuration.Ipo

Syntax

```
uc480.Configuration.Ipo.SetEnable(out bool enable)
```

Description

Disables/enables the IPO thread (thread for performance optimization at image acquisition).

Parameter

enable	1 = Enable IPO thread
	0 = Disable IPO thread

3.2.5 OpenMP

The `OpenMP` class provides methods for setting the OpenMP support.

Methods

Method	Description
GetEnable	Returns if OpenMP support is enabled.
GetEnableDefault	Returns the default setting for OpenMP support.
GetSupported	Returns if function parameters to configure OpenMP are supported.
SetEnable	Enables/disables OpenMP support.

3.2.5.1 GetEnable

Class

[uc480.Configuration.OpenMP](#)

Accessible

Configuration.OpenMP

Syntax

```
uc480.Configuration.OpenMP.GetEnable(out bool bEnable)
```

Description

Returns if OpenMP support is enabled.

Parameter

bEnable	1 = OpenMP support is enabled.
	0 = OpenMP support is disabled.

3.2.5.2 GetEnableDefault

Class

[uc480.Configuration.OpenMP](#)

Accessible

Configuration.OpenMP

Syntax

```
uc480.Configuration.OpenMP.GetEnableDefault(out bool bEnable)
```

Description

Returns the default setting for OpenMP support.

Parameter

bEnable	1 = OpenMP support enabled.
	0 = OpenMP support disabled.

3.2.5.3 GetSupported**Class**[uc480.Configuration.OpenMP](#)**Accessible**

Configuration.OpenMP

Syntax

```
uc480.Configuration.OpenMP.GetSupported(out bool b32Supported)
```

Description

Returns if function parameters to configure OpenMP are supported.

Parameter

b32Supported	1 = Configuring OpenMP is supported
	0 = Configuring OpenMP is not supported

3.2.5.4 SetEnable**Class**[uc480.Configuration.OpenMP](#)**Accessible**

Configuration.OpenMP

Syntax

```
uc480.Configuration.OpenMP.SetEnable(bool bEnable)
```

Description

Enables/disables OpenMP support.

Parameter

bEnable	1 = Enable OpenMP support.
	0 = Disable OpenMP support.

3.3 uc480.Configuration.GigE

The `Configuration.GigE` class provides classes for configuring the IP addresses of GigE uc480 cameras or uploading the camera firmware. The following classes exist:

- [AutoConfigIp](#)
- [PacketFilter](#)
- [PersistentIp](#)
- [StarterFirmware](#)

3.3.1 AutoConfigIp

The `AutoConfigIp` class provides methods for configuring the automatic IP of a GigE uc480 camera. You can configure connected GigE uc480 cameras to automatically obtain the IP address from the network adapter. To use this feature, you need to define a suitable IP address range for the network adapter. When pairing a GigE uc480 camera with the IP address 0.0.0.0, the uc480 driver automatically assigns a free IP address to the camera.



This class is only supported by cameras of the GigE uc480 series.
The IP configuration can also be changed using the IDS Camera Manager.



Note on using GigE uc480 cameras on a DHCP network

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 uc480 driver is outside the DHCP range.

Methods

Method	Description
Get	Returns the IP address range for automatic IP configuration.
GetSupported	Returns if the setting of a automatic IP is supported.
Set	Sets the IP address range for automatic IP configuration.

3.3.1.1 Get

Class

[uc480.Configuration.GigE.AutoConfigIp](#)

Accessible

Configuration.GigE.AutoConfigIp

Syntax

```
uc480.Configuration.GigE.AutoConfigIp.Get(int s32DeviceId,
                                         out uc480.Types.ETH.AutoConfigIPSetup EthIpSutoConfig)
uc480.Configuration.GigE.AutoConfigIp.Get(uc480.Types.ETH.MacAddr MacAddr,
                                         out uc480.Types.ETH.AutoConfigIPSetup EthIpAutoConfig)
```

Description

Returns the IP address range for automatic IP configuration.

Parameter

int	s32DeviceId	Device ID of the network adapter
uc480.Types.ETH.MacAddr	MacAddr	MAC address of the network adapter
uc480.Types.ETH.AutoConfigIPSetup	EthIpAutoConfig	Returns the IP address range

3.3.1.2 GetSupported

Class

[uc480.Configuration.GigE.AutoConfigIp](#)

Accessible

Configuration.GigE.AutoConfigIp

Syntax

```
uc480.Configuration.GigE.AutoConfigIp.GetSupported(int s32DeviceId,
                                                out bool bAutoConfigIpSupported)
uc480.Configuration.GigE.AutoConfigIp.GetSupported(uc480.Types.ETH.MacAddr MacAddr,
                                                out bool bAutoConfigIpSupported)
```

Description

Returns if the setting an automatic IP is supported.

Parameter

int	s32DeviceId	Device ID of the network adapter
uc480.Types.ETH.MacAddr	MacAddr	MAC address of the network adapter
bool	bAutoConfigIpSupported	1 = Automatic IP is supported 0 = Automatic IP is not supported

3.3.1.3 Set

Class

[uc480.Configuration.GigE.AutoConfigIp](#)

Accessible

Configuration.GigE.AutoConfigIp

Syntax

```
uc480.Configuration.GigE.AutoConfigIp.Set(int s32DeviceId,
                                         uc480.Types.ETH.AutoConfigIPSetup EthIpAutoConfig)
uc480.Configuration.GigE.AutoConfigIp.Set(uc480.Types.ETH.MacAddr MacAddr,
                                         uc480.Types.ETH.AutoConfigIPSetup EthIpAutoConfig)
```

Description

Sets the IP address range for automatic IP configuration.

Parameter

int	s32DeviceId	Device ID of the network adapter
uc480.Types.ETH.MacAddr	Mac	MAC address of the network adapter
uc480.Types.ETH.AutoConfigIPSetup	EthIpAutoConfig	IP address range

3.3.2 PacketFilter

The `PacketFilter` class provides a method for setting the packet filter for a network adapter.



This class is only supported by the cameras of the GigE uc480 series.



Only incoming packets are filtered.

Regardless of this setting, ICMP (Ping) and ARP packets are always forwarded to the operating system.

Methods

Method	Description
Set	Sets the packet filter for a network adapter.

3.3.2.1 Set

Class

[uc480.Configuration.GigE.PacketFilter](#)

Accessible

Configuration.GigE.PacketFilter

Syntax

`uc480.Configuration.GigE.PacketFilter.Set(int s32AdapterId, uc480.Defines.PacketFilter filter)`

Description

Sets the packet filter for a network adapter.

Parameter

<code>s32AdapterId</code>	Internal adapter ID of the network adapter. It is returned by GetDeviceInfo() .
<code>filter</code>	<p><code>uc480.Defines.PacketFilter</code>: Packet filter to be set</p> <ul style="list-style-type: none"> • <code>uc480.Defines.PacketFilter.PassAll</code>: Forward all packets to the operating system. • <code>uc480.Defines.PacketFilter.BlockUEGET</code>: Block GigE uc480 data packets directed to the operating system (recommended). • <code>uc480.Defines.PacketFilter.BlockAll</code>: Block all packets directed to the operating system.

3.3.3 PersistentIp

The `PersistentIp` class provides methods for configuring the persistent IP of a GigE uc480 camera. If you set a persistent IP address, the address is permanently stored in camera memory. The persistent IP address is retained even after the camera is disconnected from the power supply. When connecting the camera to a different PC, make sure the persistent IP address is valid on that PC, as well. If you set the persistent IP address to 0.0.0.0, the camera is configured for automatic assignment of the IP address (see below).

**Note on setting persistent IP addresses**

Persistent IP addresses can only be set for cameras that have not been initialized.



This class is only supported by cameras of the GigE uc480 series.

The IP configuration can also be changed using the IDS Camera Manager.

**Note on using GigE uc480 cameras on a DHCP network**

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 uc480 driver is outside the DHCP range.

Methods

Method	Description
Get	Returns the persistent IP address and the subnet mask of the camera.
GetSupported	Returns if the setting of a persistent IP is supported.
Set	Sets the persistent IP address and the subnet mask of the camera.

3.3.3.1 Get

Class

[uc480.Configuration.GigE.PersistentIp](#)

Accessible

Configuration.GigE.PersistentIp

Syntax

```
uc480.Configuration.GigE.PersistentIp.Get(uc480.Types.ETH.MacAddr MacAddr,
                                         out uc480.Types.ETH.IpConfiguration EthIpConfig)
uc480.Configuration.GigE.PersistentIp.Get(int s32DeviceId,
                                         out uc480.Types.ETH.IpConfiguration EthIpConfig)
```

Description

Returns the persistent IP address and the subnet mask of the camera.

Parameter

int	s32DeviceId	Device ID of the camera
uc480.Types.ETH.MacAddr	MacAddr	MAC address of the camera
uc480.Types.ETH.IpConfiguration	EthIpConfig	Returns the persistent IP address and the subnet mask.

3.3.3.2 GetSupported

Class

[uc480.Configuration.GigE.PersistentIp](#)

Accessible

Configuration.GigE.PersistentIp

Syntax

```
uc480.Configuration.GigE.PersistentIp.GetSupported(uc480.Types.ETH.MacAddr MacAddr, out bool bPersistentIp)
uc480.Configuration.GigE.PersistentIp.GetSupported(int s32DeviceId, out bool bPersistentIpSupported)
```

Description

Returns if the setting of a persistent IP is supported.

Parameter

uc480.Types.ETH.MacAddr	MacAddr	MAC address of the camera
int	s32DeviceId	Camera ID
bool	bPersistentIpSupported	1 = Persistent IP is supported 0 = Persistent IP is not supported

3.3.3.3 Set

Class

[uc480.Configuration.GigE.PersistentIp](#)

Accessible

Configuration.GigE.PersistentIp

Syntax

```
uc480.Configuration.GigE.PersistentIp.Set(int s32DeviceId,
                                         uc480.Types.ETH.IpConfiguration EthIpConfig)
uc480.Configuration.GigE.PersistentIp.Set(uc480.Types.ETH.MacAddr MacAddr,
                                         uc480.Types.ETH.IpConfiguration EthIpConfig)
```

Description

Sets the persistent IP address and the subnet mask of the camera.

Parameter

int	s32DeviceId	Device ID of the camera
uc480.Types.ETH.MacAddr	MacAddr	MAC address of the camera
uc480.Types.ETH.IpConfiguration	EthIpConfig	Persistent IP address and subnet mask

3.3.4 StarterFirmware

The StarterFirmware class provides methods for updating the starter firmware of a connected camera. This is also possible from within the IDS Camera Manager. You can also read out the estimated time it will take the uc480 driver to execute specific processes (e.g. update the camera firmware).



This class is only supported by the cameras of the GigE uc480 series.

GigE uc480 RE series cameras use the same firmware as GigE uc480 SE series cameras. For GigE uc480 RE cameras please use the GigE uc480 SE method.



The starter firmware determines the start-up behaviour of the GigE uc480. 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](#).

Methods

Method	Description
GetUploadDuration	Returns the estimated time for uploading the starter firmware to a GigE uc480 cameras. This is a generic parameter which does not require you to specify the camera type.
GetUploadDurationCP	Returns the estimated time for uploading the starter firmware to a GigE uc480 CP camera.
GetUploadDurationSE	Returns the estimated time for uploading the starter firmware to a GigE uc480 SE/RE camera.
Upload	Updates the starter firmware.

3.3.4.1 GetUploadDuration

Class

[uc480.Configuration.GigE.StarterFirmware](#)

Accessible

Configuration.GigE.StarteFirmware

Syntax

`uc480.Configuration.GigE.StarterFirmware.GetUploadDuration(int s32DeviceId, out int s32Time)`

Description

Returns the estimated time for uploading the starter firmware to a GigE uc480 cameras. This is a generic parameter which does not require you to specify the camera type.

Parameter

s32DeviceId	Device ID of the camera
s32Time	Returns the estimated time in ms

3.3.4.2 GetUploadDurationCP

Class

[uc480.Configuration.GigE.StarterFirmware](#)

Accessible

Configuration.GigE.StarteFirmware

Syntax

`uc480.Configuration.GigE.StarterFirmware.GetUploadDurationCP(int s32Id, out int s32Time)`

Description

Returns the estimated time for uploading the starter firmware to a GigE uc480 CP camera.

Parameter

s32Id	Device ID of the camera
s32Time	Returns the estimated time in ms

3.3.4.3 GetUploadDurationSE

Class

[uc480.Configuration.GigE.StarterFirmware](#)

Accessible

Configuration.GigE.StarteFirmware

Syntax

```
uc480.Configuration.GigE.StarterFirmware.GetUploadDurationSE(int s32Id, out int s32Time)
```

Description

Returns the estimated time for uploading the starter firmware to a GigE uc480 SE/RE camera.

Parameter

s32Id	Device ID of the camera
s32Time	Returns the estimated time in ms

3.3.4.4 Upload

Class

[uc480.Configuration.GigE.StarterFirmware](#)

Accessible

Configuration.GigE.StarteFirmware

Syntax

```
uc480.Configuration.GigE.StarterFirmware.Upload(int s32DeviceId, string strFilepath)
```

Description

Updates the starter firmware of a connected camera. This is also possible from within the IDS Camera Manager.

Parameter

s32DeviceID	Device ID of the camera to be updated.
strFilepath	Contains the full file path.

3.4 uc480.Info

The `Info` class provides classes for querying camera and OS information. The following classes exist:

- [Camera](#)
- [System](#)

3.4.1 Camera

The `Camera` class provides methods for querying information about the uc480 API version or the OS version.

Methods

Method	Description
GetCameraList	Returns information about the connected cameras.
GetDeviceInfo	Returns information about connected uc480 cameras.
GetNumberOfDevices	Returns the number of uc480 cameras connected to the PC.

Events

The `Camera` class also provides special camera events:

Event	Description
<code>uc480.Info.Camera.EventDeviceRemoved</code>	A camera initialized with Init() was disconnected.
<code>uc480.Info.Camera.EventNewDevice</code>	A new camera was connected. This is independent of the device handle.

3.4.1.1 GetCameraList

Class

[uc480.Info.Camera](#)

Accessible

Info.Camera

Syntax

```
uc480.Info.Camera.GetCameraList(out uc480.Types.CameraInformation[] CameraList)
```

Description

Returns 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.

The serial number or model name should not be used to find a specific camera (e.g. in order to control this specific camera). If you use the serial number, the software may not find the serial number after exchanging the camera. The model name can be changed when updating the camera driver.

Instead, we recommend identifying a camera by a fixed camera ID, the camera type or by the sensor ID. The advantage of the camera ID is that you can set it manually. That means if you exchange a camera, you can set the same camera ID for the new camera.



Parameter

- CameraList: Returns [uc480.Types.CameraInformation](#)

3.4.1.2 GetDeviceInfo

Class

[uc480.Info.Camera](#)

Accessible

Info.Camera

Syntax

```
uc480.Info.Camera.GetDeviceInfo(int s32DeviceID, out uc480.Types.DeviceInformation Info)
uc480.Info.Camera.GetDeviceInfo(int s32DeviceID, out uc480.Types.ETH.DeviceInformation EthInfo)
```

Description

Returns information about connected uc480 cameras.

Parameter

s32DeviceID	Device ID
Info	Returns uc480.Types.DeviceInformation
EthInfo	Returns uc480.Types.ETH.DeviceInformation

3.4.1.3 GetNumberOfDevices

Class

[uc480.Info.Camera](#)

Accessible

Info.Camera

Syntax

```
uc480.Info.Camera.GetNumberOfDevices(out int s32Value)
```

Description

Returns the number of uc480 cameras connected to the PC.

Parameter

- s32Value: Returns number of devices.

3.4.2 System

The `System` class provides methods for querying information about the uc480 API version or the OS version.

Methods

Method	Description
GetNetVersion	Returns the version of the uc480DotNet.dll.
GetOsVersion	Returns the version of the API.

3.4.2.1 GetNetVersion

Class

[uc480.Info.System](#)

Accessible

Info.System

Syntax

```
uc480.Info.System.GetNetVersion(out System.Version version)
uc480.Info.System.GetNetVersion(out int s32Major, out int s32Minor, out int s32Build)
```

Description

Returns the version of the uc480DotNet.dll.

Parameter

version	Returns the complete version number
s32Major	Returns the major version
s32Minor	Returns the minor version
s32Build	Returns the build version

3.4.2.2 GetApiVersion

Class

[uc480.Info.System](#)

Accessible

Info.System

Syntax

```
uc480.Info.System.GetApiVersion(out System.Version version)
uc480.Info.System.GetApiVersion(out int s32Version)
uc480.Info.System.GetApiVersion(out int s32Major, out int s32Minor, out int s32Build)
```

Description

Returns the version of the API.

Parameter

version/s32Version	Returns the complete version number
s32Major	Returns the major version
s32Minor	Returns the minor version
s32Build	Returns the build version

3.5 uc480.Tools

The `Tools` class provides a class for capturing videos. The following classes exist:

- [Video](#)

3.5.1 Video

The `Video` class provides methods for capturing videos. In contrast to the `uc480.Camera.Video` class the `uc480.Tools.Video` class requires more manually programming, but can be used in all [display modes](#). It is also possible to extract single frames from the AVI file.



Attention: After starting the video do not change the image memory otherwise it may lead to undefined behavior.

Methods

Method	Description
AddFrame	Adds a new frame to an AVI sequence.
Close	Closes an AVI file which was opened using open() .
Exit	Exits an instance of the uc480 AVI interface which was initialized by Init() .
GetFrameCount	Reads out the number of frames that have been saved.
GetLostCount	Reads out the number of frames that have been discarded.
GetSize	Use this method to retrieve the size of the frame sequence saved to the current AVI file.
Init	Initializes an instance of the uc480 AVI interface.
Open	Opens a new or existing AVI file.
ResetCounter	Resets the counters for saved and discarded images.
SetFramerate	Sets the frame rate for AVI capturing.
SetImageSize	Sets the size and position of the area of interest which will be saved to the AVI file. Additionally this method specifies the input color format of the frames.
SetQuality	Indicates the quality for the frames to be compressed.
Start	Starts the image capture thread.
Stop	Stops the image capture thread.

3.5.1.1 AddFrame

Class

`uc480.Tools.Video`

Accessible

`Tools.Video`

Syntax

```
uc480.Tools.Video.AddFrame(int s32VideoID, System.IntPtr intPtr)
```

Description

Adds a new frame to an AVI sequence.

Parameter

s32VideoID	Video ID set by Init() .
intPtr	Pointer to the memory containing the image.

3.5.1.2 Close

Class

[uc480.Tools.Video](#)

Accessible

Tools.Video

Syntax

```
uc480.Tools.Video.Close(int s32VideoID)
```

Description

Closes an AVI file which was opened using [Open\(\)](#).

Parameter

- s32VideoID: Video ID set by [Init\(\)](#).

3.5.1.3 Exit

Class

[uc480.Tools.Video](#)

Accessible

Tools.Video

Syntax

```
uc480.Tools.Video.Exit(int videoID)
```

Description

Exits an instance of the uc480 AVI interface which was initialized by [Init\(\)](#).

Parameter

- videoID: Video Id to be exited

3.5.1.4 GetFrameCount

Class

[uc480.Tools.Video](#)

Accessible

Tools.Video

Syntax

```
uc480.Tools.Video.GetFrameCount(int s32VideoID, out uint u32Count)
```

Description

Reads out the number of frames that have been saved.

Parameter

s32VideoID	Video ID set by Init() .
u32Count	Number of saved frames

3.5.1.5 GetLostCount

Class

[uc480.Tools.Video](#)

Accessible

Tools.Video

Syntax

```
uc480.Tools.Video.GetLostCount(int s32VideoID, out uint u32Count)
```

Description

Reads 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.

Parameter

s32VideoID	Video ID set by Init() .
u32Count	Number of discarded frames

3.5.1.6 GetSize

Class

[uc480.Tools.Video](#)

Accessible

Tools.Video

Syntax

```
uc480.Tools.Video.GetSize(int s32VideoID, out float f64Size)
```

Description

Use this method to retrieve the size of the frame sequence saved to the current AVI file.

Parameter

s32VideoID	Video ID set by Init() .
f64Size	Size in kBytes

3.5.1.7 Init

Class

[uc480.Tools.Video](#)

Accessible

Tools.Video

Syntax

```
uc480.Tools.Video.Init(uc480.Camera camera, out int s32VideoID)
```

Description

Initializes an instance of the uc480 AVI interface. Multiple instances can be created simultaneously.

Parameter

camera	uc480 camera
s32VideoID	Returns the Video ID

3.5.1.8 Open

Class

[uc480.Tools.Video](#)

Accessible

Tools.Video

Syntax

```
uc480.Tools.Video.Open(int s32VideoID, string strFileName)
```

Description

Opens a new or existing AVI file.

Parameter

s32VideoID	Video ID set by Init() .
strFileName	Video file to be opened.

3.5.1.9 ResetCounter

Class

[uc480.Tools.Video](#)

Accessible

Tools.Video

Syntax

```
uc480.Tools.Video.ResetCounter(int s32VideoID)
```

Description

Resets the counters for saved and discarded images.

Parameter

- s32VideoID: Video ID set by [Init\(\)](#).

3.5.1.10 SetFramerate

Class

[uc480.Tools.Video](#)

Accessible

Tools.Video

Syntax

```
uc480.Tools.Video.SetFramerate(int s32VideoID, double f64FrameRate)
```

Description

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 uc480 camera.

Parameter

s32VideoID	Video ID set by Init() .
f64FrameRate	The frame rate to be set. Default = 25.0

3.5.1.11 SetImageSize

Class

[uc480.Tools.Video](#)

Accessible

Tools.Video

Syntax

```
uc480.Tools.Video.SetImageSize(int s32VideoID, uc480.Defines.ColorMode colorMode,
                               int s32Width, int s32Height)
```

Description

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 method specifies the input color format of the frames. You define these settings only once for the entire video.



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.



When an area of interest is used, the width and height of the AOI must be at least 16 pixel. The AOI width must be a multiple of 8.

Parameter

s32VideoID	Video ID set by Init() .
colorMode	Color mode for which the converter is to be returned (see GetSupported()).
s32Width	Width of the entire frame or of the area of interest.
s32Height	Height of the entire frame or of the area of interest.

3.5.1.12 SetQuality**Class**[uc480.Tools.Video](#)**Accessible**

Tools.Video

Syntax`uc480.Tools.Video.SetQuality(int s32VideoID, int s32Quality)`**Description**

Indicates the quality for the frames to be compressed. All settings have to done before starting the video. For compression, the system uses the JPEG algorithm.

Parameter

s32VideoID	Video ID set by Init() .
s32Quality	Image quality [1 = lowest ... 100 = highest]

3.5.1.13 Start**Class**[uc480.Tools.Video](#)**Accessible**

Tools.Video

Syntax`uc480.Tools.Video.Start(int s32VideoID)`**Description**

Starts the image capture thread.

Parameter

- s32VideoID: Video ID set by [Init\(\)](#).

3.5.1.14 Stop**Class**[uc480.Tools.Video](#)**Accessible**

Tools.Video

Syntax

```
uc480.Tools.Video.Stop(int s32VideoID)
```

Description

Stops the image capture thread. Subsequent calls of [AddFrame\(\)](#) will be ignored.

Parameter

- s32VideoID: Video ID set by [Init\(\)](#).

3.6 uc480.Types

- [AoiSequenceParameter](#)
- [CameraInfo](#)
- [CameraInformation](#)
- [CaptureStatus](#)
- [CaptureStatus.CaptureStatusApi](#)
- [CaptureStatus.CaptureStatusDriver](#)
- [CaptureStatus.CaptureStatusEth](#)
- [CaptureStatus.CaptureStatusUsb](#)
- [ConversionParameter](#)
- [DeviceInfoControl](#)
- [DeviceInfoHeartbeat](#)
- [DeviceInfo](#)
- [FaceDetectionInformation](#)
- [ImageFormatInfo](#)
- [ImageInfo](#)
- [KneePointInformation](#)
- [LutState](#)
- [LutSupportInfo](#)
- [MultiIntegrationCycles](#)
- [MultiIntegrationScope](#)
- [SensorInfo](#)
- [SensorScalerInformation](#)
- [TimestampConfiguration](#)

3.6.1 AoiSequenceParameter

uc480.Types.AoiSequenceParameter

<code>uc480.Defines.BinningMode</code>	<code>BinningMode</code>	Binning mode (not supported in version 4.40)
<code>int</code>	<code>DetachImageParameter</code>	<ul style="list-style-type: none"> • 0 = every change of the exposure time and the master gain is copied from AOI 1 to the additional AOIs (default). As a change of AOI 1 also reset the exposure time, this change is also transferred to AOI 2, 3 and 4. • 1 = a change of exposure time, gain or position of AOI 1 does not affect the parameters of AOI 2, 3 and 4.
<code>double</code>	<code>Exposure</code>	Exposure
<code>int</code>	<code>Gain</code>	Gain
<code>int</code>	<code>Index</code>	Index of the AOI
<code>int</code>	<code>NumberOfCycleRepetitions</code>	Number of readout cycles
<code>uc480.Types.Point<int></code>	<code>Position</code>	X and Y position of the AOI
<code>double</code>	<code>ScalerFactor</code>	Scaling factor (not supported in version 4.40)
<code>uc480.Defines.SubsamplingMode</code>	<code>SubsamplingMode</code>	Sub-sampling mode (not supported in version 4.40)

Used in method

- [GetParams\(\)](#) (Sequences class)
- [SetParams\(\)](#)

3.6.2 CameraInfo

uc480.Types.CameraInfo

uc480.Defines.BoardType	BoardType	Camera type: <ul style="list-style-type: none">• uc480.Defines.BoardType.Usb_SE: USB uc480 SE• uc480.Defines.BoardType.Usb_ME: USB uc480 ME• uc480.Defines.BoardType.Usb_RE: USB uc480 RE• uc480.Defines.BoardType.Usb_LE: USB uc480 LE• uc480.Defines.BoardType.Usb_XS: USB uc480 XS• uc480.Defines.BoardType.Eth_HE: GigE uc480 HE• uc480.Defines.BoardType.Eth_SE: GigE uc480 SE• uc480.Defines.BoardType.Eth_CP: GigE uc480 CP• uc480.Defines.BoardType.Eth_RE: GigE uc480 RE
int	CameraID	Camera ID
System.DateTime	Date	System date of the final quality check (e.g. 01.08.2011 (DD.MM.YYYY))
string	ID	Manufacturer of the camera (e.g. IDS GmbH)
string	SerialNumber	Serial number of the camera
string	Version	For USB cameras, this value indicates the USB board hardware version (e.g. v2.10)

Used in method

- [GetCameraInfo\(\)](#) (**Information class**)

3.6.3 CameraInformation

uc480.Types.CameraInformation

long	CameraID	Customizable camera ID. This ID is stored in the camera and is persistent.
long	DeviceID	Internal device ID. This ID is generated by the driver depending on order of connection and camera type. The device ID is not persistent.
bool	InUse	1 = camera is being used. 0 = camera is not being used.
string	Model	Camera model
long	SensorID	Sensor ID, e.g. DEFINES.SENSOR.UI122X_M As the same sensor is used in different camera families, the sensor ID returns even DEFINES.SENSOR.UI122X_M if you use a GigE uc480 camera (UI-522x), as the first number describes only the camera interface.
string	SerialNumber	Serial number of the camera
long	Status	Information for the status of the camera

Used in method

- [GetCameraList\(\)](#) (Camera class)

3.6.4 CaptureStatus

uc480.Types.CaptureStatus

<u>uc480.Types.CaptureStatus.CaptureStatusApi</u>	Api	See uc480.Types.CaptureStatus.CaptureStatusApi
ulong	DevTimeout	The maximum allowable time for image capturing in the camera was exceeded. Possible cause/remedy: The selected timeout value is too low for image capture • Reduce the exposure time • Increase the timeout
<u>uc480.Types.CaptureStatus.CaptureStatusDriver</u>	Driver	See uc480.Types.CaptureStatus.CaptureStatusDriver
<u>uc480.Types.CaptureStatus.CaptureStatusEth</u>	Eth	See uc480.Types.CaptureStatus.CaptureStatusEth
ulong	Total	Returns the total number of errors occurred since the last reset.
<u>uc480.Types.CaptureStatus.CaptureStatusUsb</u>	Usb	See uc480.Types.CaptureStatus.CaptureStatusUsb

Used in method

- [GetCaptureStatus\(\)](#) (Information class)

3.6.5 CaptureStatus.CaptureStatusApi

uc480.Types.CaptureStatus.CaptureStatusApi

ulong	ConversionFailed	The current image could not be processed correctly. Possible cause: <ul style="list-style-type: none">• Internal error during internal processing of the image
ulong	ImageLocked	The destination buffers are locked and could not be written to. Possible cause/remedy: 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
ulong	NoDestMem	There is no destination memory for copying the finished image. Possible cause/remedy: 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

3.6.6 CaptureStatus.CaptureStatusDriver

uc480.Types.CaptureStatus.CaptureStatusDriver

ulong	DeviceNotReady	The camera is no longer available. It is not possible to access images that have already been transferred. Possible cause: <ul style="list-style-type: none">• The camera has been disconnected or closed.
ulong	OutOfBuffer	No free internal image memory is available to the driver. The image was discarded. Possible cause/remdy: The computer takes too long to process the images in the uc480 API (e.g. color 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, color correction, choose smaller filter mask for software color conversion)

3.6.7 CaptureStatus.CaptureStatusEth

uc480.Types.CaptureStatus.CaptureStatusEth

ulong	BufferOverrun	<p>The sensor transfers more data than the internal camera memory of the GigE uc480 can accommodate.</p> <p><input type="checkbox"/> Possible cause/remedy</p> <p>The selected data rate of the sensor is too high</p> <ul style="list-style-type: none"> • Reduce the pixel clock frequency • Reduce the frame rate • Reduce the image size
ulong	MissedImages	<p>Freerun mode: The GigE uc480 camera could neither process nor output an image captured by the sensor.</p> <p>Hardware trigger mode: The GigE uc480 camera received a hardware trigger signal which could not be processed because the sensor was still busy.</p> <p><input type="checkbox"/> Possible cause/remedy</p> <p>The camera's frame rate is too high or the bandwidth on the network is insufficient to transfer the image</p> <ul style="list-style-type: none"> • Reduce the frame rate • Increase the value for the receive descriptors in the network card settings

3.6.8 CaptureStatus.CaptureStatusUsb

uc480.Types.CaptureStatus.CaptureStatusUsb

ulong	TransferFailed	<p>The image was not transferred over the USB bus.</p> <p><input type="checkbox"/> Possible cause/remedy</p> <p>Not enough free bandwidth on the USB bus for transferring the image</p> <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
-------	----------------	--

3.6.9 ConversionParameter

uc480.Types.ConversionParameter

uc480.Defines.ColorCorrectionMode	DstColorCorrectionMode	Sets the color correction, see Correction()
int	DstEdgeEnhancement	Sets the edge enhancement, see EdgeEnhancement
int	DstGamma	Sets the gamma correction, see Software
int	DstMemID	Raw Bayer buffer which was created with Allocate()
uc480.Defines.ColorConvertMode	DstPixelConverter	Conversion mode for the target image; see Converter() for possible modes
uc480.Defines.ColorMode	DstPixelFormat	Color mode of the target image, see PixelFormat for possible modes
int	DstSaturationU	Sets the color saturation (saturation U), see Saturation
int	DstSaturationV	Sets the color saturation (saturation V), see Saturation
int	SrcMemID	Target buffer with the converted data which was created with Allocate()

Used in method

- [Convert\(\)](#) (Image class)

3.6.10 DeviceInfoControl

uc480.Types.DeviceInfoControl

int	DeviceID	Device ID of the camera
-----	----------	-------------------------

3.6.11 DeviceInfoHeartbeat

uc480.Types.DeviceInfoHeartbeat

int	Temperature	Camera temperature
-----	-------------	--------------------

3.6.12 DeviceInformation

uc480.Types.DeviceInformation

uc480.Types.DeviceInformation.DeviceInfoControl	DeviceInfoControl	Camera-related driver data
uc480.Types.DeviceInformation.DeviceInfoHeartbeat	DeviceInfoHeartbeat	Camera-related data retrieved from the camera (from the heartbeat telegram)

Used in method

- [GetDeviceInfo\(\)](#) (Camera class)
- [GetDeviceInfo\(\)](#) (Information class)

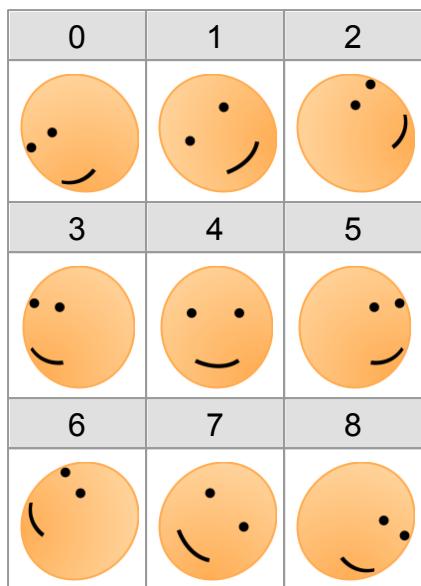
3.6.13 FaceDetectionInformation

uc480.Types.FaceDetectionInformation

int	FaceAngle	Orientation angle of the face: • 0/90/270: Angle in degrees (0° corresponds to 12 o'clock position) • -1: Undefined
uc480.Types.Point<int>	FacePosition	Face position: • X: X position of the face (center) • Y: Y position of the face (center)
int	FacePosture	Face posture (see table below)
uc480.Types.Size<int>	FaceSize	Face size • Height: Face height • Width: Face width
System.DateTime	TimestampSystem	Structure with timestamp information in PC system time format, see GetImageInfo() . The time resolution is approx. 15 ms.

Used in method

- [GetFaceList\(\)](#) (FaceDetection class)

Possible values for posture (viewing direction)

3.6.14 ImageFormatInfo

uc480.Types.ImageFormatInfo

uc480.Defines.BinningMode	BinningMode	Binning mode used (see GetSupported())
uc480.Defines.CaptureMode	CaptureMode	<p>Image capture modes supported for this format:</p> <ul style="list-style-type: none"> • uc480.Defines.CaptureMode.Single: Freerun mode, single frame (freerun snap) • uc480.Defines.CaptureMode.Freerun: Freerun mode, continuous (freerun live) • uc480.Defines.CaptureMode.TriggerSoftwareSingle: Software triggered mode, single frame • uc480.Defines.CaptureMode.TriggerSoftwareContinuous: Software triggered mode, continuous • uc480.Defines.CaptureMode.TriggerHardwareSingle: Hardware triggered mode, single frame • uc480.Defines.CaptureMode.TriggerHardwareContinuous: Hardware triggered mode, continuous
int	FormatID	Format ID of the specified image format (see " Image formats " below)
string	FormatName	Description of the format
uc480.Types.Point<int>	Position	Start points of the area of interest (X value, Y value)
double	SensorScalerFactor	Scaling factor used (only sensors that support scaling).
uc480.Types.Size<int>	Size	Height and Width of the area of interest
uc480.Defines.SubsamplingMode	SubsamplingMode	Sub-sampling mode used (see GetSupported())

Used in methods

- [GetList\(\)](#) (`ImageFormat` class)
- [set\(\)](#) (`ImageFormat` class)

Image formats of CMOS sensors

Format ID	Resolution	Name	Camera model												
			UI-149x	UI-148x	UI-158x	UI-146x	UI-155x	UI-164x	UI-154x	UI-124x	UI-125x	UI-122x	UI-336x	UI-337x	UI-112x
1	326 4x2 448	(8M)	X												
2	326 4x2 176	(8M 3:2)	X												
3	326 4x1 836	(8M 16:9)	X												
4	259 2x1 944	(5M)	X												
5	204 8x1 536	(3M)	X	X	X	X								X	
6	192 0x1 080	(Full HD 16:9)	X	X	X	X							X	X	
7	163 2x1 224	(2M)	X	X	X	X									
8	128 0x9 60	(1.2 M 4:3)	X	X	X	X	X	X	X	X	X	X			
9	128 0x7 20	(HD 16:9)	X	X	X	X	X	X	X	X	X	X			
11	960 x48 0	(WV GA 2:1)	X	X	X	X	X	X	X	X	X	X			
12	800 x48 0	(WV GA)	X	X	X	X	X	X	X	X	X	X			
13	640 x48 0	(VG A)	X	X	X	X	X	X	X	X	X	X			
14	640 x36 0	(VG A 16:9)	X	X	X	X	X	X	X	X	X	X			
15	400 x24 0	(WQ VG A)	X	X	X	X	X	X	X	X	X	X			
16	352 x28 8	(CIF)	X	X	X	X	X	X	X	X	X	X			
	288 x25	(CIF Port)													

Image formats of USB uc480 XS

Format ID	Resolution	Name	XS	UI-1008XS Live	UI-1008XS Snap
1	3264x2448	(8M)			X
2	3264x2176	(8M 3:2)			X
3	3264x1836	(8M 16:9)			X
4	2592x1944	(5M)	X		X
5	2048x1536	(3M)	X		X
6	1920x1080	(Full HD 16:9)	X		X
7	1632x1224	(2M)			X
8	1280x960	(1.2M 4:3)	X		X
9	1280x720	(HD 16:9)	X	X	X
11	960x480	(WVGA 2:1)		X	X
12	800x480	(WVGA)	X	X	X
13	640x480	(VGA)	X	X	X
14	640x360	(VGA 16:9)		X	X
15	400x240	(WQVGA)		X	X
16	352x288	(CIF)		X	X
17	288x352	(CIF Portrait)		X	X
18	320x240	(QVGA)		X	X
19	240x320	(QVGA Portrait)			
20	1600x1200	(UXGA)	X		X
21	3840x2748	(10M)			
22	1920x1080	(Full HD 16:9, HQ)			
23	2560x1920	(5M)			
24	768x576	(CCIR)			
25	1280x1024	(1.3M SXGA)			
26	2448x2048	(5M)			
27	1024x768	(XGA)			
28	1024x1024	(1M)			
29	800x600	(SVGA)			
30	1360x1024	(1.4M 4:3)			
31	640x480	(VGA HS)	X		

Image formats of CCD sensors

Format ID	Resolution	Name	Camera model								
			UI-214x	UI-221x	UI-222x	UI-223x	UI-224x	UI-225x	UI-228x	UI-234x	UI-241x
1	3264x2448	(8M)									
2	3264x2176	(8M 3:2)									
3	3264x1836	(8M 16:9)									
4	2592x1944	(5M)									
5	2048x1536	(3M)							X		
6	1920x1080	(Full HD 16:9)								X	
7	1632x1224	(2M)							X		
8	1280x960	(1.2M 4:3)	X					X	X	X	X
9	1280x720	(HD 16:9)	X					X	X	X	X
11	960x480	(WVG A 2:1)	X				X	X	X	X	X
12	800x480	(WVG A)	X				X	X	X	X	X
13	640x480	(VGA)	X	X	X	X	X	X	X	X	X
14	640x360	(VGA 16:9)	X	X	X	X					X
15	400x240	(WQV GA)	X	X	X	X					X
16	352x288	(CIF)	X	X	X	X					X
17	288x252	(CIF Portra it)	X	X	X	X					X
18	320x240	(QVG A)	X	X	X	X					X
19	240x200	(QVG A Portra it)	X	X	X	X					X
20	1600x1200	(UXG A)							X	X	
21	3840x2748	(10M)									
		(Full HD 1080x16:9)									

3.6.15 **ImageInfo**

uc480.Types.ImageInfo

int	AOICycle	Readout cycles (only AOI sequence mode of UI-124x/UI-524x)
-----	----------	--

int	AOIIndex	AOI index (only AOI sequence mode of UI-124x/UI-524x)
-----	----------	---

ulong	FrameNumber	Internal image number (not chronological) Note: Use <code>TimestampTick</code> to get the right image sequence.
-------	-------------	--

int	HostProcessTime	DIB mode only: Time in μ s which was used for image processing (time difference between raw image data arrival and the setting of the frame event). The value is an indication of the maximum possible frame rate.
-----	-----------------	--

int	ImageBuffersCount	Number of image buffers existing in the camera
-----	-------------------	--

int	ImageBuffersInUse	Number of image buffers in use in the camera
-----	-------------------	--

uc480.Types.SizeType<int>	ImageSize	Image height and width
---------------------------	-----------	------------------------

int	IoStatus	With GigE uc480 cameras: Returns the states of the digital I/Os at the time of image capture: <ul style="list-style-type: none">• Digital input (trigger): Pending signal• GPIO as input: Pending signal• GPIO as output: Set level GPIOs (programmable I/Os) are only available on GigE uc480 HE cameras. With all other cameras, <code>s32IoStatus</code> is empty.
-----	----------	--

Bit combination	State of digital input	State of GPIO 1	State of GPIO 2
000	0	0	0
001	0	0	1
010	0	1	0
011	0	1	1
100	1	0	0
101	1	0	1
110	1	1	0
111	1	1	1

System.DateTime	TimestampSystem	Structure with timestamp information in PC system time format
-----------------	-----------------	---

WORD	wYear	Timestamp year
WORD	wMonth	Timestamp month
WORD	wDay	Timestamp day
WORD	wHour	Timestamp hour
WORD	wMinute	Timestamp

Used in methods

- [GetImageInfo\(\)](#) (Information class)

3.6.16 KneePointInformation

uc480.Types.KneePointInformation

uc480.Types.Point<double>[]	DefaultKneePointList	Contains the default kneepoints (X value, Y value), see also Set() .
int	NumberOfSupportedKneepoints	Maximum number of supported knee points
int	NumberOfUsedKneepoints	Currently used number of knee points
uc480.Types.Range<double>	RangeValueX	Range of the X value of a knee point (minimum value, maximum value, increment)
uc480.Types.Range<double>	RangeValueY	Range of the Y value of a knee point (minimum value, maximum value, increment)

Used in method

- [GetInfo\(\)](#) (Kneepoints class)

3.6.17 LutState

uc480.Types.LutState

int	Bits	Used bits of the LUT
bool	Enabled	LUT is enabled
uc480.Defines.LutMode	Mode	LUT mode
uc480.Defines.LutState	State	LUT state

Used in method

- [GetState\(\)](#)
- [GetStateInfo\(\)](#)

3.6.18 LutSupportInfo

uc480.Types.LutSupportInfo

int	BitsHardware	Used bits of the hardware LUT
int	BitsSoftware	Used bits of the software LUT
int	ChannelsHardware	Supported channels for hardware LUT
int	ChannelsSoftware	Supported channels for software LUT
bool	LutHardware	Hardware LUT is supported
bool	LutSoftware	Software LUT is supported

Used in method

- [GetSupportedInfo\(\)](#)

3.6.19 MultilIntegrationCycles

uc480.Types.MultilIntegrationCycles

double	Integration_ms	Pulse duration in milliseconds
double	Pause_ms	Pause duration in milliseconds

Used in method

- [GetParams\(\)](#)
- [SetParams\(\)](#)

3.6.20 MultilIntegrationScope

uc480.Types.MultilIntegrationScope

uc480.Types.Range<double>	Cycle_ms	Cycle duration in milliseconds
double	CycleGranularity_ms	Cycle granularity in milliseconds
uc480.Types.Range<double>	Integration_ms	Pulse duration in milliseconds
double	IntegrationGranularity_ms	Pulse granularity in milliseconds
uint	MaxNumberOfCycles	Maximum number of cycles
uc480.Types.Range<double>	Pause_ms	Pause duration in milliseconds
double	PauseGranularity_ms	Pause granularity in milliseconds

Used in method

- [GetScope\(\)](#)

3.6.21 SensorInfo**uc480.Types.SensorInfo**

bool	BlueGain	Indicates whether the sensor provides analog blue channel gain
bool	GlobalShutter	Indicates whether the sensor has a global shutter. TRUE = global shutter FALSE = rolling shutter
bool	GreenGain	Indicates whether the sensor provides analog green channel gain
bool	MasterGain	Indicates whether the sensor provides analog master gain
uc480.Types.Size<int>	MaxSize	Returns the maximum image height and width
int	PixelSize	Returns the pixel size in μm (e.g. 465 is equivalent to 4.65 μm)
bool	RedGain	Indicates whether the sensor provides analog red channel gain
uc480.Defines.SensorColorMode	SensorColorMode	Returns the sensor color mode. <ul style="list-style-type: none"> • uc480.Defines.SensorColorMode.Bayer • uc480.Defines.SensorColorMode.Monochrome • uc480.Defines.SensorColorMode.CBYCRY (USB uc480 XS only) • uc480.Defines.SensorColorMode.Jpeg (XS only)
uc480.Defines.Sensor	SensorID	Returns the sensor type (e.g.: uc480.Defines.Sensor.UI1240_C).
string	SensorName	Returns the camera model (e.g.: UI224xLE-C).
BayerPixel	UpperLeftBayerPixel	Returns the color of the upper left pixel in the Bayern pattern: <ul style="list-style-type: none"> • uc480.Defines.BayerPixel.Red • uc480.Defines.BayerPixel.Green • uc480.Defines.BayerPixel.Blue

Used in method

- [GetSensorInfo\(\)](#) (Information class)

3.6.22 SensorScalerInformation

uc480.Types.SensorScalerInformation

uc480.Defines.SensorScalerMode	CurrentMode	Returns the current mode
double	FactorCurrent	Returns the current scaling factor
uc480.Types.Range<double>	FactorRange	Returns the range of the scaling factor (minimum value, maximum value, increment)
int	NumberOfSteps	Returns the number of steps for the scaling factor

Used in method

- [GetInfo\(\)](#) (SensorScaler class)

3.6.23 TimestampConfiguration

uc480.Types.TimestampConfiguration

uc480.Defines.IO.Edge	Edge	Signal on the rising/falling edge which is used for time stamp event (when uc480.Defines.IO.Pin.Trigger is used): <ul style="list-style-type: none"> uc480.Defines.IO.Edge.Falling: The falling edge signal fires the time stamp event. uc480.Defines.IO.Edge.Rising: The rising edge signal fires the time stamp event.
uc480.Defines.TimestampMode	Mode	Mode for the time stamp configuration: <ul style="list-style-type: none"> uc480.Defines.TimestampMode.Once: Resets once the time stamp of the camera to 0.
uc480.Defines.IO.Pin	Pin	Pin which is used for time stamp event: <ul style="list-style-type: none"> uc480.Defines.IO.Pin.None : No time stamp event. uc480.Defines.IO.Pin.Trigger: A signal on the trigger pin fires the time stamp event. uc480.Defines.IO.Pin.Gpio 0: A signal on the GPIO 1 fires the time stamp event. uc480.Defines.IO.Pin.Gpio 1: A signal on the GPIO 2 fires the time stamp event.

Used in method

- [Get\(\)](#) (Timestamp class)
- [set\(\)](#) (Timestamp class)

3.7 uc480.Types.AutoFeature

- [BrightStatus](#)
- [Information](#)
- [WhitebalanceChannelStatus](#)
- [WhitebalanceStatus](#)

3.7.1 BrightStatus

uc480.Types.AutoFeature.BrightStatus

int	Controller	Current parameter value
int	CtrlStatus	Current control status
int	Error	Current control deviation (error)
int	Value	Current average brightness of the image (actual value); the following rule applies independently of the image bit depth: 0 = black 255 = white

3.7.2 Information

uc480.Types.AutoFeature.Information

<u>uc480.Types.AutoFeature</u> <u>.BrightStatus</u>	BrightStatus	Status of automatic brightness control
uc480.Defines.Whitebalance.AntiFlickerMode	AAntiFlickerMode	<p>Returns a bit mask containing all supported anti flicker settings for automatic control:</p> <ul style="list-style-type: none"> • uc480.Defines.Whitebalance.AntiFlickerMode.Disable : Anti flicker mode is disabled. • uc480.Defines.Whitebalance.AntiFlickerMode.SensorAuto : The anti flicker mode is selected automatically (50 or 60 Hz). • uc480.Defines.Whitebalance.AntiFlickerMode.Sensor50Fixed : The anti flicker mode is set to a fixed value of 50 Hz. • uc480.Defines.Whitebalance.AntiFlickerMode.Sensor60Fixed : The anti flicker mode is set to a fixed value of 60 Hz.
uc480.Defines.Whitebalance.GainPhotomMode	AGainPhotomMode	<p>Returns a bit mask containing all supported photometry settings (fields of view) for auto exposure shutter:</p> <ul style="list-style-type: none"> • uc480.Defines.Whitebalance.GainPhotomMode.None : The entire field of view is used for metering. • uc480.Defines.Whitebalance.GainPhotomMode.CenterWeighted : The center portion of the field of view is weighted more heavily for metering.

		<p>ighted: Metering is based on the entire field of view, but gives greater emphasis to the center area of the image.</p> <ul style="list-style-type: none"> • uc480.Defines.Whitebalance.GainPhotomMode.CenterSpot: Only a small area in the image center is used for metering. • uc480.Defines.Whitebalance.GainPhotomMode.PhotomMode.Portrait: Metering is based on that part of the field of view that corresponds to the portrait format. • uc480.Defines.Whitebalance.GainPhotomMode.Landscape: Metering is based on that part of the field of view that corresponds to the landscape format.
uc480.Defines.Whitebalance.ShutterPhotomMode	AShutterPhotomMode	<p>Returns a bit mask containing all supported photometry settings (fields of view) for auto gain control:</p> <ul style="list-style-type: none"> • uc480.Defines.Whitebalance.ShutterPhotomMode.None: The entire field of view is used for metering. • uc480.Defines.Whitebalance.ShutterPhotomMode.CenterWeighted: Metering is based on the entire field of view, but gives greater emphasis to the center area of the image. • uc480.Defines.Whitebalance.ShutterPhotomMode.CenterSpot: Only a small area in the image center is used for metering. • uc480.Defines.Whitebalance.ShutterPhotomMode.PhotomMode.Portrait: Metering is based on that part of the field of view that corresponds to the portrait format. • uc480.Defines.Whitebalance.ShutterPhotomMode.Landscape: Metering is based on that part of the field of view that corresponds to the landscape format.

		landscape format.
uc480.Defines.AutoAbilityMode	AutoAbility	Auto-control ability
uc480.Defines.Whitebalance.WhiteBalanceMode	SensorWhitebalanceMode	<p>Returns a bit mask containing all supported settings for the sensor's auto white balance:</p> <ul style="list-style-type: none"> • uc480.Defines.Whitebalance.WhiteBalanceMode.Disable: Disables the sensor's auto white balance • uc480.Defines.Whitebalance.WhiteBalanceMode.Automatic: Sensor automatically determines auto white balance • uc480.Defines.Whitebalance.WhiteBalanceMode.AllPixel: Sensor automatically determines auto white balance using the Gray World algorithm. This algorithm assumes that the average color value in the scene is gray. • uc480.Defines.Whitebalance.WhiteBalanceMode.IncandescentLamp: Sensor sets auto white balance to incandescent light • uc480.Defines.Whitebalance.WhiteBalanceMode.FluorescentDL: Sensor sets auto white balance to fluorescent light (daylight type) • uc480.Defines.Whitebalance.WhiteBalanceMode.OutdoorClearSky: Sensor sets auto white balance to direct daylight • uc480.Defines.Whitebalance.WhiteBalanceMode.OutdoorCloudy: Sensor sets auto white balance to cloudy sky
uc480.Types.AutoFeature .WhitebalanceStatus	WhitebalanceStatus	Status of auto white balance

Used in method

- [GetInfo\(\)](#) (AutoFeatures class)

3.7.3 WhitebalanceChannelStatus

uc480.Types.AutoFeature.WhitebalanceChannelStatus

uc480.Defines.Whitebalance.ControllerStatusMode	Status	Current control status: • uc480.Defines.Whitebalance.ControllerStatusMode.Adjusting: Control is active. • uc480.Defines.Whitebalance.ControllerStatusMode.Finished: Control is completed. • uc480.Defines.Whitebalance.ControllerStatusMode.Disabled: Control is disabled.
int	Error	Current auto white balance error
int	Value	Current average gray-scale value (actual value)

3.7.4 WhitebalanceStatus

uc480.Types.AutoFeature.WhitebalanceStatus

int	Controller	Current white balance control
WhitebalanceChannelStatus	BlueChannel	See WhitebalanceChannelStatus
WhitebalanceChannelStatus	GreenChannel	See WhitebalanceChannelStatus
WhitebalanceChannelStatus	RedChannel	See WhitebalanceChannelStatus

3.8 uc480.Types.ETH

- [AdapterInformation](#)
- [AutoConfigIPSetup](#)
- [DeviceHeartbeat](#)
- [DeviceInfoControl](#)
- [DeviceInformation](#)
- [DriverInformation](#)
- [EthernetConfiguration](#)
- [IpConfiguration](#)
- [MacAddr](#)

3.8.1 AdapterInformation

uc480.Types.ETH.AdapterInformation

uint	AdapterID	Adapter ID
uc480.Types.ETH.EthernetConfiguration	EthernetConfiguration	ETH configuration of the adapter
bool	EnabledDHCP	Adapter's DHCP enabled flag
uc480.Types.ETH.AutoConfigIPSetup	AutoConfigIPSetup	
bool	ValidAutoConfigIpRange	The given range is valid when the begin and end are valid IP addresses in the subnet of the adapter
uint	CntDevicesKnown	Count of listed known devices
uint	CntDevicesPaired	Count of listed paired devices
uc480.Defines.PacketFilter	PacketFilter	<p>Setting for the incoming packets filter:</p> <ul style="list-style-type: none"> • uc480.Defines.PacketFilter.PassAll: Forward all packets to the operating system. • uc480.Defines.PacketFilter.BlockUEGET: Block GigE uc480 data packets directed to the operating system (recommended). • uc480.Defines.PacketFilter.BlockAll: Block all packets directed to the operating system.

3.8.2 AutoConfigIPSetup

uc480.Types.ETH.AutoConfigIPSetup

System.NET.IPEndPoint	AddressRangeBegin	Begin of IP address range
System.NET.IPEndPoint	AddressRangeEnd	End of IP address range

Used in methods

- [Get\(\)](#) (AutoConfigIp class)
- [Set\(\)](#) (AutoConfigIp class)

3.8.3 DeviceHeartbeat

uc480.Types.ETH.DeviceHeartbeat

string	SerialNumber	Camera's serial number
uc480.Defines.CameraType	deviceType	Device type/board type:

e		<ul style="list-style-type: none"> • uc480.Defines.CameraType.Eth_CP • uc480.Defines.CameraType.Eth_HE • uc480.Defines.CameraType.Eth_LEET • uc480.Defines.CameraType.Eth_RE • uc480.Defines.CameraType.Eth REP • uc480.Defines.CameraType.Eth_SE • uc480.Defines.CameraType.Usb3 CP • uc480.Defines.CameraType.Usb LE • uc480.Defines.CameraType.Usb ME • uc480.Defines.CameraType.Usb ML • uc480.Defines.CameraType.Usb RE • uc480.Defines.CameraType.Usb SE
int	CameraID	Camera ID
int	SensorID	Size of camera's image memory in MB
int	SizeImgMemMB	Size of camera's image memory in MB
uint	VersionStarterFirmware	Starter firmware version
uint	VersionRuntimeFirmware	Runtime firmware version
<u>uc480.Defines.EthDeviceStatus</u>	Status	Camera status flags, see table below.
uint	Temperature	Camera temperature
uint	LinkSpeedMB	Link speed in MB
<u>uc480.Types.ETH.MacAddr</u>	MacDevice	Camera's MAC address
uint	ComportOffset	Comport offset from 100, valid range from -99 to +156
<u>uc480.Types.ETH.IpConfiguration</u>	PersistentIpConfiguration	Persistent IP configuration
<u>uc480.Types.ETH.IpConfiguration</u>	CurrentIpConfiguration	Current IP configuration
<u>uc480.Types.ETH.MacAddr</u>	MacPairedHost	Paired host's MAC address
System.NET.IPAddress	IpPairedHost	Paired host's IP address
System.NET.IPAddress	IpRangeBegin	Begin of IP address range

System.NET.IPEndPoint	IpRangeEnd	End of IP address range
string	UserSpaceEEPROM	User space data (first 8 bytes)

uc480.Defines.EthDeviceStatus

Status	Description
Force100MBPS	The device is configured to 100 Mbsp.
InapplicableFirmwareRuntime	The runtime firmware is inapplicable.
InapplicableFirmwareStarter	The starter firmware is inapplicable.
InapplicableIpCurrent	The current IP of the device is inapplicable.
InapplicableIpPersistent	The persistent IP of the device is inapplicable.
InapplicableIpRange	The IP range for auto configuration is inapplicable.
NoComport	The device does not support the uc480 ETH comport function.
Paired	The device is paired.
PairingInProgress	The device being paired.
ReadyToOperate	The device is ready for operation.
RebootingFirmwareFailsafe	The device is rebooting to failsafe firmware.
RebootingFirmwareRuntime	The device is rebooting to runtime firmware.
RebootingFirmwareStarter	The device is rebooting to starter firmware.
ReceivingFirmwareRuntime	The device is receiving the runtime firmware.
ReceivingFirmwareStarter	The device is receiving the starter firmware.
RuntimeFirmwareError	A checksum error occurred for the runtime firmware.
TestingIpCurrent	The device is checking its current IP.
TestingIpPersistent	The device is checking its persistent IP.
TestingIpRange	The device is checking the IP range for auto configuration.
Unpaired	The device is unpaired.

3.8.4 DeviceInfoControl**uc480.Types.ETH.DeviceInfoControl**

uc480.Defines.EthControlsStatus	ControlStatus	Internal device ID of the camera: <ul style="list-style-type: none"> • uc480.Defines.EthControlsStatus.Accessible1: Device is accessible, i.e. directly "unicastable"
---------------------------------	---------------	---

		<ul style="list-style-type: none">• uc480.Defines.EthControls.tatus.Accessible2: Device is accessible, i.e. not on persistent IP and adapters IP range for auto configuration is valid• uc480.Defines.EthControls.tatus.AdapterOnDhcp: Adapter is configured to use DHCP• uc480.Defines.EthControls.tatus.Available: Device is available• uc480.Defines.EthControls.tatus.BootBoostActive: Boot-boost is active for this device• uc480.Defines.EthControls.tatus.BootBoostEnabled: Boot-boost is enabled for this device• uc480.Defines.EthControls.tatus.Compatible: Device is compatible• uc480.Defines.EthControls.tatus.FirmwareUploadRuntime: Device is receiving the runtime firmware• uc480.Defines.EthControls.tatus.FirmwareUploadStarter: Device is receiving the starter firmware• uc480.Defines.EthControls.tatus.Initialized: Device is initialized• uc480.Defines.EthControls.tatus.Opened: Device is opened• uc480.Defines.EthControls.tatus.Paired: Device is paired• uc480.Defines.EthControls.tatus.ParingInProgress: Device is being paired• uc480.Defines.EthControls.tatus.PersistentIpUsed: Device runs on persistent IP configuration.• uc480.Defines.EthControls.tatus.Rebooting: Device is rebooting
--	--	--

		<ul style="list-style-type: none"> • uc480.Defines.EthControls.tatus.SetupOk: The adapter setup is OK. • uc480.Defines.EthControls.tatus.ToBeDeleted: Device object is being deleted • uc480.Defines.EthControls.tatus.ToBeRemoved: Device object is being removed • uc480.Defines.EthControls.tatus.UnpairingInProgress : Device is being unpaired.
uint	DeviceID	Device control status

3.8.5 DeviceInformation

uc480.Types.ETH.DeviceInformation

uc480.Types.ETH.AdapterInformation	AdapterInformation	Adapter-related information
uc480.Types.ETH.DeviceHeartbeat	DeviceHeartbeat	Camera-related data retrieved from the camera (from the heartbeat telegram)
uc480.Types.ETH.DeviceInfoControl	DeviceInfoControl	Driver-related information
uc480.Types.ETH.DriverInformation	DriverInformation	Camera-related driver data

Used in methods

- [GetDeviceInfo\(\)](#) (Camera class)
- [GetDeviceInfo\(\)](#) (Information class)

3.8.6 DriverInformation

uc480.Types.ETH.DriverInformation

uint	MinVerStarterFirmware	Minimum version of the compatible starter firmware
uint	MaxVerStarterFirmware	Maximum version of the compatible starter firmware

3.8.7 EthernetConfiguration

uc480.Types.ETH.EthernetConfiguration

uc480.Types.ETH.IpConfiguration	IpConfiguration	IP address and subnet mask
uc480.Types.ETH.MacAddr	MacAddress	MAC address of the camera

3.8.8 IpConfiguration

uc480.Types.ETH.IpConfiguration

System.Net.IPEndPoint	Address	IPv4 address When 0x00000000 (IP address 0.0.0.0) is passed: A valid IP address from the address range specified will be automatically assigned when the camera is opened (see AutoConfigIp).
System.Net.IPEndPoint	Subnetmask	IPv4 subnet mask

Used in methods

- [Get\(\)](#) ([PersistentIp](#) class)
- [Set\(\)](#) ([PersistentIp](#) class)

3.8.9 MacAddr

uc480.Types.ETH.MacAddr

byte	FirstByte	First number of the MAC address
byte	SecondByte	Second number of the MAC address
byte	ThirdByte	Third number of the MAC address
byte	FourthByte	Fourth number of the MAC address
byte	FifthByte	Fifth number of the MAC address
byte	SixthByte	Sixth number of the MAC address
string	Address	MAC address in hexadecimal format

Used in methods

- [Get\(\)](#) ([AutoConfigIp](#) class)
- [GetSupported\(\)](#) ([AutoConfigIp](#) class)
- [Set\(\)](#) ([AutoConfigIp](#) class)
- [Get\(\)](#) ([PersistentIp](#) class)
- [GetSupported\(\)](#) ([PersistentIp](#) class)
- [Set\(\)](#) ([PersistentIp](#) class)

3.9 Complete list of all returns values

uc480.Defines.Status

No	Error	Description
-1	IS_NO_SUCCESS	General error message
0	IS_SUCCESS	Function executed successfully
1	IS_INVALID_CAMERA_HANDLE	Invalid camera handle Most of the uc480 SDK functions expect the camera handle as the first

No	Error	Description
		parameter.
2	IS_IO_REQUEST_FAILED	An IO request from the uc480 driver failed. Possibly the versions of the uc480.dll (API) and the driver file (uc480_usb.sys or uc480_eth.sys) do not match.
3	IS_CANT_OPEN_DEVICE	An attempt to initialize or select the camera failed (no camera connected or initialization 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.
32	IS_INVALID_CAPTURE_MODE	The function can not be executed in the current camera operating mode (free run, trigger or standby).
49	IS_INVALID_MEMORY_POINTER	Invalid pointer or invalid memory ID
50	IS_FILE_WRITE_OPEN_ERROR	File cannot be opened for writing or reading.
51	IS_FILE_READ_OPEN_ERROR	The file cannot be opened.
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 <code>is_SetImageMem()</code> function or create a sequence using the <code>is_AddToSequence()</code> 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.
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

No	Error	Description
		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.
123	IS_NULL_POINTER	Invalid array
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.
129	IS_ACCESS_VIOLATION	An internal error has occurred.
139	IS_NO_USB20	The camera is connected to a port which does not support the USB 2.0 high-speed standard. Cameras without a memory board cannot be operated on a USB 1.1 port.
140	IS_CAPTURE_RUNNING	A capturing operation is in progress and must be terminated first.
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.
156	IS_TRIGGER_NOT_ACTIVATED	The function is not possible as trigger is disabled.
157	IS_OPERATION_ABORTED	The dialog was canceled 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

No	Error	Description
		currently set pixel clock frequency.
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 color 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 I ² C device address
177	IS_COULD_NOT_CONVERT	The current image could not be processed.
178	IS_TRANSFER_ERROR	Transfer error. Frequent transfer errors can mostly be avoided by reducing the pixel rate.
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.
181	IS_INVALID_HOST_IP_HIBYTE	Invalid HIBYTE of host address
182	IS_CM_NOT_SUPP_IN_CURR_DISPLAYMODE	The color mode is not supported in the current display mode.
183	IS_NO_IR_FILTER	No IR filter available
184	IS_STARTER_FW_UPLOAD_NEEDED	The camera's starter firmware is not compatible with the driver and needs to

No	Error	Description
		be updated.
185	IS_DR_LIBRARY_NOT_FOUND	The DirectRenderer library could not be found.
186	IS_DR_DEVICE_OUT_OF_MEMORY	Not enough graphics memory available.
187	IS_DR_CANNOT_CREATE_SURFACE	The image surface or overlay surface could not be created.
188	IS_DR_CANNOT_CREATE_VERTEX_BUFFER	The vertex buffer could not be created.
189	IS_DR_CANNOT_CREATE_TEXTURE	The texture could not be created.
190	IS_DR_CANNOT_LOCK_OVERLAY_SURFACE	The overlay surface could not be locked.
191	IS_DR_CANNOT_UNLOCK_OVERLAY_SURFACE	The overlay surface could not be unlocked.
192	IS_DR_CANNOT_GET_OVERLAY_DC	Could not get the device context handle for the overlay.
193	IS_DR_CANNOT_RELEASE_OVERLAY_DC	Could not release the device context handle for the overlay.
194	IS_DR_DEVICE_CAPS_INSUFFICIENT	Function is not supported by the graphics hardware.
195	IS_INCOMPATIBLE_SETTING	Because of other incompatible settings the function is not possible.
196	IS_DR_NOT_ALLOWED WHILE DC IS ACTIVE	A device context handle is still open in the application.
197	IS_DEVICE_ALREADY_PAIRED	The device is already paired.
198	IS_SUBNETMASK_MISMATCH	The subnet mask of the camera and PC network card are different.
199	IS_SUBNET_MISMATCH	The subnet of the camera and PC network card are different.
200	IS_INVALID_IP_CONFIGURATION	The configuration of the IP address is invalid.
201	IS_DEVICE_NOT_COMPATIBLE	The device is not compatible to the drivers.
202	IS_NETWORK_FRAME_SIZE_INCOMPATIBLE	The settings for the image size of the camera are not compatible to the PC network card.
203	IS_NETWORK_CONFIGURATION_INVALID	The configuration of the network card is invalid.
204	IS_ERROR_CPU_IDLE_STATES_CONFIGURATION	The configuration of the CPU idle has failed.
205	IS_DEVICE_BUSY	The camera is busy ad cannot transfer the requested image.
206	IS_SENSOR_INITIALIZATION_FAILED	The initialization of the sensor failed.

4 .NET version history

New in version 4.31

New classes and methods:

- [Feature](#) class: [ImageEffect](#)
- [Focus](#) class: [Zone](#) and [Trigger](#)
- [Auto](#) class: [GetStatus](#)
- [Zoom](#) class: [GetDefault](#) and [GetRange](#)
- [uc480.Configuration](#) class: [Ipo](#)
- Updated [SensorInfo](#)

New overloading for following methods:

- [Flash](#) class: [GetDelayRange](#) and [GetDurationRange](#)
- [Pwm](#) class: [GetDutyCycleRange](#) and [GetFrequencyRange](#)

Changed method calls for:

- In the [Shutter](#) class: [GetMax](#) and [SetMax](#)

All obsolete methods from former versions are removed.

New in version 4.30

- New classes in [Device](#) class: [AdditionalPosition](#) and [Height](#)
- New class in the [Temperature](#) class: [LscModel](#)
- New method in [Temperature](#) class: [GetNumerical\(\)](#)
- New methods in [PixelClock](#) class: [GetList\(\)](#) and [GetNumber\(\)](#)
- New method in [Information](#) class: [GetLastError\(\)](#)
- New defines in [uc480.Types.ETH.DeviceHeartbeat](#) and [uc480.Types.ETH.DeviceInfoControl](#)
- New events in [Camera](#) class: [uc480.Camera.EventAutoFocusFinished](#) and [uc480.Camera.EventFirstPacket](#)
- Renamed classes, methods and defines:

	Old name	New name
Class	Model	RgbModel
Methods	AutoFeaturesSensorContrast.GetCorrection() AutoFeaturesSensorContrast.GetCorrectionRange() AutoFeaturesSensorContrast.GetDefaultCorrection() AutoFeaturesSensorContrast.GetEnableFDTAOI() AutoFeaturesSensorContrast.GetSupportedCorrection() AutoFeaturesSensorContrast.GetSupportedFDTAOI() AutoFeaturesSensorContrast.SetCorrection() AutoFeaturesSensorContrast.SetEnableFDTAOI()	Correction.Get() Correction.GetRange() Correction.GetDefault() FaceDetectionAoi.GetEnabled() Correction.GetSupported() FaceDetectionAoi.GetSupported() Correction.Set() FaceDetection.SetEnable()
Defines	uc480.Defines.ColorTemperatureModel uc480.Defines.ColorMode	uc480.Defines.ColorTemperatureRgbMode uc480.Defines.ColorTemperature

New in version 4.22

- New class in the [Display](#) class: [AutoRender](#)
- New classes in the [Feature](#) class: [AoiMerge](#), [BlackReference](#), [FpnCorrection](#), [JpegCompression](#), [Log](#), [NoiseReduction](#), [SensorBitDepth](#), [SensorSourceGain](#), [Temperature](#), [Timestamp](#)
- New class in the [Exposure](#) class for setting the dual exposure time: [Dual](#)
- The [AutoFeatures](#) class was reworked and new structured. It now contains two main classes: [Sensor](#) and [Software](#)
- New methods in the [Memory](#) class.
- Removed classes by new methods

Old class	Replaced by method
Convert	Convert()
OptCmaeraTiming	Optimal()

- Renamed classes:

Class	Old name	New name
uc480.Focus	Auto	Focus.Auto
	Manual	Focus.Manual
uc480.Size	AOI	Size.AOI
	Binning	Size.Binning
	ImageFormat	Size.ImageFormat
	Multi	Size.AOI.Multi
	Subsampling	Size.Subsampling
	SensorScaler	Size.SensorScaler
	Sequences	Size.AOI.Sequence
uc480.Gain	Boost	Gain.Hardware.Boost
	ConvertScaledToFactor	Gain.Hardware.ConvertScaledToFactor
	Factor	Gain.Hardware.Factor
	Hardware	Gain.Hardware
	Scaled	Gain.Hardware.Scaled
uc480.Trigger	BurstSize	Trigger.Burst
	Counter	Trigger.Counter
	Debounce	Trigger.Debounce
	Delay	Trigger.Delay
uc480.Timing	PixelClock	Timing.PixelClock
	Framerate	Timing.Framerate
	VsyncCount	Timing.VsyncCount
	Exposure	Timing.Exposure
	FineIncrement	Timing.Exposure.Fine
	LongExposure	Timing.Exposure.Long
	VsyncCount	Timing.VsyncCount
uc480.Lut	LUT	Lut
	Preset	Lut.Preset
uc480.IO	Flash	IO.Flash
	GPIO	IO.Gpio
	LED	IO.Led
	PWM	IO.Pwm
uc480.HDR	Kneepoints	Hdr.Kneepoints
uc480.Memory	ImageBuffer	Memory.ImageBuffer
	Sequence	Memory.Sequence
uc480.Image	Measure	Image.Measure
uc480.GigE	ImageDelay	GigE.Transfer.ImageDelay
	PacketInterval	GigE.Transfer.PacketInterval
	Transfer	GigE.Transfer
uc480.DeviceFeature	LineScan	DeviceFeature.Linescan

- Changed methods:

Obsolete methods	New methods
DisplayImage.Set()	Display.Render()
IOLed.SetState()	Led.Get()
IOLed.GetState()	Led.Set()
IOLed.ToggleState()	Led.Toggle()
uc480.IOPwm.GetSupportedGPIOs()	Pwm.SetMode()
CPUIdleState.GetDisableOnOpen_AC()	CPUIdleState.SetDisableOnOpen()
CPUIdleState.GetDisableOnOpen_DC()	CPUIdleState.GetEnable()
CPUIdleState.SetDisableOnOpen_AC()	CPUIdleState.GetDisableOnOpen()
CPUIdleState.SetDisableOnOpen_DC()	
System.GetDLLVersion()	System.GetNetVersion() System.GetApiVersion()

- Changed methods calls for: [Flash.GetMode\(\)](#), [Flash.SetMode\(\)](#), [Pwm.GetMode\(\)](#), [Pwm.SetMode\(\)](#)

New in version 4.21

- Instead of two separate dll files there is only one dll file (uc480DotNet.dll). This one fits both 32 bit and 64 bit systems.
- New classes [uc480.Video](#) and [uc480.Tools.Video](#) for capturing videos.
- [New and renamed color formats](#)
- Changed method calls for [IOPgio](#) and [I2C](#)
- New classes and methods for setting image buffer ([MemoryImageBuffer](#)), measuring ([ImageMeasure](#)) and white balance ([AutoFeaturesSoftwareWhitebalance](#)).

New in version 4.20

- Methods [Image.Load\(\)](#) changed: Optionally loads the image into a newly allocated image memory.
- Method [Image.GetValues\(\)](#) changed: Call of method slightly changed and variant for monochrome added.
- Method [System.GetOsVerion\(\)](#) removed.

5 Appendix

5.1 PCs with Energy Saving CPU Technology

This application note is related to all DCx USB cameras connected to PC systems using current CPU models that implement modern energy saving technologies.

Symptoms:

- Low USB bandwidth provided by the PC system
- TransferFailed errors occurring even at moderate pixel clock settings
- Camera operates at low speed only

Summary:

Current CPUs with modern energy saving features can cause bandwidth limitations on USB. The only available approach to this issue is to disable CPU sleep states. Unfortunately this is not possible for all systems.

Detailed explanation:

Modern CPUs like Intel i5 & i7 and others make use of advanced energy saving technologies ensuring a low power consumption and long battery life for mobile

devices. Additionally those CPU implement features for increasing the performance of single cores if there is enough thermal headroom available when other cores have little load.

A basic idea to achieve this is to put a CPU core to sleep while there is nothing to do for it. Various different activity states of CPU cores are available in modern CPUs. These CPU states are referred to as "C-states". C0 is the working state of a core.

Increasing numbers refer to less activity and longer wake up times. Current CPU fall down to variations of the C3 state which are referred to as "Sleep", "Deep Sleep" and similar.

Unfortunately negative effects of the sleep states have shown up. It is observed that the available bandwidth of PC busses drops significantly when part of the CPU enters these states.

The operation of DCx USB cameras is affected by the sleep states because they reduce the speed of the USB system. The available bandwidth on the USB may drop down to around 30% of the maximum bandwidth when the CPU, or one of its cores, enters sleeping states.

One would expect that a CPU core will not fall into a sleep state while it is obviously needed for the operation of the USB. But obviously USB data transfers do not prevent the CPU from falling to sleep. If the code execution load of a CPU core is low enough it will fall asleep and immediately reduce the USB bus speed.

For operation at high frame rates DCx cameras require an adequate USB bandwidth which might not be available when CPU cores are in sleep states.

Advice:

If you seem to be running into this low bandwidth issue please check and try the following. These first hints are general recommendations for issues with the USB

data transfer. You can check the USB performance with the "Optimum" pixel clock settings checkbox in uc480 Demo software. A good USB system should be able to reach a pixel clock setting near the maximum value.

- Please remove other USB devices from the system (USB keyboard and mouse are fine). Run tests with only one camera connected at once.
- Make sure using a USB port directly on the mainboard. Front panel or other ports are connected

to the mainboard with poor cabling quality frequently.

- Make sure to use USB2.0 certified cables to connect the camera.
- If you are using USB hubs or extensions: Run a test without these devices, connect the camera directly to the PC.
- Disable other equipment that is connected via USB. For example WLAN and Bluetooth adapters might use USB to connect.
- If you are using a mobile PC: run it on mains power, not battery.
- Check your energy saving options in the operating system. Disable energy saving features and set the available features to “full performance” or similarly named options.

If you checked the above and still observe low USB performance you might be experiencing the issue with CPU sleep states.

5.2 Exclusion of Liability and Copyright

Thorlabs Scientific Imaging has taken every possible care in preparing this Operation 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.

Should you require further information about this product or encounter specific problems that are not discussed in sufficient detail in the User Manual, please contact your nearest Thorlabs office.

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 *Thorlabs Scientific Imaging*.

Copyright © Thorlabs Scientific Imaging 2015. All rights reserved.

5.3 Thorlabs Worldwide Contacts

USA, Canada, and South America

Thorlabs, Inc.
56 Sparta Avenue
Newton, NJ 07860
USA
Tel: 973-579-7227
Fax: 973-300-3600
www.thorlabs.com
www.thorlabs.us (West Coast)
Email: sales@thorlabs.com
Support: techsupport@thorlabs.com

UK and Ireland

Thorlabs Ltd.
1 Saint Thomas Place, Ely
Cambridgeshire CB7 4EX
United Kingdom
Tel: +44-1353-654440
Fax: +44-1353-654444
www.thorlabs.com
Email: sales.uk@thorlabs.com
Support: techsupport.uk@thorlabs.com

Europe

Thorlabs GmbH
Hans-Böckler-Str. 6
85221 Dachau
Germany
Tel: +49-8131-5956-0
Fax: +49-8131-5956-99
www.thorlabs.de
Email: europe@thorlabs.com

Scandinavia

Thorlabs Sweden AB
Mölndalsvägen 3
412 63 Göteborg
Sweden
Tel: +46-31-733-30-00
Fax: +46-31-703-40-45
www.thorlabs.com
Email: scandinavia@thorlabs.com

France

Thorlabs SAS
109, rue des Côtes
78600 Maisons-Laffitte
France
Tel: +33-970 444 844
Fax: +33-811 38 17 48
www.thorlabs.com
Email: sales.fr@thorlabs.com

Brazil

Thorlabs Vendas de Fotônicos Ltda.
Rua Riachuelo, 171
São Carlos, SP 13560-110
Brazil
Tel: +55-16-3413 7062
Fax: +55-16-3413 7064
www.thorlabs.com
Email: brasil@thorlabs.com

Japan

Thorlabs Japan, Inc.
Higashi Ikebukuro
Q Building 2nd Floor 2-23-2
Toshima-ku, Tokyo 170-0013
Japan
Tel: +81-3-5979-8889
Fax: +81-3-5979-7285
www.thorlabs.jp
Email: sales@thorlabs.jp

China

Thorlabs China
Room A101, No. 100
Lane 2891, South Qilianshan Road
Putuo District
Shanghai 200331
China
Tel: +86-21-60561122
Fax: +86-21-32513480
www.thorlabs.hk
Email: chinasales@thorlabs.com

Index

*

.NET

- new in version 6
- .NET library 11

A

Acquisition	11, 14	position	81
capture	14	position range	82
finished	16	range additional position	79
freeze	15	set	85
started	16	set additional position	79
stop	16	set enable	85
Anit-flicker mode		set height	81
set	20	set position	83
Anti-flicker mode	18	supported	84
default	19	vertical	77
get	19		
supported	19	API version	366
AOI	278	Area of interest	139
auto brightness	285, 288	Auto backlight compensation	20
fast position changes	286, 288	get enable	21
fast position changesAOI	288	set enable	21
get	284	supported	21
get X position	284	Auto brightness	
get Y position	284	AOI	285, 288
multi	280	Auto contrast	22
original	285	default contrast	22
position range	286	face detection AOI	24, 25
sequence	282	get	22
set	288	range	23
size range	287	set	24
white balance	287, 289	supported	23
AOI merge mode		Auto feature	17
additional position	78	anti-flicker mode	18
default	83	auto backlight	20
default additional position	78	compensation	
default position	82	auto contrast	22
get	83	auto frame rate	25, 35
get additional position	78	auto gain	27, 38
get default height	80	auto gain/auto shutter	29
get enable	84	auto reference	41
get height	80	auto shutter	31, 43
get list	80	auto skip frame	37
get number	81	auto speed	45
get position	82	auto white balance	33
height	79	get information	58
		hysteresis	40
		sensor	17
		software	34
		Auto focus	155
		distance	156
		enable	157
		state	155
		supported	157
		Auto frame rate	25, 35
		enable	35, 36
		get enable	26
		set enable	26

Auto frame rate supported	25, 35 26, 36	gain range get hysteresis	49 50
Auto gain default	27, 38 27	get offset get skip frame	51 47
enable	28, 38, 39 38, 39	get speed	53
maximum	38, 39	hysteresis	50
photometry mode	28, 29	hysteresis range	50
state	27	offset	51
supported	28, 39	offset range	52
Auto gain/auto shutter	29	reference range	55
get enable	29	set hysteresis	51
set enable	30	set offset	52
supported	30	set skip frame	48
Auto hysteresis	40	set speed	54
get	40	skip frame	47
range	40	skip frame range	48
set	41	speed	53
Auto reference default	41 42	speed range	53
get	41	state	33
range	42	supported	34, 55
set	42	type	56, 57
Auto render default mode	134 135	AutoFeatures	11
get enable	134	Automatic IP	
mode	134, 136	GigE uEye camera	357
set enable	136	set IP	357, 358
window	136, 137	support	358
Auto shutter default	31, 43 31	B	
enable	32, 43, 44	Bandwidth	
maximum	43, 44	used	213
photometry mode	32, 33	Binning	289
state	31	factor	290
supported	32, 44	get	290
Auto skip frame	37	set	292
get	37	support	291
range	37	type	292
set	38	Black level	292
Auto speed default	45 45	default	58
get	45	default offset	59
range	46	get	59
set	46	get offset	60
Auto white balance color model	33, 46 54, 55, 57	offset range	60
default offset	51	set	61
default speed	53	set offset	62
disable	54, 57	supported	60
enable	34, 54, 57	supported offset	61
gain	48	Black reference columns	
		default	86
		get	86
		set	87
		supported	86

BlackLevel	11	COM port number	
Boot boost	346	get	180
add ID	347	GigE uEye HE	180
clear list	347	Configuration	345
enable	348, 349	GigE camera	356
get list	348	GigE uEye camera	180
remove ID	348	Configuring	
set ID	349	.NET	10
Burst trigger mode		Connecting a DCx Camera	9
default	324	Contrast	
get	324	correction	22, 23
GigE uEye camera	324	Conversion	
range	325	apply parameters	201
set	325	CPU idle state	350
support	325	get disable	350
Bus speed	207	get enable	350
		set disable	351
		supported	351

C

Camera	342	D	
connected	364	Device	11, 76
EEPROM	143	auto exit	119
exit	11, 344	get camera ID	118
get device info	365	get device ID	118
get device number	365	information	210
information	207, 364	set camera ID	119
init	11, 344	Device feature	76
optimal camera timing	322	AOI merge mode	77
parameter set	263	black reference columns	85
status	208, 213	FPN correction	87
timing	307	JPEG compression	91
Camera -> Device	76	line scan	93
Camera list	364	Log mode	95
Capture		mulit integration mode	101
status	209	noise reduction	104
Capture status		sensor bit depth	106
reset	213	sensor source gain	108
Color	11	shutter mode	110
conversion mode	63, 64, 67	temperature	112
converter	62	time stamp	113
correction	67, 68, 69	XS HS mode	116
default temperature	75	Direct3D	120
depth	69, 70	DirectRenderer	11
get temperature	74	Display	11, 133
temperature	70, 76	render	140
temperature range	75	Dual exposure	308
Color temperature model	73	default	309
default	73	get	309
get	73	range	309
set	74	set	310
supported	74		

Dual exposure supported	308 310	global parameter mode parameter	217 218, 219 215, 218, 220
E		supported GPIO	219
Edge enhancement	141	Focus	11, 154
default	142	measuring window	159
get	142	trigger	163
range	142	zone	159
set	143	Format	
EdgeEnhancement	11	image	293
EEPROM	11, 143	FPN correction	
read	144	default	88
write	144	get	88
Error		set	89
get last	212	supported	88
Error codes	405	Frame rate	315
Error report	214	default	316
Exposure	307	get	315
default	314	get current	316
dual	308	range	316
enable long time	313	set	318
fine increment	311	time range	317
get	313	G	
long time	312	Gain	11, 164
long time range	312	support	176
range	314	Gain boost	164
set	315	enable	165
support	314	state	164
support long time	313	supported	165
F		Gain factor	167
Face detection	145	default blue	168
enable	152	default green	168
face list	147	default master	169
face number	148	default red	169
overlay line width	149, 153	get blue	169
overlay number	148, 152	get green	169
resolution	147, 152	get master	170
search angle	149, 150, 153, 154	get red	170
state	146	set blue	170
support	150	set green	171
suspend	151, 154	set master	171
FaceDetection	11	set red	171
Fixed pattern noise	87	Gamma	11, 176
Flash	214	hardware	176
delay range	216	software	178
duration range	217	General purpose I/O	220
freerun	216, 219	GigE	11
		GigE uEye camera	180

GPIO	220	Image acquisition	
direction	221, 223	capture	14
state	221, 223	finished	16
support	222	freeze	15
		started	16
		stop	16
H			
Hardware gain	164	Image buffer	242
Hardware gamma	176	disable	242, 244
get enable	177	enable	242, 244
set	177	iteration	243
support	177	iteration range	243
Hdr	11	range	242
HDR mode	185	supported	243
enable	187	transfer image	244
support	187		
High dynamic range	185	Image convert	201
Hot pixel		Image data transfer	
camera correction	191	GigE uEye camera	180
camera list	188		
correction	187	Image delay	181
correction mode	195, 196	default	181
disable correction	195	get	181
enable correction	195	range	182
factory list	191	set	183
factory list	191, 192	support	182
merge list	196		
number	190	Image effect	
sensor correction	197	default	90
software list	192	get	89
user list	189, 190, 191, 193, 194	set	90
		supported	90
Hotpixel	11	Image format	293
HS mode -> XS HS mode	116	free selection of AOI	294
		get list	294
		set	294
I		Image save	203
I/O	214	Image scaling	295
I2C	11	factor	295
I2C bus	197	information	296
read	197	mode	296
write	198	range	297
Image	11	set	297
acquisition	14	steps	297
color	199		
histogram	201	Image size	278
information	211	Image stabilization	204
load	199, 202	enable	205
RGB value	202	state	204
save	199	support	205
		ImageStabilization	11
		Information	11, 206
		bus speed	207
		Input	214
		Installation	8, 9

IO	11	LUT	11, 228
		enable	234, 238
J		load	238
JPEG compression		mode	235, 239
default	92	preset	229, 230,
get	91		231, 232,
range	92		233, 237,
set	93	RAW format	234
supported	92	save	238
JPEG mode	91	state	234, 235,
			236
K		supported	236
Kneepoint	185	user value	237, 240
get	185	value	236
information	186		
set	186		
L		M	
LED	224	Manual focus	158
state	224	range	158
toggle	225	set	159
Lens shading model	71	state	158
default	72	Measure	
get	71	AOI	200
set	72	inquire	199, 200
supported	72	preset	199, 200
Line scan	93	set AOI	199
get mode	93	Memory	11, 241
get number	94	active	253, 258
set mode	95	allocate	250
set number	95	bits per pixel	254
support	94	copy	251
supported	94	copy to array	252
Log mode	93	copy to bitmap	252
default	100	free	253
default gain	97	get last	253
default manual value	98	get list	254
gain range	97	get pitch	255
get manual gain	96	height	256
get manual value	98	image	251
get mode	99	inquire	257
manual gain	96	lock	257
manual value	98	locked	255
range of manual value	99	pointer	259
set manual gain	97	size	256
set manual value	99	to bitmap	258
set mode	100	unlock	259
supported	100	width	256
Lookup table	228	Message	260
		disable	260
		enable	260
		Microsoft DirectX Runtime	120

Mode		set	359
display	138	Packet interval	183
get display mode	138	default	184
set display mode	138	get	183
Multi AOI	280	range	184
axes	281	set	185
disable	280	support	184
support	281	Parameter	11, 263
Multi integration		Parameter set	352
default	102	clear	263
get mode	101	get number	263
get params	102	get supported	264
scope	102	initial	352, 353
set mode	103	load	264
set params	104	reset	265
support	103	save	265
		supported	352
N		PCs with Energy Saving CPU	414
.NET version	366	Technology	
Noise reduction		Persistent IP	
default	105	get	360
get	104	GigE uEye camera	359
set	105	set	361
supported	105	support	360
O		Pixel clock	319
OpenGL	120	default	319
OpenMP	355	discrete	320
default	355	frequency	322
enable	356	get	319
state	355, 356	list	320
Output	214	number	320
Overlay		range	320
clear	122	set	321
graphics	124, 126	Pixel format	265
hide	125	bits per pixel	266
key color	124, 126	bytes per pixel	267
load image	125	get	266
overlay	123	set	267
position	127	PixelFormat	11
release DC	126	Position	
show	128	move	139
size	124, 125, 127	set	139
transparency	123, 127	Pulse-width modulation	225
visible	128	PWM	225
		duty cycle	225
		frequency	226
		mode	226, 227
		parameter	227, 228
		supported GPIO	227
P		Packet filter	359

R

Raster operation	269	get	273
Rendering	120, 139	set	275
disable scaling	128	support	274, 275
enable image scaling	129	ScenePreset	11
enable scaling	129	Sensor	
overlay	121	auto feature	17
position	130	blue gain channel	166
set scaling	130	free selection of AOI	294
steal function	129, 131,	gain channels	165
	132	green gain channel	166
	130	information	212
supported modes	130	master gain channel	166
synchronization	131	red gain channel	167
update	132	Sensor bit depth	
VSYNC	132, 133	default	106
window handle	131	get	106
Return values	405	set	107
Ring buffering	245, 246,	supported	107
	247, 248,	supported bit depth	107
Rop effect	249	Sensor source gain	
get	269	default	108
set	269	get	108
RopEffect	270	range	109
	11	set	109
		supported	109

S

Saturation	11, 270	clear	246
default	271	exit image queue	246
get	271	get last	247
range	271	get sequence number	247
set	272	image memory ID	249
support	272	init image queue	248
Scaled gain	171	lock	248
default blue	172	locked	247
default green	173	unlock	249
default master	173	wait	249
default red	173	Sequence AOI	282
get blue	174	enable	283
get green	174	parameter	282, 283
get master	174	state	282
get red	174	support	283
set blue	175	SeRing buffering	247
set green	175	Sharpness	11, 276
set master	175	default	276
set red	176	get	276
Scaling		range	277
image	295	set	278
Scene preset	273	support	277
default	274	Shutter mode	

Shutter mode		set	306
get	110	Timing	11, 307
set	111	Transfer	
supported	111	default image delay	181
supported mode	110	default packet interval	184
Size	11	get image delay	181
image	278	get packet interval	183
Skip frame	37	GigE uEye camera	180
Software		image delay	181, 182
auto feature	34	image delay range	182
Software gamma	178	packet interval	183, 184
default	179	packet interval range	184
get	179	set image delay	183
range	179	set packet interval	185
set	178	Trigger	11, 322
Starter firmware	361	debounce	327, 328, 329, 330
GigE uEye CP	362	force	332
GigE uEye RE	363	mode	332, 333
GigE uEye SE	363	state	332
upload	363	support	333
upload duration	362, 363	Trigger debounce	
Subsampling	298	default time	328
factor	299	delay time	330
get	298	mode	328, 329
set	301	support	329
suported	300	time	328
support	299	time range	329
type	300	Trigger delay	330
System		get	331
version	365	range	331
System requirements	8	set	331
T		Trigger mode	
Temperature		counter	326
get	112	get counter	326
supported	113	reset counter	327
Test image	302	U	
get	302	uc480 Camera Manager	9
range	303	uc480 Software Installation	9
set	305	uEye	
support	303	camera	11
supported	305	uEye camera	
TestImage	11	information	206
Time stamp		uEye.Types	
get	113	AoiSequenceParameter	373
reset	115	CameralInfo	375
set	115	CameraInformation	376
supported	114	CaptureStatus	377
Timeout	11, 306	CaptureStatusApi	378
get	306		

uEye.Types	373	reset count	337
CaptureStatusDriver	378	reset counter	370
CaptureStatusEth	379	running	336
CaptureStatusUsb	379	size	337, 369, 371
ConversionParameter	380		
DeviceInfoControl	380	start	338, 372
DeviceInfoHeartbeat	380	stop	339, 372
DeviceInfo	381	video ID	337
FaceDetectionInformation	381	width	371
ImageFormatInfo	383	VSYNC counter	321
ImageInfo	389	get	321
KneePointInformation	391		
LutState	391		
LutSupportInfo	392		
MultiIntegrationCycles	392	White balance	
MultiIntegrationScope	392	AOI	287, 289
SensorInfo	393		
SensorScalerInformation	394		
TimestampConfiguration	395	XS HS mode	116
uEye.Types.AutoFeatures	395	default	117
BrightStatus	396	get	117
Information	396	set	118
WhitebalanceChannelStatus	399	supported	117
WhitebalanceStatus	399	supported mode	117
uEye.Types.ETH	399		
AdapterInformation	400		
AutoConfigIPSetup	400		
DeviceHeartbeat	400	Zone	
DeviceInfoControl	402	count	161
DeviceInfo	404	default AOI	160
DriverInformation	404	default weight	162
EthernetConfiguration	404	get AOI	160
IpConfiguration	405	position	161
MacAddr	405	set AOI	163
		set weight	163
		size	161
		supported	162
		weight	162
V			
Video	11, 334, 367	Zoom	11, 339
add frame	367	default	340
close	368	get	339
color mode	371	get list	340
discarded frames	369	range	341
exit	368	set	342
frame count	335, 368	support	341
frame rate	335, 338, 371		
height	371		
init	370		
lost frames	336		
open	370		
quality	336, 338, 372		