



## CCD and CMOS Cameras

**DCU223x, DCU224x**

**DCC1240x**

**DCC1545M, DCC1645C**

**DCC3240X**

**DCC3260X**

## DCx Camera .NET Manual



**2018**

---

Version: 4.81  
Date: 7/30/2018

# Contents

<b>Foreword</b>	<b>5</b>
<b>1 General Information</b>	<b>6</b>
1.1 What is new in this version?	6
1.2 About this manual	7
<b>2 Installation and requirements</b>	<b>8</b>
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
<b>3 .NET library</b>	<b>11</b>
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	49
3.1.1.4 Color	53
3.1.1.5 Device	63
3.1.1.6 DirectRenderer	86
3.1.1.7 Display	99
3.1.1.8 EdgeEnhancement	107
3.1.1.9 EEPROM	109
3.1.1.10 Gain	110
3.1.1.11 Gamma	123
3.1.1.12 Hotpixel	125
3.1.1.13 I2C	134
3.1.1.14 Image	136
3.1.1.15 Information	142
3.1.1.16 IO	150
3.1.1.17 Lut	166
3.1.1.18 Memory	180
3.1.1.19 Messaging	200
3.1.1.20 Parameter	203
3.1.1.21 PixelFormat	205
3.1.1.22 RopEffect	209
3.1.1.23 Saturation	210
3.1.1.24 Size	213
3.1.1.25 TestImage	232
3.1.1.26 Timeout	236

---

3.1.1.27	Timing	237
3.1.1.28	Trigger	250
3.1.1.29	Prescaler	260
3.1.1.30	Video	265
3.1.1.31	Camera	270
3.1.1.32	Exit	271
3.1.1.33	Init	272
<b>3.2 uc480.Configuration</b>		<b>273</b>
3.2.1	BootBoost	274
3.2.1.1	AddId	274
3.2.1.2	ClearIdList	275
3.2.1.3	GetEnable	275
3.2.1.4	GetIdList	275
3.2.1.5	RemovId	276
3.2.1.6	SetEnable	276
3.2.1.7	SetIdList	276
3.2.1.8	Wait	277
3.2.2	CPUIdleState	277
3.2.2.1	GetDisableOnOpen	277
3.2.2.2	GetEnable	278
3.2.2.3	GetSupported	278
3.2.2.4	SetDisableOnOpen	279
3.2.3	InitialParameterset	279
3.2.3.1	GetEnable	280
3.2.3.2	GetSupported	280
3.2.3.3	SetEnable	280
3.2.4	Ipo	281
3.2.4.1	GetEnable	281
3.2.4.2	GetSupported	281
3.2.4.3	SetEnable	282
3.2.5	OpenMP	282
3.2.5.1	GetEnable	282
3.2.5.2	GetEnableDefault	283
3.2.5.3	GetSupported	283
3.2.5.4	SetEnable	284
3.2.6	MemoryMode	284
3.2.6.1	Get	284
3.2.6.2	GetDefault	285
3.2.6.3	GetSupported	285
3.2.6.4	Set	286
<b>3.3 uc480.Info</b>		<b>286</b>
3.3.1	Camera	286

---

---

3.3.1.1	GetCameraList	287
3.3.1.2	GetDeviceInfo	287
3.3.1.3	GetNumberOfDevices	287
3.3.2	System	288
3.3.2.1	GetNetVersion	288
3.3.2.2	GetApiVersion	288
3.4	uc480.Tools	289
3.4.1	Video	289
3.4.1.1	AddFrame	290
3.4.1.2	Close	291
3.4.1.3	Exit	291
3.4.1.4	GetFrameCount	291
3.4.1.5	GetLostCount	292
3.4.1.6	GetSize	292
3.4.1.7	Init	292
3.4.1.8	Open	293
3.4.1.9	ResetCounter	293
3.4.1.10	SetFramerate	293
3.4.1.11	SetImageSize	294
3.4.1.12	SetQuality	294
3.4.1.13	Start	295
3.4.1.14	Stop	295
3.5	uc480.Types	295
3.5.1	AoiSequenceParameter	296
3.5.2	CameraInfo	297
3.5.3	CameraInformation	298
3.5.4	CaptureStatus	298
3.5.5	CaptureStatus.CaptureStatusApi	299
3.5.6	CaptureStatus.CaptureStatusDriver	299
3.5.7	CaptureStatus.CaptureStatusEth	300
3.5.8	CaptureStatus.CaptureStatusUsb	300
3.5.9	ConversionParameter	301
3.5.10	DeviceInfoControl	301
3.5.11	DeviceInfoHeartbeat	301
3.5.12	DeviceInformation	302
3.5.13	ImageFormatInfo	303
3.5.14	ImageInfo	306
3.5.15	LutState	307
3.5.16	LutSupportInfo	307
3.5.17	SensorInfo	307
3.6	uc480.Types.AutoFeature	308
3.6.1	BrightStatus	309

---

3.6.2	Information	309
3.6.3	WhitebalanceChannelStatus	312
3.6.4	WhitebalanceStatus	312
3.7	Complete list of all returns values	312
<b>4</b>	<b>.NET version history</b>	<b>317</b>
<b>5</b>	<b>Appendix</b>	<b>321</b>
5.1	PCs with Energy Saving CPU Technology	321
5.2	Exclusion of Liability and Copyright	322
5.3	Thorlabs Worldwide Contacts	323

**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).

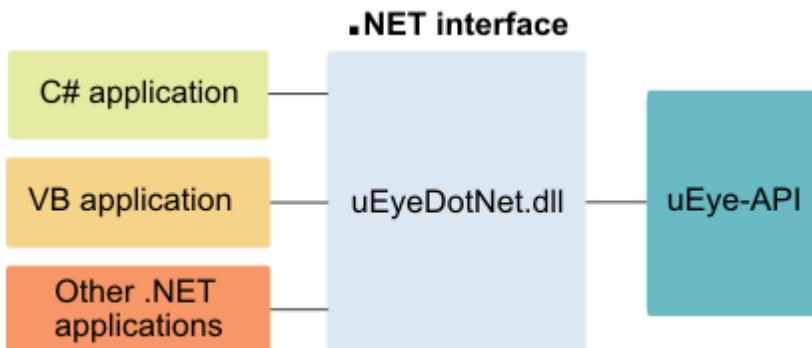


Fig. 1: .NET interface

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. 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 4.81

New classes and methods:

- New class in [Feature](#) class: ExternalInterface
- New class in [uc480.Configuration](#) class: MemoryMode

Removed methods:

- The `Set` method in the `MemoryMode` class provided by `Feature` was removed. Use the `Set` method of the `MemoryMode` class provided by `uc480.Configuration` instead.

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:

	<b>Recommended</b>
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.Info](#)
- [uc480.Tools](#)
- [uc480.Types](#)
- [uc480.Types.AutoFeature](#)
- [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:

- |  |  |  |
|--|--|--|
| <ul style="list-style-type: none"> <li>• <a href="#">Acquisition</a></li> <li>• <a href="#">AutoFeatures</a></li> <li>• <a href="#">BlackLevel</a></li> <li>• <a href="#">Color</a></li> <li>• <a href="#">Device</a></li> <li>• <a href="#">DirectRenderer</a></li> <li>• <a href="#">Display</a></li> <li>• <a href="#">EdgeEnhancement</a></li> <li>• <a href="#">EEPROM</a></li> <li>• <a href="#">Gain</a></li> </ul> | <ul style="list-style-type: none"> <li>• <a href="#">Hotpixel</a></li> <li>• <a href="#">I2C</a></li> <li>• <a href="#">Image</a></li> <li>• <a href="#">Information</a></li> <li>• <a href="#">IO</a></li> <li>• <a href="#">Lut</a></li> <li>• <a href="#">Memory</a></li> <li>• <a href="#">Messaging</a></li> <li>• <a href="#">Parameter</a></li> </ul> | <ul style="list-style-type: none"> <li>• <a href="#">PixelFormat</a></li> <li>• <a href="#">RopEffect</a></li> <li>• <a href="#">Saturation</a></li> <li>• <a href="#">Size</a></li> <li>• <a href="#">TestImage</a></li> <li>• <a href="#">Timeout</a></li> <li>• <a href="#">Timing</a></li> <li>• <a href="#">Trigger</a></li> <li>• <a href="#">Video</a></li> </ul> |
|--|--|--|

## Methods

<a href="#"><u>Camera</u></a>	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, <a href="#"><u>Init()</u></a> must be called.
<a href="#"><u>Exit</u></a>	Disables the camera handle and releases the data structures and memory areas taken up by the uc480 camera.
<a href="#"><u>Init</u></a>	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 <a href="#">Information.GetCaptureStatus()</a> .
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 <a href="#">Init()</a> 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
<a href="#">Capture</a>	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.
<a href="#">Freeze</a>	Acquires a single image from the camera.
<a href="#">HasStarted</a>	Checks whether the image digitizing process has started.
<a href="#">IsFinished</a>	Checks whether an image has been captured and stored completely in the image memory.
<a href="#">Stop</a>	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	<b>Default:</b> 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	<b>Default:</b> 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

<code>started</code>	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

<code>finished</code>	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"> <li>• uc480.Defines.DeviceParameter.DontWait: The method returns immediately. Digitizing the image is completed in the background.</li> <li>• uc480.Defines.DeviceParameter.Wait: The method waits until the image save is complete.</li> <li>• uc480.Defines.DeviceParameter.Force: Digitizing is stopped immediately.</li> </ul>

## 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
<a href="#">GetInfo</a>	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.
-

**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.

The following classes exist:

- [Framerate](#)
- [Gain](#)
- [Shutter](#)
- [Whitebalance](#)

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
<a href="#">GetEnable</a>	Returns the sensor's current auto frame rate setting. (Not all sensors support this feature.)
<a href="#">GetSupported</a>	Returns if setting the auto frame rate is supported.
<a href="#">SetEnable</a>	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
<a href="#">GetDefaultPhotom</a>	Returns the default photometry mode for auto gain control.
<a href="#">GetEnable</a>	Returns the current auto gain setting or white level adjustment of the sensor. (Not all sensors support this feature.)
<a href="#">GetPhotom</a>	Returns the photometry mode for auto gain control.
<a href="#">GetSupported</a>	Returns if auto gain control is supported.
<a href="#">SetEnable</a>	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.)
<a href="#">SetPhotom</a>	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	<b>Defines which fields of view are used for auto gain control:</b> <ul style="list-style-type: none"> <li>• uc480.Defines.Whitebalance.GainPhotomMode.None</li> <li>• uc480.Defines.Whitebalance.GainPhotomMode.CenterWeighted</li> <li>• uc480.Defines.Whitebalance.GainPhotomMode.CenterSpot</li> <li>• uc480.Defines.Whitebalance.GainPhotomMode.Portrait</li> <li>• uc480.Defines.Whitebalance.GainPhotomMode.Landscape</li> </ul>
------	---

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

## Methods

Method	Description
<a href="#">GetDefaultPhotom</a>	Returns the default photometry mode for auto exposure shutter.
<a href="#">GetEnable</a>	Returns the sensor's current auto exposure shutter setting. (Not all sensors support this feature.)
<a href="#">GetPhotom</a>	Returns the photometry mode for auto exposure shutter.
<a href="#">GetSupported</a>	Returns is the setting of the default photometry mode for auto exposure shutter is supported.
<a href="#">SetEnable</a>	Enables/disables the sensor's internal auto exposure shutter function. (Not all sensors support this feature.)
<a href="#">SetPhotom</a>	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"> <li>• uc480.Defines.Whitebalance.ShutterPhotomMode.None</li> <li>• uc480.Defines.Whitebalance.ShutterPhotomMode.CenterWeighted</li> <li>• uc480.Defines.Whitebalance.ShutterPhotomMode.CenterSpot</li> <li>• uc480.Defines.Whitebalance.ShutterPhotomMode.Portrait</li> <li>• uc480.Defines.Whitebalance.ShutterPhotomMode.Landscape</li> </ul>
------	--

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

**Methods**

Method	Description
<a href="#">GetEnable</a>	Returns the sensor's current auto white balance setting. (Not all sensors support this feature.)
<a href="#">GetSupported</a>	Returns if the sensor supports the auto white balance.
<a href="#">SetEnable</a>	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 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 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](#)
- [Whitebalance](#)

The `Framerate` class provides methods for controlling the auto frame feature.

## Methods

Method	Description
<a href="#">GetEnable</a>	Returns the current auto frame rate setting.
<a href="#">GetSupported</a>	Returns if setting the auto frame rate is supported.
<a href="#">SetEnable</a>	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
<a href="#">Get</a>	Returns the number of frames to be skipped during automatic control.
<a href="#">GetRange</a>	Returns the permissible range for the number of frames to be skipped.
<a href="#">Set</a>	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
<a href="#"><code>GetEnable</code></a>	Returns the current auto gain setting or white level adjustment.
<a href="#"><code>GetMax</code></a>	Returns the upper limit for auto gain control.
<a href="#"><code>GetSupported</code></a>	Returns if setting the auto gain is supported.
<a href="#"><code>SetEnable</code></a>	Enables/disables the auto gain control feature or, in case of HDR sensors, the white level adjustment.
<a href="#"><code>SetMax</code></a>	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
<a href="#"><code>Get</code></a>	Returns the hysteresis value for auto exposure shutter and auto gain control.
<a href="#"><code>GetRange</code></a>	Returns the permissible range for the hysteresis value.
<a href="#"><code>Set</code></a>	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

<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 hysteresis value.
<code>max</code>	Returns the maximum hysteresis value.
<code>inc</code>	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
<a href="#">Get</a>	Returns the setpoint for auto gain control/ auto exposure shutter.
<a href="#">GetDefault</a>	Returns the default setpoint for auto gain control and auto exposure shutter.
<a href="#">GetRange</a>	Returns the range for auto gain control and auto exposure shutter.
<a href="#">Set</a>	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"><li>• 0 = black</li><li>• 128 = 50% gray (default)</li><li>• 255 = white</li></ul>
-----------	---

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

**Methods**

Method	Description
<a href="#"><u>GetEnable</u></a>	Returns the current auto exposure shutter setting.
<a href="#"><u>GetMax</u></a>	Returns the upper limit for auto exposure shutter.
<a href="#"><u>GetSupported</u></a>	Returns if setting the auto exposure shutter is supported.
<a href="#"><u>SetEnable</u></a>	Enables/disables the auto exposure shutter control.
<a href="#"><u>SetMax</u></a>	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>	0 = Not supported. 1 = 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>	uc480.Defines.ActivateMode.Enable/1: <b>Enables auto shutter control</b> uc480.Defines.ActivateMode.Disable/0: <b>Disables auto shutter control</b>
----------------------------------	--

**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
<a href="#"><u>Get</u></a>	Returns the speed value for the auto function.
<a href="#"><u>GetDefault</u></a>	Returns the default value for auto speed.
<a href="#"><u>GetRange</u></a>	Returns the range for auto speed.
<a href="#"><u>Set</u></a>	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
<a href="#">GetColorModel</a>	Returns the current color space for the auto white balance.
<a href="#">GetEnable</a>	Returns the current auto white balance setting.
<a href="#">GetReferenceRange</a>	Returns the range for white balance for auto gain control/auto exposure shutter.
<a href="#">GetSupported</a>	Returns if setting the auto white balance is supported.
<a href="#">GetSupportedColorModel</a>	Returns the supported color spaces for auto white balance.
<a href="#">GetSupportedType</a>	Returns the supported types for auto white balance.
<a href="#">GetType</a>	Returns the current set type for auto white balance.
<a href="#">SetColorModel</a>	Sets the color space for auto white balance.
<a href="#">SetEnable</a>	Enables/disables auto white balance.
<a href="#">SetType</a>	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
<a href="#">Get</a>	Returns the number of frames to be skipped during automatic control.
<a href="#">GetRange</a>	Returns the permissible range for the number of frames to be skipped.
<a href="#">Set</a>	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
<a href="#">GetRange</a>	Returns the color gain limits for auto white balance.
<a href="#">SetRange</a>	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>	<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 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
<a href="#">Get</a>	Returns the offset values for the red and blue channels.
<a href="#">GetDefault</a>	Returns the default offset value.
<a href="#">GetRange</a>	Returns the range for offset.
<a href="#">Set</a>	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
<a href="#"><u>Get</u></a>	Returns the speed for auto white balance.
<a href="#"><u>GetDefault</u></a>	Returns the default value for auto white balance speed.
<a href="#"><u>GetRange</u></a>	Returns the range for auto white balance speed.
<a href="#"><u>Set</u></a>	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 colorMod`

### 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"><li>• uc480.Defines.ActivateMode.Disable</li><li>• uc480.Defines.ActivateMode.Enable</li><li>• uc480.Defines.ActivateMode.Once</li></ul>

## 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)
```

## Description

Returns the supported color spaces for auto white balance.

## Parameter

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

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

## 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"><li>• uc480.Defines.ActivateMode.Disable</li><li>• uc480.Defines.ActivateMode.Enable</li><li>• uc480.Defines.ActivateMode.Once</li></ul>

## 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.

#### Methods

Method	Description
<a href="#">Get</a>	Returns the current mode of black level correction.
<a href="#">GetDefault</a>	Returns the default mode for the black level correction.
<a href="#">GetSupported</a>	Returns if the black level correction is supported.
<a href="#">Set</a>	Sets the mode for black level correction.

### 3.1.1.3.1 Offset

The `Offset` class provides methods for configuring the black level offset.

#### Methods

Method	Description
<a href="#">Get</a>	Returns the current set value for offset.
<a href="#">GetDefault</a>	Returns the default value for offset.
<a href="#">GetRange</a>	Returns the range for the offset.
<a href="#">GetSupported</a>	Returns if offset is supported.
<a href="#">Set</a>	Sets the offset for black level correction.

#### Class

[uc480.BlacklevelOffset](#)

#### Accessible

`Camera.BlackLevel.Offset.Get`

#### Syntax

`uc480.BlacklevelOffset.Get(out int offset)`

#### 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

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

#### Class

[uc480.BlacklevelOffset](#)

#### Accessible

`Camera.BlackLevel.Offset.GetDefault`

#### Syntax

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

#### Description

Returns the default value for offset.

#### Parameter

- `offset`: Returns the standard value for the offset.

#### Class

[uc480.BlacklevelOffset](#)

#### Accessible

`Camera.BlackLevel.Offset.GetRange`

#### Syntax

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

## Description

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

## Parameter

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

## Class

[uc480.BlacklevelOffset](#)

### Accessible

Camera.BlackLevel.Offset.GetSupported

## Syntax

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

## Description

Returns if offset is supported.

## Parameter

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

## Class

[uc480.BlacklevelOffset](#)

### Accessible

Camera.BlackLevel.Offset.Set

## Syntax

```
uc480.BlacklevelOffset.Set(int offset)
```

## Description

Sets the offset for black level correction.

## Parameter

- offset: Value to be set for offset.

### 3.1.1.3.2 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: • uc480.Defines.BlackLevelMode.Enable • uc480.Defines.BlackLevelMode.Invalid • uc480.Defines.BlackLevelMode.Offset
------	---

### 3.1.1.3.3 GetDefault

#### Class

[uc480.BlackLevel](#)

#### Accessible

Camera.BlackLevel.GetDefault

#### Syntax

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

#### Description

Returns the default mode.

#### Parameter

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

### 3.1.1.3.4 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.

### 3.1.1.3.5 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"> <li>• uc480.Defines.BlackLevelMode.Enable</li> <li>• uc480.Defines.BlackLevelMode.Invalid</li> <li>• uc480.Defines.BlackLevelMode.Offset</li> </ul>
------	--

### 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 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 the USB 3 DCC3240C camera.



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

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
<a href="#">Get</a>	Returns for color cameras the set mode for the specified color mode.
<a href="#">GetDefault</a>	Returns the default converter for a color mode.
<a href="#">GetSupported</a>	Returns for color cameras all available Bayer conversion modes for the specified color mode.
<a href="#">Set</a>	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 convertMode)
```

### 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 <a href="#">GetSupported()</a> ).
convertMode	Currently selected converter (see <a href="#">GetSupported()</a> ).

## Class

[uc480.ColorConverter](#)

### Accessible

Camera.Color.Converter

### Syntax

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

### Description

Returns the default converter for a color mode.

## Parameter

colorMode	Color mode for which the converter is to be returned (see <a href="#">GetSupported()</a> ).
convertMode	Default converter (see <a href="#">GetSupported()</a> ).

## 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	<b>Color mode for which the converter is to be returned:</b> <ul style="list-style-type: none"> <li>• uc480.Defines.ColorMode.BGR10Packed</li> <li>• uc480.Defines.ColorMode.BGR10Unpacked</li> <li>• uc480.Defines.ColorMode.BGR12Unpacked</li> <li>• uc480.Defines.ColorMode.BGR565Packed</li> <li>• uc480.Defines.ColorMode.BGR5Packed</li> <li>• uc480.Defines.ColorMode.BGR8Packed</li> <li>• uc480.Defines.ColorMode.BGRA12Unpacked</li> <li>• uc480.Defines.ColorMode.BGRA8Packed</li> <li>• uc480.Defines.ColorMode.BGRY8Packed</li> <li>• uc480.Defines.ColorMode.CBYCRYPacked</li> <li>• uc480.Defines.ColorMode.Jpeg</li> <li>• uc480.Defines.ColorMode.Mono12</li> <li>• uc480.Defines.ColorMode.Mono16</li> <li>• uc480.Defines.ColorMode.Mono8</li> <li>• uc480.Defines.ColorMode.RGB10Packed</li> <li>• uc480.Defines.ColorMode.RGB10Unpacked</li> <li>• uc480.Defines.ColorMode.RGB12Unpacked</li> <li>• uc480.Defines.ColorMode.RGB8Packed</li> <li>• uc480.Defines.ColorMode.RGB8Planar</li> <li>• uc480.Defines.ColorMode.RGBA12Unpacked</li> <li>• uc480.Defines.ColorMode.RGBA8Packed</li> <li>• uc480.Defines.ColorMode.RGBY8Packed</li> <li>• uc480.Defines.ColorMode.SensorRaw12</li> <li>• uc480.Defines.ColorMode.SensorRaw16</li> <li>• uc480.Defines.ColorMode.SensorRaw8</li> <li>• uc480.Defines.ColorMode.UYVYBayerPacked</li> <li>• uc480.Defines.ColorMode.UYVYMonoPacked</li> </ul>
-----------	---

	<ul style="list-style-type: none"> <li>• uc480.Defines.ColorMode.UYVYPacked</li> <li>• uc480.Defines.ColorMode.PreferPackedSourceFormat</li> </ul> <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"> <li>• uc480.Defines.ColorConvertMode.None: No conversion</li> <li>• uc480.Defines.ColorConvertMode.Software: Only for monochrome cameras, if you want to add a gamma</li> <li>• uc480.Defines.ColorConvertMode.Software3X3: Software conversion using the standard filter mask (default)</li> <li>• uc480.Defines.ColorConvertMode.Software5X5: Software conversion using a large filter mask</li> <li>• uc480.Defines.ColorConvertMode.Hardware3X3: Hardware conversion using the standard filter mask (GigE uc480 only)</li> <li>• uc480.Defines.ColorConvertMode.OpenCL3X3: Software conversion using the standard filter mask, but conversion is done on the graphic board</li> <li>• uc480.Defines.ColorConvertMode.OpenCL5X5: Hardware conversion using the standard filter mask, but conversion is done on the graphic board</li> <li>• uc480.Defines.ColorConvertMode.Jpeg: JPEG compression, only for XS cameras</li> </ul>

## 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 <a href="#">GetSupported()</a> ).
convertMode	Convert mode to be set (see <a href="#">GetSupported()</a> ).

### 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.



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

#### Methods

Method	Description
<a href="#">Get</a>	Returns the current color correction mode.
<a href="#">GetDefault</a>	Returns the default color correction mode.
<a href="#">GetSupported</a>	Returns all supported color correction modes.
<a href="#">Set</a>	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 <a href="#">GetSupported()</a> ).
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	Returns the supported values linked by a logical OR: <ul style="list-style-type: none"><li>• <code>uc480.Defines.ColorCorrectionMode.Disable</code>: <b>Disable color correction.</b></li><li>• <code>uc480.Defines.ColorCorrectionMode.Enable</code>: <b>Enable color correction.</b></li><li>• <code>uc480.Defines.ColorCorrectionMode.EnableNormal</code>: <b>Simple color correction.</b></li><li>• <code>uc480.Defines.ColorCorrectionMode.EnableBG40Enhanced</code>: Color correction for cameras with optical IR filter glasses of the BG40 type.</li><li>• <code>uc480.Defines.ColorCorrectionMode.EnableHqEnhanced</code>: Color correction for cameras with optical IR filter glasses of the HQ type.</li><li>• <code>uc480.Defines.ColorCorrectionMode.IrAutomatic</code>: When used for monochrome cameras, the system returns 0.</li></ul>
----------------	---

**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 <a href="#">GetSupported()</a> ).
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
<a href="#">Get</a>	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>s32Col1</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:

- [RgbModel](#)

## Methods

Method	Description
<a href="#">Get</a>	Returns the set color temperature.
<a href="#">GetDefault</a>	Returns the default value for the color temperature.
<a href="#">GetRange</a>	Returns the range of the color temperature.
<a href="#">Set</a>	Sets a color temperature.

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
<a href="#">Get</a>	Returns the set color space.
<a href="#">GetDefault</a>	Returns the default color space.
<a href="#">GetSupported</a>	Returns the supported color spaces.
<a href="#">Set</a>	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

<code>modelMode</code>	<p>Returns the supported color spaces.</p> <ul style="list-style-type: none"> <li>• <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.</li> <li>• <code>uc480.Defines.ColorTemperatureRgbMode.CIE_RGB_E</code>: CIE-RGB color space with standard illumination E</li> <li>• <code>uc480.Defines.ColorTemperatureRgbMode.ECI_RGB_D50</code>: ECI-RGB color space with a white point of 5000 kelvins (warm light)</li> <li>• <code>uc480.Defines.ColorTemperatureRgbMode.SRGB_D50</code>: sRGB (standard RGB) color space with a white point of 5000 kelvins (warm light)</li> <li>• <code>uc480.Defines.ColorTemperatureRgbMode.SRGB_D65</code>: sRGB (standard RGB) color space with a white point of 6500 kelvins (mid daylight)</li> </ul>
------------------------	--

## 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
<a href="#">GetCameraID</a>	Returns the current camera ID.
<a href="#">GetDeviceID</a>	Returns the current device ID.
<a href="#">GetEnableAutoExit</a>	Returns the state of the automatic closing of the camera handle.
<a href="#">SetCameraID</a>	Sets a unique camera ID to a camera.
<a href="#">SetEnableAutoExit</a>	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 DCC1240 and DCC3240 models:
  - Set line scan mode.
  - Toggle between rolling and global shutter mode.
  - Control the Log mode
  - Set AOI merge mode

The following classes exist:

- [AOIMerge](#)
- [Linescan](#)
- [Log](#)
- [ShutterMode](#)

The `ExternalInterface` class provides methods for activating the external interface to query the timestamp via the I<sup>2</sup>C bus. For this purpose, you have to configure the two GPIOs accordingly with [Gpio](#).

 Note: This function is only supported by the DCC3260 camera.

### Methods

Method	Description
Get	Returns the current activated external interface and its configuration.
GetSupported	Returns if an external interface is supported.
Set	Enables the external interface and sets its configuration.

### Class

`uc480.DeviceFeatureExternalInterface`

#### Accessible

`Camera.Device.Feature.ExternalInterface.Get`

#### Syntax

```
uc480.DeviceFeatureExternalInterface.Get(out uc480.Defines.ExternalInterfaceType interfaceType)
uc480.DeviceFeatureExternalInterface.Get(out uc480.Types.ExternalInterfaceConfigurationI2C configuration)
```

#### Description

Returns the current activated external interface and its configuration.

## Parameter

interfaceType	<p>The use of the external interface is enabled/disabled:</p> <ul style="list-style-type: none"> <li>• uc480.Defines.ExternalInterfaceType.I2C The external I<sup>C</sup> interface is enabled.</li> <li>• uc480.Defines.ExternalInterfaceType.NoOne The external interface is disabled.</li> </ul>
---------------	---

## Example

```
// Returns the current type
uc480.Defines.ExternalInterfaceType GetType;
statusRet = Camera.Device.Feature.ExternalInterface.Get(out GetType);

// Returns the current configuration
uc480.Types.ExternalInterfaceConfigurationI2C GetConfiguration;
statusRet = Camera.Device.Feature.ExternalInterface.Get(out GetConfiguration);
```

## Class

uc480.DeviceFeatureExternalInterface

### Accessible

Camera.Device.Feature.ExternalInterface.GetSupported

### Syntax

```
uc480.DeviceFeatureExternalInterface.GetSupported(out uc480.Defines.ExternalInterfaceType supported)
uc480.DeviceFeatureExternalInterface.GetSupported(out bool supported)
```

### Description

Returns if an external interface is supported.

## Parameter

supported	<ul style="list-style-type: none"> <li>• uc480.Defines.ExternalInterfaceType.I2C The external I<sup>C</sup> interface is supported.</li> <li>• uc480.Defines.ExternalInterfaceType.NoOne The external interface is not supported.</li> </ul> <ul style="list-style-type: none"> <li>• 1 = The use of the external I<sup>C</sup> interface is supported.</li> <li>• 0 = The use of the external interface is not supported.</li> </ul>
-----------	---

## Class

uc480.DeviceFeatureExternalInterface

### Accessible

Camera.Device.Feature.ExternalInterface.Set

### Syntax

```
uc480.DeviceFeatureExternalInterface.Set(uc480.Defines.ExternalInterfaceType interfaceType)
uc480.DeviceFeatureExternalInterface.Set(uc480.Types.ExternalInterfaceConfigurationI2C configuration)
```

## Description

Enables the external interface and sets its configuration.

## Parameter

interfaceType	Enables/disable the use of the external interface: <ul style="list-style-type: none"> <li>• uc480.Defines.ExternalInterfaceType.I<sup>2</sup>C Enables the external I<sup>2</sup>C interface.</li> <li>• uc480.Defines.ExternalInterfaceType.None Disables the external interface.</li> </ul>
---------------	--

## Example

```
// Set configuration
uc480.Types.ExternalInterfaceConfigurationI2C SetConfiguration;
SetConfiguration.ackPolling = 1;
SetConfiguration.dataSelection = uc480.Defines.ExternalInterfaceData.Timestamp_HighByte;
SetConfiguration.registerAddress = 0x87;
SetConfiguration.registerType = uc480.Defines.ExternalInterfaceRegisterType.Bit_16;
SetConfiguration.sendEvent = uc480.Defines.ExternalInterfaceEvent.FallingVsync;
SetConfiguration.slaveAddress = 0x88;

statusRet = Camera.Device.Feature.ExternalInterface.Set(SetConfiguration);
```

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 DCC3240 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 DCC3240 Models

The following classes exist:

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

## Methods

Method	Description
<a href="#">Get</a>	Returns the current AOI merge mode.
<a href="#">GetDefault</a>	Returns the default AOI merge mode.
<a href="#">GetEnable</a>	Returns if the AOI merge mode is enabled.
<a href="#">GetSupported</a>	Returns if the AOI merge mode is supported.
<a href="#">Set</a>	Sets a new AOI merge mode.
<a href="#">SetEnable</a>	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 DCC3240 Models

## Methods

Method	Description
<a href="#">Get</a>	Returns the position of the additional used sensor line(s).
<a href="#">GetDefault</a>	Returns the default value for the additional AOI merge position.
<a href="#">GetRange</a>	Returns the range for the additional AOI merge position.
<a href="#">Set</a>	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 inc)
```

## 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 for m



Note: This feature is currently only supported by the DCC3240 Models

## Methods

Method	Description
<a href="#">Get</a>	Returns the current set AOI merge height.
<a href="#">GetDefault</a>	Returns the default value of the AOI merge height (2 for color, 1 for monochrome).
<a href="#">GetList</a>	Returns the list with the allowed AOI merge heights.
<a href="#">GetNumber</a>	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.
<a href="#">Set</a>	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 DCC3240 Models

## Methods

Method	Description
<a href="#">Get</a>	Returns the position of the two sensor lines.
<a href="#">GetDefault</a>	Returns the default value for the AOI merge position (default = 0, i.e. the two top lines).
<a href="#">GetRange</a>	Returns the setting range for the AOI merge position.
<a href="#">GetSet</a>	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: <b>AOI merge mode is off</b>
	uc480.Defines.VerticalAoiMergeMode.FallingGpio1: <b>Sensor is triggered by falling edge on GPIO 1</b>
	uc480.Defines.VerticalAoiMergeMode.FallingGpio2: <b>Sensor is triggered by falling edge on GPIO 2</b>
	uc480.Defines.VerticalAoiMergeMode.RisingGpio1: <b>Sensor is triggered by rising edge on GPIO 1</b>
	uc480.Defines.VerticalAoiMergeMode.RisingGpio2: <b>Sensor is triggered by rising edge on GPIO 2</b>
	uc480.Defines.VerticalAoiMergeMode.Freerun: <b>Sensor is operated with maximum speed.</b>
	uc480.Defines.VerticalAoiMergeMode.Software: <b>Sensor is triggered via software</b>

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

- On DCC1240 and DCC3240 models: Set line scan mode.

## Methods

Method	Description
<a href="#"><u>Get</u></a>	Returns the currently set line scan mode.
<a href="#"><u>GetNumber</u></a>	Returns the scan line used for the line scan mode.
<a href="#"><u>GetSupported</u></a>	Returns the supported line scan modes.
<a href="#"><u>IsSupported</u></a>	Returns if the passed line scan mode is supported.
<a href="#"><u>Set</u></a>	Sets the line scan mode.
<a href="#"><u>SetNumber</u></a>	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	<p>uc480.Defines.LinescanMode.Fast: Fast line scan mode is supported</p>
	<p>uc480.Defines.LinescanMode.Number: Line scan mode with selectable line number is supported</p>

## 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	<p>uc480.Defines.LinescanMode.Fast: Set fast line scan mode</p>
	<p>uc480.Defines.LinescanMode.Number: Set line scan mode with selectable line number</p>

## 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 UIDCC1240 and DCC3240 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 DCC1240 and DCC3240 models.

The following classes and methods exist:

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

### Methods

Method	Description
<a href="#">Get</a>	Returns the current Log mode.
<a href="#">GetDefault</a>	Returns the default settings for the Log mode.
<a href="#">GetSupported</a>	Returns if the Log mode is supported.
<a href="#">Set</a>	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 DCC1240 and DCC3240 models.

## Methods

Method	Description
<a href="#">Get</a>	Returns the current manual gain of the Log mode.
<a href="#">GetDefault</a>	Returns the default settings for the manual gain for the Log mode.
<a href="#">GetRange</a>	Returns the range for the manual gain of the Log mode.
<a href="#">Set</a>	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

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.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 DCC1240 and DCC3240 models.

## Methods

Method	Description
<a href="#">Get</a>	Returns the current manual value of the Log mode.
<a href="#">GetDefault</a>	Returns the default settings for the manual value of the Log mode.
<a href="#">GetRange</a>	Returns the range of the manual value of the Log mode.
<a href="#">Set</a>	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	<code>uc480.Defines.LogMode.Factory</code> : Default setting for the Log mode
	<code>uc480.Defines.LogMode.Manual</code> : Manual Log mode. In this case the classes <a href="#">LogManualGain</a> and <a href="#">LogManualValue</a> are used for further setting.
	<code>uc480.Defines.LogMode.Off</code> : 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

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

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 in trigger mode (see also [Trigger](#)).

## Methods

Method	Description
<a href="#">Get</a>	Returns the shutter mode.
<a href="#">GetSupported</a>	Returns the supported shutter modes.
<a href="#">IsSupported</a>	Returns if the passed shutter mode is supported.
<a href="#">Set</a>	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	uc480.Defines.Shuttermode.Rolling: <b>Rolling shutter mode</b>
	uc480.Defines.Shuttermode.RollingGlobalStart: <b>Rolling shutter with global start</b>
	uc480.Defines.Shuttermode.Global: <b>Global shutter mode</b>
	uc480.Defines.Shuttermode.GlobalAlternativeTiming: <b>Global shutter mode with alternative timing (DCC1240 and DCC3240 only)</b>

## 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\(\)](#))

### 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

bEnable	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

<code>bEnable</code>	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
<a href="#">DisableScaling</a>	Disables real-time scaling.
<a href="#">EnableImageScaling</a>	Enables real-time scaling of the image to the size of the display window. The overlay is not scaled.
<a href="#">EnableScaling</a>	Enables real-time scaling of the image to the size of the display window. The overlay is scaled together with the camera image.
<a href="#">GetStealFormat</a>	Returns the color format setting for the steal function.
<a href="#">GetSupported</a>	Returns the supported display modes.
<a href="#"> GetUserSyncPosRange</a>	Returns the minimum and maximum row position for <a href="#">SetUserSync()</a> .
<a href="#">SetScaling</a>	Enables/disables real-time scaling of the image to the size of the display window.
<a href="#">SetStealFormat</a>	Defines the color format for the steal function.
<a href="#">SetUserSync</a>	Enables synchronization of the image display with a monitor pixel row specified by the user.
<a href="#">SetWindowHandle</a>	Sets a new window handle for image output in Direct3D.
<a href="#">StealNextFrame</a>	Copies the next image to the active user memory (steal function).
<a href="#">Update</a>	Redraws the current image and overlay.
<a href="#">VsyncAuto</a>	Enables synchronization of the image display with the monitor's image rendering.
<a href="#">VsyncOff</a>	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
<a href="#">Clear</a>	Deletes the data of the overlay area by filling it with black color.
<a href="#">DisableSemiTransparent</a>	Disables the semi-transparent display of the overlay area.
<a href="#">EnableSemiTransparent</a>	Enables a semi-transparent display of the overlay area.
<a href="#">GetDC</a>	Returns the device context (DC) handle to the overlay area of the graphics card.
<a href="#">GetGraphics</a>	Returns the overlay graphics.
<a href="#">GetKeyColor</a>	Returns the RGB values of the current key color (default: black).
<a href="#">GetMaxSize</a>	Returns the width and height of the maximum overlay area supported by the graphics card.
<a href="#">GetSize</a>	Returns the size of the overlay area.
<a href="#">Hide</a>	Disables overlay display.
<a href="#">LoadImage</a>	Loads a bitmap image (*.BMP file) into the overlay area.
<a href="#">ReleaseDC</a>	Releases the device context (DC) handle and updates the overlay data.
<a href="#">SetGraphics</a>	Releases the device context (DC) handle.
<a href="#">SetKeyColor</a>	Defines the RGB values of the key color.
<a href="#">SetPosition</a>	Defines the position of the overlay area.
<a href="#">SetSemiTransparent</a>	Enables/disabled the semi-transparent display of the overlay area.
<a href="#">SetSize</a>	Defines the size of the overlay area (default: current camera image size).
<a href="#">SetVisible</a>	Sets the overlay to visible.
<a href="#">Show</a>	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.

#### Paramete

- 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
<a href="#">Render</a>	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
<a href="#">GetEnable</a>	Returns if the auto rendering is enabled.
<a href="#">GetMode</a>	Returns the current render mode.
<a href="#">GetModeDefault</a>	Returns the default render mode.
<a href="#">GetWindow</a>	Returns a pointer to a window handle.
<a href="#">SetEnable</a>	Enables/disables the auto render.
<a href="#">SetMode</a>	Sets the render mode.
<a href="#">SetWindow</a>	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

enable	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
<a href="#">Get</a>	Returns the current display mode.
<a href="#">Set</a>	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><b>Mode to be set:</b></p> <ul style="list-style-type: none"> <li>• uc480.Defines.DisplayMode.DIB: Captures an image in system memory (RAM). Using <a href="#">Render()</a>, you can define the image display (default).</li> <li>• uc480.Defines.DisplayMode.Direct3D: Image display in Direct3D mode</li> <li>• uc480.Defines.DisplayMode.Direct3D   uc480.Defines.DisplayMode.Mono: Monochrome image display in Direct3D mode</li> <li>• uc480.Defines.DisplayMode.Direct3D   uc480.Defines.DisplayMode.Bayer: Raw Bayer format image display in Direct3D mode</li> <li>• uc480.Defines.DisplayMode.OpenGL: Image display in OpenGL mode</li> <li>• uc480.Defines.DisplayMode.OpenGL   uc480.Defines.DisplayMode.Mono: Monochrome image display in OpenGL mode</li> <li>• uc480.Defines.DisplayMode.OpenGL   uc480.Defines.DisplayMode.Bayer: Raw Bayer format image display in OpenGL mode</li> </ul>
------	--

### 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
<a href="#">Set</a>	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"> <li>• uc480.Defines.DisplayRenderMode.Normal: The image is rendered normally. It will be displayed in 1:1 scale as stored in the image memory.</li> <li>• uc480.Defines.DisplayRenderMode.FitToWindow: The image size is adjusted to fit the output window.</li> <li>• uc480.Defines.DisplayRenderMode.DownScale_1_2: Displays the image at 50 % of its original size.</li> <li>• uc480.Defines.DisplayRenderMode.PlanarColorBlue: Displays only the blue channel of planar color format.</li> <li>• uc480.Defines.DisplayRenderMode.PlanarColorGreen: Displays only the green channel of planar color format.</li> <li>• uc480.Defines.DisplayRenderMode.PlanarColorRed: Displays only the red channel of planar color format.</li> <li>• uc480.Defines.DisplayRenderMode.PlanarMonoBlue: Displays only the blue channel as monochrome image.</li> <li>• uc480.Defines.DisplayRenderMode.PlanarMonoGreen: Displays only the green channel as monochrome image.</li> <li>• uc480.Defines.DisplayRenderMode.PlanarMonoRed: Displays only the red channel as monochrome image.</li> </ul> <p>The following option can be linked by a logical OR using the <code>s32Mode</code> parameter:</p> <ul style="list-style-type: none"> <li>• uc480.Defines.DisplayRenderMode.MirrorUpDown: Mirrors the displayed image along the horizontal axis.</li> </ul>

### 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
<a href="#"><u>Get</u></a>	Returns the current set edge enhancement.
<a href="#"><u>GetDefault</u></a>	Returns the default value of the edge enhancement.
<a href="#"><u>GetRange</u></a>	Returns the range (minimum, maximum and increment) of the edge enhancement.
<a href="#"><u>Set</u></a>	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	<b>Minimum:</b> Returns the minimum value.
	<b>Maximum:</b> Returns the maximum value.
	<b>Increment:</b> 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
<a href="#">Read</a>	Reads the content of the EEPROM memory.
<a href="#">Write</a>	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 Gain

The `Gain` class provides methods for controlling the gain factor and the sensor gain channels. The following class exists:

- [Hardware](#)

### 3.1.1.10.1 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
<a href="#">GetSupported</a>	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
<a href="#">GetEnable</a>	Returns the current state of the gain boost function.
<a href="#">GetSupported</a>	Indicates whether the camera supports a gain boost feature or not.
<a href="#">SetEnable</a>	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
<a href="#">Blue</a>	Converts the index value for the blue gain factor.
<a href="#">Green</a>	Converts the index value for the green gain factor.
<a href="#">Master</a>	Converts the index value for the master gain factor.
<a href="#">Red</a>	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.

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
<a href="#">GetDefaultBlue</a>	Returns the default blue gain factor.
<a href="#">GetDefaultGreen</a>	Returns the default green gain factor.
<a href="#">GetDefaultMaster</a>	Returns the default master gain factor.
<a href="#">GetDefaultRed</a>	Returns the default red gain factor.
<a href="#">GetBlue</a>	Returns the blue channel gain factor.
<a href="#">GetGreen</a>	Returns the green channel gain factor.
<a href="#">GetMaster</a>	Returns the master channel gain factor.
<a href="#">GetRed</a>	Returns the red channel gain factor.
<a href="#">SetBlue</a>	Sets the blue channel gain factor (0...100).
<a href="#">SetGreen</a>	Sets the green channel gain factor (0...100).
<a href="#">SetMaster</a>	Sets the master channel gain factor (0...100).
<a href="#">SetRed</a>	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.

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
<a href="#">GetDefaultBlue</a>	Returns the default blue channel gain factor.
<a href="#">GetDefaultGreen</a>	Returns the default green channel gain factor.
<a href="#">GetDefaultMaster</a>	Returns the default master channel gain factor.
<a href="#">GetDefaultRed</a>	Returns the default red channel gain factor.
<a href="#">GetBlue</a>	Returns the blue channel gain factor.
<a href="#">GetGreen</a>	Returns the green channel gain factor.
<a href="#">GetMaster</a>	Returns the master channel gain factor.
<a href="#">GetRed</a>	Returns the red channel gain factor.
<a href="#">SetBlue</a>	Sets the blue channel gain factor (0...100).
<a href="#">SetGreen</a>	Sets the green channel gain factor (0...100).
<a href="#">SetMaster</a>	Sets the master channel gain factor (0...100).
<a href="#">SetRed</a>	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

## 3.1.11 Gamma

The `Gamma` class provides methods for controlling the hardware gamma or the digital software gamma correction. The following class and methods exist:

- [Software](#)

### 3.1.11.1 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
<a href="#">Get</a>	Returns the current set value for the digital gamma correction.
<a href="#">GetDefault</a>	Returns the default value for the digital gamma correction.
<a href="#">GetRange</a>	Returns the range of the digital gamma correction.
<a href="#">Set</a>	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)

### 3.1.1.12 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
<a href="#">DisableSensorCorrection</a>	Disables hot pixel correction.
<a href="#">EnableSensorCorrection</a>	Enables hot pixel correction.
<a href="#">GetCorrectionMode</a>	Returns the currently set hot pixel correction mode.
<a href="#">GetMergedList</a>	Returns the number of hot pixels in a merged list.
<a href="#">GetSupportedCorrectionMode</a>	Returns the supported hot pixel correction modes.
<a href="#">SetEnableSensorCorrection</a>	Enables/disables the sensor's own hot pixel correction function.

### 3.1.1.12.1 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
<a href="#">ClearList</a>	Deletes the user-defined hot pixel list from the camera EEPROM.
<a href="#">GetList</a>	Returns the user-defined hot pixel list stored in the camera EEPROM.
<a href="#">GetListExists</a>	Indicates whether the user-defined hot pixel list exists in the camera EEPROM.
<a href="#">GetListMaxNumber</a>	Returns the maximum number of hot pixels that the user can store in the camera EEPROM.
<a href="#">GetListNumber</a>	Returns the number of hot pixels in the user-defined hot pixel list stored in the camera EEPROM.
<a href="#">SetEnable</a>	Enables hot pixel correction using the hot pixel list(s) stored in the camera EEPROM.
<a href="#">SetList</a>	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.HotpixelCameraList](#)

### Accessible

Camera.Hotpixel.Camera

### Syntax

`uc480.HotpixelCameraList.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.HotpixelCameraList](#)

### Accessible

Camera.Hotpixel.Camera

### Syntax

`uc480.HotpixelCameraList.SetEnable(bool bEnable)`

### Description

Enables hot pixel correction using the hot pixel list(s) stored in the camera EEPROM.

### Parameter

<code>bEnable</code>	1 = Enable hot pixel correction
	0 = Disable hot pixel correction

### Class

[uc480.HotpixelCameraList](#)

### Accessible

Camera.Hotpixel.Camera

### Syntax

`uc480.HotpixelCameraList.SetList(uc480.Types.Point<short>[] List)`  
`uc480.HotpixelCameraList.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.

### 3.1.1.12.2 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
<a href="#"><u>GetList</u></a>	Returns the factory-set hot pixel list.
<a href="#"><u>GetListExists</u></a>	Indicates whether the factory-set hot pixel list exists.

#### Class

[uc480.HotpixelsFactoryList](#)

#### Accessible

Camera.Hotpixels.Factory

#### Syntax

```
uc480.HotpixelsFactoryList.GetList(out uc480.Types.Point<short>[] List)
uc480.HotpixelsFactoryList.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.HotpixelsFactoryList](#)

#### Accessible

Camera.Hotpixels.Factory

#### Syntax

```
uc480.HotpixelsFactoryList.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
---------	--

### 3.1.1.12.3 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
<a href="#"><u>GetList</u></a>	Returns the user-defined hot pixel list stored in the computer.
<a href="#"><u>GetListExists</u></a>	Indicates whether the user-defined hot pixel list exists in the computer.
<a href="#"><u>LoadList</u></a>	Loads the user-defined hot pixel list from a file.
<a href="#"><u>SaveList</u></a>	Saves the user-defined hot pixel list to a file.
<a href="#"><u>SetEnable</u></a>	Enables hot pixel correction using the user's hot pixel list stored in the computer.
<a href="#"><u>SetList</u></a>	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.Hotpixelsoftware

**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.Hotpixelsoftware

**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.Hotpixelsoftware

**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.

### 3.1.1.12.4 DisableSensorCorrection

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

Camera.Hotpixel

**Syntax**

```
uc480.Hotpixel.DisableSensorCorrection()
```

**Description**

Disables hot pixel correction.

**Parameter**

None

### 3.1.1.12.5 EnableSensorCorrection

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

Camera.Hotpixel

**Syntax**

```
uc480.Hotpixel.EnableSensorCorrection()
```

**Description**

Enables hot pixel correction.

**Parameter**

None

### 3.1.1.12.6 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	<p>Returns the current hot pixel correction mode:</p> <ul style="list-style-type: none"> <li>• uc480.Defines.CorrectionMode.CameraCorrection: Hot pixel correction is possible via the hot pixel list in the camera EEPROM.</li> <li>• uc480.Defines.CorrectionMode.SoftwareCorrection: Hot pixel correction is possible via the user-defined hot pixel list.</li> <li>• uc480.Defines.CorrectionMode.SensorCorrection: Hot pixel correction is possible via the sensor-internal hot pixel correction.</li> <li>• uc480.Defines.CorrectionMode.Disabled: Hot pixel correction is disabled.</li> </ul>
------	---

### 3.1.1.12.7 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.

### 3.1.1.12.8 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"> <li>• uc480.Defines.CorrectMode.CameraCorrection: Hot pixel correction is possible via the hot pixel list in the camera EEPROM.</li> <li>• uc480.Defines.CorrectMode.SoftwareCorrection: Hot pixel correction is possible via the user-defined hot pixel list.</li> <li>• uc480.Defines.CorrectMode.SensorCorrection: Hot pixel correction is possible via the sensor-internal hot pixel correction.</li> <li>• uc480.Defines.CorrectMode.Disabled: Hot pixel correction is disabled.</li> </ul>
------	--

**3.1.1.12.9 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

**3.1.1.13 I2C**

The `I2C` class provides methods for writing and reading over the I<sup>2</sup>C bus of some uc480 cameras.

**Methods**

Method	Description
<a href="#">Read</a>	Reads data over the I <sup>2</sup> C bus of some uc480 board level cameras.
<a href="#">Write</a>	Writes data over the I <sup>2</sup> C bus of some uc480 board level cameras.

### 3.1.1.13.1 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<sup>C</sup> bus of some uc480 board level cameras.

The uc480 processes I<sup>C</sup> 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

For information on the signals applied to the I<sup>C</sup> 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

### 3.1.1.13.2 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<sup>C</sup> bus of a board level camera.

The uc480 processes I<sup>C</sup> 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

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

<code>s32DeviceAddr</code>	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 written

### 3.1.1.14 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
<a href="#">Convert</a>	Converts a raw Bayer image in the desired format.
<a href="#">GetHistogram</a>	Computes the histogram of the submitted image.
<a href="#">GetValues</a>	Returns the RGB values of a specific position of the image.
<a href="#">Load</a>	Loads an image file (BMP, JPG, PNG).
<a href="#">Save</a>	Save an image to a file.

### 3.1.1.14.1 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
<a href="#">Inquire</a>	Returns information of the set AOI, e.g. the sharpness.
<a href="#">SetAOI</a>	Sets an AOI in which the sharpness is measured. In the image are up to 5 AOIs possible. These AOIs can also overlap.
<a href="#">SetPreset</a>	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)

### 3.1.1.14.2 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](#)

### 3.1.1.14.3 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 <a href="#">PixelFormat</a> 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.

### 3.1.1.14.4 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 <a href="#">GetSupported()</a> ).
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

### 3.1.1.14.5 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 <a href="#">Free()</a> .

### 3.1.1.14.6 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	<b>File format to be saved:</b> <ul style="list-style-type: none"><li>• <code>System.Drawing.Imaging.ImageFormat.Bmp</code></li><li>• <code>System.Drawing.Imaging.ImageFormat.Jpeg</code></li><li>• <code>System.Drawing.Imaging.ImageFormat.Png</code></li></ul>
s32MemID	<b>Image memory ID</b>
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"><li>• 100 = maximum quality, that means no compression</li><li>• If the parameter is set to 0, the default value of 75 is used. For BMP the parameter is ignored.</li></ul>

### 3.1.15 Information

The `Information` class provides methods for querying information about uc480 cameras and other hardware or errors while image acquisition etc.

#### Methods

Method	Description
<a href="#"><code>GetAvgBandwidth</code></a>	Returns the average bandwidth.
<a href="#"><code>GetBusSpeed</code></a>	Queries whether a camera is connected to a USB 2.0 or USB 3.0 host controller.
<a href="#"><code>GetCameraInfo</code></a>	Returns the data hard-coded in the EEPROM.
<a href="#"><code>GetCameraStatus</code></a>	Returns various status information and settings.
<a href="#"><code>GetCaptureStatus</code></a>	Returns information on errors that occurred during an image capture.
<a href="#"><code>GetDeviceInfo</code></a>	Returns information about connected USB 3 and GigE uc480 cameras.
<a href="#"><code>GetEnableErrorReport</code></a>	Get the status (enabled/disabled) for error event logging.
<a href="#"><code>GetImageInfo</code></a>	Provides additional information on the images you take.
<a href="#"><code>GetLastError</code></a>	Returns the last error that occurred and returns the associated error code and message.
<a href="#"><code>GetPeakBandwidth</code></a>	Returns the peak bandwidth.
<a href="#"><code>GetSensorInfo</code></a>	Returns information about the sensor type used in the camera.
<a href="#"><code>GetUsedBandwidth</code></a>	Returns the bus bandwidth (in MByte/s) currently used by all initialized or selected cameras.
<a href="#"><code>ResetCaptureStatus</code></a>	Resets the capture status infomation.
<a href="#"><code>SetCameraStatus</code></a>	Sets various status information and camera settings.
<a href="#"><code>SetEnableErrorReport</code></a>	Enables/disables error event logging.

#### 3.1.15.1 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.

**3.1.1.15.2 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.

**3.1.1.15.3 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](#)

### 3.1.1.15.4 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"> <li>uc480.Defines.CameraStatus.FIFOOverflowCount: Number of FIFO overruns. Is increased if image data gets lost because the USB bus is congested.</li> <li>uc480.Defines.CameraStatus.SequenceCount: Returns the sequence count. For <a href="#">Capture()</a>, this parameter is set to 0. Each time the sequence buffer (image counter) changes, the counter is increased by 1.</li> <li>uc480.Defines.CameraStatus.SequenceSize: Returns the number of sequence buffers.</li> <li>uc480.Defines.CameraStatus.ExternalTriggerEventCount: Returns the camera's internal count of external trigger events.</li> <li>uc480.Defines.CameraStatus.TriggerMissed: Returns the number of unprocessed trigger signals. Is reset to 0 after each call.</li> <li>uc480.Defines.CameraStatus.LastCaptureError: Returns the last image capture error. For a list of all possible error events, see <a href="#">GetCaptureStatus()</a>.</li> <li>uc480.Defines.CameraStatus.ParameterExist: Indicates whether a parameter set is present on the camera EEPROM (read-only). See also <a href="#">Parameter</a>.</li> <li>uc480.Defines.CameraStatus.ParametersSet1: Indicates whether parameter set 1 including camera settings is present on the camera (read-only). See also <a href="#">Parameter</a>.</li> <li>uc480.Defines.CameraStatus.ParametersSet2: Indicates whether parameter set 2 including camera settings is present on the camera (read-only). See also <a href="#">Parameter</a>.</li> <li>uc480.Defines.CameraStatus.Standby: Sets the camera to standby mode.</li> <li>uc480.Defines.CameraStatus.StandbySupported: Queries whether the camera supports standby mode (read-only).</li> </ul>
s32Status	Returns the information specified by cameraStatus.

### 3.1.1.15.5 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](#)

### 3.1.15.6 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...](#)

### 3.1.15.7 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.
---------	---

### 3.1.1.15.8 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 <a href="#">uc480.Types.ImageInfo</a>

### 3.1.1.15.9 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.

### 3.1.1.15.10 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.

### 3.1.1.15.11 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](#)

**3.1.15.12 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.

**3.1.15.13 ResetCaptureStatus****Class**[uc480.Information](#)**Accessible**

Camera.Information

**Syntax**`uc480.Information.ResetCaptureStatus()`**Description**

Resets the capture status infomation.

**Parameter**

None

**3.1.15.14 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 <a href="#">GetCameraStatus()</a>
s32Status	Sets the information specified by cameraStatus.

**3.1.1.15 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

**3.1.1.16 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](#)

**3.1.1.16.1 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
<a href="#">ApplyGlobalParams</a>	Returns the parameters for the global exposure window and sets them as flash parameters.
<a href="#">GetAutoFreerunDefault</a>	Returns the default auto flash setting in freerun mode.
<a href="#">GetAutoFreerunEnable</a>	Returns the current auto flash setting in freerun mode.
<a href="#">GetDelayRange</a>	Returns the range for flash delay.
<a href="#">GetDurationRange</a>	Returns the range for flash duration.
<a href="#">GetGlobalParams</a>	Returns the parameters for the global exposure window.
<a href="#">GetMode</a>	Returns the current flash mode.
<a href="#">GetParams</a>	Returns the current values for flash delay and duration.
<a href="#">GetSupported</a>	Returns the GPIOs which can be used for flash output.
<a href="#">SetAutoFreerunEnable</a>	Enables/disables the auto flash in freerun mode.
<a href="#">SetMode</a>	Sets the flash mode.
<a href="#">SetParams</a>	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.GetGPIOParamsMin

**Syntax**

```
uc480.IOFlash.GetGPIOParamsMin(out uint u32Delay, out uint u32Duration)
```

**Description**

Returns the minimum possible values for flash delay and flash duration.

**Parameter**

u32GlobalDelay	Returns the minimum flash delay.
u32GlobalDuration	Returns the minimum flash 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"> <li>• uc480.Defines.IO.FlashMode.Off: Disables the digital output.</li> <li>• uc480.Defines.IO.FlashMode.TriggerLowActive: Enables the flash strobe in trigger mode. The digital output is set to low level for the flash duration.</li> <li>• uc480.Defines.IO.FlashMode.TriggerHighActive: Enables the flash strobe in trigger mode. The digital output is set to high level for the flash duration.</li> <li>• uc480.Defines.IO.FlashMode.ConstantHigh: Statically sets the digital output to high level (HIGH).</li> <li>• uc480.Defines.IO.FlashMode.ConstantLow: Statically sets the digital output to low level (LOW).</li> <li>• uc480.Defines.IO.FlashMode.FreerunLowActive: Enables the flash strobe in freerun mode. The digital output is set to low level for the flash duration.</li> <li>• uc480.Defines.IO.FlashMode.FreerunHighActive: Enables the flash strobe in freerun mode. The digital output is set to high level for the flash duration.</li> </ul>
------	--

## 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  $\mu$ 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"> <li>• uc480.Defines.IO.GPIO.One</li> <li>• uc480.Defines.IO.GPIO.Two</li> </ul>
-----------	--

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

Camera.IO.Flash.SetGPIOParams

**Syntax**

uc480.IOFlash.SetGPIOParams(uint u32Delay, uint u32Duration)

**Description**

Sets the flash delay and flash duration and allows the minimum values for GPIOs.



Attention: For values below 20 µs an unpredictable behavior can occur when flashing is done via the normal flash pin.

**Parameter**

u32Delay	Sets the flash delay.
u32Duration	Sets the flash duration.

**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	<b>Mode to be set:</b> <ul style="list-style-type: none"> <li>• uc480.Defines.IO.FlashMode.Off: Disables the digital output.</li> <li>• uc480.Defines.IO.FlashMode.TriggerLowActive: Enables the flash strobe in trigger mode. The digital output is set to low level for the flash duration.</li> <li>• uc480.Defines.IO.FlashMode.TriggerHighActive: Enables the flash strobe in trigger mode. The digital output is set to high level for the flash duration.</li> <li>• uc480.Defines.IO.FlashMode.ConstantHigh: Statically sets the digital output to high level (HIGH).</li> <li>• uc480.Defines.IO.FlashMode.ConstantLow: Statically sets the digital output to low level (LOW).</li> <li>• uc480.Defines.IO.FlashMode.FreerunLowActive: Enables the flash strobe in freerun mode. The digital output is set to low level for the flash duration.</li> <li>• uc480.Defines.IO.FlashMode.FreerunHighActive: Enables the flash strobe in freerun mode. The digital output is set to high level for the flash duration.</li> </ul>
---------------	--

## 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  $\mu$ s).

## Parameter

u32Delay	Sets the delay.
u32Duration	Sets the duration.

## 3.1.1.16.2 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/LVCMOS voltages. For information on GPIO wiring, please refer to the electrical specifications in the uc480 manual.

## Methods

Method	Description
<a href="#">GetDirection</a>	Returns the input/output state of the GPIOs.
<a href="#">GetState</a>	Returns the state of the GPIO.
<a href="#">GetSupported</a>	Returns the supported GPIOs.
<a href="#">SetDirection</a>	Set the GPIO on input/output.
<a href="#">SetState</a>	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: • uc480.Defines.IO.GPIO.One • uc480.Defines.IO.GPIO.Two
direction	Returns the direction of the GPIO: • uc480.Defines.IO.Direction.In • uc480.Defines.IO.Direction.Out

### Example

```
// set direction of 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.Defin...
```

```
// 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.GetConfiguration

### Syntax

```
uc480.IOGpio.GetConfiguration(uc480.Defines.IO.GPIO gpio, out uc480.Defines.IO.GPIOConfiguration supportedCo...
```

### Description

Returns the configuration of the GPIOs.

**Parameter**

gpio	<b>GPIO which configuration is to be returned:</b> <ul style="list-style-type: none"> <li>• uc480.Defines.IO.GPIO.One</li> <li>• uc480.Defines.IO.GPIO.Two</li> </ul>
supportedConfiguration	<b>Returns the supported configuration of the GPIO:</b> <ul style="list-style-type: none"> <li>• uc480.Defines.IO.GPIOConfiguration.ComportRX: <b>GPIO is used as serial interface</b></li> <li>• uc480.Defines.IO.GPIOConfiguration.ComportTX: <b>GPIO is used as serial interface</b></li> <li>• uc480.Defines.IO.GPIOConfiguration.Flash: <b>GPIO is used for flash</b></li> <li>• uc480.Defines.IO.GPIOConfiguration.Input: <b>GPIO is used as input</b></li> <li>• uc480.Defines.IO.GPIOConfiguration.Output: <b>GPIO is used as output</b></li> <li>• uc480.Defines.IO.GPIOConfiguration.PWM: <b>GPIO is used for pulse width modulation</b></li> <li>• uc480.Defines.IO.GPIOConfiguration.Trigger: <b>GPIO is used for trigger</b></li> </ul>
configuration	<b>Returns the current configuration of the GPIO (see also above)</b>
state	<b>Returns the state of the GPIO:</b> <ul style="list-style-type: none"> <li>• uc480.Defines.IO.State.High</li> <li>• uc480.Defines.IO.State.Low</li> </ul>

**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	<b>GPIO which state is to be returned:</b> <ul style="list-style-type: none"> <li>• uc480.Defines.IO.GPIO.One</li> <li>• uc480.Defines.IO.GPIO.Two</li> </ul>
state	<b>Returns the state of the GPIO:</b> <ul style="list-style-type: none"> <li>• uc480.Defines.IO.State.High</li> <li>• uc480.Defines.IO.State.Low</li> </ul>

## 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.SetConfiguration

#### Syntax

```
uc480.IOGpio.SetConfiguration(uc480.Defines.IO.GPIO gpio, uc480.Defines.IO.GPIOConfiguration configuration)  
uc480.IOGpio.SetConfiguration(uc480.Defines.IO.GPIO gpio, uc480.Defines.IO.GPIOConfiguration configuration,
```

#### Description

Sets the configuration of the GPIOs.

## Parameter

gpio	<b>GPIO which state is to be set:</b> <ul style="list-style-type: none"> <li>• uc480.Defines.IO.GPIO.One</li> <li>• uc480.Defines.IO.GPIO.Two</li> </ul>
configuration	<b>Sets the configuration of the GPIO:</b> <ul style="list-style-type: none"> <li>• uc480.Defines.IO.GPIOConfiguration.ComportRX: <b>GPIO is used as serial interface</b></li> <li>• uc480.Defines.IO.GPIOConfiguration.ComportTX: <b>GPIO is used as serial interface</b></li> <li>• uc480.Defines.IO.GPIOConfiguration.Flash: <b>GPIO is used for flash</b></li> <li>• uc480.Defines.IO.GPIOConfiguration.Input: <b>GPIO is used as input</b></li> <li>• uc480.Defines.IO.GPIOConfiguration.Output: <b>GPIO is used as output</b></li> <li>• uc480.Defines.IO.GPIOConfiguration.PWM: <b>GPIO is used for pulse width modulation</b></li> <li>• uc480.Defines.IO.GPIOConfiguration.Trigger: <b>GPIO is used for trigger</b></li> </ul>
state	<b>Sets the state of the GPIO:</b> <ul style="list-style-type: none"> <li>• uc480.Defines.IO.State.High</li> <li>• uc480.Defines.IO.State.Low</li> </ul>

## 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	<b>GPIO which direction is to be set:</b> <ul style="list-style-type: none"> <li>• uc480.Defines.IO.GPIO.One</li> <li>• uc480.Defines.IO.GPIO.Two</li> </ul>
direction	<b>Sets the direction of the GPIO:</b> <ul style="list-style-type: none"> <li>• uc480.Defines.IO.Direction.In</li> <li>• uc480.Defines.IO.Direction.Out</li> </ul>

## 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.Defin
```

// 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: • uc480.Defines.IO.GPIO.One • uc480.Defines.IO.GPIO.Two
state	Sets 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.I
```

// get state of gpio1  
uc480.Defines.IO.State currentState;  
statusRet = Camera.IO.IOGpio.GetState(uc480.Defines.IO.GPIO.One, **out** currentState);

### 3.1.16.3 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
<a href="#">Get</a>	Returns the state of the LED.
<a href="#">Set</a>	Sets the state of the LED.
<a href="#">Toggle</a>	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

state	<b>State to be set:</b> <ul style="list-style-type: none"> <li>• <code>uc480.Defines.IO.LedState.One</code>: Sets LED to orange.</li> <li>• <code>uc480.Defines.IO.LedState.Two</code>: Sets LED to green.</li> </ul>
-------	---

## Class

[uc480.IOLed](#)

### Accessible

Camera.IO.Led

### Syntax

```
uc480.IOLed.Toggle()
```

### Description

Toggles between the LED states.

### Parameter

None

### 3.1.1.16.4 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
<a href="#">GetDutyCycleRange</a>	Returns the range for the duty cycle.
<a href="#">GetFrequencyRange</a>	Returns the range for the frequency.
<a href="#">GetMode</a>	Returns the current PWM mode.
<a href="#">GetParams</a>	Returns the current values of the PWM parameters.
<a href="#">GetSupported</a>	Returns the GPIOs which can be used for PWM.
<a href="#">SetMode</a>	Sets the current PWM mode.
<a href="#">SetParams</a>	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	<b>Mode to be set:</b> <ul style="list-style-type: none"><li>• uc480.Defines.IO.PwmMode.Flash: Sets the flash output as output for PWM mode.</li><li>• uc480.Defines.IO.PwmMode.Gpio1: Sets GPIO 1 as output.</li><li>• uc480.Defines.IO.PwmMode.Gpio2: Sets GPIO 2 as output.</li></ul>
------	--

## 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

f64Frequency	Sets the frequency value.
f64DutyCycle	Sets the duty cycle value.

## 3.1.1.17 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
<a href="#">GetEnable</a>	Enables/disables the LUT.
<a href="#">GetMode</a>	Returns the current set LUT mode.
<a href="#">GetState</a>	Returns the current state of the LUT.
<a href="#">GetStateInfo</a>	Returns current state information of the LUT.
<a href="#">GetSupportedInfo</a>	Returns the current support information of the LUT
<a href="#">GetValuesComplete</a>	Returns the LUT values set by the user after the gamma, contrast and brightness values have been taken into account.
<a href="#">GetValuesPreset</a>	Returns the predefined LUT.
<a href="#">GetValuesUser</a>	Returns the LUT values set by the user without modifications.
<a href="#">Load</a>	Loads and sets the LUT from a file.
<a href="#">Save</a>	Saves a set LUT into a file.
<a href="#">SetEnable</a>	Enables/disables the LUT.
<a href="#">SetMode</a>	Sets the LUT calculation mode.
<a href="#">SetPreset</a>	Sets a predefined LUT.
<a href="#">SetValuesUser</a>	Sets the LUT values of the user.

### 3.1.1.17.1 Preset

The `Preset` class provides methods for setting a predefined LUT.

#### Methods

Method	Description
<a href="#">Astrol</a>	Predefined LUT, false-color display of the image
<a href="#">ColdHot</a>	Predefined LUT, false-color display of the image
<a href="#">Glow1</a>	Predefined LUT, false-color display of the image
<a href="#">Glow2</a>	Predefined LUT, false-color display of the image
<a href="#">Hot</a>	Predefined LUT, false-color display of the image
<a href="#">Identity</a>	Predefined LUT, linear LUT, no image modifications
<a href="#">Map1</a>	Predefined LUT, false-color display of the image
<a href="#">Negativ</a>	Predefined LUT, inverts the image
<a href="#">OnlyBlue</a>	Predefined LUT, shows only the blue channel of the image
<a href="#">OnlyGreen</a>	Predefined LUT, shows only the green channel of the image
<a href="#">OnlyRed</a>	Predefined LUT, shows only the red channel of the image
<a href="#">Rainbow1</a>	Predefined LUT, false-color display of the image
<a href="#">Sepic</a>	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.DigitalGain2x

**Syntax**

uc480.LutPreset.DigitalGain2x()

**Description**

Sets a predefined LUT, digital brightness correction of the image.

**Parameter**

None

**Class**

[uc480.LutPreset](#)

**Accessible**

Camera.Lut.Preset.DigitalGain4x

**Syntax**

uc480.LutPreset.DigitalGain4x()

**Description**

Sets a predefined LUT, digital brightness correction of the image.

**Parameter**

None

**Class**

[uc480.LutPreset](#)

**Accessible**

Camera.Lut.Preset.DigitalGain8x

**Syntax**

uc480.LutPreset.DigitalGain8x()

**Description**

Sets a predefined LUT, digital brightness correction 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

### 3.1.1.17.2 Raw

The `Raw` class provides methods for setting the use of the camera LUT in combination with RAW formats.

## Methods

Method	Description
<a href="#">GetEnable</a>	Returns if the camera LUT can be used in combination with RAW formats.
<a href="#">SetEnable</a>	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

**3.1.1.17.3 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

**3.1.1.17.4 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\(\)](#))

**3.1.1.17.5 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	<code>uc480.Defines.LutState.HardwareGamma:</code> <b>Gamma is calculated in hardware</b>
	<code>uc480.Defines.LutState.HardwareLut:</code> <b>LUT is calculated in hardware</b>
	<code>uc480.Defines.LutState.HardwareLutAndGamma:</code> <b>LUT and gamma are calculated in hardware</b>
	<code>uc480.Defines.LutState.SoftwareGamma:</code> <b>Gamma is calculated in software</b>
	<code>uc480.Defines.LutState.SoftwareLut:</code> <b>LUT is calculated in software</b>
	<code>uc480.Defines.LutState.SoftwareLutAndGamma:</code> <b>LUT and gamma are calculated in software</b>
	<code>uc480.Defines.LutState.Inactive:</code> <b>LUT is inactive (No gamma and no LUT is set)</b>
	<code>uc480.Defines.LutState.NotSupported:</code> <b>LUT is not supported with the current settings</b>

**3.1.1.17.6 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](#)

**3.1.1.17.7 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](#)

**3.1.1.17.8 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.

**3.1.1.17.9 GetValuesPreset****Class**[uc480.Lut](#)**Accessible**

Camera.Lut

**Syntax**

```
uc480.Lut.GetValuesPreset(uc480.Defines.LutPreset preset, out double[] f64Red, out double[] f64Green, out do
```

**Description**

Returns the predefined LUT.

**Parameter**

preset	A predefined LUT, see <a href="#">SetPreset()</a>
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.

### 3.1.1.17.10 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.

### 3.1.1.17.11 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

### 3.1.1.17.12 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

**3.1.1.17.13 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**

<code>bEnable</code>	1 = Enable LUT
	0 = Disable LUT

**3.1.1.17.14 SetMode****Class**[uc480.Lut](#)**Accessible**

Camera.Lut

**Syntax**`uc480.Lut.SetMode(uc480.Defines.LutMode mode)`**Description**

Sets the LUT calculation mode.

**Parameter**

<code>mode</code>	<code>uc480.Defines.LutMode.Default:</code> <code>Default mode: If supported the LUT/gamma is done on hardware otherwise in software</code>
	<code>uc480.Defines.LutMode.ForceHardware:</code> <code>LUT/gamma in hardware (if not possible: uc480.Defines.LutState.NotSupported)</code>
	<code>uc480.Defines.LutMode.ForceSoftware:</code> <code>IS_LUT_MODE_ID_FORCE_SOFTWARE: LUT/gamma in software (if not possible: uc480.Defines.LutState.NotSupported)</code>

### 3.1.1.17.15 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.Astrol: 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
	uc480.Defines.LutPreset.Rainbow1: false-color display of the image
	uc480.Defines.LutPreset.Sepic: uses sepia toning for coloring the image

### 3.1.1.17.16 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.

### 3.1.1.18 Memory

The `Memory` class provides methods for managing image memory. The following classes and methods exist:

- [ImageBuffer](#)
- [Sequence](#)

## Methods

Method	Description
<a href="#">Allocate</a>	Allocates an image memory for an image
<a href="#">CopyImageMem</a>	Copies the contents of the image memory to a memory area.
<a href="#">CopyImageMemLines</a>	Copies a number of lines indicated out of the image memory to a memory area.
<a href="#">CopyToArray</a>	Copies the image of the passed image memory to an array.
<a href="#">CopyToBitmap</a>	Copies the image of the passed image memory to a bitmap.
<a href="#">Free</a>	Releases an image memory and removes it from the driver management.
<a href="#">GetActive</a>	Returns the starting address and the ID number of the active image memory.
<a href="#">GetBitsPerPixel</a>	Returns the bits per pixel for the passed image memory.
<a href="#">GetHeight</a>	Returns the height of the image in the image memory.
<a href="#">GetLast</a>	Returns the last ID in the image memory.
<a href="#">GetList</a>	Returns the list of image memory IDs.
<a href="#">GetLocked</a>	Returns if the passed image memory is locked.
<a href="#">GetPitch</a>	Returns the line increment (in bytes).
<a href="#">GetSize</a>	Returns the size of the image in the image memory.
<a href="#">GetWidth</a>	Returns the width of the image in the image memory.
<a href="#">Inquire</a>	Returns the properties of an allocated image memory.
<a href="#">Lock</a>	Locks the image memory defined by the ID.
<a href="#">SetActive</a>	Makes the specified image memory the active memory.
<a href="#">ToBitmap</a>	Returns a bitmap which contains the image.
<a href="#">ToIntPtr</a>	Returns a pointer to the image memory of the active/given image memory ID.
<a href="#">Unlock</a>	Unlocks the image memory defined by the ID.

### 3.1.1.18.1 ImageBuffer

The `ImageBuffer` class allows accessing the captured images in the camera memory of a GigE uc480 camera.

#### Methods

Method	Description
<a href="#"><u>GetEnable</u></a>	Returns if the camera memory mode is supported.
<a href="#"><u>GetImageRange</u></a>	Returns the information about a specific iteration, e.g. the number of images in the iteration.
<a href="#"><u>GetIterationRange</u></a>	Returns the number of iterations which are currently in the camera memory.
<a href="#"><u>GetSupported</u></a>	Returns if memory mode is supported by the camera.
<a href="#"><u>ReleaseIteration</u></a>	Releases all iterations up to the given ID in the camera memory.
<a href="#"><u>SetEnable</u></a>	Enables the camera memory mode.
<a href="#"><u>TransferImage</u></a>	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

### 3.1.1.18.2 Sequence

The `Sequence` class provides methods for controlling the image memory in the ring buffering.

#### Methods

Method	Description
<a href="#"><u>Add</u></a>	Adds an image memory to the list of image memories used for ring buffering.
<a href="#"><u>Clear</u></a>	Removes all image memories from the sequence list.
<a href="#"><u>ExitImageQueue</u></a>	Deletes an image queue.
<a href="#"><u>GetActive</u></a>	Determines the image memory which is currently used for capturing an image.
<a href="#"><u>GetLast</u></a>	Returns the last used sequence memory ID.
<a href="#"><u>GetList</u></a>	Returns an array with all sequence memory IDs.
<a href="#"><u>GetLocked</u></a>	Returns if the passed sequence memory is locked.
<a href="#"><u>InitImageQueue</u></a>	Enables the queue mode for existing image memory sequences.
<a href="#"><u>Lock</u></a>	Locks write access to an image memory within a sequence.
<a href="#"><u>ToMemory</u></a>	Returns the image memory ID for the passed sequence memory ID.
<a href="#"><u>Unlock</u></a>	Unlocks a previously locked sequence memory.
<a href="#"><u>WaitForNextImage</u></a>	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 <sup>32</sup> -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.

### 3.1.1.18.3 Allocate

#### Class

[uc480.Memory](#)

#### Accessible

Camera.Memory

#### Syntax

```
uc480.Memory.Allocate()
uc480.Memory.Allocate(bool bActive)
uc480.Memory.Allocate(out int s32MemId, bool bSetActiveMem)
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

### 3.1.1.18.4 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.

### 3.1.1.18.5 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

### 3.1.1.18.6 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

### 3.1.1.18.7 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

**3.1.1.18.8 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

**3.1.1.18.9 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

### 3.1.1.18.10 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

### 3.1.1.18.11 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.

### 3.1.1.18.12 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

**3.1.1.18.13 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

**3.1.1.18.14 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

**3.1.1.18.15 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

## 3.1.18.16 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.

## 3.1.18.17 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.

**3.1.1.18.18 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 <a href="#">Allocate()</a>
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.

**3.1.1.18.19 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.

### 3.1.18.20 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

### 3.1.18.21 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"> <li>• <code>System.Drawing.Imaging.PixelFormat.Format32bppArgb</code> (<code>uc480.Defines.ColorMode.RGBA8Packed</code>/ <code>uc480.Defines.ColorMode.BGRA8Packed</code>)</li> <li>• <code>System.Drawing.Imaging.PixelFormat.Format24bppRgb</code> (<code>uc480.Defines.ColorMode.RGB8Packed</code>/ <code>uc480.Defines.ColorMode.BGR8Packed</code>)</li> <li>• <code>System.Drawing.Imaging.PixelFormat.Format16bppRgb565</code> (<code>uc480.Defines.ColorMode.BGR565Packed</code>)</li> <li>• <code>System.Drawing.Imaging.PixelFormat.Format16bppRgb555</code> (<code>uc480.Defines.ColorMode.BGR5Packed</code>)</li> <li>• <code>System.Drawing.Imaging.PixelFormat.Format8bppIndexed</code> (<code>uc480.Defines.ColorMode.Mono8</code>/ <code>uc480.Defines.ColorMode.SensorRaw8</code>)</li> </ul>

**3.1.1.18.22 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

**3.1.1.18.23 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.

**3.1.1.19 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
<a href="#">Disable</a>	Disables the passed message.
<a href="#">Enable</a>	Enables the passed message.



For the usage of events see also the uc480.NET C# SimpleLive and the uc480MultipleCameraScan demos.

**3.1.1.19.1 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\(\)](#)).

**3.1.1.19.2 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><b>Message to be enabled:</b></p> <ul style="list-style-type: none"> <li>• uc480.Defines.Message.AutoBrightnessFinished: The automatic brightness control in the run-once mode is completed.</li> <li>• uc480.Defines.Message.AutoFocusFinished: Automatic focus control is finished (XS only)</li> <li>• uc480.Defines.Message.CameraMemory: In the camera memory mode an image acquisition iteration is finished.</li> <li>• uc480.Defines.Message.CaptureStatus: There is an information about image capturing available. This information can be requested by <a href="#">GetCaptureStatus()</a>.</li> <li>• 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.</li> <li>• uc480.Defines.Message.DeviceReconnected: A camera initialized with <a href="#">Init()</a> and disconnected afterwards was reconnected.</li> <li>• uc480.Defines.Message.DeviceRemoved: A camera initialized with <a href="#">Init()</a> was disconnected.</li> <li>• 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.</li> <li>• uc480.Defines.Message.Frame: A new image is available.</li> <li>• uc480.Defines.Message.NewDevice: A new camera was connected. This is independent of the device handle.</li> <li>• uc480.Defines.Message.Sequence: The sequence is completed.</li> <li>• 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.</li> <li>• uc480.Defines.Message.WhitebalanceFinished: The automatic white balance control is completed.</li> </ul>
ptr	Pointer for receiving the message.

### 3.1.1.20 Parameter

The `Parameter` class provides methods for saving and loading camera parameter sets. Only camera-specific ini files can be loaded.



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
<a href="#"><u>Clear</u></a>	Deletes the camera parameter set in the EEPROM.
<a href="#"><u>GetNumSupported</u></a>	Returns the number of supported parameter sets in the camera EEPROM.
<a href="#"><u>GetSupported</u></a>	Returns if a camera parameter set in the EEPROM is supported.
<a href="#"><u>Load</u></a>	Loads a camera parameter set from the EEPROM or from a file.
<a href="#"><u>ResetToDefault</u></a>	Resets all parameters to the camera-specific defaults as specified by the driver.
<a href="#"><u>Save</u></a>	Saves a camera parameter set into the EEPROM or into a file.

#### 3.1.1.20.1 Clear

##### Class

[uc480.Parameter](#)

##### Accessible

Camera.Parameter

##### Syntax

`uc480.Parameter.Clear()`

##### Description

Deletes the camera parameter set in the EEPROM.

##### Parameter

none

#### 3.1.1.20.2 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.

### 3.1.1.20.3 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**

supported	1 = Zoom function is supported
	0 = Zoom function is not supported

### 3.1.1.20.4 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**

<none>	Parameter set is loaded from the EEPROM
strFilename	Parameter set is loaded from the file specified in strFilename.

### 3.1.1.20.5 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

### 3.1.1.20.6 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 strFilename.

### 3.1.1.21 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 bit depth

Color formats with a bit depth of more than 8 bits per channel are USB 3 uc480 camera 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
<a href="#">Get</a>	Returns the current setting.
<a href="#">GetBitsPerPixel</a>	Returns the bits per pixel for the current color mode.
<a href="#">GetBytesPerPixel</a>	Returns the bytes per pixel for the current color mode.
<a href="#">Set</a>	Sets the color mode.

### 3.1.1.21.1 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\(\)](#)).

### 3.1.1.21.2 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.

### 3.1.1.21.3 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.

### 3.1.1.21.4 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> .
<code>uc480.Defines.ColorMode.SensorRaw12</code>	Raw sensor data (12), for monochrome and color cameras, LUT/gamma active. For color cameras same as <code>uc480.Defines.ColorMode.BAYER_RG12</code> .
<code>uc480.Defines.ColorMode.SensorRaw8</code>	Raw sensor data (8), for monochrome and color cameras, LUT/gamma active. For color cameras same as <code>uc480.Defines.ColorMode.BAYER_RG8</code> .
<code>uc480.Defines.ColorMode.RGBA12Unpacked</code>	Unpacked RGB (12 12 12), for monochrome and color cameras, LUT/gamma active
<code>uc480.Defines.ColorMode.RGB12Unpacked</code>	Unpacked RGB (12 12 12), for monochrome and color cameras, LUT/gamma active
<code>uc480.Defines.ColorMode.RGB10Unpacked</code>	Unpacked RGB (10 10 10), for monochrome

ed	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 <sub>b</sub> C <sub>r</sub> 4:2:2 (8 8), for monochrome and color cameras, LUT/gamma active
uc480.Defines.ColorMode.Jpeg	JPEG for XS cameras
uc480.Defines.ColorMode.PreferPackedSourceFormat	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): <ul style="list-style-type: none"><li>• IS_CM_SENSOR_RAW10</li><li>• IS_CM_MONO10</li><li>• IS_CM_BGR10_UNPACKED</li><li>• IS_RGB10_UNPACKED</li></ul> 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.
--	---

### 3.1.1.22 RopEffect

The `RopEffect` class provides methods for controlling the real-time image geometry modification (Rop = raster operation).

#### Methods

Method	Description
<a href="#">Get</a>	Returns the current settings.
<a href="#">Set</a>	Sets the values for the image mirroring.

#### 3.1.1.22.1 Get

##### Class

[uc480.RopEffect](#)

##### Accessible

Camera.RopEffect

##### Syntax

```
uc480.RopEffect.Get(out uc480.Defines.RopEffectMode mode)
```

##### Description

Returns the current settings.

##### Parameter

mode	<b>Returns the mode of the Rop effect:</b> <ul style="list-style-type: none"> <li>• <code>uc480.Defines.RopEffectMode.UpDown</code>: Mirrors the image along the horizontal axis.</li> <li>• <code>uc480.Defines.RopEffectMode.LeftRight</code>: Mirrors the image along the vertical axis.</li> <li>• <code>uc480.Defines.RopEffectMode.MirrorNone</code>: No image mirroring.</li> </ul>
------	--

#### 3.1.1.22.2 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	<b>Mode of the Rop effect:</b> <ul style="list-style-type: none"> <li>• uc480.Defines.RopEffectMode.UpDown: Mirrors the image along the horizontal axis.</li> <li>• uc480.Defines.RopEffectMode.LeftRight: Mirrors the image along the vertical axis.</li> <li>• uc480.Defines.RopEffectMode.MirrorNone: No image mirroring.</li> </ul> <p>Depending on the sensor, this operation is performed in the camera or in the PC software.</p>
bEnable	<b>Enables/disables Rop effect:</b> 1 = Enable 0 = Disable

### 3.1.1.23 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
<a href="#">Get</a>	Returns the current setting for color saturation.
<a href="#">GetDefault</a>	Returns the default value for color saturation.
<a href="#">GetRange</a>	Returns the range for color saturation.
<a href="#">GetSupported</a>	Returns the supported modes for color saturation.
<a href="#">Set</a>	Sets the values for color saturation.

### 3.1.1.23.1 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.

**3.1.1.23.2 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.

**3.1.1.23.3 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.

**3.1.1.23.4 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.
------------	--

**3.1.1.23.5 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.

### 3.1.1.24 Size

The `Size` class provides classes for changing the image size or format. The following classes exist:

- [AOI](#)
- [Binning](#)
- [Subsampling](#)

#### 3.1.1.24.1 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
<a href="#">Get</a>	Returns the AOI.
<a href="#">GetAbsX</a>	Returns the X position.
<a href="#">GetAbsY</a>	Returns the Y position.
<a href="#">GetAutoBrightness</a>	Returns the AOI for automatic brightness control
<a href="#">GetOriginal</a>	Returns the AOI without binning, sub-sampling or scaling.
<a href="#">GetPosFastSupported</a>	Returns if fast AOI position changes are supported.
<a href="#">GetPosRange</a>	Returns the range of possible positions.
<a href="#">GetSizeRange</a>	Returns the range possible sizes.
<a href="#">GetWhiteBalance</a>	Returns the AOI for automatic white balance
<a href="#">Set</a>	Sets the AOI.
<a href="#">SetAutoBrightness</a>	Sets the AOI for automatic brightness control
<a href="#">SetPosFast</a>	Allows changing the AOI position very quickly.
<a href="#">SetWhiteBalance</a>	Sets the AOI for automatic white balance



Note that only the camera models DCC1240 and DCC3240 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
<a href="#">Disable</a>	Disables Multi AOI.
<a href="#">GetAxes</a>	Returns the set multi AOI mode.
<a href="#">GetSupported</a>	Returns the supported multi AOI modes.
<a href="#">SetAxes</a>	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.GetAxes(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.SetAxes(uint[] u32Axes)`**Description**

Sets the multi AOI mode.

**Parameter**

- u32Axes: Array with the axes of the mutli AOI.



Note that only the camera models DCC1240 and DCC3240 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 4.81 binning, subsampling and scaler are not supported.

**Methods**

Method	Description
<a href="#">GetEnable</a>	Returns the bitmask.
<a href="#">GetParams</a>	Returns the parameters of AOI 2, 3 or 4.
<a href="#">GetSupported</a>	Returns a bitmask with the supported AOIs.
<a href="#">SetEnable</a>	Set a bitmask defining which AOIs should be active.
<a href="#">SetParams</a>	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 DCC1240 and DCC3240 camera models).

## Parameter

- mode: Returns the bitmask.

## Class

[uc480.SizeAOISquence](#)

## Accessible

Camera.Size.AOI.Sequences

## Syntax

```
uc480.SizeAOISquence.GetParams(out uc480.Types.AoiSequenceParameter parameter)
```

## Description

Returns the parameters of AOI 2, 3 or 4 (only DCC1240 and DCC3240 camera models).

## Parameter

- parameter: Returns [uc480.Types.AoiSequenceParameter](#)

## Class

[uc480.SizeAOISquence](#)

## Accessible

Camera.Size.AOI.Sequences

## Syntax

```
uc480.SizeAOISquence.GetSupported(out uc480.Defines.AOISquenceMode mode)  
uc480.SizeAOISquence.GetSupported(out int s32Mask)
```

## Description

Returns a bitmask with the supported AOIs (only DCC1240 and DCC3240 camera models).

## Parameter

- mode/s32Mask: Returns the supported AOIs.

## Class

[uc480.SizeAOISquence](#)

## Accessible

Camera.Size.AOI.Sequences

## Syntax

```
uc480.SizeAOISquence.SetEnable(uc480.Defines.AOISquenceMode mode)
```

## Description

Set a bitmask defining which AOIs should be active (only DCC1240 and DCC3240 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.SizeAOISequences](#)

### Accessible

Camera.Size.AOI.Sequences

## Syntax

```
uc480.SizeAOISequences.SetParams(uc480.Types.AoiSequenceParameter parameter)
```

## Description

Sets the parameters of AOI 2, 3 or 4 (only DCC1240 and DCC3240 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

<code>s32PosX</code>	Returns the X position.
<code>s32PosY</code>	Returns the Y position.
<code>s32Width</code>	Returns the width.
<code>s32Height</code>	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,
```

### 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	Height 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	Heighth of the AOI

### 3.1.1.24.2 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 be 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](#)



For the DCC1240 and DCC3240 models, you can use binning only combined for the horizontal and the vertical direction. Please see also the information in the DCx manual.

## Methods

Method	Description
<a href="#">Get</a>	Returns the current setting.
<a href="#">GetFactorHorizontal</a>	Returns the horizontal binning factor.
<a href="#">GetFactorVertical</a>	Returns the vertical binning factor.
<a href="#">GetSupported</a>	Returns the supported binning modes.
<a href="#">GetType</a>	Indicates whether the camera uses color-proof binning or not.
<a href="#">Set</a>	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.Horizontal12X</code>	Enables horizontal binning with factor 2.
<code>uc480.Defines.BinningMode.Vertical3X</code>	Enables vertical binning with factor 3.
<code>uc480.Defines.BinningMode.Horizontal13X</code>	Enables horizontal binning with factor 3.
<code>uc480.Defines.BinningMode.Vertical4X</code>	Enables vertical binning with factor 4.
<code>uc480.Defines.BinningMode.Horizontal14X</code>	Enables horizontal binning with factor 4.
<code>uc480.Defines.BinningMode.Vertical5X</code>	Enables vertical binning with factor 5.
<code>uc480.Defines.BinningMode.Horizontal15X</code>	Enables horizontal binning with factor 5.
<code>uc480.Defines.BinningMode.Vertical6X</code>	Enables vertical binning with factor 6.
<code>uc480.Defines.BinningMode.Horizontal16X</code>	Enables horizontal binning with factor 6.
<code>uc480.Defines.BinningMode.Vertical8X</code>	Enables vertical binning with factor 8.
<code>uc480.Defines.BinningMode.Horizontal18X</code>	Enables horizontal binning with factor 8.
<code>uc480.Defines.BinningMode.Vertical16X</code>	Enables vertical binning with factor 16.
<code>uc480.Defines.BinningMode.Horizontal116X</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.

### 3.1.1.24.3 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
<a href="#">Get</a>	Returns the current setting.
<a href="#">GetFactorHorizontal</a>	Returns the horizontal sub-sampling factor.
<a href="#">GetFactorVertical</a>	Returns the vertical sub-sampling factor.
<a href="#">GetSupported</a>	Returns the supported sub-sampling modes.
<a href="#">GetType</a>	Indicates whether the camera uses color-proof sub-sampling.
<a href="#">IsSupported</a>	Returns if the passed subsampling mode is supported.
<a href="#">Set</a>	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.

### 3.1.1.25 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.

#### 3.1.1.25.1 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.

### 3.1.1.25.2 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 <a href="#">GetSupported()</a> )
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.

### 3.1.1.25.3 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

### 3.1.1.25.4 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\(\)](#)).

### 3.1.1.25.5 Set

#### Class

[uc480.TestImage](#)

#### Accessible

Camera.TestImage

#### Syntax

uc480.TestImage.Set(uc480.Defines.TestImageMode mode, int s32Param)

#### Description

Enables/disables 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 <a href="#">GetSupported()</a>
s32Param	Additional parameter used for modifying the test image. Not available for all test images.

**3.1.1.26 Timeout**

The `Timeout` class provides methods for controlling the user-defined timeout values of the uc480 API.

**Methods**

Method	Description
<a href="#">Get</a>	Returns the user-defined timeout values of the uc480 API.
<a href="#">Set</a>	Sets the user-defined timeout values of the uc480 API.

**3.1.1.26.1 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: • <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.

**3.1.1.26.2 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 • <code>uc480.Defines.TimeoutMode.Trigger</code> : 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.

### 3.1.1.27 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
<a href="#">Optimal()</a>	Sets the highest possible pixel clock.

### 3.1.1.27.1 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:

- [Fine](#)
- [Long](#)

### Methods

Method	Description
<a href="#">Get</a>	Returns the currently set exposure time (in ms).
<a href="#">GetDefault</a>	Returns the default setting for the exposure time.
<a href="#">GetRange</a>	Returns the exposure time range.
<a href="#">GetSupported</a>	Returns the if the exposure time is supported.
<a href="#">Set</a>	Sets the exposure time.

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
<a href="#">GetRange</a>	Returns the exposure time range in fine increments for some sensors.
<a href="#">GetSupported</a>	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
<a href="#">GetEnable</a>	Returns the current settings for long exposure.
<a href="#">GetRange</a>	Returns the value range for long exposure.
<a href="#">GetSupported</a>	Returns is long time exposure is supported.
<a href="#">SetEnable</a>	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.

### 3.1.1.27.2 Framerate

The `Framerate` class provides methods for controlling the frame rate of a uc480 camera.

## Methods

Method	Description
<a href="#">Get</a>	Returns the current setting of the frame rate.
<a href="#">GetCurrentFps</a>	In live capture mode, the method returns the number of frames actually captured per second.
<a href="#">GetDefault</a>	Returns the default frame rate.
<a href="#">GetFrameRateRange</a>	Returns the minimum and maximum frame rate and the increment.
<a href="#">GetFrameTimeRange</a>	Returns the possible frame rate settings which are available for the current pixel clock setting.
<a href="#">Set</a>	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:

$$fps_{\min} = \frac{1}{f_{\max}}$$

$$fps_{\max} = \frac{1}{f_{\min}}$$

$$fps_n = \frac{1}{(f_{\min} + n * \text{intervall})}$$

The call of this method makes only sense in the freerun mode.

## Parameter

range	<b>Minimum:</b> Returns the minimum value. <b>Maximum:</b> Returns the maximum value. <b>Increment:</b> 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.

### 3.1.1.27.3 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
<a href="#">Get</a>	Returns the current set pixel clock in MHz.
<a href="#">GetDefault</a>	Returns the default pixel clock.
<a href="#">GetList</a>	Returns the list with discrete pixel clocks of the camera.
<a href="#">GetNumber</a>	Returns the number of discrete pixel clock which are supported by the camera.
<a href="#">GetRange</a>	Returns the range for the pixel clock.
<a href="#">Set</a>	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

### Syntax

```
uc480.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.

### 3.1.1.27.4 VsyncCount

The `VsyncCount` class provides methods for reading-out the VSYNC counter.

#### Methods

<a href="#">Get</a>	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

<code>s64Vsync</code>	Current VSYNC count
<code>s64Fsync</code>	Current Frame SYNC count

### 3.1.1.27.5 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 be 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).

### 3.1.1.28 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.

The following classes and methods exist:

- [Burst](#)
- [Counter](#)
- [Debounce](#)
- [Delay](#)

## Methods

Method	Description
<a href="#">Force</a>	Forces a software-controlled capture of an image while a capturing process triggered by hardware is in progress.
<a href="#">Get</a>	Returns the current trigger mode.
<a href="#">GetStatus</a>	Returns the current signal level at the trigger input.
<a href="#">GetSupported</a>	Returns the supported trigger modes.
<a href="#">Set</a>	Sets the trigger mode.

### 3.1.1.28.1 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](#)).



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
<a href="#">Get</a>	Returns the currently set number of images in a burst.
<a href="#">GetDefault</a>	Returns the default number of images in a burst.
<a href="#">GetRange</a>	Returns the minimum and maximum value the increment for the number of images in a burst.
<a href="#">GetSupported</a>	Returns if the camera supports the burst trigger mode.
<a href="#">Set</a>	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.

**3.1.1.28.2 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
<a href="#">Get</a>	Returns the current count for triggered image captures.
<a href="#">Reset</a>	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

### 3.1.1.28.3 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
<code>Get</code>	Returns the set mode.
<code>GetDefault</code>	Returns the default mode.
<code>GetDelayTime</code>	Returns the set delay time.
<code>GetDelayTimeDefault</code>	Returns the default delay time.
<code>GetDelayTimeRange</code>	Returns the range for the delay time.
<code>GetSupported</code>	Returns the supported modes.
<code>Set</code>	Sets the mode.
<code>SetDelayTime</code>	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  $\mu$ 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  $\mu$ 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  $\mu$ 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.FallingEdge	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 <a href="#">Set()</a> ).

## Class

[uc480.TriggerDebounce](#)

### Accessible

Camera.Trigger.Debounce

### Syntax

```
uc480.TriggerDebounce.SetDelayTime(uint u32Value)
```

### Description

Sets the delay time (in  $\mu$ s).

### Parameter

- `u32Value`: Delay time to be set

### 3.1.1.28.4 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
<a href="#">Get</a>	Returns the currently set delay time.
<a href="#">GetRange</a>	Returns the range for the delay time.
<a href="#">Set</a>	Sets the time by which the image capture is delayed (in $\mu$ 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  $\mu$ s). "0" deactivates the trigger delay.

## Parameter

- `s32Value`: Delay time to be set.

## 3.1.1.28.5 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

### 3.1.1.28.6 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.

### 3.1.1.28.7 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.

### 3.1.1.28.8 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\(\)](#))

### 3.1.1.28.9 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 <a href="#">Capture()</a> (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.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.Off</code>	Off	-
<code>uc480.Defines.TriggerMode.Software</code>	Software trigger	Call of <a href="#">Freeze()</a> (single frame mode)

### 3.1.1.29 Prescaler

The `Prescaler` class provides methods for setting a trigger prescaler. Via the trigger prescaler (frequency divider) you can set for the DCC3240 models that the trigger signal is divided by the set value. This is necessary when the trigger signal delivers more pulses as needed for the captures. With the set value, e.g. 40, you define that the camera captures an image only every 40th trigger signal.



The trigger prescaler is currently supported by the following camera models:

- DCC3240C, DCC3240M, and DCC3240N

The following classes exist:

- [Frame](#)
- [Line](#)

### 3.1.1.29.1 Frame

The `Frame` class provides methods for setting the trigger prescaler for image recordings.



The trigger prescaler is currently supported by the following camera models:

- DCC3240C, DCC3240M, and DCC3240N

## Methods

<a href="#">Get</a>	Returns the current set trigger prescaler for image recordings.
<a href="#">GetDefault</a>	Returns the default prescaler value for image recordings.
<a href="#">GetRange</a>	Returns the range for trigger prescaler for image recordings.
<a href="#">GetSupported</a>	Returns if a trigger prescaler for image recordings is supported.
<a href="#">Set</a>	Sets the trigger prescaler for image recordings.

## Class

[uc480.TriggerPrescalerFrame](#)

### Accessible

Camera.Trigger.Prescaler.Frame.Get

### Syntax

```
uc480.TriggerPrescalerFrame.Get(out uint count)
```

### Description

Returns the current set trigger prescaler for image recordings.

### Parameter

- `count`: Returns the set prescaler value

## Class

[uc480.TriggerPrescalerFrame](#)

### Accessible

Camera.Trigger.Prescaler.Frame.GetDefault

### Syntax

```
uc480.TriggerPrescalerFrame.GetDefault(out uint)
```

## Description

Returns the default prescaler value for image recordings.

## Parameter

- u32Value: default value

## Class

[uc480.TriggerPrescalerFrame](#)

## Accessible

Camera.Trigger.Prescaler.Frame.GetRange

## Syntax

```
uc480.TriggerPrescalerFrame.GetRange(out uc480.Types.Range<uint> range)
uc480.TriggerPrescalerFrame.GetRange(out uint u32Min, out uint u32Max, out uint u32Inc)
```

## Description

Returns the range for trigger prescaler for image recordings.

## 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.TriggerPrescalerFrame](#)

## Accessible

Camera.Trigger.Prescaler.Frame.GetSupported

## Syntax

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

## Description

Returns if a trigger prescaler for image recordings is supported.

## Parameter

supported	1 = Trigger prescaler for image recordings is supported. 0 = Trigger prescaler for image recordings is not supported.
-----------	--

## Class

[uc480.TriggerPrescalerFrame](#)

### Accessible

Camera.Trigger.PrescalerFrame.Set

### Syntax

`uc480.TriggerPrescalerFrame.Set(uint count)`

### Description

Sets the trigger prescaler for image recordings.

### Parameter

- `count`: Prescaler value to be set

## 3.1.1.29.2 Line

The `Line` class provides methods for setting the trigger prescaler for line recordings when using the [AOI merge mode](#).



The trigger prescaler is currently supported by the following camera models:

- DCC3240C, DCC3240M, and DCC3240N

## Methods

<a href="#">Get</a>	Returns the current set trigger prescaler for line recordings.
<a href="#">GetDefault</a>	Returns the default prescaler value for line recordings.
<a href="#">GetRange</a>	Returns the range for trigger prescaler for line recordings.
<a href="#">GetSupported</a>	Returns if a trigger prescaler for line recordings is supported.
<a href="#">Set</a>	Sets the trigger prescaler for line recordings.

## Class

[uc480.TriggerPrescalerLine](#)

### Accessible

Camera.Trigger.PrescalerLine.Get

### Syntax

`uc480.TriggerPrescalerLine.Get(out uint count)`

### Description

Returns the current set trigger prescaler for line recordings.

### Parameter

- `count`: Returns the set prescaler value

## Class

[uc480.TriggerPrescalerLine](#)

### Accessible

Camera.Trigger.Prescaler.Line.GetDefault

### Syntax

`uc480.TriggerPrescalerLine.GetDefault(out uint u32Value)`

### Description

Returns the default prescaler value for line recordings.

### Parameter

- `u32Value`: default value

## Class

[uc480.TriggerPrescalerLine](#)

### Accessible

Camera.Trigger.PrescalerLine.GetRange

### Syntax

`uc480.TriggerPrescalerLine.GetRange(out uc480.Types.Range<uint> range)`  
`uc480.TriggerPrescalerLine.GetRange(out uint u32Min, out uint u32Max, out uint u32Inc)`

### Description

Returns the range for trigger prescaler for line recordings.

### 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.TriggerPrescalerLine](#)

### Accessible

Camera.Trigger.PrescalerLine.GetSupported

### Syntax

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

### Description

Returns if a trigger prescaler for line recordings is supported.

**Parameter**

supported	1 = Trigger prescaler for line recordings is supported. 0 = Trigger prescaler for line recordings is not supported.
-----------	--

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

Camera.Trigger.PrescalerLine.Set

**Syntax**`uc480.TriggerPrescalerLine.Set(uint count)`**Description**

Sets the trigger prescaler for line recordings.

**Parameter**

- `count`: Prescaler value to be set

**3.1.1.30 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
<a href="#">GetFrameCount</a>	Reads out the number of frames that have been saved.
<a href="#">GetFrameRate</a>	Returns the set frame rate for AVI capturing.
<a href="#">GetLostCount</a>	Reads out the number of frames that have been discarded.
<a href="#">GetQuality</a>	Returns the set quality for the frame compression.
<a href="#">GetRunning</a>	Returns the current status of the video.
<a href="#">GetSize</a>	Use this method to retrieve the size of the frame sequence saved to the current AVI file.
<a href="#">GetVideoID</a>	Returns the ID of the current AVI file.
<a href="#">ResetCount</a>	Resets the counters for saved and discarded images.
<a href="#">SetFrameRate</a>	Sets the frame rate for AVI capturing.
<a href="#">SetQuality</a>	Indicates the quality for the frames to be compressed.
<a href="#">Start</a>	Starts the image capture thread.
<a href="#">Stop</a>	Stops the image capture thread.

### 3.1.1.30.1 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

### 3.1.1.30.2 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.

### 3.1.1.30.3 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

### 3.1.1.30.4 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]

### 3.1.1.30.5 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).

### 3.1.1.30.6 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

### 3.1.1.30.7 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

### 3.1.1.30.8 ResetCount

## Class

[uc480.Video](#)

## Accessible

Camera.Video

## Syntax

```
uc480.Video.ResetCount()
```

**Description**

Resets the counters for saved and discarded images.

**Parameter**

None

**3.1.1.30.9 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

**3.1.1.30.10 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 be done before starting the video. For compression, the system uses the JPEG algorithm.

**Parameter**

- `s32Quality`: Image quality [1 = lowest ... 100 = highest]

**3.1.1.30.11 Start****Class**

[uc480.Video](#)

**Accessible**

Camera.Video

**Syntax**

```
uc480.Video.Start(string strFileName)
```

**Description**

Starts the image capture thread.

**Parameter**

- strFileName: AVI file name

**3.1.130.12 Stop****Class**

[uc480.Video](#)

**Accessible**

Camera.Video

**Syntax**

`uc480.Video.Stop()`

**Description**

Stops the image capture thread.

**Parameter**

None

**3.1.131 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.

**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

s32CamID	<ul style="list-style-type: none"> <li>• 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.</li> <li>• s32Cam   uc480.Defines.DeviceEnumeration.UseDeviceID: The camera is opened using the device ID instead of the camera ID.</li> <li>• s32Cam   uc480.Defines.DeviceEnumeration.AllowStarterFwUpload: 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 uc480.Defines.DeviceEnumeration.AllowStarterFwUpload parameter first. Only if an error occurs, call the method with this parameter set.</li> </ul>
hWnd	<p>Window where the Direct3D image will be displayed</p> <p>If hWnd = NULL, DIB mode will be used for image display.</p>

### 3.1.1.32 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

### 3.1.1.33 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.

#### 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

s32Cam	<ul style="list-style-type: none"> <li>• 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.</li> <li>• s32Cam   uc480.Defines.DeviceEnumeration.UseDeviceID: The camera is opened using the device ID instead of the camera ID.</li> <li>• s32Cam uc480.Defines.DeviceEnumeration.AllowStarterFwUpload: 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.</li> </ul>
s32hWnd/ WindowHandle	Window where the Direct3D image will be displayed If <code>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.



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.



#### 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.



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
<a href="#">AddId</a>	Adds an ID to the boot boost list.
<a href="#">ClearIdList</a>	Deletes the boot boost list.
<a href="#">GetEnable</a>	Returns if the boot boost mode is enabled or disabled.
<a href="#">GetIdList</a>	Returns the boot boost list.
<a href="#">RemoveId</a>	Deletes an ID from the boot boost list.
<a href="#">SetEnable</a>	Enables/disables the boot boost mode.
<a href="#">SetIdList</a>	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

<code>bEnable</code>	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 RemoveId

## 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

<code>bEnable</code>	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

**3.2.1.8 Wait****Class**

[uc480.Configuration.BootBoost](#)

**Accessible**

Configuration.BootBoost.Wait

**Syntax**

```
uc480.Configuration.BootBoost.Wait(uint timeout)
uc480.Configuration.BootBoost.Wait()
```

**Description**

Waits until all connected camera have adopted the current settings. If no timeout value is passed, the default timeout is used.

**Parameter**

- `timeout`: Timeout value in seconds. The default timeout is 60 seconds.

**3.2.2 CPUIidleState**

The `CPUIidleState` class provides methods for setting the processor operating states.

**Methods**

Method	Description
<a href="#">GetDisableOnOpen</a>	Returns the processor operating states for power supply unit operation when a uc480 camera is opened.
<a href="#">GetEnable</a>	Returns the state of processor operating states for power supply unit operation and battery operation.
<a href="#">GetSupported</a>	Returns if function parameters for setting the processor operating states are supported.
<a href="#">SetDisableOnOpen</a>	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.CPUIdleState.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.CPUIdleState](#)

## Accessible

Configuration.CPUIdleState

## Syntax

```
uc480.Configuration.CPUIdleState.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.CPUIdleState](#)

## Accessible

Configuration.CPUIdleState

## Syntax

```
uc480.Configuration.CPUIdleState.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
<a href="#">GetEnable</a>	Returns the state of the load of camera parameters during initialization.
<a href="#">GetSupported</a>	Returns if function parameters to load camera parameters during initialization are supported.
<a href="#">SetEnable</a>	Enables/disables the load of camera parameters during initialization.

### 3.2.3.1 GetEnable

#### Class

[uc480.Configuration.InitialParameterSet](#)

#### Accessible

Configuration.InitialParametersSet

#### Syntax

`uc480.Configuration.InitialParametersSet.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.InitialParametersSet

#### Syntax

`uc480.Configuration.InitialParametersSet.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
<a href="#">GetEnable</a>	Returns if the IPO thread is enabled.
<a href="#">GetSupported</a>	Returns if setting the IPO thread is supported.
<a href="#">SetEnable</a>	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
<a href="#">GetEnable</a>	Returns if OpenMP support is enabled.
<a href="#">GetEnableDefault</a>	Returns the default setting for OpenMP support.
<a href="#">GetSupported</a>	Returns if function parameters to configure OpenMP are supported.
<a href="#">SetEnable</a>	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.
---------	---

### 3.2.6 MemoryMode

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

- On USB 3 uc480 CP Rev. 2 models: Enable or disable the internal image memory of the camera. By default, the camera is operated without image memory. Note that this class uses the device ID while the `MemoryMode` class provided by the [Feature](#) class uses the camera object.

#### Methods

Method	Description
Get	Returns if the internal image memory of the camera is enabled.
GetDefault	Returns the default setting of the internal image memory.
GetSupported	Returns if the internal image memory is supported.
Set	Enables/disables the image memory of the camera.

#### 3.2.6.1 Get

#### Class

[uc480.Configuration.MemoryMode](#)

#### Accessible

Configuration.MemoryMode.Get

#### Syntax

```
uc480.Configuration.MemoryMode.Get(int deviceID, out uc480.Defines.MemoryMode mode)
```

#### Description

Returns if the internal image memory of the camera is enabled.

**Parameter**

deviceID	Device ID of the USB 3 uc480 CP Rev. 2 camera	
mode	uc480.Defines.MemoryMode.On:	The internal image memory is enabled.
	uc480.Defines.MemoryMode.Off:	The internal image memory is disabled.

**3.2.6.2 GetDefault****Class**

uc480.Configuration.MemoryMode

**Accessible**

Configuration.MemoryMode.GetDefault

**Syntax**

uc480.Configuration.MemoryMode.GetDefault(int deviceID, out uc480.Defines.MemoryMode mode)

**Description**

Returns the default setting of the internal image memory.

**Parameter**

deviceID	Device ID of the USB 3 uc480 CP Rev. 2 camera	
mode	uc480.Defines.MemoryMode.On:	The internal image memory is enabled.
	uc480.Defines.MemoryMode.Off:	The internal image memory is disabled.

**3.2.6.3 GetSupported****Class**

uc480.Configuration.MemoryMode

**Accessible**

Configuration.MemoryMode.GetSupported

**Syntax**

uc480.Configuration.MemoryMode.GetSupported(int deviceID, out uc480.Defines.MemoryMode supported)

**Description**

Returns if the internal image memory is supported.

**Parameter**

deviceID	Device ID of the USB 3 uc480 CP Rev. 2 camera	
supported	1 = Supported	
	0 = Not supported	

### 3.2.6.4 Set

#### Class

`uc480.Configuration.MemoryMode`

#### Accessible

`Configuration.MemoryMode.Set`

#### Syntax

`uc480.Configuration.MemoryMode.Set(int deviceID, uc480.Defines.MemoryMode mode)`

#### Description

Enables/disables the image memory of the camera.

#### Parameter

<code>deviceID</code>	Device ID of the USB 3 uc480 CP Rev. 2 camera
<code>mode</code>	<code>uc480.Defines.MemoryMode.On</code> : Enables the internal image memory.
	<code>uc480.Defines.MemoryMode.Off</code> : Disables the internal image memory.

## 3.3 uc480.Info

The `Info` class provides classes for querying camera and OS information. The following classes exist:

- [Camera](#)
- [System](#)

### 3.3.1 Camera

The `Camera` class provides methods for querying information about the uc480 API version or the OS version.

#### Methods

Method	Description
<a href="#"><code>GetCameraList</code></a>	Returns information about the connected cameras.
<a href="#"><code>GetDeviceInfo</code></a>	Returns information about connected uc480 cameras.
<a href="#"><code>GetNumberOfDevices</code></a>	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 <a href="#"><code>Init()</code></a> was disconnected.
<code>uc480.Info.Camera.EventNewDevice</code>	A new camera was connected. This is independent of the device handle.

### 3.3.1.1 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](#)

### 3.3.1.2 GetDeviceInfo

#### Class

[uc480.Info.Camera](#)

#### Accessible

Info.Camera

#### Syntax

```
uc480.Info.Camera.GetDeviceInfo(int s32DeviceID, out uc480.Types.DeviceInformation Info)
```

#### Description

Returns information about connected uc480 cameras.

#### Parameter

s32DeviceID	Device ID
Info	Returns <a href="#">uc480.Types.DeviceInformation</a>

### 3.3.1.3 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.

## 3.3.2 System

The `System` class provides methods for querying information about the uc480 API version or the OS version.

### Methods

Method	Description
<a href="#">GetNetVersion</a>	Returns the version of the uc480DotNet.dll.
<a href="#">GetOsVersion</a>	Returns the version of the API.

### 3.3.2.1 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

<code>version</code>	Returns the complete version number
<code>s32Major</code>	Returns the major version
<code>s32Minor</code>	Returns the minor version
<code>s32Build</code>	Returns the build version

### 3.3.2.2 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

## 3.4 uc480.Tools

The `Tools` class provides a class for capturing videos. The following classes exist:

- [Video](#)

### 3.4.1 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
<a href="#">AddFrame</a>	Adds a new frame to an AVI sequence.
<a href="#">Close</a>	Closes an AVI file which was opened using <a href="#">Open()</a> .
<a href="#">Exit</a>	Exits an instance of the uc480 AVI interface which was initialized by <a href="#">Init()</a> .
<a href="#">GetFrameCount</a>	Reads out the number of frames that have been saved.
<a href="#">GetLostCount</a>	Reads out the number of frames that have been discarded.
<a href="#">GetSize</a>	Use this method to retrieve the size of the frame sequence saved to the current AVI file.
<a href="#">Init</a>	Initializes an instance of the uc480 AVI interface.
<a href="#">Open</a>	Opens a new or existing AVI file.
<a href="#">ResetCounter</a>	Resets the counters for saved and discarded images.
<a href="#">SetFramerate</a>	Sets the frame rate for AVI capturing.
<a href="#">SetImageSize</a>	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.
<a href="#">SetQuality</a>	Indicates the quality for the frames to be compressed.
<a href="#">Start</a>	Starts the image capture thread.
<a href="#">Stop</a>	Stops the image capture thread.

### 3.4.1.1 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 <a href="#">Init()</a> .
intPtr	Pointer to the memory containing the image.

### 3.4.1.2 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\(\)](#).

### 3.4.1.3 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

### 3.4.1.4 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

<code>s32VideoID</code>	Video ID set by <a href="#">Init()</a> .
<code>u32Count</code>	Number of saved frames

### 3.4.1.5 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 <a href="#">Init()</a> .
u32Count	Number of discarded frames

### 3.4.1.6 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 <a href="#">Init()</a> .
f64Size	Size in kBytes

### 3.4.1.7 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

**3.4.1.8 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 <a href="#">Init()</a> .
strFileName	Video file to be opened.

**3.4.1.9 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\(\)](#).

**3.4.1.10 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 <a href="#">Init()</a> .
f64FrameRate	The frame rate to be set. Default = 25.0

**3.4.1.11 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 <a href="#">Init()</a> .
colorMode	Color mode for which the converter is to be returned (see <a href="#">GetSupported()</a> ).
s32Width	Width of the entire frame or of the area of interest.
s32Height	Height of the entire frame or of the area of interest.

**3.4.1.12 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 <a href="#">Init()</a> .
s32Quality	Image quality [1 = lowest ... 100 = highest]

## 3.4.1.13 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\(\)](#).

## 3.4.1.14 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\(\)](#).

## 3.5 uc480.Types

- [AoiSequenceParameter](#)
- [CameraInfo](#)
- [CameraInformation](#)
- [CaptureStatus](#)
- [CaptureStatus.CaptureStatusApi](#)
- [CaptureStatus.CaptureStatusDriver](#)
- [CaptureStatus.CaptureStatusEth](#)
- [CaptureStatus.CaptureStatusUsb](#)
- [ConversionParameter](#)
- [DeviceInfoControl](#)

- [DeviceInfoHeartbeat](#)
- [DeviceInformation](#)
- [ImageFormatInfo](#)
- [ImageInfo](#)
- KneePointInformation
- [LutState](#)
- [LutSupportInfo](#)
- [SensorInfo](#)

### 3.5.1 AoiSequenceParameter

#### uc480.Types.AoiSequenceParameter

uc480.Defines.BinningMode	BinningMode	Binning mode (not supported in version 4.81)
int	DetachImageParameter	<ul style="list-style-type: none"> <li>• 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.</li> <li>• 1 = a change of exposure time, gain or position of AOI 1 does not affect the parameters of AOI 2, 3 and 4.</li> </ul>
double	Exposure	Exposure
int	Gain	Gain
int	Index	Index of the AOI
int	NumberOfCycleRepetitions	Number of readout cycles
uc480.Types.Point<int>	Position	X and Y position of the AOI
double	ScalerFactor	Scaling factor (not supported in version 4.81)
uc480.Defines.SubsamplingMode	SubsamplingMode	Sub-sampling mode (not supported in version 4.81)

#### Used in method

- [GetParams \(\)](#) (Sequences class)
- [SetParams \(\)](#)

### 3.5.2 CameraInfo

#### uc480.Types.CameraInfo

uc480.Defines.BoardType	BoardType	<b>Camera type:</b> <ul style="list-style-type: none"> <li>• uc480.Defines.BoardType.Usb_SE: <b>USB uc480 SE</b></li> <li>• uc480.Defines.BoardType.Usb_ME: <b>USB uc480 ME</b></li> <li>• uc480.Defines.BoardType.Usb_RE: <b>USB uc480 RE</b></li> <li>• uc480.Defines.BoardType.Usb_LE: <b>USB uc480 LE</b></li> <li>• uc480.Defines.BoardType.Usb_XS: <b>USB uc480 XS</b></li> <li>• uc480.Defines.BoardType.Eth_HE: <b>GigE uc480 HE</b></li> <li>• uc480.Defines.BoardType.Eth_SE: <b>GigE uc480 SE</b></li> <li>• uc480.Defines.BoardType.Eth_CP: <b>GigE uc480 CP</b></li> <li>• uc480.Defines.BoardType.Eth_RE: <b>GigE uc480 RE</b></li> </ul>
int	CameraID	<b>Camera ID</b>
System.DateTime	Date	<b>System date of the final quality check (e.g. 01.08.2011 (DD.MM.YYYY))</b>
string	ID	<b>Manufacturer of the camera (e.g. IDS GmbH)</b>
string	SerialNumber	<b>Serial number of the camera</b>
string	Version	<b>For USB cameras, this value indicates the USB board hardware version (e.g. v2.10)</b>

#### Used in method

- [GetCameraInfo\(\)](#) (**Information class**)

### 3.5.3 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. DEFINESSENSOR.UI122X_M As the same sensor is used in different camera families, the sensor ID returns even DEFINESSENSOR.UI122X_M.
string	SerialNumber	Serial number of the camera
long	Status	Information for the status of the camera

#### Used in method

- [GetCameraList\(\)](#) (Camera class)

### 3.5.4 CaptureStatus

#### uc480.Types.CaptureStatus

<a href="#">uc480.Types.CaptureStatus.CaptureStatusApi</a>	Api	<b>See</b> <a href="#">uc480.Types.CaptureStatus.CaptureStatusApi</a>
ulong	DevTimeout	The maximum allowable time for image capturing in the camera was exceeded. <b>Possible cause/remedy:</b> The selected timeout value is too low for image capture • Reduce the exposure time • Increase the timeout
<a href="#">uc480.Types.CaptureStatus.CaptureStatusDriver</a>	Driver	<b>See</b> <a href="#">uc480.Types.CaptureStatus.CaptureStatusDriver</a>
<a href="#">uc480.Types.CaptureStatus.CaptureStatusEth</a>	Eth	<b>See</b> <a href="#">uc480.Types.CaptureStatus.CaptureStatusEth</a>
ulong	Total	Returns the total number of errors occurred since the last reset.
<a href="#">uc480.Types.CaptureStatus.CaptureStatusUsb</a>	Usb	<b>See</b> <a href="#">uc480.Types.CaptureStatus.CaptureStatusUsb</a>

**Used in method**

- [GetCaptureStatus\(\)](#) (Information class)

**3.5.5 CaptureStatus.CaptureStatusApi****uc480.Types.CaptureStatus.CaptureStatusApi**

ulong	ConversionFailed	The current image could not be processed correctly. <b>Possible cause:</b> <ul style="list-style-type: none"><li>• Internal error during internal processing of the image</li></ul>
ulong	ImageLocked	The destination buffers are locked and could not be written to. <b>Possible cause/remedy:</b> All destination buffers locked by the application <ul style="list-style-type: none"><li>• Release locked destination memory</li><li>• Allocate more destination memory</li><li>• Reduce the frame rate so that there is more time to process the filled destination memory</li></ul>
ulong	NoDestMem	There is no destination memory for copying the finished image. <b>Possible cause/remedy:</b> Not enough destination memory allocated or all destination buffers locked by the application. <ul style="list-style-type: none"><li>• Release locked destination memory</li><li>• Allocate more destination memory</li><li>• Reduce the frame rate so that there is more time to process the filled destination memory</li></ul>

**3.5.6 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. <b>Possible cause:</b> <ul style="list-style-type: none"><li>• The camera has been disconnected or closed.</li></ul>
ulong	OutOfBuffer	No free internal image memory is available to the driver. The image was discarded. <b>Possible cause/remdy:</b> The computer takes too long to process the images in the uc480 API (e.g. color conversion) <ul style="list-style-type: none"><li>• Reduce the frame rate so that there is more time to process the filled image memory of the driver</li><li>• Disable resource-intensive API image pre-processing functions (e.g. edge enhancement, color correction, choose smaller filter mask for software color conversion)</li></ul>

### 3.5.7 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"> <li>• Reduce the pixel clock frequency</li> <li>• Reduce the frame rate</li> <li>• Reduce the image size</li> </ul>
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"> <li>• Reduce the frame rate</li> <li>• Increase the value for the receive descriptors in the network card settings</li> </ul>

### 3.5.8 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"> <li>• Reduce the pixel clock frequency</li> <li>• Operate fewer cameras simultaneously on a USB bus</li> <li>• Check the quality of the USB cabling and components</li> </ul>
-------	----------------	--

### 3.5.9 ConversionParameter

#### uc480.Types.ConversionParameter

uc480.Defines.ColorCorrectionMode	DstColorCorrectionMode	Sets the color correction, see <a href="#">Correction()</a>
int	DstEdgeEnhancement	Sets the edge enhancement, see <a href="#">EdgeEnhancement</a>
int	DstGamma	Sets the gamma correction, see <a href="#">Software</a>
int	DstMemID	Raw Bayer buffer which was created with <a href="#">Allocate()</a>
uc480.Defines.ColorConvertMode	DstPixelConverter	Conversion mode for the target image; see <a href="#">Converter()</a> for possible modes
uc480.Defines.ColorMode	DstPixelFormat	Color mode of the target image, see <a href="#">PixelFormat</a> for possible modes
int	DstSaturationU	Sets the color saturation (saturation U), see <a href="#">Saturation</a>
int	DstSaturationV	Sets the color saturation (saturation V), see <a href="#">Saturation</a>
int	SrcMemID	Target buffer with the converted data which was created with <a href="#">Allocate()</a>

#### Used in method

- [Convert\(\)](#) (`Image` class)

### 3.5.10 DeviceInfoControl

#### uc480.Types.DeviceInfoControl

int	DeviceID	Device ID of the camera
-----	----------	-------------------------

### 3.5.11 DeviceInfoHeartbeat

#### uc480.Types.DeviceInfoHeartbeat

int	Temperature	Camera temperature
-----	-------------	--------------------

### 3.5.12 DeviceInformation

#### uc480.Types.DeviceInformation

<a href="#"><u>uc480.Types.DeviceInformation.DeviceInfoControl</u></a>	DeviceInfoControl	Camera-related driver data
<a href="#"><u>uc480.Types.DeviceInformation.DeviceInfoHeartbeat</u></a>	DeviceInfoHeartbeat	Camera-related data retrieved from the camera (from the heartbeat telegram)

#### Used in method

- [GetDeviceInfo\(\)](#) (Camera class)
- [GetDeviceInfo\(\)](#) (Information class)

### 3.5.13 ImageFormatInfo

#### uc480.Types.ImageFormatInfo

uc480.Defines.BinningMode	BinningMode	Binning mode used (see <a href="#">GetSupported()</a> )
uc480.Defines.CaptureMode	CaptureMode	<p>Image capture modes supported for this format:</p> <ul style="list-style-type: none"> <li>• uc480.Defines.CaptureMode.Single: Freerun mode, single frame (freerun snap)</li> <li>• uc480.Defines.CaptureMode.Freerun: Freerun mode, continuous (freerun live)</li> <li>• uc480.Defines.CaptureMode.TriggerSoftwareSingle: Software triggered mode, single frame</li> <li>• uc480.Defines.CaptureMode.TriggerSoftwareContinuous: Software triggered mode, continuous</li> <li>• uc480.Defines.CaptureMode.TriggerHardwareSingle: Hardware triggered mode, single frame</li> <li>• uc480.Defines.CaptureMode.TriggerHardwareContinuous: Hardware triggered mode, continuous</li> </ul>
int	FormatID	Format ID of the specified image format (see " <a href="#">Image formats</a> " 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 <a href="#">GetSupported()</a> )

**Image formats of CMOS sensors**

Format ID	Resolution	Name			
			DCC1545/1645	DCC1240/3240	DCC3260
1	3264x2448	(8M)			
2	3264x2176	(8M 3:2)			
3	3264x1836	(8M 16:9)			
4	2592x1944	(5M)			
5	2048x1536	(3M)			
6	1920x1080	(Full HD 16:9)			X
7	1632x1224	(2M)			
8	1280x960	(1.2M 4:3)	X	X	X
9	1280x720	(HD 16:9)	X	X	X
11	960x480	(WVGA 2:1)	X	X	
12	800x480	(WVGA)	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	X
19	240x320	(QVGA Portrait)	X	X	
20	1600x1200	(UXGA)			X
21	3840x2748	(10M)			
22	1920x1080	(Full HD 16:9, HQ)			
23	2560x1920	(5M)			
24	768x576	(CCIR)	X	X	X
25	1280x1024	(1.3M SXGA)	X	X	X
26	2448x2048	(5M)			
27	1024x768	(XGA)	X	X	X
28	1024x1024	(1M)	X		X
29	800x600	(SVGA)	X	X	X
30	1360x1024	(1.4M 4:3)			
35	1920x1200				X
36	Sensor Maximum				1936x1216

## Image formats of CCD sensors

Format ID	Resolution	Name		
			DCU223	DCU224
1	3264x2448	(8M)		
2	3264x2176	(8M 3:2)		
3	3264x1836	(8M 16:9)		
4	2592x1944	(5M)		
5	2048x1536	(3M)		
6	1920x1080	(Full HD 16:9)		
7	1632x1224	(2M)		
8	1280x960	(1.2M 4:3)		X
9	1280x720	(HD 16:9)		X
11	960x480	(WVGA 2:1)	X	X
12	800x480	(WVGA)	X	X
13	640x480	(VGA)	X	X
14	640x360	(VGA 16:9)	X	
15	400x240	(WQVGA)	X	
16	352x288	(CIF)	X	
17	288x352	(CIF Portrait)	X	
18	320x240	(QVGA)	X	
19	240x320	(QVGA Portrait)	X	
20	1600x1200	(UXGA)		
21	3840x2748	(10M)		
22	1920x1080	(Full HD 16:9, HQ)		
23	2560x1920	(5M)		
24	768x576	(CCIR)	X	X
25	1280x1024	(1.3M SXGA)		X
26	2448x2048	(5M)		
27	1024x768	(XGA)	X	X
28	1024x1024	(1M)		X
29	800x600	(SVGA)	X	X
30	1360x1024	(1.4M 4:3)		

### 3.5.14 ImageInfo

#### uc480.Types.ImageInfo

int	AOICycle	Readout cycles (only AOI sequence mode of DCC1240)																								
int	AOIIndex	AOI index (only AOI sequence mode of DCC1240)																								
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 $\mu$ 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																								
System.DateTime	TimestampSystem	Structure with timestamp information in PC system time format <table border="1" data-bbox="822 1163 1425 1731"> <tr> <td>WORD</td><td>wYear</td><td>Timestamp year</td></tr> <tr> <td>WORD</td><td>wMonth</td><td>Timestamp month</td></tr> <tr> <td>WORD</td><td>wDay</td><td>Timestamp day</td></tr> <tr> <td>WORD</td><td>wHour</td><td>Timestamp hour</td></tr> <tr> <td>WORD</td><td>wMinute</td><td>Timestamp minute</td></tr> <tr> <td>WORD</td><td>wSecond</td><td>Timestamp second</td></tr> <tr> <td>WORD</td><td>wMilliseconds</td><td>Timestamp millisecond</td></tr> <tr> <td>WORD</td><td>wReserved[2]</td><td>Reserved</td></tr> </table>	WORD	wYear	Timestamp year	WORD	wMonth	Timestamp month	WORD	wDay	Timestamp day	WORD	wHour	Timestamp hour	WORD	wMinute	Timestamp minute	WORD	wSecond	Timestamp second	WORD	wMilliseconds	Timestamp millisecond	WORD	wReserved[2]	Reserved
WORD	wYear	Timestamp year																								
WORD	wMonth	Timestamp month																								
WORD	wDay	Timestamp day																								
WORD	wHour	Timestamp hour																								
WORD	wMinute	Timestamp minute																								
WORD	wSecond	Timestamp second																								
WORD	wMilliseconds	Timestamp millisecond																								
WORD	wReserved[2]	Reserved																								
ulong	TimestampTick	Internal timestamp of image capture (tick count of the camera in 0.1 $\mu$ s steps)																								

#### Used in methods

- [GetImageInfo\(\)](#) (Information class)

### 3.5.15 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\(\)](#)

### 3.5.16 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\(\)](#)

### 3.5.17 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 $\mu\text{m}$ (e.g. 465 is equivalent to 4.65 $\mu\text{m}$ )
bool	RedGain	Indicates whether the sensor provides analog red channel gain
uc480.Defines.SensorColorMode	SensorColorMode	<p>Returns the sensor color mode.</p> <ul style="list-style-type: none"> <li>• uc480.Defines.SensorColorMode.Bayer</li> <li>• uc480.Defines.SensorColorMode.Monochrome</li> <li>• uc480.Defines.SensorColorMode.CBYCRY <b>(USB uc480 XS only)</b></li> <li>• uc480.Defines.SensorColorMode.Jpeg <b>(XS only)</b></li> </ul>
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	<p>Returns the color of the upper left pixel in the Bayern pattern:</p> <ul style="list-style-type: none"> <li>• uc480.Defines.BayerPixel.Red</li> <li>• uc480.Defines.BayerPixel.Green</li> <li>• uc480.Defines.BayerPixel.Blue</li> </ul>

### Used in method

- [GetSensorInfo\(\)](#) (Information class)

## 3.6 uc480.Types.AutoFeature

- [BrightStatus](#)
- [Information](#)
- [WhitebalanceChannelStatus](#)
- [WhitebalanceStatus](#)

### 3.6.1 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

### 3.6.2 Information

#### uc480.Types.AutoFeature.Information

<a href="#">uc480.Types.AutoFeature.BrightStatus</a>	BrightStatus	Status of automatic brightness control
uc480.Defines.Whitebalance.AntiFlickerMode	AAntiFlickerMode	Returns a bit mask containing all supported anti flicker settings for automatic control: <ul style="list-style-type: none"> <li>• uc480.Defines.Whitebalance.AntiFlickerMode.Disable : Anti flicker mode is disabled.</li> <li>• uc480.Defines.Whitebalance.AntiFlickerMode.SensorAuto : The anti flicker mode is selected automatically (50 or 60 Hz).</li> <li>• uc480.Defines.Whitebalance.AntiFlickerMode.Sensor50Fixed : The anti flicker mode is set to a fixed value of 50 Hz.</li> <li>• uc480.Defines.Whitebalance.AntiFlickerMode.Sensor60Fixed : The anti flicker mode is set to a fixed value of 60 Hz.</li> </ul>
uc480.Defines.Whitebalance.GainPhotomMode	AGainPhotomMode	Returns a bit mask containing all supported photometry settings (fields of view) for auto exposure shutter: <ul style="list-style-type: none"> <li>• uc480.Defines.Whitebalance.GainPhotomMode.None : The entire field of view is used for metering.</li> <li>• uc480.Defines.Whitebalance.GainPhotomMode.CenterWe</li> </ul>



		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"> <li>• uc480.Defines.Whitebalance.WhiteBalanceMode.Disable: Disables the sensor's auto white balance</li> <li>• uc480.Defines.Whitebalance.WhiteBalanceMode.Automatic: Sensor automatically determines auto white balance</li> <li>• 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.</li> <li>• uc480.Defines.Whitebalance.WhiteBalanceMode.IncandescentLamp: Sensor sets auto white balance to incandescent light</li> <li>• uc480.Defines.Whitebalance.WhiteBalanceMode.FluorescentDL: Sensor sets auto white balance to fluorescent light (daylight type)</li> <li>• uc480.Defines.Whitebalance.WhiteBalanceMode.OutdoorClearSky: Sensor sets auto white balance to direct daylight</li> <li>• uc480.Defines.Whitebalance.WhiteBalanceMode.OutdoorCloudy: Sensor sets auto white balance to cloudy sky</li> </ul>
<a href="#"><u>uc480.Types.AutoFeature.WhitebalanceStatus</u></a>	WhitebalanceStatus	Status of auto white balance

## Used in method

- [GetInfo\(\)](#) (AutoFeatures class)

### 3.6.3 WhitebalanceChannelStatus

#### uc480.Types.AutoFeature.WhitebalanceChannelStatus

<code>uc480.Defines.Whitebalance.ControllerStatusMode</code>	Status	Current control status: <ul style="list-style-type: none"> <li>• <code>uc480.Defines.Whitebalance.ControllerStatusMode.Adjusting</code>: Control is active.</li> <li>• <code>uc480.Defines.Whitebalance.ControllerStatusMode.Finished</code>: Control is completed.</li> <li>• <code>uc480.Defines.Whitebalance.ControllerStatusMode.Disabled</code>: Control is disabled.</li> </ul>
<code>int</code>	Error	Current auto white balance error
<code>int</code>	Value	Current average gray-scale value (actual value)

### 3.6.4 WhitebalanceStatus

#### uc480.Types.AutoFeature.WhitebalanceStatus

<code>int</code>	Controller	Current white balance control
<a href="#">WhitebalanceChannelStatus</a>	BlueChannel	See <a href="#">WhitebalanceChannelStatus</a>
<a href="#">WhitebalanceChannelStatus</a>	GreenChannel	See <a href="#">WhitebalanceChannelStatus</a>
<a href="#">WhitebalanceChannelStatus</a>	RedChannel	See <a href="#">WhitebalanceChannelStatus</a>

## 3.7 Complete list of all returns values

#### uc480.Defines.Status

No	Error	Description
-1	<code>IS_NO_SUCCESS</code>	General error message
0	<code>IS_SUCCESS</code>	Function executed successfully
1	<code>IS_INVALID_CAMERA_HANDLE</code>	Invalid camera handle Most of the uc480 SDK functions expect the camera handle as the first parameter.
2	<code>IS_IO_REQUEST_FAILED</code>	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	<code>IS_CANT_OPEN_DEVICE</code>	An attempt to initialize or select the camera failed (no camera connected or initialization error).
11	<code>IS_CANT_OPEN_REGISTRY</code>	Error opening a Windows registry key
12	<code>IS_CANT_READ_REGISTRY</code>	Error reading settings from the

No	Error	Description
		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 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.

No	Error	Description
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 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.

No	Error	Description
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 <sup>2</sup> 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 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.

No	Error	Description
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 4.50

New classes and methods:

- New class in [Trigger](#) class: [Prescaler](#)
- New methods in [Gpio](#) class: [GetConfiguration](#) and [SetConfiguration](#)
- New methods in [Flash](#) class: [GetPIOParamsMin](#) and [SetGPIOParams](#)
- New class and changed methods in [BlackLevel](#) class: [Offset](#)
- New methods in the [BootBoost](#) class: [Wait](#)
- New methods in the [Preset](#) class: [DigitalGain2x](#), [DigitalGain4x](#) and [DigitalGain8x](#)

Added new features:

- New render modes: [Render](#) class and [AutoRender](#) class

Changed classes and methods:

- [Vertical](#) class: [GetSupported](#)
- [Correction](#) class: overloading of methods
- [Factor](#) class: [SetBlue](#), [SetGreen](#), [SetMaster](#), and [SetRed](#)
- [Sequences](#) class: [GetParams\(\)](#) and [SetParams\(\)](#)
- [Binning](#) class: [GetType](#)
- [Subsampling](#) class: [GetType](#)

### New in version 4.40

New classes and methods:

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

Added new features:

- New auto Log mode for camera models DCC1240 and DCC3240

Changes classes and methods:

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

### New in version 4.31

New classes and methods:

- [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 methods in [PixelClock](#) class: [GetList\(\)](#) and [GetNumber\(\)](#)
- New method in [Information](#) class: [GetLastError\(\)](#)
- New events in [Camera](#) class: uc480.Camera.EventAutoFocusFinished and uc480.Camera.EventFirstPacket
- Renamed classes, methods and defines:

	<b>Old name</b>	<b>New name</b>
<b>Class</b>	Model	<a href="#">RgbModel</a>
	AutoFeaturesSensorContrast.GetEnableFDTAOI()	FaceDetectionAoi.GetEnable()
	AutoFeaturesSensorContrast.GetSupportedFDTAoi()	FaceDetectionAoi.GetSupported()
	AutoFeaturesSensorContrast.SetEnableFDTAOI()	FaceDetection.SetEnable()
<b>Defines</b>	uc480.Defines.ColorTemperatureModel	uc480.Defines.ColorTemperatureRgbMode
	uc480.Defines.ColorModelMode	uc480.Defines.ColorTemperatureRgbMode

## New in version 4.22

- New class in the [Display](#) class: [AutoRender](#)
- New classes in the [Feature](#) class: [AoiMerge](#), [Log](#),
- 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

<b>Old class</b>	<b>Replaced by method</b>
Convert	<a href="#">Convert()</a>
OptCmaeraTiming	<a href="#">Optimal()</a>

- Renamed classes:

<b>Class</b>	<b>Old name</b>	<b>New name</b>
uc480.Size	AOI	<a href="#">Size.AOI</a>
	Binning	<a href="#">Size.Binning</a>
	Multi	<a href="#">Size.AOI.Multi</a>
	Subsampling	<a href="#">Size.Subsampling</a>
	Sequences	<a href="#">Size.AOI.Sequence</a>
uc480.Gain	Boost	<a href="#">Gain.Hardware.Boost</a>
	ConvertScaledToFactor	<a href="#">Gain.Hardware.ConvertScaledToFactor</a>
	Factor	<a href="#">Gain.Hardware.Factor</a>
	Hardware	<a href="#">Gain.Hardware</a>
	Scaled	<a href="#">Gain.Hardware.Scaled</a>
uc480.Trigger	BurstSize	<a href="#">Trigger.Burst</a>
	Counter	<a href="#">Trigger.Counter</a>
	Debounce	<a href="#">Trigger.Debounce</a>
	Delay	<a href="#">Trigger.Delay</a>
uc480.Timing	PixelClock	<a href="#">Timing.PixelClock</a>
	Framerate	<a href="#">Timing.Framerate</a>
	VsyncCount	<a href="#">Timing.VsyncCount</a>
	Exposure	<a href="#">Timing.Exposure</a>
	FineIncrement	<a href="#">Timing.Exposure.Fine</a>
	LongExposure	<a href="#">Timing.Exposure.Long</a>
	VsyncCount	<a href="#">Timing.VsyncCount</a>
uc480.Lut	LUT	<a href="#">Lut</a>
	Preset	<a href="#">Lut.Preset</a>
uc480.IO	Flash	<a href="#">IO.Flash</a>
	GPIO	<a href="#">IO.Gpio</a>
	LED	<a href="#">IO.Led</a>
	PWM	<a href="#">IO.Pwm</a>
uc480.Memory	ImageBuffer	<a href="#">Memory.ImageBuffer</a>
	Sequence	<a href="#">Memory.Sequence</a>
uc480.Image	Measure	<a href="#">Image.Measure</a>
uc480.DeviceFeature	LineScan	<a href="#">DeviceFeature.Linescan</a>
	XSHSMode	<a href="#">DeviceFeature.XSHSMode</a>
	ShutterMode	<a href="#">DeviceFeature.Shutter</a>
uc480.Display	Mode	<a href="#">Display.Mode</a>
	Position	<a href="#">Display.Position</a>
uc480.ColorTemperature	RGBColorModel	<a href="#">Color.Temperature.RgbModel</a>
uc480.AutoFeatures	Sensor	<a href="#">AutoFeatures.Sensor</a>
	Software	<a href="#">AutoFeatures.Software</a>
uc480.AutoFeaturesSoftware	WhiteBalance	<a href="#">AutoFeatures.Software.Whitebalance</a>

- Changed methods:

Obsolete methods	New methods
DisplayImage.Set()	<a href="#">Display.Render()</a>
IOLed.SetState()	<a href="#">Led.Get()</a>
IOLed.GetState()	<a href="#">Led.Set()</a>
IOLed.ToggleState()	<a href="#">Led.Toggle()</a>
uc480.IOPwm.GetSupportedGPIOs()	<a href="#">Pwm.SetMode()</a>
CPUIdleState.GetDisableOnOpen_AC()	<a href="#">CPUIdleState.SetDisableOnOpen()</a>
CPUIdleState.GetDisableOnOpen_DC()	<a href="#">CPUIdleState.GetEnable()</a>
CPUIdleState.SetDisableOnOpen_AC()	<a href="#">CPUIdleState.GetDisableOnOpen()</a>
CPUIdleState.SetDisableOnOpen_DC()	
System.GetDLLVersion()	<a href="#">System.GetNetVersion()</a>
	<a href="#">System.GetApiVersion()</a>

- 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 `IOGpio` 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 2018. All rights reserved.

## 5.3 Thorlabs Worldwide Contacts

For technical support or sales inquiries, please visit us at [www.thorlabs.com/contact](http://www.thorlabs.com/contact) for our most up-to-date contact information.

# Index

\*

.NET  
  new in version                                 6  
.NET library                                     11

A

Acquisition	11, 14	set height	70
capture	14	set position	72
finished	16	supported	73
freeze	15	vertical	66
started	16		
stop	16	API version	288
AOI	213	Area of interest	105
auto brightness	219, 222	Auto brightness	
fast position changes	220, 223	AOI	219, 222
fast position changes	223	Auto feature	17
AOI		auto frame rate	18, 26
get	218	auto gain	19, 29
get X position	218	auto reference	32
get Y position	219	auto shutter	22, 34
multi	214	auto skip frame	27
original	219	auto speed	36
position range	220	auto white balance	24
sequence	216	get information	49
set	222	hysteresis	31
size range	221	sensor	17
white balance	222, 223	software	25
AOI merge mode		Auto frame rate	18, 26
additional position	67	enable	26, 27
default	72	get enable	18
default additional position	67	set enable	19
default position	71	supported	18, 27
get	72	Auto gain	19, 29
get additional position	67	default	19
get default height	69	enable	21, 29, 30
get enable	73	maximum	29, 30
get height	69	photometry mode	20, 21
get list	69	state	20
get number	70	supported	20, 30
get position	71	Auto hysteresis	31
height	68	get	31
position	70	range	31
position range	71	set	32
range additional position	68	Auto reference	32
set	74	default	32
set additional position	68	get	32
set enable	74	range	33
		set	33
		Auto render	100
		default mode	101
		get enable	100
		mode	100, 102
		set enable	102
		window	102, 103
		Auto shutter	22, 34
		default	22
		enable	23, 34, 35

Auto shutter	22, 34	set	226
maximum	34, 35	support	225
photometry mode	23, 24	type	226
state	22	Black level	49
supported	23, 35	default	52
Auto skip frame	27	default offset	50
get	28	get	52
range	28	get offset	50
set	28	offset range	50
Auto speed	36	set	53
default	36	set offset	51
get	36	supported	52
range	36	supported offset	51
set	37	BlackLevel	11
Auto white balance	24, 37	Boot boost	274
color model	45, 46, 48	add ID	274
default offset	42	clear list	275
default speed	44	enable	275, 276
disable	45, 48	get list	275
enable	25, 45, 48	remove ID	276
gain	39	set ID	276
gain range	40	wait	277
get hysteresis	41	Burst trigger mode	
get offset	42	default	251
get skip frame	38	get	251
get speed	44	GigE uc480 camera	250
hysteresis	41	range	252
hysteresis range	41	set	252
offset	42	support	252
offset range	43	Bus speed	143
reference range	46		
set hysteresis	42	<b>C</b>	
set offset	43		
set skip frame	39	Camera	270
set speed	45	connected	287
skip frame	38	EEPROM	109
skip frame range	39	exit	11, 271
speed	44	get device info	287
speed range	44	get device number	287
state	24	information	143, 286
supported	25, 46	init	11, 272
type	47, 48	optimal camera timing	249
AutoFeatures	11	parameter set	203
		status	144, 149
		timing	237
<b>B</b>		Camera -> Device	63
Bandwidth		Camera list	287
used	149	Capture	
Binning	223	status	145
factor	224, 225	Capture status	
get	224	reset	149

Color	11	range	108
conversion mode	54, 55, 56	set	109
converter	53	EdgeEnhancement	11
correction	57, 58	EEPROM	11, 109
default temperature	62	read	109
depth	59	write	110
get temperature	62	Error	
temperature	59, 63	get last	148
temperature range	62	Error codes	312
Color temperature model	60	Error report	150
default	61	Exposure	237
get	60	default	241
set	62	enable long time	241
supported	61	fine increment	238, 239
Configuration	273	get	241
memory mode	284	long time	239, 240
Configuring		long time range	240
.NET	10	range	242
Connecting a DCx Camera	9	set	242
Conversion		support	242
apply parameters	138	support long time	240
CPU idle state	277	External interface	
get disable	277	get	64
get enable	278		
set disable	279	<b>F</b>	
supported	278	FaceDetection	11
		Flash	150
Device	11, 63	delay range	152
auto exit	85, 86	duration range	153
get camera ID	85	freerun	152, 156
get device ID	85	global parameter	153
information	146	GPIO	154, 156
set camera ID	86	mode	154, 156
Device feature	64	parameter	151, 154,
AOI merge mode	66		155, 156,
external interface	64	supported GPIO	157
line scan	75	Focus	11
Log mode	77	Frame rate	243
shutter mode	83	default	244
Direct3D	86	get	243
DirectRenderer	11	get current	243
Display	11, 99	range	244
render	106	set	245
		time range	244
<b>E</b>		<b>G</b>	
Edge enhancement	107	Gain	11, 110
default	108	support	123
get	108		

Gain boost	111	
enable	112	I
state	111	
supported	111	I/O
Gain factor	114	150
default blue	115	I2C
default green	115	11
default master	116	I2C bus
default red	116	134
get blue	116	read
get green	116	135
get master	117	write
get red	117	135
set blue	117	Image
set green	118	11
set master	118	acquisition
set red	118	14
Gamma	11, 123	color
software	123	136
General purpose I/O	157	histogram
GigE	11	138
GPIO	157	information
configuration	158, 160	147
direction	158, 161	load
state	159, 162	136, 139
support	160	memory
		284
		RGB value
		139
		save
		136
Hardware gain	111	Image acquisition
Hdr	11	capture
Hot pixel		14
camera correction	128	finished
camera list	126	16
correction	125	freeze
correction mode	133	15
disable correction	132	started
enable correction	132	16
factory list	129	stop
fatcory list	129	Image buffer
merge list	133	182
number	127, 128	disable
sensor correction	134	182, 184
software list	130	enable
user list	126, 127,	iteration
	128, 130,	iteration range
	131, 132	range
Hotpixel	11	183
		supported
		183
		transfer image
		184
		Image convert
		138
		Image save
		140
		Image size
		213
		ImageStabilization
		11
		Information
		11, 142
		bus speed
		143
		Input
		150
		Installation
		8, 9
		IO
		11
		L
		LED
		162
		state
		163
		toggle
		163
		Line scan
		75
		get mode
		75
		get number
		75
		set mode
		76
		set number
		77

Line scan support	75	get pitch height	195 194, 196
supported	76	image inquire	191 197
Log mode default	82	lock	197
default gain	78	locked	195
default manual value	80	pointer	199
gain range	78	size	196
get manual gain	78	to bitmap	198
get manual value	80	unlock	199
get mode	81	width	196
manual gain	77	Memory mode	
manual value	79	default (device ID)	285
range of manual value	80	get (device ID)	284
set manual gain	79	set (device ID)	286
set manual value	81	supported (device ID)	285
set mode	82	Memory mode (device ID)	284
supported	82	Message	200
Lookup table	166	disable	200
LUT	11, 166	enable	200
enable	173, 178	Microsoft DirectX Runtime	86
load	177	Mode	
mode	174, 178	display	104
preset	168, 169, 170, 171, 172, 176, 179	get display mode	104
		set display mode	104
		Multi AOI	214
		axes	215, 216
RAW format	173	disable	215
save	177	support	215
state	174, 175		
supported	175		
user value	177, 179		
value	176	N	
		NET version	288

**M**

Measure		OpenGL	86
AOI	137	OpenMP	282
inquire	136, 137	default	283
preset	136, 137	enable	284
set AOI	136	state	282, 283
Memory	11, 180	Output	150
active	193, 198	Overlay	
allocate	190	clear	88
bits per pixel	194	graphics	90, 92
copy	191	hide	91
copy to array	192	key color	90, 92
copy to bitmap	192	load image	92
free	193	overlay	89
get last	194	position	93
get list	195	release DC	92

**O**

Overlay		enable image scaling	95
show	94	enable scaling	95
size	90, 91, 94	overlay	87
transparency	89, 93	position	96
visible	94	set scaling	97
		steal function	96, 97, 98
		supported modes	96
		synchronization	97
<b>P</b>		update	98
Parameter	11, 203	VSYNC	99
Parameter set	279	window handle	98
clear	203		
get number	203	Return values	312
get supported	204	Ring buffering	185, 186, 187, 188, 189
initial	280		
load	204	Rop effect	209
reset	204	get	209
save	205	set	209
supported	280		
PCs with Energy Saving CPU Technology	321	RopEffect	11
Pixel clock	246	<b>S</b>	
default	247	Saturation	11, 210
discrete	247	default	211
frequency	249	get	210
get	246	range	211
list	247	set	212
number	247	support	212
range	248		
set	248	Scaled gain	118
Pixel format	205	default blue	119
bits per pixel	206	default green	120
bytes per pixel	206	default master	120
get	206	default red	120
set	207	get blue	121
PixelFormat	11	get green	121
Position		get master	121
move	105	get red	121
set	105	set blue	122
Pulse-width modulation	163	set green	122
PWM	163	set master	122
duty cycle	164	set red	123
frequency	164	ScenePreset	11
mode	165, 166	Sensor	
parameter	165, 166	auto feature	17
supported GPIO	165	blue gain channel	112
		gain channels	112
		green gain channel	113
		information	148
Raster operation	209	master gain channel	113
Rendering	86, 105	red gain channel	113
disable scaling	95	Sequence	185

**R**

Raster operation	209		
Rendering	86, 105		
disable scaling	95		

Sequence	185	set	235
active sequence number	186	support	233
add ID	185	supported	235
clear	186	TestImage	11
exit image queue	186	Timeout	11, 236
get last	187	get	236
get sequence number	187	set	236
image memory ID	189	Timing	11, 237
init image queue	188	Trigger	11, 250
lock	188	debounce	254, 255,
locked	187	force	258
unlock	189	mode	259, 260
wait	189	state	259
Sequence AOI	216	support	259
enable	217	Trigger debounce	217, 218
parameter	217, 218	default time	255
state	216	delay time	257
support	217	mode	254, 255,
SeRing buffering	187	time	256
Sharpness	11	time range	256
Shutter mode		Trigger delay	257
get	83	get	257
set	84	range	258
supported	84	set	258
supported mode	83	Trigger mode	28
Size	11	counter	253
image	213	get counter	253
Skip frame		reset counter	253
Software		<b>U</b>	
auto feature	25	uc480	
Software gamma	123	camera	11
default	124	uc480 camera	
get	125	information	142
range	124	uc480 Camera Manager	9
set	124	uc480 Software Installation	9
Subsampling	227	uc480.Types	295
factor	228, 229	AoiSequenceParameter	296
get	228	CameraInfo	297
set	231	CameraInformation	298
supoorted	230	CaptureStatus	298
support	229	CaptureStatusApi	299
type	230	CaptureStatusDriver	299
System		CaptureStatusEth	300
version	288	CaptureStatusUsb	300
System requirements	8	ConversionParameter	301
<b>T</b>		DeviceInfoControl	301
Test image	232		
get	232		
range	233		

uc480.Types	295
DeviceInfoHeartbeat	301
DeviceInformation	302
ImageFormatInfo	303
ImageInfo	306
LutState	307
LutSupportInfo	307
SensorInfo	307
uc480.Types.AutoFeatures	308
BrightStatus	309
Information	309
WhitebalanceChannelStatus	312
WhitebalanceStatus	312

**V**

Video	11, 265, 289
add frame	290
close	291
color mode	294
discarded frames	292
exit	291
frame count	266, 291
frame rate	266, 269, 293
height	294
init	292
lost frames	267
open	293
quality	267, 269, 294
reset count	268
reset counter	293
running	267
size	268, 292, 294
start	269, 295
stop	270, 295
video ID	268
width	294
VSYNC counter	249
get	249

**W**

White balance	
AOI	222, 223

**Z**

Zoom	11
------	----