Instruction manual

Thorlabs Instrumentation

DCU USB 2.0 Cameras

Development Kit (SDK)

2007

THORLABS

Version: 2.4
Date: 23.07.2007

Contents page

Preface

We have taken every possible care in preparing this manual. Nevertheless, we are unable to provide any guarantee with regard to content, entirety or quality of the details contained in this manual. The contents of this manual are revised regularly and brought up to latest standards. Furthermore, we are also unable to guarantee that the product will operate fault-free even if the specifications and recommended computer configuration are observed.

Under no circumstances whatsoever are we able to guarantee that a specific application objective can be achieved with the purchase of this product.

Liability for immediate damages, subsequent damages and damages to others resulting from the purchase of this product is excluded within the terms of existing legislation. Liability under any circumstances is restricted to the product price.

Thorlabs GmbH grants the purchaser the right to use the software herewith. Copying the software in any form whatsoever, with the exception of a backup copy, is strictly forbidden.

Trademarks

IBM is a registered trademark of International Business Machines Corporation. MICROSOFT and WINDOWS are trademarks or registered trademarks of the Microsoft Corporation. All other products or company names which are mentioned in this manual are used solely for the purpose of identification and/or description and can be the trademark or registered trademark of the respective owners.

1 Introduction

Thank you for your decision to purchase a camera of the *Thorlabs GmbH*. This manual describes the operation mode of the DCU camera family.

The DCU cameras are made for industrial, medical and multimedia applications in monochrome and colour. The cameras support USB standard 2.0 or higher. The images are available in 8 bit monochrome resp. 24 bit true colour quality.
To integrate a DCU camera into own programs under Windows 2000, Windows XP and Linux a SDK (software development kit) is contained in the scope of delivery.

This manual describes the functions of the DCU Software Development Kit (SDK).

Please, read the file *WhatsNew.txt*, which is on the installation CD. Here you can find additional information, which are possibly not contained in the printed edition of the manual.

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

2 Programming

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

NOTE

Please note that Microsoft modified the format of the lib File starting from version 6.0. The DCU driver has been created with Visual C++ 7.1. Thus the file uc480.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 DCU functions directly from Visual BASIC, if you replace the functions is_<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 DCU 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. „Thorlabs 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 = DCU USB2.0
Char	Reserved[8]	reserved

Table 1: CAMINFO data structure of the EEPROMS

2.4 Colour formats

Each of the colour formats supported by the DCU 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				
UYVY	Y1	V0	Y0	U0
Y8	Y3	Y2	Y1	Y0

Table 2: Colour formats

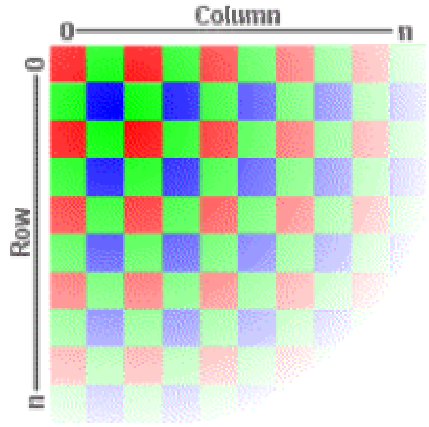


Fig. 1: Principle structure of the Bayer-Pattern

HINT

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 Mode (BMP/DIB)
Once the *Camera viewer (uc480Viewer)* sample application has been started the bitmap mode is activated. The image from the DCU 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 dialog boxes.

Bitmap Mode

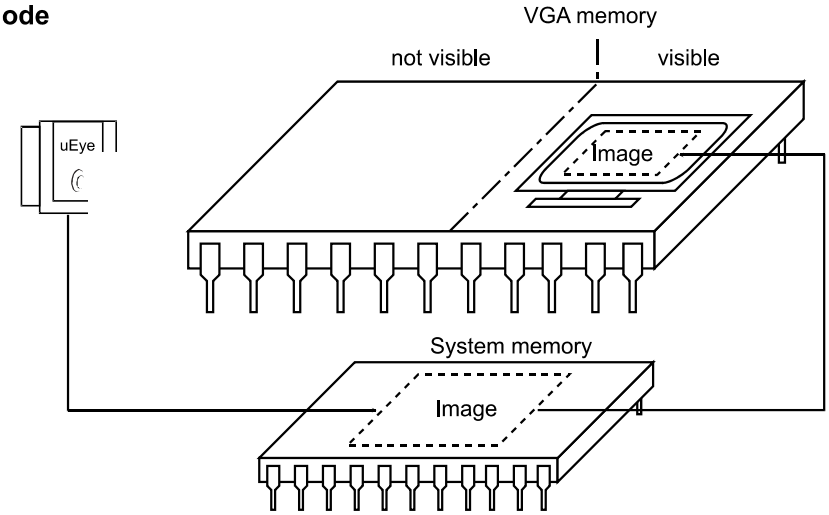


Fig. 2: Bitmap Mode

DirectDraw BackBuffer mode (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 *4.28 is_GetDC* and *4.65 is_ReleaseDC*). The overlay is not always faded on. It has to be made visible with *is_ShowDDOverlay()* (see *4.119 is_ShowDDOverlay*). As its key color, the overlay uses black, thus an overlay cannot contain any black color.
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 BackBuffer mode.
The BackBuffer mode is set as follows:
Mode = *IS_SET_DM_DIRECTDRAW* | *IS_SET_DM_BACKBUFFER*

DirectDraw Backbuffer Mode with Overlay

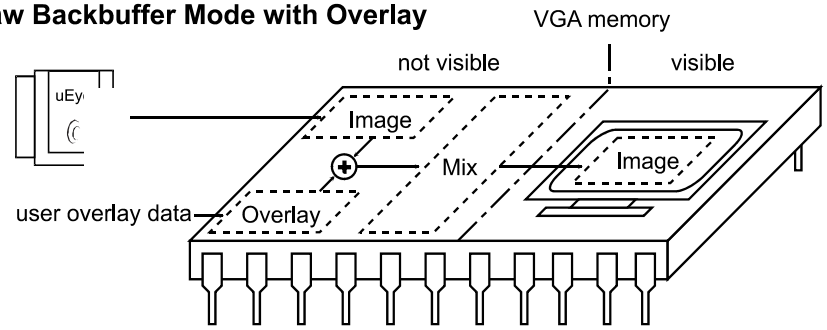


Fig. 3: DirectDraw Back-Buffer Mode

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*

DirectDraw Overlay Surface Mode

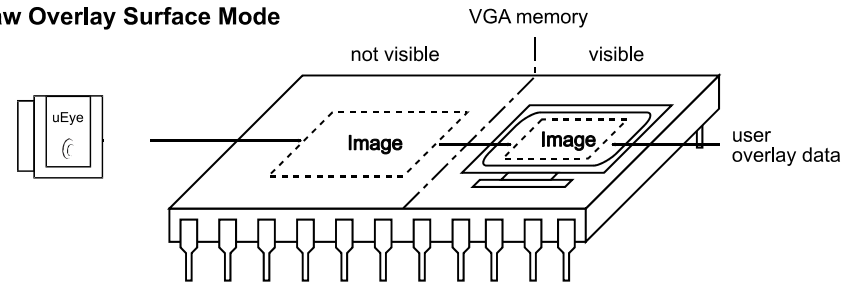


Fig. 4: DirectDraw Overlay-Surface-Mode

HINT

The Back Buffer mode of the FALCON frame grabber is activated by setting the parameter IS_SET_DM_DIRECTDRAW. The BackBuffer mode of the DCU

cameras is activated as follows: IS_SET_DM_DIRECTDRAW | IS_SET_DM_BACKBUFFER.

3 Function lists

3.1 Initialization und 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 und 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	Gets event counter and counter value
is_GetAutoInfo	Returns status informationen of the auto functionality
is_GetCameraList	Gets information about the connected cameras
is_GetCameraType	Gets type of camera (e.g. DCU USB)
is_GetColorDepth	Gets current colour mode from VGA card
is_GetDLLVersion	Returns the version of uc480.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	Aktivates/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	Acitvates/deactivates the additional hardware gain
is_SetGamma	Set the gamma value (digital reworking)
Fehler! Verweisquelle konnte nicht gefunden werden.	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
is_GetRevisionInfo	Revision information of the individual DCU 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

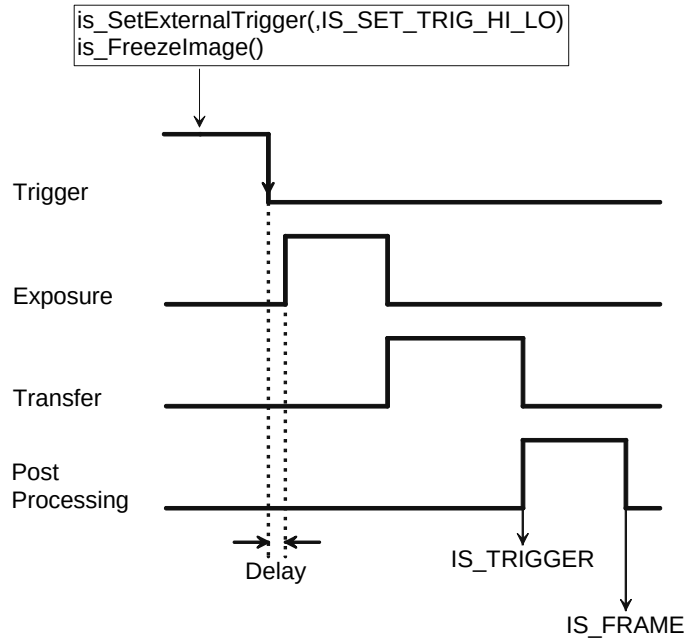


Fig. 5: Events with single trigger recording

Events in live mode (sequence of 3 images)

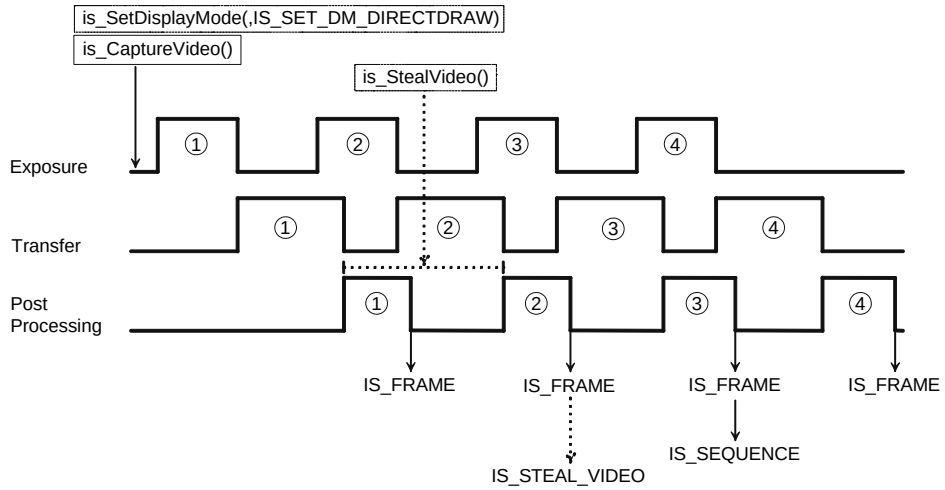


Fig. 6: Events in live mode

Events in memory mode

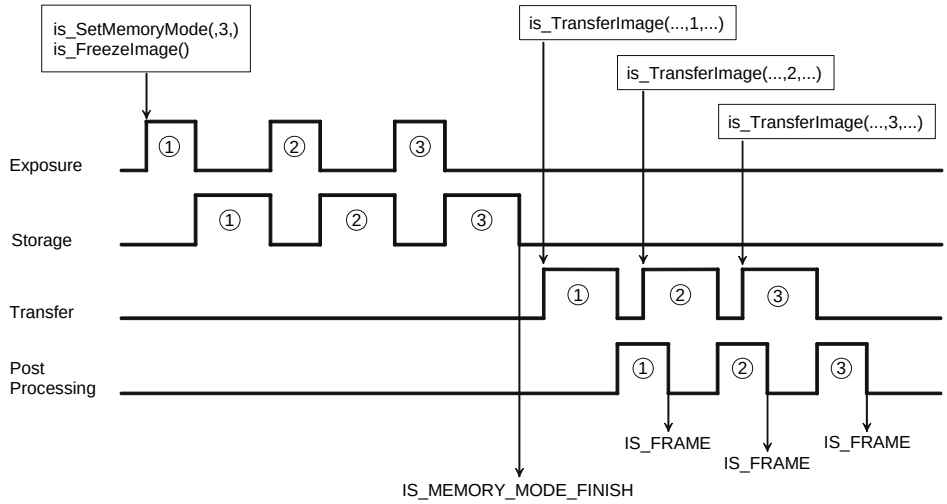


Fig. 7: Events in memory mode

3.10 Control of inputs / 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 (only UI-1543-M)	Set the additional digital outputs
is_SetTriggerDelay	Specify a delay time of a trigger signal

Table 12: Function list Control of sync generator and input / outputs

3.11 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.

Funktion	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 neccessary*

Table 13: Validity of functions

HINT

DirectDraw is not supported under LINUX

4 Description of the functions

To aid the integration of the DCU 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 function>

Syntax:

Function prototype from the header file uc480.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 *Camera Viewer.EXE*, which uses the DCU 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 when *line* without rest is divisible by 4
adjust = 4 - rest(line / 4) when *line* without rest is not divisible by 4

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!

HINT

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	Contains pointer to start of memory location
pid	Contains the ID for this memory location

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	Timeout 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):
ulValue	
IS_GET_STATUS	Read the parameter value for nInfo.

Return value:

IS_SUCCESS, IS_NO_SUCCESS or current value of ulValue = *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*
  It exists no memory for the image.
- *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 memeories added to the sequence.

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 or 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.

HINT

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.

HINT

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 *4.53 is_InitEvent*.

Parameters:

hf	Camera handle
which	ID of the event
See *4.53 is_InitEvent*	

Return value:

IS_SUCCESS, IS_NO_SUCCESS

Example:

See *4.53 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:

Actual 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 * color 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 *4.119 is_ShowDDOverlay*). As its key color, the overlay uses black, an thus an overlay cannot contain any black color.

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 *4.53 is_InitEvent*.

Parameters:

hf	Camera handle
which	ID of the event to release
See *4.53 is_InitEvent*	A new image is available.

Return value:

IS_SUCCESS, IS_NO_SUCCESS

Example:

See See *4.53 is_InitEvent*.

4.14 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_DCU_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_SET_STEAL_VIDEO	An image detracted from the overlay is available.
IS_TRANSFER_FAILED	An Error occurred during data transfer (Frame rate, Pixelclock too high)
IS_TRIGGER	An image, ist 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 or IS_NO_SUCCESS

4.15 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 DCU camera. The image memory which has been allocated by the user and which has not been released yet, is released with *is_ExitCamera*.

HINT

The DCU 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

4.16 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 event that should be deleted

See *4.53 is_InitEvent*

Return value:

IS_SUCCESS, IS_NO_SUCCESS

Example:

See *4.53 is_InitEvent*.

4.17 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 *4.19 is_FreezeVideo* and *4.94 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)
  {
    // no trigger signal, force to take picture
    is_ForceTrigger(hf);
  }

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

4.18 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!

Parameters:

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

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.19 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). When working with ring buffering, image acquisition only takes place in the first image memory of the sequence.
If there is a change in camera settings (exposure, Pclk, AOI, ...) after selecting the function with the parameter IS_DONT_WAIT, the recording is aborted and the new camera settings are installed.
The picture recording takes place triggered, if the trigger mode were activated before with *is_SetExternalTrigger()*.

HINT

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)

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 timout.
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);
```

4.20 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 *is_GetImgMem()*.

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

4.21 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 to 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

4.22 is_GetAutoInfo

Syntax:

INT is_GetAutoInfo (HIDS hf, PDCU_AUTO_INFO info)

Description:

With the function *is_GetAutoInfo()* status information of autofunctionality can be readout. The information is available in the structure *DCU_AUTO_INFO*.

HINT

The status information in the structure DCU_AUTO_INFO is only valid if at least one autofunctionality is activated.

DCU_AUTO_INFO

INT	AutoAbility	0x01: AutoShutter possible (AC_SHUTTER) 0x02: AutoGain possible (AC_GAIN) 0x03: AutoGain und AutoShutter possible 0x04: AutoWhiteBalance possible (AC_WHITEBAL) 0x07: All auto functions are possible
AUTO_BRIGHT_STATUS	sBrightCtrlStatus	See AUTO_BRIGHT_STATUS
AUTO_WB_STATUS	sWBCtrlStatus	See AUTO_WB_STATUS
DWORD	reserved	Reserved for extensions

In the structure DCU_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_CHANNEL
AUTO_WB_CHANNEL_STATUS	RedChannel	See AUTO_WB_CHANNEL_STATUS
AUTO_WB_CHANNEL_STATUS	GreenChannel	See AUTO_WB_CHANNEL_STATUS
AUTO_WB_CHANNEL_STATUS	BlueChannel	See AUTO_WB_CHANNEL_STATUS

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 DCU_AUTO_INFO (see above)

Return value:

IS_SUCCESS, *IS_NO_SUCCESS*

Example:

Readout *AutoInfo*:

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

4.23 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

4.24 is_GetCameraInfo

Syntax:

INT is_GetCameraInfo (HIDS hf, PCAMINFO pInfo)

Description:

The function *is_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 CAMINFO

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.25 is_GetCameraList

Syntax:

INT is_GetCameraList (PDCU_CAMERA_LIST pucl)

Description:

With the function *is_GetCameraList()* information about the attached cameras can be queried. In the following tables the used structures are described.

DCU_CAMERA_LIST

ULONG	dwCount	Number of cameras attached at the system.
DCU_CAMERA_INFO	uci[1]	Placeholder for 1 .. n structures DCU_CAMERA_INFO

DCU_CAMERA_INFO

DWORD	dwCameraID	User defined camera ID
DWORD	dwDeviceID	System internal devices ID
DWORD	dwSensorID	Sensor ID
DWORD	dwInUse	1 = camer in use 0 = camera not in use
Char	SerNo[16]	serial number of the camera
Char	Model[16]	camera model
DWORD	dwReserved[16]	Reserved

Parameters:

pucl	Structure DCU_CAMERA_LIST

Return value:

IS_SUCCESS, *IS_ACCESS_VIOLATION* (not enough memory allocated) or
IS_CANT_OPEN_DEVICE bzw. IS_IO_REQUEST_FAILED (Communication with driver failed)

Example:

```
PDCU_CAMERA_LIST pucl = new DCU_CAMERA_LIST;
pucl->dwCount = 0;
if (is_GetCameraList (pucl) == IS_SUCCESS){
   DWORD dwCameraCount = pucl->dwCount;
   delete pucl;      //reallocate the required list size
   pucl = (PDCU_CAMERA_LIST) new char [sizeof (DWORD) + dwCameraCount *
   sizeof (DCU_CAMERA_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);
      }
   }
}
```

4.26 is_GetCameraType

Syntax:

INT is_GetCameraType (HIDS hf)

Description:

is_GetCameraType() returns the result of which type of camera family is installed. With the DCU camera family is always returned *IS_CAMERA_TYPE_DCU_USB*.

Parameters:

hf	Camera handle

Return value:

IS_CAMERA_TYPE_DCU_USB

4.27 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

4.28 is_GetDC

Syntax:

INT is_GetDC (HIDS hf, HDC* phDC)

Description:

In DirectDraw BackBuffer mode *is_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

4.29 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

4.30 is_GetDLLVersion

Syntax:

INT is_GetDLLVersion()

Description:

Returns the version number of the DCU_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

4.31 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. 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

4.32 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

4.33 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 or *IS_NO_SUCCESS*

4.34 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*.

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

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

$$fps(n) = \frac{1}{\min + m}$$ m = n*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

4.35 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.

4.36 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 folowing color 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]
IS_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 color 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);
```

4.37 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).

HINT

In DirectDraw mode the address pointer changes when the output window is moved (see also 4.60 is_LockDDMem). When ring buffering is being used (4.1 is_AddToSequence), is_GetImageMem() returns the previously active image memory.

Parameters:

hf	Camera handle
pMem	Pointer to beginning of image memory

Return value:

IS_SUCCESS, *IS_NO_SUCCESS*

4.38 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 - when *line* without remainder is divisible by 4
adjust = 4 - rest(line / 4) when *line* with remainder is divisible by 4

Parameters:

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

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.39 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

4.40 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

4.41 is_GetNumberOfCameras

Syntax:

INT is_GetNumberOfCameras (INT* pnNumCams)

Description:

Function *is_GetNumberOfCameras()* delivers the number of DCU cameras attached to the PC.

Parameters:

pnNumCams	Number of cameras found

Return value:

IS_SUCCESS

4.42 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

4.43 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:

no parameter

Return value:

IS_OS_WIN2000, IS_OS_WINXP*, IS_OS_WINSERVER2003*

4.44 is_GetPixelClockRange

Syntax:

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

Description:

is_GetPixelClockRange() returns the adjustable range for the pixel clock.

Parameters:

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

Return value:

IS_SUCCESS or IS_NO_SUCCESS

4.45 is_GetRevisionInfo

Syntax:

INT is_GetRevisionInfo(HCAM hf, PREVISIONINFO prevInfo)

Description:

The function *is_GetRevisionInfo()* readout the revision information of the individual DCU components. The revision structure is built-on as follows:

Size	Size of the structure
Sensor	Hardware sensor revision
Cypress	
Highbyte:	1 = Type FX2 2 = Type FX2LP
Lowbyte:	Cypress revision
Blackfin	DSP revision
DSPFirmware	Blackfin firmware version
USB_Board	USB Board revision
Sensor_Board	Sensor Board revision
Processing_Board	Processing Board revision
Housing	Housing model
Filter	Embedded filtertype
Timing_Board	Timing Board revision
Product	Product revision
Reserved	reserved

Parameters:

hf	Camera handle
prevInfo	Pointer to Revision Information Structure

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.46 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**
		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]	Sensor name e.g. " DCU224C "		
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

4.47 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

4.48 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

4.49 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

4.50 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

4.51 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

4.52 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 DCU 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 *4.24 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.

HINT

The DCU 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)

4.53 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_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_DONT_WAIT);
   if (WaitForSingleObject(hEvent, 1000) == WAIT_OBJECT_0) {
      // Picture successful taken up
   }
   is_DisableEvent(hf, IS_SET_EVENT_FRAME);
   is_ExitEvent(hf, IS_SET_EVENT_FRAME);

}
```

4.54 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 *is_AllocImageMem()*
NID	ID of image memory from von *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 the line increment of the image memory.

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.55 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

4.56 is_LoadBadPixelCorrectionTable

Syntax:

INT is_LoadBadPixelCorrectionTable (HIDS hf, char *File)

Description:

is_LoadBadPixelCorrectionTable() loads a table saved previously with the function *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

4.57 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 *4.37 is_GetImageMem* and *4.105 is_SetImageMem*).

Parameters:

hf	Camera handle
File	Pointer on filename. 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.

4.58 is_LoadImageMem

Syntax:

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

Description:

is_LoadImage() 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 color format and depth. With the function *is_FreeImageMem ()* (see *4.18 is_FreeImageMem*) the image memory is released.

Parameters:

Hf	Camera handle
File	Name of the image file, NULL -> ”Open” dialog 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 (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)

4.59 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.

HINT

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 filename. Both the absolute and the relative path can be passed.

Return value:

IS_SUCCESS, IS_NO_SUCCESS

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

4.60 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

4.61 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

4.62 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.
Only one image memory can be disabled at the same time. To access the image memory use function *is_UnlockSeqBuf()*.

Parameters:

hf	Camera handle
nNum	Number of the disabled image memory (1...max).
pcMem	Start address of the image memory, which should be protected.

HINT

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

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.63 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 modus
IS_SET_STEAL_COPY	Copy modus
StealColorMode	Not used

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.64 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 *4.126 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);
```

4.65 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 from *is_GetDC()*

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.66 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*.

HINT

The color 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 OR combined with the modes above:	
IS_RENDER_MIRROR_UPDOWN	horizontal mirroring of the image

Return value:

IS_SUCCESS or 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_UPDOWN);
```

4.67 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

4.68 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 *file* 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

4.69 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 *4.37 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 *is_CameraStatus()* for maintaining the selected directory path). Both the absolute and the relative path can be passed.

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.70 is_SaveImageEX

Syntax:

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

Description:

Saves an image from the computer's memory as bitmap or JPEG to a file (see also *4.69 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* - dialog is displayed (see *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

4.71 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” dialog box is opened (see *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

4.72 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 color 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 color depth.

Parameters:

Hf	Camera handle
File	Name of the image file, NULL -> ”Save as” dialog 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 file format
IS_IMG_JPG	JPeg file format
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)

4.73 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* dialog is displayed.

Parameters:

hf	Camera handle
pFilename	Pointer on filename. Both the absolute and the relative path can be passed.

Return value:

IS_SUCCESS, IS_NO_SUCCESS

Example ini-File:

See *4.59 is_LoadParameters*

4.74 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 DCU 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)*;

HINT

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

Now the image memory can be used. Sequences are also possible.
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

4.75 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.

HINT

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.

With the functions *is_SetImagePos()* (see *4.106 is_SetImagePos*)and *is_SetImageSize()* (see *4.107 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

Examples:

Set Auto_Brightness_AOI:

```
Int nRet = SetAOI (m_hCam,          // Camera handle
   IS_SET_AUTO_BRIGHT_AOI,          // AOI Typ + command
   &xpos,
   &ypos,
```

```
            &width,
            &height);
```

4.76 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 color rendering to the given value and to hold the picture framerate 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 framerate can be changed further manually or automatically with activated shutter control, in order to maintain the range of the exposure control dynamically. The automatic framerate control has the purpose of adjusting the framerate to an optimal value. Thus the necessary control range is available for shutter control in all situations with a framerate 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 color 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 color 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 *4.53 is_InitEvent*). Alternatively the control keeps active and reacts to deviations from the target value.

HINT

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.
pval1	Parameter (value)
pval2	Parameter (value)

The parameters *pval1* und *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 (inaktiv) 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-pinter 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 uc480.h)

IS_DEFAULT_AUTO_BRIGHT_REFERENCE	Default setpoint for AutoGain and AutoShutter
IS_MIN_AUTO_BRIGHT_REFERENCE	Minimum setpoint for AutoGain and AutoShutter
IS_MAX_AUTO_BRIGHT_REFERENCE	Maximum setpoint for AutoGain and AutoShutter
IS_DEFAULT_AUTO_SPEED	Default value for AutoSpeed
IS_MIN_AUTO_SPEED	Minimum setpoint for AutoSpeed
IS_MIN_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 *4.100 is_SetHardwareGain*).

AutoExposure funtionality 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 *4.93 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);
```

4.77 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.

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*

4.78 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.

Examples for reading out the table:

```
WORD *pList = NULL;
// number of coordinates in the table
DWORD nCount = is_SetBadPixelCorrectionTable(hf, IS_GET_LIST_SIZE,
NULL);
// allocate memory for table
pList = new WORD[ 1 + 2*nCount ];
is_SetBadPixelCorrectionTable(hf, IS_GET_BADPIXEL_LIST, pList);

// clear table
delete [] pList;
```

Return value:

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

4.79 is_SetBayerConversion

Syntax:

INT is_SetBayerConversion (HIDS hf, INT nMode)

Description:

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

Boundary conditions

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 values:

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

4.80 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 framerate 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_4X_VERTICAL	Activates 4x vertical binning
IS_BINNING_2X_HORIZONTAL	Activates 2x 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*

4.81 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	Turns the compensation off
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_OFFET	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 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 preset offset with *IS_GET_BL_OFFSET*

4.82 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*).

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*.

4.83 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 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* or *IS_NO_SUCCESS*.

4.84 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.

HINT

After changing these parameters, a white balance adjustment must be done (see 4.76 is_SetAutoParameter)

Parameters:

hf	Camera handle
IS_CCOR_DISABLE	Deactivates colour correction
IS_CCOR_ENABLE	Activates 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*

4.85 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*.

4.86 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%.

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.

4.87 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:

- color correction
- bayer conversion mode
- color mode
- gamma
- white balance multipliers.

Parameters:

Hf	Camera handle
ColorCorrection	activates/desactivates the color 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 color 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*

4.88 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

4.89 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 colormode 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

HINT

The DCU 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 mode (back buffer mode)
IS_SET_DM_DIRECTDRAW | IS_SET_DM_ALLOW_OVERLAY	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*

Example:

is_SetDisplayMode (hf, Mode);
Bitmap-Modus (digitized in system memory):

```
Mode = IS_SET_DM_DIB
```

DirectDraw back buffer mode (Overlay possible. See *is_EnableDDOverlay*)

```
Mode = IS_SET_DM_DIRECTDRAW
```

Allows memory storage in system memory:

```
Mode =  IS_SET_DM_DIRECTDRAW  |  IS_SET_DM_ALLOW_SYSMEM
```

```
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
```

4.90 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

4.91 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 or *IS_NO_SUCCESS*

4.92 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 dialog box. When you quit the dialog box with *Cancel*, the error reporting is automatically de-activated. 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*.

4.93 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.
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 DCU timing.
Depending on the time of the change of the exposure time this affects only with the recording of the next image.

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/framerate is used.
IS_GET_DEFAULT_EXPOSURE	Returns the default exposure time.
newEXP	Actual exposure time.

HINT

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 *4.76 is_SetAutoParameter*).

Return value:

IS_SUCCESS or *IS_NO_SUCCESS*

4.94 is_SetExternalTrigger

Syntax:

INT is_SetExternalTrigger (HIDS hf, INT nTriggerMode)

Description:

is_SetExternalTrigger() activates the trigger input. This 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 (TTL) IS_SET_TRIG_HI_LO
- Action on low high edge (TTL) IS_SET_TRIG_LO_HI
- Deactivates trigger IS_SET_TRIG_OFF

If wihtout 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.

HINT

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

Parameters:

hf	Camera handle
nTriggerMode	
IS_SET_TRIG_OFF	Disable trigger processing.
IS_SET_TRIG_HI_LO	Sets active trigger flank to the negative edge.
IS_SET_TRIG_LO_HI	Sets active trigger flank to the positive 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);
```

4.95 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. (See also *4.35 is_GetGlobalFlashDelays*).
If 0 is given for *ulDelay*, the normal flash function is used.
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*.

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	The resolution of the adjustable flash dduration.
ulDuration	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*.

4.96 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 *4.95 is_SetFlashDelay*). For cameras with Rolling Shutter sensors it is recommended to use the values from the function *is_GetGlobalFlashDelays ()* (see also *4.35 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 *4.94 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	Switches the strobe output off.
IS_SET_FLASH_LO_ACTIVE	Switches the strobe output Low-Active.
IS_SET_FLASH_HI_ACTIVE	Switches the strobe output High-Active.
IS_SET_FLASH_LO_ACTIVE_FREERUN	Switches the strobe output Low-Active in freerun mode.
IS_SET_FLASH_HI_ACTIVE_FREERUN	Switches the strobe output High-Active in freerun mode.
IS_SET_FLASH_HIGH	Switches the strobe output HIGH.
IS_SET_FLASH_LOW	Switches the strobe output LOW.
IS_GET_FLASHSTROBE_MODE	Return the current mode.
nLine	reserved

HINT

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

Return value:

IS_SUCCESS, IS_NO_SUCCESS, 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);

4.97 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 Framerate 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 DCU timing.
Similarly to the Exposure time changes at the window size or at the Pixelclock affect the Framerate.
Framerate 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 or *IS_NO_SUCCESS*

4.98 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.

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*

4.99 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 lies 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).

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*, sonst *IS_SUCCESS* or *IS_NO_SUCCESS*

Example:

Set Gamma value to 1.42:

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

4.100 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

HINT

In the case of use of the constant *IS_SET_ENABLE_AUTO_GAIN* for the parameter *EXP* the AutoGain functionality is activated. Setting a value will deactivate the functionality. (see also *4.76 is_SetAutoParameter*).

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*

4.101 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*

4.102 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.

Pparameters:

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_ MASTER_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 contains the value 363 for the UI-1460-C
```

Read the maximal gain factor of the red channel:

```
ret = is_SetHWGainFactor(hf, IS_INQUIRE_RED_GAIN_FACTOR, 100);
//ret contains the value 725 for the UI-1460-C
```

4.103 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

4.104 is_SetImageAOI

Syntax:

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

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 values:

IS_SUCCESS or *IS_NO_SUCCESS*.

4.105 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()* function *is_SetImageSize()* must follow to set the image size of the active memory. A pointer from function *is_AllocImgMem()* has to be given to parameter *pcImgMem*.

HINT

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

All other pointer lead to an error report! A repeated hand-over of the same pointer is allowed.

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

4.106 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()*. In this case, the following described order for selecting functions must be observed:

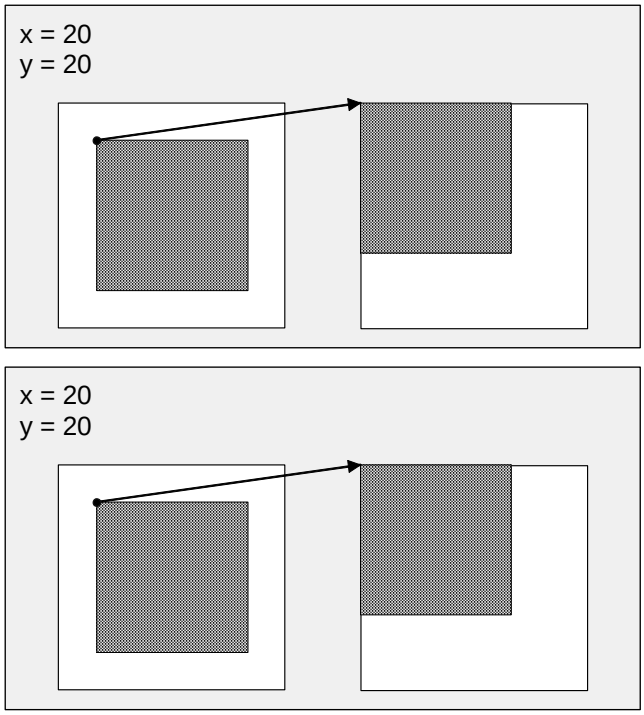
1. *is_SetImageSize()*
2. *is_SetImagePos()*

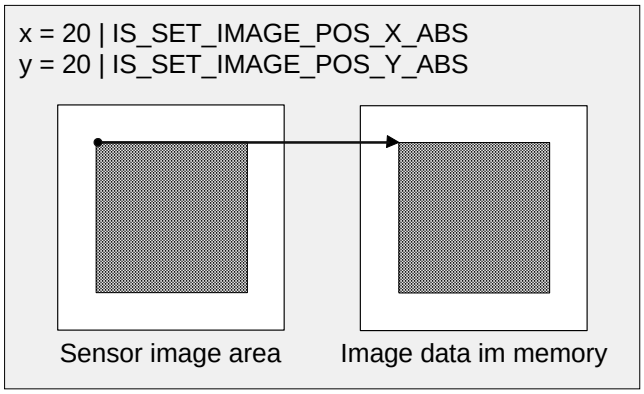
The function *is_SetAOI()* (see 4.75 is_SetAOI) combines the two functions. With *is_SetAOI()* the position and the size of an AOI can be set with one function call.

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 parameter *IS_SET_IMAGE_POS_X_ABS* and *IS_SET_IMAGE_POS_Y_ABS*.

Example:




x = 20 | IS_SET_IMAGE_POS_X_ABS
y = 20 | IS_SET_IMAGE_POS_Y_ABS
Sensor image area
Image data im memory

Boundary conditions

See *4.75 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_GET_IMAGE_POS_X_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 a parameter for x, else *IS_SUCCESS, IS_NO_SUCCESS*

4.107 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()*.
The following described order for selecting functions must be observed:

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

The function *is_SetOI()* (see *4.75 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 *4.75 is_SetAOI*.

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_INC	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*.

4.108 is_SetIO (only UI-1543-M)

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 bitmask
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 bitmask

Return value:

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

4.109 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_GREEEN	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

4.110 is_SetLED

Syntax:

INT is_SetLED (HIDS hf, INT nValue)

Description:

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

Parameters:

hf	Handle auf Kamera
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

4.111 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 Pixelclock has also influence on frame rate and the exposure time. The current recording of an image is cancelled.

Parameters:

hf	Camera handle
Clock	Pixelclock in MHz
IS_GET_PIXEL_CLOCK	Returns current Pixelclock
IS_GET_DEFAULT_PCLK	Returns standard pixelclock

Return value:

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

4.112 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 or the current settings in connection with *IS_SET_ROP_EFFECT*

4.113 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 color 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*.

4.114 is_SetSubSampling

Syntax:

INT is_SetSubSampling (HIDS hf, INT mode)

Description:

The function *is_SetSubSampling()* is used to activate horizontal and/or vertical *subsampling*. In the 2x mode 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 increased. This is equal to hardware scaling the image to 2:1. If a colour camera is used, 4:2 scaling is deployed in which two pixels are read and two left out.
Due to its design the UI-154x-M accomplishes only a colour subsampling. This can lead with fine picture structures to slight artifacts.

Monochrom-Sensor

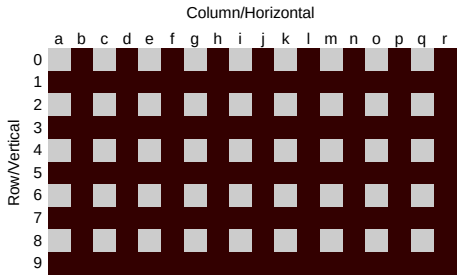


Fig. 8: twofold subsampling

Color-Sensor

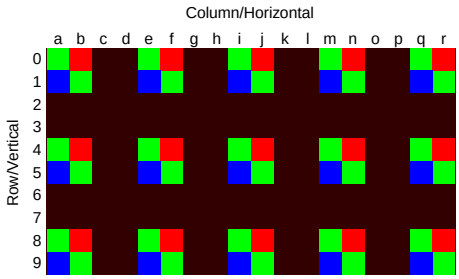


Fig. 9: twofold subsampling

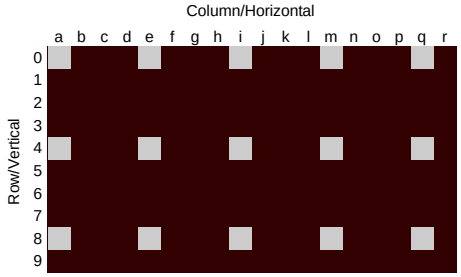


Fig. 10: fourfoldfold subsampling

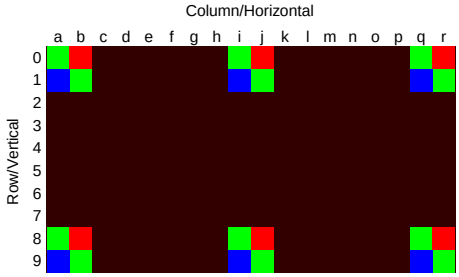


Fig. 11: fourfoldfold subsampling

Parameters:

hf	Camera handle
mode	
IS_SUBSAMPLING_DISABLE	Disables subsampling.
IS_SUBSAMPLING_2X_VERTICAL	Enables 2x vertical subsampling.
IS_SUBSAMPLING_4X_VERTICAL	Enables 4x vertical subsampling.
IS_SUBSAMPLING_2X_HORIZONTAL	Enables 2x horizontal subsampling.
IS_SUBSAMPLING_4X_HORIZONTAL	Enables 4x 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:

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

4.115 is_SetTestImage

Syntax:

INT is_SetTestImage (HIDS hf, INT nMode)

Description:

The function *is_SetTestImage()* switches on different pre-defined test imges 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. Testimages vary per recording one line (greyvalue 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* or *IS_NO_SUCCESS*.

4.116 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):

- 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, current settings in connection with *IS_GET_TRIGGER_DELAY*

4.117 is_SetWhiteBalance

Syntax:

INT is_SetWhiteBalance (HIDS hf, INT nMode)

Description:

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

HINT

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 color 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*

4.118 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 *4.117 is_SetWhiteBalance.*

HINT

is_SetWhiteBalanceMultipliers() set the parameters for the software color scaling which ist 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

4.119 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

4.120 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 preset color format is used. With the function *is_PrepareStealVideo()* can be selected if the picture muss be stolen or 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 *Fig. 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

4.121 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

4.122 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

4.123 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

4.124 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).
pcMem	Start address of image memory

HINT

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

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.125 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 *4.89 is_SetDisplayMode*.

Parameters:

hf	Camera handle

Return value:

IS_SUCCESS, IS_NO_SUCCESS

4.126 is_WriteEEPROM

Syntax:

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

Description:

In the DCU 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 to 63
pcString	Character chain, which contains certain data
Count	Number of readable characters

Return value:

IS_SUCCESS, IS_NO_SUCCESS

5 Error messages

Nr	Fehler	Beschreibung
-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 DCU SDk expect the amera handle as first parameter.
2	IS_IO_REQUEST_FAILED	An IO requirement of the DCU 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.
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 allcated.
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.
140	IS_CAPTURE_RUNNING	An aquisition 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)

Nr	Fehler	Beschreibung
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 Triggersignal.
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 color format.

Table of figures

Index of tables

Addresses

Our Company is represented by several distributors and sales offices throughout the world.

Europe
Thorlabs GmbH
Hans-Boeckler-Str. 6
D-85221 Dachau / Munich
Germany

Sales and Support
Phone: +49 (0) 81 31 / 5956-0
Fax: +49 (0) 81 31 / 5956-99
Email: europe@thorlabs.com
Web: www.thorlabs.com

USA
Thorlabs, Inc.
435 Route 206 North
Newton, NJ 07860
USA

Sales and Support
Phone: 1-973-579-7227
Fax: 1-973-300-3600
Email: sales@thorlabs.com
Email: techsupport@thorlabs.com
Web: www.thorlabs.com

Japan
Thorlabs, Inc.
6th Floor, Fujimizaka Building
5-17-1, Ohtsuka
Bunkyo-ku, Tokyo 112-0012
Japan

Sales and Support
Phone: +81-3-5977-8401
Fax: +81-3-5977-8402
Email: sales@thorlabs.jp
Web: www.thorlabs.jp

Please call our hotlines, send an Email to ask for your nearest distributor or just visit our homepage **http://www.thorlabs.com**