



CCD and CMOS Cameras

DCU223x, DCU224x

DCC1240x

DCC1545M, DCC1645C

DCC3240X

DCC3260X

DCx Camera DirectShow Programming Interface



2018

Version: 4.81
Date: 7/30/2018

Contents

Foreword	8
1 Welcome	9
1.1 About this manual	9
1.2 What's new in this version?	9
2 Installation and requirements	12
2.1 System Requirements	12
2.2 uc480 Software Installation	13
2.3 Connecting a DCx Camera	14
2.4 Configuring DCx Cameras for DirectShow	15
3 Interfaces	17
3.1 Output pin interfaces	17
3.1.1 IAMStreamConfig	17
3.1.2 IKsPropertySet	20
3.1.3 ISpecifyPropertyPages	20
3.1.4 Iuc480CapturePin	22
3.1.4.1 GetUsedBandwidth	23
3.1.4.2 GetPixelClockRange	23
3.1.4.3 GetPixelClock	24
3.1.4.4 SetPixelClock	24
3.1.4.5 GetRGB8ColorMode	25
3.1.4.6 SetRGB8ColorMode	25
3.1.4.7 GetExposureRange	26
3.1.4.8 GetExposureTime	26
3.1.4.9 SetExposureTime	27
3.1.5 Iuc480Resample	27
3.1.5.1 Subsampling_SetMode	29
3.1.5.2 Subsampling_GetMode	29
3.1.5.3 Subsampling_GetVerticalResolution	29
3.1.5.4 Subsampling_GetHorizontalResolution	30
3.1.5.5 Subsampling_Is2xVertSupported	30
3.1.5.6 Subsampling_Is2xHorSupported	30
3.1.5.7 Subsampling_Is3xVertSupported	31
3.1.5.8 Subsampling_Is3xHorSupported	31
3.1.5.9 Subsampling_Is4xVertSupported	31
3.1.5.10 Subsampling_Is4xHorSupported	32
3.1.5.11 Subsampling_Is5xVertSupported	32
3.1.5.12 Subsampling_Is5xHorSupported	32
3.1.5.13 Subsampling_Is6xVertSupported	33

3.1.5.14	Subsampling_Is6xHorSupported_____	33
3.1.5.15	Subsampling_Is8xVertSupported_____	33
3.1.5.16	Subsampling_Is8xHorSupported_____	34
3.1.5.17	Subsampling_Is16xVertSupported_____	34
3.1.5.18	Subsampling_Is16xHorSupported_____	34
3.1.5.19	Subsampling_IsColorSubsamplingSupported_____	35
3.1.5.20	Binning_SetMode_____	35
3.1.5.21	Binning_GetMode_____	35
3.1.5.22	Binning_GetVerticalResolution_____	36
3.1.5.23	Binning_GetHorizontalResolution_____	36
3.1.5.24	Binning_GetImageWidth_____	36
3.1.5.25	Binning_GetImageHeight_____	37
3.1.5.26	Binning_Is2xVertSupported_____	37
3.1.5.27	Binning_Is2xHorSupported_____	37
3.1.5.28	Binning_Is3xVertSupported_____	38
3.1.5.29	Binning_Is3xHorSupported_____	38
3.1.5.30	Binning_Is4xVertSupported_____	38
3.1.5.31	Binning_Is4xHorSupported_____	39
3.1.5.32	Binning_Is6xVertSupported_____	39
3.1.5.33	Binning_Is6xHorSupported_____	39
3.1.5.34	Binning_IsColorBinningSupported_____	40
3.1.6	luc480Scaler _____	40
3.1.6.1	GetSensorScalerInfo_____	40
3.1.6.2	SetSensorScaler_____	41
3.1.6.3	GetScalerImageWidth_____	42
3.1.6.4	GetScalerImageHeight_____	42
3.1.6.5	SetImageSize_____	43
3.2	Filter interfaces _____	43
3.2.1	IAMDroppedFrames _____	44
3.2.2	IAMFilterMiscFlags _____	45
3.2.3	IAMVideoControl _____	45
3.2.4	IAMVideoProcAmp _____	46
3.2.5	IKsPropertySet _____	46
3.2.6	ISpecifyPropertyPages _____	47
3.2.7	luc480AOI _____	53
3.2.7.1	AOI_IsImageAOISupported_____	54
3.2.7.2	AOI_GetImageAOI_____	55
3.2.7.3	AOI_SetImageAOI_____	55
3.2.7.4	AOI_GetAutoBrightnessAOI_____	56
3.2.7.5	AOI_SetAutoBrightnessAOI_____	56
3.2.7.6	AOI_GetAutoWBAOI_____	57
3.2.7.7	AOI_SetAutoWBAOI_____	57

3.2.7.8	AOI_GetIncPosX_____	58
3.2.7.9	AOI_GetIncPosY_____	58
3.2.7.10	AOI_GetIncSizeX_____	58
3.2.7.11	AOI_GetIncSizeY_____	59
3.2.7.12	AOI_GetMinMaxPosX_____	59
3.2.7.13	AOI_GetMinMaxPosY_____	59
3.2.7.14	AOI_GetMinMaxSizeX_____	60
3.2.7.15	AOI_GetMinMaxSizeY_____	60
3.2.8	luc480AutoFeatures _____	61
3.2.8.1	SetAutoBrightnessReference_____	62
3.2.8.2	GetAutoBrightnessReference_____	63
3.2.8.3	SetAutoBrightnessMaxExposure_____	63
3.2.8.4	GetAutoBrightnessMaxExposure_____	64
3.2.8.5	SetAutoBrightnessMaxGain_____	64
3.2.8.6	GetAutoBrightnessMaxGain_____	64
3.2.8.7	SetAutoBrightnessSpeed_____	65
3.2.8.8	GetAutoBrightnessSpeed_____	65
3.2.8.9	SetAutoBrightnessAOI_____	65
3.2.8.10	GetAutoBrightnessAOI_____	66
3.2.8.11	SetAutoWBGainOffsets_____	66
3.2.8.12	GetAutoWBGainOffsets_____	67
3.2.8.13	SetAutoWBGainRange_____	67
3.2.8.14	GetAutoWBGainRange_____	68
3.2.8.15	SetAutoWBSpeed_____	68
3.2.8.16	GetAutoWBSpeed_____	68
3.2.8.17	SetAutoWBAOI_____	69
3.2.8.18	GetAutoWBAOI_____	69
3.2.9	luc480AutoFramerate _____	70
3.2.9.1	AutoFramerateSensor_IsSupported_____	70
3.2.9.2	AutoFramerateSensor_IsEnabled_____	70
3.2.9.3	AutoFramerateSensor_Enable_____	71
3.2.9.4	AutoFramerateDriver_IsSupported_____	71
3.2.9.5	AutoFramerateDriver_IsEnabled_____	72
3.2.9.6	AutoFramerateDriver_Enable_____	72
3.2.9.7	AutoFramerate_GetFramerate_____	72
3.2.10	luc480AutoParameter _____	73
3.2.10.1	AutoParameter_GetAWBType_____	73
3.2.10.2	AutoParameter_SetAWBType_____	74
3.2.10.3	AutoParameter_GetSupportedAWBType_____	74
3.2.10.4	AutoParameter_GetEnableAWB_____	75
3.2.10.5	AutoParameter_SetEnableAWB_____	75
3.2.10.6	AutoParameter_GetRGBColorModelAWB_____	75

3.2.10.7	AutoParameter_SetRGBColorModelAWB_____	76
3.2.10.8	AutoParameter_GetSupportedRGBColorModelAWB_____	76
3.2.11	luc480CameraLUT _____	77
3.2.11.1	CameraLUT_GetCameraLUT_____	77
3.2.11.2	CameraLUT_SetCameraLUT_____	78
3.2.12	luc480Capture _____	80
3.2.12.1	GetDeviceInfo_____	80
3.2.12.2	GetDLLVersion_____	81
3.2.12.3	HotPixel_____	82
3.2.12.4	SetBadPixelCorrection_____	84
3.2.12.5	GetBadPixelCorrection_____	84
3.2.12.6	SaveSettings_____	84
3.2.12.7	LoadSettings_____	85
3.2.12.8	ResetDefaults_____	85
3.2.12.9	GetWhiteBalanceMultipliers_____	86
3.2.12.10	SetWhiteBalanceMultipliers_____	86
3.2.13	luc480CaptureEx _____	86
3.2.13.1	SetGainBoost_____	87
3.2.13.2	GetGainBoost_____	87
3.2.13.3	SetHardwareGamma_2_____	88
3.2.13.4	GetHardwareGamma_2_____	88
3.2.13.5	LoadParameters_____	89
3.2.13.6	SaveParameters_____	89
3.2.13.7	ParameterSet_____	90
3.2.14	luc480ColorConverter _____	91
3.2.14.1	ColorConverter_GetCurrentMode_____	92
3.2.14.2	ColorConverter_GetDefaultMode_____	92
3.2.14.3	ColorConverter_GetSupportedModes_____	93
3.2.14.4	ColorConverter_SetMode_____	93
3.2.15	luc480ColorTemperature _____	94
3.2.15.1	RGBModel_IsSupported_____	94
3.2.15.2	RGBModel_GetMode_____	95
3.2.15.3	RGBModel_SetMode_____	95
3.2.15.4	RGBModel_GetDefaultMode_____	96
3.2.15.5	RGBModel_GetSupportedModes_____	96
3.2.15.6	ColorTemperature_IsSupported_____	97
3.2.15.7	ColorTemperature_GetValue_____	97
3.2.15.8	ColorTemperature_SetValue_____	97
3.2.15.9	ColorTemperature_GetDefaultValue_____	98
3.2.15.10	ColorTemperature_GetRange_____	98
3.2.16	luc480DeviceFeature _____	99
3.2.16.1	DeviceFeature_GetSupportedFeatures_____	101

3.2.16.2	DeviceFeature_GetAllowRawWithLUT_____	103
3.2.16.3	DeviceFeature_SetAllowRawWithLUT_____	103
3.2.16.4	DeviceFeature_GetDefaultLogMode_____	103
3.2.16.5	DeviceFeature_GetLogMode_____	104
3.2.16.6	DeviceFeature_SetLogMode_____	105
3.2.16.7	DeviceFeature_GetLogModeManualGain_____	105
3.2.16.8	DeviceFeature_GetLogModeManualGainDefault_____	106
3.2.16.9	DeviceFeature_GetLogModeManualGainRange_____	106
3.2.16.10	DeviceFeature_SetLogModeManualGain_____	107
3.2.16.11	DeviceFeature_GetLogModeManualValue_____	108
3.2.16.12	DeviceFeature_GetLogModeManualValueDefault_____	108
3.2.16.13	DeviceFeature_GetLogModeManualValueRange_____	109
3.2.16.14	DeviceFeature_SetLogModeManualValue_____	109
3.2.16.15	DeviceFeature_GetLineScanMode_____	110
3.2.16.16	DeviceFeature_SetLineScanMode_____	110
3.2.16.17	DeviceFeature_GetLineScanNumber_____	111
3.2.16.18	DeviceFeature_SetLineScanNumber_____	111
3.2.16.19	DeviceFeature_GetShutterMode_____	112
3.2.16.20	DeviceFeature_SetShutterMode_____	112
3.2.16.21	DeviceFeature_GetVerticalAOIMergeMode_____	113
3.2.16.22	DeviceFeature_SetVerticalAOIMergeMode_____	113
3.2.16.23	DeviceFeature_GetVerticalAOIMergePosition_____	114
3.2.16.24	DeviceFeature_SetVerticalAOIMergePosition_____	114
3.2.17	luc480EdgeEnhancement_____	115
3.2.17.1	EdgeEnhancement_GetEdgeEnhancement_____	115
3.2.17.2	EdgeEnhancement_GetEdgeEnhancementDefault_____	115
3.2.17.3	EdgeEnhancement_GetEdgeEnhancementRange_____	116
3.2.17.4	EdgeEnhancement_SetEdgeEnhancement_____	116
3.2.18	luc480Event_____	117
3.2.18.1	InitEvent_____	117
3.2.18.2	EnableEvent_____	118
3.2.18.3	DisableEvent_____	120
3.2.18.4	ExitEvent_____	120
3.2.18.5	EnableMessage_____	120
3.2.19	luc480Flash_____	122
3.2.19.1	Flash_SetStrobeMode_____	122
3.2.19.2	Flash_GetStrobeMode_____	123
3.2.19.3	Flash_GetDuration_____	124
3.2.19.4	Flash_GetDurationRange_____	124
3.2.19.5	Flash_SetDelayDuration_____	124
3.2.19.6	Flash_GetDelay_____	125
3.2.19.7	Flash_GetDelayRange_____	125

3.2.19.8	Flash_GetGlobalExposureWindow_____	126
3.2.19.9	Flash_GetSupportedGPIOPorts_____	126
3.2.19.10	Flash_EnableGPIOPort_____	126
3.2.20	luc480Gain _____	127
3.2.20.1	Gain_IsMasterSupported_____	128
3.2.20.2	Gain_IsRGBSupported_____	128
3.2.20.3	Gain_GetHwGain_____	129
3.2.20.4	Gain_SetHwGain_____	129
3.2.20.5	Gain_GetHwGainDefaults_____	130
3.2.20.6	Gain_GetHwGainRange_____	130
3.2.20.7	Gain_GetHwGainFactor_____	131
3.2.20.8	Gain_SetHwGainFactor_____	131
3.2.20.9	Gain_GetHwGainFactorDefaults_____	132
3.2.20.10	Gain_InquireHwGainFactor_____	132
3.2.20.11	Gain_GetHwGainFactorRange_____	133
3.2.20.12	Gain_IsGainBoostSupported_____	133
3.2.20.13	Gain_GetGainBoostValue_____	133
3.2.20.14	Gain_SetGainBoostValue_____	134
3.2.21	luc480HotPixel _____	134
3.2.21.1	HotPixel_DeleteCameraUserList_____	136
3.2.21.2	HotPixel_DisableCorrection_____	136
3.2.21.3	HotPixel_EnableCameraCorrection_____	137
3.2.21.4	HotPixel_EnableSoftwareUserCorrection_____	137
3.2.21.5	HotPixel_GetCameraFactoryList_____	137
3.2.21.6	HotPixel_GetCameraFactoryListExist_____	138
3.2.21.7	HotPixel_GetCameraFactoryListNumber_____	138
3.2.21.8	HotPixel_GetCameraUserList_____	139
3.2.21.9	HotPixel_GetCameraUserListExist_____	139
3.2.21.10	HotPixel_GetCameraUserListMaxNumber_____	140
3.2.21.11	HotPixel_GetCameraUserListNumber_____	140
3.2.21.12	HotPixel_GetCorrectionMode_____	141
3.2.21.13	HotPixel_GetMergedCameraList_____	141
3.2.21.14	HotPixel_GetMergedCameraListNumber_____	142
3.2.21.15	HotPixel_GetSoftwareUserList_____	142
3.2.21.16	HotPixel_GetSoftwareUserListExist_____	143
3.2.21.17	HotPixel_GetSoftwareUserListNumber_____	143
3.2.21.18	HotPixel_GetSupportedCorrectionModes_____	143
3.2.21.19	HotPixel_LoadUserList_____	144
3.2.21.20	HotPixel_LoadUserListUnicode_____	144
3.2.21.21	HotPixel_SaveUserList_____	145
3.2.21.22	HotPixel_SaveUserListUnicode_____	146
3.2.21.23	HotPixel_SetCameraUserList_____	146

3.2.21.24	HotPixel_SetSoftwareUserList_____	147
3.2.21.25	HotPixel_SensorCorrection_____	147
3.2.22	luc480IO _____	148
3.2.22.1	IO_SetGPIO_____	148
3.2.22.2	IO_GetGPIO_____	149
3.2.22.3	IO_SetIOMask_____	149
3.2.22.4	IO_GetIOMask_____	150
3.2.22.5	IO_IOMaskInputSupported_____	150
3.2.22.6	IO_IOMaskOutputSupported_____	151
3.2.23	luc480Trigger _____	151
3.2.23.1	Trigger_GetBurstSize_____	152
3.2.23.2	Trigger_GetBurstSizeRange_____	153
3.2.23.3	Trigger_GetBurstSizeSupported_____	153
3.2.23.4	Trigger_SetBurstSize_____	154
3.2.23.5	Trigger_GetTriggerMode_____	154
3.2.23.6	Trigger_SetTriggerMode_____	154
3.2.23.7	Trigger_GetTriggerStatus_____	155
3.2.23.8	Trigger_IsFallingEdgeSupported_____	155
3.2.23.9	Trigger_IsRisingEdgeSupported_____	156
3.2.23.10	Trigger_IsSoftwareTriggerSupported_____	156
4	Troubleshooting	157
5	Appendix	158
5.1	PCs with Energy Saving CPU Technology _____	158
5.2	Exclusion of Liability and Copyright _____	160
5.3	Thorlabs Worldwide Contacts _____	160

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 Welcome

Thank you for purchasing a DCx Camera!

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

Important information

- [What's new in this version?](#)
- [Configuring DCx Cameras for DirectShow](#)
- [Supported Interfaces](#)

Example program

The uc480 software package includes the uc480 DirectShow Demo example program, which shows how DCx Cameras are integrated using the DirectShow interface. The source code of the program is included as a C++ project for Visual Studio 2005.

Enjoy your new DCx Camera!

1.1 About this manual

The DCx DirectShow Manual contains all the information you need for programming your own applications with your DCx Camera and the DirectShow API. The uc480 DirectShow 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 and a Software Development Kit (SDK) for creating your own uc480 programs in Windows. Demo applications make it easier to start uc480 programming.

For detailed information on your DCx Camera please refer to the DCx Camera Manual.

1.2 What's new in this version?

Version 4.81 of the uc480 DirectShow interface contains several new features and improvements. The key new functions are listed in the table below.

New in version 4.81

Cameras & functions	Description in chapter
New interface for long exposure and querying discrete pixel clocks	Output Pin Interfaces: <code>Iuc480CapturePinEx</code>
New interface for OpenMP support	Filter Interfaces: <code>Iuc480Configuration</code>



Note for version 4.81

After updating the uc480 DirectShow interface, compile your program again to avoid problems with the binary compatibility.

New in version 4.60

Cameras & functions	Description in chapter
New interfaces for fully supporting the	<code>Iuc480AntiFlicker</code> <code>Iuc480AutoBacklight</code> <code>Iuc480AutoContrast</code> <code>Iuc480DigitalZoom</code>

	Iuc480FaceDetection Iuc480ImageStabilization Iuc480Photometry Iuc480Saturation Iuc480Sharpness
--	------------------------------------------------------------------------------------------------------------

New in version 4.41

Cameras & functions	Description in chapter
Interface enhancements for AOI functions	Filter Interfaces: Iuc480AOI
Added image formats and XS functions to ISpecificPropertyPages	ISpecifyPropertyPages

New in version 4.32

Cameras & functions	Description in chapter
New interface for color conversion	Filter Interfaces: Iuc480ColorConverter
Interface enhancements to control the lens correction of the XS	Filter Interfaces: Iuc480ColorTemperature
Interface enhancements to control specific functions of the XS	Filter Interfaces: Iuc480DeviceFeature
New interface for controlling the auto functions	Filter Interfaces: Iuc480AutoParameter
New interface for controlling the camera LUT	Filter Interfaces: Iuc480CameraLUT
New interface for controlling camera-specific functions	Filter Interfaces: Iuc480DeviceFeature
New interface for using the software edge filter	Filter Interfaces: Iuc480EdgeEnhancement
New interface for the correction of the sensor's hot pixels	Filter Interfaces: Iuc480HotPixel
New functions for setting the burst trigger mode are added	Filter Interfaces: Iuc480Trigger

New in version 4.00

Cameras & functions	Description in chapter
New interface for setting events	Filter Interfaces: Iuc480Event
New interface for setting trigger and GPIO	Filter Interfaces: Iuc480IO

New in version 3.90

Cameras & functions	Description in chapter
The maximum number of cameras is 24 now.	
The DirectShow service is installed and activated via the setup when you choose the complete installation.	Configuring DCx camera for DirectShow
New HotPixel function integrated	Iuc480Capture
New interface for setting the scaler with the functions GetSensorScaler and SetSensorScaler	Iuc480Scaler
New "Scaler" tab in the property pages	Interfaces: ISpecifyPropertyPages

When opening a camera a in the camera saved parameter set can be loaded automatically.	Filter interfaces: ISpecifyPropertyPages
----------------------------------------------------------------------------------------	----------------------------------------------------------

New in version 3.82

Cameras & functions	Description in chapter
New interface for setting an area of interest (AOI)	Iuc480AOI
New interface for setting analog sensor gain	Iuc480Gain
Support for VideoProcAmp_Brightness and VideoProcAmp_Contrast	IAMVideoProcAmp
Support for GetWhiteBalanceMultipliers and SetWhiteBalanceMultipliers	Iuc480Capture

New in version 3.80

Cameras & functions	Description in chapter
New Interface for setting a color temperature and a color space	Iuc480ColorTemperature
New Interface for supporting the flash function	Iuc480Flash
New Interface for setting the trigger signal (falling edge, rising edge, software trigger)	Iuc480Trigger
New Interface for supporting binning/subsampling	Iuc480Resample
New Interface for activating the automatic frame rate control	Iuc480AutoFramerate
Automatic registration of connected cameras	Configuring DCx cameras for DirectShow
Registration of the connected cameras with a camera name and serial number	Configuring DCx cameras for DirectShow
Automatic image mirroring	Configuring DCx cameras for DirectShow
Information in the manual	Description in chapter
Grouping all functions by the applicable interfaces	Interfaces

2 Installation and requirements

The DirectShow interface

DirectShow is part of the Windows Platform SDK from Microsoft and describes a generic programming interface for audio and video devices which is not specific to any manufacturer. It replaces the Video for Windows (VfW) interface.

The uc480 interface for DirectShow makes it possible to use DCx Cameras in DirectShow-based applications. The uc480 DirectShow interface is a DirectShow Video Capture Filter. This means that you do not have to replace the DCx Camera driver, which is necessary with the WDM Stream Class drivers. The DirectShow interface uses the standard camera driver.

Attention

The uc480 DirectShow interface does not work with programs that only support the VfW interface.

Note

The designation WDM (Windows Driver Model) driver only means that a driver was programmed using the Windows driver model standardized by Microsoft. A WDM Stream Class driver that supports kernel streaming is linked to DirectShow via the Microsoft `ksproxy` filter.

2.1 System Requirements

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

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

Drivers for network cards

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

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

USB interface

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

Large multi-camera systems

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

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



Note on color cameras with high frame rates

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

Direct3D graphics functions

The uc480 driver can use Direct3D to display the camera image with overlay information (Microsoft DirectX Runtime had to be installed). On Windows systems, you can use the supplied "DXDiag" diagnostic tool to check whether your graphics card supports Direct3D functions. To start the diagnostic tool, click "Run..." on the Windows start menu (shortcut: Windows+R) and enter "DXDiag" in the input box.

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

OpenGL graphics functions

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

2.2 uc480 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

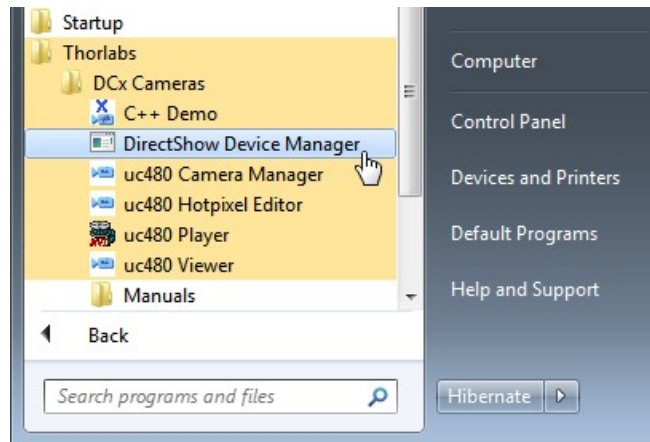
To use maximum bandwidth, we recommend connecting the cameras directly to the USB ports on the mainboard. Many USB ports on PC/PCIe cards and the USB ports on the front of the PC often supply lower bandwidth.

Attention

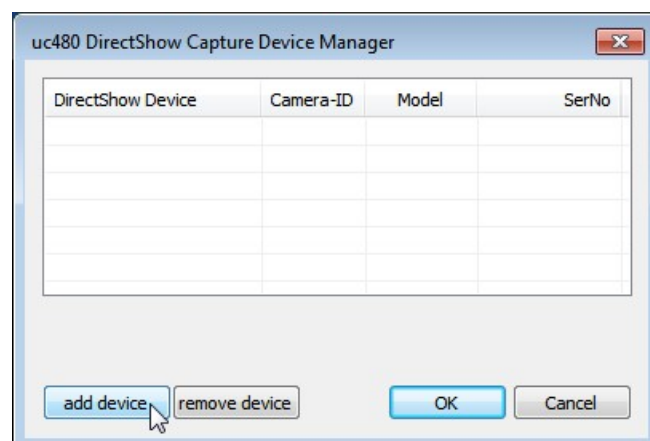
USB cables with non-standard connectors must be connected to the camera first and then to the PC. Otherwise the camera might not be recognized correctly.

2.4 Configuring DCx Cameras for DirectShow

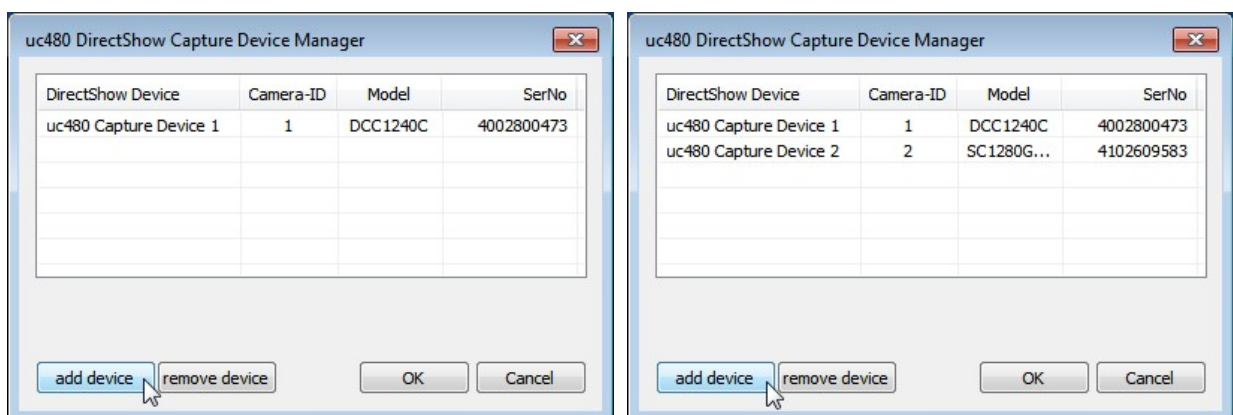
Each camera to be used in DirectShow must first be registered using the *DllRegisterServer* operating system function. This registration can be completed by calling the DirectShow Device Capture Manager:



or by entering at the command prompt: `C:\WINDOWS\system32\regsvr32.exe uc480Capture.ax`



Click to "add device" in order to add cameras, when finished, click OK:



Note

Up to 24 cameras can be used simultaneously in DirectShow.

To use cameras in DirectShow, you must first assign each camera a unique ID (identification

number). DCx camera IDs are issued in *uc480 camera manager* using the "Camera Information" function. Valid IDs for use in DirectShow must meet the following requirements:

- Each DCx camera that is connected must have a unique ID.
- The IDs must be a number from 1 through 8. Up to eight cameras can be used simultaneously in DirectShow.

On installing the DirectShow interface, one camera is automatically registered with the ID 1. Additional cameras can be added using the *add device* button.

An entry entitled `uc480 Capture Device <Camera ID>` is displayed for each of the registered cameras (see above). These entries are also present if no cameras are connected to the PC.

Attention

If a camera that is not present is addressed in DirectShow using `IMoniker->BindToStorage`, the error `VFW_E_NO_CAPTURE_HARDWARE` is returned.

Note

64 bit operating systems If you want to use cameras in both 32 and 64 bit operating systems, you also have to manually register the cameras in the 32 bit subsystem. For this, use the 32 bit version of the DirectShow device manager. You can find the 32 bit DirectShow device manager in the Windows start menu.

Registering a camera using the command line

Registration can also be done using the command line or a batch file. To do this, you must first assign each camera a unique ID (identification number - see above). Next, call up the *regsvr32.exe* system program with the following parameters:

```
regsvr32.exe <Path> /s /n /i:<No>
```

where `<Path>` is the path of the `uc480Capture.ax` file (usually `C:\Windows\System32\uc480Capture.ax`) and `<No>` is the number of cameras to be registered. If the number entered for `<No>` is less than the number of cameras that are currently registered, the cameras above and beyond this number are un-registered.

Addressing DCx cameras in DirectShow

DCx cameras are listed in the *Video Capture Device Filter* category in DirectShow. If manual camera registration (see above) is used, an entry entitled `uc480 Capture Device <Camera ID>` is always displayed for each device. If automatic registration is used, the entry displays the camera name and serial number in the format `<Camera name>_<Serial number>`.

Note

Changes to the camera list (e.g. newly-connected cameras) are only visible once the DirectShow application has been re-started. This is also the case when using automatic registration.

Attention

Mirrored live image

Some DirectShow-based programs interpret the image height as a negative value, causing the live image to be displayed in mirrored format. In such cases, the "use bottom-up images" option can be activated on the Device page via the [Filter Property Pages](#). This displays the images correctly again.

3 Interfaces

DirectShow functions for cameras are classified as either [output pin interfaces](#) or [filter interfaces](#).

Syntax

Prototype of the function from the `uc480CaptureInterface.h` header file.

Description

Description of the function

Parameters

Description of the function parameters including their value ranges

Interface

Name of the interface to which the function belongs

Related functions

List with similar or related functions

Note

Some functions use structures which are not defined in the `uc480CaptureInterface.h` file. These structures can be looked up in the `uc480.h` file. As the `uc480CaptureInterface.h` file, the `uc480.h` file can be found in the `<Installation directory>\uc480\Develop\include` directory.

3.1 Output pin interfaces

IAMCameraControl	This interface can be used to set the uc480's exposure time and autofocus.
IAMStreamConfig	This interface can be used to query and set video stream parameters such as the color format, image size, and frame rate.
IKsPropertySet	This interface can be used to query the pin category.
ISpecifyPropertyPages	This interface can be used to access property pages on which camera parameters can be graphically set.
Iuc480CapturePin	Special functions for the uc480 camera timing.
Iuc480Resample	Special functions for subsampling and binning.
Iuc480Scaler	Special function for setting the internal scaler on some uc480 models.

3.1.1 IAMStreamConfig

IID_IAMStreamConfig

This interface can be used to query and set video stream parameters such as the color format, image size, and frame rate.



Setting the stream format

The `IAMStreamConfig::SetFormat` function must not be accessed with a running or paused filter. If this is done, the error `VFW_E_NOT_STOPPED` will be returned. The filter graph should be stopped before the stream format is changed.

**Using binning/subsampling and setting AOI via IAMStreamConfig**

The following limitations apply when setting binning/subsampling and AOI via IAMStreamConfig. For this reason we recommend to set binning/subsampling and AOI using the uc480 specific Interfaces [Iuc480Resample](#) and [Iuc480AOI](#).

- DirectShow does not differentiate between binning and subsampling. If a uc480 camera supports both functions, binning is always enabled.
- Binning and subsampling cannot be used with an AOI. If an AOI is set, binning and subsampling cannot be enabled (and vice versa).
- Binning and subsampling are only possible with the 2x factor.

**Setting the frame rate**

The possible frame rate of a camera depends on the set [pixel clock](#).

Setting an area of interest (AOI) using the MediaType

To set an AOI with an offset (xOff, yOff), rcSource must be correspondingly set in the media type's VIDEOINFOHEADER. When doing so, it is important for

- the width and height of the rcSource to match the fields biWidth and biHeight in the BITMAPINFOHEADER
- the width and height to be less than/equal to the minimum image dimensions
- the two RECT structures to be fully positioned within the maximum image area
- the vertices of the RECT structures to be correctly aligned

The minimum and maximum image dimensions and alignment requirements can be determined via IAMStreamConfig::GetStreamCaps(). Information acquired via GetStreamCaps() always relates to the specified media type. However, the data used for the image section are generally applicable and can be used for all media types.

Example: Setting an AOI with an offset

```
HRESULT SetAOI(IUnknown* pUnknown)
{
    /* we want to set an AOI of 100 x 100 pixels at offset (10, 10) */
    int iAOI_Width = 100;
    int iAOI_Height = 100;
    int iAOI_PosX = 10;
    int iAOI_PosY = 10;

    HRESULT hr = S_OK;
    IAMStreamConfig* pStreamCfg = NULL;

    /* query the filter pin's IID_IAMStreamConfig interface */
    hr = pUnknown->QueryInterface(IID_IAMStreamConfig, reinterpret_cast<void**>(&pStreamCfg));
    if (FAILED(hr))
    {
        /* return with error */
        return hr;
    }

    AM_MEDIA_TYPE* pmt= NULL;

    #if defined WITH_VALIDATION
    /* query the capabilities and value ranges for the media type with index 0.
     * \note we are not interested in the media type itself, we are interested in
     * the generic capabilities that apply to all supported media types.
     */
    VIDEO_STREAM_CONFIG_CAPS vscc;
    ZeroMemory(&vscc, sizeof(VIDEO_STREAM_CONFIG_CAPS));
```

```

hr = pStreamCfg->GetStreamCaps(0, &pmt, (BYTE*)&vscc);
if (FAILED(hr))
{
    /* release the IID_IAMStreamConfig interface */
    pStreamCfg->Release();
    pStreamCfg = NULL;
    /* return with error */
    return hr;
}
/* the media type 0 is not needed here */
DeleteMediaType(pmt);
pmt = NULL;

/* maximum dimensions of the AOI */
int iMaxWidth = vscc.InputSize.cx;
int iMaxHeight = vscc.InputSize.cy;
/* minimum dimensions of the AOI */
int iMinWidth = vscc.MinCroppingSize.cx;
int iMinHeight = vscc.MinCroppingSize.cy;
/* step width for the AOI dimensions (not needed here, just for demonstration) */
int iSteppingX = vscc.CropGranularityX;
int iSteppingY = vscc.CropGranularityY;
/* required alignment for the AOI coordinates */
int iAlignX = vscc.CropAlignX;
int iAlignY = vscc.CropAlignY;

/* check our AOI to be acceptable for the pin */
if (iAOI_Width > iMaxWidth || iAOI_Width < iMinWidth)
{
    /* release the IID_IAMStreamConfig interface */
    pStreamCfg->Release();
    pStreamCfg = NULL;
    /* return with error */
    return E_INVALIDARG;
}
if (iAOI_Height > iMaxHeight || iAOI_Height < iMinHeight)
{
    /* release the IID_IAMStreamConfig interface */
    pStreamCfg->Release();
    pStreamCfg = NULL;
    /* return with error */
    return E_INVALIDARG;
}
if ((iAOI_PosX + iAOI_Width) > iMaxWidth)
{
    /* release the IID_IAMStreamConfig interface */
    pStreamCfg->Release();
    pStreamCfg = NULL;
    /* return with error */
    return E_INVALIDARG;
}
if ((iAOI_PosY + iAOI_Height) > iMaxHeight)
{
    /* release the IID_IAMStreamConfig interface */
    pStreamCfg->Release();
    pStreamCfg = NULL;
    /* return with error */
    return E_INVALIDARG;
}

/* force alignment */
iAOI_Width &= ~(iAlignX - 1);
iAOI_Height &= ~(iAlignY - 1);
iAOI_PosX &= ~(iAlignX - 1);
iAOI_PosY &= ~(iAlignY - 1);
#endif /* defined WITH_VALIDATION */

/* query the currently active media type from the pin.
* \note we want to change this media type to define our AOI.

```

```
*/
hr = pStreamCfg->GetFormat(&pmt);
if (FAILED(hr))
{
    /* release the IID_IAMStreamConfig interface */
    pStreamCfg->Release();
    pStreamCfg = NULL;
    /* return with error */
    return hr;
}

/* change the media type to define our AOI */
VIDEOINFOHEADER* pvih = (VIDEOINFOHEADER*)pmt->pbFormat;
/* set the source RECT to the AOI coordinates */
SetRect(&pvih->rcSource, iAOI_PosX, iAOI_PosY, iAOI_PosX + iAOI_Width, iAOI_PosY + iAOI_Height);
/* adjust the width and height fields of the media type's BITMAPINFOHEADER to reflect the AOI's size */
pvih->bmiHeader.biWidth = iAOI_Width;
pvih->bmiHeader.biHeight = iAOI_Height;
/* \note .biWidth, .biHeight and rcSource must be consistently
 * specified when we set the changed format to the filter.
 * The filter implementation will adjust the remaining format
 * related or depending fields of the media type to meet these.
 */

/* set the new format to the pin */
hr = pStreamCfg->SetFormat(pmt);

/* release the media type */
pvih = NULL;
DeleteMediaType(pmt);
pmt = NULL;

/* release the IID_IAMStreamConfig interface */
pStreamCfg->Release();
pStreamCfg = NULL;

return hr;
}
```

3.1.2 IKsPropertySet

IID_IKsPropertySet

This interface can be used to query the pin category. The uc480 DirectShow interface supports the following parameter of `IKsPropertySet`:

Get	Returns the pin category (with the uc480 DirectShow interface, it is always <code>PIN_CATEGORY_CAPTURE</code>)
-----	-----------------------------------------------------------------------------------------------------------------

3.1.3 ISpecifyPropertyPages

IID_ISpecifyPropertyPages

This interface can be used to access property pages on which camera parameters can be graphically set. The property pages contain all the settings that are contained in the "Output pin interfaces".

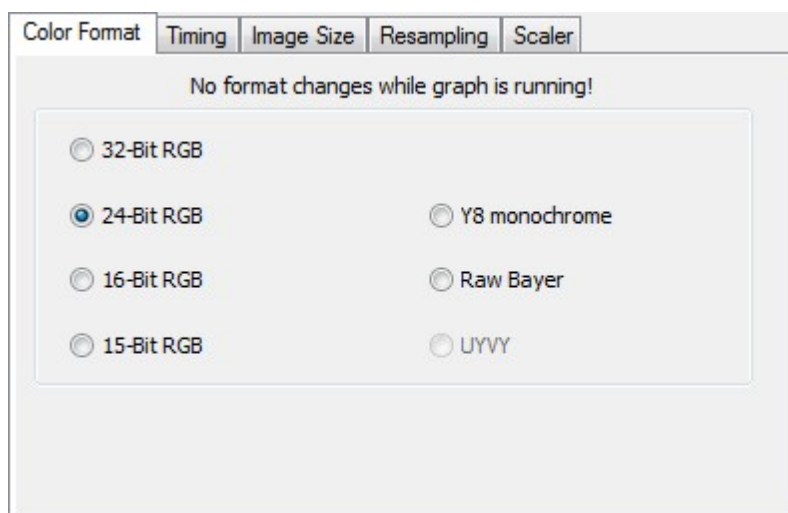


Fig. 1: Properties of the uEye Capture pin: Color Format

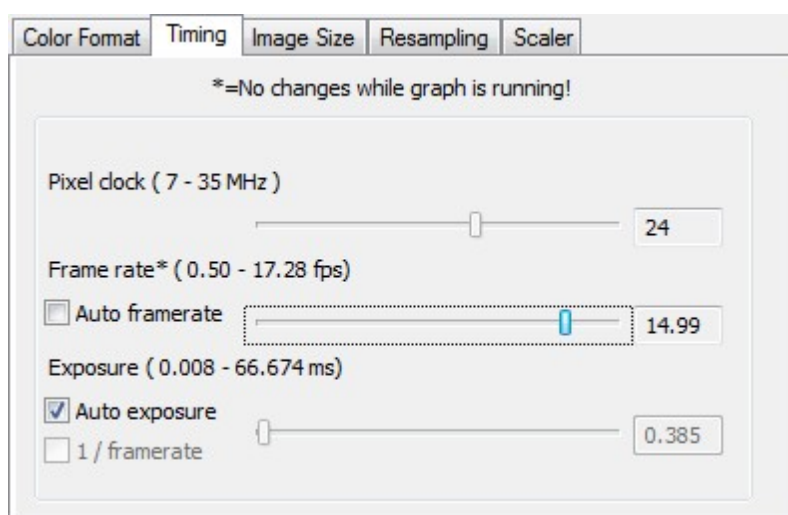


Fig. 2: Properties of the uEye Capture pin: Timing

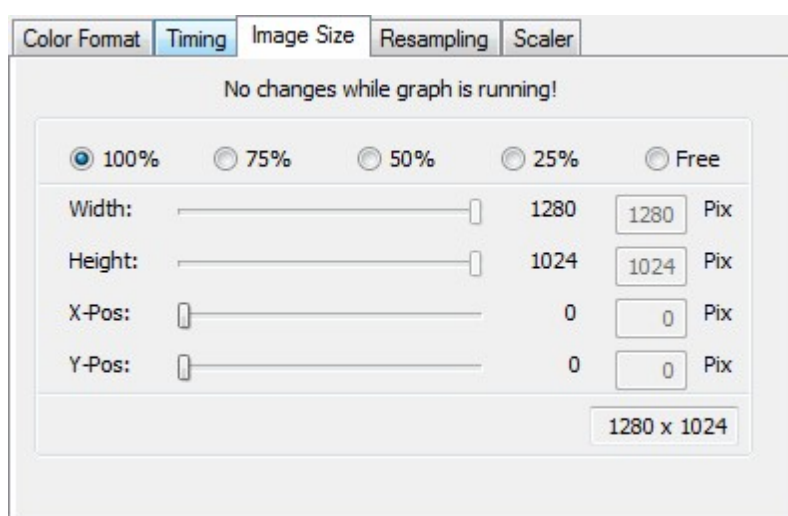


Fig. 3: Properties of the uEye Capture pin: Image Size

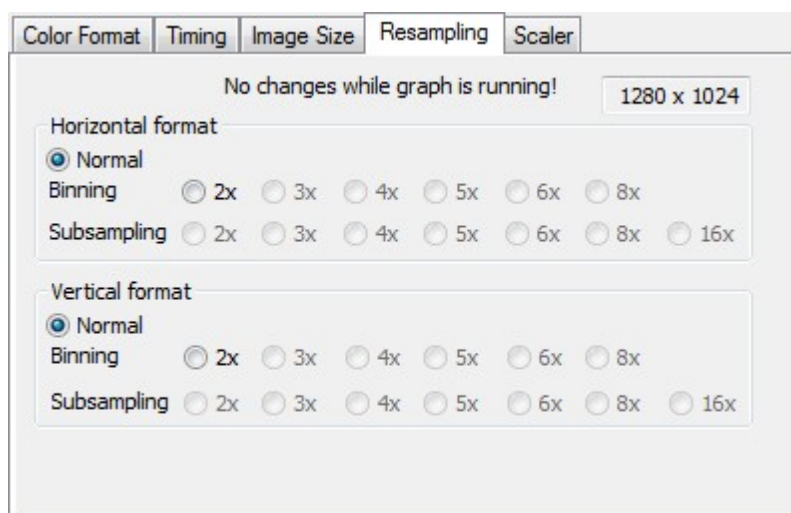


Fig. 4: Properties of the uEye Capture pin: Resampling

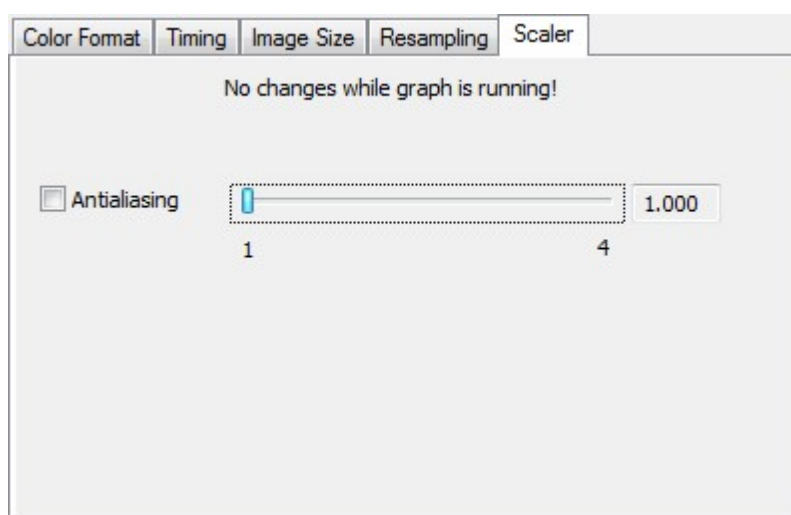


Fig. 5: Properties of the uEye Capture pin: Scaler



3.1.4 Iuc480CapturePin

IID_Iuc480CapturePin

This interface provides the uc480 camera with several special functions for timing that are not covered by the DirectShow standard. To use the interface, you must integrate the `Iuc480CaptureInterface.h` header file into your project.

`Iuc480CapturePin` provides the following functions:

GetUsedBandwidth	Returns the bus bandwidth currently generated by all opened cameras in MByte/s
GetPixelClockRange	Returns the value range for the pixel clock of a camera (minimum, maximum or default value).
GetPixelClock	Returns the current pixel clock of a camera.
SetPixelClock	Sets the pixel clock of a camera.
GetRGB8ColorMode	Returns the color mode that is used if <code>RGB8</code> was selected as the media type.
SetRGB8ColorMode	Sets the color mode that is used if <code>RGB8</code> was selected as the media type.
GetExposureRange	Returns the value range for the exposure time of a camera (minimum, maximum or step width).
GetExposureTime	Returns the current exposure time of a camera.
SetExposureTime	Sets the exposure time of a camera.

3.1.4.1 GetUsedBandwidth

Syntax

```
GetUsedBandwidth (long *plClock)
```

Description

`GetUsedBandwidth` returns the bus bandwidth currently generated by all opened cameras in MByte/s. This is an approximate value calculated from the set pixel clock and the data format (bits per pixel). The actual generated data load on the bus may be slightly different from this value.

Parameter

- `plClock`: Pointer to the variable in which the bandwidth is written.

Interface

- [Iuc480CapturePin](#)

Related functions

- [GetPixelClock](#)

3.1.4.2 GetPixelClockRange

Syntax

```
GetPixelClockRange (long *plMin, long *plMax, long *plDefault)
```

Description

`GetPixelClockRange` returns the value range for the pixel clock of a camera (minimum, maximum or default value).

Parameter

<code>p1Min</code>	Pointer to the variable in which the minimum value of the pixel clock is written in MHz.
<code>p1Max</code>	Pointer to the variable in which the maximum value of the pixel clock is written in MHz.
<code>p1Default</code>	Pointer to the variable in which the default value of the pixel clock is written in MHz.

Interface

- [Iuc480CapturePin](#)

Related functions

- [GetPixelClock](#)
- [SetPixelClock](#)

3.1.4.3 GetPixelClock

Syntax

```
GetPixelClock (long *p1Clock)
```

Description

`GetPixelClock` returns the current pixel clock of the camera.

Parameter

- `p1Clock`: Pointer to the variable in which the pixel clock is written in MHz.

Interface

- [Iuc480CapturePin](#)

Related functions

- [GetPixelClockRange](#)
- [SetPixelClock](#)
- [GetUsedBandwidth](#)

3.1.4.4 SetPixelClock

Syntax

```
SetPixelClock (long lClock)
```

Description

`SetPixelClock` sets the pixel clock of the camera.



Changes to the image geometry or pixel clock affect the value ranges for the frame rate and exposure time. After running `SetPixelClock`, we recommend resetting the frame rate (see output pin interface [IAMStreamConfig](#)) and the [exposure time](#).

Parameter

- `lClock`: Pixel clock in MHz

Interface

- [Iuc480CapturePin](#)

Related functions

- [GetPixelClockRange](#)
- [GetPixelClock](#)

3.1.4.5 GetRGB8ColorMode

Syntax

```
GetRGB8ColorMode (long *plMode)
```

Description

`GetRGB8ColorMode` returns the color mode that is used if RGB8 was selected as the media type.

Parameter

- `plMode`: Pointer to the variable in which the color mode is written.

Interface

- [Iuc480CapturePin](#)

Related functions

- [SetRGB8ColorMode](#)

3.1.4.6 SetRGB8ColorMode

Syntax

```
SetRGB8ColorMode (long lMode)
```

Description

`SetRGB8ColorMode` sets the color mode that is used if RGB8 was selected as the media type.



DirectShow does not support the Raw Bayer format. Monochrome image data is stored in the RGB8 image format (media type). To access image data in the Raw Bayer format with uc480 color cameras, you can use the `SetRGB8ColorMode` function to select whether the image data is to be saved in the Raw Bayer format or monochrome format.



With monochrome cameras, `RGB8ColorMode` must be set to Y8 (`lmode = 6`). After calling up `SetRGB8ColorMode`, the media type must be set to RGB8 with `IAMStreamConfig::SetFormat` (see [IAMStreamConfig](#)).

Parameter

<code>lMode</code>	Sets the color mode: <ul style="list-style-type: none"> • 11 = Raw Bayer (for color cameras only) • 6 = Y8/monochrome (for color and monochrome cameras)
--------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Interface

- [Iuc480CapturePin](#)

Related functions

- [GetRGB8ColorMode](#)

3.1.4.7 GetExposureRange

Syntax

```
GetExposureRange (long *plMinExp, long *plMaxExp, long *plInterval)
```

Description

GetExposureRange returns the value range for the exposure time of a camera (minimum, maximum or step width). The value range for the exposure time depends on the [pixel clock](#) settings and frame rate (see output pin interface [IAMStreamConfig](#)).

Parameter

plMinExp	Pointer to the variable in which the minimum value of the exposure time in μ s is written.
plMaxExp	Pointer to the variable in which the maximum value of the exposure time in μ s is written.
plInterval	Pointer to the variable in which the step width of the exposure time in μ s is written.

Interface

- [Iuc480CapturePin](#)

Related functions

- [GetExposureTime](#)
- [SetExposureTime](#)
- [SetPixelClock](#)

3.1.4.8 GetExposureTime

Syntax

```
GetExposureTime (long *plExp)
```

Description

GetExposureTime returns the current exposure time of a camera.

Parameter

- plExp: Pointer to the variable in which the current exposure time in μ s is written.

Interface

- [Iuc480CapturePin](#)

Related functions

- [GetExposureRange](#)
- [SetExposureTime](#)

3.1.4.9 SetExposureTime

Syntax

```
SetExposureTime (long lExp)
```

Description

`SetExposureTime` sets the exposure time of a camera.

The minimum and maximum exposure times and other dependencies of the individual sensors are listed in the "Specifications: Sensors" section in the uc480 Manual.



Newer driver versions make slightly expanded value ranges possible for the exposure time. This is why we recommend querying these ranges and setting values explicitly each time.



Changing the image size, frame rate or pixel clock also changes the exposure time. For this reason, `SetExposureTime` must be called up again afterward.

Parameter

- `lExp`: Exposure time in μs

Interface

- [Iuc480CapturePin](#)

Related functions

- [GetExposureRange](#)
- [GetExposureTime](#)

3.1.5 Iuc480Resample

IID_Iuc480Resample

This interface provides special functions for subsampling and binning that are not covered by the DirectShow standard. These can be used to reduce the image size and increase the frame rate without reducing the image area (field of view). To use the interface, you must integrate the `uc480CaptureInterface.h` header file into your project.

`Iuc480Resample` provides the following functions:

Subsampling_SetMode	Sets a subsampling mode.
Subsampling_GetMode	Returns the set subsampling mode.
Subsampling_GetVerticalResolution	Returns the set vertical subsampling factor.
Subsampling_GetHorizontalResolution	Returns the set horizontal subsampling factor.
Subsampling_Is2xVertSupported	Returns information on whether the camera supports vertical 2x subsampling.
Subsampling_Is2xHorSupported	Returns information on whether the camera supports horizontal 2x subsampling.
Subsampling_Is3xVertSupported	Returns information on whether the camera supports vertical 3x subsampling.
Subsampling_Is3xHorSupported	Returns information on whether the camera supports horizontal 3x subsampling.

<u>Subsampling_Is4xVertSupported</u>	Returns information on whether the camera supports vertical 4x subsampling.
<u>Subsampling_Is4xHorSupported</u>	Returns information on whether the camera supports horizontal 4x subsampling.
<u>Subsampling_Is5xVertSupported</u>	Returns information on whether the camera supports vertical 5x subsampling.
<u>Subsampling_Is5xHorSupported</u>	Returns information on whether the camera supports horizontal 5x subsampling.
<u>Subsampling_Is6xVertSupported</u>	Returns information on whether the camera supports vertical 6x subsampling.
<u>Subsampling_Is6xHorSupported</u>	Returns information on whether the camera supports horizontal 6x subsampling.
<u>Subsampling_Is8xVertSupported</u>	Returns information on whether the camera supports vertical 8x subsampling.
<u>Subsampling_Is8xHorSupported</u>	Returns information on whether the camera supports horizontal 8x subsampling.
<u>Subsampling_Is16xVertSupported</u>	Returns information on whether the camera supports vertical 16x subsampling.
<u>Subsampling_Is16xHorSupported</u>	Returns information on whether the camera supports horizontal 16x subsampling.
<u>Subsampling_IsColorSubsamplingSupported</u>	Returns information on whether the camera supports color-preserving subsampling.
<u>Binning_SetMode</u>	Sets a binning mode.
<u>Binning_GetMode</u>	Returns the set binning mode.
<u>Binning_GetVerticalResolution</u>	Returns the set vertical binning factor.
<u>Binning_GetHorizontalResolution</u>	Returns the set horizontal binning factor.
<u>Binning_GetImageWidth</u>	Returns the image width achieved with the current binning or subsampling settings.
<u>Binning_GetImageHeight</u>	Returns the image height achieved with the current binning or subsampling settings.
<u>Binning_Is2xVertSupported</u>	Returns information on whether the camera supports vertical 2x binning.
<u>Binning_Is2xHorSupported</u>	Returns information on whether the camera supports horizontal 2x binning.
<u>Binning_Is3xVertSupported</u>	Returns information on whether the camera supports vertical 3x binning.
<u>Binning_Is3xHorSupported</u>	Returns information on whether the camera supports horizontal 3x binning.
<u>Binning_Is4xVertSupported</u>	Returns information on whether the camera supports vertical 4x binning.
<u>Binning_Is4xHorSupported</u>	Returns information on whether the camera supports horizontal 4x binning.
<u>Binning_Is6xVertSupported</u>	Returns information on whether the camera supports vertical 6x binning.
<u>Binning_Is6xHorSupported</u>	Returns information on whether the camera supports horizontal 6x binning.

Binning_IsColorBinningSupported	Returns information on whether the camera supports color-preserving binning.
-------------------------------------------------	------------------------------------------------------------------------------

3.1.5.1 Subsampling_SetMode

Syntax

```
Subsampling_SetMode (long lMode)
```

Description

Sets a subsampling mode.

Parameter

- `lMode`: mode to be set

Interface

- [Iuc480Resample](#)

Related functions

- [Subsampling_GetMode](#)

3.1.5.2 Subsampling_GetMode

Syntax

```
Subsampling_GetMode (long* plMode)
```

Description

Returns the set subsampling mode.

Parameter

- `plMode`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Subsampling_SetMode](#)

3.1.5.3 Subsampling_GetVerticalResolution

Syntax

```
Subsampling_GetVerticalResolution (unsigned long* pulResolution)
```

Description

Returns the set vertical subsampling factor.

Parameter

- `pulResolution`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Subsampling_GetHorizontalResolution](#)

3.1.5.4 Subsampling_GetHorizontalResolution

Syntax

```
Subsampling_GetHorizontalResolution (unsigned long* pulResolution)
```

Description

Returns the set horizontal subsampling factor.

Parameter

- `pulResolution`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Subsampling_GetVerticalResolution](#)

3.1.5.5 Subsampling_Is2xVertSupported

Syntax

```
Subsampling_Is2xVertSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports vertical 2x subsampling.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Subsampling_Is2xHorSupported](#)

3.1.5.6 Subsampling_Is2xHorSupported

Syntax

```
Subsampling_Is2xHorSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports horizontal 2x subsampling.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Subsampling_Is2xVertSupported](#)

3.1.5.7 Subsampling_Is3xVertSupported

Syntax

```
Subsampling_Is3xVertSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports vertical 3x subsampling.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Subsampling_Is3xHorSupported](#)

3.1.5.8 Subsampling_Is3xHorSupported

Syntax

```
Subsampling_Is3xHorSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports horizontal 3x subsampling.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Subsampling_Is3xVertSupported](#)

3.1.5.9 Subsampling_Is4xVertSupported

Syntax

```
Subsampling_Is4xVertSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports vertical 4x subsampling.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Subsampling_Is4xHorSupported](#)

3.1.5.10 Subsampling_Is4xHorSupported

Syntax

```
Subsampling_Is4xHorSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports horizontal 4x subsampling.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Subsampling_Is4xVertSupported](#)

3.1.5.11 Subsampling_Is5xVertSupported

Syntax

```
Subsampling_Is5xVertSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports vertical 5x subsampling.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Subsampling_Is5xHorSupported](#)

3.1.5.12 Subsampling_Is5xHorSupported

Syntax

```
Subsampling_Is5xHorSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports horizontal 5x subsampling.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Subsampling_Is5xVertSupported](#)

3.1.5.13 Subsampling_Is6xVertSupported

Syntax

```
Subsampling_Is6xVertSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports vertical 6x subsampling.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Subsampling_Is6xHorSupported](#)

3.1.5.14 Subsampling_Is6xHorSupported

Syntax

```
Subsampling_Is6xHorSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports horizontal 6x subsampling.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Subsampling_Is6xVertSupported](#)

3.1.5.15 Subsampling_Is8xVertSupported

Syntax

```
Subsampling_Is8xVertSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports vertical 8x subsampling.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Subsampling_Is8xHorSupported](#)

3.1.5.16 Subsampling_Is8xHorSupported

Syntax

```
Subsampling_Is8xHorSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports horizontal 8x subsampling.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Subsampling_Is8xVertSupported](#)

3.1.5.17 Subsampling_Is16xVertSupported

Syntax

```
Subsampling_Is16xVertSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports vertical 16x subsampling.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Subsampling_Is16xHorSupported](#)

3.1.5.18 Subsampling_Is16xHorSupported

Syntax

```
Subsampling_Is16xHorSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports horizontal 16x subsampling.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Subsampling_Is16xVertSupported](#)

3.1.5.19 Subsampling_IsColorSubsamplingSupported

Syntax

```
Subsampling_IsColorSubsamplingSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports color-preserving subsampling.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Subsampling_SetMode](#)
- [Subsampling_GetMode](#)

3.1.5.20 Binning_SetMode

Syntax

```
Binning_SetMode (long lMode)
```

Description

Sets a binning mode.

Parameter

- `lMode`: mode to be set

Interface

- [Iuc480Resample](#)

Related functions

- [Binning_GetMode](#)

3.1.5.21 Binning_GetMode

Syntax

```
Binning_GetMode (long* plMode)
```

Description

Returns the set binning mode.

Parameter

- `plMode`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Binning_SetMode](#)

3.1.5.22 Binning_GetVerticalResolution

Syntax

```
Binning_GetVerticalResolution (unsigned long* pulResolution)
```

Description

Returns the set vertical binning factor.

Parameter

- `pulResolution`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Binning_GetHorizontalResolution](#)

3.1.5.23 Binning_GetHorizontalResolution

Syntax

```
Binning_GetHorizontalResolution (unsigned long* pulResolution)
```

Description

Returns the set horizontal binning factor.

Parameter

- `pulResolution`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Binning_GetVerticalResolution](#)

3.1.5.24 Binning_GetImageWidth

Syntax

```
Binning_GetImageWidth (int* pnWidth)
```

Description

Returns the image width achieved with the current binning settings.

Parameter

- `pnWidth`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Binning_GetImageHeight](#)

3.1.5.25 Binning_GetImageHeight

Syntax

```
Binning_GetImageHeight (int* pnHeight)
```

Description

Returns the image height achieved with the current binning settings.

Parameter

- `pnHeight`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Binning_GetImageWidth](#)

3.1.5.26 Binning_Is2xVertSupported

Syntax

```
Binning_Is2xVertSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports vertical 2x binning.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Binning_Is2xHorSupported](#)

3.1.5.27 Binning_Is2xHorSupported

Syntax

```
Binning_Is2xHorSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports horizontal 2x binning.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Binning_Is2xVertSupported](#)

3.1.5.28 Binning_Is3xVertSupported

Syntax

```
Binning_Is3xVertSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports vertical 3x binning.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Binning_Is3xHorSupported](#)

3.1.5.29 Binning_Is3xHorSupported

Syntax

```
Binning_Is3xHorSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports horizontal 3x binning.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Binning_Is3xVertSupported](#)

3.1.5.30 Binning_Is4xVertSupported

Syntax

```
Binning_Is4xVertSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports vertical 4x binning.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Binning_Is4xHorSupported](#)

3.1.5.31 Binning_Is4xHorSupported

Syntax

```
Binning_Is4xHorSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports horizontal 4x binning.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Binning_Is4xVertSupported](#)

3.1.5.32 Binning_Is6xVertSupported

Syntax

```
Binning_Is6xVertSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports vertical 6x binning.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Binning_Is6xHorSupported](#)

3.1.5.33 Binning_Is6xHorSupported

Syntax

```
Binning_Is6xHorSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports horizontal 6x binning.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Binning_Is6xVertSupported](#)

3.1.5.34 Binning_IsColorBinningSupported

Syntax

```
Binning_IsColorBinningSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports color-preserving binning.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Resample](#)

Related functions

- [Binning_GetMode](#)
- [Binning_SetMode](#)

3.1.6 Iuc480Scaler

IID_Iuc480Scaler

This interface provides uc480 special functions for the scaler function for some uc480 models that are not covered by the DirectShow standard. To use the interface, you must integrate the `uc480CaptureInterface.h` header file into your project.



Internal image scaling is only supported by DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N cameras.

`Iuc480Scaler` provides the following functions:

GetSensorScalerInfo	Returns information for the internal scaling for some uc480 models
SetSensorScaler	Enables/Disables in some uc480 models the internal scaling
GetScalerImageWidth	Returns the image width achieved with the current scaling settings.
GetScalerImageHeight	Returns the image height achieved with the current scaling settings.
SetImageSize	Sets the image width and height for the current scaling settings.

3.1.6.1 GetSensorScalerInfo

Syntax

```
GetSensorScalerInfo(SENSORSCALERINFO *pSensorScalerInfo, INT nSensorScalerInfoSize)
```

Description

Returns for some uc480 models information for the internal scaling.



Internal image scaling is only supported by DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N cameras.

Parameter

pSensorScalerInfo	Pointer to a <code>SENSORSCALERINFO</code> type structure to which the information will be written
nSensorScalerInfoSize	Size of the structure

Contents of the `SENSORSCALERINFO` structure

INT	nCurrMode	Returns the current mode
INT	nNumberOfSteps	Returns the number of steps for the scaling factor
double	dblFactorIncrement	Returns the increment for the scaling factor
double	dblMinFactor	Returns the minimum scaling factor
double	dblMaxFactor	Returns the maximum scaling factor
double	dblCurrFactor	Returns the current scaling factor
INT	nSupportedModes	Returns the supported function modes, see SetSensorScaler
BYTE	bReserved[84]	Reserved

Interface

- [Iuc480Scaler](#)

Related functions

- [SetSensorScaler](#)

3.1.6.2 SetSensorScaler

Syntax

```
SetSensorScaler(UINT nMode, double dblFactor)
```

Description

Enables internal image scaling for some sensors. This allows to reduce the image resolution by adjustable factors. Thus, the amount of data from high resolution sensors can be reduced.



Internal image scaling is only supported by DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N cameras.

Parameter

nMode	
IS_ENABLE_SENSOR_SCALER	Enable image scaling
IS_ENABLE_SENSOR_SCALER IS_ENABLE_ANTI_ALIASING	Enable image scaling with smoothed edges (anti aliasing effect)
dblFactor	Scaling factor

Interface

- [Iuc480Scaler](#)

Related functions

- [GetSensorScalerInfo](#)

3.1.6.3 GetScalerImageWidth

Syntax

```
GetScalerImageWidth (int *pnWidth)
```

Description

Returns the image width achieved with the current scaling settings.



Internal image scaling is only supported by DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N cameras.

Parameter

- pnWidth: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Scaler](#)

Related functions

- [GetScalerImageHeight](#)
- [SetImageSize](#)

3.1.6.4 GetScalerImageHeight

Syntax

```
GetScalerImageHeight (int *pnHeight)
```

Description

Returns the image height achieved with the current scaling settings.



Internal image scaling is only supported by DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N cameras.

Parameter

- pnHeight: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Scaler](#)

Related functions

- [GetScalerImageWidth](#)
- [SetImageSize](#)

3.1.6.5 SetImageSize

Syntax

```
SetImageSize (int nWidth, int nHeight)
```

Description

Sets the image width and height for the current scaling settings.



Internal image scaling is only supported by DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N cameras.

Parameter

nWidth	Sets the image width
nHeight	Sets the image height

Interface

- [Iuc480Scaler](#)

Related functions

- [GetScalerImageWidth](#)
- [GetScalerImageHeight](#)

3.2 Filter interfaces

IAMDroppedFrames	This interface can be used to call up information on captured and dropped images
IAMFilterMiscFlags	This interface can be used to query various pieces of information
IAMVideoControl	This interface can be used to set vertical and horizontal image mirroring and the trigger mode
IAMVideoProcAmp	This interface can be used to set the image parameters of a video stream
IKsPropertySet	This interface can be used to query the pin category
ISpecifyPropertyPages	This interface can be used to access the property pages on which camera parameters can be graphically set
Iuc480AntiFlicker	Special functions for controlling the anti-flicker function on some uc480 models
Iuc480AOI	Special functions for setting an area of interest (AOI)
Iuc480AutoBacklight	Special functions for correcting the backlight on some uc480 models
Iuc480AutoContrast	Special functions for correcting the auto contrast on some uc480

	models
Iuc480AutoFeatures	Special functions for automatic image control on the uc480 camera
Iuc480AutoFramerate	Special functions for controlling the automatic frame rate control on some uc480 models
Iuc480AutoParameter	This interface provides special functions for controlling the automatic functions
Iuc480CameraLUT	Special functions for controlling the camera LUT
Iuc480Capture	Special functions for hotpixel correction on the uc480 camera, as well as other factors
Iuc480CaptureEx	Special functions for gamma and gain on the uc480 camera
Iuc480ColorConverter	Special functions for color conversion
Iuc480ColorTemperature	Special functions for setting the color temperature
Iuc480DeviceFeature	Special functions for controlling camera-specific functions
Iuc480DigitalZoom	Special functions for the digital zoom on some uc480 models
Iuc480EdgeEnhancement	Special functions for using the software edge filter
Iuc480Event	Special functions for the event handling
Iuc480FaceDetection	Special functions for face detection on some uc480 models
Iuc480Flash	Special functions for controlling the camera inputs/outputs
Iuc480Focus	Special functions for controlling the auto focus of some uc480 models
Iuc480Gain	Special functions for setting analog sensor gain
Iuc480HotPixel	Special functions for controlling camera-specific functions
Iuc480ImageFormat	Special functions for image formats
Iuc480ImageStabilization	Special functions for image stabilization on some uc480 models
Iuc480IO	Special functions for controlling the GPIOs on some uc480 models
Iuc480Photometry	Special functions for controlling the automatic image control on some uc480 models.
Iuc480Saturation	Special functions for gradually increasing or decreasing the color saturation on some uc480 models
Iuc480ScenePreset	Special functions for controlling the scene modes on some uc480 models
Iuc480SensorAWB	Special functions for the sensor-oriented automatic white balance on some uc480 models
Iuc480Sharpness	Special functions for gradually increasing or decreasing the image sharpness on some uc480 models
Iuc480Trigger	Special functions for triggering the camera
Iuc480TriggerDebounce	Special functions for suppressing (debouncing) faults at the digital input when using GigE uc480 cameras in trigger mode

3.2.1 IAMDroppedFrames

IID_IAMDroppedFrames

This interface can be used to call up information on captured and dropped images. The uc480 DirectShow interface supports the following parameters of `IAMDroppedFrames` :

GetAverageFrameSize	Size of the allocated image memory in bytes
GetNumDropped	Number of images dropped by the DirectShow engine
GetNumNotDropped	Number of images captured by the DirectShow engine



Images may be dropped by the DirectShow engine for the following reasons:

The camera pixel clock is set too high for the system. When operating several USB cameras on a single connection, the total pixel clock of all the cameras should not exceed approximately 40 MHz.

3.2.2 IAMFilterMiscFlags

IID_IAMFilterMiscFlags

This interface can be used to query various pieces of information. The uc480 DirectShow interface supports the following parameter of IAMFilterMiscFlags:

GetMiscFlags	Returns the <code>AM_FILTER_MISC_FLAGS_IS_SOURCE</code> flag if the current filter represents a live source.
--------------	--------------------------------------------------------------------------------------------------------------

3.2.3 IAMVideoControl

IID_IAMVideoControl

This interface can be used to set vertical and horizontal image mirroring and the trigger mode. For the trigger mode, you can select a hardware trigger (external) or software trigger (see [Iuc480Trigger](#)). The uc480 DirectShow interface supports the following parameters of IAMVideoControl:

VideoControlFlag_FlipHorizontal	Mirrors the image from left to right.
VideoControlFlag_FlipVertical	Mirrors the image from top to bottom.
VideoControlFlag_ExternalTriggerEnable	Enable the trigger mode
VideoControlFlag_Trigger	Select a hardware/software trigger

Example

```
HRESULT status = S_OK;
LONG lMode = 0;

IAMVideoControl* pIAMVideoControl = NULL;

status = m_pActiveVideoSource->QueryInterface(IID_IAMVideoControl,
                                              (void**) &pIAMVideoControl);

/* get flip mode */
status = pIAMVideoControl->GetMode(NULL, &lMode);

/* activate flip mode vertical */
status = pIAMVideoControl->SetMode(NULL, lMode | VideoControlFlag_FlipVertical);

/* deactivate flip mode vertical */
status = pIAMVideoControl->SetMode(NULL, lMode & ~VideoControlFlag_FlipVertical);

/* activate flip mode horizontal */
status = pIAMVideoControl->SetMode(NULL, lMode | VideoControlFlag_FlipHorizontal);

/* deactivate flip mode horizontal */
status = pIAMVideoControl->SetMode(NULL, lMode & ~VideoControlFlag_FlipHorizontal);
```

```
/* activate flip mode vertical and deactivate flip mode horizontal */
status = pIAMVideoControl->SetMode(NULL, VideoControlFlag_FlipVertical);

/* activate flip mode horizontal and deactivate flip mode vertical */
status = pIAMVideoControl->SetMode(NULL, VideoControlFlag_FlipHorizontal);
```

3.2.4 IAMVideoProcAmp

IID_IAMVideoProcAmp

This interface can be used to set the image parameters of a video stream. The uc480 DirectShow interface supports the following parameters of IAMVideoProcAmp:

VideoProcAmp_Sharpness	Set edge enhancement [0,2] ^{*1}
VideoProcAmp_Gamma	Set the gamma value of the image [0,255] ^{*1}
VideoProcAmp_ColorEnable	Enable color correction [0,1] ^{*1}
VideoProcAmp_WhiteBalance	Set white balance of the camera: Select automatic [0,1] or color temperature [2200,10000]K ^{*1} The actual value range can be prompted via GetRange (VideoProcAmp_WhiteBalance...).
VideoProcAmp_BacklightCompensation	Enable black level correction of the sensor: Select automatic [0,1] or offset [0,255]
VideoProcAmp_Gain	Enable analog sensor gain: Select automatic [0,1] or gain value [0,100]
VideoProcAmp_Brightness	Enable black level correction of the sensor: Select automatic [0,1] or offset [0,255]
VideoProcAmp_Contrast	Enable analog sensor gain: Select automatic [0,1] or gain value [0,100]

^{*1} Software correction in the uc480 API

With the flags VideoProcAmp_Flags_Manual and VideoProcAmp_Flags_Auto you specify if a value is set automatically or manually. Thus, e.g. for black level correction you set the offset [0,255] in combination with VideoProcAmp_Flags_Manual. In combination with VideoProcAmp_Flags_Auto you disable/enable the automatic correction [0,1].

Example for manually setting the gain

```
// ===== btn: Set Gain vai IAMVideoProcAmp =====
void uc480_DirectShow_Demo_Dlg::OnBnClickedIamvideoprocampgain()
{
    HRESULT status = S_OK;

    IAMVideoProcAmp* pIAMVideoProcAmp = NULL;
    status = m_pActiveVideoSource->QueryInterface( IID_IAMVideoProcAmp, (void**) &pIAMVideoProcAmp);

    status = pIAMVideoProcAmp->Set(VideoProcAmp_Gain, 80, VideoProcAmp_Flags_Manual);
}
```

3.2.5 IKsPropertySet

IID_IKsPropertySet

This interface can be used to query the pin category. The uc480 DirectShow interface supports the following parameter of IKsPropertySet:

Get	Returns the pin category (with the uc480 DirectShow interface, it is always PIN_CATEGORY_CAPTURE)
-----	---------------------------------------------------------------------------------------------------

3.2.6 ISpecifyPropertyPages

IID_ISpecifyPropertyPages

This interface can be used to access property pages on which camera parameters can be graphically set. The property pages contain all the settings that are contained in the "filter interfaces".

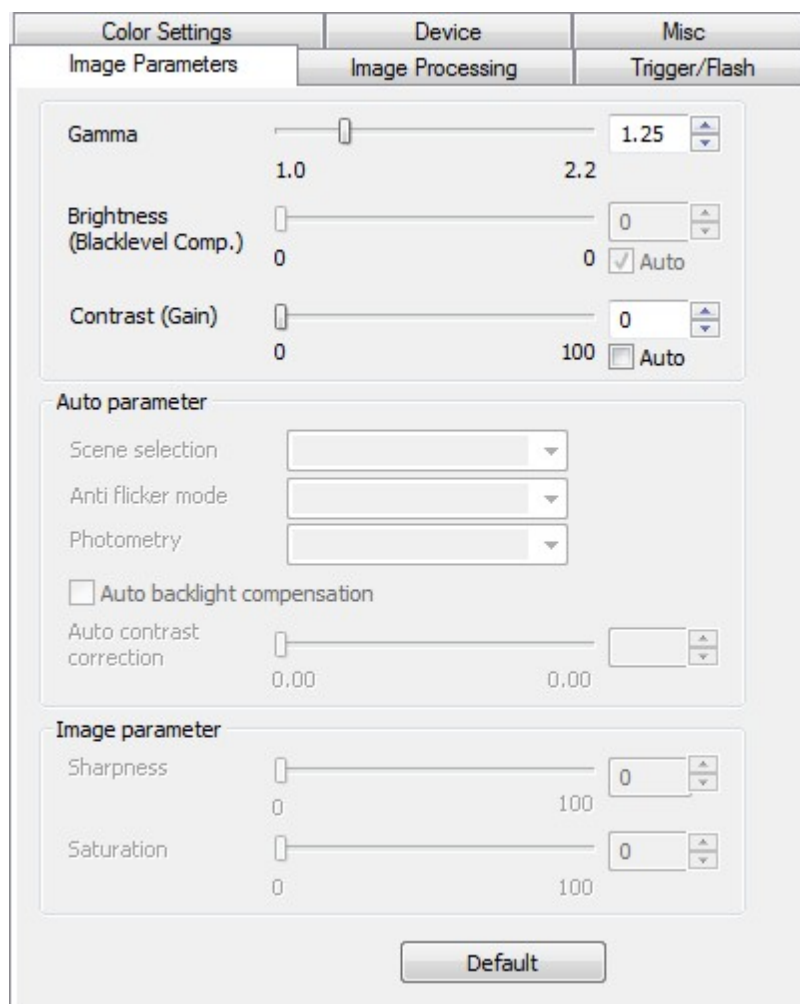


Fig. 8: Properties of the uEye Capture filter: Image Parameters

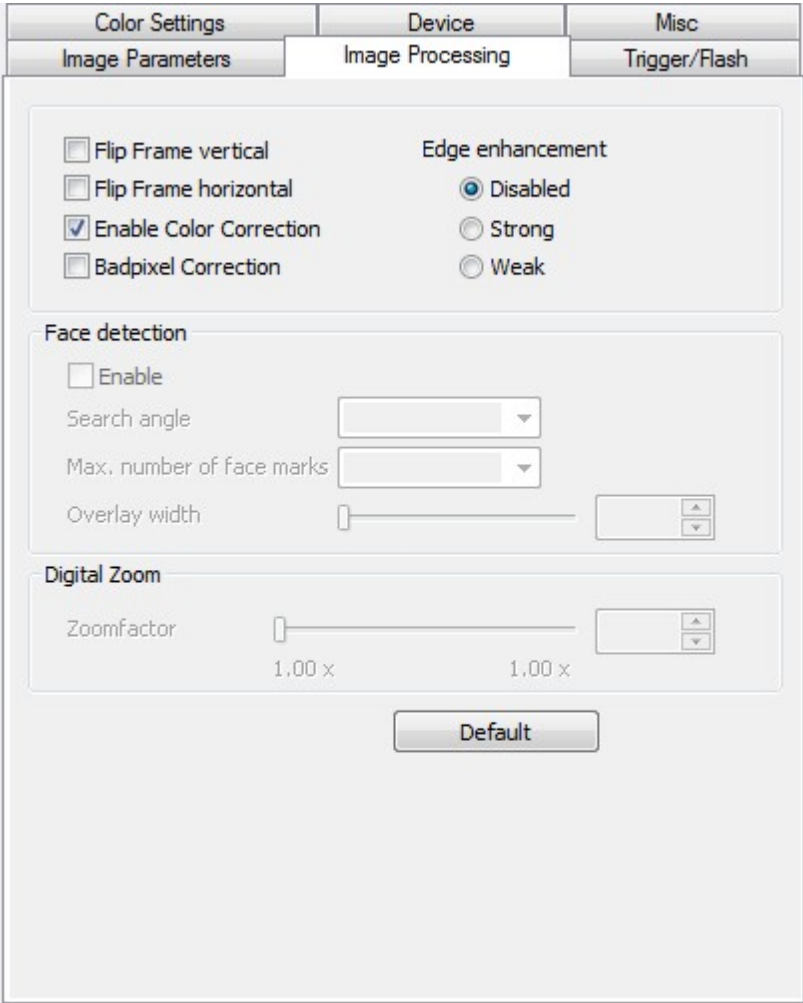


Fig. 9: Properties of the uEye Capture filter: Image Processing

Color Settings	Device	Misc
Image Parameters	Image Processing	Trigger/Flash

Trigger Mode

☐ Trigger off

☒ Trigger falling edge Force Trigger

☐ Trigger rising edge

☐ Software Trigger

Trigger Debouncing

Mode off

Delay 0 μ s 0 μ s

Flash Output

Strobe low active Global Exposure Window

Delay 40 μ s 66675 μ s

Duration (0 = Exp.) 0 μ s 66675 μ s

Use GPIO as Flash Output ☐ GPIO 1 ☐ GPIO 2

Default

Fig. 10: Properties of the uEye Capture filter: Trigger/Flash

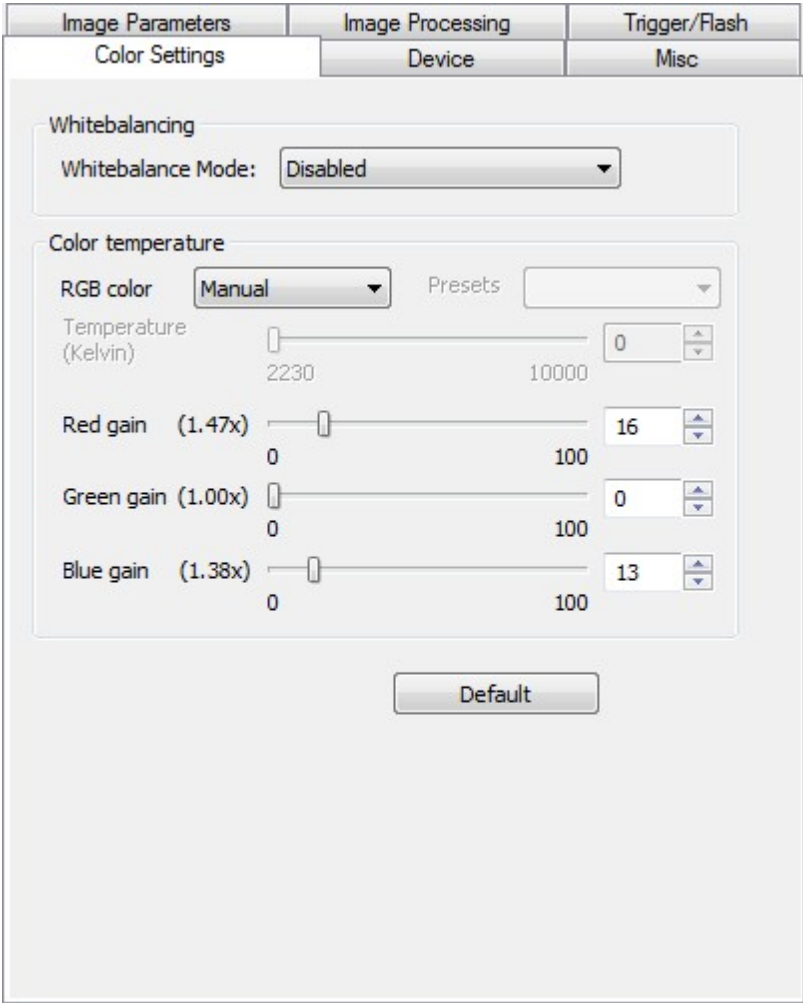


Fig. 11: Properties of the uEye Capture filter: Image Parameters

Image Parameters	Image Processing	Trigger/Flash
Color Settings	Device	Misc
Camera		
Camera model: UI124xSE-C		
Serial No.: 4002776916		
Native resolution: 1280 x 1024		
Frames received: 40857 Frames dropped: 0		
Load/Save location: display file selection box ▼		
<div>Save Settings Load Settings Reset to defaults</div>		
Interface		
Changes may take no effect without restarting your application!		
<input type="checkbox"/> Use bottom-up images		
<input type="checkbox"/> Use automatic registering		
<input checked="" type="checkbox"/> Use camera names		

Fig. 12: Properties of the uEye Capture filter: Device

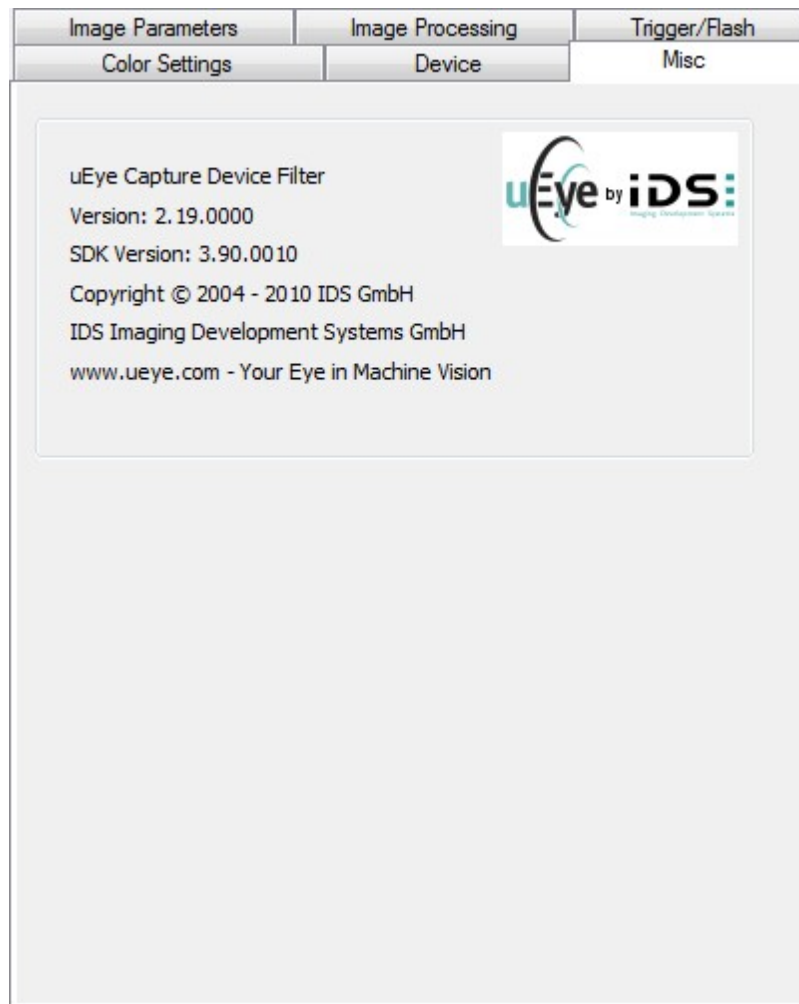


Fig. 13: Properties of the uEye Capture filter: Misc

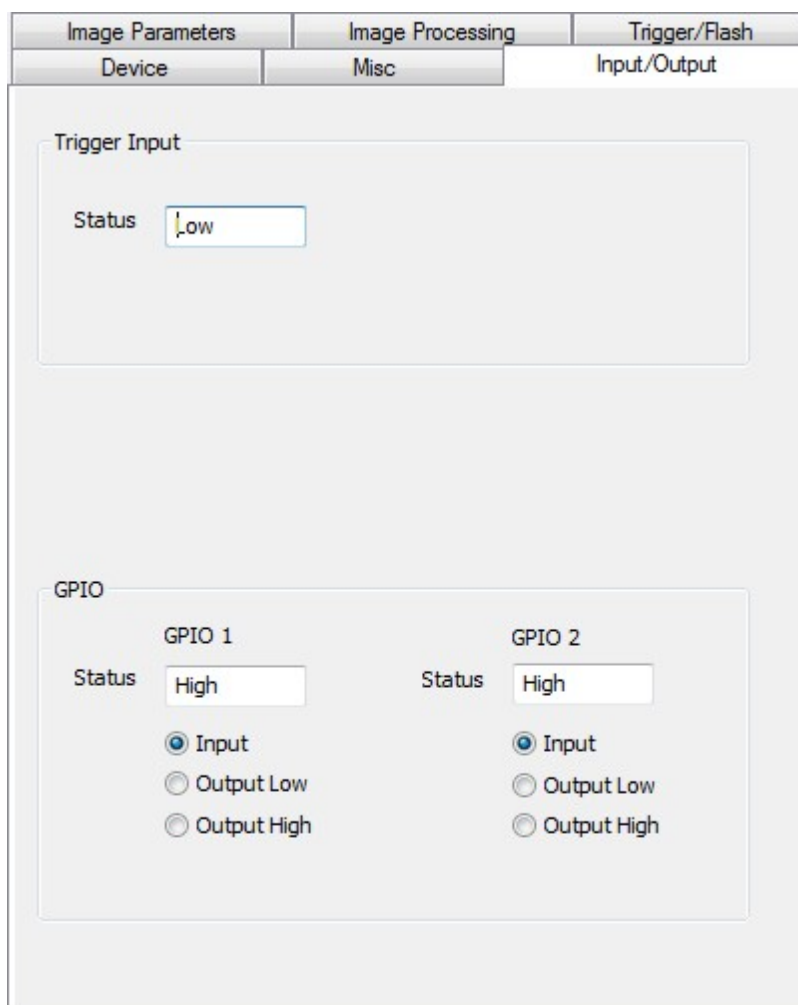


Fig. 14: Properties of the uEye Capture filter: Input/Output

3.2.7 Iuc480AOI

IID_Iuc480AOI

This interface provides special features for setting the image area (Area of Interest, AOI) of an image available with some uc480 models that are not covered by the DirectShow standard. Possible AOIs include:

- Image AOI – Display partial image
- Auto-brightness AOI – Reference image area for automatic brightness control
- Auto-white balance AOI – Reference image area for automatic white balance

To use the interface, you must integrate the `uc480CaptureInterface.h` header file into your project.

`IID_Iuc480AOI` provides the following features:

AOI_IsImageAOISupported	Returns whether the camera supports image AOIs.
AOI_GetImageAOI	Returns the coordinates of the current image AOI.
AOI_SetImageAOI	Sets the coordinates of the image AOI.
AOI_GetAutoBrightnessAOI	Returns the coordinates of the AOI for automatic brightness control.
AOI_SetAutoBrightnessAOI	Sets the coordinates of the AOI for automatic brightness control.
AOI_GetAutoWBAOI	Returns the coordinates of the AOI for automatic white balance.
AOI_SetAutoWBAOI	Sets the coordinates of the AOI for automatic white balance.
AOI_GetIncPosX	Returns the increment for the position on the x axis.
AOI_GetIncPosY	Returns the increment for the position on the y axis.
AOI_GetIncSizeX	Returns the increment for the size on the x axis.
AOI_GetIncSizeY	Returns the increment for the size on the y axis.
AOI_GetMinMaxPosX	Returns the minimum and maximum possible AOI positions on the x axis.
AOI_GetMinMaxPosY	Returns the minimum and maximum possible AOI positions on the y axis.
AOI_GetMinMaxSizeX	Returns the minimum and maximum possible AOI sizes on the x axis.
AOI_GetMinMaxSizeY	Returns the minimum and maximum possible AOI sizes on the y axis.

3.2.7.1 AOI_IsImageAOISupported

Syntax

```
AOI_IsImageAOISupported (bool* pbSupported)
```

Description

Returns whether the camera supports user-selected image AOIs.

Parameter

- `pbSupported`: Pointer to the variable to which the return value is written.

Interface

- [Iuc480AOI](#)

Related functions

- [AOI_GetImageAOI](#)
- [AOI_SetImageAOI](#)

3.2.7.2 AOI_GetImageAOI

Syntax

```
AOI_GetImageAOI(IS_RECT *pRectAOI)
AOI_GetImageAOI (RECT *prcAOI)
```

Description

Returns the coordinates of the current image AOI.

Parameter

pRectAOI	Pointer to an object of the type <code>IS_RECT</code> in which the top left corner, width and height of the AOI are returned [<code>s32X</code> , <code>s32Y</code> , <code>s32Width</code> , <code>s32Height</code>].
prcAOI	Pointer to an object of the type <code>RECT</code> in which the coordinates are returned in the sequence [left, top, right, bottom].

Interface

- [Iuc480AOI](#)

Related functions

- [AOI_SetImageAOI](#)
- [AOI_IsImageAOISupported](#)

3.2.7.3 AOI_SetImageAOI

Syntax

```
AOI_SetImageAOI(IS_RECT rectAOI)
AOI_SetImageAOI (RECT rcAOI)
```

Description

Sets the coordinates of the image AOI.

Parameter

rectAOI	Sets the position and size of the AOI using an object of the type <code>IS_RECT</code> . The top left corner, width and height of the AOI must be transferred [<code>s32X</code> , <code>s32Y</code> , <code>s32Width</code> , <code>s32Height</code>].
rcAOI	Sets the position and size of the image using an object of the type <code>RECT</code> . Coordinates must be transferred in the sequence [left, top, right, bottom].

Interface

- [Iuc480AOI](#)

Related functions

- [AOI_GetImageAOI](#)

- [AOI_IsImageAOISupported](#)

3.2.7.4 AOI_GetAutoBrightnessAOI

Syntax

```
AOI_GetAutoBrightnessAOI(IS_RECT *pRectAOI)
AOI_GetAutoBrightnessAOI (RECT *prcAOI)
```

Description

Returns the coordinates of the AOI for automatic brightness control.

Parameter

pRectAOI	Pointer to an object of the type <code>IS_RECT</code> in which the top left corner, width and height of the AOI are returned [<code>s32X</code> , <code>s32Y</code> , <code>s32Width</code> , <code>s32Height</code>].
prcAOI	Pointer to an object of the type <code>RECT</code> in which the coordinates are returned in the sequence [left, top, right, bottom].

Interface

- [Iuc480AOI](#)

Related functions

- [AOI_SetAutoBrightnessAOI](#)

3.2.7.5 AOI_SetAutoBrightnessAOI

Syntax

```
AOI_SetAutoBrightnessAOI(IS_RECT rectAOI)
AOI_SetAutoBrightnessAOI (RECT rcAOI)
```

Description

Sets the coordinates of the AOI for automatic brightness control.

Parameter

rectAOI	Sets the position and size of the AOI using an object of the type <code>IS_RECT</code> . The top left corner, width and height of the AOI must be transferred [<code>s32X</code> , <code>s32Y</code> , <code>s32Width</code> , <code>s32Height</code>].
rcAOI	Sets the position and size of the image using an object of the type <code>RECT</code> . Coordinates must be transferred in the sequence [left, top, right, bottom].

Interface

- [Iuc480AOI](#)

Related functions

- [AOI_GetAutoBrightnessAOI](#)

3.2.7.6 AOI_GetAutoWBAOI

Syntax

```
AOI_GetAutoWBAOI(IS_RECT *pRectAOI)
AOI_GetAutoWBAOI (RECT *prcAOI)
```

Description

Returns the coordinates of the AOI for automatic white balance.

Parameter

pRectAOI	Pointer to an object of the type <code>IS_RECT</code> in which the top left corner, width and height of the AOI are returned [<code>s32X</code> , <code>s32Y</code> , <code>s32Width</code> , <code>s32Height</code>].
prcAOI	Pointer to an object of the type <code>RECT</code> in which the coordinates are returned in the sequence [left, top, right, bottom].

Interface

- [Iuc480AOI](#)

Related functions

- [AOI_SetAutoWBAOI](#)

3.2.7.7 AOI_SetAutoWBAOI

Syntax

```
AOI_SetAutoWBAOI(IS_RECT rectAOI)
AOI_SetAutoWBAOI (RECT rcAOI)
```

Description

Sets the coordinates of the AOI for automatic white balance.

Parameter

rectAOI	Sets the position and size of the AOI using an object of the type <code>IS_RECT</code> . The top left corner, width and height of the AOI must be transferred [<code>s32X</code> , <code>s32Y</code> , <code>s32Width</code> , <code>s32Height</code>].
rcAOI	Sets the position and size of the image using an object of the type <code>RECT</code> . Coordinates must be transferred in the sequence [left, top, right, bottom].

Interface

- [Iuc480AOI](#)

Related functions

- [AOI_GetAutoWBAOI](#)

3.2.7.8 AOI_GetIncPosX

Syntax

```
AOI_GetIncPosX (INT* pnInc)
```

Description

Returns the increment for the position on the x axis.

Parameter

- `pnInc`: Pointer to the variable in which the increment is returned.

Interface

- [Iuc480AOI](#)

Related functions

- [AOI_GetIncPosY](#)
- [AOI_GetMinMaxPosX](#)

3.2.7.9 AOI_GetIncPosY

Syntax

```
AOI_GetIncPosY (INT* pnInc)
```

Description

Returns the increment for the position on the y axis.

Parameter

- `pnInc`: Pointer to the variable in which the increment is returned.

Interface

- [Iuc480AOI](#)

Related functions

- [AOI_GetIncPosX](#)
- [AOI_GetMinMaxPosY](#)

3.2.7.10 AOI_GetIncSizeX

Syntax

```
AOI_GetIncSizeX (INT* pnInc)
```

Description

Returns the increment for the size on the x axis.

Parameter

- `pnInc`: Pointer to the variable in which the increment is returned.

Interface

- [Iuc480AOI](#)

Related functions

- [AOI_GetIncSizeY](#)
- [AOI_GetMinMaxSizeX](#)

3.2.7.11 AOI_GetIncSizeY**Syntax**

```
AOI_GetIncSizeY (INT* pnInc)
```

Description

Returns the increment for the size on the y axis.

Parameter

- `pnInc`: Pointer to the variable in which the increment is returned.
- **Interface**
- [Iuc480AOI](#)

Related functions

- [AOI_GetIncSizeX](#)
- [AOI_GetMinMaxSizeY](#)

3.2.7.12 AOI_GetMinMaxPosX**Syntax**

```
AOI_GetMinMaxPosX (INT* pnMin, INT* pnMax)
```

Description

Returns the minimum and maximum possible AOI positions on the x axis.

Parameter

<code>pnMin</code>	Pointer to the variable in which the minimum position is returned.
<code>pnMax</code>	Pointer to the variable in which the maximum position is returned.

Interface

- [Iuc480AOI](#)

Related functions

- [AOI_GetMinMaxPosY](#)
- [AOI_GetIncPosX](#)

3.2.7.13 AOI_GetMinMaxPosY**Syntax**

```
AOI_GetMinMaxPosY (INT* pnMin, INT* pnMax)
```

Description

Returns the minimum and maximum possible AOI positions on the y axis.

Parameter

pnMin	Pointer to the variable in which the minimum position is returned.
pnMax	Pointer to the variable in which the maximum position is returned.

Interface

- [Iuc480AOI](#)

Related functions

- [AOI_GetMinMaxPosX](#)
- [AOI_GetIncPosY](#)

3.2.7.14 AOI_GetMinMaxSizeX**Syntax**

```
AOI_GetMinMaxSizeX (INT* pnMin, INT* pnMax)
```

Description

Returns the minimum and maximum possible AOI sizes on the x axis.

Parameter

pnMin	Pointer to the variable in which the minimum size is returned.
pnMax	Pointer to the variable in which the maximum size is returned.

Interface

- [Iuc480AOI](#)

Related functions

- [AOI_GetMinMaxSizeY](#)
- [AOI_GetIncSizeX](#)

3.2.7.15 AOI_GetMinMaxSizeY**Syntax**

```
AOI_GetMinMaxSizeY (INT* pnMin, INT* pnMax)
```

Description

Returns the minimum and maximum possible AOI sizes on the y axis.

Parameter

pnMin	Pointer to the variable in which the minimum size is returned.
pnMax	Pointer to the variable in which the maximum size is returned.

Interface

- [Iuc480AOI](#)

Related functions

- [AOI_GetMinMaxSizeX](#)

- [AOI_GetIncSizeY](#)

3.2.8 Iuc480AutoFeatures

IID_Iuc480AutoFeatures

This interface provides special functions for automatic image adjustment on the uc480 camera that are not covered by the DirectShow standard. To use the interface, you must integrate the `uc480CaptureInterface.h` header file into your project.

For more information on automatic adjustment, see also the "Automatic image control" chapter in the uc480 Manual.



- Adjustment is only active while the camera is capturing images.
- The automatic functions are disabled when the settings for the exposure time and gain are manually changed.
- The pixel clock cannot be changed if the **automatic exposure time** is enabled.
- **Automatic gain** can only be used on cameras with a total gain controller. Automatic white balance can only be used on cameras with RGB gain-controller hardware.

`Iuc480AutoFeatures` provides the following functions:

<u>SetAutoBrightnessReference</u>	Sets the automatic gain/automatic exposure time reference value.
<u>GetAutoBrightnessReference</u>	Returns the automatic gain/automatic exposure time reference value.
<u>SetAutoBrightnessMaxExposure</u>	Sets the upper adjustment limit for the automatic exposure time.
<u>GetAutoBrightnessMaxExposure</u>	Returns the upper adjustment limit for automatic exposure time.
<u>SetAutoBrightnessMaxGain</u>	Sets the upper adjustment limit for automatic gain.
<u>GetAutoBrightnessMaxGain</u>	Returns the upper adjustment limit for automatic gain.
<u>SetAutoBrightnessSpeed</u>	Sets the speed value for the automatic gain/automatic exposure time.
<u>GetAutoBrightnessSpeed</u>	Returns the speed value for the automatic gain/automatic exposure time.
<u>SetAutoBrightnessAOI</u>	Sets the reference AOI for the automatic gain/automatic exposure time.
<u>GetAutoBrightnessAOI</u>	Returns the reference AOI for the automatic gain/automatic exposure time.
<u>SetAutoWBGainOffsets</u>	Sets the offset for the red and blue channel of the automatic white balance.
<u>GetAutoWBGainOffsets</u>	Returns the offset for the red and blue channel of the automatic white balance.
<u>SetAutoWBGainRange</u>	Sets the color gain adjustment limits for automatic white balance.
<u>GetAutoWBGainRange</u>	Returns the color gain adjustment limits for automatic white balance.
<u>SetAutoWBSpeed</u>	Sets the speed for automatic white balance.
<u>GetAutoWBSpeed</u>	Returns the speed for automatic white balance.
<u>SetAutoWBAOI</u>	Sets the AOI for automatic white balance.
<u>GetAutoWBAOI</u>	Returns the AOI for automatic white balance.

3.2.8.1 SetAutoBrightnessReference

Syntax

`SetAutoBrightnessReference (long lReference)`

Description

`SetAutoBrightnessReference` sets the reference value (setpoint) for the automatic gain/automatic exposure time.

Parameter

- `lReference`: Reference value for automatic brightness

Interface

- [Iuc480AutoFeatures](#)

Related functions

- [GetAutoBrightnessReference](#)
- [SetAutoBrightnessMaxExposure](#)
- [SetAutoBrightnessMaxGain](#)

3.2.8.2 GetAutoBrightnessReference

Syntax

```
GetAutoBrightnessReference(long* pReference)
```

Description

`GetAutoBrightnessReference` returns the reference value (setpoint) for the automatic gain/automatic exposure time.

Parameter

- `pReference`: Pointer to the variable in which the reference value of automatic brightness is written.

Interface

- [Iuc480AutoFeatures](#)

Related functions

- [SetAutoBrightnessReference](#)

3.2.8.3 SetAutoBrightnessMaxExposure

Syntax

```
SetAutoBrightnessMaxExposure (long lMaxExposure)
```

Description

`SetAutoBrightnessMaxExposure` sets the maximum value for the automatic control of the exposure time.

Parameter

- `lMaxExposure`: Maximum value of automatic exposure time

Interface

- [Iuc480AutoFeatures](#)

Related functions

- [GetAutoBrightnessMaxExposure](#)
- [SetAutoBrightnessMaxGain](#)
- [SetAutoBrightnessReference](#)

3.2.8.4 GetAutoBrightnessMaxExposure

Syntax

```
GetAutoBrightnessMaxExposure (long* plMaxExposure)
```

Description

GetAutoBrightnessMaxExposure returns the maximum value for the automatic adjustment of the exposure time.

Parameter

- **plMaxExposure**: Pointer to the variable in which the maximum value of the automatic exposure time is written.

Interface

- [Iuc480AutoFeatures](#)

Related functions

- [SetAutoBrightnessMaxExposure](#)

3.2.8.5 SetAutoBrightnessMaxGain

Syntax

```
SetAutoBrightnessMaxGain (long lMaxGain)
```

Description

SetAutoBrightnessMaxGain sets the maximum value for automatic adjustment of the total gain.

Parameter

- **lMaxGain**: Maximum value of the automatic gain

Interface

- [Iuc480AutoFeatures](#)

Related functions

- [GetAutoBrightnessMaxGain](#)
- [SetAutoBrightnessMaxExposure](#)
- [SetAutoBrightnessReference](#)

3.2.8.6 GetAutoBrightnessMaxGain

Syntax

```
GetAutoBrightnessMaxGain (long* plMaxGain)
```

Description

GetAutoBrightnessMaxGain returns the maximum value for the automatic adjustment of the total gain.

Parameter

- **plMaxGain**: Pointer to the variable in which the maximum value of the automatic gain is written.

Interface

- [Iuc480AutoFeatures](#)

Related functions

- [SetAutoBrightnessMaxGain](#)

3.2.8.7 SetAutoBrightnessSpeed

Syntax

```
SetAutoBrightnessSpeed (long lSpeed)
```

Description

SetAutoBrightnessSpeed returns the adjustment speed for the automatic gain/automatic exposure time.

Parameter

- lSpeed: Automatic brightness adjustment speed from 1 % (slow) to 100 % (fast)

Interface

- [Iuc480AutoFeatures](#)

Related functions

- [GetAutoBrightnessSpeed](#)
- [SetAutoBrightnessReference](#)

3.2.8.8 GetAutoBrightnessSpeed

Syntax

```
GetAutoBrightnessSpeed (long* plSpeed)
```

Description

GetAutoBrightnessSpeed returns the adjustment speed for the automatic gain/automatic exposure time.

Parameter

- plSpeed: Pointer to variable in which the adjustment speed of the automatic brightness is written.

Interface

- [Iuc480AutoFeatures](#)

Related functions

- [SetAutoBrightnessSpeed](#)

3.2.8.9 SetAutoBrightnessAOI

Syntax

```
SetAutoBrightnessAOI (long lXPos, long lYPos, long lWidth, long lHeight)
```

Description

SetAutoBrightnessAOI sets the reference AOI for the automatic gain/automatic exposure time. The

actual value for automatic adjustment is determined from the reference AOI.

Parameter

lXPos	X position of the AOI
lYPos	Y position of the AOI
lWidth	Width of the AOI
lHeight	Height of the AOI

Interface

- [Iuc480AutoFeatures](#)

Related functions

- [GetAutoBrightnessAOI](#)

3.2.8.10 GetAutoBrightnessAOI

Syntax

```
GetAutoBrightnessAOI (long* plXPos, long* plYPos, long* plWidth, long* plHeight)
```

Description

GetAutoBrightnessAOI returns the reference AOI for automatic gain/automatic exposure time.

Parameter

plXPos	Pointer to the variable in which the X position of the AOI is written.
plYPos	Pointer to the variable in which the Y position of the AOI is written.
plWidth	Pointer to the variable in which the width of the AOI is written.
plHeight	Pointer to the variable in which the height of the AOI is written.

Interface

- [Iuc480AutoFeatures](#)

Related functions

- [SetAutoBrightnessAOI](#)

3.2.8.11 SetAutoWBGainOffsets

Syntax

```
SetAutoWBGainOffsets (long lRedOffset, long lBlueOffset)
```

Description

SetAutoWBGainOffsets sets the offset for the red and blue channel of the automatic white balance.

Parameter

lRedOffset	Offset of the red channel
lBlueOffset	Offset of the blue channel

Interface

- [Iuc480AutoFeatures](#)

Related functions

- [GetAutoWBGainOffsets](#)
- [SetAutoWBGainRange](#)

3.2.8.12 GetAutoWBGainOffsets

Syntax

```
GetAutoWBGainOffsets (long* pRedOffset, long* pBlueOffset)
```

Description

`GetAutoWBGainOffsets` returns the offset for the red and blue channel of the automatic white balance.

Parameter

<code>pRedOffset</code>	Pointer to the variable in which the offset of the red channel is written.
<code>pBlueOffset</code>	Pointer to the variable in which the offset of the blue channel is written.

Interface

- [Iuc480AutoFeatures](#)

Related functions

- [SetAutoWBGainOffsets](#)

3.2.8.13 SetAutoWBGainRange

Syntax

```
SetAutoWBGainRange (long lMinRGBGain, long lMaxRGBGain)
```

Description

`SetAutoWBGainRange` sets the color gain adjustment limits for the automatic white balance.

Parameter

<code>lMinRGBGain</code>	Lowest possible gain value
<code>lMaxRGBGain</code>	Highest possible gain value

Interface

- [Iuc480AutoFeatures](#)

Related functions

- [GetAutoWBGainRange](#)
- [SetAutoWBGainOffsets](#)

3.2.8.14 GetAutoWBGainRange

Syntax

```
GetAutoWBGainRange (long* pMinRGBGain, long* pMaxRGBGain)
```

Description

GetAutoWBGainRange returns the color gain adjustment limits for the automatic white balance.

Parameter

pMinRGBGain	Pointer to the variable in which the lowest possible gain value is written.
pMaxRGBGain	Pointer to the variable in which the highest possible gain value is written.

Interface

- [Iuc480AutoFeatures](#)

Related functions

- [SetAutoWBGainRange](#)

3.2.8.15 SetAutoWBSpeed

Syntax

```
SetAutoWBSpeed (long lSpeed)
```

Description

SetAutoWBSpeed sets the adjustment speed for the automatic white balance.

Parameter

- lSpeed: Adjustment speed of the automatic white balance from 1 % (slow) to 100 % (fast)

Interface

- [Iuc480AutoFeatures](#)

Related functions

- [GetAutoWBSpeed](#)
- [SetAutoWBGainOffsets](#)

3.2.8.16 GetAutoWBSpeed

Syntax

```
GetAutoWBSpeed (long* plSpeed)
```

Description

GetAutoWBSpeed returns the adjustment speed for the automatic white balance.

Parameter

- plSpeed: Pointer to the variable in which the adjustment speed of the automatic white balance is written.

Interface

- [Iuc480AutoFeatures](#)

Related functions

- [SetAutoWBSpeed](#)

3.2.8.17 SetAutoWBAOI

Syntax

SetAutoWBAOI (long lXPos, long lYPos, long lWidth, long lHeight)

Description

SetAutoWBAOI sets the reference AOI for the automatic white balance. The actual value for automatic adjustment is determined from the reference AOI.

Parameter

lXPos	X position of the AOI
lYPos	Y position of the AOI
lWidth	Width of the AOI
lHeight	Height of the AOI

Interface

- [Iuc480AutoFeatures](#)

Related functions

- [GetAutoWBAOI](#)

3.2.8.18 GetAutoWBAOI

Syntax

GetAutoWBAOI (long* plXPos, long* plYPos, long* plWidth, long* plHeight)

Description

GetAutoWBAOI returns the reference AOI for the automatic white balance.

Parameter

plXPos	Pointer to the variable in which the X position of the AOI is written.
plYPos	Pointer to the variable in which the Y position of the AOI is written.
plWidth	Pointer to the variable in which the width of the AOI is written.
plHeight	Pointer to the variable in which the height of the AOI is written.

Interface

- [Iuc480AutoFeatures](#)

Related functions

- [SetAutoWBAOI](#)

3.2.9 Iuc480AutoFramerate

IID_Iuc480AutoFramerate

This interface provides special functions for controlling the automatic frame rate control on some uc480 models that are not covered by the DirectShow standard. The purpose of the automatic frame rate control is to set the frame rate to an optimal value. As a result, the auto-exposure control has the required control range at the highest possible frame rate in all situations. To use the interface, you must integrate the `uc480CaptureInterface.h` header file into your project.

`Iuc480AutoFramerate` provides the following functions:

AutoFramerateSensor_IsSupported	Returns information on whether the sensor supports auto frame rate.
AutoFramerateSensor_IsEnabled	Returns information on whether the sensor's auto frame rate is enabled.
AutoFramerateSensor_Enable	Activates the sensor's auto frame rate.
AutoFramerateDriver_IsSupported	Returns information on whether the camera supports auto frame rate.
AutoFramerateDriver_IsEnabled	Returns information on whether the camera's auto frame rate is enabled.
AutoFramerateDriver_Enable	Activates the camera's auto frame rate.
AutoFramerate_GetFramerate	Returns the value set for the frame rate.

3.2.9.1 AutoFramerateSensor_IsSupported

Syntax

```
AutoFramerateSensor_IsSupported (bool* pbSupported)
```

Description

Returns information on whether the sensor supports auto frame rate.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480AutoFramerate](#)

Related functions

- [AutoFramerateSensor_IsEnabled](#)
- [AutoFramerateSensor_Enable](#)
- [AutoFramerate_GetFramerate](#)

3.2.9.2 AutoFramerateSensor_IsEnabled

Syntax

```
AutoFramerateSensor_IsEnabled (bool* pbEnabled)
```

Description

Returns information on whether the sensor's auto frame rate is enabled.

Parameter

- `pbEnabled`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480AutoFramerate](#)

Related functions

- [AutoFramerateSensor_IsSupported](#)
- [AutoFramerateSensor_Enable](#)
- [AutoFramerate_GetFramerate](#)

3.2.9.3 AutoFramerateSensor_Enable**Syntax**

```
AutoFramerateSensor_Enable (bool bEnable)
```

Description

Activates the sensor's auto frame rate.

Parameter

<code>bEnable</code>	<ul style="list-style-type: none"> • TRUE = enable auto frame rate • FALSE = disable auto frame rate
----------------------	------------------------------------------------------------------------------------------------------------------------------

Interface

- [Iuc480AutoFramerate](#)

Related functions

- [AutoFramerateSensor_IsEnabled](#)
- [AutoFramerate_GetFramerate](#)

3.2.9.4 AutoFramerateDriver_IsSupported**Syntax**

```
AutoFramerateDriver_IsSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports auto frame rate.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480AutoFramerate](#)

Related functions

- [AutoFramerateDriver_IsEnabled](#)
- [AutoFramerateDriver_Enable](#)

3.2.9.5 AutoFramerateDriver_IsEnabled

Syntax

```
AutoFramerateDriver_IsEnabled (bool* pbEnabled)
```

Description

Returns information on whether the camera's auto frame rate is enabled.

Parameter

- `pbEnabled`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480AutoFramerate](#)

Related functions

- [AutoFramerateDriver_IsSupported](#)
- [AutoFramerateDriver_Enable](#)

3.2.9.6 AutoFramerateDriver_Enable

Syntax

```
AutoFramerateDriver_Enable (bool bEnable)
```

Description

Activates the camera's auto frame rate.

Parameter

<code>bEnable</code>	<ul style="list-style-type: none">• TRUE = enable auto frame rate• FALSE = disable auto frame rate
----------------------	---------------------------------------------------------------------------------------------------------------------------

Interface

- [Iuc480AutoFramerate](#)

Related functions

- [AutoFramerateDriver_IsSupported](#)
- [AutoFramerateDriver_IsEnabled](#)

3.2.9.7 AutoFramerate_GetFramerate

Syntax

```
AutoFramerate_GetFramerate (double* dblFramerate)
```

Description

Returns the value set for the frame rate.

Parameter

- `dblFramerate`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480AutoFramerate](#)

Related functions

- [AutoFramerateSensor_IsSupported](#)
- [AutoFramerateSensor_IsEnabled](#)

3.2.10 Iuc480AutoParameter

IID_Iuc480AutoParameter

This interface provides special functions for controlling the automatic functions that are not covered by the DirectShow standard. This interface enables/disables the auto white balance. 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. In addition the supported color spaces are queried and set. To use the interface, you must integrate the `uc480CaptureInterface.h` header file into your project.

`Iuc480AutoParameter` provides the following functions:

AutoParameter_GetAWBType	Returns the current set type of the auto white balance.
AutoParameter_SetAWBType	Sets the type of the auto white balance.
AutoParameter_GetSupportedAWBType	Returns the supported types for auto white balance.
AutoParameter_GetEnableAWB	Returns if the auto white balance is enabled.
AutoParameter_SetEnableAWB	Enables/Disables the auto white balance.
AutoParameter_GetRGBColorModelAWB	Returns the current color space for the auto white balance.
AutoParameter_SetRGBColorModelAWB	Sets the color space for the auto white balance.
AutoParameter_GetSupportedRGBColorModelAWB	Returns the supported color spaces for the auto white balance.

3.2.10.1 AutoParameter_GetAWBType

Syntax

```
AutoParameter_GetAWBType (UINT* pnType)
```

Description

Returns the current set type of the auto white balance.

Parameter

- `pnTypes`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480CameraLUT](#)

Related functions

- [AutoParameter_GetSupportedAWBType](#)

- [AutoParameter_SetAWBType](#)

3.2.10.2 AutoParameter_SetAWBType

Syntax

AutoParameter_SetAWBType (UINT nType)

Description

Sets the type of the auto white balance:

- IS_AWB_GREYWORLD: 0x0001
- IS_AWB_COLOR_TEMPERATURE: 0x0002

Parameter

- nType: value to be set

Interface

- [Iuc480CameraLUT](#)

Related functions

- [AutoParameter_GetAWBType](#)
- [AutoParameter_GetSupportedAWBType](#)

3.2.10.3 AutoParameter_GetSupportedAWBType

Syntax

AutoParameter_GetSupportedAWBTypes (UINT* pnTypes)

Description

Returns the supported types for auto white balance.

Parameter

pnTypes	Pointer to the variable in which the return value is written: <ul style="list-style-type: none">• IS_AWB_GREYWORLD: 0x0001• IS_AWB_COLOR_TEMPERATURE: 0x0002
---------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Interface

- [Iuc480CameraLUT](#)

Related functions

- [AutoParameter_GetAWBType](#)
- [AutoParameter_SetAWBType](#)

3.2.10.4 AutoParameter_GetEnableAWB

Syntax

```
AutoParameter_GetEnableAWB (UINT* pnEnable)
```

Description

Returns if the auto white balance is enabled.

Parameter

pnEnable	<ul style="list-style-type: none"> • IS_AUTOPARAMETER_DISABLE: 0 • IS_AUTOPARAMETER_ENABLE: 1 • IS_AUTOPARAMETER_ENABLE_RUNONCE: 2
----------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------

Interface

- [Iuc480CameraLUT](#)

Related functions

- [AutoParameter_SetEnableAWB](#)

3.2.10.5 AutoParameter_SetEnableAWB

Syntax

```
AutoParameter_SetEnableAWB (UINT nEnable)
```

Description

Enables/Disables the auto white balance.

Parameter

nEnable	<ul style="list-style-type: none"> • IS_AUTOPARAMETER_DISABLE: 0 • IS_AUTOPARAMETER_ENABLE: 1 • IS_AUTOPARAMETER_ENABLE_RUNONCE: 2
---------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------

Interface

- [Iuc480CameraLUT](#)

Related functions

- [AutoParameter_GetEnableAWB](#)

3.2.10.6 AutoParameter_GetRGBColorModelAWB

Syntax

```
AutoParameter_GetRGBColorModelAWB (UINT* pnColorModel)
```

Description

Returns the current color space for the auto white balance.

Parameter

- pnColorModel: Pointer to the variable in which the return value is written.

Interface

- [Iuc480CameraLUT](#)

Related functions

- [AutoParameter_GetSupportedRGBColorModelAWB](#)
- [AutoParameter_SetRGBColorModelAWB](#)

3.2.10.7 AutoParameter_SetRGBColorModelAWB

Syntax

```
AutoParameter_SetRGBColorModelAWB (UINT nColorModel)
```

Description

Sets the color space for the auto white balance. (see [AutoParameter_GetSupportedRGBColorModelAWB](#))

Parameter

- nColorModel: value to be set

Interface

- [Iuc480CameraLUT](#)

Related functions

- [AutoParameter_GetRGBColorModelAWB](#)
- [AutoParameter_GetSupportedRGBColorModelAWB](#)

3.2.10.8 AutoParameter_GetSupportedRGBColorModelAWB

Syntax

```
AutoParameter_GetSupportedRGBColorModelAWB (UINT* pnSupported)
```

Description

Returns the supported color spaces for the auto white balance.



Note: You must pass a valid [type for the auto white balance](#) in pnSupported when calling the function. If no type is passed, the function returns IS_NOT_SUPPORTED.

Parameter

pnSupported	Pointer to the variable in which the return value is written (IS_AWB_COLOR_TEMPERATURE): <ul style="list-style-type: none">• RGB_COLOR_MODEL_SRGB_D50: 0x0001• RGB_COLOR_MODEL_SRGB_D65: 0x0002• RGB_COLOR_MODEL_CIE_RGB_E: 0x0004• RGB_COLOR_MODEL_ECI_RGB_D50: 0x0008• RGB_COLOR_MODEL_ADOBE_RGB_D65: 0x0010
-------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Interface

- [Iuc480CameraLUT](#)

Related functions

- [AutoParameter_GetRGBColorModelAWB](#)
- [AutoParameter_SetRGBColorModelAWB](#)

3.2.11 Iuc480CameraLUT

IID_Iuc480CameraLUT

This interface provides special functions for controlling the camera LUT that are not covered by the DirectShow standard. To use the interface, you must integrate the `uc480CaptureInterface.h` header file into your project.

Iuc480CameraLUT provides the following functions:

CameraLut_GetCameraLUT	Returns the current LUT values.
CameraLut_SetCameraLUT	Enables a hardware LUT for GigE and USB 3 uc480 cameras. This LUT which will be applied to the image in the camera.

3.2.11.1 CameraLUT_GetCameraLUT

Syntax

```
GetCameraLUT (UINT nMode, UINT nNumberOfEntries, double* pRed_Grey, double* pGreen, double* pBlue)
```

Description

GetCameraLUT returns the current LUT values. Using [CameraLUT_SetCameraLUT](#), you can select a different LUT for the camera.

Parameter

<div> <div></div> <div>nMode</div> </div>	
IS_GET_CAMERA_LUT_USER	Returns the LUT values set by the user without modifications.
IS_GET_CAMERA_LUT_COMPLETE	Returns the LUT values set by the user after the gamma, contrast and brightness values have been taken into account.
nNumberOfEntries	Number of the LUT values
IS_CAMERA_LUT_64	LUT with 64 values
pRed_Grey	Pointer to the array to which the red channel values or the grayscale value (GigE uc480 SE cameras) of the LUT are written.
pGreen	Pointer to the array to which the green channel values of the LUT are written.
pBlue	Pointer to the array to which the blue channel values of the LUT are written.

Interface

- [Iuc480CameraLUT](#)

Related functions

- [CameraLUT_SetCameraLUT](#)

3.2.11.2 CameraLUT_SetCameraLUT

Syntax

```
SetCameraLUT (UINT nMode, UINT nNumberOfEntries, double* pRed_Grey, double* pGreen, double* pBlue)
```

Description

Using `SetCameraLUT`, you can enable a hardware LUT for GigE and USB 3 uc480 cameras. This LUT which will be applied to the image in the camera. A number of predefined LUTs are available. Alternatively, you define your own LUT. It is possible to define a LUT without enabling it at the same time. You can query the current LUT used by the camera by calling the [GetCameraLUT](#).

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.

For further information on LUTs, please refer to the "LUT properties" chapter in the uc480 manual.

Parameter

nMode: These modes can be linked by a logical OR.	
IS_CAMERA_LUT_IDENTITY	Predefined LUT, linear LUT, no image modifications
IS_CAMERA_LUT_NEGATIV	Predefined LUT, inverts the image.
IS_CAMERA_LUT_GLOW1	Predefined LUT, false-color display of the image
IS_CAMERA_LUT_GLOW2	Predefined LUT, false-color display of the image
IS_CAMERA_LUT_ASTRO1	Predefined LUT, false-color display of the image
IS_CAMERA_LUT_RAINBOW1	Predefined LUT, false-color display of the image
IS_CAMERA_LUT_MAP1	Predefined LUT, false-color display of the image
IS_CAMERA_LUT_COLD_HOT	Predefined LUT, false-color display of the image
IS_CAMERA_LUT_SEPIC	Predefined LUT, uses sepia toning for coloring the image.
IS_CAMERA_LUT_ONLY_RED	Predefined LUT, shows only the red channel of the image.
IS_CAMERA_LUT_ONLY_GREEN	Predefined LUT, shows only the green channel of the image.
IS_CAMERA_LUT_ONLY_BLUE	Predefined LUT, shows only the blue channel of the image.
IS_SET_CAMERA_LUT_VALUES	Applies the LUT values.
IS_ENABLE_CAMERA_LUT	Enables the LUT. If no other LUT has been defined, the system sets the linear LUT as specified by IS_CAMERA_LUT_IDENTITY.
IS_ENABLE_RGB_GRAYSCALE	The camera converts a color image to a grayscale image.
nNumberOfEntries: Indicates the number of knee points used.	
IS_CAMERA_LUT_64	Defines a LUT with 64 knee points. This results in 32 sections with a start and end point each.
pRed_Grey	Array containing the values for the LUT red channel or the LUT grayscale channel (GigE uc480 SE cameras).
pGreen	Array containing the values for the LUT green channel.
pBlue	Array containing the values for the LUT blue channel.

Structure of the LUT arrays

The `pRed_Grey`, `pGreen` and `pBlue` arrays contain double values between 0.0 and 1.0. The array size must correspond exactly to the value predefined by `NumberOfEntries` (64 or 128).

GigE uc480 SE/RE/LE/CP and USB 3 uc480 CP color and monochrome cameras ignore the `pGreen` and `pBlue` parameters.

GigE uc480 HE cameras use all three parameters: `pRed_Grey`, `pGreen` and `pBlue`. If you set all three parameters to the same value for GigE uc480 HE monochrome cameras, the LUT curve creates a monochrome output image. Assigning different values to the three parameters will result in false-color representation.

Interface

- [Iuc480CameraLUT](#)

Related functions

- [CameraLUT_GetCameraLUT](#)

3.2.12 Iuc480Capture

IID_Iuc480Capture

This interface provides the uc480 camera with several special functions that are not covered by the DirectShow standard. To use the interface, you must integrate the `uc480CaptureInterface.h` header file into your project.



The functions of the [Iuc480Capture](#), [Iuc480CaptureEx](#) and [Iuc480AutoFeatures](#) interfaces are only supported by uc480 cameras. Programs that use these interfaces are no longer fully generic.

`Iuc480Capture` provides the following functions:

GetDeviceInfo	Returns information about a camera.
GetDLLVersion	Returns the version of the installed uc480 camera driver.
HotPixel	Configures the sensor hot pixel correction
SetBadPixelCorrection	Enables hot pixel correction.
GetBadPixelCorrection	Returns the status of hot pixel correction.
SaveSettings	Saves the current camera settings in the Windows registry.
LoadSettings	Loads saved camera settings from the Windows registry.
ResetDefaults	Resets all camera parameters to the default values.
GetWhiteBalanceMultipliers	Returns the current RGB gain values.
SetWhiteBalanceMultipliers	Sets new RGB gain values.

3.2.12.1 GetDeviceInfo

Syntax

```
GetDeviceInfo (SENSORINFO *psInfo, CAMERAINFO *pcInfo)
```

Description

`GetDeviceInfo` returns information on the camera and sensor.

Parameter

<code>psInfo</code>	Pointer to the <code>SENSORINFO</code> structure that contains information about the sensor.
<code>pcInfo</code>	Pointer to the <code>CAMERAINFO</code> structure that contains information about the camera.

Contents of the SENSORINFO structure

<code>WORD</code>	<code>SensorID</code>	Returns the sensor type (e.g.: <code>IS_SENSOR_UI224X_C</code>)
-------------------	-----------------------	------------------------------------------------------------------

Char	strSensorName[32]	Returns the camera model (e.g.: UI224xLE-C)
Char	nColorMode	Returns the sensor color mode <ul style="list-style-type: none"> • IS_COLORMODE_BAYER • IS_COLORMODE_MONOCHROME
DWORD	nMaxWidth	Returns the maximum frame width
DWORD	nMaxHeight	Returns the maximum frame height
BOOL	bMasterGain	Indicates whether the sensor provides analog total gain
BOOL	bRGain	Indicates whether the sensor provides analog red gain
BOOL	bGGain	Indicates whether the sensor offers analog green gain
BOOL	bBGain	Indicates whether the sensor offers analog blue gain
BOOL	bGlobShutter	Indicates whether the sensor has a global shutter <ul style="list-style-type: none"> • TRUE = GlobalShutter • FALSE = RollingShutter
Char	Reserved[16]	Reserved

Contents of the CAMERAINFO structure

char	SerNo[12]	Camera's serial number
char	ID[20]	Camera manufacturer (e.g. IDS GmbH)
char	Version[10]	For USB cameras, this contains the hardware version of the USB board (e.g. V2.10)
char	Date[12]	System date of the final quality test (e.g. 01.08.2012)
unsigned char	Select	Camera ID
unsigned char	Type	<div> <div>Camera type</div> <div> IS_CAMERA_TYPE_uc480_USB_SE IS_CAMERA_TYPE_uc480_USB_ME IS_CAMERA_TYPE_uc480_USB_RE IS_CAMERA_TYPE_uc480_USB_LE IS_CAMERA_TYPE_uc480_ETH_HE IS_CAMERA_TYPE_uc480_ETH_SE </div> <div> USB uc480 SE USB uc480 ME USB uc480 RE USB uc480 LE GigE uc480 HE GigE uc480 SE </div> </div>
char	Reserved[8]	Reserved

Interface

- [Iuc480Capture](#)

3.2.12.2 GetDLLVersion

Syntax

```
GetDLLVersion (long *pversion)
```

Description

GetDLLVersion returns the version of the installed uc480 camera driver (uc480_api.dll).



The version number of the uc480 API may differ from the version number of the uc480 DirectShow interface uc480Capture.ax.

Parameter

pversion	Pointer to the variable in which the version number is written with the following encoding: <ul style="list-style-type: none">• Bits 31-24: Major version• Bits 23-16: Minor version• Bits 15-0: Build version
----------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Interface

- [Iuc480Capture](#)

3.2.12.3 HotPixel

Syntax

```
HotPixel(UINT nMode, void *pParam, UINT SizeOfParam)
```

Description

`HotPixel` configures the correction of sensor hot pixels. 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.

Via the [Iuc480HotPixel](#) interface you can call the single functions directly.

Parameter

nMode	Mode for hot pixel correction
IS_HOTPIXEL_DISABLE_CORRECTION	Disables hot pixel correction
IS_HOTPIXEL_ENABLE_CAMERA_CORRECTION	Enables hot pixel correction using the hot pixel list(s) stored in the camera EEPROM.
IS_HOTPIXEL_ENABLE_SOFTWARE_USER_CORRECTION	Enables hot pixel correction using the user's hot pixel list stored in the computer.
IS_HOTPIXEL_ENABLE_SENSOR_CORRECTION	Enables sensor's own hot pixel correction function (if available).
IS_HOTPIXEL_DISABLE_SENSOR_CORRECTION	Disables the sensor's own hot pixel correction function.
IS_HOTPIXEL_GET_CORRECTION_MODE	Returns the currently set hot pixel correction mode.
IS_HOTPIXEL_GET_SUPPORTED_CORRECTION_MODES	Returns the supported hot pixel correction modes.
IS_HOTPIXEL_GET_SOFTWARE_USER_LIST_EXISTS	Indicates whether the user-defined hot pixel list exists in the computer.
IS_HOTPIXEL_GET_SOFTWARE_USER_LIST_NUMBER	Returns the number of hot pixels in the user-defined hot pixel list stored in the computer.
IS_HOTPIXEL_GET_SOFTWARE_USER_LIST	Returns the user-defined hot pixel list stored in the computer.
IS_HOTPIXEL_SET_SOFTWARE_USER_LIST	Sets the user-defined hot pixel list that is stored in the computer.

IS_HOTPIXEL_SAVE_SOFTWARE_USER_LIST IS_HOTPIXEL_SAVE_SOFTWARE_USER_LIST_UNICODE	Saves the user-defined hot pixel list to a file in the computer. The function can also be used with Unicode file names.
IS_HOTPIXEL_LOAD_SOFTWARE_USER_LIST IS_HOTPIXEL_LOAD_SOFTWARE_USER_LIST_UNICODE	Loads the user-defined hot pixel list from a file. The function can also be used with Unicode file names.
IS_HOTPIXEL_GET_CAMERA_FACTORY_LIST_EXISTS	Indicates whether the factory-set hot pixel list exists.
IS_HOTPIXEL_GET_CAMERA_FACTORY_LIST_NUMBER	Returns the number of hot pixels in the factory-set hot pixel list.
IS_HOTPIXEL_GET_CAMERA_FACTORY_LIST	Returns the factory-set hot pixel list.
IS_HOTPIXEL_GET_CAMERA_USER_LIST_EXISTS	Indicates whether the user-defined hot pixel list exists in the camera EEPROM.
IS_HOTPIXEL_GET_CAMERA_USER_LIST_NUMBER	Returns the number of hot pixels in the user-defined hot pixel list stored in the camera EEPROM.
IS_HOTPIXEL_GET_CAMERA_USER_LIST	Returns the user-defined hot pixel list stored in the camera EEPROM.
IS_HOTPIXEL_SET_CAMERA_USER_LIST	Sets the user-defined hot pixel list stored in the camera EEPROM.
IS_HOTPIXEL_DELETE_CAMERA_USER_LIST	Deletes the user-defined hot pixel list from the camera EEPROM.
IS_HOTPIXEL_GET_CAMERA_USER_LIST_MAX_NUMBER	Returns the maximum number of hot pixels that the user can store in the camera EEPROM.
IS_HOTPIXEL_GET_MERGED_CAMERA_LIST_NUMBER	Returns the number of hot pixels in a merged list that combines the entries from the factory-set hot pixel list with those of the user-defined hot pixels list stored in the camera EEPROM.
IS_HOTPIXEL_GET_MERGED_CAMERA_LIST	Returns the merged list.
pParam	Pointer to a function parameter; which function parameter is referred to here depends on nMode.
SizeOfParam	Size (in bytes) of the memory area to which pParam refers.

Interface

- [Iuc480Capture](#)

3.2.12.4 SetBadPixelCorrection

Syntax

```
SetBadPixelCorrection(long lEnable)
```

Description

`SetBadPixelCorrection` enables/disables hot pixel correction.

Parameter

<code>lEnable</code>	<ul style="list-style-type: none">• 1 = hot pixel correction enabled• 0 = hot pixel correction disabled
----------------------	--------------------------------------------------------------------------------------------------------------------------------

Interface

- [Iuc480Capture](#)

Related functions

- [GetBadPixelCorrection](#)

3.2.12.5 GetBadPixelCorrection

Syntax

```
GetBadPixelCorrection (long *plEnable)
```

Description

`GetBadPixelCorrection` returns the status of the hot pixel correction.

Parameter

<code>plEnable</code>	Pointer to the variable in which the current status of the hot pixel correction is written: <ul style="list-style-type: none">• 1 = hot pixel correction enabled• 0 = hot pixel correction disabled
-----------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Interface

- [Iuc480Capture](#)

Related functions

- [SetBadPixelCorrection](#)

3.2.12.6 SaveSettings

Syntax

```
SaveSettings (void)
```

Description

`SaveSettings` saves the current camera settings in the Windows registry. The settings are loaded automatically each time the camera is opened.

Parameter

None

Interface

- [Iuc480Capture](#)

Related functions

- [LoadSettings](#)
- [ResetDefaults](#)

3.2.12.7 LoadSettings

Syntax

`LoadSettings (void)`

Description

`LoadSettings` loads saved camera settings from the Windows registry. The settings are also loaded automatically each time the camera is opened.

Parameter

None

Interface

- [Iuc480Capture](#)

Related functions

- [SaveSettings](#)
- [ResetDefaults](#)

3.2.12.8 ResetDefaults

Syntax

`ResetDefaults (void)`

Description

`ResetDefaults` sets all camera parameters to the default values.

Parameter

None

Interface

- [Iuc480Capture](#)

Related functions

- [LoadSettings](#)
- [SaveSettings](#)

3.2.12.9 GetWhiteBalanceMultipliers

Syntax

```
GetWhiteBalanceMultipliers (long *plRed, long *plGreen, long *plBlue)
```

Description

`GetWhiteBalanceMultipliers` returns the current RGB gain values in percent.

Parameter

<code>plRed</code>	Pointer to the variable in which the current red channel gain value is written.
<code>plGreen</code>	Pointer to the variable in which the current green channel gain value is written.
<code>plBlue</code>	Pointer to the variable in which the current blue channel gain value is written.

Interface

- [Iuc480Capture](#)

Related functions

- [SetWhiteBalanceMultipliers](#)

3.2.12.10 SetWhiteBalanceMultipliers

Syntax

```
SetWhiteBalanceMultipliers (long lRed, long lGreen, long lBlue)
```

Description

`SetWhiteBalanceMultipliers` sets new RGB gain values.

Parameter

<code>lRed</code>	Sets the red channel gain factor. The value to be set must be given in percent. That means a value of 100 corresponds to a factor of 1.0.
<code>lGreen</code>	Sets the green channel gain factor. The value to be set must be given in percent. That means a value of 100 corresponds to a factor of 1.0.
<code>lBlue</code>	Sets the blue channel gain factor. The value to be set must be given in percent. That means a value of 100 corresponds to a factor of 1.0.

Interface

- [Iuc480Capture](#)

Related functions

- [GetWhiteBalanceMultipliers](#)

3.2.13 Iuc480CaptureEx

IID_Iuc480CaptureEx

This interface provides additional special functions for the uc480 camera that are not covered by the DirectShow standard. To use the interface, you must integrate the `uc480CaptureInterface.h` header file into your project.

`Iuc480CaptureEx` provides the following functions:

SetGainBoost	Enables analog gain boost on the sensor.
GetGainBoost	Returns the status of the analog gain boost on the sensor.
SetHardwareGamma	Enables hardware gamma correction for GigE uc480 cameras.
GetHardwareGamma	Returns the status of the hardware gamma correction for GigE uc480 cameras.
LoadParameters	Loads saved camera settings from a uc480 parameter file (INI file) or from the EEPROM memory of the camera.
SaveParameters	Saves the current camera settings to a uc480 parameter file (INI file) or the EEPROM memory of the camera.
ParameterSet	Saves the current camera parameters to a file or to the EEPROM of the camera and loads the parameter set from a file or the EEPROM. Only camera-specific ini files can be loaded. In the uc480 manual appendix the structure of a uc480 ini file is described.

3.2.13.1 SetGainBoost

Syntax

```
SetGainBoost (long lGainBoost)
```

Description

SetGainBoost enables the analog gain boost on the sensor.

Parameter

lGainBoost	<ul style="list-style-type: none"> • 1 = enable analog gain boost • 0 = disable analog gain boost
------------	---------------------------------------------------------------------------------------------------------------------------

Interface

- [Iuc480CaptureEx](#)

Related functions

- [GetGainBoost](#)

3.2.13.2 GetGainBoost

Syntax

```
GetGainBoost (long *plGainBoost)
```

Description

GetGainBoost returns the status of the analog gain boost on the sensor.

Parameter

<code>plGainBoost</code>	Pointer to the variable in which the status of the analog gain boost is written: <ul style="list-style-type: none">• 1 = enabled• 0 = disabled
--------------------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------

Interface

- [Iuc480CaptureEx](#)

Related functions

- [SetGainBoost](#)

3.2.13.3 SetHardwareGamma_2

Syntax

```
SetHardwareGamma (long lHWGamma)
```

Description

`SetHardwareGamma` enables hardware gamma correction for GigE uc480 cameras.

Parameter

<code>lHWGamma</code>	<ul style="list-style-type: none">• 1 = enable hardware gamma correction• 0 = disable hardware gamma correction
-----------------------	----------------------------------------------------------------------------------------------------------------------------------------

Interface

- [Iuc480CaptureEx](#)

Related functions

- `GetHardwareGamma`

3.2.13.4 GetHardwareGamma_2

Syntax

```
GetHardwareGamma (long *plHWGamma)
```

Description

`GetHardwareGamma` returns the status of the hardware gamma correction for GigE uc480 cameras.

Parameter

<code>plHWGamma</code>	Pointer to the variable in which the status of the hardware gamma correction is written: 1 = enabled 0 = disabled
------------------------	-------------------------------------------------------------------------------------------------------------------------

Interface

- [Iuc480CaptureEx](#)

Related functions

- [SetHardwareGamma](#)

3.2.13.5 LoadParameters

Syntax

```
LoadParameters (const char* cszFileName)
```

Description

`LoadParameters` loads saved camera settings from a uc480 parameter file (INI file) or from the EEPROM memory of the camera.



Only camera-specific ini files can be loaded. The "uc480 parameter" file chapter in the uc480 manual describes the structure of a uc480 ini file.

When loading an ini file, ensure that the memory that has already been allocated matches the parameters of the ini file concerning image size (AOI) and color depth. If this is not the case, display errors may occur. The camera type specified in the ini file must match the opened camera type.

Parameter

<code>cszFileName</code>	<p>Pointer to file name. Either the absolute path or the relative path can be specified. For the camera's internal parameter sets, these would be:</p> <ul style="list-style-type: none"> • "\\cam\\set1" or "/cam/set1" <p>A parameter of <code>NULL</code> displays the "Open File" window.</p>
--------------------------	------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Interface

- [Iuc480CaptureEx](#)

Related functions

- [SaveParameters](#)
- [ResetDefaults](#)

3.2.13.6 SaveParameters

Syntax

```
SaveParameters (const char* cszFileName)
```

Description

`SaveParameters` saves the current camera settings to a uc480 parameter file (INI file) or the EEPROM memory of the camera.

Parameter

<code>cszFileName</code>	<p>Pointer to file name. Either the absolute path or the relative path can be specified. For the camera's internal parameter sets, these would be:</p> <ul style="list-style-type: none"> • "\\cam\\set1" or "/cam/set1" <p>Passing <code>NULL</code> displays the "Save As" window.</p>
--------------------------	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Interface

- [Iuc480CaptureEx](#)

Related functions

- [LoadParameters](#)
- [ResetDefaults](#)

3.2.13.7 ParameterSet

Syntax

```
ParametersSet(UINT nMode, void *pParam, UINT SizeOfParam)
```

Description

`ParameterSet` saves the current camera parameters to a file or to the EEPROM of the camera and loads the parameter set from a file or the EEPROM. Only camera-specific ini files can be loaded. In the uc480 manual appendix the structure of a uc480 ini file is described.



When loading an ini file, ensure that the memory that has already been allocated matches the parameters of the ini file concerning image size (AOI) and color depth. If this is not the case, display errors may occur.

Parameter

nMode	
IS_PARAMETERSET_CMD_LOAD_EEPROM	Loads a camera parameter set from the EEPROM
IS_PARAMETERSET_CMD_LOAD_FILE	Loads a camera parameter set from a file. You must pass the path to the ini file as Unicode string. You can pass either a relative or an absolute path. If you pass <code>NULL</code> the "Open file" dialog opens.
IS_PARAMETERSET_CMD_SAVE_EEPROM	Saves a camera parameter set in the EEPROM.
IS_PARAMETERSET_CMD_SAVE_FILE	Saves a camera parameter set in a file. You must pass the path to the ini file as Unicode string. You can pass either a relative or an absolute path. If you pass <code>NULL</code> the "Save as" dialog opens.
IS_PARAMETERSET_CMD_GET_NUMBER_SUPPORTED	Returns the number of supported parameter sets in the camera EEPROM. At the moment this is "1" for all cameras.
IS_PARAMETERSET_CMD_GET_HW_PARAMETERSET_AVAILABLE	Returns if a camera parameter set in the EEPROM is supported.
IS_PARAMETERSET_CMD_ERASE_HW_PARAMETERSET	Deletes the camera parameter set in the EEPROM.
pParam	Pointer to a function parameter; which function parameter is referred to here depends on <code>nMode</code> .
SizeOfParam	Size (in bytes) of the memory area to which <code>pParam</code> refers.

Interface

- [Iuc480CaptureEx](#)

Related functions

- [ResetDefaults](#)

3.2.14 Iuc480ColorConverter

IID_Iuc480ColorConverter

This interface provides additional special functions for color conversion that are not covered by the DirectShow standard. To use the interface, you must integrate the `uc480CaptureInterface.h` header file into your project.

`Iuc480ColorConverter` provides the following functions:

ColorConverter_GetCurrentMode	Returns the current set mode for color conversion.
ColorConverter_GetDefaultMode	Returns the default mode for color conversion.
ColorConverter_GetSupportedModes	Returns the supported modes for color conversion.
ColorConverter_SetMode	Sets the mode for color conversion.

3.2.14.1 ColorConverter_GetCurrentMode

Syntax

```
ColorConverter_GetCurrentMode( INT* piConvertMode )
```

Description

Returns the current set mode for color conversion. The return value depends on the selected color mode.

Parameter

- `piConvertMode`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480ColorConverter](#)

Related functions

- [ColorConverter_GetDefaultMode](#)
- [ColorConverter_GetSupportedModes](#)
- [ColorConverter_SetMode](#)

3.2.14.2 ColorConverter_GetDefaultMode

Syntax

```
ColorConverter_GetDefaultMode( INT* piConvertMode )
```

Description

Returns the default mode for color conversion.

Parameter

- `piConvertMode`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480ColorConverter](#)

Related functions

- [ColorConverter_GetCurrentMode](#)
- [ColorConverter_GetSupportedModes](#)
- [ColorConverter_SetMode](#)

3.2.14.3 ColorConverter_GetSupportedModes

Syntax

```
ColorConverter_GetSupportedModes(INT* piConvertMode)
```

Description

Returns the supported modes for color conversion. Possible converters are:

- `IS_CONV_MODE_NONE`: No conversion
- `IS_CONV_MODE_SOFTWARE`: Only for monochrome cameras, if you want to add a gamma
- `IS_CONV_MODE_SOFTWARE_3X3`: Software conversion using the standard filter mask (default)
- `IS_CONV_MODE_SOFTWARE_5X5`: Software conversion using a large filter mask
- `IS_CONV_MODE_HARDWARE_3X3`: Hardware conversion using the standard filter mask (GigE uc480 HE/USB 3 uc480 CP)
- `IS_CONV_MODE_OPENCL_3X3`: Software conversion using the standard filter mask, but conversion is done on the graphic board
- `IS_CONV_MODE_OPENCL_5X5`: Software conversion using the standard filter mask, but conversion is done on the graphic board
- `IS_CONV_MODE_JPEG`: Hardware conversion into JPEG format (XS only)

Parameter

- `piConvertMode`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480ColorConverter](#)

Related functions

- [ColorConverter_GetCurrentMode](#)
- [ColorConverter_GetDefaultMode](#)
- [ColorConverter_SetMode](#)

3.2.14.4 ColorConverter_SetMode

Syntax

```
ColorConverter_SetMode(INT iConvertMode)
```

Description

Sets the mode for color conversion.

Parameter

- `iConvertMode`: value to be set

Example

```
/* Enable JPEG mode (XS only) */
status = pIuc480ColorConverter->ColorConverter_SetMode(IS_CONV_MODE_JPEG);

/* Disable JPEG mode and enable software conversion */
status = pIuc480ColorConverter->ColorConverter_SetMode(IS_CONV_MODE_SOFTWARE);
```

Interface

- [Iuc480ColorConverter](#)

Related functions

- [ColorConverter_GetCurrentMode](#)
- [ColorConverter_GetDefaultMode](#)
- [ColorConverter_GetSupportedModes](#)

3.2.15 Iuc480ColorTemperature

IID_Iuc480ColorTemperature

This interface provides special functions for setting the color temperature that are not covered by the DirectShow standard. For color cameras, this can be used to set the image color temperature to a specific value (in Kelvin). As far as possible, the function uses the sensor's hardware gain controller for this. It is also possible to choose between different color spaces. Depending on the color space, a specific color temperature results in slightly different RGB values. To use the interface, you must integrate the `uc480CaptureInterface.h` header file into your project.

`Iuc480ColorTemperature` provides the following functions:

RGBModel_IsSupported	Returns information on whether the camera supports configurable color spaces.
RGBModel_GetMode	Returns the set color space.
RGBModel_SetMode	Sets a color space.
RGBModel_GetDefaultMode	Returns the default color space.
RGBModel_GetSupportedModes	Returns the supported color spaces.
ColorTemperature_IsSupported	Returns information on whether the camera supports configurable color temperatures.
ColorTemperature_GetValue	Returns the value set for the color temperature.
ColorTemperature_SetValue	Sets the value for the color temperature.
ColorTemperature_GetDefaultValue	Returns the default value for the color temperature.
ColorTemperature_GetRange	Returns the range for the color temperature.

3.2.15.1 RGBModel_IsSupported

Syntax

```
RGBModel_IsSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports configurable color spaces.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480ColorTemperature](#)

Related functions

- [RGBModel_GetSupportedModes](#)
- [RGBModel_GetDefaultMode](#)
- [RGBModel_GetMode](#)

3.2.15.2 RGBModel_GetMode

Syntax

```
RGBModel_GetMode (unsigned long* pulMode)
```

Description

Returns the set color space.

Parameter

- `pulMode`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480ColorTemperature](#)

Related functions

- [RGBModel_SetMode](#)
- [RGBModel_GetSupportedModes](#)
- [RGBModel_IsSupported](#)

3.2.15.3 RGBModel_SetMode

Syntax

```
RGBModel_SetMode (unsigned long ulMode)
```

Description

Sets a color space.

Parameter

ulMode	Value to be set.
RGB_COLOR_MODEL_SRGB_D50	sRGB (standard RGB) color space with a white point of 5000 Kelvin (warm light)
RGB_COLOR_MODEL_SRGB_D65	sRGB (standard RGB) color space with a white point of 6500 Kelvin (medium daylight)
RGB_COLOR_MODEL_CIE_RGB_E	CIE RGB color space with standard illumination E
RGB_COLOR_MODEL_ECI_RGB_D50	ECI RGB color space with a white point of 5000 Kelvin (warm light)
RGB_COLOR_MODEL_ADOBE_RGB_D65	Adobe RGB color space with a white point of 6500 Kelvin (medium daylight). The Adobe RGB color space is larger than the sRGB color space but cannot be returned by some devices.

Interface

- [Iuc480ColorTemperature](#)

Related functions

- [RGBModel_GetSupportedModes](#)
- [RGBModel_GetDefaultMode](#)
- [RGBModel_GetMode](#)

3.2.15.4 RGBModel_GetDefaultMode

Syntax

`RGBModel_GetDefaultMode (unsigned long* pulMode)`

Description

Returns the default color space.

Parameter

- `pulMode`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480ColorTemperature](#)

Related functions

- [RGBModel_GetSupportedModes](#)
- [RGBModel_GetMode](#)

3.2.15.5 RGBModel_GetSupportedModes

Syntax

`RGBModel_GetSupportedModes (unsigned long* pulModes)`

Description

Returns the supported color spaces.

Parameter

- `pulModes`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480ColorTemperature](#)

Related functions

- [RGBModel_GetDefaultMode](#)
- [RGBModel_GetMode](#)

3.2.15.6 ColorTemperature_IsSupported

Syntax

```
ColorTemperature_IsSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports configurable color temperatures.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480ColorTemperature](#)

Related functions

- [ColorTemperature_GetDefaultValue](#)
- [ColorTemperature_GetRange](#)
- [ColorTemperature_GetValue](#)

3.2.15.7 ColorTemperature_GetValue

Syntax

```
ColorTemperature_GetValue (unsigned long* pulValue)
```

Description

Returns the value set for the color temperature.

Parameter

- `pulValue`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480ColorTemperature](#)

Related functions

- [ColorTemperature_GetDefaultValue](#)
- [ColorTemperature_GetRange](#)
- [ColorTemperature_SetValue](#)

3.2.15.8 ColorTemperature_SetValue

Syntax

```
ColorTemperature_SetValue (unsigned long ulValue)
```

Description

Sets the value for the color temperature in Kelvin.

Parameter

- `ulValue`: value to be set

Interface

- [Iuc480ColorTemperature](#)

Related functions

- [ColorTemperature_GetDefaultValue](#)
- [ColorTemperature_GetRange](#)
- [ColorTemperature_GetValue](#)

3.2.15.9 ColorTemperature_GetDefaultValue

Syntax

```
ColorTemperature_GetDefaultValue (unsigned long* pulDefValue)
```

Description

Returns the default value for the color temperature.

Parameter

- `pulDefValue`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480ColorTemperature](#)

Related functions

- [ColorTemperature_GetRange](#)
- [ColorTemperature_GetValue](#)

3.2.15.10 ColorTemperature_GetRange

Syntax

```
ColorTemperature_GetRange (unsigned long* pulMin, unsigned long* pulMax, unsigned long* pulInc)
```

Description

Returns the range for the color temperature.

Parameter

<code>pulMin</code>	Pointer to the variable in which the minimum value is written.
<code>pulMax</code>	Pointer to the variable in which the maximum value is written.
<code>pulInc</code>	Pointer to the variable in which the increment is written.

Interface

- [Iuc480ColorTemperature](#)

Related functions

- [ColorTemperature_GetDefaultValue](#)
- [ColorTemperature_GetValue](#)

3.2.16 Iuc480DeviceFeature

IID_Iuc480DeviceFeature

This interface provides special functions for controlling camera-specific functions that are not covered by the DirectShow standard. To use the interface, you must integrate the `Iuc480CaptureInterface.h` header file into your project.

`Iuc480DeviceFeature` provides the following functions:

DCx Camera DirectShow Programming Interface

<u>DeviceFeature_GetSupportedFeatures</u>	Returns the functions supported by the camera.
<u>DeviceFeature_GetAllowRawWithLUT</u>	Returns if the camera LUT can be used in combination with RAW formats.
<u>DeviceFeature_SetAllowRawWithLUT</u>	If the value 1 is passed, the camera LUT can also be used with RAW formats.
<u>DeviceFeature_GetDefaultLogMode</u>	DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N, DCC3260M, DCC3260C: Returns the default settings for the Log mode.
<u>DeviceFeature_GetLogMode</u>	DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N, DCC3260M, DCC3260C: Returns the current Log mode.
<u>DeviceFeature_SetLogMode</u>	DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N, DCC3260M, DCC3260C: Sets the Log mode.
<u>DeviceFeature_GetLogModeManualGain</u>	DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N, DCC3260M, DCC3260C: Returns the current manual gain of the Log mode.
<u>DeviceFeature_GetLogModeManualGainDefault</u>	DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N, DCC3260M, DCC3260C: Returns the default settings for the manual gain for the Log mode.
<u>DeviceFeature_GetLogModeManualGainRange</u>	DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N, DCC3260M, DCC3260C: Returns the range for the manual gain of the Log mode.
<u>DeviceFeature_SetLogModeManualGain</u>	DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N, DCC3260M, DCC3260C: Sets the manual gain of the Log mode.
<u>DeviceFeature_GetLogModeManualValue</u>	DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N, DCC3260M, DCC3260C: Returns the current manual value of the Log mode.
<u>DeviceFeature_GetLogModeManualValueDefault</u>	DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N, DCC3260M, DCC3260C: Returns the default settings for the manual value of the Log mode.
<u>DeviceFeature_GetLogModeManualValueRange</u>	DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N, DCC3260M, DCC3260C: Returns the range of the manual value of the Log mode.
<u>DeviceFeature_SetLogModeManualValue</u>	DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N, DCC3260M, DCC3260C: Sets the manual value of the Log mode.
<u>DeviceFeature_GetLineScanMode</u>	DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N, DCC3260M, DCC3260C: Returns the current set line scan mode.
<u>DeviceFeature_SetLineScanMode</u>	DCC1240C DCC1240M, DCC3240C,

3.2.16.1 DeviceFeature_GetSupportedFeatures

Syntax

```
DeviceFeature_GetSupportedFeatures (INT* pnCap)
```

Description

Returns the functions supported by the camera.

IS_DEVICE_FEATURE_CAP_SHUTTER_MODE_ROLLING	Rolling shutter mode is supported/Set mode
IS_DEVICE_FEATURE_CAP_SHUTTER_MODE_ROLLING_GLOBAL_START	Rolling shutter mode with global start is supported/Set mode
IS_DEVICE_FEATURE_CAP_SHUTTER_MODE_GLOBAL	Global shutter mode is supported/Set mode
IS_DEVICE_FEATURE_CAP_SHUTTER_MODE_GLOBAL_ALTERNATIVE_TIMING	Global shutter mode with different timing parameters is supported/Set mode
IS_DEVICE_FEATURE_CAP_LINESCAN_MODE_FAST	Fast line scan mode is supported/Set mode
IS_DEVICE_FEATURE_CAP_LINESCAN_NUMBER	Line number at fast line scan mode is supported/Set number
IS_DEVICE_FEATURE_CAP_PREFER_XS_HS_MODE	HS mode is supported/Set mode
IS_DEVICE_FEATURE_CAP_LOG_MODE	<p>Log mode is supported/Set mode</p> <ul style="list-style-type: none"> IS_LOG_MODE_FACTORY_DEFAULT: Factory setting for the Log mode IS_LOG_MODE_OFF: Log mode off IS_LOG_MODE_MANUAL: Manual Log mode. In this case the LogMode value and the LogMode gain are effective. IS_LOG_MODE_AUTO: Automatic Log mode (default setting)
IS_DEVICE_FEATURE_CAP_VERTICAL_AOI_MERGE	<p>Special AOI merge mode which combines the lines of an AOI to a new image (DCC3240C, DCC3240M, DCC3240N: only).</p> <p>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.</p> <ul style="list-style-type: none"> IS_VERTICAL_AOI_MERGE_MODE_OFF: Disables the AOI merge mode IS_VERTICAL_AOI_MERGE_MODE_ON: Enables the AOI merge mode IS_VERTICAL_AOI_MERGE_MODE_FREERUN: The sensor runs with maximum speed. IS_VERTICAL_AOI_MERGE_MODE_TRIGGERED_SOFTWARE: The sensor is triggered via software. IS_VERTICAL_AOI_MERGE_MODE_TRIGGERED_FALLING_GPIO1: The sensor is triggered on GPIO 1 (falling edge). IS_VERTICAL_AOI_MERGE_MODE_TRIGGERED_RISING_GPIO1: The sensor is triggered on GPIO 1 (rising edge). IS_VERTICAL_AOI_MERGE_MODE_TRIGGERED_FALLING_GPIO2: The sensor is triggered on GPIO 2 (falling edge). IS_VERTICAL_AOI_MERGE_MODE_TRIGGERED_RISING_GPIO2: The sensor is triggered on GPIO 2 (rising edge).

Parameter

- `pnCap`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480DeviceFeature](#)

3.2.16.2 DeviceFeature_GetAllowRawWithLUT

Syntax

```
DeviceFeature_GetAllowRawWithLUT (UINT* pnAllowRawWithLut)
```

Description

Returns if the camera LUT can be used in combination with RAW formats.

Parameter

- `pnAllowRawWithLut`: Pointer to the variable in which the return value is written (0 = FALSE, 1 = TRUE).

Interface

- [Iuc480DeviceFeature](#)

Related functions

- [DeviceFeature_SetAllowRawWithLUT](#)

3.2.16.3 DeviceFeature_SetAllowRawWithLUT

Syntax

```
DeviceFeature_SetAllowRawWithLUT (UINT nAllowRawWithLut)
```

Description

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

- `nAllowRawWithLut`: value to be set

Interface

- [Iuc480DeviceFeature](#)

Related functions

- [DeviceFeature_GetAllowRawWithLUT](#)

3.2.16.4 DeviceFeature_GetDefaultLogMode

Syntax

```
DeviceFeature_GetDefaultLogMode (UINT* pnDefault)
```

Description

DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N, DCC3260M, DCC3260C:
Returns the default settings for the Log mode.

Parameter

- `pnDefault`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480DeviceFeature](#)

Related functions

- [DeviceFeature_GetLogMode](#)
- [DeviceFeature_GetLogModeManualGain](#)
- [DeviceFeature_GetLogModeManualGainDefault](#)
- [DeviceFeature_GetLogModeManualGainRange](#)
- [DeviceFeature_GetLogModeManualValue](#)
- [DeviceFeature_GetLogModeManualValueDefault](#)
- [DeviceFeature_GetLogModeManualValueRange](#)
- [DeviceFeature_SetLogMode](#)
- [DeviceFeature_SetLogModeManualGain](#)
- [DeviceFeature_SetLogModeManualValue](#)

3.2.16.5 DeviceFeature_GetLogMode

Syntax

`DeviceFeature_GetLogMode (UINT* pnMode)`

Description

DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N, DCC3260M, DCC3260C:
Returns the current Log mode (see [DeviceFeature_SetLogMode](#)).

Parameter

- `pnMode`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480DeviceFeature](#)

Related functions

- [DeviceFeature_GetDefaultLogMode](#)
- [DeviceFeature_GetLogModeManualGain](#)
- [DeviceFeature_GetLogModeManualGainDefault](#)
- [DeviceFeature_GetLogModeManualGainRange](#)
- [DeviceFeature_GetLogModeManualValue](#)
- [DeviceFeature_GetLogModeManualValueDefault](#)
- [DeviceFeature_GetLogModeManualValueRange](#)
- [DeviceFeature_SetLogMode](#)
- [DeviceFeature_SetLogModeManualGain](#)
- [DeviceFeature_SetLogModeManualValue](#)

3.2.16.6 DeviceFeature_SetLogMode

Syntax

```
DeviceFeature_SetLogMode (UINT nMode)
```

Description

DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N, DCC3260M, DCC3260C: Sets the Log mode.

- `IS_LOG_MODE_FACTORY_DEFAULT`: Default setting for the Log mode
- `IS_LOG_MODE_OFF`: Log mode off
- `IS_LOG_MODE_MANUAL`: Manual Log mode. In this case the LogMode value and the LogMode gain are effective.

Parameter

- `nMode`: value to be set

Interface

- [Iuc480DeviceFeature](#)

Related functions

- [DeviceFeature_GetDefaultLogMode](#)
- [DeviceFeature_GetLogMode](#)
- [DeviceFeature_GetLogModeManualGain](#)
- [DeviceFeature_GetLogModeManualGainDefault](#)
- [DeviceFeature_GetLogModeManualGainRange](#)
- [DeviceFeature_GetLogModeManualValue](#)
- [DeviceFeature_GetLogModeManualValueDefault](#)
- [DeviceFeature_GetLogModeManualValueRange](#)
- [DeviceFeature_SetLogModeManualGain](#)
- [DeviceFeature_SetLogModeManualValue](#)

3.2.16.7 DeviceFeature_GetLogModeManualGain

Syntax

```
DeviceFeature_GetLogModeManualGain (UINT* pnGain)
```

Description

DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N, DCC3260M, DCC3260C:: Returns the current manual gain of the Log mode.

Parameter

- `pnGain`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480DeviceFeature](#)

Related functions

- [DeviceFeature_GetDefaultLogMode](#)
- [DeviceFeature_GetLogMode](#)
- [DeviceFeature_GetLogModeManualGainDefault](#)
- [DeviceFeature_GetLogModeManualGainRange](#)
- [DeviceFeature_GetLogModeManualValue](#)
- [DeviceFeature_GetLogModeManualValueDefault](#)
- [DeviceFeature_GetLogModeManualValueRange](#)
- [DeviceFeature_SetLogMode](#)
- [DeviceFeature_SetLogModeManualGain](#)
- [DeviceFeature_SetLogModeManualValue](#)

3.2.16.8 DeviceFeature_GetLogModeManualGainDefault

Syntax

DeviceFeature_GetLogModeManualGainDefault (UINT* pnDefault)

Description

DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N, DCC3260M, DCC3260C:: Returns the default settings for the manual gain for the Log mode.

Parameter

- pnDefault: Pointer to the variable in which the return value is written.

Interface

- [Iuc480DeviceFeature](#)

Related functions

- [DeviceFeature_GetDefaultLogMode](#)
- [DeviceFeature_GetLogMode](#)
- [DeviceFeature_GetLogModeManualGain](#)
- [DeviceFeature_GetLogModeManualGainRange](#)
- [DeviceFeature_GetLogModeManualValue](#)
- [DeviceFeature_GetLogModeManualValueDefault](#)
- [DeviceFeature_GetLogModeManualValueRange](#)
- [DeviceFeature_SetLogMode](#)
- [DeviceFeature_SetLogModeManualGain](#)
- [DeviceFeature_SetLogModeManualValue](#)

3.2.16.9 DeviceFeature_GetLogModeManualGainRange

Syntax

DeviceFeature_GetLogModeManualGainRange (INT* pnMin, INT* pnMax, INT* pnInc)

Description

DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N, DCC3260M, DCC3260C:

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

Parameter

pnMin	Pointer to the variable in which the minimum value is written.
pnMax	Pointer to the variable in which the maximum value is written.
pnInc	Pointer to the variable in which the increment is written.

Interface

- [Iuc480DeviceFeature](#)

Related functions

- [DeviceFeature_GetDefaultLogMode](#)
- [DeviceFeature_GetLogMode](#)
- [DeviceFeature_GetLogModeManualGain](#)
- [DeviceFeature_GetLogModeManualGainDefault](#)
- [DeviceFeature_GetLogModeManualValue](#)
- [DeviceFeature_GetLogModeManualValueDefault](#)
- [DeviceFeature_GetLogModeManualValueRange](#)
- [DeviceFeature_SetLogMode](#)
- [DeviceFeature_SetLogModeManualGain](#)
- [DeviceFeature_SetLogModeManualValue](#)

3.2.16.10 DeviceFeature_SetLogModeManualGain

Syntax

DeviceFeature_SetLogModeManualGain (UINT nGain)

Description

DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N, DCC3260M, DCC3260C: Sets the manual gain of the Log mode.

Parameter

- nGain: value to be set

Interface

- [Iuc480DeviceFeature](#)

Related functions

- [DeviceFeature_GetDefaultLogMode](#)
- [DeviceFeature_GetLogMode](#)
- [DeviceFeature_GetLogModeManualGain](#)
- [DeviceFeature_GetLogModeManualGainDefault](#)
- [DeviceFeature_GetLogModeManualGainRange](#)
- [DeviceFeature_GetLogModeManualValue](#)
- [DeviceFeature_GetLogModeManualValueDefault](#)

- [DeviceFeature_GetLogModeManualValueRange](#)
- [DeviceFeature_SetLogMode](#)
- [DeviceFeature_SetLogModeManualValue](#)

3.2.16.11 DeviceFeature_GetLogModeManualValue

Syntax

DeviceFeature_GetLogModeManualValue (UINT* pnValue)

Description

DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N, DCC3260M, DCC3260C:
Returns the current manual value of the Log mode.

Parameter

- `pnValue`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480DeviceFeature](#)

Related functions

- [DeviceFeature_GetDefaultLogMode](#)
- [DeviceFeature_GetLogMode](#)
- [DeviceFeature_GetLogModeManualGain](#)
- [DeviceFeature_GetLogModeManualGainDefault](#)
- [DeviceFeature_GetLogModeManualGainRange](#)
- [DeviceFeature_GetLogModeManualValueDefault](#)
- [DeviceFeature_GetLogModeManualValueRange](#)
- [DeviceFeature_SetLogMode](#)
- [DeviceFeature_SetLogModeManualGain](#)
- [DeviceFeature_SetLogModeManualValue](#)

3.2.16.12 DeviceFeature_GetLogModeManualValueDefault

Syntax

DeviceFeature_GetLogModeManualValueDefault (UINT* pnDefault)

Description

DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N, DCC3260M, DCC3260C:
Returns the default settings for the manual value of the Log mode.

Parameter

- `pnDefault`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480DeviceFeature](#)

Related functions

- [DeviceFeature_GetDefaultLogMode](#)

- [DeviceFeature_GetLogMode](#)
- [DeviceFeature_GetLogModeManualGain](#)
- [DeviceFeature_GetLogModeManualGainDefault](#)
- [DeviceFeature_GetLogModeManualGainRange](#)
- [DeviceFeature_GetLogModeManualValue](#)
- [DeviceFeature_GetLogModeManualValueRange](#)
- [DeviceFeature_SetLogMode](#)
- [DeviceFeature_SetLogModeManualGain](#)
- [DeviceFeature_SetLogModeManualValue](#)

3.2.16.13 DeviceFeature_GetLogModeManualValueRange

Syntax

```
DeviceFeature_GetLogModeManualValueRange (INT* pnMin, INT* pnMax, INT* pnInc)
```

Description

DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N, DCC3260M, DCC3260C:
Returns the range of the manual value of the Log mode.

Parameter

pnMin	Pointer to the variable in which the minimum value is written.
pnMax	Pointer to the variable in which the maximum value is written.
pnInc	Pointer to the variable in which the increment is written.

Interface

- [Iuc480DeviceFeature](#)

Related functions

- [DeviceFeature_GetDefaultLogMode](#)
- [DeviceFeature_GetLogMode](#)
- [DeviceFeature_GetLogModeManualGain](#)
- [DeviceFeature_GetLogModeManualGainDefault](#)
- [DeviceFeature_GetLogModeManualGainRange](#)
- [DeviceFeature_GetLogModeManualValue](#)
- [DeviceFeature_GetLogModeManualValueDefault](#)
- [DeviceFeature_SetLogMode](#)
- [DeviceFeature_SetLogModeManualGain](#)
- [DeviceFeature_SetLogModeManualValue](#)

3.2.16.14 DeviceFeature_SetLogModeManualValue

Syntax

```
DeviceFeature_SetLogModeManualValue (UINT nValue)
```

Description

DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N, DCC3260M, DCC3260C:Sets

the manual value of the Log mode.

Parameter

- `nValue`: value to be set

Interface

- [Iuc480DeviceFeature](#)

Related functions

- [DeviceFeature_GetDefaultLogMode](#)
- [DeviceFeature_GetLogMode](#)
- [DeviceFeature_GetLogModeManualGain](#)
- [DeviceFeature_GetLogModeManualGainDefault](#)
- [DeviceFeature_GetLogModeManualGainRange](#)
- [DeviceFeature_GetLogModeManualValue](#)
- [DeviceFeature_GetLogModeManualValueDefault](#)
- [DeviceFeature_GetLogModeManualValueRange](#)
- [DeviceFeature_SetLogMode](#)
- [DeviceFeature_SetLogModeManualGain](#)

3.2.16.15 DeviceFeature_GetLineScanMode

Syntax

```
DeviceFeature_GetLineScanMode (INT* pnMode)
```

Description

DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N: Returns the current set line scan mode.

Parameter

- `pnMode`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480DeviceFeature](#)

Related functions

- [DeviceFeature_GetLineScanNumber](#)
- [DeviceFeature_SetLineScanMode](#)
- [DeviceFeature_SetLineScanNumber](#)

3.2.16.16 DeviceFeature_SetLineScanMode

Syntax

```
DeviceFeature_SetLineScanMode (INT nMode)
```

Description

DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N: Sets the line scan mode:

- `IS_DEVICE_FEATURE_CAP_LINESCAN_MODE_FAST`: Fast line scan mode is supported/Set mode

- `IS_DEVICE_FEATURE_CAP_LINESCAN_NUMBER`: Line number at fast line scan mode is supported/
Set number

Parameter

- `nMode`: value to be set

Interface

- [Iuc480DeviceFeature](#)

Related functions

- [DeviceFeature_GetLineScanMode](#)
- [DeviceFeature_GetLineScanNumber](#)
- [DeviceFeature_SetLineScanNumber](#)

3.2.16.17 DeviceFeature_GetLineScanNumber

Syntax

`DeviceFeature_GetLineScanNumber (INT* pnNumber)`

Description

DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N: Returns the scan line used for the line scan mode.

Parameter

- `pnNumber`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480DeviceFeature](#)

Related functions

- [DeviceFeature_GetLineScanNumber](#)
- [DeviceFeature_SetLineScanMode](#)
- [DeviceFeature_SetLineScanNumber](#)

3.2.16.18 DeviceFeature_SetLineScanNumber

Syntax

`DeviceFeature_SetLineScanNumber (INT nNumber)`

Description

DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N: Sets the scan line used for the line scan mode.

Parameter

- `nNumber`: value to be set

Interface

- [Iuc480DeviceFeature](#)

Related functions

- [DeviceFeature_GetLineScanMode](#)
- [DeviceFeature_GetLineScanNumber](#)
- [DeviceFeature_SetLineScanMode](#)

3.2.16.19 DeviceFeature_GetShutterMode

Syntax

DeviceFeature_GetShutterMode (INT* pnMode)

Description

DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N: Returns the shutter mode.

Parameter

- pnMode: Pointer to the variable in which the return value is written.

Interface

- [Iuc480DeviceFeature](#)

Related functions

- [DeviceFeature_SetShutterMode](#)

3.2.16.20 DeviceFeature_SetShutterMode

Syntax

DeviceFeature_SetShutterMode (INT nMode)

Description

DCC1240C DCC1240M, DCC3240C, DCC3240M, DCC3240N: Sets the shutter mode:

- IS_DEVICE_FEATURE_CAP_SHUTTER_MODE_ROLLING: Rolling shutter mode is supported/Set mode
- IS_DEVICE_FEATURE_CAP_SHUTTER_MODE_ROLLING_GLOBAL_START: Rolling shutter mode with global start is supported/Set mode
- IS_DEVICE_FEATURE_CAP_SHUTTER_MODE_GLOBAL: Global shutter mode is supported/Set mode
- IS_DEVICE_FEATURE_CAP_SHUTTER_MODE_GLOBAL_ALTERNATIVE_TIMING: Global shutter mode with different timing parameters is supported/Set mode

Parameter

- nMode: value to be set

Interface

- [Iuc480DeviceFeature](#)

Related functions

- [DeviceFeature_GetShutterMode](#)

3.2.16.21 DeviceFeature_GetVerticalAOIMergeMode

Syntax

```
DeviceFeature_GetVerticalAOIMergeMode (INT* pnMode)
```

Description

DCC3240C, DCC3240M, DCC3240N, DCC3260M, DCC3260C: Returns the current set AOI merge mode.

Parameter

- `pnMode`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480DeviceFeature](#)

Related functions

- [DeviceFeature_GetVerticalAOIMergePosition](#)
- [DeviceFeature_SetVerticalAOIMergeMode](#)
- [DeviceFeature_SetVerticalAOIMergePosition](#)

3.2.16.22 DeviceFeature_SetVerticalAOIMergeMode

Syntax

```
DeviceFeature_SetVerticalAOIMergeMode (INT nMode)
```

Description

DCC3240C, DCC3240M, DCC3240N, DCC3260M, DCC3260C: Sets the AOI merge mode which combines the lines of an AOI to a new image.

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.

- `IS_VERTICAL_AOI_MERGE_MODE_OFF`: Disables the AOI merge mode
- `IS_VERTICAL_AOI_MERGE_MODE_ON`: Enables the AOI merge mode
- `IS_VERTICAL_AOI_MERGE_MODE_FREERUN`: The sensor runs with maximum speed.
- `IS_VERTICAL_AOI_MERGE_MODE_TRIGGERED_SOFTWARE`: The sensor is triggered via software.
- `IS_VERTICAL_AOI_MERGE_MODE_TRIGGERED_FALLING_GPIO1`: The sensor is triggered on GPIO 1 (falling edge).
- `IS_VERTICAL_AOI_MERGE_MODE_TRIGGERED_RISING_GPIO1`: The sensor is triggered on GPIO 1 (rising edge).
- `IS_VERTICAL_AOI_MERGE_MODE_TRIGGERED_FALLING_GPIO2`: The sensor is triggered on GPIO 2 (falling edge).
- `IS_VERTICAL_AOI_MERGE_MODE_TRIGGERED_RISING_GPIO2`: The sensor is triggered on GPIO 2 (rising edge).

Parameter

- `nMode`: value to be set

Interface

- [Iuc480DeviceFeature](#)

Related functions

- [DeviceFeature_GetVerticalAOIMergeMode](#)
- [DeviceFeature_GetVerticalAOIMergePosition](#)
- [DeviceFeature_SetVerticalAOIMergePosition](#)

3.2.16.23 DeviceFeature_GetVerticalAOIMergePosition

Syntax

DeviceFeature_GetVerticalAOIMergePosition (INT* pnPosition)

Description

DCC3240C, DCC3240M, DCC3240N, DCC3260M, DCC3260C: Returns the position of the two sensor lines (default = 0, i.e. the two top lines).

Parameter

- pnPosition: Pointer to the variable in which the return value is written.

Interface

- [Iuc480DeviceFeature](#)

Related functions

- [DeviceFeature_GetVerticalAOIMergeMode](#)
- [DeviceFeature_SetVerticalAOIMergeMode](#)
- [DeviceFeature_SetVerticalAOIMergePosition](#)

3.2.16.24 DeviceFeature_SetVerticalAOIMergePosition

Syntax

DeviceFeature_SetVerticalAOIMergePosition (INT nPosition)

Description

DCC3240C, DCC3240M, DCC3240N, DCC3260M, DCC3260C: Sets the position of the two sensor lines.

Parameter

- nPosition: value to be set

Interface

- [Iuc480DeviceFeature](#)

Related functions

- [DeviceFeature_GetVerticalAOIMergeMode](#)
- [DeviceFeature_GetVerticalAOIMergePosition](#)
- [DeviceFeature_SetVerticalAOIMergeMode](#)

3.2.17 Iuc480EdgeEnhancement

IID_Iuc480EdgeEnhancement

This interface provides special functions for using the software edge filter that are not covered by the DirectShow standard. To use the interface, you must integrate the `uc480CaptureInterface.h` header file into your project.

`Iuc480EdgeEnhancement` provides the following functions:

EdgeEnhancement_GetEdgeEnhancement	Returns the current set edge enhancement.
EdgeEnhancement_GetEdgeEnhancementDefault	Returns the standard value of the edge enhancement.
EdgeEnhancement_GetEdgeEnhancementRange	Returns the range of the edge enhancement.
EdgeEnhancement_SetEdgeEnhancement	Sets the edge enhancement (0: no edge enhancement).

3.2.17.1 EdgeEnhancement_GetEdgeEnhancement

Syntax

```
GetEdgeEnhancement (UINT* pnEdgeEnhancement)
```

Description

Returns the current set edge enhancement.

Parameter

- `pnEdgeEnhancement`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480EdgeEnhancement](#)

Related functions

- [EdgeEnhancement_GetEdgeEnhancementDefault](#)
- [EdgeEnhancement_GetEdgeEnhancementRange](#)
- [EdgeEnhancement_SetEdgeEnhancement](#)

3.2.17.2 EdgeEnhancement_GetEdgeEnhancementDefault

Syntax

```
GetEdgeEnhancementDefault (UINT* pnDefault)
```

Description

Returns the standard value of the edge enhancement.

Parameter

- `pnDefault`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480EdgeEnhancement](#)

Related functions

- [EdgeEnhancement_GetEdgeEnhancement](#)
- [EdgeEnhancement_GetEdgeEnhancementRange](#)
- [EdgeEnhancement_SetEdgeEnhancement](#)

3.2.17.3 EdgeEnhancement_GetEdgeEnhancementRange

Syntax

```
GetEdgeEnhancementRange (UINT* pnMin, UINT* pnMax, UINT* pnInc)
```

Description

Returns the range of the edge enhancement.

Parameter

pnMin	Pointer to the variable in which the minimum value is written.
pnMax	Pointer to the variable in which the maximum value is written.
pnInc	Pointer to the variable in which the increment is written.

Interface

- [Iuc480EdgeEnhancement](#)

Related functions

- [EdgeEnhancement_GetEdgeEnhancement](#)
- [EdgeEnhancement_GetEdgeEnhancementDefault](#)
- [EdgeEnhancement_SetEdgeEnhancement](#)

3.2.17.4 EdgeEnhancement_SetEdgeEnhancement

Syntax

```
SetEdgeEnhancement (UINT nEdgeEnhancement)
```

Description

Sets the edge enhancement (0: no edge enhancement).

Parameter

- `nEdgeEnhancement` : value to be set

Interface

- [Iuc480EdgeEnhancement](#)

Related functions

- [EdgeEnhancement_GetEdgeEnhancement](#)
- [EdgeEnhancement_GetEdgeEnhancementDefault](#)
- [EdgeEnhancement_GetEdgeEnhancementRange](#)

3.2.18 Iuc480Event

IID_Iuc480Event

This interface provides special functions for the event handling that are not covered by the DirectShow standard. To use the interface, you must integrate the `uc480CaptureInterface.h` header file into your project.

`Iuc480Event` provides the following functions:

InitEvent	Initializes the event handle
EnableEvent	Activates the event handle
DisableEvent	Deactivates the event handle
ExitEvent	Deletes the event handle
EnableMessage	Activates/Deactivates Windows messages

3.2.18.1 InitEvent

Syntax

```
InitEvent (HANDLE hEv, INT nWhich)
```

Description

Initializes the event handle for the specified event object. This registers the event object in the uc480 kernel driver.

Note on using USB cameras under Windows

The following events require a Windows message loop. This message loop has to be executed by the thread that loads the uc480 API. The message loop is usually provided by the application window. In some cases, the message loop might not be created automatically (e.g. in console applications). In this case you will need to implement the message loop yourself.

This applies to the following uc480 events:

- `IS_SET_EVENT_REMOVE`
- `IS_SET_EVENT_REMOVAL`
- `IS_SET_EVENT_DEVICE_RECONNECTED`
- `IS_SET_EVENT_NEW_DEVICE`

If no message loop exists, a USB camera will not be automatically detected after reconnecting.



Parameter

<code>hEv</code>	Event handle created by the <code>CreateEvent()</code> Windows API function.
<code>nWhich</code>	ID of the event to be initialized (see EnableEvent)

Interface

- [Iuc480Event](#)

Related functions

- [EnableEvent](#)

- [EnableMessage](#)

3.2.18.2 EnableEvent

Syntax

EnableEvent (INT nWhich)

Description

Activates an event object. After the release the event messages for the created event object are enabled. Depending on the operating system different functions are to call.

- Event has to be provided by the application program
- Event has to be declared by [InitEvent](#)
- Event has to be activated by `EnableEvent`
- You have to wait for the event in the application program by `WaitForSingleObject` or `WaitForMultipleObject`
- Event has to be deactivated by [DisableEvent](#)
- Event has to be deleted by [ExitEvent](#)

Parameter

nWhich	<ul style="list-style-type: none"> • <code>IS_SET_EVENT_AUTOBRIGHTNESS_FINISHED</code> The automatic brightness control in the run-once mode is completed. • <code>IS_SET_EVENT_AUTOFOCUS_FINISHED</code> Automatic focus control is finished (XS only). • <code>IS_SET_EVENT_CAMERA_MEMORY</code> In the camera memory mode an image acquisition iteration is finished. • <code>IS_SET_EVENT_CAPTURE_STATUS</code> There is a information about image capturing available. • <code>IS_SET_EVENT_CONNECTIONSPEED_CHANGED</code> 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. • <code>IS_SET_EVENT_DEVICE_RECONNECTED</code> An initialized and disconnected afterwards camera was reconnected. • <code>IS_SET_EVENT_EXTTRIG</code> An image which was captured following the arrival of a trigger has been transferred completely. This is the earliest possible moment for a new capturing process. The image must then be post-processed by the driver and will be available after the <code>IS_FRAME</code> processing event. • <code>IS_SET_EVENT_FIRST_PACKET_RECEIVED</code> 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. • <code>IS_SET_EVENT_FRAME</code> A new image is available. • <code>IS_SET_EVENT_NEW_DEVICE</code> A new camera was connected. • <code>IS_SET_EVENT_OVERLAY_DATA_LOST</code> Direct3D mode: Because of a re-programming the parameters of the overlay are invalid. The overlay must be draw new. • <code>IS_SET_EVENT_REMOVAL</code> A camera was removed. • <code>IS_SET_EVENT_REMOVE</code> An initialized camera was disconnected. • <code>IS_SET_EVENT_SEQ</code> The sequence is completed. • <code>IS_SET_EVENT_STEAL</code> An image extracted from the overlay is available. • <code>IS_SET_EVENT_WB_FINISHED</code> The automatic white balance control is completed.
--------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Interface

- [InitEvent](#)

Related functions

- [InitEvent](#)
- [DisableEvent](#)

3.2.18.3 DisableEvent

Syntax

```
DisableEvent (INT nWhich)
```

Description

Disables the event indicated here. The event (e.g. image capture completed) will usually still occur, but will no longer trigger an event signal. Disabled events are no longer signaled to the application. You can re-enable the desired event using [EnableEvent](#). See also [InitEvent](#).

Parameter

- **nWhich**: ID of the event to be disabled. See [EnableEvent](#).

Interface

- [InitEvent](#)

Related functions

- [EnableEvent](#)

3.2.18.4 ExitEvent

Syntax

```
ExitEvent (INT nWhich)
```

Description

Deletes an existing event object. After an event has been deleted, you can no longer enable it by calling [EnableEvent](#).

Parameter

- **nWhich**: ID of the event to be deleted. See [EnableEvent](#).

Interface

- [InitEvent](#)

Related functions

- [InitEvent](#)

3.2.18.5 EnableMessage

Syntax

```
EnableMessage (INT which, HWND hWnd)
```

Description

Activates/deactivates Windows Messages. If a particular event occurs, the messages are sent to the application. Each message is structured as follows:

- **Message**: `IS_uc480_MESSAGE`
- **wParam**: event (see table)
- **lParam**: uc480 camera handle associated with the message



You have to deactivate Windows messages with `hWnd == NULL` before you free the uc480 API library. Otherwise the application may not close properly.

Parameter

which: ID of the message to be enabled/disabled	
IS_AUTOBRIGHTNESS_FINISHED	Automatic brightness control is completed (only if this control was started using the <code>IS_SET_AUTO_BRIGHTNESS_ONCE</code> function).
IS_AUTOFOCUS_FINISHED	Automatic focus control is finished (XS only)
IS_CAMERA_MEMORY	In the camera memory mode an image acquisition iteration is finished.
IS_CAPTURE_STATUS	An error occurred during the data transfer.
IS_CONNECTIONSPEED_CHANGED	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.
IS_FIRST_PACKET_RECEIVED	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.
IS_FRAME	A new image is available.
IS_DEVICE_REMOVAL	A camera was removed.
IS_DEVICE_REMOVED	An opened camera was disconnected.
IS_DEVICE_RECONNECTED	An opened and disconnected camera afterwards was reconnected.
IS_NEW_DEVICE	A new camera was connected.
IS_OVERLAY_DATA_LOST	Direct3D/OpenGL mode: Because of a re-programming the parameters of the overlay are invalid. The overlay must be draw new.
IS_SEQUENCE	The sequence is completed.
IS_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 is available after the <code>IS_FRAME</code> message has occurred.
IS_WB_FINISHED	Automatic white balance control is completed (only if this control was started using the <code>IS_SET_AUTO_WB_ONCE</code> function).
hWnd	Application window for receiving the message. <code>NULL</code> disables the message designated by the <code>which</code> parameter.

Interface

- [InitEvent](#)

Related functions

- [InitEvent](#)

3.2.19 Iuc480Flash

IID_Iuc480Flash

This interface provides special functions for managing the camera inputs/outputs that are not covered by the DirectShow standard. The digital outputs can be used in either freerun or trigger mode. The output level can be synchronized with the exposure time or statically set. To use the interface, you must integrate the `uc480CaptureInterface.h` header file into your project.

Iuc480Flash provides the following functions:

Flash_SetStrobeMode	Sets the flash mode.
Flash_GetStrobeMode	Returns the set flash mode.
Flash_GetDuration	Returns the set flash duration.
Flash_GetDurationRange	Returns the range for the flash duration.
Flash_SetDelayDuration	Sets the flash delay and flash duration.
Flash_GetDelay	Returns the set flash delay.
Flash_GetDelayRange	Returns the range for the flash delay.
Flash_GetGlobalExposureWindow	Returns the time window for a global flash during which all sensor rows are simultaneously exposed.
Flash_GetSupportedGPIOPorts	Returns the programmable inputs/outputs (GPIO) that can be used for the flash function.
Flash_EnableGPIOPort	Enables a programmable input/output (GPIO) for the flash function.

3.2.19.1 Flash_SetStrobeMode

Syntax

```
Flash_SetStrobeMode (long lMode)
```

Description

Sets the flash mode.

Parameter

lMode	Mode to be set
Flash with exposure time synchronization	
IS_SET_FLASH_OFF	Disables digital output.
IS_SET_FLASH_LO_ACTIVE	Enables the flash in trigger mode. LO_ACTIVE: The digital output is switched to low for the flash duration.
IS_SET_FLASH_HI_ACTIVE	Enables the flash in trigger mode. HI_ACTIVE: The digital output is switched to high for the flash duration.
IS_SET_FLASH_HIGH	Statically sets the digital output to high level (HIGH).
IS_SET_FLASH_LOW	Statically sets the digital output to low level (LOW).
IS_SET_FLASH_LO_ACTIVE_FREERUN	Enables the flash in freerun mode. LO_ACTIVE: The digital output is switched to low for the flash duration.
IS_SET_FLASH_HI_ACTIVE_FREERUN	Enables the flash in freerun mode. HI_ACTIVE: The digital output is switched to high for the flash duration.
Statically sets the output level	
IS_SET_FLASH_HIGH	Statically sets the digital output to high (HIGH).
IS_SET_FLASH_LOW	Statically sets the digital output to low (LOW).

Interface

- [Iuc480Flash](#)

Related functions

- [Flash_GetStrobeMode](#)

3.2.19.2 Flash_GetStrobeMode

Syntax

```
Flash_GetStrobeMode (long* plMode)
```

Description

Returns the set flash mode.

Parameter

- plMode: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Flash](#)

Related functions

- [Flash_SetStrobeMode](#)

3.2.19.3 Flash_GetDuration

Syntax

```
Flash_GetDuration (unsigned long* pulDuration)
```

Description

Returns the set flash duration.

Parameter

- `pulDuration`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Flash](#)

Related functions

- [Flash_GetDurationRange](#)
- [Flash_SetDelayDuration](#)

3.2.19.4 Flash_GetDurationRange

Syntax

```
Flash_GetDurationRange (unsigned long* pulMin, unsigned long* pulMax, unsigned long* pulInc)
```

Description

Returns the range (minimum, maximum and increment) for the flash duration.

Parameter

<code>pulMin</code>	Pointer to the variable in which the minimum value is written.
<code>pulMax</code>	Pointer to the variable in which the maximum value is written.
<code>pulInc</code>	Pointer to the variable in which the increment is written.

Interface

- [Iuc480Flash](#)

Related functions

- [Flash_GetDuration](#)

3.2.19.5 Flash_SetDelayDuration

Syntax

```
Flash_SetDelayDuration (unsigned long ulDelay, unsigned long ulDuration)
```

Description

Sets the flash delay and flash duration.

Parameter

<code>ulDelay</code>	Value to be set for the flash delay.
<code>ulDuration</code>	Value to be set for the flash duration.

Interface

- [Iuc480Flash](#)

Related functions

- [Flash_GetDelay](#)
- [Flash_GetDuration](#)

3.2.19.6 Flash_GetDelay

Syntax

```
Flash_GetDelay (unsigned long* pulDelay)
```

Description

Returns the set flash delay.

Parameter

- `pulDelay`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Flash](#)

Related functions

- [Flash_GetDelayRange](#)

3.2.19.7 Flash_GetDelayRange

Syntax

```
Flash_GetDelayRange (unsigned long* pulMin, unsigned long* pulMax, unsigned long* pulInc)
```

Description

Returns the range (minimum, maximum and increment) for the flash delay.

Parameter

<code>pulMin</code>	Pointer to the variable in which the minimum value is written.
<code>pulMax</code>	Pointer to the variable in which the maximum value is written.
<code>pulInc</code>	Pointer to the variable in which the increment is written.

Interface

- [Iuc480Flash](#)

Related functions

- [Flash_GetDelay](#)

3.2.19.8 Flash_GetGlobalExposureWindow

Syntax

```
Flash_GetGlobalExposureWindow (unsigned long* pulDelay, unsigned long* pulDuration)
```

Description

Returns the time window for a global flash during which all sensor rows are simultaneously exposed.

Parameter

pulDelay	Pointer to the variable in which the delay for the flash window is written.
pulDuration	Pointer to the variable in which the duration of the flash window is written.

Interface

- [Iuc480Flash](#)

3.2.19.9 Flash_GetSupportedGPIOPorts

Syntax

```
Flash_GetSupportedGPIOPorts (unsigned long* pulPorts)
```

Description

Returns the programmable inputs/outputs (GPIO) that can be used for the flash function.

Parameter

- pulPorts: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Flash](#)

Related functions

- [Flash_EnableGPIOPort](#)

3.2.19.10 Flash_EnableGPIOPort

Syntax

```
Flash_EnableGPIOPort (unsigned long ulPort)
```

Description

Enables a programmable input/output (GPIO) for the flash function.

Parameter

- ulPort: GPIO to be used for the flash function

Interface

- [Iuc480Flash](#)

Related functions

- [Flash_GetSupportedGPIOPorts](#)

3.2.20 Iuc480Gain

IID_Iuc480Gain

This interface provides special features for setting the analog sensor gain for uc480 models that are not covered by the standard DirectShow interface. The various gain channels for the sensor can each be set independently from 0 % to 100 %. The gain factor actually achieved at a setting of 100 % will depend on each sensor and is detailed in the section "Camera and Sensor Data" of the uc480 Handbook.



Note on usage of sensor gain

Amplifying signals will cause an increased level of image noise. For this reason it is not recommended to use a very high gain setting.

It is recommended to use the following sequence when making your gain settings:

1. Enable the `Gain_SetGainBoostValue` feature (for additional analog gain)
2. Readjust where necessary using `Gain_SetHwGainFactor`

Depending on when you have adjusted the gain, the change will be applied only to the next image you take.



Note on the linearity of sensor gain

The gain for uc480 cameras can be set from 0 to 100. Based on the sensor the increments do not increase in a linear manner over the entire range. The increment will generally be larger in the upper part of the range than lower down.

To use the interface, you must integrate the `uc480CaptureInterface.h` header file into your project.

`Iuc480Gain` provides the following features:

Gain_IsMasterSupported	Returns whether the camera provides total gain master gain.
Gain_IsRGBSupported	Returns whether the camera provides RGB color gain.
Gain_GetHwGain	Returns the gain values set [0,100].
Gain_SetHwGain	Sets the gain values [0,100].
Gain_GetHwGainDefaults	Returns the default values for the gain.
Gain_GetHwGainRange	Returns the value range for the gain.
Gain_GetHwGainFactor	Returns the set gain factors as percentages (e.g.: 200 corresponds to a factor of 2.0).
Gain_SetHwGainFactor	Sets the gain factors as percentages.
Gain_GetHwGainFactorDefaults	Returns the default values for the gain factors.
Gain_InquireHwGainFactor	Converts a normalized value [0,100] for the gain into a gain factor.
Gain_GetHwGainFactorRange	Returns the value range for the gain factors.
Gain_IsGainBoostSupported	Returns whether the camera provides the analog Gain Boost feature.
Gain_GetGainBoostValue	Returns whether the Gain Boost feature is enabled [0,1].
Gain_SetGainBoostValue	Sets the value of the Gain Boost [0,1].

3.2.20.1 Gain_IsMasterSupported

Syntax

```
Gain_IsMasterSupported (bool* pbSupported)
```

Description

Returns whether the camera provides total gain (Master Gain).

Parameter

- `pbSupported`: Pointer to the variable to which the return value is written.

Interface

- [Iuc480Gain](#)

Related functions

- [Gain_IsRGBSupported](#)

3.2.20.2 Gain_IsRGBSupported

Syntax

```
Gain_IsRGBSupported (bool* pbSupported)
```

Description

Returns whether the camera provides RGB color gain.

Parameter

- `pbSupported`: Pointer to the variable to which the return value is written.

Interface

- [Iuc480Gain](#)

Related functions

- [Gain_IsMasterSupported](#)

3.2.20.3 Gain_GetHwGain**Syntax**

```
Gain_GetHwGain (INT nWhich, INT *pnValue)
```

Description

Returns the gain values set [0,100].

Parameter

<code>nWhich</code>	Sets the gain channel for which the value is queried: <ul style="list-style-type: none"> • 0 = total gain • 1 = red gain • 2 = green gain • 3 = blue gain
<code>pnValue</code>	Pointer to the variable to which the return value is written.

Interface

- [Iuc480Gain](#)

Related functions

- [Gain_SetHwGain](#)

3.2.20.4 Gain_SetHwGain**Syntax**

```
Gain_SetHwGain (INT nWhich, INT nValue)
```

Description

Sets the gain values [0,100].

Parameter

<code>nWhich</code>	Specifies the gain channel for which the value is to be set: <ul style="list-style-type: none"> • 0 = total gain • 1 = red gain • 2 = green gain • 3 = blue gain
<code>nValue</code>	Value to be set

Interface

- [Iuc480Gain](#)

Related functions

- [Gain_GetHwGain](#)

3.2.20.5 Gain_GetHwGainDefaults

Syntax

```
Gain_GetHwGainDefaults)(INT *pnMaster, INT *pnRed, INT *pnGreen, INT *pnBlue)
```

Description

Returns the default values for the gain.

Parameter

pnMaster	Pointer to the variable to which the default value for the total gain is written.
pnRed	Pointer to the variable to which the default value for the red gain is written.
pnGreen	Pointer to the variable to which the default value for the green gain is written.
pnBlue	Pointer to the variable to which the default value for the blue gain is written.

Interface

- [Iuc480Gain](#)

Related functions

- [Gain_GetHwGainRange](#)

3.2.20.6 Gain_GetHwGainRange

Syntax

```
Gain_GetHwGainRange (INT nWhich, INT *pnMin, INT *pnMax)
```

Description

Returns the value range for the gain.

Parameter

nWhich	Specifies the gain channel for which values are queried: <ul style="list-style-type: none">• 0 = total gain• 1 = red gain• 2 = green gain• 3 = blue gain
pnMin	Pointer to the variable to which the minimum value is written.
pnMax	Pointer to the variable to which the maximum value is written.

Interface

- [Iuc480Gain](#)

Related functions

- [Gain_GetHwGainDefaults](#)

3.2.20.7 Gain_GetHwGainFactor

Syntax

```
Gain_GetHwGainFactor (INT nWhich, INT* pnFactor)
```

Description

Returns the set gain factors as percentages (e.g.: 200 corresponds to a factor of 2.0).

Parameter

nWhich	Specifies the gain channel for which the value is queried: <ul style="list-style-type: none"> • 0 = total gain • 1 = red gain • 2 = green gain • 3 = blue gain
pnFactor	Pointer to the variable to which the return value is written.

Interface

- [Iuc480Gain](#)

Related functions

- [Gain_SetHwGainFactor](#)

3.2.20.8 Gain_SetHwGainFactor

Syntax

```
Gain_SetHwGainFactor (INT nWhich, INT nFactor)
```

Description

Sets the gain factors as percentages.

Parameter

nWhich	Specifies the gain channel for which the value is to be set: <ul style="list-style-type: none"> • 0 = total gain • 1 = red gain • 2 = green gain • 3 = blue gain
nFactor	Value to be set

Interface

- [Iuc480Gain](#)

Related functions

- [Gain_GetHwGainFactor](#)

3.2.20.9 Gain_GetHwGainFactorDefaults

Syntax

```
Gain_GetHwGainFactorDefaults (INT *pnMasterFactor, INT *pnRedFactor,  
                              INT *pnGreenFactor, INT *pnBlueFactor)
```

Description

Returns the default values for the gain factors.

Parameter

pnMasterFactor	Pointer to the variable to which the default value for the total gain is written.
pnRedFactor	Pointer to the variable to which the default value for the red gain is written.
pnGreenFactor	Pointer to the variable to which the default value for the green gain is written.
pnBlueFactor	Pointer to the variable to which the default value for the blue gain is written.

Interface

- [Iuc480Gain](#)

Related functions

- [Gain_GetHwGainFactor](#)

3.2.20.10 Gain_InquireHwGainFactor

Syntax

```
Gain_InquireHwGainFactor (INT nWhich, INT nGain, INT* pnFactor)
```

Description

Converts a normalized value [0,100] for the gain into a gain factor.

Parameter

nWhich	Specifies the gain channel for which values are queried: <ul style="list-style-type: none">• 0 = total gain• 1 = red gain• 2 = green gain• 3 = blue gain
nGain	The normalized gain value to be converted into a gain factor
pnFactor	Pointer to the variable to which the gain value is written.

Interface

- [Iuc480Gain](#)

Related functions

- [Gain_GetHwGainFactorRange](#)

3.2.20.11 Gain_GetHwGainFactorRange

Syntax

```
Gain_GetHwGainFactorRange (INT nWhich, INT* pnMin, INT* pnMax)
```

Description

Returns the value range for the gain factors.

Parameter

nWhich	Specifies the gain channel for which values are queried: <ul style="list-style-type: none"> • 0 = total gain • 1 = red gain • 2 = green gain • 3 = blue gain
pnMin	Pointer to the variable to which the minimum value is written.
pnMax	Pointer to the variable to which the maximum value is written.

Interface

- [Iuc480Gain](#)

Related functions

- [Gain_InquireHwGainFactor](#)

3.2.20.12 Gain_IsGainBoostSupported

Syntax

```
Gain_IsGainBoostSupported (bool* pbSupported)
```

Description

Returns whether the camera provides the analog gain boost feature.

Parameter

- pbSupported: Pointer to the variable to which the return value is written.

Interface

- [Iuc480Gain](#)

Related functions

- [Gain_GetGainBoostValue](#)
- [Gain_SetGainBoostValue](#)

3.2.20.13 Gain_GetGainBoostValue

Syntax

```
Gain_GetGainBoostValue (long* plValue)
```

Description

Returns whether the Gain Boost feature is enabled [0,1].

Parameter

- `pIValue`: Pointer to the variable to which the return value is written.

Interface

- [Iuc480Gain](#)

Related functions

- [Gain_IsGainBoostSupported](#)
- [Gain_SetGainBoostValue](#)

3.2.20.14 Gain_SetGainBoostValue

Syntax

```
Gain_SetGainBoostValue (long lValue)
```

Description

Sets the value for Gain Boost [0,1].

Parameter

- `lValue`: value to be set

Interface

- [Iuc480Gain](#)


Related functions

- [Gain_IsGainBoostSupported](#)
- [Gain_GetGainBoostValue](#)

3.2.21 Iuc480HotPixel

IID_Iuc480HotPixel

This interface provides special functions for the correction of the sensor's hot pixel that are not covered by the DirectShow standard. 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. To use the interface, you must integrate the `uc480CaptureInterface.h` header file into your project.



This correction will not work with subsampling or with binning factors greater than 2.

Iuc480HotPixel provides the following functions:

HotPixel_DeleteCameraUserList	Deletes the user-defined hot pixel list from the camera EEPROM.
HotPixel_DisableCorrection	Disables hot pixel correction.
HotPixel_EnableCameraCorrection	Enables hot pixel correction using the hot pixel list(s) stored in the camera EEPROM.
HotPixel_EnableSoftwareUserCorrection	Enables hot pixel correction using the user's hot pixel list stored in the computer. This requires

	the user's hot pixel list to be set (HotPixel_SetSoftwareUserList).
HotPixel_GetCameraFactoryList	Returns the factory-set hot pixel list.
HotPixel_GetCameraFactoryListExist	Indicates whether the factory-set hot pixel list exists.
HotPixel_GetCameraFactoryListNumber	Returns the number of hot pixels in the factory-set hot pixel list.
HotPixel_GetCameraUserList	Returns the user-defined hot pixel list stored in the camera EEPROM.
HotPixel_GetCameraUserListMaxNumber	Returns the maximum number of hot pixels that the user can store in the camera EEPROM.
HotPixel_GetCameraUserListNumber	Returns the number of hot pixels in the user-defined hot pixel list stored in the camera EEPROM.
HotPixel_GetCameraUserListExist	Indicates whether the user-defined hot pixel list exists in the camera EEPROM.
HotPixel_GetCorrectionMode	Returns the currently set hot pixel correction mode.
HotPixel_GetMergedCameraList	Returns the merged list.
HotPixel_GetMergedCameraListNumber	Returns the number of hot pixels in a merged list that combines the entries from the factory-set hot pixel list with those of the user-defined hot pixels list stored in the camera EEPROM.
HotPixel_GetSoftwareUserList	Returns the user-defined hot pixel list stored in the computer.
HotPixel_GetSoftwareUserListExist	Indicates whether the user-defined hot pixel list exists in the computer.
HotPixel_GetSoftwareUserListNumber	Returns the number of hot pixels in the user-defined hot pixel list stored in the computer.
HotPixel_GetSupportedCorrectionModes	Returns the supported hot pixel correction modes.
HotPixel_LoadUserList	Loads the user-defined hot pixel list from a file.
HotPixel_LoadUserListUnicode	Loads the user-defined hot pixel list from a file. The function can also be used with Unicode file names.
HotPixel_SaveUserList	Saves the user-defined hot pixel list to a file.
HotPixel_SaveUserListUnicode	Saves the user-defined hot pixel list to a file. The function can also be used with Unicode file names.
HotPixel_SetCameraUserList	Sets the user-defined hot pixel list stored in the camera EEPROM.
HotPixel_SetSoftwareUserList	Sets the user-defined hot pixel list that is stored in the computer.
HotPixel_SensorCorrection	Enables/disables on some uc480 cameras the sensor's own hot pixel correction.

3.2.21.1 HotPixel_DeleteCameraUserList

Syntax

HotPixel_DeleteCameraUserList (void)

Description

Deletes the user-defined hot pixel list from the camera EEPROM.

Parameter

<none>

Interface

- [Iuc480HotPixel](#)

Related functions

- [HotPixel_GetCameraUserList](#)
- [HotPixel_GetCameraUserListExist](#)
- [HotPixel_GetCameraUserListMaxNumber](#)
- [HotPixel_GetCameraUserListNumber](#)
- [HotPixel_SetCameraUserList](#)
- [HotPixel_LoadUserList](#)
- [HotPixel_LoadUserListUnicode](#)
- [HotPixel_SaveUserList](#)
- [HotPixel_SaveUserListUnicode](#)

3.2.21.2 HotPixel_DisableCorrection

Syntax

HotPixel_DisableCorrection (void)

Description

Disables hot pixel correction.

Parameter

<none>

Interface

- [Iuc480HotPixel](#)

Related functions

- [HotPixel_EnableCameraCorrection](#)
- [HotPixel_EnableSoftwareUserCorrection](#)
- [HotPixel_SensorCorrection](#)

3.2.21.3 HotPixel_EnableCameraCorrection

Syntax

HotPixel_EnableCameraCorrection (void)

Description

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

Parameter

<none>

Interface

- [Iuc480HotPixel](#)

Related functions

- [HotPixel_DisableCorrection](#)
- [HotPixel_EnableSoftwareUserCorrection](#)
- [HotPixel_SensorCorrection](#)

3.2.21.4 HotPixel_EnableSoftwareUserCorrection

Syntax

HotPixel_EnableSoftwareUserCorrection (void)

Description

Enables hot pixel correction using the user's hot pixel list stored in the computer. This requires the user's hot pixel list to be set (see [HotPixel_SetCameraUserList](#)).

Parameter

<none>

Interface

- [Iuc480HotPixel](#)

Related functions

- [HotPixel_DisableCorrection](#)
- [HotPixel_EnableCameraCorrection](#)
- [HotPixel_SensorCorrection](#)

3.2.21.5 HotPixel_GetCameraFactoryList

Syntax

HotPixel_GetCameraFactoryList (WORD *pList, INT nNumber)

Description

Returns the factory-set hot pixel list.

Parameter

pList	Pointer to the factory-set hot pixel list
nNumber	Number of hot pixels in the hot pixel list

Interface

- [Iuc480HotPixel](#)

Related functions

- [HotPixel_GetCameraFactoryListExist](#)
- [HotPixel_GetCameraFactoryListNumber](#)

3.2.21.6 HotPixel_GetCameraFactoryListExist

Syntax

HotPixel_GetCameraFactoryListExist (void)

Description

Indicates whether the factory-set hot pixel list exists.

Parameter

<none>

Interface

- [Iuc480HotPixel](#)

Related functions

- [HotPixel_GetCameraFactoryList](#)
- [HotPixel_GetCameraFactoryListNumber](#)

3.2.21.7 HotPixel_GetCameraFactoryListNumber

Syntax

HotPixel_GetCameraFactoryListNumber (INT* pnNumber)

Description

Returns the number of hot pixels in the factory-set hot pixel list.

Parameter

- pnNumber: Pointer to the variable in which the return value is written.

Interface

- [Iuc480HotPixel](#)

Related functions

- [HotPixel_GetCameraFactoryList](#)
- [HotPixel_GetCameraFactoryListExist](#)

3.2.21.8 HotPixel_GetCameraUserList

Syntax

```
HotPixel_GetCameraUserList (WORD *pList, INT nNumber)
```

Description

Returns the user-defined hot pixel list stored in the camera EEPROM.

Parameter

pList	Pointer to the user-defined hot pixel list
nNumber	Number of hot pixels in the hot pixel list

Interface

- [Iuc480HotPixel](#)

Related functions

- [HotPixel_DeleteCameraUserList](#)
- [HotPixel_GetCameraUserListExist](#)
- [HotPixel_GetCameraUserListMaxNumber](#)
- [HotPixel_GetCameraUserListNumber](#)
- [HotPixel_SetCameraUserList](#)
- [HotPixel_LoadUserList](#)
- [HotPixel_LoadUserListUnicode](#)
- [HotPixel_SaveUserList](#)
- [HotPixel_SaveUserListUnicode](#)

3.2.21.9 HotPixel_GetCameraUserListExist

Syntax

```
HotPixel_GetCameraUserListExist (void)
```

Description

Indicates whether the user-defined hot pixel list exists in the camera EEPROM.

Parameter

<none>

Interface

- [Iuc480HotPixel](#)

Related functions

- [HotPixel_DeleteCameraUserList](#)
- [HotPixel_GetCameraUserList](#)
- [HotPixel_GetCameraUserListMaxNumber](#)
- [HotPixel_GetCameraUserListNumber](#)
- [HotPixel_SetCameraUserList](#)

- [HotPixel_LoadUserList](#)
- [HotPixel_LoadUserListUnicode](#)
- [HotPixel_SaveUserList](#)
- [HotPixel_SaveUserListUnicode](#)

3.2.21.10 HotPixel_GetCameraUserListMaxNumber

Syntax

```
HotPixel_GetCameraUserListMaxNumber (INT* pnNumber)
```

Description

Returns the maximum number of hot pixels that the user can store in the camera EEPROM.

Parameter

- `pnNumber`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480HotPixel](#)

Related functions

- [HotPixel_DeleteCameraUserList](#)
- [HotPixel_GetCameraUserList](#)
- [HotPixel_GetCameraUserListExist](#)
- [HotPixel_GetCameraUserListNumber](#)
- [HotPixel_SetCameraUserList](#)
- [HotPixel_LoadUserList](#)
- [HotPixel_LoadUserListUnicode](#)
- [HotPixel_SaveUserList](#)
- [HotPixel_SaveUserListUnicode](#)

3.2.21.11 HotPixel_GetCameraUserListNumber

Syntax

```
HotPixel_GetCameraUserListNumber (INT* pnNumber)
```

Description

Returns the number of hot pixels in the user-defined hot pixel list stored in the camera EEPROM.

Parameter

- `pnNumber`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480HotPixel](#)

Related functions

- [HotPixel_DeleteCameraUserList](#)
- [HotPixel_GetCameraUserList](#)

- [HotPixel_GetCameraUserListMaxNumber](#)
- [HotPixel_GetCameraUserListExist](#)
- [HotPixel_SetCameraUserList](#)
- [HotPixel_LoadUserList](#)
- [HotPixel_LoadUserListUnicode](#)
- [HotPixel_SaveUserList](#)
- [HotPixel_SaveUserListUnicode](#)

3.2.21.12 HotPixel_GetCorrectionMode

Syntax

```
HotPixel_GetCorrectionMode (INT* pnMode)
```

Description

Returns the currently set hot pixel correction mode.

Parameter

pnMode	<p>The following Integer values are returned:</p> <ul style="list-style-type: none"> • 0 (IS_HOTPIXEL_DISABLE_CORRECTION) • 1 (IS_HOTPIXEL_ENABLE_SENSOR_CORRECTION) • 2 (IS_HOTPIXEL_ENABLE_CAMERA_CORRECTION) • 3 (IS_HOTPIXEL_ENABLE_SENSOR_CORRECTION IS_HOTPIXEL_ENABLE_CAMERA_CORRECTION) • 4 (IS_HOTPIXEL_ENABLE_SOFTWARE_USER_CORRECTION) • 5 (IS_HOTPIXEL_ENABLE_SENSOR_CORRECTION IS_HOTPIXEL_ENABLE_SOFTWARE_USER_CORRECTION)
--------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Interface

- [Iuc480HotPixel](#)

Related functions

- [HotPixel_GetSupportedCorrectionModes](#)

3.2.21.13 HotPixel_GetMergedCameraList

Syntax

```
HotPixel_GetMergedCameraList (WORD *pList, INT nNumber)
```

Description

Returns the merged list.

Parameter

pList	Pointer to the merged hot pixel list.
nNumber	Number of hot pixels in the merged hot pixel list

Interface

- [Iuc480HotPixel](#)

Related functions

- [HotPixel_GetMergedCameraListNumber](#)

3.2.21.14 HotPixel_GetMergedCameraListNumber

Syntax

```
HotPixel_GetMergedCameraListNumber (INT* pnNumber)
```

Description

Returns the number of hot pixels in a merged list that combines the entries from the factory-set hot pixel list with those of the user-defined hot pixels list stored in the camera EEPROM.

Parameter

- `pnNumber`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480HotPixel](#)

Related functions

- [HotPixel_GetMergedCameraList](#)

3.2.21.15 HotPixel_GetSoftwareUserList

Syntax

```
HotPixel_GetSoftwareUserList (WORD *pList, INT nNumber)
```

Description

Returns the user-defined hot pixel list stored in the computer.

Parameter

<code>pList</code>	Pointer to the user-defined hot pixel list
<code>nNumber</code>	Number of hot pixels in the hot pixel list

Interface

- [Iuc480HotPixel](#)

Related functions

- [HotPixel_GetSoftwareUserListExist](#)
- [HotPixel_GetSoftwareUserListNumber](#)
- [HotPixel_SetSoftwareUserList](#)

3.2.21.16 HotPixel_GetSoftwareUserListExist

Syntax

```
HotPixel_GetSoftwareUserListExist (void)
```

Description

Indicates whether the user-defined hot pixel list exists in the computer.

Parameter

<none>

Interface

- [Iuc480HotPixel](#)

Related functions

- [HotPixel_GetSoftwareUserList](#)
- [HotPixel_GetSoftwareUserListNumber](#)
- [HotPixel_SetSoftwareUserList](#)

3.2.21.17 HotPixel_GetSoftwareUserListNumber

Syntax

```
HotPixel_GetSoftwareUserListNumber (INT* pnNumber)
```

Description

Returns the number of hot pixels in the user-defined hot pixel list stored in the computer.

Parameter

- `pnNumber`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480HotPixel](#)

Related functions

- [HotPixel_GetSoftwareUserList](#)
- [HotPixel_GetSoftwareUserListExist](#)
- [HotPixel_SetSoftwareUserList](#)

3.2.21.18 HotPixel_GetSupportedCorrectionModes

Syntax

```
HotPixel_GetSupportedCorrectionModes (INT* pnMode)
```

Description

Returns the supported hot pixel correction modes. The return value is a bitmask with the following constants (combined by OR):

- `IS_HOTPIXEL_ENABLE_CAMERA_CORRECTION`: Hot pixel correction is possible via the hot pixel list in the camera EEPROM.
- `IS_HOTPIXEL_ENABLE_SOFTWARE_USER_CORRECTION`: Hot pixel correction is possible via the user-

defined hot pixel list.

- `IS_HOTPIXEL_ENABLE_SENSOR_CORRECTION`: Hot pixel correction is possible via the sensor-internal hot pixel correction.

Parameter

- `pMode`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480HotPixel](#)

Related functions

- [HotPixel_GetCorrectionMode](#)

3.2.21.19 HotPixel_LoadUserList

Syntax

```
HotPixel_LoadUserList (char* pFile)
```

Description

Loads the user-defined hot pixel list from a file.

Parameter

- `pFile`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480HotPixel](#)

Related functions

- [HotPixel_DeleteCameraUserList](#)
- [HotPixel_GetCameraUserList](#)
- [HotPixel_GetCameraUserListMaxNumber](#)
- [HotPixel_GetCameraUserListNumber](#)
- [HotPixel_GetCameraUserListExist](#)
- [HotPixel_SetCameraUserList](#)
- [HotPixel_LoadUserListUnicode](#)
- [HotPixel_SaveUserList](#)
- [HotPixel_SaveUserListUnicode](#)

3.2.21.20 HotPixel_LoadUserListUnicode

Syntax

```
HotPixel_LoadUserListUnicode (wchar_t* pFile)
```

Description

Loads the user-defined hot pixel list from a file. The function can also be used with Unicode file names.

Parameter

- `pFile`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480HotPixel](#)

Related functions

- [HotPixel_DeleteCameraUserList](#)
- [HotPixel_GetCameraUserList](#)
- [HotPixel_GetCameraUserListMaxNumber](#)
- [HotPixel_GetCameraUserListNumber](#)
- [HotPixel_GetCameraUserListExist](#)
- [HotPixel_SetCameraUserList](#)
- [HotPixel_LoadUserList](#)
- [HotPixel_SaveUserList](#)
- [HotPixel_SaveUserListUnicode](#)

3.2.21.21 HotPixel_SaveUserList

Syntax

```
HotPixel_SaveUserList (char* pFile)
```

Description

Saves the user-defined hot pixel list to a file.

Parameter

- `pFile`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480HotPixel](#)

Related functions

- [HotPixel_DeleteCameraUserList](#)
- [HotPixel_GetCameraUserList](#)
- [HotPixel_GetCameraUserListMaxNumber](#)
- [HotPixel_GetCameraUserListNumber](#)
- [HotPixel_GetCameraUserListExist](#)
- [HotPixel_SetCameraUserList](#)
- [HotPixel_LoadUserList](#)
- [HotPixel_LoadUserListUnicode](#)
- [HotPixel_SaveUserListUnicode](#)

3.2.21.22 HotPixel_SaveUserListUnicode

Syntax

```
HotPixel_SaveUserListUnicode (wchar_t* pFile)
```

Description

Saves the user-defined hot pixel list to a file. The function can also be used with Unicode file names.

Parameter

- **pFile**: Pointer to the variable in which the return value is written.

Interface

- [Iuc480HotPixel](#)

Related functions

- [HotPixel_DeleteCameraUserList](#)
- [HotPixel_GetCameraUserList](#)
- [HotPixel_GetCameraUserListMaxNumber](#)
- [HotPixel_GetCameraUserListNumber](#)
- [HotPixel_GetCameraUserListExist](#)
- [HotPixel_SetCameraUserList](#)
- [HotPixel_LoadUserList](#)
- [HotPixel_LoadUserListUnicode](#)
- [HotPixel_SaveUserList](#)

3.2.21.23 HotPixel_SetCameraUserList

Syntax

```
HotPixel_SetCameraUserList (WORD *pList, INT nNumber)
```

Description

Sets the user-defined hot pixel list stored in the camera EEPROM.

Parameter

pList	Pointer to the user-defined hot pixel list
nNumber	Number of hot pixels in the hot pixel list

Interface

- [Iuc480HotPixel](#)

Related functions

- [HotPixel_DeleteCameraUserList](#)
- [HotPixel_GetCameraUserList](#)
- [HotPixel_GetCameraUserListMaxNumber](#)
- [HotPixel_GetCameraUserListNumber](#)

- [HotPixel_GetCameraUserListExist](#)
- [HotPixel_LoadUserList](#)
- [HotPixel_LoadUserListUnicode](#)
- [HotPixel_SaveUserList](#)
- [HotPixel_SaveUserListUnicode](#)

3.2.21.24 HotPixel_SetSoftwareUserList

Syntax

HotPixel_SetSoftwareUserList (WORD *pList, INT nNumber)

Description

Sets the user-defined hot pixel list that is stored in the computer.

Parameter

pList	Pointer to the user-defined hot pixel list
nNumber	Number of hot pixels in the hot pixel list

Interface

- [Iuc480HotPixel](#)

Related functions

- [HotPixel_GetSoftwareUserList](#)
- [HotPixel_GetSoftwareUserListExist](#)
- [HotPixel_GetSoftwareUserListNumber](#)

3.2.21.25 HotPixel_SensorCorrection

Syntax

HotPixel_SensorCorrection (bool bEnable)

Description

Enables/disables on some uc480 cameras the sensor's own hot pixel correction.

Parameter

bEnable	TRUE = Enables sensor's hot pixel correction FALSE = Disables sensor's hot pixel correction
---------	------------------------------------------------------------------------------------------------

Interface

- [Iuc480HotPixel](#)

Related functions

- [HotPixel_DisableCorrection](#)
- [HotPixel_EnableCameraCorrection](#)
- [HotPixel_EnableSoftwareUserCorrection](#)

3.2.22 Iuc480IO

IID_Iuc480IO

This interface provides special functions for the digital in-/outputs (GPIO) for some uc480 models that are not covered by the DirectShow standard. To use the interface, you must integrate the uc480CaptureInterface.h header file into your project.



The GPIOs are only available on some DCx camera models. The GPIOs are not provided with optocouplers and use TTL/LVCMOS voltages. For information on GPIO wiring, please refer to the "Electrical specifications" chapter in the DCx manual.

Iuc480IO provides the following functions:

IO_SetGPIO	Sets the digital output (GPIO).
IO_GetGPIO	Returns the state of the digital output (GPIO).
IO_SetIOMask	Sets the direction of the additional digital in-/outputs (GPIO).
IO_GetIOMask	Returns the direction of the additional in-/outputs (GPIO).
IO_IOMaskInputSupported	Returns the GPIOs, which can be used for input.
IO_IOMaskOutputSupported	Returns the GPIOs, which can be used for output.

3.2.22.1 IO_SetGPIO

Syntax

```
IO_SetGPIO (INT nIO)
```

Description

Sets the additional digital outputs (GPIO) on some uc480 models.



To be able to set the Status of a GPIO you must first configure the GPIO as output using [IO_SetIOMask](#).
If only one GPIO is configured as output the command IO_SetGPIO has no effect on the other GPIO.



The GPIOs are only available on some DCx camera models. The GPIOs are not provided with optocouplers and use TTL/LVCMOS voltages. For information on GPIO wiring, please refer to the "Electrical specifications" chapter in the DCx manual.

Parameter

nIO	Bit mask for outputs
0x00 (00)	Sets both outputs to 0
0x01 (01)	Sets the first output to 1, the second one to 0
0x02 (10)	Set th first output to 0, the second one to 1
0x03 (11)	Sets both outputs to 1.

Interface

- [Iuc480IO](#)

Related functions

- [IO_GetGPIO](#)
- [IO_SetIOMask](#)

3.2.22.2 IO_GetGPIO

Syntax

```
IO_GetGPIO (INT* pnIO)
```

Description

Returns the additional digital outputs (GPIO) on some uc480 models.



The GPIOs are only available on some DCx camera models. The GPIOs are not provided with optocouplers and use TTL/LVCMOS voltages. For information on GPIO wiring, please refer to the "Electrical specifications" chapter in the DCx manual.

Parameter

- `pnIO`: Reads the states of the GPIOs. If a GPIO is configured as input this reads the signal applied to the GPIO.

Interface

- [Iuc480IO](#)

Related functions

- [IO_SetGPIO](#)
- [IO_SetIOMask](#)

3.2.22.3 IO_SetIOMask

Syntax

```
IO_SetIOMask (INT nIOMask)
```

Description

Using `IO_SetIOMask`, you can configure the direction of the additional in-/outputs (GPIO) of some uc480 models. The [IO_SetGPIO](#) function sets or returns the current GPIO states.



The GPIOs are only available on some DCx camera models. The GPIOs are not provided with optocouplers and use TTL/LVCMOS voltages. For information on GPIO wiring, please refer to the "Electrical specifications" chapter in the DCx manual.

Parameter

nIOMask	Bit mask for in-/outputs
0x00 (00)	Use both GPIOs as inputs
0x01 (01)	Use the first GPIO as output, the second one as input
0x02 (10)	Use the first GPIO as input, the second one as output
0x03 (11)	Use both GPIOs as output

Interface

- [Iuc480IO](#)

Related functions

- [IO_GetGPIO](#)
- [IO_IOMaskInputSupported](#)

3.2.22.4 IO_GetIOMask

Syntax

```
IO_GetIOMask (INT* pnIOMask)
```

Description

Returns the direction of the additional digital outputs (GPIO) on some uc480 models.



The GPIOs are only available on some DCx camera models. The GPIOs are not provided with optocouplers and use TTL/LVCMOS voltages. For information on GPIO wiring, please refer to the "Electrical specifications" chapter in the DCx manual.

Parameter

- `pnIOMask`: Returns the current bit mask

Interface

- [Iuc480IO](#)

Related functions

- [IO_GetGPIO](#)
- [IO_SetIOMask](#)
- [IO_IOMaskInputSupported](#)

3.2.22.5 IO_IOMaskInputSupported

Syntax

```
IO_IOMaskInputSupported (INT* pnIOMaskInSupp)
```

Description

Returns the GPIOs which can be used as inputs.



The GPIOs are only available on some DCx camera models. The GPIOs are not provided with optocouplers and use TTL/LVCMOS voltages. For information on GPIO wiring, please refer to the "Electrical specifications" chapter in the DCx manual.

Parameter

- `pnIOMaskInSupp`: Returns the IOs to be used as inputs

Interface

- [Iuc480IO](#)

Related functions

- [IO_GetGPIO](#)
- [IO_IOMaskOutputSupported](#)

3.2.22.6 IO_IOMaskOutputSupported

Syntax

```
IO_IOMaskOutputSupported (INT* pnIOMaskOutSupp)
```

Description

Returns which GPIOs can be used as outputs.



The GPIOs are only available on some DCx camera models. The GPIOs are not provided with optocouplers and use TTL/LVCMOS voltages. For information on GPIO wiring, please refer to the "Electrical specifications" chapter in the DCx manual.

Parameter

- `pnIOMaskOutSupp`: Returns the IOs to be used as outputs

Interface

- [Iuc480IO](#)

Related functions

- [IO_GetGPIO](#)
- [IO_IOMaskInputSupported](#)

3.2.23 Iuc480Trigger

IID_Iuc480Trigger

This interface provides special functions for triggering the camera that are not covered by the DirectShow standard. To use the interface, you must integrate the `Iuc480CaptureInterface.h` header file into your project.

`Iuc480Trigger` provides the following functions:

Trigger_GetBurstSize	Returns the currently set number of images in a burst.
Trigger_GetBurstSizeRange	Returns the value range and the increment for the number of images in a burst.
Trigger_GetBurstSizeSupported	Returns if the camera supports the burst trigger mode.
Trigger_SetBurstSize	Sets the number of images in a burst.
Trigger_GetTriggerMode	Returns the set trigger mode.
Trigger_SetTriggerMode	Sets the trigger mode.
Trigger_GetTriggerStatus	Returns the current trigger status.
Trigger_IsFallingEdgeSupported	Returns information on whether the camera supports hardware triggering on the falling signal edge.
Trigger_IsRisingEdgeSupported	Returns information on whether the camera supports hardware triggering on the rising signal edge.
Trigger_IsSoftwareTriggerSupported	Returns information on whether the camera supports software triggering.


Note on enabling/disabling triggered image capture

luc480Trigger only selects the trigger mode. The DirectShow interface [IAMVideoControl](#) must be used to enable/disable triggered image capture.


Note on the software trigger mode of the USB uc480 XS

With the USB uc480 XS, images with a high resolution of up to 8 megapixels can be captured in the triggered mode as follows:

- Stop the image capture
- Activate the software trigger
- Select the new image format. For a list of permissible image formats, please see Image Size on the [ISpecifyPropertyPages](#).
- Trigger the image capture via the default function [IAMVideoControl](#) with the set flag `VideoControlFlag_Trigger`

Disabling the use of large image formats:

- Select an image format that can be used in live mode (smaller than or equal to 1280x720 pixels)
- Disable the software trigger

3.2.23.1 Trigger_GetBurstSize

Syntax

```
Trigger_GetBurstSize (UINT* pnBurstSize)
```

Description

Returns the currently set number of images in a burst.

Parameter

- `pnBurstSize`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Trigger](#)

Related functions

- [Trigger_GetBurstSizeRange](#)
- [Trigger_GetBurstSizeSupported](#)
- [Trigger_SetBurstSize](#)

3.2.23.2 Trigger_GetBurstSizeRange

Syntax

```
Trigger_GetBurstSizeRange (UINT* pnMin, UINT* pnMax, UINT* pnInc)
```

Description

Returns the value range and the increment for the number of images in a burst.

Parameter

pnMin	Pointer to the variable in which the minimum value is written.
pnMax	Pointer to the variable in which the maximum value is written.
pnInc	Pointer to the variable in which the increment is written.

Interface

- [Iuc480Trigger](#)

Related functions

- [Trigger_GetBurstSize](#)
- [Trigger_GetBurstSizeSupported](#)
- [Trigger_SetBurstSize](#)

3.2.23.3 Trigger_GetBurstSizeSupported

Syntax

```
Trigger_GetBurstSizeSupported (UINT* pnSupported)
```

Description

Returns if the camera supports the burst trigger mode.

Parameter

- pnSupported: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Trigger](#)

Related functions

- [Trigger_GetBurstSize](#)
- [Trigger_GetBurstSizeRange](#)
- [Trigger_SetBurstSize](#)

3.2.23.4 Trigger_SetBurstSize

Syntax

```
Trigger_SetBurstSize (UINT nBurstSize)
```

Description

Sets the number of images in a burst.

Parameter

- `nBurstSize`: value to be set

Interface

- [Iuc480Trigger](#)

Related functions

- [Trigger_GetBurstSize](#)
- [Trigger_GetBurstSizeRange](#)
- [Trigger_GetBurstSizeSupported](#)

3.2.23.5 Trigger_GetTriggerMode

Syntax

```
Trigger_GetTriggerMode (long* pnMode)
```

Description

Returns the set trigger mode.

Parameter

- `pnMode`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Trigger](#)

Related functions

- [Trigger_SetTriggerMode](#)
- [Trigger_GetTriggerStatus](#)

3.2.23.6 Trigger_SetTriggerMode

Syntax

```
Trigger_SetTriggerMode (long nMode)
```

Description

Sets the trigger mode.

Parameter

nMode	Trigger mode	Trigger event
IS_SET_TRIGGER_HI_LO	Hardware trigger	Falling signal edge
IS_SET_TRIGGER_LO_HI	Hardware trigger	Rising signal edge
IS_SET_TRIGGER_SOFTWARE	Software trigger	Starts a continually triggered live image.

Interface

- [Iuc480Trigger](#)

Related functions

- [Trigger_GetTriggerMode](#)
- [Trigger_GetTriggerStatus](#)

3.2.23.7 Trigger_GetTriggerStatus**Syntax**

```
Trigger_GetTriggerStatus (long* pnMode)
```

Description

Returns the current trigger status.

Parameter

- pnMode: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Trigger](#)

Related functions

- [Trigger_GetTriggerMode](#)
- [Trigger_IsSoftwareTriggerSupported](#)

3.2.23.8 Trigger_IsFallingEdgeSupported**Syntax**

```
Trigger_IsFallingEdgeSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports hardware triggering on the falling signal edge.

Parameter

- pbSupported: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Trigger](#)

Related functions

- [Trigger_IsSoftwareTriggerSupported](#)
- [Trigger_IsRisingEdgeSupported](#)

3.2.23.9 Trigger_IsRisingEdgeSupported

Syntax

```
Trigger_IsRisingEdgeSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports hardware triggering on the rising signal edge.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Trigger](#)

Related functions

- [Trigger_IsSoftwareTriggerSupported](#)
- [Trigger_IsFallingEdgeSupported](#)

3.2.23.10 Trigger_IsSoftwareTriggerSupported

Syntax

```
Trigger_IsSoftwareTriggerSupported (bool* pbSupported)
```

Description

Returns information on whether the camera supports software triggering.

Parameter

- `pbSupported`: Pointer to the variable in which the return value is written.

Interface

- [Iuc480Trigger](#)

Related functions

- [Trigger_IsRisingEdgeSupported](#)
- [Trigger_GetTriggerMode](#)

4 Troubleshooting

The table below lists most common issues that may occur when using DCx cameras with DirectShow. More information on setting and operating your DCx camera can be found in the DCx Camera Manual.

Failure description	Possible causes and remedy
Camera does not work in DirectShow: A DCx camera is connected and works in other applications. The camera is not displayed in DirectShow programs, however.	<ul style="list-style-type: none"> For use in DirectShow and manual camera registration, the DCx camera must have a unique camera ID between 1 and 24. This ID must be registered using the DirectShow Device Manager utility program. See also Configuring DCx cameras for DirectShow.
Some cameras are not displayed: Several cameras are connected, but some of the cameras are not displayed in a DirectShow application.	<ul style="list-style-type: none"> Use DirectShow Device Manager to check whether all cameras are registered properly. See also Configuring DCx cameras for DirectShow. Use uc480 Camera Manager or the status LED to check whether all cameras were detected correctly. See also the Installation and Connection chapters in the DCx Camera Manual.
Camera image is displayed upside down: The image is displayed right-side up in other uc480 applications. The image is upside down in DirectShow programs.	<ul style="list-style-type: none"> If a program displays the image upside down, the image can be mirrored with the Flip Image vertically option (see IAMVideoControl). You can enable this option permanently, see Configuring DCx cameras for DirectShow. Explanation: Not all DirectShow render filters interpret the image height in the same way. Some filters expect a negative value, and others require a positive value. The DirectShow interface of the DCx camera supplies positive image heights.
The exposure time cannot be set precisely: In DirectShow, the exposure time of the DCx camera can only be set in rough steps.	<ul style="list-style-type: none"> The exposure time can be set very precisely using the uc480 specific function SetExposureTime or using the uc480 properties page. Explanation: By default, the exposure time is set using the <code>IAMCameraControl</code> interface in DirectShow. This setting of the exposure time uses very large increments, however. Finer adjustment is possible using the proprietary uc480 functions.

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 for our most up-to-date contact information.

Index

A

AOI	17
GetAutoBrightnessAOI	56
GetAutoWBAOI	57
GetImageAOI	55
GetIncPosX	58
GetIncPosY	58
GetMinMaxPosX	59
GetMinMaxPosY	59
GetMinMaxSizeX	60
GetMinMaxSizeY	60
GetSizeX	58
GetSizeY	59
IsImageAOISupported	54
SetAutoBrightnessAOI	56
SetAutoWBAOI	57
SetImageAOI	55
AutoFeatures	
GetAutoBrightnessAOI	66
GetAutoBrightnessMaxExposure	64
GetAutoBrightnessMaxGain	64
GetAutoBrightnessReference	63
GetAutoBrightnessSpeed	65
GetAutoWBAOI	69
GetAutoWBGainOffsets	67
GetAutoWBGainRange	68
GetAutoWBSpeed	68
SetAutoBrightnessAOI	65
SetAutoBrightnessMaxExposure	63
SetAutoBrightnessMaxGain	64
SetAutoBrightnessReference	62
SetAutoBrightnessSpeed	65
SetAutoWBAOI	69
SetAutoWBGainOffsets	66
SetAutoWBGainRange	67
SetAutoWBSpeed	68
AutoFramerate	
GetFramerate	72
AutoFramerateDriver	
Enable	72
IsEnabled	72
IsSupported	71

AutoFramerateSensor	
Enable	71
IsEnabled	70
IsSupported	70
Automatic image control	
Exposure time (Auto Exposure Shutter, AES)	43
AutoParameter	
GetAWBType	73
GetEnableAWB	75
GetRGBColorModelAWB	75
GetSupportedAWBType	74
GetSupportedRGBColorModelAWB	76
SetAWBType	74
SetEnableAWB	75
SetRGBColorModelAWB	76

B

Binning	17
---------	----

C

CameraLUT	
GetCameraLUT	77
SetCameraLUT	78
Capture	
GetBadPixelCorrection	84
GetDeviceInfo	80
GetDLLVersion	81
GetWhiteBalanceMultipliers	86
Hotpixel	82
LoadSettings	85
ResetDefaults	85
SaveSettings	84
SetBadPixelCorrection	84
SetWhiteBalanceMultipliers	86
CaptureEx	
GetGainBoost	87
GetHardwareGamma	88
LoadParameters	89
ParameterSet	90
SaveParameters	89
SetGainBoost	87
SetHardwareGamma	88
Color format	17
ColorConverter	
GetCurrentMode	92
GetDefaultMode	92
GetSupportedModes	93
SetMode	93

ColorTemperature		GetEdgeEnhancementRange	116
GetDefaultValue	98	e	
GetRange	98	SetEdgeEnhancement	116
GetValue	97	Event	
IsSupported	97	DisableEvent	120
SetValue	97	EnableEvent	118
Connecting a DCx Camera	13, 14	EnableMessage	120
		ExitEvent	120
		InitEvent	117
		Example program	9
		Exposure time	157
D			
DeviceFeature			
GetAllowRawWithLUT	103		
GetDefaultLogMode	103	F	
GetLineScanMode	110	Filter interfaces	43
GetLineScanNumber	111	IAMDroppedFrames	44
GetLogMode	104	IAMFilterMiscFlags	45
GetLogModeManualGain	105	IAMVideoControl	45
GetLogModeManualGainDefault	106	IAMVideoProcAmp	46
GetLogModeManualGainRange	106	IKsPropertySet	46
GetLogModeManualValue	108	ISpecifyPropertyPages	47
GetLogModeManualValueDefault	108	Iuc480AOI	53
GetLogModeManualValueRange	109	Iuc480AutoFeatures	61
GetShutterMode	112	Iuc480AutoFramerate	70
GetSupportedFeatures	101	Iuc480AutoParameter	73
GetVerticalAOIMergeMode	113	Iuc480CameraLUT	77
GetVerticalAOIMergePosition	114	Iuc480Capture	80
SetAllowRawWithLUT	103	Iuc480CaptureEx	86
SetLineScanMode	110	Iuc480ColorConverter	91
SetLineScanNumber	111	Iuc480ColorTemperature	94
SetLogMode	105	Iuc480DeviceFeatures	99
SetLogModeManualGain	107	Iuc480EdgeEnhancement	115
SetLogModeManualValueRange	109	Iuc480Flash	122
SetShutterMode	112	Iuc480Gain	127
SetVerticalAOIMergeMode	113	Iuc480HotPixel	134
SetVerticalAOIMergePosition	114	Iuc480Trigger	151
DirectShow	15	Flash	
DirectShow service	15	EnableGPIOPort	126
		GetDelay	125
		GetDelayRange	125
		GetDuration	124
		GetDurationRange	124
		GetGlobalExposureWindow	126
		GetStrobeMode	123
		GetSupportedGPIOPorts	126
		SetDelayDuration	124
		SetStrobeMode	122
E		Frame rate	17
EdgeEnhancement			
GetEdgeEnhancement	115	G	
GetEdgeEnhancementDefault	115	Gain	

Gain

GetGainBoostValue	133
GetHwGain	129
GetHwGainDefaults	130
GetHwGainFactor	131
GetHwGainFactorDefaults	132
GetHwGainFactorRange	133
GetHwGainRange	130
InquireHwGainFactor	132
IsGainBoostSupported	133
IsMasterSupported	128
IsRGBSupported	128
SetGainBoostValue	134
SetHwGain	129
SetHwGainFactor	131

H

HotPixel

DeleteCameraUserList	136
DisableCorrection	136
EnableCameraCorrection	137
EnableSoftwareUserCorrection	137
GetCameraFactoryList	137
GetCameraFactoryListExist	138
GetCameraFactoryListNumber	138
GetCameraUserList	139
GetCameraUserListExist	139
GetCameraUserListMaxNumber	140
GetCameraUserListNumber	140
GetCorrectionMode	141
GetMergedCameraList	141
GetMergedCameraListNumber	142
GetSoftwareUserList	142
GetSoftwareUserListExist	143
GetSoftwareUserListNumber	143
GetSupportedCorrectionModes	143
LoadUserList	144
LoadUserListUnicode	144
SaveUserList	145
SaveUserListUnicode	146
SensorCorrection	147
SetCameraUserList	146
SetSoftwareUserList	147

I

Installation	13
--------------	----

IO

GetGPIO	149
GetIOMask	150
IOMaskInputSupported	150
IOMaskOutputSupported	151
SetGPIO	148
SetIOMask	149
luc480CapturePin	17
bandwidth	23
color mode	25
exposure	26, 27
pixel clock	23, 24

luc480Resample

binning	35, 36, 37, 38, 39
---------	--------------------

binning factor	36
color binning	40
color subsampling	35
subsampling	29, 30, 31, 32, 33, 34
subsampling factor	29, 30

luc480Scaler

image height	42
image size	43
image width	42
scaler	40, 41

O

Output pin interfaces	17
IAMStreamConfig	17
IKsPropertySet	20
ISpecifyPropertyPages	20
luc480CapturePin	22
luc480Resample	27
luc480Scaler	40

P

PCs with Energy Saving CPU Technology	158
Pixel clock	17
Property pages	17, 43

R

RGBModel	
GetDefaultMode	96
GetMode	95

RGBModel	
GetSupportedModes	96
IsSupported	94
SetMode	95

S

Subsampling	17
System requirements	12

T

Trigger	
GetBurstSize	152
GetBurstSizeRange	153
GetBurstSizeSupported	153
GetTriggerMode	154
GetTriggerStatus	155
IsFallingEdgeSupported	155
IsRisingEdgeSupported	156
IsSoftwareTriggerSupported	156
SetBurstSize	154
SetTriggerMode	154
Troubleshooting	157