User Manual

uEye® Software Development Kit (SDK)

uEye® Cameras

Driver version 3.24

Status: September 2008

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

Preface

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

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

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

Status: September 2008

Copyright

Security Advice

We would like to remind you that the contents of this operating manual do not constitute part of any previous or existing agreement, commitment or legal relationship, or an alteration thereof. All obligations of *IDS Imaging Development Systems GmbH* result from the respective contract of sale, which also includes the complete and exclusively applicable warranty regulations. These contractual warranty regulations are neither extended nor limited by the information contained in this operating manual. Should you require further information on this device, or encounter specific problems that are not discussed in sufficient detail in the operating manual, please contact your specialised dealer or system installer.

The device may be connected, taken into operation and maintained only by appropriately qualified personnel.

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

Trademarks

Contacting us

Visit our web site where you will find all the latest drivers and information about our software and hardware products.

Internet:	http://www.ueye.de http://www.ids-imaging.de	
Address:	IDS Imaging Development Systems GmbH D-74182 Obersulm Dimbacher Straße 6-8	
Fax:	07134/96196-99	
Email:	Sales: Support:	sales@ids-imaging.de support@ids-imaging.de

Inhalt

1 Introduction

Thank you for purchasing a *uEye® c*amera from *IDS Imaging Development Systems GmbH*.

The Software Development Kit (SDK) is included in delivery to enable the integration of *uEye®* Gigabit Ethernet cameras into proprietary programs under Windows 2000/XP/Vista 32-Bit and LINUX.

This manual describes the functions of the *uEye® Software Development Kit (SDK)*.

The uEye software development kit is, up to additional functionalities and/or design and connection-caused changes, almost identical with the SDK of the FALCON and/or EAGLE frame grabber.

Please also read the file *WhatsNew.txt*, which can be found on the installation CD. This file contains current information which may not yet be included in this edition of the printed manual.

We would like to wish you every success with our product.

2 Programming

2.1 Programming with Visual C++ 6.0, 7.0, 7.1 and 8.0

Please note that Microsoft modified the format of the lib File starting from version 6.0. The uEye driver has been created with Visual C++ 7.1. Thus the file uEye_api.lib is to be used only with a compiler of the version 6.0 or higher.

2.2 Programming with Visual Basic

The functions of the software development kit are exported with the call convention *_cdecl*. Visual BASIC needs however functions with the convention *_stdcall* (pascal convention). You can call up the uEye functions directly from Visual BASIC, if you replace the functions i*s_<function name>* by *iss_<function name>*.
All in this manual described “is_<Function name>” functions are _cdecl functions. _stdcall functions exist parallel to these functions as “iss_<Function name>”. Function parameters and return values are identical.

2.3 CAMINFO – data structure of the EEPROM

Using the *is_GetCameraInfo()* function the data which has been written to the uEye camera can be read out. The data structure (64 Byte) is build up as follows:

Char	SerNo[12]	Serial number of the camera
Char	ID[20]	e.g. „IDS GmbH“
Char	Version[10]	„V1.00“ or later versions
Char	Date[12]	„01.08.2004“ date of quality check
unsigned char	Select	Camera ID
unsigned char	Type	Camera type 64 = uEye USB2.0
Char	Reserved[8]	reserved

Table 1: CAMINFO data structure of the EEPROM

2.4 Colour Formats

Each of the colour formats supported by the uEye cameras has a different memory format. These are shown in the following table:

	Pixel Data			
Format	Byte 3 [Bit 31:24]	Byte 2 [Bit 23:16]	Byte 1 [Bit 15:8]	Byte 0 [Bit 7:0]
RGB32				
RGB24				
RGB16				
RGB15				
Bayer				
UYVY	Y1	V0	Y0	U0
Y8	Y3	Y2	Y1	Y0

Table 2: Colour formats

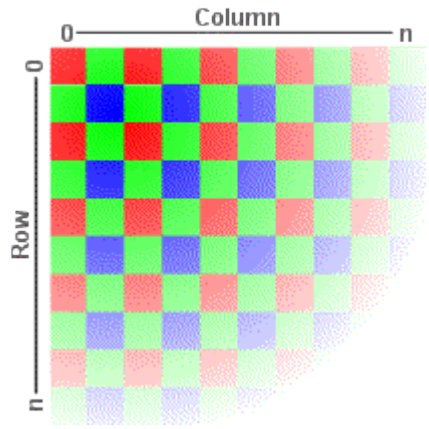

Figure 1: Principle structure of the Bayer-Pattern

In the case of the RGB16 and RGB15 data formats, from the internal 8 bits the upper ones are in use from R, G and B colours.

2.5 Possible image output modes

Bitmap Modus (BMP/DIB)

Once the *uEye Demo* sample application has been started the bitmap mode is activated. The image from the uEye camera is stored in the main memory of the PC. The live video display has to be programmed by the user. This should be done by using the CPU to generate a bitmap and then copying it to the VGA board. The great advantage of this mode is that it is compatible with all VGA cards and it is possible to access the image data in the memory. Overlay functions have to be programmed by the user. As Windows takes over the control of the image display, the image can be completely or partly overlapped by many other windows and dialogue boxes.

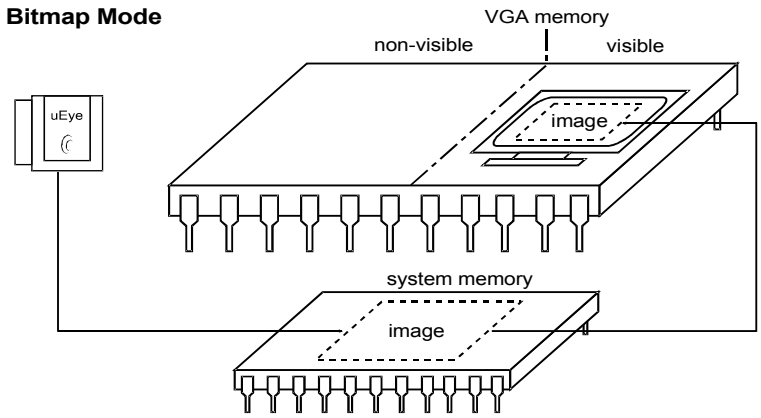

Figure 2: Bitmap mode

DirectDraw BackBuffer Modus (not available under LINUX)

In this mode, the image data is written to the non-visible area of the VGA card. The requirements for this are: installed DirectDraw driver, sufficient memory on the VGA card and back buffer support from the VGA cards' manufacturer. In overlay mode three non-visible image buffers are used:

- BackBuffer
- OverlayBuffer
- MixBuffer

The size of the three buffers is: video_x * video_y * color depth (in bytes per pixel). The video image is written into the back buffer. The graphics data can be written in the overlay buffer (see also *4.29 is_GetDC* and *4.72 is_ReleaseDC*). The overlay is not always faded on. It has to be made visible with *is_ShowDDOverlay()* (see *4.137 is_ShowDDOverlay*). As its key colour, the overlay uses black, thus an overlay cannot contain any black colour.

BackBuffer and OverlayBuffer are written into the MixBuffer. The overlay data is overlaid on the video image. The mix buffer is then copied to the visible area of the VGA card. The result is an live image with overlaid text and graphic overlay. The frame rate and the load of the CPU depend on the adjusted colour depth and on the place (system memory of the PC or memory of the VG card) of the BackBuffer.

The driver tries to allocate the buffers directly in the VGA card in order to use the cards high-speed image transfer when mixing the three buffers. If this allocation fails the buffers are stored in the system memory, from which image transfers may be slower and, depending on the graphics card, sometimes not possible at all. A scaling of the video picture is not possible in the Back-Buffer mode.

The BackBuffer mode is set as follows:

Mode = IS_SET_DM_DIRECTDRAW | IS_SET_DM_BACKBUFFER

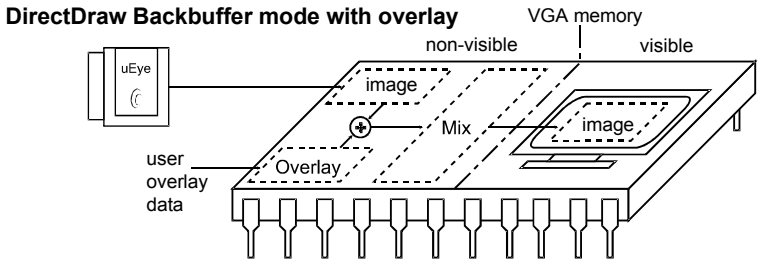

Figure 3: DirectDraw BackBuffer mode

DirectDraw overlay surface mode (not available under LINUX)

In this mode, a live image and at the same time display of the overlay can be achieved.
The video image is digitized to a non-visible area of the VGA board. This area always has to be on the VGA board. By defining a key colour and drawing with that colour to the output window, the video image will be displayed only in those parts where the key colour is used. When the window is filled with the key colour, the video image is displayed completely. Graphics and text elements which use a different colour will not be destroyed (non-destructive overlay).
The fading is done by the VGA chip and requires hardly any CPU cycles. This mode is not supported by all VGA chips and only in 8, 15, and 16 bit mode available.

The best text and graphics overlay is achieved with the following video mode:

Mode = *IS_SET_DM_DIRECTDRAW | IS_SET_DM_ALLOW_OVERLAY*

If the video image should be scaled to the size of a window, the following can be used:

Mode = *IS_SET_DM_DIRECTDRAW | IS_SET_DM_ALLOW_OVERLAY | IS_SET_DM_ALLOW_SCALING*

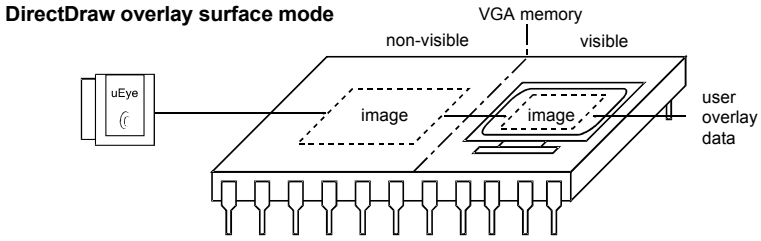

Figure 4: DirectDraw Overlay-Surface mode

The Back Buffer mode of the FALCON frame grabber is activated by setting the parameter *IS_SET_DM_DIRECTDRAW*. The BackBuffer mode of the uEye cameras is activated as follows: *IS_SET_DM_DIRECTDRAW* | *IS_SET_DM_BACKBUFFER*.

3 Function lists

3.1 Initialization and termination

Function list

is_ExitCamera	Closes the camera and deallocates memory which was allocated with the SDK
is_InitCamera	Hardware initialization
is_LoadParameters	Load and use the camera parameters
is_SaveParameters	Save the current camera parameters
is_SetCameraID	Sets a new camera ID

Table 3: Function list Initialization and termination

3.2 Image acquisition and memory management

Function list

is_AllocImageMem	Allocates image memory
is_CaptureVideo	Acquires live video
is_ConvertImage	Converts the Raw *Bayer* input picture to the desired format
is_CopyImageMem	Copies image to memory as defined by programmer
is_CopyImageMemLines	Copies single lines to memory as defined by programmer
is_FreeImageMem	Frees allocated image memory
is_FreezeVideo	Acquires image and writes to destined frame buffer
is_GetActiveImageMem	Returns number and address of active image memory
is_GetBusSpeed	Returns whether the camera is connected to a USB 2.0 host controller.
is_GetImageHistogram	Calculates the histogram of the given picture
is_GetImageMem	Returns start pointer to image memory
is_GetImageMemPitch	Returns line offset (n) to (n+1)
is_HasVideoStarted	Has the image acquisition started?
is_InquireImageMem	Returns image memory's properties
is_IsVideoFinish	Has the image acquisition finished?
is_SaveImageMem	Save image memory as bitmap
is_SetAllocatedImageMem	User makes the memory area available for the image recording
is_SetBayerConversion	Selects *Bayer* algorithm

is_SetImageMem	Activates an image memory
is_SetTestImage	Activates test images
is_StopLiveVideo	Terminates the recording (continuous or single frame)

Table 4: Function list image acquisition and memory management

3.3 Selection of operating modes and return of properties

Function list

is_CameraStatus	Standby; gets event counter and counter value
is_GetAutoInfo	Returns status information of the auto functionality
is_GetCameraList	Gets information about the connected cameras
is_GetCameraType	Gets type of camera (e.g. uEye USB)
is_GetColorDepth	Gets current colour mode from VGA card
is_GetDLLVersion	Returns the version of ueye_api.dll
is_GetError	Calls error message
is_GetExposureRange	Determines the exposure range
is_GetFramesPerSecond	Return current frame rate in live mode
is_GetFrameTimeRange	Determines the range of the frame rate
is_GetNumberOfCameras	Detects the number of attached cameras
is_GetOsVersion	Calls operating system type
is_GetPixelClockRange	Returns the adjustable range for the pixel clock
is_GetUsedbandwidth	Sum of the current pixel clocks
is_GetVsyncCount	Output of VSYNC counter
is_GetWhiteBalanceMultipliers	Readout the current white balance parameters
is_LoadBadPixelCorrectionTable	Load a user defined hot pixel list from a file
is_PrepareStealVideo	Sets the steal mode
is_ResetToDefault	Reset the camera parameters to default values
is_SaveBadPixelCorrectionTable	Stores the current, user defined hot pixel list
is_SetAOI	Set size and position of an AOI
is_SetAutoParameter	Activates/deactivates Gain/Shutter/Whitebalance auto functionality
is_SetBadPixelCorrection	Activate, deactivate and parameterise hot pixel correction
is_SetBadPixelCorrectionTable	Pass a user defined hot pixel list to the SDK
is_SetBinning	Controls the binning mode
is_SetBlCompensation	Activates/deactivates the black level compensation
is_SetBrightness	Sets brightness (digital reworking)
is_SetColorCorrection	Sets colour correction

is_SetColorMode	Selects colour mode
is_SetContrast	Sets contrast (digital reworking)
is_SetConvertParam	Sets the conversion parameters for the Raw *Bayer* image
is_SetDisplayMode	Selects image display mode
is_SetEdgeEnhancement	Sets edge filter
is_SetErrorReport	Activates or deactivates error output
is_SetExposureTime	Set the exposure time
is_SetFrameRate	Set the frame rate
is_SetGainBoost	Activates/deactivates the additional hardware gain
is_SetGamma	Set the gamma value (digital reworking)
is_SetGlobalShutter	Activates or deactivates the global start shutter
is_SetHardwareGain	Adjusting the hardware gain
is_SetHardwareGamma	Activates/deactivates the gamma control of the camera
is_SetHWGainFactor	Controlling of the camera amplifiers
is_SetHwnd	Controlling of the camera gain
is_SetImageAOI	Sets image position and image size
is_SetImagePos	Sets image position within image window
is_SetImageSize	Sets the size of the image
is_SetLED	Switch LED on/off
is_SetPixelClock	Set pixel clock
is_SetRopEffect	Sets real time ROP effects
is_SetSaturation	Sets the software image saturation
is_SetSubSampling	Controls the subsampling mode
is_SetWhiteBalance	Activate white balance
is_SetWhiteBalanceMultipliers	Set the white balance parameters

Table 5: Function list selection of operating modes and return of properties

3.4 Double and multi buffering

Function list

is_AddToSequence	Records image memory in sequence list
is_ClearSequence	Delete complete sequence list
is_GetActSeqBuf	Determines the image memory which is currently being used for the sequence.
is_LockSeqBuf	Protects image memory of the sequence from being overwritten.
is_UnlockSeqBuf	Allows image memory of the sequence to be overwritten.

Table 6: Function list double and multi buffering

3.5 Reading from and writing to EEPROM

Function list

is_GetCameraInfo	Reads pre-programmed manufacturer's information
	Revision information of the individual uEye components
is_GetSensorInfo	Readout the sensor information
is_ReadEEPROM	Reads own data from EEPROM
is_WriteEEPROM	Writes own data to EEPROM

Table 7: Function list reading from and writing to EEPROM

3.6 Saving and loading images

Function list

is_LoadImage	Load bitmap file in the current image memory
is_LoadImageMem	Loads an image from a file
is_SaveImage	Saves video image as a bitmap (BMP)
is_SaveImageEx	Saves an video image to a file
is_SaveImageMem	Saves image memory as a bitmap (BMP)
is_SaveImageMemEx	Saves an image to a file

Table 8: Function list saving and loading images

3.7 Image output

Function list

is_RenderBitmap	Displays images in a window
is_SetDisplayPos	Enables the offset for the image output
is_UpdateDisplay	Displays refresh with DirectDraw

Table 9: Function list Image output

3.8 Supplementary DirectDraw functions

Function list

is_DisableDDOverlay	Deactivates the overlay mode
is_EnableDDOverlay	Activates the live overlay mode
is_GetDC	Retrieves the device context handle's overlay memory
is_GetDDOvlSurface	Returns pointer to the DirectDraw surface
is_HideDDOverlay	Fades out the overlay
is_LockDDMem	Enables VGA card to access the back buffer
is_LockDDOverlayMem	Enables access to overlay memory
is_ReleaseDC	Releases the device context handle's overlay memory
is_SetDDUpdateTime	Timer interval for update cycle
is_SetKeyColor	Sets the keying colour for the overlay display
is_ShowDDOverlay	Fades on the overlay
is_StealVideo	Steals an image from a DirectDraw live mode and puts this down into the image memory in the RAM
is_UnlockDDMem	Disables VGA card to access the back buffer
is_UnlockDDOverlayMem	Disables access to overlay memory

Table 10: Function list supplementary DirectDraw functions

3.9 Event Handling

Function list

is_DisableEvent	Lock the event objects
is_EnableEvent	Release the event objects
is_EnableMessage	Activating/deactivating the windows messages
is_ExitEvent	Quit the event handle
is_InitEvent	Setup the event handle
is_EnableAutoExit	Camera resources are released automatically when taking the USB cable off

Table 11: Function list event handling

Events with single trigger recording

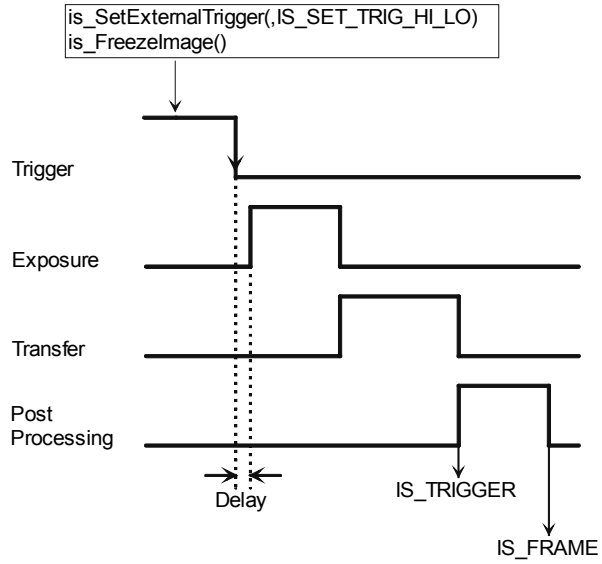

Figure 5: Events with single trigger recording

Events in live mode (sequence of 3 images)

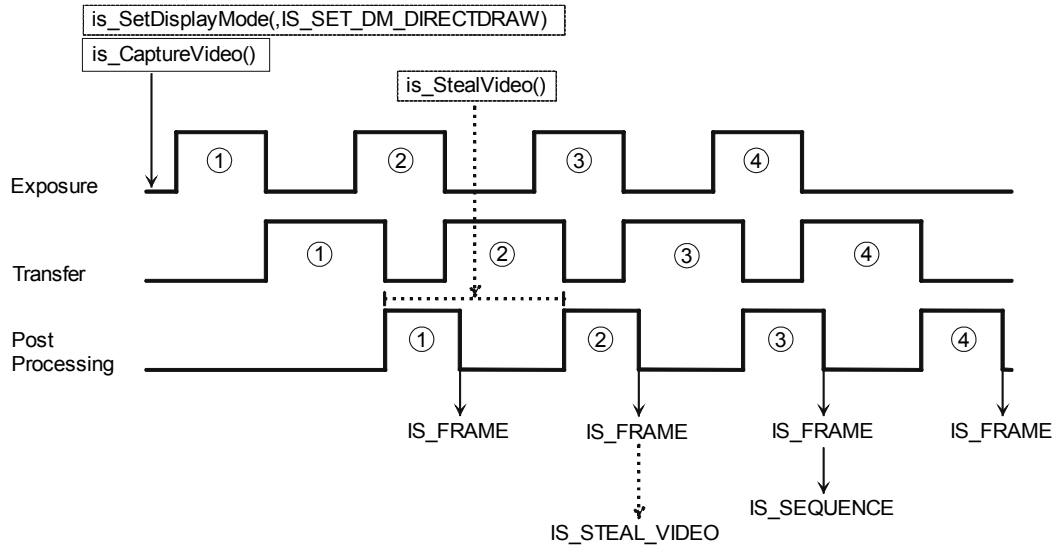

Figure 6: Events in live mode

Events im Memory mode

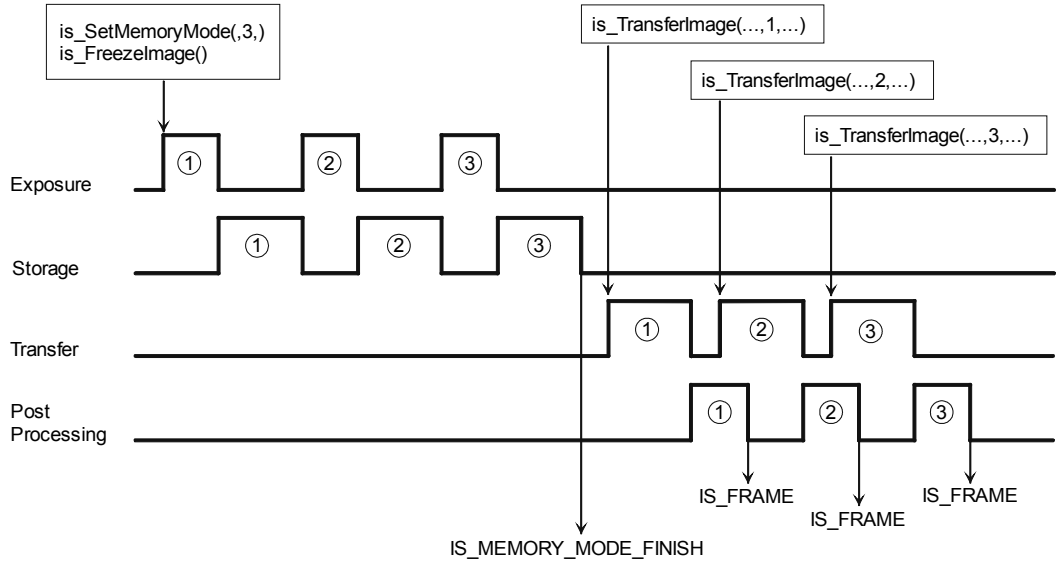

Figure 7: Events in Memory mode

3.10 Control of input / outputs

Function list

is_ForceTrigger	Release a hardware trigger.
is_GetGlobalFlashDelays	Determine delay and duration of the flash output for cameras with rolling shutter.
is_SetExternalTrigger	Activate external trigger input or the static input
is_SetFlashDelay	Set delay and duration of the flash output
is_SetFlashStrobe	Set the flash / strobe output or the static output
is_SetIO	Set the additional digital outputs
is_SetIOMask	Set direction of the digital in-/outputs
is_SetTriggerDelay	Specify a delay time of a trigger signal

Table 12: Function list control of input / outputs

3.11 I2C functions (uEyeLE cameras only)

Function list

is_ReadI2C (nur uEyeLE)	Read data over the I2C-bus
is_WriteI2C (nur uEyeLE)	Write data over the I2C-bus

Table 13: Function list I2C functions

3.12 Gigabit Ethernet functions (uEye Gigabit Ethernet cameras only)

Function list

is_GetEthDeviceInfo	Reads information about the connected cameras.
is_SetAutoCfgIpSetup	Presets the attributes for the IP auto configuration of a network adapter.
is_SetPacketFilter	Sets the packet filter for a network adapter.
is_SetPersistentIpCfg	Sets the attributes of the persistent IP configuration of a connected camera.
is_SetStarterFirmware	Updates the starter firmware of a connected camera.

Table 14: Function list Gigabit Ethernet functions

3.13 Memory handling

Function list

is_GetLastMemorySequence	Supplies ID of the last recorded sequence on the memory board
is_GetMemorySequenceWindow	Supplies window size for a specified memory board sequence
is_GetNumberOfMemoryImages	Supplies number of valid images within the specified sequence ID that are located in the camera memory
is_IsMemoryBoardConnected	Check whether the optional memory board is available
is_MemoryFreezeVideo	Record individual image via the memory board
is_ResetMemory	Reset the storage of the memory board
is_SetMemoryMode	Activate the optional memory board
is_TransferImage	Read in 1 image from the camera memory
is_TransferMemorySequence	Read in several images from the camera memory in a SDK sequence

Table 15: Function list memory handling

There is the possibility of new recording variations in combination with the memory expansion module for the uEye camera family. New SDK functions have been provided for control and the functionality of existing functions has been expanded.

Designations for the two recording modes depend on whether a trigger signal ends or starts the recording

- Pre-Trigger and
- Post-Trigger Modus.

Pre-Trigger

The memory board records continuous images in this mode. When the trigger is released, the recording is terminated and the *n* last images are available in the camera memory.

- Preparation
  The function *is_SetExternalTrigger* activates the trigger input of the camera. The camera is prepared for memory saving with the function *is_SetMemoryMode*.

The order of the functions *is_SetMemoryMode* and *is_SetExternalTrigger* is random.
If the trigger is deactivated with *is_SetExternalTrigger(hCamera, IS_SET_TRIG_OFF)* the recording is terminated directly upon invoking *is_StopLiveVideo*.

This function is provided with a figure as parameter which describes the number of images in a circulating memory which is overwritten in cycles until the trigger occurs.

The maximum possible number of images that can be kept in the memory depends on the set frame size which is why this must not be changed after activating the memory mode.

- Recording
  Recording is started by selecting *is_CaptureVideo*. *n* images are written to memory and the oldest is overwritten respectively upon arrival of new images.
  Upon selecting *is_StopLiveVideo* with an optional *timeout* value, recording of images and storage to camera memory continues until a trigger signal is registered. An image currently being recorded is written to end. After that, the last n recorded images can be selected from memory. The recorded sequence has been assigned a clear sequence ID with which the images can be indexed. This sequence ID can be selected with the command *is_GetLastMemorySequence* upon the end of recording.
  An error is sent if the period specified by *timeout* comes to an end before the trigger signal has been activated. Any images recorded in this sequence are thrown out and the sequence data is invalid. *is_GetLastSequence()* sends back 0 in this case (0 is not a valid sequence ID).
- Sample code:

```
int nNumberOfImages = 5, nSequence = 0;

is_SetExternalTrigger(hCamera, IS_TRIG_HI_LO);

//wenn die Speicherung so vieler Bilder möglich ist,
if (is_SetMemoryMode(hCamera, nNumberOfImages, 0) == IS_SUCCESS)
{
      // starte Bildaufnahme
      is_CaptureVideo(hCamera, IS_WAIT);

      // warten auf Triggersignal
      is_StopLiveVideo(hCamera, IS_WAIT);

      //Sequenz ist gültig?
is_GetLastMemorySequence(hCamera, &nSequence);
      if (nSequence != 0)
            is_TransferImage(hCamera, 0, nSequence, 3, 0);
      ...
}
```

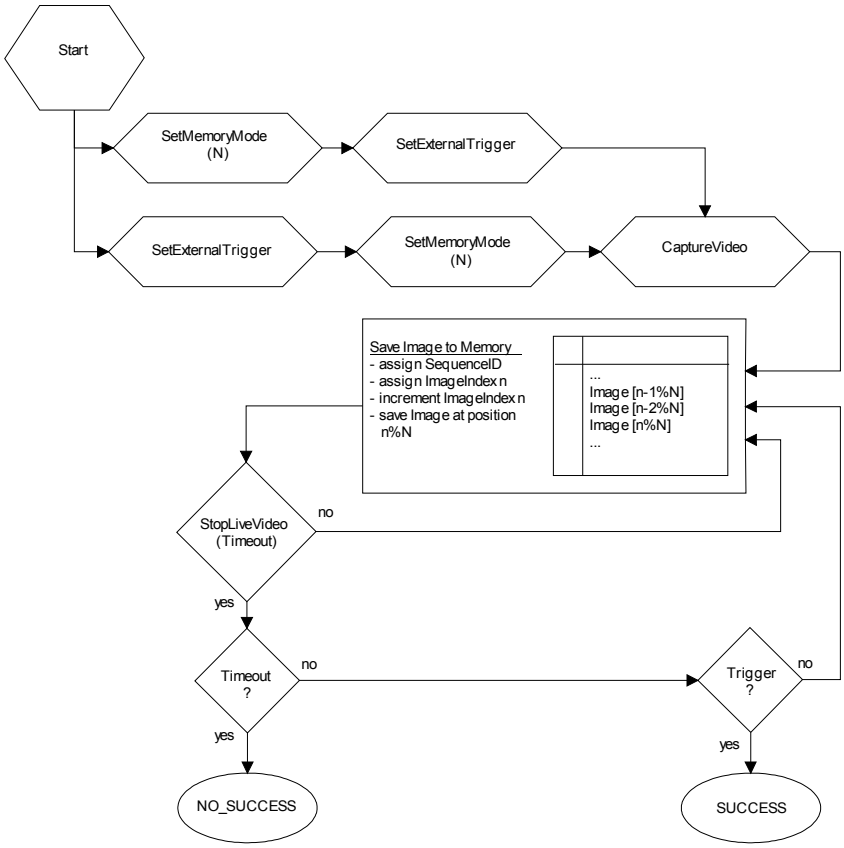

Figure 8: Pre-Trigger Mode

Post-Trigger

In this mode, image recording only commences after the trigger signal has been registered.

- Preparation
  The trigger input of the camera is activated with the function *is_SetExternalTrigger*.
  The camera is prepared for memory mode with the function *is_SetMemoryMode*. The time between two recordings is specified here along with the number of images that are to be recorded. The maximum number of possible images that can be stored to memory depends on the setting for image size which is why this must no longer be changed after activating memory mode.

The order of the functions *is_SetMemoryMode* and *is_SetExternalTrigger* is random. If the trigger is deactivated with *is_SetExternalTrigger(hCamera, IS_SET_TRIG_OFF)* the recording is started directly upon invoking *is_FreezeVideo()*.

- Recording
  This mode is activated by selecting *is_FreezeVideo*. An optional *timeout* value specifies how long to wait for the trigger signal. The camera is in standby as long as this signal has not arrived. Recording starts upon arrival of a trigger signal and the number of images specified by *is_SetMemoryMode* is recorded. When this number of images has been recorded, a clear sequence ID is issued with which these images can be indexed at a later date. If, however, the *timeout* ends before all images have been recorded, the complete sequence is invalid (sequence ID = 0) and an error is signalled. The ID of a sequence can be selected after all images have been recorded with the command *is_GetLastMemorySequence*.
- Sample code

```
int nNumberOfImages = 5, nSequence = 0;

is_SetExternalTrigger(hCamera, IS_TRIG_HI_LO);

//wenn die Speicherung so vieler Bilder möglich ist,
if (is_SetMemoryMode(hCamera, nNumberOfImages, 100) == IS_SUCCESS)
{
       // starte Bildaufnahme und warten auf Triggersignal
is_FreezeVideo(hCamera, IS_WAIT);

       //Sequenz ist gültig?
       is_GetLastMemorySequence(hCamera, &nSequence);
if (nSequence != 0)
             is_TransferImage(hCamera, 0, nSequence, 1, 0);
}
```


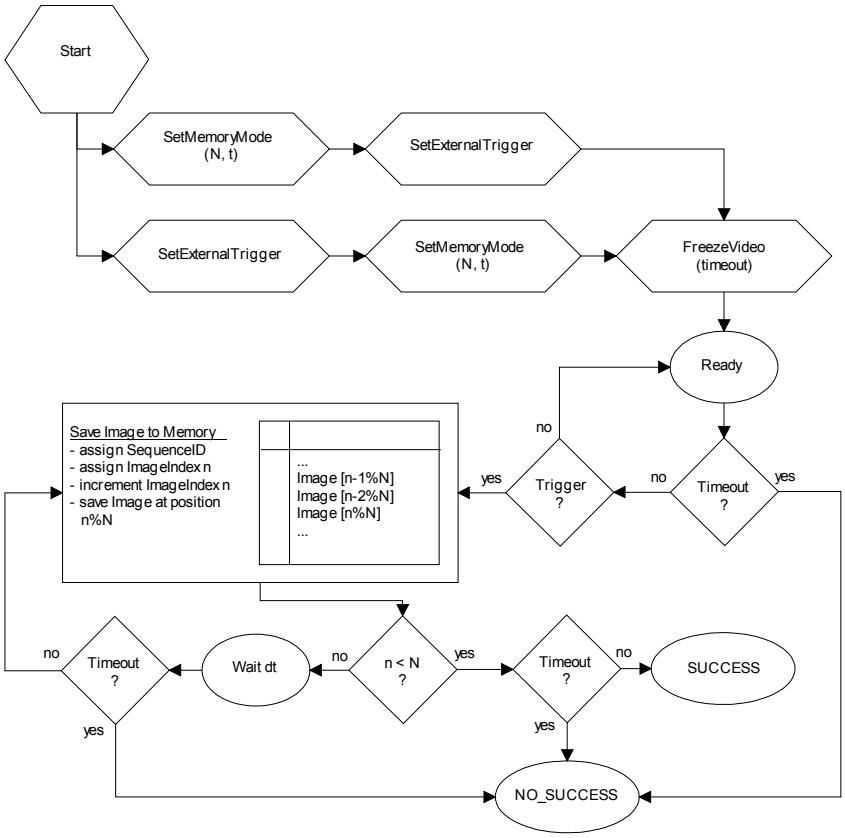

Figure 9: Post-Trigger mode

Several sequences

It is quite possible to record several sequences without the images recorded between two sequences having to be read out of the camera memory.
However, there can be no guarantee in the case of older sequences that they are still retained in the camera memory.
The images of a new sequence are preferably stored in free camera memory. If there is not enough space available, at least parts of old sequences are overwritten.
As a result, possibly only a few images may be left from an old sequence.
The function *is_TransferImage*, as a parameter, expects the sequence ID of a valid sequence and the number of the image within this sequence to be transferred. If the image or the complete sequence has since become invalid, the function sends back a corresponding error.

Example:

A sequence (A) with 5 images has already been recorded. A second sequence (B) does not fit completely in the memory without having to overwrite.
If a third sequence is now recorded with 3 images that together are larger than the available memory, writing to memory starts again from the beginning if an image no longer fits completely into the remaining available memory (individual images, therefore, are not wrapped; a small remainder is left unused).
Any images left in these memory areas are overwritten. In the example, image C3 extends into the memory area of A3, making the complete image A3 invalid. Sequence A now only consists of 2 images.

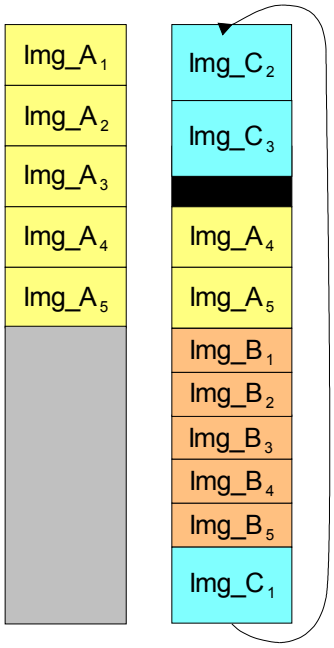

Figure 10: Appending storage mode

Timing in Memoryboard operation

The data of a recorded image is sent in normal operation mode direct from the sensor to the computer via the USB interface.
The situation is different if using the memory board. In this case, the sensor is in trigger mode. As soon as the trigger signal is received, one image or a sequence of images is recorded and stored on the memory board. After the last image has been stored, transmission of the stored images to the computer can take place via the USB interface. No new images can be recorded and stored whilst the data is being transferred.

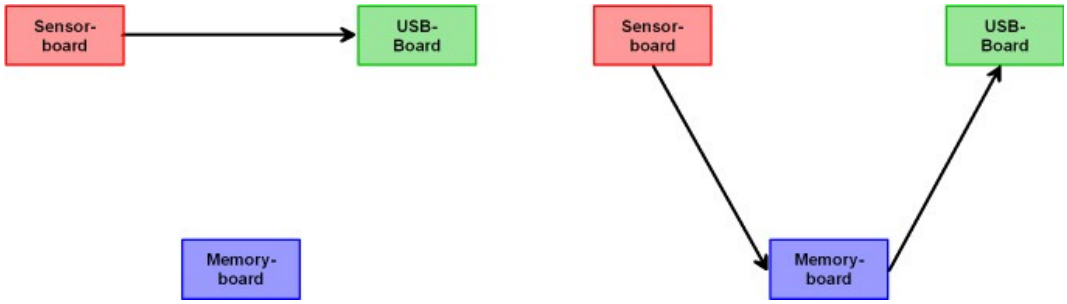

Figure 11: Functional principle of the memory board

The following diagram shows the chronological procedures when using the memory board.

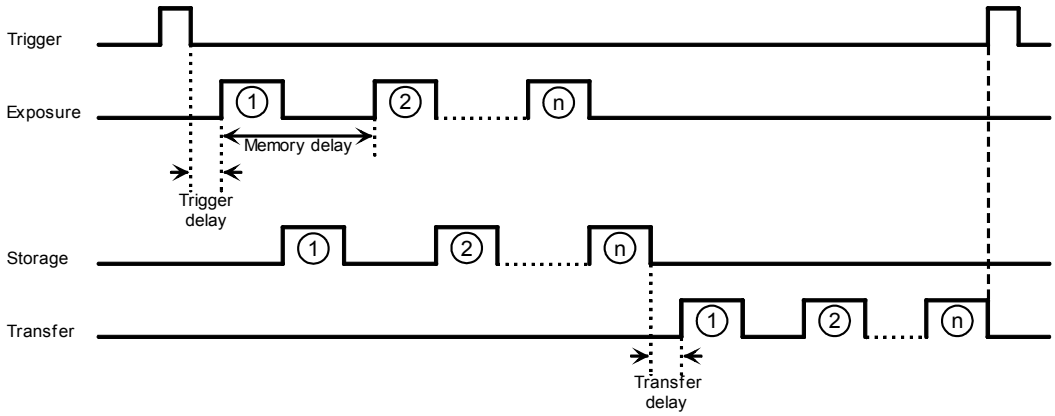

Figure 12: Timing diagram - memory board operation

3.14 Validity of functions

Some of the functions are responsible for all display modes, whereas others are only for DirectDraw display modes. In the following tables the validity of each function is listed.

Function	Bitmap	DD-BackBuffer Surface	DD-Overlay Surface
is_AddToSequence	☒		
is_AllocImageMem	☒	✹	✹
is_CaptureVideo	☒	☒	☒
is_ClearSequence	☒		
is_ConvertImage	☒		
is_CopyImageMem	☒	✹	✹
is_CopyImageMemLines	☒	✹	✹
is_DisableDDOverlay		☒	
is_EnableDDOverlay		☒	
is_FreeImageMem	☒	✹	✹
is_FreezeVideo	☒	☒	☒
is_GetActiveImageMem	☒		
is_GetActSeqBuf	☒		
is_GetDC		☒	☒
is_GetDDOvlSurface		☒	
is_GetImageHistogram	☒	✹	✹
is_GetImageMem	☒	☒	☒
is_GetImageMemPitch	☒	✹	✹
is_GetVsyncCount	☒	☒	☒
is_HasVideoStarted	☒	☒	☒
is_HideDDOverlay		☒	
is_InquireImageMem	☒	✹	✹
is_IsVideoFinish	☒	☒	☒
is_LoadImage	☒		
is_LoadImageMem	☒		
is_LockDDMem		☒	☒
is_LockDDOverlayMem		☒	
is_LockSeqBuf	☒		
is_PrepareStealVideo	☒	☒	☒
is_ReleaseDC		☒	☒
is_RenderBitmap	☒		
is_SaveImage	☒	☒	☒

is_SaveImageEx	☒	☒	☒
is_SaveImageMem	☒	✹	✹
is_SaveImageMemEx	☒	✹	✹
is_SetAllocatedImageMem	☒	✹	✹
is_SetBayerConversion	☒	☒	☒
is_SetBinning	☒	☒	☒
is_SetColorMode	☒	☒	☒
is_SetConvertParam	☒		
is_SetDisplayMode	☒	☒	☒
is_SetDisplayPos	☒		
is_SetHwnd		☒	☒
is_SetImageMem	☒	✹	✹
is_SetImagePos	☒	☒	☒
is_SetImageSize	☒	☒	☒
is_SetKeyColor			☒
is_SetRopEffect	☒	☒	☒
is_SetSaturation			☒
is_SetSubSampling	☒	☒	☒
is_ShowDDOverlay		☒	
is_StealVideo		☒	☒
is_StopLiveVideo	☒	☒	☒
is_UnlockDDMem		☒	☒
is_UnlockDDOverlayMem		☒	
is_UnlockSeqBuf	☒		
is_UpdateDisplay		☒	☒

✹ *only in steal mode; memory in DD modes not necessary*

Table 16: Validity of functions

DirectDraw is not supported under LINUX.

3.15 Not supported functions

The functions specified in the following are special functions for the FALCON frame grabber family and are not supported by the uEye camera family.

is_GetCurrentField()
is_GetIRQ()
is_GetPciSlot()
is_OvlSurfaceOffWhileMove()
is_ScaleDDOverlay()
is_SetAGC()
is_SetCaptureMode()
is_SetDecimationMode()
is_SetDisplaySize()
is_SetHorFilter()
is_SetHue()
is_SetIOMask()
is_SetKeyOffset()
is_SetParentHwnd()
is_SetPassthrough()
is_SetRenderMode()
is_SetSync()
is_SetSyncLevel()
is_SetToggleMode()
is_SetUpdateMode()
is_SetVertFilter()
is_SetVideoCrossbar()
is_SetVideoInput()
is_SetVideoMode()
is_SetVideoSize()
is_ShowColorBars()
is_Watchdog()
is_WatchdogTime()

Table 17: Not supported functions

4 Description of the Functions

To aid the integration of the uEye cameras into your own programs, the functions from the driver library, which are shipped with the camera, are described in this section. The functions are sorted alphabetically and structured as follows:

<Name of the function>

Syntax:

Function prototype from the header file *ueye.h*

Description:

Function description with cross reference to affected functions

Parameters:

Description of the function parameters with range of values

Return value:

Description and range of return value. If function returns *IS_NO_SUCCESS (-1)*, the error can be called with the function *is_GetError()*.

The source code of the example program *UEYEDEMO.EXE*, which uses the uEye library, is delivered with the camera on the installations-CD. In the source code the initialization of and access to the camera is shown as well as the various operating modes.

4.1 is_AddToSequence

Syntax

INT is_AddToSequence (HIDS hf, char* pcImgMem, INT nID)

Description

is_AddToSequence() inserts image memory into the image memory list, which is to be used for ring buffering. The image memory has to be allocated with *is_AllocImageMem()*. All image memory which is used for ring buffering must have been allocated the same colour depth (i.e. bits per pixel). The number of image memories for a sequence (*nID*) is limited to the integer value range.

Parameters

hf	Camera handle
pcMem	Pointer to image memory
nID	ID of image memory

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.2 is_AllocImageMem

Syntax

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

Description

is_AllocImageMem() allocates image memory for an image with *width*, *width* and *height*, *height* and *colour depth bitspixel*. Memory size is at least:

size = [width * ((bitspixel + 1) / 8) + adjust] * height

adjust see below

Line increments are calculated with:

line = width * [(bitspixel + 1) / 8]

lineinc = line + adjust.

adjust = 0, if *line* is divisible by 4 without rest

adjust = 4 - rest(line / 4), if *line* is not divisible by 4 without rest

The line increment can be read with the *is_GetImgMemPitch()* function.

The start address in the image memory is returned with *ppcImgMem*.

pid contains an identification number of the allocated memory. A newly activated memory location is not directly activated. In other words, images are not directly digitized to this new memory location. Before this can happen, the new memory location has to be activated with *is_SetImageMem()*. After *is_SetImageMem()* an *is_SetImageSize()* must follow so that the image conditions can be transferred to the newly activated memory location. The returned pointer has to be saved and may be reused, as it is required for all further *ImageMem* functions! The freeing of the memory is achieved with *is_FreeImageMem()*.

In the DirectDraw modes, the allocation of an image memory is not required!

Most operating Systems begin to swap older parts of the main memory to the hard disk, if there is a lack of free physical main memory. Because of this, image acquisition could become slower, if there was more Image memory allocated than can be kept in the physical main memory at the same time.

Parameters

hf	Camera handle
width	Width of image
height	Height of image
bitspixel	Colour depth of image (bits per pixel)
ppcImgMem	Enthält dann den Zeiger auf den Speicheranfang
pid	Enthält dann die ID für diesen Speicher

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.3 is_CameraStatus

Syntax

ULONG is_CameraStatus (HIDS hf, INT nInfo, ULONG ulValue)

Description

is_CameraStatus() returns various status information and settings. Some of the settings can be changed with *is_CameraStatus()*.

Parameters

hf	Camera handle
nInfo	
IS_FIFO_OVR_CNT	Number of FIFO Overruns. Will be increased each time image data is lost due to USB bus overload.
IS_SEQUENCE_CNT	Is set to zero with *is_CaptureVideo()*. With each change of the sequence Buffers (image counter) the increase takes place by 1.
IS_SEQUENCE_SIZE	Number of sequence buffer (read only)
IS_EXT_TRIGGER_EVENT_CNT	Trigger interrupt counter
IS_CAMERA_REVISION	Returns the hardware revision of the camera (read only).
IS_WAIT_TIMEOUT	Time out for hardware trigger (on use with IS_WAIT or IS_DONT_WAIT), 1ms steps.
IS_TRIGGER_MISSED	Number of not processed trigger signals. It is set to 0 after each call.
IS_LAST_CAPTURE_ERROR	Error during capturing (read only)
IS_PARAMETER_SET_1	(read only) Return value: TRUE parameter set 1 available FALSE parameter set 1 not available
IS_PARAMETER_SET_2	(read only) Return value: TRUE parameter set 2 available FALSE parameter set 2 not available
IS_STANDBY	Return value: TRUE Switch camera to standby mode FALSE Switch camera to freerun mode
IS_STANDBY_SUPPORTED	(read only) Return value: TRUE camera supports standby mode FALSE camera does not support standby mode
ulValue	
IS_GET_STATUS	Read the parameter value for *nInfo*.

Return value

IS_SUCCESS, IS_NO_SUCCESS or current value of uIValue = IS_GET_STATUS

After the event *IS_SET_TRANSFER_FAILED* or the message *IS_TRANSFER_FAILED* the occured error can be read out with *IS_LAST_CAPTURE_ERROR*. The following return values are possible:

- *IS_SUCCESS*
- *IS_TRANSFER_ERROR*
  Capturing was cancelled.
- *IS_TIMED_OUT*
  The maximally permissible capturing time was exceeded.
- *IS_NO_ACTIVE_IMAGE_MEM*
  There is no target image buffer, or the existing target image buffers are locked.
- *IS_SEQUENCE_BUF_ALREADY_LOCKED*
  The memory image is not describable.
- *IS_COULD_NOT_CONVERT*
  The current picture could not be converted.

4.4 is_CaptureVideo

Syntax

INT is_CaptureVideo (HIDS hf, INT Wait)

Description

is_CaptureVideo() digitizes video images in real time and transfers the images to the previously allocated image memory. Alternatively if you are using DirectDraw the images can be transferred to the graphics board. The image acquisition (DIB Mode) takes place in the memory which has been set by *is_SetImageMem()* and *is_AllocImageMem()*. *Is_GetImageMem()* determines exactly where the start address in memory is.

In case of ring buffering, then image acquisition loops endlessly through all image memories added to the sequence. Locked sequence buffers (*is_LockSeqBuf()*) are skipped. When the last non-locked sequence buffer has been filled, the sequence event/the sequence message is triggered. Recording always begins with the first element of the sequence.

After Activation of the memory mode with *is_SetMemoryMode()* or *is_MemoryFreezeVideo()* the images taken with *is_CaptureVideo()* are stored in the camera memory. In order to get image acquisition without memory mode, the memory mode must be switched off again with the function *is_SetMemoryMode(IS_MEMORY_MODE_DISABLE, 0)* (see *4.124 is_SetMemoryMode*).

Parameters

hf	Camera handle
Wait	
IS_DONT_WAIT	This function synchronizes the image acquisition of the V-SYNC, but returns immediately.
IS_WAIT	This function synchronizes the image acquisition of the V-SYNC and only then does return (i.e. waits until image acquisition begins)
10 < Wait < 32768	Wait time in 10 ms steps. A maximum of 327.68 seconds (this is approx. 5 minutes and 20 seconds) can be waited. For 1 < Wait < 10 Wait becomes equal to 10. (Exp.: Wait = 100 ⇨ wait 1 sec.)

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.5 is_ClearSequence

Syntax

INT is_ClearSequence (HIDS hf)

Description

is_ClearSequence() deletes all image memory from the sequence list that was inserted with *is_AddToSequence()*. After *is_ClearSequence()* no more image memory is active. To make a certain part of the image memory active, *is_SetImageMem()* and *is_SetImageSize()* have to be executed.

Parameters

hf	Camera handle

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.6 is_ConvertImage

Syntax

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

Description

Converts the Raw *Bayer* input picture to the desired format. If the pointer of the destination image data is NULL a new memory is allocated.

Parameters

hf	Camera handle
pcSource	Pointer of memory of the input picture
nIDSource	Id of memory of the input picture
ppcDest	Pointer of memory of the output picture
nIDDest	Id of memory of the output picture

Return value

IS_SUCCESS, IS_NO_SUCCESS

Example:

Convert a Raw *Bayer* picture in RGB24. The memory will be allocated automatically:

```
// Create a Raw Bayer test picture
char * pcSource;
INT nIDSource;
is_AllocImageMem (hf, 256, 256, 8, &pcSource, &nIDSource);
Int nX,nY,nBits,nPitch;
is_InquireImageMem (hf, pcSource, nIDSource, &nX, &nY, &nBits, &nPitch);
for (int j = 0;j< nY;j++)
for (int i = 0;i< nX;i++)
pcSource[i + j*nPitch] = i;

INT Gamma = 120;
double rgbGains[3];

rgbGains[0] = 1.0 ; // Red gain
rgbGains[1] = 3.0 ; // Green gain
rgbGains[2] = 1.0 ; // Blue gain

char* pcDest;  // Pointer to the data of the new allocated picture
INT nIDDest;  // id of the new allocated picture

INT nRet;

// Set the conversion parameters
nRet = is_SetConvertParam(hf, TRUE, IS_SET_BAYER_CV_BETTER, IS_SET_CM_RGB24, Gamma,
```

```
rgbGains);

// Convert the picture
if (nRet == IS_SUCCESS){
pcDest = NULL;
is_ConvertImage(hf, pcSource, nIDSource, &pcDest, &nIDDest, 0);
}

// Free the allocated memory
is_FreeImageMem (m_hCam, pcSource, nIDSource);
is_FreeImageMem (m_hCam, pcDest, nIDDest);
```


4.7 is_CopyImageMem

Syntax

INT is_CopyImageMem (HIDS hf, char* pcSource, INT nID, char* pcDest)

Description

is_CopyImageMem() copies the contents of the image memory, as described is *pcSource* and *nID* to the area in memory, which *pcDest* points to.

The user must make sure that the allocated memory *pcDest* is large enough to store the whole image (not just the area of interest) in the current format (bits per pixel)

Parameters

hf	Camera handle
pcSource	Pointer to image memory
nID	ID of this image memory
pcDest	Pointer to destination memory to which the image should be copied.

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.8 is_CopyImageMemLines

Syntax

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

Description

is_CopyImageMemLines() copies the contents of the image memory, as described is *pcSource* and *nID* to the area in memory, which *pcDest* points to. *nLines* lines are copied.

The user must make sure that the allocated memory *pcDest* is large enough to store the whole image (not just the area of interest) in the current format (bits per pixel).

Parameters

hf	Camera handle
pcSource	Pointer to image memory
nID	ID of this image memory
nLines	Number of lines which are copied
pcDest	Pointer to destination memory to which the image should be copied

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.9 is_DisableDDOverlay

Syntax

INT is_DisableDDOverlay (HIDS hf)

Description

When in DirectDraw BackBuffer mode, *is_DisableDDOverlay()* deactivates the overlay mode and releases the memory which is currently occupied by the overlay, which causes the overlay data to be deleted.

Parameters

hf	Camera handle

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.10 is_DisableEvent

Syntax

INT is_DisableEvent (HIDS hf, INT which)

Description

is_DisableEvent() blocks the event given here. The event (e.g. a frame) will generally still occur, but not trigger an event signal any more. After this function is called, the application does not notice the blocked events any more. A desired event can be reactivated with *is_EnableEvent()* if required. See also *4.57 is_InitEvent*.

Parameters

hf	Camera handle
which	ID of the event to release

See *4.57 is_InitEvent*.

Return value

IS_SUCCESS, IS_NO_SUCCESS

Example

See *4.57 is_InitEvent*.

4.11 is_EnableAutoExit

Syntax

INT is_EnableAutoExit (HIDS hf, INT nMode)

Description

is_EnableAutoExit() activates the automatic closing of the camera handle after a camera was removed during operation. With closing, all reserved memory by the SDK will be released.

Parameters

hf	Camera handle
nMode	
IS_ENABLE_AUTO_EXIT	Activates automatic closing
IS_DISABLE_AUTO_EXIT	Deactivates automatic closing
IS_GET_AUTO_EXIT_ENABLED	Read the status

Return value

Current settings when called with *IS_GET_AUTO_EXIT_ENABLED*, else *IS_SUCCESS,* or *IS_NO_SUCCESS*

4.12 is_EnableDDOverlay

Syntax

INT is_EnableDDOverlay (HIDS hf)

Description

When in DirectDraw BackBuffer mode *is_EnableDDOverlay()* activates the live overlay mode. In BackBuffer mode three non-visible image buffers are used: Back buffer, overlay buffer and mix buffer. The video image is digitized in the back buffer. The graphics data can be written in the overlay buffer and thus the overlay data is overlaid on the video image. The mix buffer is then copied to the visible area of the VGA card. The size of the three buffers is: video_x * video_y * colour depth (in bytes per pixel). The driver tries to allocate the buffer directly on the VGA card, (making best use of the high speed image transfer that the VGA card can offer) when mixing the three buffers. If the buffers cannot be allocated in the VGA card, they will be stored in system memory. The image transfer from the system memory is slower and, depending on the graphics card, sometimes not at all possible. The overlay is not always faded on. It has to be made visible with *is_ShowDDOverlay()* (see *4.137 is_ShowDDOverlay*). As its key colour, the overlay uses black, an thus an overlay cannot contain any black colour.

Parameters

hf	Camera handle

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.13 is_EnableEvent

Syntax

INT is_EnableEvent (HIDS hf, INT which)

Description

Release of the equipped event object. After the release, the event signalling of the current event object is allowed. See also *4.57 is_InitEvent*.

Parameters

hf	Camera handle
which	ID of the event to initialize

See *4.57 is_InitEvent*

Return value

IS_SUCCESS, IS_NO_SUCCESS

Example

See *4.57 is_InitEvent*.

4.14 is_EnableHdr

Syntax

INT is_EnableHdr (HIDS hf, INT Enable)

Description

Some sensors support HDR mode (High Dynamic Range). HDR mode can be activated/deactivated with *is_EnableHdr()*. The knee points must be set via *is_SetHdrKneepoints()*.

Currently, only the UI-122X-X and UI-522X-X models support HDR.
For cameras of types UI-122X-C and UI-522X-C, the RGB gains must be set to the same values to ensure accurate colour rendition in HDR mode.

Parameters

hf	Camera handle
Enable	
IS_ENABLE_HDR	Activates HDR mode
IS_DISABLE_HDR	Deactivates HDR mode.

Return value

IS_SUCCESS or *IS_NO_SUCCESS* for supported sensor types
IS_NOT_SUPPORTED (*Enable*) or *IS_SUCCESS* (*Disable)* for unsupported sensor types

4.15 is_EnableMessage

Syntax

INT is_EnableMessage (HIDS hf, INT which, HWND hWnd)

Description

With *is_EnableMessage()* messages can be activated or deactivated (*hWnd* = NULL), which are sent when occurring a certain event to the user program. The message is build up as follows:

Msg: IS_UEYE_MESSAGE
wParam: Arrived event (see Table)
lParam: Camera handle that belongs to the message

Parameters

hf	Camera handle
which	ID of the message to be activated/deactivated
IS_FRAME	A new image is available.
IS_SEQUENCE	The sequence was gone through.
IS_STEAL_VIDEO	An image detracted from the overlay is available.
IS_TRANSFER_FAILED	An Error occurred during data transfer (frame rate, pixel clock too high)
IS_TRIGGER	An image, which recording was released by a trigger, was completely received. This is the earliest time for a new recording. The picture must pass the post processing of the driver and is after *IS_FRAME* available for the processing.
IS_MEMORY_MODE_FINISH	Recording images to the optional memory module has been terminated.
IS_DEVICE_REMOVED	A camera, opened with *is_InitCamera()* has been removed.
IS_DEVICE_RECONNECTED	A camera opened with *is_InitCamera()* and thereafter removed was reconnected.
IS_NEW_DEVICE	A new camera was attached. Independent of the device handle (*hf* will be ignored).
IS_DEVICE_REMOVAL	A camera was removed. Independent of the device handle (*hf* will be ignored).
IS_WB_FINISHED	The automatic white balance control is finished.
IS_AUTOBRIGHTNESS_FINISHED	The automatic brightness control is finished (if *IS_SET_AUTO_BRIGHTNESS_ONCE* has been specified).
hWnd	Application window, which gets the message. (NULL deactivates the message, defined with which).

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.16 is_ExitCamera

Syntax

INT is_ExitCamera (HIDS hf)

Description

is_ExitCamera() cancels the active device handle *hf* and deallocates the data structures and memory areas currently associated with the uEye camera. The image memory which has been allocated by the user and which has not been released yet, is released with *is_ExitCamera*.

The uEye SDK is thread safe in general. The API function calls are all done in critical sections. Due to internal structures of DirectDraw we strongly recommend to call the following functions from only one thread to avoid unpredictable behaviour of your application.

- *is_InitCamera()*
- *is_SetDisplayMode()*
- *is_ExitCamera()*

Parameters

hf	Camera handle

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.17 is_ExitEvent

Syntax

INT is_ExitEvent (HIDS hf, INT which)

Description

Deletes set event object. After deleting it can not be activated with *is_EnableEvent()*.

Parameters

hf	Camera handle
which	ID of the event to release

See *4.57 is_InitEvent*

Return value

IS_SUCCESS, IS_NO_SUCCESS

Example

See *4.57 is_InitEvent*.

4.18 is_ForceTrigger

Syntax

INT is_ForceTrigger (HIDS hf)

Description

The function *is_ForceTrigger()* enables to force a trigger during a hardware trigger recording to take up a picture independently of a real trigger signal. This function can only be used, if the trigger recording was started with the parameter *IS_DONT_WAIT*.

See also *4.20 is_FreezeVideo* and *4.104 is_SetExternalTrigger*.

Parameters

hf	Camera handle

Return value

IS_SUCCESS, IS_NO_SUCCESS

Example:

Activate the trigger and wait 1 second for an external trigger. Force a recording with *is_ForceTrigger()* if no trigger was released.

```
HANDLE hEvent = CreateEvent(NULL, TRUE, FALSE, "");
if (hEvent != NULL)
{
is_InitEvent(hf, m_hEvent, IS_SET_EVENT_FRAME);
is_EnableEvent(hf, IS_SET_EVENT_FRAME);

is_SetExternalTrigger(hf, IS_SET_TRIG_HI_LO);
is_FreezeVideo(hf, IS_DONT_WAIT);
if (WaitForSingleObject(m_hEvent, 1000) != WAIT_OBJECT_0)
{
// Noch kein Trigger empfangen, also Bildaufnahme erzwingen
is_ForceTrigger(hf);
}

is_DisableEvent(hf, IS_SET_EVENT_FRAME);
is_ExitEvent(hf, IS_SET_EVENT_FRAME);
}
```


4.19 is_FreeImageMem

Syntax

INT is_FreeImageMem (HIDS hf, char* pcImgMem, INT id)

Description

is_FreeImageMem() deallocates previously allocated image memory. For *pcImgMem* one of the pointers from *is_AllocImgMem()* has to be used. All other pointers lead to an error message! The repeated handing over of the same pointers also leads to an error message!

hf	Camera handle
pcImgMem	Pointer to image memory
id	ID of this image memory

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.20 is_FreezeVideo

Syntax

INT is_FreezeVideo (HIDS hf, INT Wait)

Description

is_FreezeVideo() digitizes an image and transfers it to the active image memory. In DirectDraw mode the image is digitized in the DirectDraw buffer (either on the VGA card or in a back buffer).
If you are using ring buffering, the image is recorded to the next non-locked image buffer in the sequence. As soon as the last non-locked sequence buffer has been filled, the sequence event/the sequence message is triggered.
The picture recording takes place triggered, if the trigger mode were activated before with *is_SetExternalTrigger()*.

After Activation of the memory mode with *is_SetMemoryMode()* or *is_MemoryFreezeVideo()* the images taken with *is_FreezeVideo()* are stored in the camera memory. In order to get image acquisition without memory mode, the memory mode must be switched off again with the function *is_SetMemoryMode(IS_MEMORY_MODE_DISABLE, 0)* (see *4.124 is_SetMemoryMode*) .

Parameters

hf	Camera handle
Wait	
IS_WAIT	The function waits until an image is grabbed. If the fourfold frame time is exceeded, this is acknowledged with a time out.
IS_DONT_WAIT	The function returns straight away
10 <= Wait < 21474836	Wait time in 10 ms steps. A maximum of 214748.36 seconds (this is approx. 59 hours) can be waited. For 1 < Wait < 10 Wait becomes equal to 10. (Exp.: Wait = 100 ⇒ wait 1 sec.)

Return value

IS_SUCCESS, IS_NO_SUCCESS

Example

Activate trigger mode, set High-Active flash mode and record image.

```
is_SetExternalTrigger(hf, IS_SET_TRIG_SOFTWARE);
is_SetFlashStrobe(hf, IS_SET_FLASH_HI_ACTIVE);
is_FreezeVideo(hf, IS_WAIT);
```

4.21 is_GetActiveImageMem

Syntax

INT is_GetActiveImageMem (HIDS hf, char** ppcMem, INT* pnID)

Description

is_GetActiveImageMem() returns the pointer to the beginning and the ID number of the active memory. If DirectDraw mode is active and image memory has been allocated, this function returns the pointer and the ID of the image memory, which was activated with *is_SetImageMem()*. However, it should be noted that in DirectDraw mode, this memory is not used for digitizing. Also see *4.42 is_GetImageMem*.

Parameters

hf	Camera handle
ppcMem	Contains the pointer to the beginning of the image memory.
pnID	Contains the ID of the image memory.

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.22 is_GetActSeqBuf

Syntax

INT is_GetActSeqBuf (HIDS hf, INT* pnNum, char** ppcMem, char** ppcMemLast);

Description

With *is_GetActSeqBuf()* the image memory in which image acquisition (*ppcMem*) is currently taking place and the image memory which was last used for image acquisition (*ppcMemLast*) can be determined. This function is only available when the ring buffer is active. If image acquisition is started for a ring buffer, *is_GetActSeqBuf* returns 0 in *pnNum* as long as data is acquired to the first image memory of the sequence. And thus *pnNum* receives the number of the sequence image memory, in which image acquisition is currently taking place. The number is not the ID of the image memory which is provided from *is_AllocImageMem()*, but the running number in the sequence as defined in *is_AddToSequence()*.

Parameters

hf	Camera handle
pnNum	Contains the number of the image memory in which image acquisition is currently taking place. 0: image acquisition has not started in the first image memory 1...max: image acquisition in the sequence image memory N has started.
ppcMem	Contains the start address of the image memory in which the current image acquisition is taking place.
ppcMemLast	Contains the start address of the image memory, which was last used for image acquisition.

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.23 is_GetAutoInfo

Syntax

INT is_GetAutoInfo (HIDS hf, PUEYE_AUTO_INFO info)

Description

With the function *is_GetAutoInfo()* status information of auto functionality can be readout. The in-formation is available in the structure *UEYE_AUTO_INFO*.

The status information in the structure *UEYE_AUTO_INFO* is only valid if at least one auto functionality is activated.

UEYE_AUTO_INFO

INT	AutoAbility	0x01: AutoShutter possible (AC_SHUTTER) 0x02: AutoGain possible (AC_GAIN) 0x03: AutoGain and AutoShutter possible 0x04: AutoWhiteBalance possible (AC_WHITEBAL)
AUTO_BRIGHT_STATUS	sBrightCtrlStatus	See *AUTO_BRIGHT_STATUS*
AUTO_WB_STATUS	sWBCtrlStatus	See *AUTO_WB_STATUS*
DWORD	reserviert	Reserved for extensions

In the structure *UEYE_AUTO_INFO* the structures *AUTO_BRIGHT_STATUS* and *AUTO_WB_STATUS* are used.

AUTO_BRIGHT_STATUS

INT	curValue	Current average grey value (Ist)
INT	curError	Current control deviation (Error)
INT	curController	Current brightness control 0x01: AC_SHUTTER 0x02: AC_GAIN
INT	curCtrlStatus	Current control status 0x01: ACS_ADJUSTING 0x02: ACS_FINISHED 0x04: ACS_DISABLED

AUTO_WB_STATUS

INT	curController	Current white balance controller 0x08: AC_WB_RED_CHANNEL 0x10: AC_WB_GREEN_CHANNEL 0x20: AC_WB_BLUE_CHANNE
AUTO_WB_CHANNEL_STATUS	RedChannel	See *AUTO_WB_CHANNEL_STATUS*
AUTO_WB_CHANNEL_STATUS	GreenChannel	See *AUTO_WB_CHANNEL_STATUS*
AUTO_WB_CHANNEL_STATUS	BlueChannel	See *AUTO_WB_CHANNEL_STATUS*

In the following the structure *AUTO_WB_CHANNEL_STATUS* is described which is used in the structure *AUTO_WB_STATUS*.

AUTO_WB_CHANNEL_STATUS

INT	curValue	Current average grey value
INT	curError	Current control error
INT	curCtrlStatus	Current control status 0x01: ACS_ADJUSTING 0x02: ACS_FINISHED 0x04: ACS_DISABLED

Parameters

hf	Camera handle
info	Structure UEYE_AUTO_INFO (see above)

Return value

IS_SUCCESS, IS_NO_SUCCESS

Example

Readaout *Auto Info*:

```
UEYE_AUTO_INFO autoinfo;
Int ret = is_GetAutoInfo(m_hCam, &autoinfo);
```

4.24 is_GetBusSpeed

Syntax

INT is_GetBusSpeed (HIDS hf)

Description

The function *is_GetBusSpeed()* can be used to check whether a camera is connected to a USB 2.0 host controller.
If the value zero (0) is sent for the camera handle, a check is made whether a USB 2.0 controller is present in the system.

Parameters

hf	Camera handle

Return value

IS_SUCCESS	USB 2.0 Controller available (*hf*=0)
IS_NO_SUCCESS	No USB 2.0 Controller available (*hf*=0)
IS_USB_10	Controller port to which the camera is connected supports no USB 2.0
IS_USB_20	Camera is connected to a USB 2.0 controller

4.25 is_GetCameraInfo

Syntax

INT is_GetCameraInfo (HIDS hf, PCAMINFO pInfo)

Description

The function i*s_GetCameraInfo()* reads the data from the EEPROM and writes it to the data structure *pInfo*. The data structure is described in chapter *2.3 CAMINFO – data structure of the EEPROM*.
Reading and writing own data in and from the EEPROM are accomplished over the functions *is_ReadEEPROM()* and *is_WriteEEPROM()*.

Parameters

hf	Camera handle
pInfo	Pointer to data structure with the information from the CAM-INFO

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.26 is_GetCameraList

Syntax

INT is_GetCameraList (PUEYE_CAMERA_LIST pucl)

Description

With the function *is_GetCameraList()* information about the attached cameras can be queried.In order to obtain all information, the field size must be adapted to the number of connected cameras
In the following tables the used structures are described.

UEYE_CAMERA_LIST

ULONG	dwCount	Number of cameras connected to the system.
UEYE_CAMERA_INFO	uci[1]	Place holder for 1 .. n *UEYE_CAMERA_INFO* structures

UEYE_CAMERA_INFO

DWORD	dwCameraID	Number of cameras attached at the system
DWORD	dwDeviceID	system internal device ID.
DWORD	dwSensorID	Sensor ID
DWORD	dwInUse	1 = camera in use 0 = camera not in use
IS_CHAR	SerNo[16]	Serial number of the camera
IS_CHAR	Model[16]	Camera model
DWORD	dwReserved[16]	Reserved

Parameters

pucl	Structure *UEYE_CAMERA_LIST*

Return value

IS_SUCCESS, IS_ACCESS_VIOLATION (not enough memory allocated) or *IS_CANT_OPEN_DEVICE* respetively I*S_IO_REQUEST_FAILED* (communication with driver failed)

Example

```
PUEYE_CAMERA_LIST pucl = new UEYE_CAMERA_LIST;
//first request number of cameras to determine the array size
//within the UEYE_CAMERA_LIST structure
pucl->dwCount = 0;
if (is_GetCameraList (pucl) == IS_SUCCESS){
  //get number of cameras
  DWORD dwCameraCount = pucl->dwCount;
  delete pucl;
  //reallocate the required list size
  pucl = (PUEYE_CAMERA_LIST) new char [sizeof (DWORD) + dwCameraCount * sizeof (UEYE_CAM-
  ERA_INFO)];
  pucl->dwCount = dwCameraCount;
  //let the DLL fill in the camera info
    if (is_GetCameraList(pucl) == IS_SUCCESS){
    for (int iCamera = 0; iCamera < (int) pucl->dwCount; iCamera++){
      //process camera info
      printf("Camera %i Id: %d", iCamera,
      pucl->uci[iCamera].dwCameraID);
    }
  }
}
```

4.27 is_GetCameraType

Syntax

INT is_GetCameraType (HIDS hf)

Description

is_GetCameraType() returns the result of which type of camera family is installed.

hf	Camera handle

Return value

IS_CAMERA_TYPE_UEYE_USB (uEye USB2.0 cameras)
IS_CAMERA_TYPE_UEYE_ETH (uEye Gigabit Ethernet cameras)

4.28 is_GetColorDepth

Syntax

INT is_GetColorDepth(HIDS hf, INT* pnCol, INT* pnColMode)

Description

is_GetColorDepth() gets the current VGA card colour setting and returns the bit depth (*pnCol*) and the related colour mode (*pnColMode*). The colour mode can be directly passed to the *is_SetColorMode()* function.

Parameters

hf	Camera handle
PnCol	Returns bit depth of colour setting 8 at 256 colours 15 at 32768 colours (5:5:5 mode) 16 at 65536 colours (5:6:5 mode) 24 at 16777216 colours (8:8:8 mode) 32 at 16777216 colours (0:8:8:8 mode)
pnColMode	Returns colour mode for *is_SetColorMode()* IS_SET_CM_Y8 at pnCol = 8 IS_SET_CM_RGB15 at pnCol = 15 IS_SET_CM_RGB16 at pnCol = 16 IS_SET_CM_RGB24 at pnCol = 24 IS_SET_CM_RGB32 at pnCol = 32

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.29 is_GetDC

Syntax

INT is_GetDC (HIDS hf, HDC* phDC)

Description

In DirectDraw BackBuffer mode i*s_GetDC()* returns the overlay buffer's device context handle. Using this handle Windows GDI functions can access the overlay. All graphics commands such as line, circle, rectangle and text out from Windows are available. The device context handle must be released as soon as possible with the *is_ReleaseDC()* function. Within the *GetDC* - *ReleaseDC* blocks there are no updates of the overlay buffer on the display.

Parameters

hf	Camera handle
phDC	Pointer to the variable, which the device handle takes over.

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.30 is_GetDDOvlSurface

Syntax

INT is_GetDDOvlSurface (HIDS hf, LPDIRECTDRAWSURFACE* ppDDSurf)

Description

In DirectDraw back buffer mode *is_GetDDOvlSurface()* returns the pointer to the internal DirectDraw surface. And thus functions from the DirectDraw surface interface can be used.

Parameters

hf	Camera handle
ppDDSurf	Contains pointer to the DirectDraw surface interface

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.31 is_GetDLLVersion

Syntax

INT is_GetDLLVersion()

Description

Returns the version number of the *uEye_api DLL*.
The return value contains the version number in the following coding:

Bits 31-24:	major version number
Bits 16-23:	minor version number
Bits 15-0:	build version number

Parameters

<none>

Return value

Number of the version

4.32 is_GetError

Syntax

INT is_GetError (HIDS hf, INT* pErr, char** ppcErr)

Description

is_GetError() finds out what the last error was and returns the error code and error message. It is recommended to call this function after an error occurred (Return value ≠ IS_SUCCESS). The last error message is not deleted, but overwritten with a new one.

Parameters

hf	Camera handle
PErr	Pointer to variable, which will contain the error code.
PpcErr	Pointer to the string, which then contains the error message

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.33 is_GetEthDeviceInfo (uEye Gigabit Ethernet cameras only)

Syntax

INT is_GetEthDeviceInfo(HIDS hf, UEYE_ETH_DEVICE_INFO* pDeviceInfo,
UINT uStructSize)

Description

With the function *is_GetEthDeviceInfo()* information about the connected cameras can be readout. The information is available in the structure *UEYE_ETH_DEVICE_INFO*.

UEYE_ETH_DEVICE_INFO

UEYE_ETH_DEVICE_INFO_HEARTBEAT	infoDevHeartbeat	Camera related data (from the heartbeat telegram) See UEYE_ETH_DEVICE_INFO_HEARTBEAT
UEYE_ETH_DEVICE_INFO_CONTROL	infoDevControl	Camera related data of the driver See UEYE_ETH_DEVICE_INFO_CONTROL
UEYE_ETH_ADAPTER_INFO	infoAdapter	Adapter related data of the driver See UEYE_ETH_ADAPTER_INFO
UEYE_ETH_DRIVER_INFO	infoDriver	General driver data See UEYE_ETH_DRIVER_INFO

UEYE_ETH_DEVICE_INFO_HEARTBEAT

BYTE	abySerialNumber[12]	Serial number (String)
BYTE	byDeviceType	Type of the camera family, 0x80 for Eth
BYTE	byCameraID	User defined ID of the camera
WORD	wSensorID	Sensors ID
WORD	wSizeImgMem_MB	Size of the image memory in MByte
BYTE	reserved_1[2]	reserved
DWORD	dwVerStarterFirmware	Version starter firmware
DWORD	dwVerRuntimeFirmware	Version runtime firmware
DWORD	dwStatus	Status word
BYTE	reserved_2[4]	reserved
WORD	wTemperature	Camera temperature
WORD	wLinkSpeed_Mb	Band width of the link in Mbit
UEYE_ETH_ADDR_MAC	macDevice	MAC-Address of the camera
BYTE	reserved_3[2]	reserved

UEYE_ETH_IP_CONFIGURATION	ipcfgPersistentIpCfg	Persistent IP configuration
UEYE_ETH_IP_CONFIGURATION	ipcfgCurrentIpCfg	Current IP configuration
UEYE_ETH_ADDR_MAC	macPairedHost	MAC address of the connected PCs
BYTE	reserved_4[2]	reserved
UEYE_ETH_ADDR_IPV4	ipPairedHostIp	IP address of the connected PCs
UEYE_ETH_ADDR_IPV4	ipAutoCfgIpRangeBegin	First IP address of the auto configuration range
UEYE_ETH_ADDR_IPV4	ipAutoCfgIpRangeEnd	Last IP address of the auto configuration range
BYTE	abyUserSpace[8]	First eight bytes of the user EEPROM
BYTE	reserved_5[84]	reserved
BYTE	reserved_6[64]	reserved

UEYE_ETH_DEVICE_INFO_CONTROL

DWORD	dwDeviceID	system internal device ID of the camera
DWORD	dwControlStatus	Status word for the camera administration by the driver
BYTE	reserved_1[80]	reserved
BYTE	reserved_2[64]	reserved

UEYE_ETH_ADAPTER_INFO

WORD	wAdapterID	Driver internal ID of the network adapter
BYTE	reserved_1[4]	reserved.
UEYE_ETH_ETHERNET_CONFIGURATION	ethcfg	Ethernet configuration of the network adapter
BYTE	reserved_2[2]	reserved
BOOL	bIsEnabledDHCP	The adapter is configured for DHCP
UEYE_ETH_AUTOCFG_IP_SETUP	autoCfgIp	Setting of the IP addresses of the auto configuration
BOOL	bIsValidAutoCfgIpRange	The setting for the IP auto configuration is invalid
DWORD	dwCntDevicesKnown	Number of known cameras at this network adapter
DWORD	dwCntDevicesPaired	Number of opened cameras at this network adapter
WORD	wPacketFilter	Filter setting for incoming packets
BYTE	reserved_3[38]	reserved
BYTE	reserved_4[64]	reserved

UEYE_ETH_DRIVER_INFO

DWORD	dwMinVerStarterFirmware	Minimal compatible version of the starter firmware
DWORD	dwMaxVerStarterFirmware	Maximal compatible version of the tarter firmware
BYTE	reserved_1[8]	reserved
BYTE	reserved_2[64]	reserved

Parameters

hf	*DevID* | *IS_USE_DEVICE_ID*, *DevID* = system internal device ID of the camera
pDeviceInfo	Pointer to an object *UEYE_ETH_DEVICE_INFO*
uStructSize	Size of the structure *UEYE_ETH_DEVICE_INFO* in Bytes

Return values

IS_SUCCESS	The data have been readout error free
IS_INVALID_PARAMETER	The parameter *pDeviceInfo* is invalid
IS_BAD_STRUCTURE_SIZE	The specified structure size is invalid
IS_NOT_SUPPORTED	*hf* was not marked as device ID or it is not a device ID for an ethernet camera.
IS_CANT_OPEN_DEVICE	Driver not found
IS_IO_REQUEST_FAILED	Communication with driver aborted.

Example

```
UEYE_ETH_DEVICE_INFO di;
//Prepare handle parameter. mark the given device id with IS_USE_DEVICE_ID.
HIDS h= (HIDS)(iDeviceID | IS_USE_DEVICE_ID);
INT iErr= is_GetEthDeviceInfo( h, &di, sizeof(UEYE_ETH_DEVICE_INFO));
if( iErr != IS_SUCCESS)
{
    // ...
}
```


4.34 is_GetExposureRange

Syntax

INT is_GetExposureRange (HIDS hf, double *min,double *max, double *intervall)

Description

The function *is_GetExposureRange()* can be used to check the possible exposure values in milliseconds for the currently set timing (pixel clock, frame rate). The possible times lie between *min* and *max* and can be set in large stages in *interval*.

Parameters

hf	Camera handle
min	Contains the minimum possible exposure time
max	Contains the maximum possible exposure time
intervall	Contains the step sizes with which the image duration can be changed

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.35 is_GetFramesPerSecond

Syntax

INT is_GetFramesPerSecond (HIDS hf, double *dblFPS)

Description

is_GetFramesPerSecond() returns the number of the actual frame rate in live mode (see *4.4 is_CaptureVideo*).

Parameters

hf	Camera handle
dblFPS	Frame rate

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.36 is_GetFrameTimeRange

Syntax

INT is_GetFrameTimeRange (HIDS hf, double *min, double *max, double *intervall)

Description

is_GetFrameTimeRange() can be used to read out the frame rate settings that are possible for the current settings of the pixel clock. The returned values state the minimum and maximum possible duration of an image in seconds. The possible frame duration can be set in steps in *interval* between *min* and *max*.

$$fps_{(min)} = \frac{1}{max}$$

$$fps_{(max)} = \frac{1}{min}$$

$$fps_{(n)} = \frac{1}{min + m} \qquad m = n \cdot intervall$$

Parameters

hf	Camera handle
min	Contains the minimum possible duration of an image
max	Contains the maximum possible duration of an image
intervall	Contains the step sizes with which the image duration can be changed

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.37 is_GetGlobalFlashDelays

Syntax

INT is_GetGlobalFlashDelays (HIDS hf, ULONG *pulDelay, ULONG *pulDuration)

Description

Rolling Shutter cameras:

The function *is_GetGlobalFlashDelays()* allows the required times to be determined in order to implement a global flash function for rolling shutter cameras. Thus a rolling shutter camera can be operated as global shutter camera, if the scene to be taken is dark in the lightning break between two images.

If the exposure time is set too short, so that a global lightning is no longer possible, *IS_NO_SUCCESS* is returned.

Global Shutter cameras:

The exposure is delayed with global shutter cameras in freerun mode, if the exposure time is not set to the maiximum value.The function *is_GetGlobalFlashDelays()* acquires the necessary delay, in order to synchronize exposure and flash. In triggered mode the return values for the delay and duration of the flash are 0, because there is no delay needed up to the beginning of the exposure.

Parameters

hf	Camera handle
ulDelay	Delay time of the flash (in µs)
ulDuration	Flash duration time (in µs)

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.38 is_GetHdrKneepointInfo

Syntax

INT is_GetHdrKneepointInfo (HIDS hf, KNEEPOINTINFO *KneepointInfo, INT KneepointInfoSize)

Description

Some sensors support HDR mode (High Dynamic Range). HDR mode can be activated/deactivated with *is_EnableHdr()*. Use *is_GetHdrKneepointinfo()* to access general information on the knee points. It is returned in a KNEEPOINTINFO structure.

Currently, only the UI-122X-X and UI-522X-X models support HDR.
For cameras of types UI-122X-C and UI-522X-C, the RGB gains must be set to the same values to ensure accurate colour rendition in HDR mode.

Parameters

hf	Camera handle
KneepointInfo	Pointer to a structure with the following parameters:
INT NumberOfSupportedKneepoints	Maximum number of supported knee points
INT NumberOfUsedKneepoints	Current number of knee points used
double MinValueX	Minimum possible X value of a knee point
double MaxValueX	Maximum possible X value of a knee point
double MinValueY	Minimum possible Y value of a knee point
double MaxValueY	Maximum possible Y value of a knee point
INT Reserved[10]	

Return value

IS_SUCCESS or *IS_NO_SUCCESS* for supported sensor types
IS_NOT_SUPPORTED for unsupported sensor types

4.39 is_GetHdrKneepoints

Syntax

INT is_GetHdrKneepoints (HIDS hf, KNEEPOINT_ARRAY *KneepointArray, INT KneepointArraySize)

Description

Some sensors support HDR mode (High Dynamic Range). HDR mode can be activated/deactivated with *is_EnableHdr()*. Use *is_GetHdrKneepoints()* to access the currently set knee points. (See also *4.113 is_SetHdrKneepoints*)

Currently, only the UI-122X-X and UI-522X-X models support HDR.
For cameras of types UI-122X-C and UI-522X-C, the RGB gains must be set to the same values to ensure accurate colour rendition in HDR mode.

Parameters

hf	Camera handle
KneepointArray	Pointer to a KNEEPOINT_ARRAY

Return value

IS_SUCCESS or *IS_NO_SUCCESS* for supported sensor types
IS_NOT_SUPPORTED for unsupported sensor types

4.40 is_GetHdrMode

Syntax

INT is_GetHdrMode (HIDS hf, INT *Mode)

Description

Some sensors support HDR mode (High Dynamic Range). HDR mode can be activated/deactivated with *is_EnableHdr()*. Use *is_GetHdrMode()* to query the HDR mode supported by the sensor.

Currently, only the UI-122X-X and UI-522X-X models support HDR.
For cameras of types UI-122X-C and UI-522X-C, the RGB gains must be set to the same values to ensure accurate colour rendition in HDR mode.

Parameters

hf	Camera handle
Mode	
IS_HDR_KNEEPOINTS	HDR supported
IS_HDR_NOT_SUPPORTED	HDR not supported

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.41 is_GetImageHistogram

Syntax

INT is_GetImageHistogram (HIDS hf, int nID, INT ColorMode, DWORD* pHistoMem)

Description

is_GetImageHistogram() calculates the histogram of the given picture, only the following colour formats are supported RGB32, RGB24, RGB16, RGB15, Raw *Bayer* and Y8.

Parameters

Hf	Camera handle
nID	Memory ID
ColorMode	Colour mode of the image with the memory id *nID*
IS_SET_CM_RGB32	DWORD Array [256*3]
S_SET_CM_RGB24	DWORD Array [256*3]
IS_SET_CM_RGB16	DWORD Array [256*3]
IS_SET_CM_RGB15	DWORD Array [256*3]
IS_SET_CM_BAYER	DWORD Array [256*3]
IS_SET_CM_Y8	DWORD Array [256]
pHistoMem	Pointer to a DWORD array

Return value

IS_SUCCESS, IS_NO_SUCCESS
IS_NULL_POINTER invalid array
IS_INVALID_COLOR_FORMAT unsupported colour format
IS_INVALID_PARAMETER unknown parameter *ColorMode*

Example

Create a RGB test picture

```
char * pcSource;
INT nIDSource;
is_AllocImageMem (hf, 256, 256, 24, &pcSource, &nIDSource);
Int nX,nY,nBits,nPitch;
is_InquireImageMem (hf,pcSource,nIDSource,&nX,&nY,&nBits,&nPitch);
for (int j = 0;j< nY;j++){
    for (int i = 0;i< nX*3;i+=3){
        pcSource[i + j*nPitch] = 0; // Blue pixels
        pcSource[i + j*nPitch + 1] = i/3; // Green pixels
        pcSource[i + j*nPitch + 2] = 255; // Red pixels
    }
}
// memory for the rgb histogram
DWORD bgrBuffer [256*3];
```

```
//Assign a pointer to each color histogram
DWORD * pBlueHisto  = bgrBuffer;
DWORD *pGreenHisto = bgrBuffer + 256;
DWORD * pRedHisto = bgrBuffer + 512;
is_GetImageHistogram (hf, nIDSource, IS_SET_CM_RGB24, bgrBuffer);
is_FreeImageMem (hf, pcSource, nIDSource);
```

4.42 is_GetImageMem

Syntax

INT is_GetImageMem (HIDS hf, VOID** pMem)

Description

is_GetImageMem() returns the pointer to the start of the active image memory. In DirectDraw mode the pointer is returned to the back buffer (or the visible area - DirectDraw Primary Surface mode).

In DirectDraw mode the address pointer changes when the output window is moved (see also *4.65 is_LockDDMem*). When ring buffering is being used (*4.1 is_AddToSequence*) *is_GetImageMem()* returns the start address of the previously active image memory.

Parameters

hf	Camera handle
pMem	Pointer to beginning of image memory

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.43 is_GetImageMemPitch

Syntax

INT is_GetImageMemPitch (HIDS hf, INT* pPitch)

Description

is_GetImageMemPitch() returns the line increment in bytes. The line increment is the number of bytes from the beginning of a line to the beginning of the next line. If required, the line increment can be larger than the returned parameters from *is_AllocImageMem()*. The line increment is always a multiple of 4 (also see *4.2 is_AllocImageMem*);

The line increment is calculated as follows:

line = width. [(bitspixel + 1) / 8]

lineinc = line + adjust.

adjust = 0 if *line* is divisible by 4 without rest

adjust = 4 - rest(line / 4) if *line* is not divisible by 4 without rest

Parameters

hf	Camera handle
pPitch	Pointer to variable, which contains line increment

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.44 is_GetLastMemorySequence

Syntax

INT is_GetLastMemorySequence (HIDS hf, INT *pID)

Description

The function *is_GetLastMemorySequence()* returns the ID of the last recorded sequence in the memory board. This parameter can then be used in combination with the function *is_TransferImage()* to read images out of the camera memory.

Parameters

hf	Camera handle
pID	Returns the ID of the last recorded sequence to the memory.

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.45 is_GetMemorySequenceWindow

Syntax

INT is_GetMemorySequenceWindow (HIDS hf, INT nID, INT *left, INT *top, INT *right, INT *bottom)

Description

The function *is_GetMemorySequenceWindow()* can be used to check the window size of a specified memory board sequence. The assigned sequence ID is required as a parameter.

Parameters

hf	Camera handle
nID	Sequence ID for which window coordinates can be checked.
left	Returns the start column.
top	Returns the start row.
right	Returns the end column.
bottom	Returns the end row

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.46 is_GetNumberOfCameras

Syntax

INT is_GetNumberOfCameras (INT* pnNumCams)

Description

Function i*s_GetNumberOfCameras()* delivers the number of uEye cameras attached to the PC.

Parameters

pNumCams	Number of cameras found.

Return value

IS_SUCCESS

4.47 is_GetNumberOfMemoryImages

Syntax

INT is_GetNumberOfMemoryImages (HIDS hf, INT nID, INT *pnCount)

Description

The function *is_GetNumberOfMemoryImages()* returns the number of valid images that are currently located in the camera memory within the specified sequence ID. This number can differ from the originally recorded number of images because of overwriting.

Parameters

hf	Camera handle
nID	Specifies the ID at which the number of existing images is to be indicated.
pnCount	This parameter indicates the number of images in the sequence.

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.48 is_GetOsVersion

Syntax

INT is_GetOsVersion()

Description

is_GetOsVersion() returns the type of operating system, which is currently running on the machine at which the camera is attached.

Parameters

<none>

Return value

IS_OS_WIN2000, IS_OS_WINXP, IS_OS_WINSERVER2003

4.49 is_GetPixelClockRange

Syntax

INT is_GetPixelClockRange (HIDS hf, INT *pnMin, INT *pnMax)

Description

is_GetPixelClockRange() returns the adjustable range for the pixel clock.
Depending upon camera model and operating mode the limit values for the pixel clock can vary. Thus the minimum pixel clock increases with a UI-1220x and activated 4x-Binning (horizontal) from 8 MHz to 16 MHz and with a UI-141x and activated 2x-Sampling (horizontal) from 5MHz to 10MHz.

Parameters

hf	Camera handle
pnMin	Returns the lower limit.
pnMax	Returns the upper limit.

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.50 is_GetSensorInfo

Syntax

INT is_GetSensorInfo (HIDS hf, PSENSORINFO pInfo)

Description

With *is_GetSensorInfo()* information about the used camera sensor can be queried. The information contained in the structure *SENSORINFO* is listed in the following table:

WORD	SensorID	Sensor type	hex	dec
		CMOS		
		IS_SENSOR_INVALID	0	0
		IS_SENSOR_UI141X_M	1	1
		IS_SENSOR_UI141X_C	2	2
		IS_SENSOR_UI144X_M	3	3
		IS_SENSOR_UI144X_C	4	4
		IS_SENSOR_UI145X_C	8	8
		IS_SENSOR_UI146X_C	A	10
		IS_SENSOR_UI121X_M	10	16
		IS_SENSOR_UI121X_C	11	17
		IS_SENSOR_UI122X_M	12	18
		IS_SENSOR_UI122X_C	13	19
		IS_SENSOR_UI164X_C	20	32
		IS_SENSOR_UI154X_M	30	48
		IS_SENSOR_UI154X_C	31	49
		IS_SENSOR_UI1543_M	32	50
		IS_SENSOR_UI1543_C	33	51
		CCD		
		IS_SENSOR_UI223X_M	80	128
		IS_SENSOR_UI223X_C	81	129
		IS_SENSOR_UI241X_C	82	130
		IS_SENSOR_UI241X_C	83	131
		IS_SENSOR_UI221X_M	88	136
		IS_SENSOR_UI221X_C	89	137
		IS_SENSOR_UI231X_M	90	144
		IS_SENSOR_UI231X_C	91	145
		IS_SENSOR_UI222x_M	92	146

IS_SENSOR_UI222x_C	93	147
IS_SENSOR_UI233x_M	94	148
IS_SENSOR_UI233x_C	95	149
IS_SENSOR_UI224x_M	96	150
IS_SENSOR_UI224x_C	97	151
IS_SENSOR_UI225x_M	98	152
IS_SENSOR_UI225x_C	99	153

Char	strSensorName[32]	Camera model e.g. "UI141x-M"
Char	nColorMode	Sensor colour mode: IS_COLORMODE_BAYER IS_COLORMODE_MONOCHROME
DWORD	nMaxWidth	Maximum windows width e.g. 1280
DWORD	nMaxHeight	Maximum window height e.g. 1024
BOOL	bMasterGain	Is a common amplifier present? e.g. FALSE
BOOL	bRGain	Red amplifier present? e.g. TRUE
BOOL	bGGain	Green amplifier present? e.g. TRUE
BOOL	bBGain	Blue amplifier present? e.g. TRUE
BOOL	bGlobShutter	Global or Rolling Shutter ? e.g. TRUE (=global shutter present)
Char	Reserved[16]	Reserved

Parameters

hf	Camera handle
pInfo	Sensor Info structure

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.51 is_GetUsedbandwidth

Syntax

INT is_GetUsedbandwidth (HIDS hf)

Description

is_GetUsedBandwidth() indicates the bandwidth currently being used with the pixel clock of all active cameras.

Parameters

hf	Camera handle

Return value

Sum of the pixelclock

4.52 is_GetVsyncCount

Syntax

INT is_GetVsyncCount (HIDS hf, long* pIntr, long* pFrame)

Description

is_GetVsyncCount() reads the VSYNC counter. This is increased by 1 for each VSYNC.

Parameters

hf	Camera handle
pIntr	Current VSYNC counter
pFrame	Current Frame-SYNC counter

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.53 is_GetWhiteBalanceMultipliers

Syntax

INT is_GetWhiteBalanceMultipliers (HIDS hf, double *pdblRed, double *pdblGreen, double *pdblBlue)

Description

is_GetWhiteBalanceMultipliers() returns the present multipliers of the white balance. These have been set previously with *is_SetWhiteBalanceMultipliers()* or have been calculated with the automatic white balance.

Parameters

hf	Camera handle
pdblRed	Multiplier for the red channel
pdblGreen	Multiplier for the green channel
pdblBlue	Multiplier for the blue channel

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.54 is_HasVideoStarted

Syntax

INT is_HasVideoStarted (HIDS hf, BOOL* pbo)

Description

is_HasVideoStarted() can check whether the digitizing of an image has started or not. This function is useful when used together with *is_FreezeVideo()* and the parameter *IS_DONT_WAIT*.

Parameters

hf	Camera handle
pbo	Contains the digitization status: 0 = not started 1 = image acquisition started

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.55 is_HideDDOverlay

Syntax

INT is_HideDDOverlay (HIDS hf)

Description

is_HideDDOverlay() fades out the overlay in DirectDraw back buffer mode. Then only the image buffer is displayed, so that the image refresh frequency is higher than then the overlay, which is being superimposed. When the overlay is faded out the overlay data is not lost.

Parameters

hf	Camera handle

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.56 is_InitCamera

Syntax

INT is_InitCamera (HIDS* phf, HWND hWnd)

Description

is_InitCamera() opens the driver and establishes contact to the hardware. If the hardware is successfully initialized, this function allocates a handle to the camera. All of the following functions require this as their first parameter. If DirectDraw is not used for image output, *hWnd* can be set to Null.
To install several uEye cameras at one computer (multi camera operation), you have to decide during initialisation of the handle which camera is to be initialized. The Camera-ID is fixed in the EEPROM (see also *4.25 is_GetCameraInfo*). If a special board is to be addressed, you have to initialize the handle *hf* with the relevant Camera-ID.
If you don't want multi camera operation, handle *hf* must be initialized with the value zero before calling function *is_InitCamera()*. Here the value zero means, that the first not used camera will be used.

The uEye SDK is *thread safe* in general. The API function calls are all done in critical sections. Due to internal structures of DirectDraw we strongly recommend to call the following functions from only one thread to avoid unpredictable behaviour of your application.

- *is_InitCamera()*
- *is_SetDisplayMode()*
- *is_ExitCamera()*

Parameters

phf	Pointer to the camera handle. The values have the following meaning: 0: use first unused camera 1-254: use camera with this Camera-ID
hWnd	Handle to the window in which the image is to be displayed

Return value

IS_SUCCESS, error code (see header file)

4.57 is_InitEvent

Syntax

INT is_InitEvent (HIDS hf, HANDLE hEv, INT which)

Description

Initializes the event handler by registering the event object in the central driver.

Parameters

hf	Camera handle
hEv	Event-Handle of the C/C++-Function *CreateEvent()*
which	Event ID to be initialized:
IS_SET_EVENT_FRAME	A new image is available.
IS_SET_EVENT_SEQ	The sequence was gone through.
IS_SET_EVENT_STEAL	An image detracted from the overlay is available
IS_SET_EVENT_TRANSFER_FAILED	Data loss during transmission
IS_SET_EVENT_EXTTRIG	An image, its recording was released by a trigger, was completely received. This is the earliest time for a new recording. The picture must pass the post processing of the driver and is after IS_FRAME available for the processing.
IS_SET_EVENT_MEMORY_MODE_FINISH	Recording images to the optional memory module has been terminated.
IS_SET_EVENT_REMOVE	A camera, opened with *is_InitCamera()* has been removed.
IS_SET_EVENT_DEVICE_RECONNECTED	A camera opened with *is_InitCamera()* and thereafter removed was reconnected.
IS_SET_EVENT_NEW_DEVICE	A new camera was attached. Independent of the device handle (*hf* will be ignored).
IS_SET_EVENT_REMOVAL	A camera was removed. Independent of the device handle (*hf* will be ignored).
IS_SET_EVENT_WB_FINISHED	The automatic white balance control is finished.

Return value

IS_SUCCESS, IS_NO_SUCCESS

Example

Activate frame events, start image recording and wait for an event:

```
HANDLE hEvent = CreateEvent(NULL, TRUE, FALSE, "");
if (hEvent != NULL) {
    is_InitEvent(hf, hEvent, IS_SET_EVENT_FRAME);
    is_EnableEvent(hf, IS_SET_EVENT_FRAME);
    is_FreezeVideo(hf, IS_DON_T_WAIT);
    if (WaitForSingleObject(hEvent, 1000) == WAIT_OBJECT_0) {
    // Bild Erfolgreich aufgenommen }
    is_DisableEvent(hf, IS_SET_EVENT_FRAME);
    is_ExitEvent(hf, IS_SET_EVENT_FRAME);
}
```

4.58 is_InquireImageMem

Syntax

INT is_InquireImageMem (HIDS hf, char* pcMem, int nID, int* pnX, int* pnY, int* pnBits, int* pnPitch);

Description

is_InquireImageMem() reads the properties of the allocated image memory.

Parameters

hf	Camera handle
pMem	Pointer to beginning of image memory from i*s_AllocImage-Mem()*
NID	ID of image memory of *is_AllocImageMem()*
pnX	Contains the width, with which the image memory was created.
pnY	Contains the height, with which the image memory was created.
pnBits	Contains the bit width, with which the image memory was created.
pnPitch	Contains he line increment of the image memory.

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.59 is_IsMemoryBoardConnected

Syntax

INT is_IsMemoryBoardConnected (HIDS hf, BOOL* pConnected)

Description

The function *is_IsMemoryBoardConnected()* can be used to check whether the optional memory board is available.

Parameters

hf	Camera handle
pConnected	TRUE = Memoryboard is connected FALSE = No Memoryboard available

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.60 is_IsVideoFinish

Syntax

INT is_IsVideoFinish (HIDS hf, BOOL* pbo)

Description

is_IsVideoFinish() can check to see whether an image has been completely and fully acquired in the image memory. This function is useful when used together with *is_FreezeVideo()* with the *IS_DONT_WAIT* parameter.

If **pbo* is preset with *IS_TRANSFER_FAILED* before the call of *is_IsVideoFinish()*, the return value contains additional information whether a transfer error or an error in the pixel path occurred.

Parameters

hf	Camera handle
pbo	*pbo != IS_TRANSFER_FAILED before the function call pbo contains the digitizer status: IS_VIDEO_NOT_FINISH = digitizing not finished IS_VIDEO_FINISH = digitizing of image finished *pbo == IS_TRANSFER_FAILED before the function call pbo contains the digitizer status: IS_VIDEO_NOT_FINISH = Digitization of the picture not yet finished IS_VIDEO_FINISH = Digitization of the picture finished IS_TRANSFER_FAILED = Transfer error or problem when converting (e.g. goal memory invalidly)

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.61 is_LoadBadPixelCorrectionTable

Syntax

INT is_LoadBadPixelCorrectionTable (HIDS hf, char *File)

Description

is_LoadBadPixelCorrectionTable() loads a table saved previously with the function *is_SaveBad-PixelCorrectionTable()*. *File* specifies the file in which the coordinates were saved; if a ZERO pointer is sent, a dialogue is displayed for selection of the file.

Parameters

hf	Camera handle
File	Pointer to file with saved coordinates. Both the absolute and the relative path can be passed.

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.62 is_LoadImage

Syntax

INT is_LoadImage (HIDS hf, char* File)

Description

is_LoadImage() loads an image from a file. The image must be available in BMP format. The image is loaded into the active image memory (see also *4.42 is_GetImageMem* and *4.117 is_SetImageMem*).

Parameters

hf	Camera handle
File	Pointer on file name. Both the absolute and the relative path can be passed.

Return value

IS_SUCCESS	Image is loaded error free.
IS_FILE_READ_INVALID_BMP_SIZE	The image to be load is larger than the active image memory.
IS_FILE_READ_INVALID_BMP_ID	The file to be load does not have a valid bitmap format.
IS_FILE_READ_OPEN_ERROR	The file cannot be opened.

4.63 is_LoadImageMem

Syntax

INT is_LoadImageMem (HIDS hf, char* File, char** ppcImgMem, int* pid)

Description

is_LoadImageMem() loads an image from a file. The image must be available in BMP format. The image is loaded into a new allocated image memory with the properties colour format and depth.
With the function *is_FreeImageMem()* (see *4.19 is_FreeImageMem*) the image memory is released.

Parameters

hf	Camera handle
File	Name of the image file, NULL -> *Open* dialogue box is opened. Both the absolute and the relative path can be passed.
ppcImgMem	Pointer to variable receiving the start address
pid	Pointer to variable receiving a memory ID

Return value

IS_SUCCESS, IS_NO_SUCCESS (image is loaded error free)
IS_FILE_READ_INVALID_BMP_ID (the file to be loaded does not have a valid bitmap format)
IS_FILE_READ_OPEN_ERROR (the file cannot be opened)

4.64 is_LoadParameters

Syntax

INT is_LoadParameters (HIDS hf, char* pFilename)

Description

is_LoadParameters() loads the parameters of a camera that were previously saved as an ini file with *is_SaveParameters()*. If NULL is passed as the value of *pFilename*, the Windows Open file dialog is displayed.

The parameter sets in the camera's non-volatile memory can be loaded with the help of specific filenames:

	pFilename
Parameter set 1	"*cam**set1*" or "*/cam/set1*"
Parameter set 2	"*cam**set2*" or "*/cam/set2*"

See also *4.3 is_CameraStatus*.

ini files can only be loaded for the sensor type they were saved for.
While loading an ini-file it is to consider that already allocated memory matches to the parameters of the ini-file concerning size (AOI) and colour depth. Otherwise this leads to incorrect displaying.

Parameters

hf	Camera handle
pFilename	Pointer on file name. Both the absolute and the relative path can be passed. For the camera's internal parameter sets these would be "*cam**set1*" or "*/cam/set1*", or "*cam**set2*" or "*/cam/set2*".

Return value

IS_SUCCESS, IS_NO_SUCCESS
IS_INVALID_CAMERA_TYPE if the ini file belongs to a different camera type

Example ini-File:

[Sensor]
Sensor=UI122x-M
[Image size]
Start X=0

Start Y=0
Start X absolute=0
Start Y absolute=0
Width=752
Height=480
Binning=0
Subsampling=0
[Timing]
Pixelclock=20
Framerate=30.003810
Exposure=33.329100
[Parameters]
Colormode=6
Brightness=100
Contrast=215
Gamma=1.000000
Blacklevel Mode=1
Blacklevel Offset=0
[Gain]
Master=0
Red=0
Green=0
Blue=0
[Processing]
EdgeEnhancement=0
RopEffect=0
Whitebalance=0
Whitebalance Red=1.000000
Whitebalance Green=1.000000
Whitebalance Blue=1.000000
Color correction=0
[Auto features]
Auto Framerate control=0
Brightness exposure control=0
Brightness gain control=0
Brightness reference=128
Brightness speed=50
Brightness max gain=-1.000000
Brightness max exposure=1.000000
Brightness Aoi Left=0
Brightness Aoi Top=0
Brightness Aoi Width=1280

Brightness Aoi Height=1024
Auto WB control=0
Auto WB offsetR=0
Auto WB offsetB=0
Auto WB gainMin=0
Auto WB gainMax=100
Auto WB speed=50
Auto WB Aoi Left=0
Auto WB Aoi Top=0
Auto WB Aoi Width=1280
Auto WB Aoi Height=1024
Auto WB Once=0

4.65 is_LockDDMem

Syntax

INT is_LockDDMem (HIDS hf, void** ppMem, INT* pPitch)

Description

In DirectDraw mode *is_LockDDMem()* gives access to the image memory and returns the address pointer to the beginning of the image memory. In most cases the image memory is on the VGA card. Using the pointer the image memory can be accessed directly. Access is disabled with *is_UnlockDDMem()* – this must be done as soon as possible.
Digitizing to memory cannot be terminated with *is_LockDDMem()*!
When in DirectDraw BackBuffer mode, within a *LockDDMem –UnlockDDMem* block there are no updates of the BackBuffer on the display!

Parameters

hf	Camera handle
ppMem	Pointer to variable, which will contain address pointer.
pPitch	Pointer to variable, which will contain the pitch value.

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.66 is_LockDDOverlayMem

Syntax

INT is_LockDDOverlayMem(HIDS hf, void** ppMem, INT* pPitch)

Description

In DirectDraw mode *is_LockDDOverlayMem()* gives access to the image memory and returns the address pointer to the beginning of the image memory. And thus the overlay buffer can be directly written to, without having to use the Windows GDI functions. *pPitch* returns the line offset in bytes from the beginning of one line to the beginning of the following line. Access must be disabled as soon as possible with the *UnlockDDOverlayMem()* function. Within a *LockDDMem – UnlockDDMem* block there are no updates of the BackBuffer on the display.

Parameters

hf	Camera handle
ppMem	Pointer to variable, which will contain address pointer.
pPitch	Pointer to variable, which will contain the pitch value.

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.67 is_LockSeqBuf

Syntax

INT is_LockSeqBuf (HIDS hf, INT nNum, char* pcMem)

Description

is_LockSeqBuf() can be used to disable the overwriting of the image memory with new image data. And thus it is possible to prevent images which are required for further processing from being overwritten. Full access to the image memory is still available.
You can simultaneously lock as many image buffers as you like. To access the image memory use function *is_UnlockSeqBuf()*.

Parameters

hf	Camera handle
nNum	Number of the disabled image memory (1 ... max.) or IS_IGNORE_PARAMETER: The image buffer is only identified by the start address
pcMem	Start address of the image memory, which should be protected.

nNum indicates the position in the sequence list and not the memory ID assigned with *is_AllocImageMem()*.

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.68 is_MemoryFreezeVideo

Syntax

INT is_MemoryFreezeVideo (HIDS hf, INT nMemID, INT Wait)

Description

The function *is_MemoryFreezeVideo()* can be used to record a single frame via the memory board using a single command. First of all an image is transferred to the camera memory and then to the specified image memory.
The advantage of this function is that a recorded image can be transferred safely without transfer error even at times when the computer is being subjected to heavy workloads.
Effected after the call of the function with the parameter *IS_DONT_WAIT* a change of the camera settings (Exposure, Pclk, AOI...) the recording is cancelled and the new camera settings are taken over.

With call of the function *is_MemoryFreezeVideo()* the memory mode is activated automatically. To toggle back to image capture without memory mode, the memory mode must be switched off with the function *is_SetMemoryMode(IS_MEMORY_MODE_DISABLE, 0)* (see *4.124 is_SetMemoryMode*).

Parameters

hf	Camera handle
nMemID	ID of the memory where the image has been transferred; if 0, the currently active memory is used.
Wait	
IS_WAIT	Function waits until the picture is taken.
IS_DONT_WAIT	Function returns immediately
10 <= Wait < 21474836	Waiting period in 10 ms steps. Maximally 214748,36 seconds ca be waited. 1 < Wait < 10 wird Wait = 10 (Example.: Wait = 100 ⇨ wait 1 sec)

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.69 is_PrepareStealVideo

Syntax

INT is_PrepareStealVideo (HIDS hf, int Mode, ULONG StealColorMode)

Description

Sets the steal mode. There are two different steal modes

- normal steal
  This option initiates the stealing of a single image out of a DirectDraw live stream.
  This option redirect a single image out of a DirectDraw live stream into the current active user memory.
- copy steal
  This option shows the picture with DirectDraw and copies it into the current active image memory.

Parameters

hf	Camera handle
Mode	
IS_SET_STEAL_NORMAL	Normal mode
IS_SET_STEAL_COPY	Copy mode
StealColorMode	reserved

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.70 is_ReadEEPROM

Syntax

INT is_ReadEEPROM (HIDS hf, INT Adr, char* pcString, INT Count)

Description

There is a rewritable EEPROM in the camera which serves as a small memory. Additionally to the information which is stored in the EEPROM, 64 extra bytes can be written. With the *is_ReadEEPROM()* command the contents of these 64 bytes can be read. Also see *4.146 is_WriteEEPROM*.

Parameters

hf	Camera handle
Adr	Begin address from where data is to be read: Value range 0 to 63
pcString	Buffer for data to be read (min. size = Count)
Count	Number of characters to be read

Return value

IS_SUCCESS, IS_NO_SUCCESS

Example

```
char buffer[64];
is_ReadEEPROM(m_hCam, 0x00, buffer, 64);
```

4.71 is_ReadI2C (nur uEyeLE)

Syntax

INT is_ReadI2C (HIDS hf, INT nDeviceAddr, INT nRegisterAddr, BYTE* pbData, INT nLen)

Description

With *is_ReadI2C()* data can be read over the I2C-bus. I2C bus clock is 400 kHz.

Parameters

hf	Camera handle
nDeviceAdr	Slave address
nRegisterAddr	Register address (only 8 Bit addresses valid).
data	Data to read
length	Length of data

Invalid addresses for *nRegisterAddr*:
0x48, 0x4C, 0x51, 0x52, 0x55, 0x5C, 0x5D, 0x69 (these are used internal).

Return value

IS_SUCCESS, IS_NO_SUCCESS or *IS_INVALID_I2C_DEVICE_ADDRESS*

4.72 is_ReleaseDC

Syntax

INT is_ReleaseDC (HIDS hf, HDC hDC)

Description

In DirectDraw BackBuffer mode *is_ReleaseDC()* releases the overlay buffer's device context handle. Once the handle has been released, an update of the overlay buffer on the display follows (if the overlay has been blended on with *is_ShowDDOverlay()*).

Parameters

hf	Camera handle
hDC	Device context handle of *is_GetDC()*

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.73 is_RenderBitmap

Syntax

INT is_RenderBitmap (HIDS hf, INT nMemID, HWND hwnd, INT nMode)

Description

is_RenderBitmap() displays images from an image memory in the defined window. For displaying the images Win32 Bitmap functions are used. Displaying generally takes place in the format, which was specified during allocation of the image memory. In the function *is_AllocImageMem()* the parameter *bitspixel* specifies the colour depth and the presentation format. RGB16 and RGB 15 uses the same amount of memory, however they can be differentiated with the parameter *bitspixel*.

The colour format UYVY can only displayed expediently in the overlay mode. However this is not realizable with the functions of the Windows API.

Parameters

hf	Camera handle
nMemID	ID of the memory which should be displayed
hwnd	Window handle of the display window
nMode	
IS_RENDER_NORMAL	Image output 1:1 from the memory
IS_RENDER_FIT_TO_WINDOW	Fit the image into the size of the window
IS_RENDER_ DOWNSCALE_1_2	Display the image with a size of 50% of the original size.
Options which can be combined with the options above by using logical OR:	
IS_RENDER_MIRROR_UPDOWN	Horizontal mirroring of the image

Return value

IS_SUCCESS, IS_NO_SUCCESS

Example

Fit the image into a window with horizontal mirroring:

```
is_RenderBitmap (hf, nMemID, hwnd, IS_RENDER_FIT_TO_WINDOW | IS_RENDER_MIRROR_UP-
DOWN);
```

4.74 is_Renumerate

Syntax

INT is_Renumerate(HIDS hf, INT nMode)

Description

The function *is_Renumerate()* disconnects the camera from the USB bus and causes a reinitialization. Therefore this function is only reasonable for USB cameras. uEye Gigabit Ethernet cameras will return *IS_SUCCESS*, although the function has no affect.

Parameters

hf	Camera handle
nMode	
IS_RENUM_BY_CAMERA	The disconnection is initialized by the camera.
IS_RENUM_BY_HOST	The disconnection is initialized by the the Bus.

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.75 is_ResetMemory

Syntax

INT is_ResetMemory (HIDS hf, INT reserved)

Description

Stored image data remain in the storage of the memory board, as long as the camera is power supplied. Thus it is possible to access the data until the storage of the memory board is reset by the call of *is_ResetMemory()*.

Parameters

hf	Camera handle
reserved	Currently not used.

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.76 is_ResetToDefault

Syntax

INT is_ResetToDefault (HIDS hf)

Description

is_ResetToDefault() resets all parameters of the camera to the standard values.

Parameters

hf	Camera handle

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.77 is_SaveBadPixelCorrectionTable

Syntax

INT is_SaveBadPixelCorrectionTable (HIDS hf, char *File)

Description

is_SaveBadPixelCorrectionTable() stores the current, user-defined hot pixel list into the file, indicated with the F*ile* parameter.
With handing over the value NULL for the parameter *File* a dialogue for the selection of the file is shown.

Parameters

hf	Camera handle
File	Pointer to the file in which the data are stored. Both the absolute and the relative path can be passed.

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.78 is_SaveImage

Syntax

INT is_SaveImage (HIDS hf, char* File)

Description

Saves an image from the computer's memory to a file. The file is saved in BMP format. The images are read from the currently valid image memory (also see *4.42 is_GetImageMem*). The readout of the graphic card memory can last several 100msec, depending on the image size. In DirectDraw modes the image data are read from the corresponding DirectDraw buffer. Consider in DirectDraw primary surface mode that the image is saved as displayed on the screen. The maximum image size is the size of the output window, independent of which image size is set with *is_SetImageSize()*. Overlapped parts of the output window will be saved with the content of the overlapping window.
The bitmap is saved with an 8, 15, 16, 24 or 32 bit colour depth, as was allocated in the image memory and/or the colour mode of the DirectDraw mode. Some image processing programs do not support all 15 bit, 16 bit and 32 bit bitmaps and therefore will be unable to load the images.
The file name can contain an absolute as well as a relative file path. With *is_LoadImage()* the BMP images can be loaded.
Overlay data are not saved!

Parameters

hf	Camera handle
File	BMP file name NULL -> Save as - dialog is displayed (see i*s_CameraStatus()* for maintaining the selected directory path). Both the absolute and the relative path can be passed.

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.79 is_SaveImageEx

Syntax

INT is_SaveImageEx (HIDS hf, char* File, INT fileFormat, INT Param)

Description

Saves an image from the computer's memory as bitmap or JPEG to a file (see also *4.78 is_SaveImage*). The images are read from the currently valid image memory. The bitmap is saved with an 8, 15, 16, 24 or 32 bit colour depth, as it was allocated in the image memory and/or the colour mode of the DirectDraw mode. Some image processing programs do not support all 15 bit, 16 bit and 32 bit bitmaps and therefore will be unable to load the images. The file name can contain an absolute as well as a relative file path.

Parameters

hf	Camera handle
File	BMP file name NULL -> *Save as* - dialogue is displayed (see *i4.3 is_CameraStatus* for maintaining the selected directory path). Both the absolute and the relative path can be passed.
fileFormat	Determines the output format of the file
IS_IMG_BMP	Bitmap
IS_IMG_JPG	JPEG
Param	If JPEG is selected as file format, the quality of the compression can be set with this parameter. The quality can be set between 1 and 100, if this parameter is 0 the default quality (75) is used.

Return value

IS_SUCCESS, IS_NO_SUCCESS
IS_INVALID_PARAMETER (invalid file format or invalid JPEG quality)

4.80 is_SaveImageMem

Syntax

INT is_SaveImageMem (HIDS hf, char* File, char* pcMem, int nID)

Description

Saves an image in bitmap format (BMP) to a file. The images are read from the image memory. The bitmap is saved with an 8, 15, 16, 24 or 32 bit colour depth, as was allocated in the image memory. Some image processing programs do not support all 15 bit, 16 bit and 32 bit bitmaps and therefore will be unable to load the images.
Overlay data are not saved!

Parameters

hf	Camera handle
File	Name of the BMP image file NULL -> *Save as* dialogue box is opened *(see i4.3 is_CameraStatus* for maintaining the selected directory path. Both the absolute and the relative path can be passed.)
pcMem	Pointer to image memory
nID	ID of image memory *USE_ACTUAL_IMAGE_SIZE* OR *nID*: Save the image with the current setting of height

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.81 is_SaveImageMemEx

Syntax

INT is_SaveImageMemEx (HIDS hf, char* File, char* pcMem, int nID,INT fileFormat, INT Param).

Description

Saves an image in bitmap format (BMP) or JPEG format. The images are read from the image memory. The bitmap is saved with an 8, 15, 16, 24 or 32 bit colour depth, as allocated in the image memory. Some image processing programs do not support all 15 bits, 16 bits and 32 bits bitmaps and therefore will be unable to load the images. The JPEG file is always saved with a 8 bits or 24 bits colour depth.

Parameters

hf	Camera handle
File	Name of the image file, NULL -> *Save as* dialogue box is opened. Both the absolute and the relative path can be passed.
pcMem	Pointer to image memory
nID	ID of image memory
fileFormat	Determines the output format of the file.
IS_IMG_BMP	Bitmap
IS_IMG_JPG	JPEG
Param	If JPEG is selected as file format, the quality of the compression can be set with this parameter. The quality can be set between 1 and 100, if this parameter is 0 the default quality (75) is used.

Return value

IS_SUCCESS, IS_NO_SUCCESS
IS_INVALID_PARAMETER (invalid file format or JPEG quality)

4.82 is_SaveParameters

Syntax

INT is_SaveParameters (HIDS hf, char* pFilename)

Description

is_SaveParameters() saves the parameters of a camera to an ini file, which can later be loaded with *is_LoadParameters()*. If NULL is passed as the value of *pFilename*, the Windows *Save file* dialogue is displayed.

Two parameter sets can be stored in the camera's non-volatile memory

	pFilename
Parameter set 1	"*cam**set1*" or "*/cam/set1*"
Parameter set 2	"*cam**set2*" or "*/cam/set2*"

Parameters

hf	Camera handle
pFilename	Pointer on file name. Both the absolute and the relative path can be passed. For the camera's internal parameter sets these would be "*cam**set1*" or "*/cam/set1*", or "*cam**set2*" or "*/cam/set2*".

Return value

IS_SUCCESS, IS_NO_SUCCESS

Example ini-Datei

See *4.64 is_LoadParameters*.

4.83 is_SetAllocatedImageMem

Syntax

INT is_SetAllocatedImageMem (HIDS hf, INT width, INT height, INT bitspixel, char* pcImgMem, int* pid)

Description

is_SetAllocatedImageMem() defines an allocated memory to the current memory, in which the image will be digitized. The size of memory must be large enough and must be locked global! A memory indicated with *is_SetAllocatedImageMem()* can be built into a sequence. The memory must be taken out of the driver-internal administration with *is_FreeImageMem()*, before it is actually released. The allocated memory must not be reallocated.

The following steps have to be done:

- Allocate memory: HANDLE hgMem = GlobalAlloc (size);
- Lock memory: char* pcMem = (char*) GlobalLock (hgMem);

With the function *is_SetAllocatedImageMem()* the address of the memory is given to the uEye driver. Additionally as with function *is_AllocImageMem()* the size of the image must be set and as a result an identification number (*pid*) returns. This number might be needed in other functions.

is_SetAllocatedImageMem (hf, width, height, bitspixel, pcMem, &ID);

size >= (width * height * bitspixel / 8)

The area of the memory must be taken from driver administration again with the function *is_FreeImageMem (hf, pcMem, ID)*. With this the memory is NOT released.
The user must take care, that the memory is released:

- GlobalUnlock (*hgMem*);
- GlobalFree (*hgMem*);

Parameters

hf	Camera handle
width	Width of image
height	Height of image
bitspixel	Colour depth of image (bits per pixel)
pcImgMem	Contains pointer to start of memory location
pid	Contains the ID for this memory location

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.84 is_SetAOI

Syntax

INT is_SetAOI (HIDS hf, INT type, INT *pXPos, INT *pYPos, INT *pWidth, INT *pHeight)

Description

With *is_SetAOI()* the size and position of an AOI can be set with one command call. Possible AOI are:

- Image AOI
- Auto Brightness AOI
- Auto Whitebalance AOI
  The auto image areas are used for appropriate auto functionality. As default value the window is always maximally, thus always as large as the current image AOI.

After changes of image geometry, e.g. by new setting of the image AOI the Auto Feature AOI are always restored to the default value. This means that after a change of the image AOI or activation of binning the Auto Feature AOI must be set again.
Auto Whitebalance AOI is not available in DirectDraw mode and in UYVY mode.

Changing the image size with *is_SetAOI()* or *is_SetImageSize()* also effects the current frame rate and exposure time. Those can be adjusted with *is_SetFrameRate()* and *is_SetExposureTime()*, respectively.

With the functions *is_SetImagePos()* (see *4.118 is_SetImagePos*) and *is_SetImageSize()* (see *4.119 is_SetImageSize*) information about the size and position of the AOI can be read out.

Parameters

hf	Camera handle
type	
IS_SET_IMAGE_AOI	Set Image AOI
IS_GET_IMAGE_AOI	Returns the current Image AOI
IS_SET_AUTO_BRIGHT_AOI	Set the Auto Feature AOI for Auto Gain and Auto Shutter
IS_GET_AUTO_BRIGHT_AOI	Returns current average value of the AOI
IS_SET_AUTO_WB_AOI	Set AOI for Auto Whitebalance
IS_GET_AUTO_WB_AOI	Returns the current Auto Whitebalance AOI
pXPos	Horizontal position of the AOI
pYPos	Vertical position the AOI

pWidth	Width of the AOI
pHeight	Height of the AOI

Return value

IS_SUCCESS, IS_NO_SUCCESS

Examplee

Set Auto_Brightness_AOI:

```
Int nRet = SetAOI (m_hCam, // Kamera Handle
IS_SET_AUTO_BRIGHT_AOI,    // AOI Typ + Kommando
&xpos,
&ypos,
&width,
&height);
```

4.85 is_SetAutoCfgIpSetup (uEye Gigabit Ethernet cameras only)

Syntax

INT is_SetAutoCfgIpSetup(INT iAdapterID, const UEYE_ETH_AUTOCFG_IP_SETUP* pSetup, UINT uStructSize)

Description

With the function *is_SetAutoCfgIpSetup()* the attributes for the IP auto configuration of a network adapter are preset.
The information contained in the *UEYE_ETH_AUTOCFG_IP_SETUP* structure are listed in the following table:

UEYE_ETH_ADDR_IPV4	ipAutoCfgIpRangeBegin	First IPv4 address of the auto configuration range
UEYE_ETH_ADDR_IPV4	ipAutoCfgIpRangeEnd	Last IPv4 address of the auto configuration range
BYTE	reserved[4]	reserved

Parameters

iAdapterID	System internal adapter ID of the network adapter
pSetup	Pointer to an *UEYE_ETH_AUTOCFG_IP_SETUP object*
uStructSize	Size of the *UEYE_ETH_AUTOCFG_IP_SETUP structure* in bytes

Return values

IS_SUCCESS	Success.
IS_INVALID_PARAMETER	*pSetup* invalid.
IS_BAD_STRUCTURE_SIZE	The specified structure size is invalid
IS_CANT_OPEN_DEVICE	Driver not found
IS_IO_REQUEST_FAILED	Communication with the driver aborted

Example

```
// prepare the data parameter.
UEYE_ETH_AUTOCFG_IP_SETUP autocfgip;
autocfgip.ipAutoCfgIpRangeBegin.dwAddr= 0xC0A80A0F;
// IP address 192.168.10.15
autocfgip.ipAutoCfgIpRangeEnd.dwAddr=   0xC0A80A23;
// IP address 192.168.10.35
INT iErr= is_SetAutoCfgIpSetup( iAdapterId, &autocfgip,
                                sizeof(UEYE_ETH_AUTOCFG_IP_SETUP));
if( iErr != IS_SUCCESS)
{ // ...    }
```

4.86 is_SetAutoParameter

Syntax

INT is_SetAutoParameter (HIDS hf, INT param, double* pval1, double* pval2)

Description

is_SetAutoParameter() controls Auto Gain, Auto Shutter, Auto Framerate and Auto Whitebalance functionality. Purpose of the auto functions is it to control the camera image in its average brightness and colour rendering to the given value and to hold the picture frame rate to a maximum value. While controlling the average brightness the exposure control is preferred. That means that the maximally possible exposure time is set, before the gain is controlled. AutoGain controls the MasterGain of the camera in the range of 0-100%. AutoExposure uses the current exposure range which result from pixelclock and framerate. The control limits can be set separately for exposure and gain.
The frame rate can be changed further manually or automatically with activated shutter control, in order to maintain the range of the exposure control dynamically. The automatic frame rate control has the purpose of adjusting the frame rate to an optimal value. Thus the necessary control range is available for shutter control in all situations with a frame rate as high as possible.
During the control of the white balance the RGB gain settings of the camera are changed within the range 0-100% until the red and the blue channel reaches the average brightness of the green channel. In order to get a desired colour rendering the setpoint for the red and the blue channel can be adjusted by an offset relative to the green channel.
The speed of the auto functions can be adjusted in a range of 0 – 100%. Thus the absorbability and/or the inertia of the regulation are affected. High speed (100%) results in a small absorbability for a fast reacting regulation and in reverse. In this connection the control for the average brightness and the control for the colour rendering use separated speeds.
By setting the parameter *IS_SET_AUTO_WB_ONCE* the white balance can be terminated automatically, as soon as the target value was reached. The end of the control is communicated to the system over an event/message (see also *4.57 is_InitEvent*).
Alternatively the control keeps active and reacts to deviations from the target value.

The parameter *IS_SET_AUTO_BRIGHTNESS_ONCE* activates the same functionality for the auto brightness function.

With activating AutoShutter the adjustment of the pixel clock using the function *is_SetPixelClock()* is deactivated.
AutoFramerate is only possible with activated AutoShutter control because the Auto-Framerate adjusts the shutter range dynamically and AutoGain was never activated. Therefore these two features are mutually locked.
AutoGain is only possible for cameras with MasterGain adjustment.
Auto Whitebalance is possible only for cameras with hardware RGB Gain adjustment. With activated Auto Whitebalance the activation of the software Whitebalance over the function *is_SetWhiteBalance()* is disabled. This is also deactivated, if it is active when activating the Auto Whitebalance.

Parameters

hf	Camera handle
param	
IS_SET_ENABLE_AUTO_GAIN	Acvtivates/deactivates the AutoGain functionality
IS_GET_ENABLE_AUTO_GAIN	Returns the current AutoGain settings
IS_SET_ENABLE_AUTO_SHUTTER	Acvtivates/deactivates the AutoShutter functionality
IS_GET_ENABLE_AUTO_SHUTTER	Returns the current AutoShutter settings
IS_SET_ENABLE_AUTO_WHITEBALANCE	Acvtivates/deactivates the AutoWhitebalance functionality
IS_GET_ENABLE_AUTO_WHITEBALANCE	Returns the current AutoWhitebalance settings
IS_SET_ENABLE_AUTO_FRAMERATE	Acvtivates/deactivates the AutoFramerate functionality
IS_GET_ENABLE_AUTO_FRAMERATE	Returns the current AutoFramerate settings
IS_SET_AUTO_REFERENCE	Set setpoint for AutoGain/AutoShutter. Corresponds to a grey value of 0-255.
IS_GET_AUTO_REFERENCE	Returns the setpoint of AutoGain/AutoShutter
IS_SET_AUTO_GAIN_MAX	Set upper control level for AutoGain
IS_GET_AUTO_GAIN_MAX	Returns the upper control level of AutoGain
IS_SET_AUTO_SHUTTER_MAX	Set upper control level for AutoShutter
IS_GET_AUTO_SHUTTER_MAX	Returns upper control level of AutoShutter
IS_SET_AUTO_SPEED	Sets the speed value of the auto function.
IS_GET_AUTO_SPEED	Returns the speed value of the auto function.
IS_SET_AUTO_WB_OFFSET	Sets the offset for the red and the blue channel.
IS_GET_AUTO_WB_OFFSET	Returns the offset values of the red channel and the blue channel.
IS_SET_AUTO_WB_GAIN_RANGE	Sets the control range for Auto Whitebalance.
IS_GET_AUTO_WB_GAIN_RANGE	Returns the control range of Auto Whitebalance.
IS_SET_AUTO_WB_SPEED	Sets the speed for Auto Whitebalance.
IS_GET_AUTO_WB_SPEED	Returns the speed range of Auto Whitebalance.
IS_SET_AUTO_WB_ONCE	Sets the automatic termination of the Auto Whitebalance.
IS_GET_AUTO_WB_ONCE	Returns the current state of the Auto Whitebalance.
IS_SET_AUTO_BRIGHTNESS_ONCE	Sets the automatic termination of the Auto Brightness.
IS_GET_AUTO_BRIGHTNESS_ONCE	Returns the current state of the Auto Brightness.
pval1	Parameter (value)
pval2	Parameter (value)

The parameters *pval1* and *pval2* can take different values depending upon the used type of the parameter *param*.

Auto Brightness function

For the enable parameter variable *pval1* accepts the values 0 (inactive) and 1(active). In the case of the nominal value the value describes a grey tone of 0-255. For the control limits valid values for GAIN (0-100) and SHUTTER (exposure time) are adjusted.
If the value 0 is handed over for the parameter *IS_SET_AUTO_SHUTTER_MAX* the maximum shutter time is set. If this is changed by adjustment of the camera timing, the value is updated. However if a value within the current limits is handed over, this remains so long set, until a new value is handed over. With the readout of the shutter limit the user value or, if 0 was handed over, the maximum limit value is returned.

Auto Whitebalance function

With the settings for offset and GainRange of the Auto Whitebalance function both variables are used. In order to adjust the offset for the red and the blue channel, the offset for red is handed over as variable pval1 and the offset for blue is handed over as *pval2*. The offset can be adjusted in a range from -50 to +50. Within the specified limits *pval1* stand for the minimum and *pval2* stand for the maximum gain value. The limits can be specified in a range from 0 to 100. If the maximum gain value should be smaller than the minimum, the maximum is set on the minimum gain value and reverse.
If only one of the parameters *pval1/pval2* is to be handed over, then a NULL-pointer is to be used for the other parameter.
If the settings are to be returned, then the values are written to the addresses indicated as pval1 and pval2.

Pre-defined values for the auto features: (values obviously out of uEye.h)

IS_DEFAULT_AUTO_BRIGHT_REFERENCE	Default setpoint for AutoGain and AutoShutter
IS_MIN_AUTO_BRIGHT_REFERENCE	Minimum setpoint for AutoGain and AutoShutter- Maximum setpoint for AutoGain and AutoShutter
IS_MAX_AUTO_BRIGHT_REFERENCE	Default value for AutoSpeed
IS_DEFAULT_AUTO_SPEED	Minimum setpoint for AutoSpeed
IS_MAX_AUTO_SPEED	Maximum setpoint for AutoSpeed – currently not used
IS_DEFAULT_WB_OFFSET	Default value for Auto Whitebalance Offset
IS_MIN_WB_OFFSET	Minimum value for Auto Whitebalance Offset
IS_MAX_WB_OFFSET	Maximum value for Auto Whitebalance Offset
IS_DEFAULT_AUTO_WB_SPEED	Default value for Auto Whitebalance speed
IS_MIN_AUTO_WB_SPEED	Minimum value for Auto Whitebalance speed
IS_MAX_AUTO_WB_SPEED	Maximum value for Auto Whitebalance speed

Further possibilities of activate/deactivating the AutoBrightness functions:

AutoGain functionality can be activated with the function *is_SetHardwareGain()*, if the parameter *IS_SET_ENABLE_AUTO_GAIN* will be used instead of the first value. The auto functionality is again deactivated by setting a value (see also *4.111 is_SetHardwareGain*).
AutoExposure functionality can be activated with the function *is_SetExposureTime()*, if the parameter *IS_SET_ENABLE_AUTO_SHUTTER* will be used. The auto functionality is again deactivated by setting a value (see also *4.103 is_SetExposureTime*).

Return value

IS_SUCCESS, IS_NO_SUCCESS

Examples

Activate AutoGain:

```
Double dEnable = 1;
int ret = is_SetAutoParameter (m_hCam ,IS_SET_ENABLE_AUTO_GAIN, &dEnable, 0);
```

Set brightness setpoint to 128:

```
double soll = 128;
int ret = is_SetAutoParameter (m_hCam,IS_SET_AUTO_REFERENCE, &soll, 0);
```

Readout shutter control limit:

```
double maxShutter;
int ret = is_SetAutoParameter (m_hCam, IS_GET_AUTO_SHUTTER_MAX, & max-Shutter, 0);
```

4.87 is_SetBadPixelCorrection

Syntax

INT is_SetBadPixelCorrection (HIDS hf, INT nEnable, INT threshold)

Description

is_SetBadPixelCorrection() switches the hot pixel correction on or off. A selection among three different versions is possible.

iis_SetBadPixelCorrection() does not work if subsampling or binning is activated, or in Raw Bayer mode.

Parameters

hf	Camera handle
nEnable	
IS_BPC_DISABLE	Switches the correction off
IS_BPC_ENABLE_HARDWARE	Activates hardware correction (only UI_141x sensors), threshold parameter is used
IS_BPC_ENABLE_SOFTWARE	Activates software correction on basis of the hot pixel list deposited in the EEPROM.
IS_BPC_ENABLE_USER	Activates software correction with user defined values. The function *is_SetBadPixelCorrectionTable()* must selected first
IS_GET_BPC_MODE	Indicates the current mode.
IS_GET_BPC_THRESHOLD	Indicates the current threshold value.
threshold	Only used with UI_141x sensors in connection with parameter *IS_BPC_ENABLE_HARDWARE*. Affects the degree of the correction.

There is the possibility of linking hardware correction with a software mode; both software corrections, however, cannot be used together. Possible combinations are:

1. *IS_BPC_DISABLE*
2. *IS_BPC_ENABLE_HARDWARE*
3. *IS_BPC_ENABLE_SOFTWARE*
4. *IS_BPC_ENABLE_USER*
5. *IS_BPC_ENABLE_HARDWARE* | *IS_BPC_ENABLE_SOFTWARE*
6. *IS_BPC_ENABLE_HARDWARE* | *IS_BPC_ENABLE_USER*

Return value

IS_SUCCESS, IS_NO_SUCCESS, the current mode in connection with *IS_GET_BPC_MODE* or the current level in connection with *IS_GET_BPC_THRESHOLD*

4.88 is_SetBadPixelCorrectionTable

Syntax

INT is_SetBadPixelCorrectionTable (HIDS hf, INT nMode, WORD *pList)

Description

is_SetBadPixelCorrectionTable() sets the table with hot pixels that is used for the user defined hot pixel correction. The hotpixel correction is switched on. Each value in the table consists of one 2-byte WORD data type. The first value specifies the number of pixel coordinates within the table; this is followed by the coordinates (first X, then Y).
A table with 3 hot pixels must be structured as follows:

3	X1	Y1	X2	Y2	X3	Y3

Parameters

hf	Camera handle
nMode	
IS_SET_BADPIXEL_LIST	Sets a new user defined list. The parameter *pList* indicates a list in the previously written format.
IS_GET_LIST_SIZE	Indicates the number of pixel coordinates that there are in the user-defined list.
IS_GET_BADPIXEL_LIST	Copies the user defined table in the parameter *pList*; the memory must be reserved first.

Return value

IS_SUCCESS, IS_NO_SUCCESS, or the number or coordinates in the list with *IS_GET_LIST_SIZE*

Example

Readout the HotPixelCorrection table:

```
WORD *pList = NULL;
// Anzahl an Koordinaten in der Liste
DWORD nCount = is_SetBadPixelCorrectionTable(hf, IS_GET_LIST_SIZE, NULL);
// Speicher für komplette Liste reservieren
pList = new WORD[ 1 + 2*nCount ];
is_SetBadPixelCorrectionTable(hf, IS_GET_BADPIXEL_LIST, pList);

// Liste wieder freigeben
delete [] pList;
```


4.89 is_SetBayerConversion

Syntax

INT is_SetBayerConversion (HIDS hf, INT nMode)

Description

is_SetBayerConversion() makes it possible to select among three different colour conversion algorithms, which differ in the quality and the necessary CPU load.

Randbedingungen

Function only valid for 24-Bit, 32-Bit and Y8 colour format.

Parameters

hf	Camera handle
nMode	
IS_SET_BAYER_CV_NORMAL	Standard conversion is no longer supported. If this mode is selected the better algorithm is used.
IS_SET_BAYER_CV_BETTER	Better quality – higher CPU load
IS_SET_BAYER_CV_BEST	Highest quality – highest CPU load
IS_GET_BAYER_CV_MODE	Current settings are returned.

Return value

In connection with IS_GET_BAYER_CV_MODE the current settings are read, else IS_SUCCESS or IS_NO_SUCCESS.

4.90 is_SetBinning

Syntax

INT is_SetBinning (HIDS hf, INT mode)

Description

With *is_SetBinning()* the binning mode can be activated both in horizontal and in vertical direction. Thus the image size for each binnig direction can be halved or quartered. Depending on the sensor the sensitivity or the frame rate can be increased with activated binning. For the simultaneous activation of horizontal and vertical binning the horizontal and vertical binning parameters can be combined by a logical OR.

Parameters

hf	Camera handle
mode	
IS_BINNING_DISABLE	Deactivates binning.
IS_BINNING_2X_VERTICAL	Activates 2x vertical binning
IS_BINNING_3X_VERTICAL	Activates 3x vertical binning
IS_BINNING_4X_VERTICAL	Activates 4x vertical binning
IS_BINNING_2X_HORIZONTAL	Activates 2x horizontal binning
IS_BINNING_3X_HORIZONTAL	Activates 3x horizontal binning
IS_BINNING_4X_HORIZONTAL	Activates 4x horizontal binning
IS_GET_BINNING	Returns the current settings
IS_GET_SUPPORTED_BINNING	Returns the supported binning modes
IS_GET_BINNING_TYPE	The return value specifies whether the camera uses colour conserving Binning (IS_BINNING_COLOR) or not (IS_BINNING_MONO)

Return value

IS_SUCCESS, IS_NO_SUCCESS or the current settings with *IS_GET_BINNING*

4.91 is_SetBlCompensation

Syntax

INT is_SetBlCompensation (HIDS hf, INT nEnable, INT offset, INT reserved)

Description

is_SetBlCompensation() activates black level compensation which can be used to improve image quality if necessary. Depending on the time of the change of the compensation this affects only with the recording of the next image.

Parameters

hf	Camera handle
nEnable	
IS_BL_COMPENSATION_DISABLE	Disables the automatic compensation
IS_BL_COMPENSATION_ENABLE	Activates black level compensation using the set offset value.
IS_GET_BL_COMPENSATION	Returns the current mode.
IS_GET_BL_OFFSET	Returns the current offset.
IS_GET_BL_DEFAULT_MODE	Returns the standard mode
IS_GET_BL_DEFAULT_OFFSETE	Returns the standard offset
IS_GET_BL_SUPPORTED_MODE	Returns the supported modes Possible values: IS_BL_COMPENSATION_ENABLE The used sensor supports black level compensation IS_BL_COMPENSATION_OFFSET With the used sensor only the offset of the automatic as well as of the manual black level compensation can be set
IS_IGNORE_PARAMETER	The parameter *nEnable* is ignored.
offset	Contains the offset which is used for the compensation. Valid values lie between 0 and 255.
IS_IGNORE_PARAMETER	The parameter *offset* is ignored.

Return value

IS_SUCCESS, IS_NO_SUCCESS, current settings with IS_GET_BL_COMPENSATION, or the pre set offset with IS_GET_BL_OFFSET

4.92 is_SetBrightness

Syntax

INT is_SetBrightness (HIDS hf, INT Bright)

Description

is_SetBrightness() sets the brightness of the image digital. The parameter Bright can have a value of between 0 and 255. The adjustment of the brightness is carried out by changing the luminance value, with an offset of the Bright parameter. Bright is set at the default value of 128 (*IS_DEFAULT_BRIGHTNESS*).

Changing brightness digitally may result in gaps in the histogram. We therefore recommend changing parameters that affect image recording directly, rather than *is_SetContrast()*. These include *is_SetHardwareGain()* and *is_SetExposureTime().*

Parameters

hf	Camera handle
Bright	
0...255	Brightness in the range of 0 and 255
IS_GET_BRIGHTNESS	Retrieves the current setting

Return value

Current setting when called with *IS_GET_BRIGHTNESS* else *IS_SUCCESS, IS_NO_SUCCESS*

4.93 is_SetCameraID

Syntax

INT is_SetCameraID (HIDS hf, INT nID)

Description

is_SetCameraID() makes it possible to assign the ID number, which can be used to open the Camera (see also *4.56 is_InitCamera*).

Parameters

hf	Camera handle
nID	
1...254	New camera ID
IS_GET_CAMERA_ID	Returns the current ID

Return value

In connection with *IS_GET_CAMERA_ID* the current ID is returned, else *IS_SUCCESS, IS_NO_SUCCESS*

4.94 is_SetColorCorrection

Syntax

INT is_SetColorCorrection (HIDS hf, INT nEnable, double *factors)

Description

In order to receive a better colour reproduction, with *is_SetColorCorrection()* a colour matching can be activated.

After changing these parameters, a white balance adjustment must be done (see *4.86 is_SetAuto-Parameter*).

Parameters

hf	Camera handle
nEnable	
IS_CCOR_ENABLE_NORMAL	Activates normal colour correction. Replaces the old parameter IS_CCOR_ENABLE.
IS_CCOR_ENABLE_BG40_ENHANCED	Activates colour correction for cameras with IR-cut filter glasses of type BG40.
IS_CCOR_ENABLE_HQ_ENHANCED	Activates colour correction for cameras with IR-cut filter glasses of type HQ.
IS_GET_DEFAULT_CCOR_MODE	Returns the default colour correction mode.
IS_GET_SUPPORTED_CCOR_MODE	Returns all supported colour correction modes. See return value.
IS_CCOR_DISABLE	Deactivates colour correction
IS_GET_CCOR_MODE	Returns current settings
factors	Reserved

Return value

Current settings when called with *IS_GET_CCOR_MODE,* else
IS_SUCCESS, IS_NO_SUCCESS

With parameter *IS_GET_SUPPORTED_CCOR_MODE* the supported modes are returned for colour cameras. The return value represents the following values combined by logical OR:

IS_CCOR_ENABLE_NORMAL = 1 (binary 001)

IS_CCOR_ENABLE_BG40_ENHANCED = 2 (binary 010)

IS_CCOR_ENABLE_HQ_ENHANCED = 4 (binary 100)
For monochrome cameras the return value is zero.

With parameter *IS_GET_DEFAULT_CCOR_MODE* the default mode is returned for colour cameras. For monochrome cameras the return value is zero.

4.95 is_SetColorMode

Syntax

INT is_SetColorMode (HIDS hf, INT Mode)

Description

is_SetColorMode() sets the required colour mode with which the image data is to be saved or displayed by the VGA board. For the first case it is important that, depending upon the colour mode which is used, the allocated image memory is large enough. A 24 bit colour image requires three times as much memory as an 8 bit monochrome image. When accessing the image data, it is important to know how the memory is arranged in each of the memory modes (see *2.4 Colour Formats*). Incorrect interpretation of the memory contents leads to incorrect results. With the direct transfer to the VGA card's image memory, it is important to ensure that the display settings correspond to those of the colour mode. Under certain circumstances the images can be displayed with either the incorrect colours or they can become so distorted that it is impossible to recognize what is actually being displayed.

Parameters

hf	Camera handle
Mode	
IS_SET_CM_RGB32	32 bit true colour mode; R-G-B-Dummy
IS_SET_CM_RGB24	24 bit true colour mode; R-G-B
IS_SET_CM_RGB16	Hi colour mode; 5 R - 6 G - 5 B
IS_SET_CM_RGB15	Hi colour mode; 5 R - 5 G - 5 B
IS_SET_CM_Y8	8 bit monochrome image
IS_SET_CM_BAYER	Raw *Bayer* data at colour sensors
IS_SET_CM_UYVY	16 Bit UYVY Format
IS_GET_COLOR_MODE	Retrieval of current setting

Return value

Current setting when called with *IS_GET_COLOR_MODE,* else *IS_SUCCESS, IS_NO_SUCCESS, IS_INVALID_MODE on standby.*

4.96 is_SetContrast

Syntax

INT is_SetContrast (HIDS hf, INT Cont)

Description

is_SetContrast() changes the images contrast (luminance amplification) digital in the range of 0% to 200%. The default value for *Cont* is 215 (*IS_DEFAULT_CONTRAST*).

Changing brightness digitally may result in gaps in the histogram. We therefore recommend changing parameters that affect image recording directly, rather than *is_SetContrast()*. These include *is_SetHardwareGain()* and *is_SetExposureTime().*

Parameters

hf	Camera handle
Cont	
0...511	Sets contrast
IS_GET_CONTRAST	Retrieval of current setting

Return value

Current settings when called with *IS_GET_CONTRAST* else *IS_SUCCESS, IS_NO_SUCCESS.*

4.97 is_SetConvertParam

Syntax

INT is_SetConvertParam(HIDS hf, BOOL ColorCorrection, INT BayerConversionMode, INT ColorMode, INT Gamma, double *WhiteBalanceMultipliers)

Description

Sets the conversion parameters for the Raw *Bayer* image that will be converted to other format with the function *is_ConvertImage()*. This function sets:

- colour correction
- *Bayer* conversion mode
- colour mode
- gamma
- white balance multipliers

Parameters

hf	Camera handle
ColorCorrection	Activates/deactivates the colour correction
BayerConversionMode	Sets the *Bayer* conversion mode
IS_SET_BAYER_CV_BETTER	Better quality
IS_SET_BAYER_CV_BEST	Highest quality – higher CPU load
ColorMode	sets the colour mode of the output image
IS_SET_CM_RGB32	32 bit true colour mode; R-G-B-Dummy
IS_SET_CM_RGB24	24 bit true colour mode; R-G-B
IS_SET_CM_RGB16	Hi colour mode; 5 R - 6 G - 5 B
IS_SET_CM_RGB15	Hi colour mode; 5 R - 5 G - 5 B
IS_SET_CM_Y8	8 bit monochrome image
IS_SET_CM_UYVY	16 Bit UYVY Format
Gamma	Gamma value multiplied with 100. - Range: [1…1000]
WhiteBalanceMultipliers	pointer to a double array with the red, green and blue gains

Return value

IS_SUCCESS, IS_NO_SUCCESS, IS_INVALID_COLOR_FORMAT or *IS_INVALID_PARAMETER*

Example

See *4.6 is_ConvertImage*

4.98 is_SetDDUpdateTime

Syntax

INT is_SetDDUpdateTime (HIDS hf, INT ms)

Description

is_SetDDUpdateTime() sets the timer interval for the update cycle of the video image in DirectDraw BackBuffer mode. Valid values are between 20ms and 2000ms.

Parameters

hf	Camera handle
ms	Time in milliseconds

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.99 is_SetDisplayMode

Syntax

INT is_SetDisplayMode (HIDS hf, INT Mode)

Description

is_SetDisplayMode() defines the way in which images are displayed on the screen. For real live video plus overlay, the DirectDraw overlay surface mode has been introduced. The availability of this mode depends upon the type of VGA card used.
To use any DirectDraw mode colour mode has to be set to UYVY.
Using a VGA resolution of 640x480 the image size must be limited to the same size before starting a Direct-Draw mode. For this purpose function *is_SetImageSize()* can be used.
For example: VGA with 1024x768x16 = 1.5 MB -> OverlayBuffer with up to 1.5 MB

The uEye SDK is *thread safe* in general. The API function calls are all done in critical sections. Due to internal structures of DirectDraw we strongly recommend to call the following functions from only one thread to avoid unpredictable behaviour of your application.

- *is_InitCamera()*
- *is_SetDisplayMode()*
- *is_ExitCamera()*

Parameters

hf	Camera handle
Mode	
IS_SET_DM_DIB	Acquire image in image memory (RAM) (no automatic display – display possible with *is_RenderBitmap()*!)
IS_SET_DM_DIRECTDRAW | IS_SET_DM_BACKBUFFER	DirectDraw BackBuffer mode
IS_SET_DM_DIRECTDRAW | IS_SET_DM_ALLOW_OVERLAY	DirectDraw Overlay Surface mode
DirectDraw overlay surface extension	
IS_SET_DM_ALLOW_SCALING	Real time scaling in overlay surface mode
Return mode	
IS_GET_DISPLAY_MODE	Return the current settings

Return value

Current setting with *IS_GET_DISPLAY_MODE*, else
IS_SUCCESS, IS_NO_SUCCESS

Examples

Resolution 1024x768x16 = 1,5 MB -> OverlayBuffer up to 1,5 MB

```
is_SetDisplayMode (hf, Mode);
```

Bitmap mode (digitized in system memory):

```
Mode = IS_SET_DM_DIB
```

DirectDraw BackBuffer mode

```
Mode = IS_SET_DM_DIRECTDRAW
```

DirectDraw Overlay-Surface mode (best Live-Overlay):

```
Mode =  IS_SET_DM_DIRECTDRAW  |
        IS_SET_DM_ALLOW_OVERLAY
```

Allows automatic scaling to the size of window:

```
Mode =  IS_SET_DM_DIRECTDRAW  |     IS_SET_DM_ALLOW_OVERLAY  |
        IS_SET_DM_ALLOW_SCALING
```

4.100 is_SetDisplayPos

Syntax

INT is_SetDisplayPos (HIDS hf, INT x, INT y)

Description

The function *is_SetDisplayPos()* enables the offset for the image output, produced with *is_RenderBitmap*. The offset takes place by the parameters *x* and *y*.

Parameters

hf	Camera handle
x	Offset in x-direction
y	Offset in y-direction

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.101 is_SetEdgeEnhancement

Syntax

INT is_SetEdgeEnhancement (HIDS hf, INT nEnable)

Description

Due to the colour conversion of the *Bayer* format the original edges can become out of focus. The activation of the digital edge enhancer, counteract this effect. Two settings (*IS_EDGE_EN_STRONG, IS_EDGE_EN_WEAK*) with different effects are available. Using this function increases CPU load.

Parameters

hf	Camera handle
nEnable	
IS_EDGE_EN_DISABLE	Deactivates the edge enhancer
IS_EDGE_EN_STRONG	Activates the strong edge enhancer
IS_EDGE_EN_WEAK	Activates the weak edge enhancer
IS_GET_EDGE_ENHANCEMENT	Returns the current settings

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.102 is_SetErrorReport

Syntax

INT is_SetErrorReport (HIDS hf, INT Mode)

Description

Toggles the error report mode. When the error report mode is active, errors are displayed in a dialogue box. When you quit the dialogue box with Cancel, the error reporting is automatically deactivated. If the error report mode is not active, errors may be called up with *is_GetError()*. The camera handle is not analysed, *is_SetErrorReport()* is working global, not device related. *is_SetErrorReport()* can be called up before function *is_InitCamera()*.

Parameters

hf	Camera handle or NULL
Mode	
IS_DISABLE_ERR_REP	Disable error report
IS_ENABLE_ERR_REP	Enable error report

Return value

Current setting when called with *IS_GET_ERR_REP_MODE* else *IS_SUCCESS, IS_NO_SUCCESS.*

4.103 is_SetExposureTime

Syntax

INT is_SetExposureTime (HIDS hf, double EXP, double* newEXP)

Description

The function *is_SetExposureTime()* sets the with EXP indicated exposure time in ms. Since this is adjustable only in multiples of the time, a line needs, the actually used time can deviate from the desired value.

The actual duration adjusted after the call of this function is readout with the parameter *newEXP*. By changing the window size or the readout timing (pixel clock) the exposure time set before is changed also. Therefore *is_SetExposureTime()* must be called again thereafter.

Exposure-time interacting functions:

is_SetImageSize()

is_SetPixelClock()

is_SetFrameRate() (only if the new image time will be shorter than the exposure time)

Which minimum and maximum values are possible and the dependence of the individual sensors is explained in detail in the description to the uEye timing.
Depending on the time of the change of the exposure time this affects only with the recording of the next image.

The sensor of UI-146x-C does not allow changes of the exposure time while in trigger mode. If *is_SetExposureTime()* is called while in trigger mode, the sensor will temporarily be switched to freerun. This results in longer holding time (depending on the frame rate) at function call.

Parameters

hf	Camera handle
EXP	New desired exposure-time.
IS_GET_EXPOSURE_TIME	Returns the actual exposure-time through parameter *newEXP*. If EXP = 0.0 is passed, an exposure time of (1/frame rate) is used.
IS_GET_DEFAULT_EXPOSURE	Returns the default exposure time
newEXP	Actual exposure time.

In the case of use of the constant *IS_SET_ENABLE_AUTO_SHUTTER* for the parameter *EXP* the AutoExposure functionality is activated. Setting a value will deactivate the functionality. (see also *4.86 is_SetAutoParameter*).

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.104 is_SetExternalTrigger

Syntax

INT is_SetExternalTrigger (HIDS hf, INT nTriggerMode)

Description

is_SetExternalTrigger() activates the trigger input. If the camera is on standby, it will exit standby mode and start trigger mode. The function call sets the edge on which an action takes place. When the trigger input is active, *is_FreezeVideo()* function waits for an input of the trigger signal.

- Action on high low edge High-Low (TTL) *IS_SET_TRIG_HI_LO*
  - Action on low high edge *IS_SET_TRIG_LO_HI*
  - Deactivates trigger *IS_SET_TRIG_OFF*

If without trigger functionality (*IS_SET_TRIG_OFF*) the level at the trigger input can be queried statically. Thus the trigger input is used as digital input. If this option is set, the camera will switch to freerun mode.

Due to the time response of the UI-144x-xx with this model the exposure time must be set to the value (1/frame rate) in the trigger mode.

Due to the hardware of board level *uEye LEs* only the falling edge can be triggered.

Parameters

hf	Camera handle
nTriggerMode	
IS_SET_TRIG_OFF	Disable trigger processing.
IS_SET_TRIG_HI_LO	Sets active trigger flank to the falling edge.
IS_SET_TRIG_LO_HI	Sets active trigger flank to the rising edge.
IS_SET_TRIG_SOFTWARE	Activate the software trigger mode; With call of the function *is_FreezeVideo()* the camera is triggered and supplies a picture.
IS_GET_EXTERNALTRIGGER	Retrieval of trigger mode settings
IS_GET_TRIGGER_STATUS	Return the current level at the trigger input.
IS_GET_SUPPORTED_TRIGGER_MODE	Return the supported trigger modes.

Return value

IS_SUCCESS, IS_NO_SUCCESS or current setting with *IS_GET_EXTERNALTRIGGER IS_SET_TRIG_SOFTWARE* | *IS_SET_TRIG_HI_LO* | *IS_SET_TRIG_LO_HI* using *IS_GET_SUPPORTED_TRIGGER_MODE*

Example

Activate trigger mode and set High-Active flash mode.

```
is_SetExternalTrigger (hf, IS_SET_TRIG_SOFTWARE);
is_SetFlashStrobe (hf, IS_SET_FLASH_HI_ACTIVE);
is_FreezeVideo (hf, IS_WAIT);
```


4.105 is_SetFlashDelay

Syntax

INT is_SetFlashDelay (HIDS hf, ULONG ulDelay, ULONG ulDuration)

Description

is_SetFlashDelay() allows you to set the time by which flash activation is delayed. In addition, the duration of the flash can be set. This allows a global flash function to be implemented where all rows of a rolling shutter sensor are exposed. Further the start of the flash can be adapted to the start of the exposure with global shutters sensors in freerun mode (see also *4.37 is_GetGlobalFlashDelays*).
If 0 is given for *ulDelay*, no delay will be used for the start of the flash. If only the flash is to be activated with a delay, but not deactivated before the end of the image, 0 can be passed for the flash duration *ulDuration*.
With the delay it is to be noted that global shutter sensors in freerun mode or triggered mode, as well as rolling shutter sensors behave differently. See also chapter Digital output (Flash/strobe) in the user manual.

Parameters

hf	Camera handle
ulDelay	Time the flash is delayed (in µs)
IS_GET_FLASH_DELAY	Returns the current delay time.
IS_GET_FLASH_DURATION	Returns the current flash duration time.
IS_GET_MIN_FLASH_DELAY	Returns the minimum adjustable value for the flash delay.
IS_GET_MIN_FLASH_DURATION	Returns the minimum adjustable value for the flash duration.
IS_GET_MAX_FLASH_DELAY	Returns the maximum adjustable value for the flash delay..
IS_GET_MAX_FLASH_DURATION	Returns the maximum adjustable value for the flash duration.
IS_GET_FLASH_DELAY_GRANULARITY	The resolution of the adjustable flash delay.
IS_GET_FLASH_DURATION_GRANULARITY	
ulDuration	The resolution of the adjustable flash duration. Time in that is switched on lightning (in µs)

Return value

IS_SUCCESS, IS_NO_SUCCESS, current settings in connection with
IS_GET_FLASH_DELAY or
IS_GET_FLASH_DURATION

Examples

Global shutter sensor in freerun mode:

```
is_SetFrameRate(m_hCamera, 10.0, &dFPS);
is_SetExposureTime(m_hCamera, 40.0, &dExp);
is_SetFlashDelay(m_hCamera, 65000, 15000);
is_SetFlashStrobe(m_hCamera, IS_SET_FLASH_HI_ACTIVE_FREERUN, 0);
```

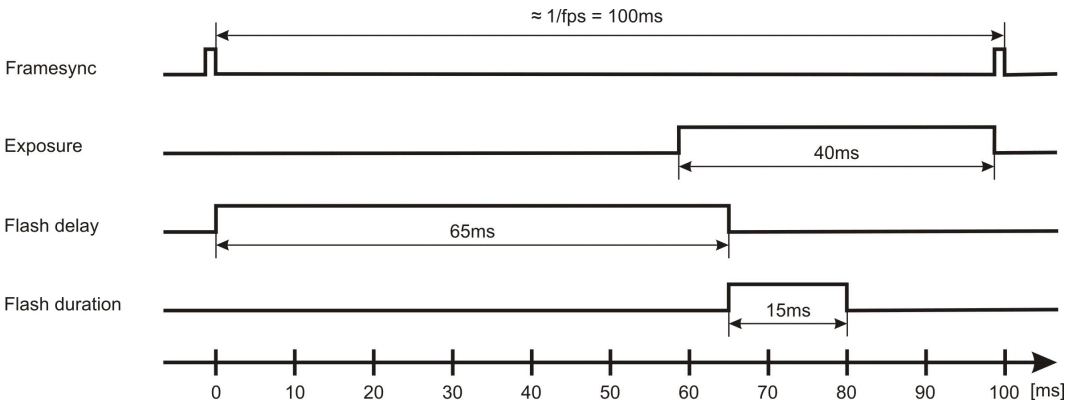

Figure 13: Timing for flash/strobe with gobal shutter sensor in freerun mode

Global shutter sensor in triggered mode:

```
is_SetExternalTrigger(m_hCamera, IS_SET_TRIG_LO_HI);
is_SetPixelClock(m_hCamera, 20);
is_SetExposureTime(m_hCamera, 70, &dExp);
is_SetFlashDelay(m_hCamera, 20000, 40000);
is_SetFlashStrobe(m_hCamera, IS_SET_FLASH_HI_ACTIVE, 0);
```

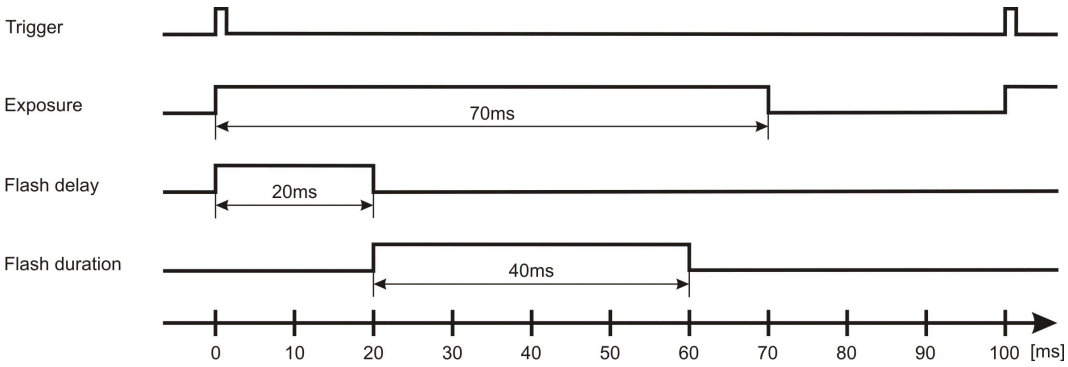

Figure 14: Timing for flash/strobe with gobal shutter sensor in triggered mode

4.106 is_SetFlashStrobe

Syntax

INT is_SetFlashStrobe (HIDS hf, INT nMode, INT nLine)

Description

is_SetFlashStrobe() activates the flash strobe. Using the *nMode* parameter the actuation can be activated or deactivated. You can also set the active level (high or low). By default, the strobe is set to high-active.
The constants *IS_SET_FLASH_HIGH* and *IS_SET_FLASH_LOW* allow the strobe output to be used as a digital output.
Flash duration and delay can be set with the *is_SetFlashDelay()* function (see *4.105 is_SetFlashDelay*). For cameras with Rolling Shutter sensors it is recommended to use the values from the function *is_GetGlobalFlashDelays()* (see also 4.37 is_GetGlobalFlashDelays) as flash delay and flash duration, because otherwise unexpected results can occour. For flash output in capture mode this is especially to be attended.
For the modes high-active and low-active the respective parameter must be selected suitably for the camera mode.
IS_SET_FLASH_LO_ACTIVE and *IS_SET_FLASH_HI_ACTIVE* are used in trigger mode (see also *4.104 is_SetExternalTrigger*).
IS_SET_FLASH_LO_ACTIVE_FREERUN and *IS_SET_FLASH_HI_ACTIVE_FREERUN* are working only in capture mode (see 4.4 is_CaptureVideo).

Parameters

hf	Camera handle
nMode	
IS_SET_FLASH_OFF	Disable the strobe output
IS_SET_FLASH_LO_ACTIVE	Set the strobe output to low-active mode
IS_SET_FLASH_HI_ACTIVE	Set the strobe output to high-active mode
IS_SET_FLASH_LO_ACTIVE_FREERUN	Set the strobe output to low-active in freerun mode
IS_SET_FLASH_HI_ACTIVE_FREERUN	Set the strobe output to high-active in freerun mode .
IS_SET_FLASH_HIGH	Set the strobe output HIGH
IS_SET_FLASH_LOW	Set the strobe output LOW
IS_GET_FLASHSTROBE_MODE	Return the current flash strobe mode
IS_SET_FLASH_IO_1	Enable flash on I/O port 1 (only uEyeLE)
IS_SET_FLASH_IO_2	Enable flash on I/O port 2 (only uEyeLE)
IS_GET_SUPPORTED_FLASH_IO	Return the I/O ports which support flash (only uEyeLE)
nLine	reserved

The parameters *IS_SET_FLASH_LO_ACTIVE_FREERUN* and *IS_SET_FLASH_HI_ACTIVE_-FREERUN* are not supported with the cameras UI-1440 and UI-1210.

Return value

IS_SUCCESS, IS_NO_SUCCESS or the current mode when used with *IS_GET_FLASHSTROBE_MODE*

Example

Activate trigger mode and set High-Active flash mode.

```
is_SetExternalTrigger (hf, IS_SET_TRIG_SOFTWARE);
is_SetFlashStrobe (hf, IS_SET_FLASH_HI_ACTIVE, 0);
is_FreezeVideo (hf, IS_WAIT);
```


4.107 is_SetFrameRate

Syntax

INT is_SetFrameRate (HIDS hf, double FPS, double* newFPS)

Description

is_SetFrameRate() sets the number of frames/s the sensor shall work with. If the frame rate is too highly adjusted, not every frame can be read in and the actual frame rate drops.
Like exposure time, it is not possible to adjust any value. Therefore the new frame rate is returned after the function call with the parameter *newFPS*. You can find more exact details in the description to the uEye timing.
Similarly to the Exposure time changes at the window size or at the pixel clock affect the rram-e rate.

Frame rate affecting functions:

- *is_SetImageSize()*
- *is_SetPixelClock()*

Parameters

hf	Camera handle
FPS	Number of pictures per second.
IS_GET_FRAMERATE	Returns the actual frame rate with the parameter *newFPS*.
IS_GET_DEFAULT_FRAMERATE	Returns the standard frame rate
newFPS	Returns the frame rate value.

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.108 is_SetGainBoost

Syntax

INT is_SetGainBoost (HIDS hf, INT mode)

Description

is_SetGainBoost() activates or deactivates an extra hardware camera gain. This feature is supported by the following cameras:1220-C/M, 1440-C/M, 1540-C/M, 1450-C, 1460-C, 1480-C and monochrome CCD models.

- UI-1220-C/M
- UI-1440-C/M
- UI-1540-C/M
- UI-1450-C
- UI-1460-C
- UI-1480-C
- monochrome CCD models.

Parameters

hf	Camera handle
mode	
IS_GET_GAINBOOST	Returns the current mode of the gain boost, or *IS_NOT_SUPPORTED* if the camera doesn't support this features
IS_SET_GAINBOOST_ON	Activates the gain boost
IS_SET_GAINBOOST_OFF	Deactivates the gain boost
IS_GET_SUPPORTED_GAINBOOST	Returns *IS_SET_GAINBOOST_ON* if this features is supported, otherwise returns *IS_SET_GAINBOOST_OFF*

Return value

Current setting when called with *IS_GET_GAINBOOST*, else *IS_NOT_SUPPORTED, IS_SUCCESS* or *IS_NO_SUCCESS*

4.109 is_SetGamma

Syntax

INT is_SetGamma (HIDS hf, INT Gamma)

Description

is_SetGamma() sets the value for the digital gamma correction. The valid range of values is between 0.01 and 10. However the parameter *gamma* must be fixed as integer value with a range from 1 to 1000 (gamma value * 100). The default value for *gamma* is 100, which equals a gamma of 1.0 (*IS_DEFAULT_GAMMA*).

Parameters

hf	Camera handle
Gamma	Gamma value multiplied with 100. - Range: [1…1000]
IS_GET_GAMMA	Returns the current settings

Return value

Current settings in connection with *IS_GET_BRIGHTNESS*, else *IS_SUCCESS, IS_NO_SUCCESS*

Example

Set Gamma value to 1.42:

```
ret = SetGamma(hf, 142);
```

4.110 is_SetGlobalShutter

Syntax

INT is_SetGlobalShutter (HIDS hf, INT mode)

Description

is_SetGlobalShutter() activates or deactivates the global start shutter. This functionality is only supported by the 1480-C.

Parameters

hf	Camera handle
mode	
IS_GET_GLOBAL_SHUTTER	Returns the current mode of the global Shutter, or *IS_NOT_SUPPORTED* if the camera doesn't support this features.
IS_SET_GLOBAL_SHUTTER_ON	Activates the Global Shutter.
IS_SET_GLOBAL_SHUTTER_OFF	Deactivates the Global Shutter.
IS_GET_SUPPORTED_GLOBAL_SHUTTER	Returns *IS_SET_GLOBAL_SHUTTER_ON* if this features is supported, otherwise *IS_SET_GLOBAL_SHUTTER_OFF*

Return value

Current setting when called with *IS_GET_GLOBAL_SHUTTER, else IS_NOT_SUPPORTED, IS_SUCCESS* or *IS_NO_SUCCESS*

4.111 is_SetHardwareGain

Syntax

INT is_SetHardwareGain (HIDS hf, INT nMaster, INT nRed, INT nGreen, INT nBlue)

Description

is_SetHardwareGain() controls the camera integrated amplifier. They can be tuned between 0% - 100% independent of each other. *is_GetSensorInfo()* returns the available amplifiers. Depending on the time of the change of the hardware gain this affects only with the recording of the next image.

Parameters

hf	Camera handle
nMaster	Entire amplification.
IS_IGNORE_PARAMETER	No changes to master-amplifier
IS_GET_MASTER_GAIN	Returns master amplification
IS_GET_RED_GAIN	Returns red amplification
IS_GET_GREEN_GAIN	Returns green amplification
IS_GET_BLUE_GAIN	Returns blue amplification
IS_GET_DEFAULT_MASTER	Returns standard master amplification
IS_GET_ DEFAULT_RED	Returns standard red amplification
IS_GET_ DEFAULT_GREEN	Returns standard green amplification
IS_GET_ DEFAULT_BLUE	Returns standard blue amplification
nRed	Red channel
IS_IGNORE_PARAMETER	No changes to red-amplifier
nGreen	Green channel
IS_IGNORE_PARAMETER	No changes to green-amplifier
nBlue	Blue channel
IS_IGNORE_PARAMETER	No changes to blue-amplifier

In the case of use of the constant *IS_SET_ENABLE_AUTO_GAIN* for the parameter *EXP* the Auto-Gain functionality is activated. Setting a value will deactivate the functionality. (see also *4.86 is_SetAutoParameter*).

The values for the default settings for red, green and blue gain depend on the colour correction mode used. When the colour correction mode is changed it is possible that the default values for red, green and blue gain will also change (see also *4.94 is_SetColorCorrection*).

Return value

Current setting when called with *IS_GET_MASTER_GAIN, IS_GET_RED_GAIN, IS_GET_GREEN_GAIN, IS_GET_BLUE_GAIN,* else
IS_SUCCESS or *IS_NO_SUCCESS, IS_INVALID_MODE on standby*

4.112 is_SetHardwareGamma

Syntax

INT is_SetHardwareGamma (HIDS hf, INT nMode)

Description

is_SetHardwareGamma() activates/deactivates the gamma control of the camera.

Parameters

hf	Camera handle
nMode	
IS_GET_HW_SUPPORTED_GAMMA	IS_SET_HW_GAMMA_ON Gamma control is supported by the camera. IS_SET_HW_GAMMA_OFF Gamma control is not suported by the camera.
IS_SET_HW_GAMMA_OFF	Activates gamma control.
IS_SET_HW_GAMMA_ON	Deactivates gamma control.
IS_GET_HW_GAMMA	Returns the current settings.

Return value

IS_SUCCESS, IS_NO_SUCCESS or *IS_NOT_SUPPORTED*

4.113 is_SetHdrKneepoints

Syntax

INT is_SetHdrKneepoints (HIDS hf, KNEEPOINT_ARRAY* KneepointArray, INT KneepointArraySize)

Description

Some sensors support HDR mode (High Dynamic Range). HDR mode can be activated/deactivated with *is_EnableHdr()*. Use *is_SetHdrKneepoints()* to see the knee points.
The x value of a knee point indicates the first exposure phase in percent of the current exposure time. The y value gives the proportion of maximum pixel intensity in percent.
A setting of x = 60, y = 80 would therefore produce the following results:
The first exposure phase lasts for 60% of the set exposure time. In this first exposure phase, all pixels are exposed to at most 80% of maximum pixel intensity and remain at 80% until this phase is over. In the second exposure phase, they are exposed again and may reach the full pixel intensity.
If two knee points are used, two Y limits can be set at which the exposure is interrupted. Two corresponding time points are determined on the X axis.

Currently, only the UI-122X-X and UI-522X-X models support HDR.
For cameras of types UI-122X-C and UI-522X-C, the RGB gains must be set to the same values to ensure accurate colour rendition in HDR mode.

Parameters

hf	Camera handle
KneepointArray	Pointer to a field with the following parameters:
INT NumberOfUsedKneepoints	Number of knee points used
KNEEPOINT Kneepoint[10]	Knee point

Return value

IS_SUCCESS or *IS_NO_SUCCESS* for supported sensor types
IS_NOT_SUPPORTED for unsupported sensor types

4.114 is_SetHWGainFactor

Syntax

INT is_SetHWGainFactor (HIDS hf, INT nMode, INT nFactor)

Description

The function *is_SetHWGainFactor()* is used for the control of the amplifier in the camera. These can be adjusted independently, from minimum to maximum gain. *is_GetSensorInfo()* returns the available amplifiers.
For the conversion of a gain value from the function *is_SetHardwareGain()* the parameter *nMode* can be set to one of the *IS_INQUIRE_x_FACTOR* values. In this case the range of values for *nFactor* is between 0 and 100.
For setting the gain the parameter *nFactor* must be defined with *IS_GET_x_GAIN_FACTOR* as an integer value with a range from 100 to maximum. The maximum can be acquired with *IS_INQUIRE_x_FACTOR* and a value of 100 for *nFactor*. A gain factor with the value 100 correspond to no amplification, a value of 200 to an amplification factor of two and so on.
The return value corresponds to the adjusted gain. This can deviate from the desired value, because only certain values can be set. Always the gain value is set which is next to the desired value.
Depending on time of the change of the gain value this affects only with the recording of the next image.

Parameters

hf	Camera handle
nMode	
IS_GET_MASTER_GAIN_FACTOR	Returns the master gain value
IS_GET_RED_GAIN_FACTOR	Returns the red gain value
IS_GET_GREEN_GAIN_FACTOR	Returns the green gain value
IS_GET_BLUE_GAIN_FACTOR	Returns the blue gain value
IS_SET_MASTER_GAIN_FACTOR	Sets the master gain value
IS_SET_RED_GAIN_FACTOR	Sets the red gain value
IS_SET_GREEN_GAIN_FACTOR	Sets the green gain value
IS_SET_BLUE_GAIN_FACTOR	Sets the blue gain value
IS_GET_DEFAULT_MASTER_GAIN_FACTOR	Returns the standard master gain value
IS_GET_DEFAULT_RED_GAIN_FACTOR	Returns the standard red gain value
IS_GET_DEFAULT_GREEN_GAIN_FACTOR	Returns the standard green gain value
IS_GET_DEFAULT_BLUE_GAIN_FACTOR	Returns the standard blue gain value
IS_INQUIRE_MASTER_GAIN_FACTOR	Conversion of the index value of the master gain
IS_INQUIRE_RED_GAIN_FACTOR	Conversion of the index value of the red gain
IS_INQUIRE_GREEN_GAIN_FACTOR	Conversion of the index value of the green gain

IS_INQUIRE_BLUE_GAIN_FACTOR	Conversion of the index value of the blue gain
nFactor	Gain value

Return value

Current settings in connection with *IS_GET_MASTER_GAIN_FACTOR, IS_GET_RED_GAIN_FACTOR, IS_GET_ GREEN_GAIN_FACTOR, IS_GET_BLUE_GAIN_FACTOR.*

Adjusted settings after the use of *IS_SET_MASTER_GAIN_FACTOR, IS_SET_RED_GAIN_FACTOR, IS_SET_ GREEN_GAIN_FACTOR, IS_SET_BLUE_GAIN_FACTOR.*

Standard settings after the use of *IS_GET_DEFAULT_MASTER_GAIN_FACTOR, IS_GET_DEFAULT_RED_GAIN_FACTOR, IS_GET_DEFAULT_ GREEN_GAIN_FACTOR, IS_GET_DEFAULT_BLUE_GAIN_FACTOR.*

Converted gain index after the use of *IS_INQUIRE_ MAS-TER_GAIN_FACTOR, IS_INQUIRE_ RED_GAIN_FACTOR, IS_INQUIRE _ GREEN_GAIN_FACTOR, IS_INQUIRE _BLUE_GAIN_FACTOR.*

Example

Set master gain value to 3.57:

```
ret = is_SetHWGainFactor(hf, IS_SET_MASTER_GAIN_FACTOR, 357);
//ret hat für die UI-1460-C den Wert 363
```

Read the maximal gain factor of the red channel:

```
ret = is_SetHWGainFactor(hf, IS_INQUIRE_RED_GAIN_FACTOR, 100);
//ret hat für die UI-1460-C den Wert 725
```


4.115 is_SetHwnd

Syntax

INT is_SetHwnd (HIDS hf, HWND hwnd)

Description

is_SetHwnd() sets a new window handle for the image output with DirectDraw. The new handle and the image output is only activated when *is_SetDisplayMode()* is called.

Parameters

hf	Camera handle
hwnd	Handle to a window

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.116 is_SetImageAOI

Syntax

INT is_SetImageAOI (HIDS hf, INT xPos, INT yPos, INT width, INT height)

Description

is_SetImageAOI() is completely replaced by the function *is_SetAOI()* ersetzt (see *4.84 is_SetAOI*).

Parameters

hf	Camera handle
xPos	x-position of the upper left corner
yPos	y-position of the upper left corner
width	image width
height	image height

Return value

IS_SUCCESS, IS_NO_SUCCESS, IS_INVALID_MODE on standby

4.117 is_SetImageMem

Syntax

INT is_SetImageMem (HIDS hf, char* pcImgMem, INT id)

Description

is_SetImageMem() sets the allocated image memory to active memory. Only an active image memory can receive image data. After calling *is_SetImageMem()* the function *is_SetImageSize()* must follow to set the image size of the active memory, if compatibility with FALCON is required. A pointer from function *is_AllocImgMem()* has to be given to parameter *pcImgMem*.
Every deviant pointer will lead to an error message! Repeated handover of the same pointer is not allowed.

In DirectDraw mode it is not necessary to set the image memory!

Parameters

hf	Camera handle
pcImgMem	Pointer to the beginning of the image memory
id	ID for this memory

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.118 is_SetImagePos

Syntax

INT is_SetImagePos (HIDS hf, INT x, INT y)

Description

is_SetImagePos() defines the starting point of the window. A section can be cut out of the full video image using the function *is_SetImageSize()*. To avoid the display area falling outside the image area, the order of function calls must be selected carefully. Starting from the original, the order described below is therefore absolutely mandatory:

1. *is_SetImageSize()*
2. *is_SetImagePos()*

The function *is_SetAOI()* (see *4.84 is_SetAOI*) combines the two functions. With *is_SetAOI()* the position and the size of an AOI can be set with one function call.
The parameters *x* and *y* represent an offset in relation to the left upper corner of the image.
The cut-out window is copied to the start of the memory. If the image is to be copied to the same offset within the memory, the new position can be logical OR involved with the parameters *IS_SET_IMAGE_POS_X_ABS* and *IS_SET_IMAGE_POS_Y_ABS*.

Example

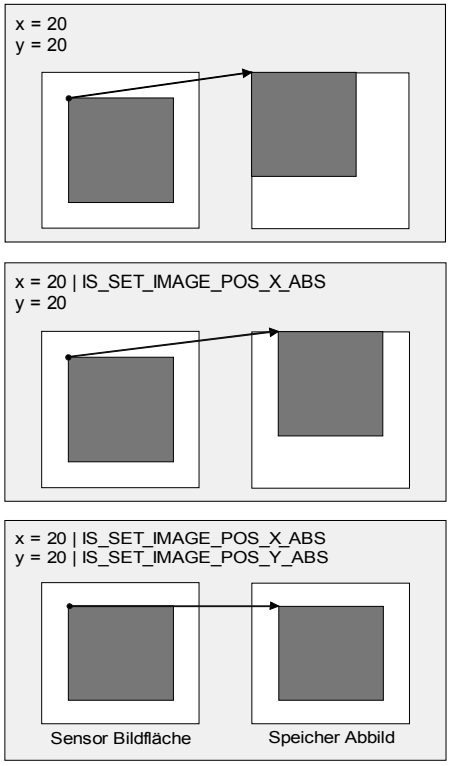

Figure 15: Example is_SetImagePos

Boundary conditions

See *4.84 is_SetAOI*.

Parameters

hf	Camera handle
x	
0...xMax	Horizontal position
IS_GET_IMAGE_POS_X	Retrieval of current X-position
IS_GET_IMAGE_POS_X_ABS	Retrieval whether the X-position for the current memory is assumed
IS_GET_IMAGE_POS_X_MIN	Smallest value for the horizontal AOI position
IS_GET_IMAGE_POS_X_MAX	Largest value for the horizontal AOI position
IS_GET_IMAGE_POS_X_INC	Increment for the horizontal AOI position
IS_GET_IMAGE_POS_Y	Retrieval of current Y-position
IS_SET_IMAGE_POS_Y_ABS	Retrieval whether the X-position for the current memory is assumed
IS_GET_IMAGE_POS_Y_MIN	Smallest value for the vertical AOI position
IS_GET_IMAGE_POS_Y_MAX	Largest value for the vertical AOI position
IS_GET_IMAGE_POS_Y_INC	Increment for the vertical AOI position
y	
0...yMax	Vertical position

Return value

Current settings when called with *IS_GET_IMAGE_POS_X and IS_GET_IMAGE_POS_Y as* parameters for *x*, else
IS_SUCCESS, IS_NO_SUCCESS, IS_INVALID_MODE on standby

4.119 is_SetImageSize

Syntax

INT is_SetImageSize (HIDS hf, INT x, INT y)

Description

is_SetImageSize() determines the image size when used with the setting from *is_SetImagePos()*.
To avoid the display area falling outside the image area, the order of function calls must be selected carefully. Starting from the original, the order described below is therefore absolutely mandatory:

1. *is_SetImageSize()*
2. is_SetImagePos()

The function *is_SetAOI()* (see *4.84 is_SetAOI*) combines the two functions. With *is_SetAOI()* the position and the size of an AOI can be set with one function call.

Boundary conditions

See *4.84 is_SetAOI*.

Changing the image size with *is_SetAOI()* or *is_SetImageSize()* also effects the current frame rate and exposure time. Those can be adjusted with *is_SetFrameRate()* and *is_SetExposureTime()*, respectively.

Parameters

hf	Camera handle
x	
1...xMax	Width of the image
IS_GET_IMAGE_SIZE_X	Retrieval of current width
IS_GET_IMAGE_SIZE_X_MIN	Smallest value for the AOI width
IS_GET_IMAGE_SIZE_X_MAX	Largest value for the AOI width
IS_GET_IMAGE_SIZE_X_INC	Increment for the AOI width
IS_GET_IMAGE_SIZE_Y	Retrieval of current height
IS_GET_IMAGE_SIZE_Y_MIN	Smallest value for the AOI height
IS_GET_IMAGE_SIZE_Y_MAX	Largest value for the AOI height
IS_GET_IMAGE_SIZE_Y_MINC	Increment for the AOI height
y	
1…yMax	Height of the image

Return value

When used with *IS_GET_IMAGE_SIZE_X* and *IS_GET_IMAGE_SIZE_Y* the current settings are read, else
IS_SUCCESS or *IS_NO_SUCCESS, IS_INVALID_MODE on standby.*

4.120 is_SetIO

Syntax

INT is_SetIO (HIDS hf, INT nIO)

Description

is_SetIO() sets the two additional digital outputs or reads the currents settings.

Parameters

hf	Camera handle
nIO	Output bit mask
0x00 (00)	Both outputs on 0
0x01 (01)	First output on 1 second on 0
0x02 (10)	First output on 0 second on 1
0x03 (11)	Both outputs on 1
IS_GET_IO	Returns the current bit mask

Return value

IS_SUCCESS, IS_NO_SUCCESS, current settings in connection with *IS_GET_IO.*

4.121 is_SetIOMask

Syntax

INT is_SetIOMask (HIDS hf, INT nMask)

Description

With *is_SetIOMask()* the direction of the in-/outputs can be set.

Parameters

hf	Camera handle
nMask	IO bit mask
0x00 (00)	Use both IOs as input
0x01 (01)	Use first IO as input, second IO as output
0x02 (10)	Use first IO as output, second IO as input
0x03 (11)	Use both IOs as output
IS_GET_IO_MASK	Returns the current bit mask
IS_GET_INPUT_MASK	Returns the IOs applicable as inputs
IS_GET_OUTPUT_MASK	Returns the IOs applicable as outputs

Return value

IS_SUCCESS,
IS_NO_SUCCESS,
current settings in connection with *IS_GET_IO*,
bit masks of the available IOs in connection with *IS_GET_INPUT_MASK* and *IS_GET_OUTPUT_MASK*.

4.122 is_SetKeyColor

Syntax

INT is_SetKeyColor (HIDS hf, INT r, INT g, INT b)

Description

With *is_SetKeyColor()* the keying colour for the DirectDraw overlay surface mode is defined. The function is also used to read out the key colour. The colour value to be read out is transferred via the parameter *r*. Depending on selection, the function supplies as a result either the value of a colour component (0...255) or the RGB value (0 ... 16777215).

Parameters

hf	Camera handle
r	Red channel of keying colour (0...255).
IS_GET_KC_RED	Retrieval of the R channel
IS_GET_KC_GREEN	Retrieval of the G channel
IS_GET_KC_BLUE	Retrieval of the B channel
IS_GET_KC_RGB	Retrieval of the RGB
g	Green channel of keying colour (0...255).
b	Blue channel of keying colour (0...255).

Return value

Colour value in connection with *IS_GET_KC_RGB, IS_GET_KC_RED, IS_GET_KC_GREEN, IS_GET_KC_BLUE, else*
IS_SUCCESS, IS_NO_SUCCESS

4.123 is_SetLED

Syntax

INT is_SetLED (HIDS hf, INT nValue)

Description

With is_SetLED() the LED on the rear side of the camera housing is switched on/off

Parameters

hf	Camera handle
nValue	
IS_SET_LED_OFF	Switches the LED off.
IS_SET_LED_ON	Switches the LED on.
IS_SET_LED_TOGGLE	Toggles the LED between ON and OFF.

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.124 is_SetMemoryMode

Syntax

INT is_SetMemoryMode (HIDS hf, INT nCount, INT nDelay)

Description

Image recording using the optional memory board is activated using the function *is_SetMemoryMode()*. The number of images that are to be recorded in the memory is set as parameter. If using the pre-trigger mode, this parameter determines the size of the circulating memory. The memory is deactivated if the constant *IS_MEMORY_DISABLE* is entered for *nCount*.
The interval in milliseconds at which the images are to be transferred to the memory is set with the parameter *nDelay*. In the sequence mode will be wait for a new trigger between image recordings if the constant *IS_MEMORY_USE_TRIGGER* is handed over to the parameter *nDelay*. This parameter is ignored in pre-trigger mode (see also 3.13 Memory handling).

Parameters

hf	Camera handle
nCount	Determines the number of images that are to be transferred to memory in post-trigger mode. The size of the circulating memory is set with this in pre-trigger mode.
IS_MEMORY_MODE_DISABLE	Deactivates the memory board.
IS_MEMORY_GET_COUNT	Shows the set number of images.
IS_MEMORY_GET_DELAY	Shows the delay between two images in post-trigger mode.
nDelay	Sets the delay in milliseconds between two recorded images in post-trigger mode. The image acquisition is done using an internally timed software trigger.
IS_MEMORY_USE_TRIGGER	Waits at a sequence recording for the next trigger.

Return value

IS_SUCCESS, IS_NO_MEMORY_BOARD_CONNECTED, IS_TOO_LESS_MEMORY, current settings in connection with *IS_MEMORY_GET_COUNT* or *IS_MEMORY_GET_DELAY*

After the memory mode has been activated, camera parameters that influence the image size can no longer be changed. This includes the following functions:

- *is_SetWindowPos*
- *is_SetWindowSize*

4.125 is_SetOptimalCameraTiming

Syntax

INT is_SetOptimalCameraTiming (HIDS hf, INT Mode, INT Timeout, INT *pMaxPxlClk, double *pMaxFrameRate)

Description

With *is_SetOptimalCameraTiming()* the highest possible pixel clock is set where no transmission errors occur during the time set by *Timeout*. Furthermore the maximum frame rate for this pixel clock is returned.
is_SetOptimalCameraTiming() only runs while the camera operates in freerun. Whenever the return value ≠ *IS_SUCCESS*, no settings will be changed.

This function is not supported by cameras of type UI-121X-X, UI-141X-X, and UI-144X-X.

Parameters

hf	Camera handle
Mode	
IS_BEST_PCLK_RUN_ONCE	The function searches for the optimal pixel clock once and returns afterwards.
Timeout [4000...20000]	Sets the time in ms whereas no transmission error may occur. Time may vary between 4 and 20 seconds. The higher the value, the safer the determined pixel clock is. Yet the functions run-time will be elongated accordingly.
pMaxPxlClk	Determined pixel clock in MHz
pMaxFrameRate	Determined framerate in fps

Return value

IS_AUTO_EXPOSURE_RUNNING	Automatic exposure is active
IS_INVALID_PARAMETER	The parameter *Timeout* is not correct
IS_NOT_SUPPORTED	Function is not supported (I.e. invalid *Mode*)
IS_SUCCESS	Success
IS_TRIGGER_ACTIVATED	The camera operates in trigger mode.

The function should be run in an own thread and in the background, so that the load produced by additional colour conversions etc. can be allowed for. Otherwise the function cannot provide the optimal values.

4.126 is_SetPacketFilter (uEye Gigabit Ethernet cameras only)

Syntax

INT is_SetPacketFilter(INT iAdapterID, UINT uFilterSetting)

Description

With *is_SetPacketFilter()* the packet filter for a network adapter is set.

- Only the incoming packets are filtered.
- Ping (ICMP) and ARP are always passed to the operating system, independent of the settings of the packet filter.

Parameters

iAdapterID	System internal adapter ID of a network adapter
uFilterSetting	
IS_ETH_PCKTFLT_PASSALL	All packets are passed to the operating system
IS_ETH_PCKTFLT_BLOCKUEGET	Block IDS packets to the operating system (recommended).
IS_ETH_PCKTFLT_BLOCKALL	Block all packets to the operating system.

Return values

IS_SUCCESS	Success.
IS_INVALID_PARAMETER	*uFilterSetting* invalid.
IS_CANT_OPEN_DEVICE	Driver not found
IS_IO_REQUEST_FAILED	Communication with driver aborted.

Example

```
INT iErr= is_SetPacketFilter(iAdapterId, IS_ETH_PCKTFLT_BLOCKALL);
if(iErr != IS_SUCCESS)
{
    // ...
}
```

4.127 is_SetPixelClock

Syntax

INT is_SetPixelClock (HIDS hf, INT Clock)

Description

is_SetPixelClock() sets the frequency, with that the sensor is read out. If the frequency is too high, it is possible, that images get lost during the transmission. Changing the pixel clock has also influence on frame rate and the exposure time. The current recording of an image is cancelled.

Parameters

hf	Camera handle
Clock	Pixel clock in MHz
IS_GET_PIXEL_CLOCK	Returns current pixel clock
IS_GET_DEFAULT_PIXEL_CLK	Returns standard pixel clock

Return value

Current settings when called with *IS_GET_PIXEL_CLOCK,* else
IS_SUCCESS, IS_NO_SUCCESS, IS_INVALID_MODE on standby

4.128 is_SetPersistentIpCfg (uEye Gigabit Ethernet cameras only)

Syntax

INT is_SetPersistentIpCfg(HIDS hf, UEYE_ETH_IP_CONFIGURATION* pIpCfg,
UINT uStructSize)

Description

The funtion *is_SetPersistentIpCfg()* is used to set the attributes of the persistent IP configuration of a connected camera.
The information contained in the *UEYE_ETH_IP_CONFIGURATION* structure is listed in the following table

UEYE_ETH_ADDR_IPV4	ipAddress	IPv4 address
UEYE_ETH_ADDR_IPV4	ipSubnetmask	IPv4 subnet mask
BYTE	reserved[4]	reserved

Übergabeparameter

hf	*DevID* | *IS_USE_DEVICE_ID*, *DevID* = system internal device ID of the camera
pIpCfg	Pointer to an object *UEYE_ETH_IP_CONFIGURATION*
uStructSize	Size of the *UEYE_ETH_IP_CONFIGURATION* structure in bytes

Rückgabewert

IS_SUCCESS	Success.
IS_INVALID_PARAMETER	*pIpCfg* invalid.
IS_BAD_STRUCTURE_SIZE	The specified structure size is invalid
IS_NOT_SUPPORTED	*hf* was not marked as device ID or it is not a device ID for an ethernet camera.
IS_CANT_OPEN_DEVICE	Driver not found
IS_IO_REQUEST_FAILED	Communication with driver aborted.

Beispiel

```
//Prepare data parameter.
UEYE_ETH_IP_CONFIGURATION ipcfg;
ipcfg.ipAddress.dwAddr=       0xC0A80A02;       // IP address 192.168.10.2
ipcfg.ipSubnetmask.dwAddr= 0xFFFFFF00;       // IP address 255.255.255.0
// prepare handle parameter. mark the given device id with IS_USE_DEVICE_ID.
HIDS h= (HIDS)(iDeviceID | IS_USE_DEVICE_ID);
INT iErr=is_SetPersistentIpCfg( h, &ipcfg, sizeof(UEYE_ETH_IP_CONFIGURATION));
if( iErr != IS_SUCCESS)
{
        // ...
}
```

4.129 is_SetRopEffect

Syntax

INT is_SetRopEffect (HIDS hf, INT effect, INT param, INT reserved)

Description

With function *is_SetRopEffect()* raster operation effects can be set.

Parameters

hf	Camera handle
effect	
IS_SET_ROP_MIRROR_UPDOWN	Reflects a frame around the horizontal axis in real time.
IS_SET_ROP_MIRROR_LEFTRIGHT	Reflects a frame in the camera around the vertical axis. This function is a hardware or a software function, depending on the camera.
IS_GET_ROP_EFFECT	Returns the current settings
param	Toggles the ROP effects 0 = turn off 1 = turn on
reserved	Not used

Return value

IS_SUCCESS, IS_NO_SUCCESS, IS_INVALID_MODE on standby or the current settings in connection with
IS_SET_ROP_EFFECT

4.130 is_SetSaturation

Syntax

INT is_SetSaturation (HIDS hf, INT ChromU, INT ChromV)

Description

Set the software colour saturation. This function will work only with the colour format YUV.

Parameters

hf	Camera handle
ChromU	Saturation-u value multiplied with 100. Range: [*IS_MIN_SATURATION … IS_MAX_SATURATION*]
IS_GET_SATURATION_U	Returns the value for the U-saturation
ChromV	Saturation-v value multiplied with 100. Range: [*IS_MIN_SATURATION … IS_MAX_SATURATION*]
IS_GET_SATURATION_V	Returns the value for the V-saturation

Return value

IS_SUCCESS, IS_NO_SUCCESS
IS_INVALID_PARAMETER (invalid *ChromU* or *ChromV* value)
Current settings in connection with *IS_GET_SATURATION_U* or *IS_GET_SATURATION_V*.

4.131 is_SetStarterFirmware (uEye Gigabit Ethernet cameras only)

Syntax

INT is_SetStarterFirmware(HIDS hf, const CHAR* pcFilepath, UINT uFilepathLen)

Description

Updates the starter firmware of a connected camera.

Parameters

hf	*DevID* | *IS_USE_DEVICE_ID*, *DevID* = system internal device ID of the camera
pcFilepath	Pointer to a null terminated ASCII-String, which contains the complete file path.
uFilepathLen	Length of the file path in byte

Return values

IS_SUCCESS	Success
IS_INVALID_PARAMETER	*pcFilepath* or *uFilepathLen* invalid.
IS_BAD_STRUCTURE_SIZE	The specified structure size is invalid
IS_NOT_SUPPORTED	*hf* was not marked as device ID or it is not a device ID for an ethernet camera.
IS_CANT_OPEN_DEVICE	Driver not found
IS_IO_REQUEST_FAILED	Communication with driver aborted.

Example

```
// Prepare the data parameter.
const CHAR k_sfp[]= “c:\\ids\\firmware.fw”;
// Prepare the handle parameter. mark the given device id with IS_USE_DEVICE_ID.
HIDS h= (HIDS)(iDeviceID | IS_USE_DEVICE_ID);
INT iErr= is_SetStarterFirmware( h, k_sfp, sizeof(k_sfp));
if( iErr != IS_SUCCESS)
{
   // ...
}
```


4.132 is_SetSubSampling

Syntax

INT is_SetSubSampling (HIDS hf, INT mode)

Description

The function *is_SetSubSampling()* is used to activate horizontal and/or vertical subsampling. Depending on the sensor type, subsampling factors 2, 3, 4 or 5 can be set. In the 2x mode, for example, every second pixel is read out from the sensor so the image size is reduced to the half for each subsampling direction and the frame rate may be increased. This is equal to a hardware scaling of the image to 2:1. If a colour camera is used with colour subsampling, always two pixels are read and two left out.
Note: The UI-154x-M, though a monochrome sensor, only features colour subsampling. This can lead to slightly visible artifacts on fine image details..

Monochrom-Sensor

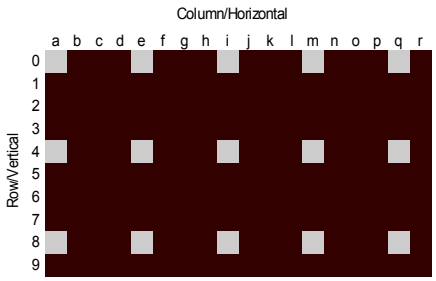

Figure 16: twofold subsampling

colour-Sensor

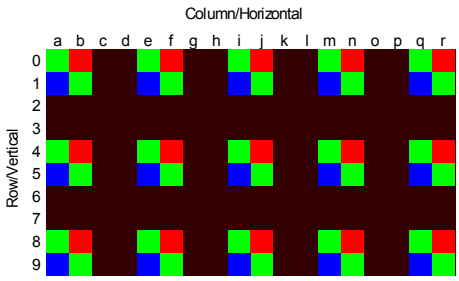

Figure 17: twofold subsampling

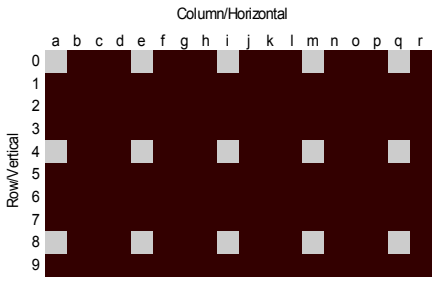

Figure 18: fourfold subsampling

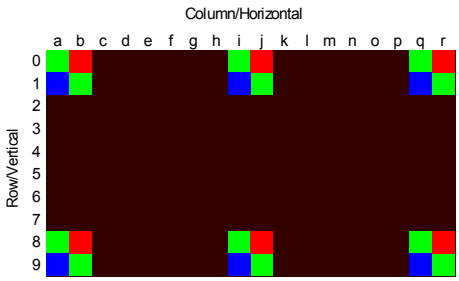

Figure 19: fourfold subsampling

Parameters

hf	Camera handle
mode	
IS_SUBSAMPLING_DISABLE	Disables subsampling.
IS_SUBSAMPLING_2X_VERTICAL	Enables 2x vertical subsampling.
IS_SUBSAMPLING_3X_VERTICAL	Enables 3x vertical subsampling.
IS_SUBSAMPLING_4X_VERTICAL	Enables 4x vertical subsampling.
IS_SUBSAMPLING_5X_VERTICAL	Enables 5x vertical subsampling.
IS_SUBSAMPLING_2X_HORIZONTAL	Enables 2x horizontal subsampling.
IS_SUBSAMPLING_3X_HORIZONTAL	Enables 3x horizontal subsampling.
IS_SUBSAMPLING_4X_HORIZONTAL	Enables 4x horizontal subsampling.
IS_SUBSAMPLING_5X_HORIZONTAL	Enables 5x horizontal subsampling.
IS_GET_SUBSAMPLING	Returns current settings.
IS_GET_SUBSAMPLING_TYPE	Returns weather the camera uses colour conserving binning (IS_SUBSAMPLING_COLOR) or not.
IS_GET_SUPPORTED_SUBSAMPLING	Returns supported subsampling modes.

Return value

IS_SUCCESS, IS_NO_SUCCESS or Current setting when called with *IS_GET_ SUB-SAMPLING*

4.133 is_SetTestImage

Syntax

INT is_SetTestImage (HIDS hf, INT nMode)

Description

The function *is_SetTestImage()* switches on different pre-defined test images in the memory mode. If a certain mode was activated, this test pattern is transferred with the next memory recording in place of the origin image data. Test images vary per recording one line (grey value etc.)

Parameters

hf	Camera handle
nMode	
IS_SET_TEST_IMAGE_DISABLED	Test images are deactivated
IS_SET_TEST_IMAGE_MEMORY_1	1. Test image is activated (a monochrome grey ramp)
IS_SET_TEST_IMAGE_MEMORY_2	2. Test image is activated (a *Bayer* rgb ramp)
IS_SET_TEST_IMAGE_MEMORY_3	3. Test image is activated (a white horizontal line over a black image)
IS_GET_TEST_IMAGE	Current settings are returned

Return value

In connection with *IS_GET_TEST_IMAGE* the current settings are read, else *IS_SUCCESS, IS_NO_SUCCESS*

4.134 is_SetTriggerDelay

Syntax

INT is_SetTriggerDelay (HIDS hf, INT nDelay)

Description

is_SetTriggerDelay() can be used to specify a delay time between occourance of an external trigger signal and start of the expression. The adjusted delay time affects itself additive to the default delay time. This is the delay time, which is always present, until the external trigger signal has been routed to the sensor. The following values apply to the default delay time (with TTL signal level and 50% trigger level):

- CMOS
  - HI_LO: 38,0µs
  - LO_HI: 19,7µs
- CCD
  - HI_LO: 61,5µs
  - LO_HI: 43,2µs

Parameters

hf	Camera handle
nDelay	Delay time (in µs)
IS_GET_TRIGGER_DELAY	Return the current delay time
IS_GET_MIN_TRIGGER_DELAY	Return the min. value
IS_GET_MAX_TRIGGER_DELAY	Return the max. value
IS_GET_TRIGGER_DELAY_GRANULARITY	Returns the resolution of the adjustable delay time

Return value

IS_SUCCESS, IS_NO_SUCCESS or current settings in connection with *IS_GET_TRIGGER_DELAY*

4.135 is_SetWhiteBalance

Syntax

INT is_SetWhiteBalance (HIDS hf, INT nMode)

Description

is_SetWhiteBalance() activates the white balance defined with *nMode*.

The software white balance cannot be activated, if the automatic white balance has been switched on with the function *is_SetAutoParameter*().
is_SetWhiteBalance() accomplishes a colour scaling by software. In order to obtain optimal results the function *is_SetAutoParameter()* is to be used..

Parameters

hf	Camera handle
nMode	
IS_SET_WB_DISABLE	Deactivates white balance
IS_SET_WB_USER	User defined values are used. Factors must be set with *is_SetWhiteBalanceMultipliers()*.
IS_SET_WB_AUTO_ENABLE	Activates automatic white balance for every frame.
IS_SET_WB_AUTO_ENABLE_ONCE	Automatic white balance with the next recorded frame.
IS_SET_WB_DAYLIGHT_65	Industrial standard Daylight 65
IS_SET_WB_COOL_WHITE	Industrial standard CWF (Cool White Fluorescent)
IS_SET_WB_ILLUMINANT_A	Industrial standard Illuminant A
IS_SET_WB_U30	Industrial standard Ultralume 30
IS_SET_WB_HORIZON	Industrial standard Horizon
IS_GET_WB_MODE	Returns the actual mode

Return value

Current settings when called with *IS_GET_WB_MODE*, else
IS_SUCCESS, IS_NO_SUCCESS

4.136 is_SetWhiteBalanceMultipliers

Syntax

INT is_SetWhiteBalanceMultipliers (HIDS hf, double dblRed, double dblGreen, double dblBlue)

Description

is_SetWhiteBalanceMultipliers() sets the user defined factors used for the white balance. Details can also be found at *4.135 is_SetWhiteBalance*.

is_SetWhiteBalanceMultipliers() set the parameters for the software colour scaling which is executed with the function *is_SetWhiteBalance()*. A better controlling of the fundamental colours can be achieved with the function *is_SetAutoParameter()*.

Parameters

hf	Camera handle
dblRed	New factor for red
dblGreen	New factor for green
dblBlue	New factor for blue

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.137 is_ShowDDOverlay

Syntax

INT is_ShowDDOverlay (HIDS hf)

Description

is_ShowDDOverlay() fades on the overlay in DirectDraw back buffer mode. The last data received in the overlay buffer will be displayed. The display is created with only three image buffers. Depending upon the VGA card, the image refresh can be smaller than the overlay display.

Parameters

hf	Camera handle

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.138 is_StealVideo

Syntax

INT is_StealVideo (HIDS hf, int Wait)

Description

The function *is_StealVideo()* initiates the stealing or copy of a single image out of a DirectDraw live stream. The stolen picture is written into the current active image memory in the RAM. Thereby the present colour format is used. With the function *is_PrepareStealVideo()* can be selected if the picture has to be stolen or can be copied. If the copy option is selected the same picture will be shown with DirectDraw and copied into the current active image memory.

See also *6*: *Events in live mode* in *3.9 Event Handling*.

Parameters

hf	Camera handle
Wait	
IS_WAIT	Function waits until the image is recorded
IS_DON'T_WAIT	Function returns immediately

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.139 is_StopLiveVideo

Syntax

INT is_StopLiveVideo (HIDS hf, INT Wait)

Description

The *is_StopLiveVideo()* function freezes the image in the VGA card or in the PC's system memory. The function is controlled with the parameter Wait.
The function has two modes: Using the first mode, the function immediately returns to the calling function and grabs the image in the background. In the second mode the function waits until the image has been completely acquired and only then does the function return.
By the use of *IS_FORCE_VIDEO_STOP* a single frame recording which is started with *is_FreezeVideo(hf, IS_DONT_WAIT)* can be terminated immediately.

Parameters

hf	Camera handle
Wait	
IS_WAIT	Function waits until the entire image is in memory
IS_DONT_WAIT	Function returns straight away.
IS_FORCE_VIDEO_STOP	Terminated digitization immediately

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.140 is_TransferImage

Syntax

INT is_TransferImage (HIDS hf, INT nMemID, INT seqID, INT imageNr, INT reserved)

Description

The function *is_TransferImage()* is used for transferring images from the camera memory to the optional memory board. The ID of the memory where the image is to be transferred is stated as parameter. Also required are the sought sequence ID and the number of the image. The last recorded image is transferred if *0* is specified for the sequence and the image number.

Parameters

hf	Camera handle
nMemID	The memory ID must have been created beforehand with *is_AllocImageMem()* or *is_SetAllocatedImageMem()*.
seqID	The ID of the sequence from where an image has been input.
imageNr	Determines the number of the image within the sequence.

Return value

IS_SUCCESS, IS_NO_SUCCESS, IS_IMAGE_NOT_PRESENT

4.141 is_TransferMemorySequence

Syntax

INT is_TransferMemorySequence (HIDS hf, INT seqID, INT StartNr, INT nCount, INT nSeqPos)

Description

The function *is_TransferMemorySequence()* is used for reading several images from a camera memory into an SDK sequence. Sufficient memory created with *is_AllocateMemory()* must be added to the SDK sequence first with *is_AddToSequence()*. The ID of the camera memory sequence is required as parameter. The number where image read-in should start can also be set. *nCount* is used for stipulating how many images are to be transferred as of *StartNr*.
The last parameter is used for transferring the images as of *nSeqPos* in the sequence.

Example

Transfer of 3 images from the camera sequence 1 as of image number 2 in the SDK sequence as of position 3.

Call: `is_TransferMemorySequence(hf, 1, 2, 3, 3);`

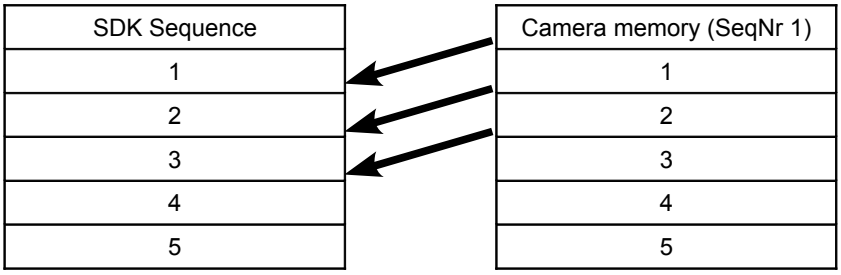

Figure 20: is_TransferMemorySequence

Parameters

hf	Camera handle
seqID	The ID of the sequence in the camera memory where images are to be read out.
StartNr	Image index as of which images are to be read out of the sequence.
nCount	Number of images that are to be transferred from the memory into the sequence. (0 = All images of the sequence as of *StartNo* are transferred).
nSeqPos	Start position of the sequence.

Return value

IS_SUCCESS, IS_NO_SUCCESS, IS_IMAGE_NOT_PRESENT

4.142 is_UnlockDDMem

Syntax

INT is_UnlockDDMem (HIDS hf)

Description

is_UnlockDDMem() disables access to the image memory in DirectDraw mode. Then the contents of the back buffer are updated on the display.

Parameters

hf	Camera handle

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.143 is_UnlockDDOverlayMem

Syntax

INT is_UnlockDDOverlayMem (HIDS hf)

Description

is_UnlockDDOverlayMem() disables access to the overlay buffer in DirectDraw back buffer mode. The contents of the overlay buffer are updated on the display (if *is_ShowDDOverlay()* has faded on the overlay)

Parameters

hf	Camera handle

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.144 is_UnlockSeqBuf

Syntax

INT is_UnlockSeqBuf (HIDS hf, INT nNum, char* pcMem)

Description

With is_UnlockSeqBuf() image acquisition is allowed in a previously locked image memory. The image memory is put to the previous position in the sequence list.

Parameters

hf	Camera handle
nNum	Number of the image memory which is to be released (1... max.) or IS_IGNORE_PARAMETER:The image buffer is only identified by the start address.
pcMem	Start address of image memory

nNum indicates the position in the sequence list and not the memory ID assigned with *is_AllocImageMem()*.

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.145 is_UpdateDisplay

Syntax

INT is_UpdateDisplay (HIDS hf)

Description

is_UpdateDisplay() updates the display when the display mode is set with *IS_SET_DM_DIRECTDRAW*. Also see function *4.99 is_SetDisplayMode*).

Parameters

hf	Camera handle

Return value

IS_SUCCESS, IS_NO_SUCCESS

4.146 is_WriteEEPROM

Syntax

INT is_WriteEEPROM (HIDS hf, INT Adr, char* pcString, INT Count)

Description

In the uEye camera there is a rewritable EEPROM, where 64 bytes of information can be written. With the *is_ReadEEPROM()* function the contents of this 64 byte block can be read.

Parameters

hf	Camera handle
Adr	Start address to where data will be written. Value range (0...63)
pcString	String which contains certain data
Count	Number of readable characters

Return value

IS_SUCCESS, IS_NO_SUCCESS, IS_INVALID_MODE on standby

4.147 is_WriteI2C (uEye LE cameras only)

Syntax

INT is_WriteI2C (HIDS hf, INT nDeviceAddr, INT nRegisterAddr, BYTE* pbData, INT nLen)

Description

With *is_WriteI2C()* data can be written over the I2C-bus. I2C bus clock is 400 kHz.

Parameters

hf	Camera handle
nDeviceAdr	Slave address
nRegisterAddr	Register address (only 8 Bit addresses valid).
data	Data to write
length	Length of data

Invalid addresses for *nRegisterAddr*:
0x48, 0x4C, 0x51, 0x52, 0x55, 0x5C, 0x5D, 0x69 (these are used internal).

Return value

IS_SUCCESS, IS_NO_SUCCESS

5 Error Messages

Nr	Error	Description
-1	IS_NO_SUCCESS	General error message
0	IS_SUCCESS	General success message – no error
1	IS_INVALID_CAMERA_HANDLE	Camera handle is invalid. Most of the functions of the uEye SDk expect the camera handle as first parameter.
2	IS_IO_REQUEST_FAILED	An IO requirement of the uEye driver failed. Possibly Api DLL and driver file do not fit.
3	IS_CANT_OPEN_DEVICE	An attempt to open the camera failed (camera missing or error when initialising).
11	IS_CANT_OPEN_REGISTRY	Error while opening a Windows Registry key
12	IS_CANT_READ_REGISTRY	Error while reading settings from the windows registry
15	IS_NO_IMAGE_MEM_ALLOCATED	The driver could not allocate memory.
16	IS_CANT_CLEANUP_MEMORY	The driver could not release the used memory.
17	IS_CANT_COMMUNICATE_WITH_DRIVER	Communication with the driver failed, because no driver is loaded.
18	IS_FUNCTION_NOT_SUPPORTED_YET	The function is not supported yet.
49	IS_INVALID_MEMORY_POINTER	Invalid pointer or invalid memory ID
50	IS_FILE_WRITE_OPEN_ERROR	File cannot be opened for writing.
51	IS_FILE_READ_OPEN_ERROR	File cannot be opened for reading.
52	IS_FILE_READ_INVALID_BMP_ID	The indicated file is not a valid bit-map
53	IS_FILE_READ_INVALID_BMP_SIZE	The size of the bit-map is wrong (too large).
108	IS_NO_ACTIVE_IMG_MEM	No activated bit map memory available. The memory must be activated with the function *is_SetActiveImageMem*, or a sequence must be created with the function *is_AddTo_Sequence*.
112	IS_SEQUENCE_LIST_EMPTY	The sequence list is empty and can not be deleted.
113	IS_CANT_ADD_TO_SEQUENCE	The bit map memory is already in the sequence and cannot be added twice.
117	IS_SEQUENCE_BUF_ALREADY_LOCKED	The memory could not be locked. The pointer on the buffer is invalid.
118	IS_INVALID_DEVICE_ID	The device ID is invalid. Valid IDs are between 0 and 255.
119	IS_INVALID_BOARD_ID	The board ID is invalid. Valid IDs are between 1 and 255.
120	IS_ALL_DEVICES_BUSY	All cameras are in use
122	IS_TIMED_OUT	A *timeout* arose. A picture recording could not be finished in the prescribed time.
125	IS_INVALID_PARAMETER	One of the handed over parameters is outside of the valid range, or is not supported for this sensor and/or is not accessible in this mode.
127	IS_OUT_OF_MEMORY	No memory could be allocated.
139	IS_NO_USB20	The camera is connected at a port, which does not support the high speed standard USB 2,0. Cameras without memory wont be operate at USB 1.1 ports.

Nr	Error	Description
140	IS_CAPTURE_RUNNING	An acquisition is running. This must be terminated, before a new one can be begun.
141	IS_MEMORY_BOARD_ACTIVATED	The memory mode is active. Therefore no changes of the dimension of the picture can be made.
143	IS_NO_MEMORY_BOARD_CONNECTED	No memory board was detected. The function requires a camera with memory board.
144	IS_TOO_LESS_MEMORY	The adjusted number of pictures exceeds in the current size the capacity of the memory board (4MB)
145	IS_IMAGE_NOT_PRESENT	The requested picture is missing or no longer valid in the camera memory.
147	IS_MEMORYBOARD_DISABLED	The memory board was deactivated due to an error. The function is not available any longer.
148	IS_TRIGGER_ACTIVATED	The function is not possible, because the camera waits for a trigger signal.
151	IS_CRC_ERROR	Reading the settings a CRC error arose.
152	IS_NOT_YET_RELEASED	This function is not yet activated in this driver version.
153	IS_NOT_CALIBRATED	The camera does not contain calibration data.
154	IS_WAITING_FOR_KERNEL	Still waiting for a feedback of the kernel driver.
155	IS_NOT_SUPPORTED	The used camera model does not support this function or attitude.
157	IS_OPERATION_ABORTED	No file could be stored, because the dialogue was terminated without selection.
158	IS_BAD_STRUCTURE_SIZE	An internal structure has the wrong size.
159	IS_INVALID_BUFFER_SIZE	The bit image memory has the wrong size to take the picture in the desired format.
160	IS_INVALID_PIXEL_CLOCK	This attitude is not possible with the current adjusted pixel clock.
161	IS_INVALID_EXPOSURE_TIME	This attitude is not possible with the current exposure time.
162	IS_AUTO_EXPOSURE_RUNNING	The attitude cannot be changed, as long as the automatic exposure control is activated.
163	IS_CANNOT_CREATE_BB_SURF	Error creating backbuffer surface.
164	IS_CANNOT_CREATE_BB_MIX	BackBuffer mixer surfaces can not be created
165	IS_BB_OVLMEM_NULL	BackBuffer overlay mem could not be locked
166	IS_CANNOT_CREATE_BB_OVL	BackBuffer overlay mem could not be created .
167	IS_NOT_SUPP_IN_OVL_SURF_MODE	Function not supported in overlay surface mode.
168	IS_INVALID_SURFACE	Surface invalid.
169	IS_SURFACE_LOST	Surface has been lost.
170	IS_RELEASE_BB_OVL_DC	Error releasing backbuffer overlay DC.
171	IS_BB_TIMER_NOT_CREATED	BackBuffer timer could not be created.
172	IS_BB_OVL_NOT_EN	BackBuffer overlay has not been enabled.
173	IS_ONLY_IN_BB_MODE	Only possible in backbuffer mode.
174	IS_INVALID_COLOR_FORMAT	Invalid colour format.
175	IS_INVALID_WB_BINNING_MODE	While Monobinning/-subsampling is activated, an AutoWhitebalance is not possible.
176	IS_INVALID_I2C_DEVICE_ADDRESS	Invalid I2C device address.

Nr	Error	Description
177	IS_COULD_NOT_CONVERT	The active image could not be processed.
178	IS_TRANSFER_ERROR	Transmission error.
179	IS_PARAMETER_SET_NOT_PRESENT	The parameter set is not present
180	IS_INVALID_CAMERA_TYPE	The camera type given in the .ini file does not match the active camera modell.

Table 18: Error messages

6 Index of Illustrations

7 Index of Tables

