

3iCube

USB3.0 CMOS cameras



iCube

USB2.0 CMOS cameras



Before You Start

This manual should help you in installation and setting of the camera and we recommend you to carefully follow the instruction described.

To ensure that your warranty remains valid, read the manual carefully before using the camera.

DO NOT disassemble, modify or repair the camera since there is no user serviceable part inside and may void warranty. For prevention of fire or electric shock DO NOT remove screws or cover from the camera.

Operation in wet environment is NOT recommended and camera SHOULD NOT be exposed to rain or moisture. For long life and use of camera's CCD, do not point the camera directly to the sun or strong spotlight which may result CCD blooming and permanent damage. DO NOT operate camera beyond operation temperature range stated and AVOID usage in conditions exceeding 90% humidity.

DO NOT use unregulated power supply source to prevent camera's circuit damage.

Use soft materials such as lens tissue or cotton tipped applicator with ethanol for CCD faceplate cleaning ONLY when necessary and AVOID contact with fingers or any hard object. Do not use solvent, abrasives or detergent in case of cleaning camera body.

Warranty shall be voided for improper usage or fault caused by user or damage caused by other equipments due to negligence

Warranty

NET GMBH warrants the original components free of defects for one year from purchase date. This warranty covers failures and damage due to defect which may occur during normal use. It does not cover damages or failure resulting from mishandling, abuse, misuse or modification. For every repair or replacement, RMA numbers must be obtained in advance.

Disclaimer

The information in this document has been carefully checked and is believed to be reliable. However, no responsibility is assumed for inaccuracies, nor is any responsibility assumed by NET GMBH. There is no legal obligation to documenting internal relationships in any functional module of its products, which is realized in either hardware or software.

Copyright

All the materials in this document are protected by copyright and other laws for intellectual property. They are not allowed to be copied, reproduced or modified for any use without the permission of NET GmbH. NET GMBH reserves the right to make changes in specifications, functions or designs at any time and without any notice. The company names in this document may be the trademarks and trade-names of their respective owner and are hereby acknowledged.

Copyright © 2014 NET GMBH. All rights reserved.

Table of Contents

TABLE OF CONTENTS	3
LICENSE	5
LIMITED LICENSE FOR EVALUATION VERSION	5
NOTE	5
LEGAL NOTICE	5
LIST OF FIGURES	6
LIST OF TABLES	6
STANDARD CONFORMITY	7
SAFETY PRECAUTIONS	6
SYSTEM REQUIREMENTS	11
HARDWARE REQUIREMENTS.....	11
SOFTWARE REQUIREMENTS	11
USAGE NOTES.....	12
SOFTWARE.....	15
SOFTWARE CD.....	15
SOFTWARE INSTALLATION	17
SOFTWARE AND DRIVER UPDATE	18
PROBLEMS	18
APPLICATIONS	19
ICUBE ICONTROL	19
CALIBRATION (OPTIONAL)	19
ICUBE DX-REGISTRATION	20
ICUBE SDK SAMPLES (WINDOWS)	21
SDK-INTERFACE	22
MICROSOFT VISUAL STUDIO / C++	22
C++ BUILDER	22
VISUAL BASIC .NET.....	22
VISUAL BASIC 6.0	22
C#	23

AVAILABLE FUNCTIONS	24
FUNCTION DETAILED OVERVIEW	26
CAMERA CONTROL FUNCTIONS	26
VERSION FUNCTIONS	33
ROI FUNCTIONS	35
MODE FUNCTIONS	37
BIN SKIP FUNCTIONS	39
SAVE FUNCTIONS	41
TRIGGER FUNCTIONS	43
PARAMETER FUNCTIONS	45
EXPOSURE FUNCTIONS	47
ADDITIONAL PARAMETER FUNCTIONS	49
ERROR CODES	51
DIRECTSHOW-INTERFACES	52
SUPPORTED STANDARD-DIRECTSHOW-INTERFACES	52
ICUBE-DIRECTSHOW-INTERFACES.....	52
ICUBE-DIRECTSHOW-SETTING	53
VIDEO CONTROL PARAMETERS	54
VIDEO STREAM CONTROL PARAMETERS.....	56
TECHNICAL SUPPORT.....	57
IMPRINT.....	58

License

Limited License for Evaluation Version

Evaluation version of the NET GmbH Camera API(**iCube**SDK Library) is only compliant with cameras manufactured by NET GmbH and may not be operable with other cameras.

User may purchase the license by contacting our sales department or your local distributor for unlimited use of the API and its function. Please refer to the standard EULA documents for details concerned with API License.

Note

NET GmbH Camera API(**iCube**SDK Library) only supports NET GmbH hardware and strictly forbidden to use or build Application for cameras or hardware from other vendors with this API. The EVALUATION VERSION SOFTWARE is provided to you "AS IS" without warranty. The entire risk of the quality and performance of the software is with its users. We would appreciate feedback bug report of any kind, however, we can not guarantee satisfactory response.

Legal Notice

By installing, copying or otherwise using the SOFTWARE, you agree to be bound by the terms of the End User License Agreements (EULA). The SOFTWARE includes NET GmbH and NET GmbH suppliers' intellectual property.

Please read NET GmbH and NET GmbH suppliers' EULA before installing the SOFTWARE. If you do not accept the terms of the license agreements, please do not install copy or use this SOFTWARE.

List of Figures

FIGURE 1:	C-MOUNT LENS.....	13
FIGURE 2:	DEVICE MANAGER	17
FIGURE 3:	ICONTROL VIEWER SOFTWARE.....	19
FIGURE 4:	MACBETH STANDARD COLOR CHECKER	19
FIGURE 5:	ICUBE DX REGISTRATION.....	20
FIGURE 6:	VIDEO CONTROL PARAMETERS.....	53
FIGURE 7:	CAMERA CONTROL PARAMETERS.....	55
FIGURE 8:	VIDEO STREAM CONTROL PARAMETERS	56

List of Tables

TABLE 1:	STANDARD CAMERA FUNCTION CONTROL.....	24
TABLE 2:	ERROR CODES	51
TABLE 3:	VIDEO CONTROL PARAMETERS	54
TABLE 4:	CAMERA CONTROL PARAMETERS	55
TABLE 5:	VIDEO STREAM CONTROL PARAMETERS.....	56

Standard Conformity

Legal Notice

The cameras fully implement the USB3.0 standard.

RoHS II

The product fulfills the requirements of the **EU directive RoHS 2011/65/EU** in the currently valid version from 8 June 2011 regarding the restrictive use of certain hazardous materials in electric applications within the allowable limits.

FCC

This equipment has been tested and found to comply with the limits for a class A digital device, pursuant to Part 15 of the FCC Rules. These limits are designed to provide reasonable protection against harmful interference when the equipment is operated in a commercial environment. This equipment generates, uses, and can radiate radio frequency energy and, if not installed and used in accordance with the instruction manual, may cause harmful interference to radio communication. Operation of this equipment in a residential area is likely to cause harmful interference in which case the user will be required to correct the interference at his own expense.

CE



This apparatus has been certified to meet or exceed the standards for CE compliance per Council Directives. Pertinent testing documentation is available for verification. This product following the provision of directive 2004/108/EC.

Safety Precautions

Before using this product read these safety precautions carefully. Important information is shown in this Operational Manual to protect users from injuries and property damages and to enable them to use the product safely and correctly.

Please be sure to thoroughly understand the meanings of the following signs and symbols before reading the main text that follows, and observe the instructions given herein.

[Definition of Safety Signs]

Safety Signs	Description
 WARNING	Indicates a potentially hazardous situation that may result in death or serious injury (*1) in the event of improper handling.
 CAUTION	Indicates a potentially hazardous situation that may result in light to moderate injuries (*2) or only in property damage (*3) in the event of improper handling.



Notes

*1: "Serious injury" refers to cases of loss of eyesight, wounds, burns (high or low temperature), electric shock, broken bones, poisoning, etc., which leave after-effects or which requires hospitalization or a long period of outpatient treatment of cure.

*2: "Light to moderate injuries" refers to injuries, burns, electric shock etc. that do not require hospitalization or long-term treatment.

*3: "Property damage" refers to cases of extensive damage involving damage to buildings, equipment, farm animals, pet animals and other belongings.



[Explanation of Safety Symbols]

Safety Symbols	Description
 PROHIBITED	This sign indicates PROHIBITION (Do not). The content of prohibition is shown by a picture or words beside the symbol.
 MANDATORY	This sign indicates MANDATORY ACTION (You are required to do). The content of action is shown by a picture or words beside the symbol.

General Handling

WARNING



 Unplug	Stop operation immediately when any abnormality or defect occurs. If abnormal conditions are present, such as smoke, a burning smell, ingress of water or foreign matter, or if the equipment is dropped or malfunctions, fire or electric shock may result. Be always sure to disconnect the power cable from the wall socket at once and contact your dealer.
 Do not wet	Do not use the equipment in locations subject to water splashes. Otherwise, fire or electric shock may result.



Never pull
apart

Do not disassemble, repair, or modify the equipment. Otherwise, fire or electric shock may result. For internal repair, inspection or cleaning, contact your sales representative.



Avoid

Do not place anything on the equipment.

If metallic objects, liquid, or other foreign matter enters the equipment, fire or electric shock may result.



Avoid

Do not install the equipment in an unstable or inclined location or locations subject to vibration or impact. Otherwise, the equipment may topple over and cause personal injury.



Do not touch

During an electrical storm, do not touch the power cable and the connection cable. Otherwise, an electric shock may result.



Instruction

Use the specified voltage. Use of an unspecified voltage may result in fire or electric shock.



Avoid

Do not handle roughly, damaged, fabricated, bent forcefully, pulled, twisted, bundled, placed under heavy objects or heated the power cable and the connection cable. Otherwise, fire or electric shock may result.

CAUTION



Instruction

Observe the following when installing the equipment:

Do not cover the equipment with a cloth, etc.

Do not place the equipment in a narrow location where heat is likely to accumulate.

Otherwise, heat will accumulate inside the equipment, possibly resulting in a fire.



Avoid

Do not place the equipment in locations subject to high moisture, oil fumes, steam, or dust. Otherwise, fire or electric shock may result.



Avoid

Do not install the equipment in locations exposed to direct sunlight or humidity.

Otherwise, the internal temperature of the equipment will rise, which may cause a fire.



Use only specified the power cable and the connection cables. Otherwise, fire or electric shock may result.

Instruction



Avoid

Do not give strong impact against the equipment. It may cause the trouble.



Instruction

When performing connection, turn off power. When connecting the power cable and the connection cable, turn off the equipment power. Otherwise, fire or electric shock may result.



Avoid

Do not expose the camera head to any intensive light (such as direct sunlight). Otherwise, its inner image pickup device might get damaged.



Avoid

Avoid short-circuiting signal output. Otherwise, a malfunction may occur.



Avoid

Avoid giving a strong shock against the camera body. It might cause a breakdown or damage. If your camera is used in a system where its camera connector is subjected to strong repetitive shocks, its camera connector is possible to break down. If you intend to use your camera in such a situation, if possible, bundle and fix a camera cable in the place near the camera, and do not transmit a shock to the camera connector.

System Requirements

Hardware requirements

- USB 3.0 on board Interface. NET successfully tested USB 3.0 adapters, which use the **Renesas** chipset **μPD720202** or **μPD720200A**. Please be sure that you installed the latest USB 3.0 adapter driver.
- lockable SuperSpeed USB3.0 cable up to 3m. If you want to use your own USB 3.0 cables, you have to ensure that the data quality and shielding of the cable is sufficient. Better cable qualities which go alongside with thicker cable diameter will allow longer distances. We recommend using the cables we supply to be on the safe side.
- state of the art PC or notebook. (minimum Pentium IV processor with a clock frequency of at least 1.5 GHz or higher)

Software requirements

iCube iControl– Viewer Software

The iControl software allows you to test the functionalities of the 3iCube camera on your own application. Apart from controlling the 3iCube camera, you can grab images and save them as jpg, bmp and tiff files

SynView – Software Development Kit (SDK)



/ only with USB3 Vision

compliance	USB 3.0 standard
supported image processing libraries	<i>Adaptive Vision Studio, Halcon, Imaging Library, VisionPro, LabView Vision, Matlab (and all GenTL consumer)</i>
supported operating systems	<i>Windows XP (32 bit), Windows 7 (32/64 bit), Windows 8 (32/64 bit), Linux (32/64 bit)</i>

All necessary drivers for Windows and Linux are contained on the CD-ROM. For newer driver versions we recommend to visit the NET website at www.net-gmbh.com

Usage Notes

Read the documentation

Read the camera documentation before using the camera.

Camera power

Incorrect input power can damage the camera. Do not reverse power polarity. Do not connect or disconnect other cables when the camera power is on. Use always a USB 3.0 cable as power supply supported by USB 3.0 port.

Opening the camera

Do not open the camera. Do not let liquid, dust, flammable or metallic material to get inside the camera.

Environmental storage conditions

Temperature: -20°C ~ 60°C (-4°F 140°F)

Humidity: 90% or less (no condensation)

Environmental operating conditions

Always use the camera in conditions meeting the specification in this chapter. Do not use the product in locations where the ambient temperature or humidity exceeds the specifications. In a thermal challenging environment the customer needs to ensure sufficient heat dissipation with a thermal connection to the bottom of the camera housing and sufficient airflow.

Non adequate thermal connection may increase heat induced noise or degrade image quality in other ways and internal components may be adversely affected up to camera outages due to overheating.

Temperature	Range	Measurement
Environmental	0°C ~ 45°C (32°F 113°F)	close to the camera case
Camera housing	≤ 50°C	at camera case

Humidity	Relative	
Environmental	20 %–90 %	non-condensing

The conditions for shock and vibrations are *on request* by NET.

Maintenance

Turn off power to the equipment and wipe it with a dry cloth. If it becomes severely contaminated, gently wipe the affected areas with a soft cloth dampened with diluted neutral detergent. Never use alcohol, benzene, thinner, or other chemicals because such chemicals may damage or discolor the paint and indications.

Cleaning the sensor window

Avoid cleaning the sensor window if possible. Keep lens cap closed as long as no lens is attached, avoid touching the sensor. If necessary, clean the sensor window using compressed air. If further cleaning is required, use lint-free, ESD-safe cloth wiper. Avoid cloth that could generate static charge or that could scratch the window. The camera should be cleaned in an ESD-safe area. The person performing cleaning should be earthed.

Connectors

Take care when handling the camera so that no damage can be done to the connectors. Prevent foreign objects in the connectors.

Handle carefully

Always transport the camera in its original packaging. Do not drop the equipment or allow it to be subject to strong impact or vibration, as such action may cause malfunctions. Do not damage the connection cable, since this may cause wire breakage. If the camera is not in use, attach the lens cap to the camera to protect the image pickup surface. If the equipment is not to be used for a long duration, turn off power to the camera for safety.

Check compatibility of lens

Depending on lens and lighting an image can be reflected as a ghost into the imaging area. This is not a fault of the camera. Depending on the lens the performance of the camera might not be brought out fully due to deterioration in resolution and brightness in the peripheral area, aberration and other side effects. Be sure to check lens and lightning you plan to use for compatibility with your camera. When installing a lens in the camera make sure that it is not tilted. Use a mounting screw free from defects and dirt. Otherwise the lens might not be removable from the camera. Install lenses with a protrusion from bottom of the screw equal or less than 10 mm. If a lens does not fulfill this condition it might damage the camera when trying to be installed.

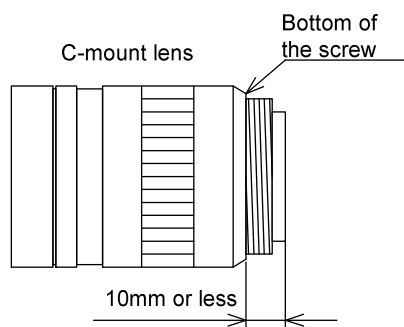


Figure 1: C-Mount Lens Dropping Frames

Depending on your PC or USB3.0 interface board configurations, images may not be captured properly (e.g. dropping frames). In this case, change pixel clock setting to lower value.

Occurrence of moiré

If you shoot thin stripe patterns, moiré patterns (interference fringes) may appear. This is not a malfunction.

Electromagnetic fields

Keep the camera away from strong electromagnetic fields. Avoid static charging and handle the camera in ESD protected area. If an intense magnetic or electromagnetic field is generated near the camera or connection cable, noise may be generated on the screen. If this occurs, move the camera or the cable.

Following information is only for EU-member states:

The use of the symbol indicates that this product may not be treated as household waste. By ensuring this product is disposed correctly, you help to prevent potential negative consequences for the environment and human health, which could otherwise be caused by inappropriate waste handling of this product. For more detailed information about the take-back and recycling of this product, please contact your supplier where you purchased the product.



Software

Software CD

The software CD includes the following directories:

WINDOWS

00_Documentation

3iCube Operation Manual
3iCube SDI API Manual

03_Driver

3iCube Camera Device Driver x32 / x64
3iCube Cognex AIK Setup

04_Viewer SW

iControl: (viewer Software)

05_Interfaces

DShow (additional COM-interface for DShow applications)
SDK
- 4133_MultiROI: (Microsoft Visual Studio multi roi example)
- C#: (C# SDK Example)
- C++: (Microsoft Visual Studio examples)
- C++Builder: (Borland C++ Builder SDK Example)
- iCubeSDKSample_x32_VC6: (Visual Studio 6.0 32bit SDK Example)
- iCubeSDKSample_x32_x64_vs2010: (Visual Studio 2010 32bit/64bit SDK Example)
- vb.6: (Visual Basic 6.0 SDK Example)
- VB.NET (VB.NET SDK Example)

06_Tools

dxRegistration:
Register more than one device as direct show -filter; fix positioning for direct show and
SDK (see readme.txt in this folder)

UnInstall_V10_ICUBE_Driver
3iCube driver uninstaller

LINUX

00_Documentation

3iCube Operation Manual
3iCube SDI API Manual

03_Driver

netusbcam_x.xx-1_i386_libudev.deb: (debian packet which uses libudev interface)
netusbcam_x.xx-1_i386_usbfs.deb: (debian packet which uses usbfs interface, used for older debian distrutions)
readme.txt: (describes requirements of usbfs and libudev packets)
99-netusbcam.rules: (rules for netusbcam libudev packet)

05_Interfaces

SDK (SDK.tar.gz): (SDK packet)
API: (NETUSBCAM_API.h)
HelloICube: (minimal iCube example)
MultiROI_SIMR_Test: (multi roi sdk example)
MultiROITest (multi roi sdk example)
sdk_sample1: (QT based SDK example)

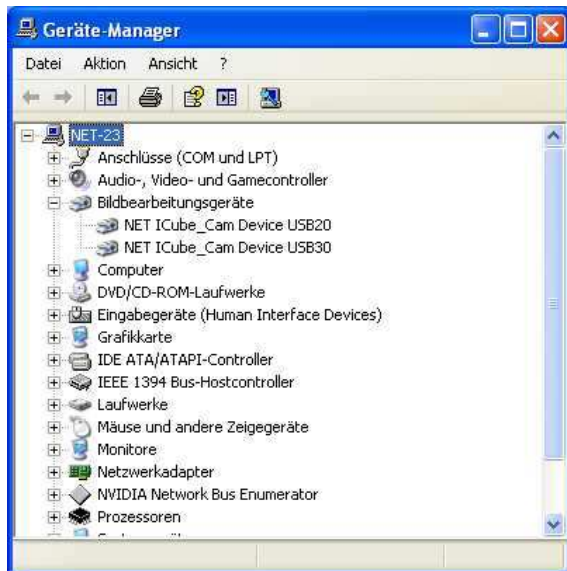
Software Installation (Windows)

Administrator rights are necessary for installing a driver

1. Copy the CD-Rom to your computer directory.
2. Plug in the USB 3.0 cable into your USB 3.0 port and 3iCube.
3. Windows plug and play manager recognizes the new hardware.
4. Follow the instruction of the Windows plug and play manager.
5. After the 3iCube driver is installed, you can see on the device Manager / imaging devices the recognized 3iCube camera.

3iCube: → NET ICube_Cam Device USB30

Windows (german version)



Windows (english version)

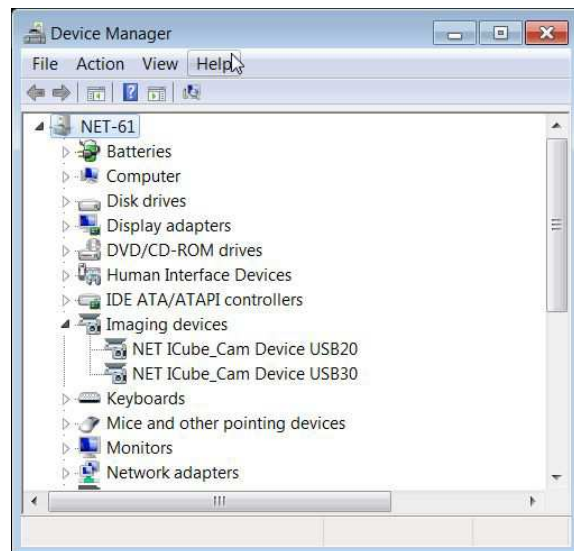


Figure 2: Device manager

Software and Driver update

The latest drivers and documentations are available on NET's homepage www.net-gmbh.com.

The software package includes following files:

- iControl viewer software
- USB driver
- API
- iCube Cognex AIK Setup

Please install the full package (iControl and USB driver) to get the right function.

After you have installed the full software package, you have to update the camera driver. If an 3iCube camera is connected to the PC, please update the camera-driver (new .inf file) on the device manager (imaging devices) and select the driver manually.

Problems

Due to heavy real-time data transfer and processing, system performance (especially CPU) is crucial for smooth operation. Possible performance degradation such as actual frame rate drop may occur for systems with lower performance than of Pentium IV 1.5 GHz computer. Faulty cables can drop the frame rate. The maximum of the bandwidth is defined by the USB chip set and the internal PC hardware.

If you can see following effects, please reduce the pixel clock of the 3iCube camera or disable the smart power management (CPU sleep states) of the PC:

Effects:

- Black image
- Bad frames
- Surge image
- No maximal frame rate

Applications

iCube iControl viewer software

The iControl software allows you to test the functionalities of the 3iCube camera on your own application. Apart from controlling the 3iCube camera, you can grab images and save them as jpg, bmp and tiff files.

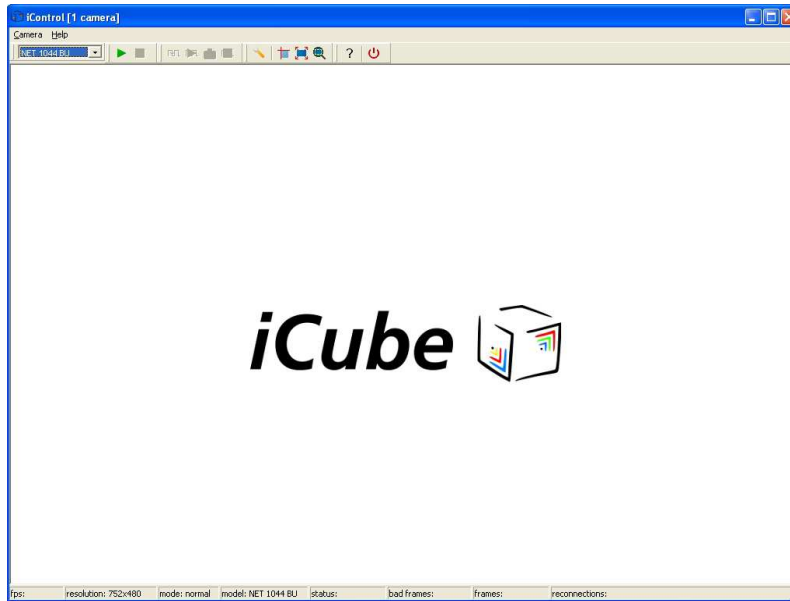


Figure 3: iControl viewer software

Calibration (optional)

The **Color Calibration** application uses a Macbeth standard color checker to evaluate the predefined color spots and calculate the correction values for the RGB color matrix in the camera.



Figure 4: Macbeth standard color checker

iCube dx-Registration

The 3iCube dx-Registration software is to register more than one device as dx-capture filter

The 3iCube dx-Registration software can be accessed as follows:

Connect all **3iCube** cameras to PC.

1) Choose device to register.

You will see the connected camera with serial numbers in the ComboBox.

The selection of the dx-capture filter in 2) will change automatically,
when changing the device.

2) Register the selected device.

The name in the square brackets is the dx-friendly-name,
which will appear in amcap for example.

Additional functionalities are explained in Tools\dxRegistration\readme.txt

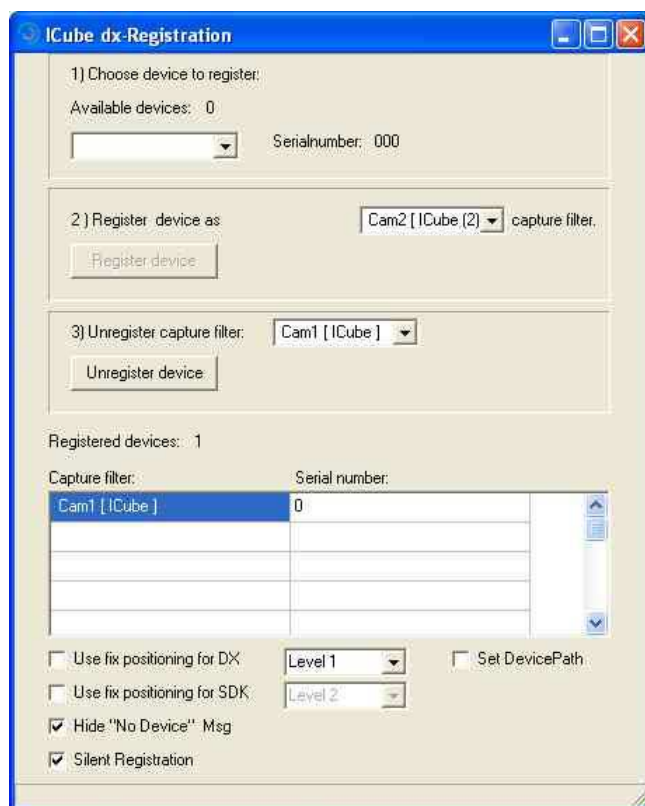


Figure 5: iCube dx registration

iCube SDK samples (windows)

The iCube SDK samples shows, how you can develop or integrate the 3iCube in your own application software.

On the CD-Rom you can find examples for following development software:

- | | |
|----------------------------------|---|
| - 4133_MultiROI: | (Microsoft Visual Studio multi-roi example) |
| | This example works with 4133 and 4203 cameras only. |
| | The multi-roi offers two modes: |
| | <ul style="list-style-type: none"> • MIMR (Multiple Integration Multiple ROI) mode allows the user to define an acquisition cycle comprising 1 to 4 ROI cycle(s). • SIMR (Single Integration Multiple ROI) mode allows 1, 2 or 4 areas of interest to be acquired within the same integrated image. |
| - C++: | (Microsoft Visual Studio examples) |
| - C++Builder: | (Borland C++ Builder SDK Example) |
| - ICubeSDKSample_x32_VC6: | (Visual Studio 6.0 32bit SDK Example) |
| - ICubeSDKSample_x32_x64_vs2010: | (Visual Studio 2010 32bit/64bit SDK Example) |
| - VB.NET: | (VB.NET SDK Example) |
| - C# : | (C# SDK Example) |
| - vb.6.0: | (Visual Basic 6.0 SDK Example) |

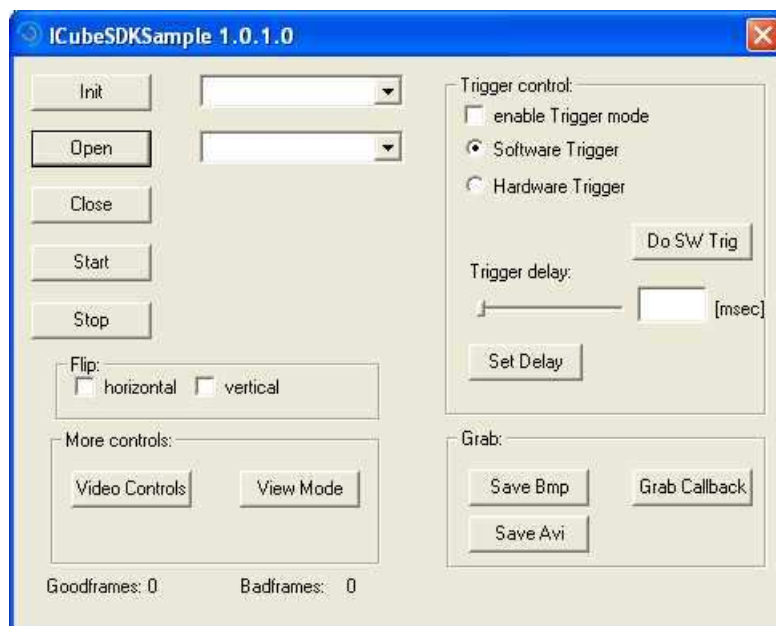


Figure 6: iCube SDK sample

SDK-Interface:

Setup:

Microsoft Visual Studio / C++:

Static linking:

If the user want to use „**iCube**SDK.lib“ file while compiling, add the path lib/ **iCube**SDK.lib to library modules.

Uncomment the define „#define LIBFILE“ in the "NET_**iCube**_API.h" file.

dynamic linking:

Comment the define „#define LIBFILE“ in the "NET_**iCube**_API.h" file.

C++ Builder:

Static linking:

If the user want to use „**iCube**SDK.lib“ file while compiling, add the path lib/ **Icube**SDK.lib to library modules.

Uncomment the define „#define LIBFILE“ in the "NET_**iCube**_API.h" file.

Uncomment the define „#define BORLAND_C“ in the "NET_**iCube**_API.h" file.

dynamic linking:

Comment the define „#define LIBFILE“ in the "NET_**iCube**_API.h" file.

Visual Basic .NET:

available Visual Basic.NET example:

Interface definitions: **iCube**_API.vb

Visual basic 6.0

available Visual Basic 6.0 example:

Interface definitions: **ICube**_API.bas

C#

available C# example:

Interface definitions: *ICube.cs*

Available functions:

Explanation of constants like IC_SUCCESS, ON, PARAM_PROPERTY, ... see api definition files (c++: NET_iCube_API.h, ...).

For the linux SDK the functions are called NETUSBCAM_ instead of iCube_SDK_.

Differences are explained at the respective function explanations.

Category	Function	Description
Camera control functions	<i>iCube SDK Init</i>	<i>initializes the camera connected to your computer</i>
	<i>iCube SDK Open</i>	<i>opens the camera interface</i>
	<i>iCube SDK Close</i>	<i>closes the camera interface</i>
	<i>iCube SDK_IsOpen</i>	<i>checks for open camera interfaces</i>
	<i>iCube SDK_IsOpenEx</i>	<i>Checks for open camera interfaces, also in other processes (programs)</i>
	<i>iCube SDK_SetCallback</i>	<i>sets the callback function</i>
	<i>iCube SDK_SetCallback</i>	<i>sets the event-callback function</i>
	<i>iCube SDK_Start</i>	<i>starts the video stream</i>
	<i>iCube SDK_IsStarted</i>	<i>checks for open image stream</i>
	<i>iCube SDK_Stop</i>	<i>stops the video stream</i>
	<i>iCube SDK_GetSize</i>	<i>get the current frame sizes</i>
	<i>iCube SDK_GetName</i>	<i>get the name of the selected camera</i>
	<i>iCube SDK_GetBrokenFrames</i>	<i>get the number of broken frames since the last start</i>
	<i>iCube SDK_GetGoodFrames</i>	<i>get the number of good frames since the last start</i>
Version function	<i>iCube SDK_SetDisplayMode</i>	<i>sets the display mode</i>
	<i>iCube SDK_GetVersion</i>	<i>get the SDK version</i>
	<i>iCube SDK_GetFWVersion</i>	<i>get the firmware version</i>
	<i>iCube SDK_GetSerialNum</i>	<i>get the serial number of the camera</i>
ROI function	<i>iCube SDK_GetFPGAVersion</i>	<i>get the camera fpga firmware version</i>
	<i>iCube SDK_SetResolution</i>	<i>set the resolution and position of the Region of Interest</i>
	<i>iCube SDK_GetResolution</i>	<i>get the resolution and position of the Region of Interest (ROI).</i>
	<i>iCube SDK_GetResolutionRange</i>	<i>get the min/max resolution of the</i>

		Region of Interest (ROI)
	<i>iCube SDK_SetResolutionParam</i>	starts the image stream of roi 2-4 in multi roi applications
Table 1: Standard camera function control		
Mode function	<i>iCube SDK_SetMode</i>	the basic format (e.g. 640x480)
	<i>iCube SDK_GetMode</i>	get the basic format
	<i>iCube SDK_GetModeList</i>	get the possible formats of the camera
Bin Skip function	<i>iCube SDK_SetBinSkip</i>	set the camera into a skipping or binning mode
	<i>iCube SDK_GetBinSkip</i>	get the current skipping or binning mode
	<i>iCube SDK_GetBinSkipList</i>	get the possible skipping or binning formats of the camera
Save functions	<i>iCube SDK_SaveToFile</i>	saves a bitmap, jpg or tiff
	<i>iCube SDK_SaveAVI</i>	saves an avi stream to the hard disk
Trigger function	<i>iCube SDK_SetTrigger</i>	sets the Trigger mode
	<i>iCube SDK_GetTrigger</i>	gets the current trigger mode
Parameter functions	<i>iCube SDK_SetCamParameter</i>	set parameter value
	<i>iCube SDK_GetCamParameter</i>	get parameter value
	<i>iCube SDK_GetCamParameterRange</i>	get parameter min/max values, default value, auto, onepush and enabled information
Exposure functions	<i>iCube SDK_SetExposure</i>	set Exposure time (Input)
	<i>iCube SDK_GetExposure</i>	get Exposure time (Output)
	<i>iCube SDK_GetExposureRange</i>	get Exposure time Range (Output)
Color Transformation Control	<i>iCube SDK_GetParamAuto</i>	check, if the parameter supports auto mode
	<i>iCube SDK_SetParamAuto</i>	if auto mode is supported, set/unset auto mode of parameter
	<i>iCube SDK_SetParamDef</i>	set parameter to default setting
	<i>iCube SDK_SetParamOnePush</i>	if one push mode is supported, set/unset one push mode of parameter

Functions detailed overview:

Camera control functions:

int iCube SDK_Init()

Purpose: initializes the camera connected to your pc.

	Description
Parameters	-
Return value	Int CamCount

Return value:

int CamCount: number of connected cameras.

int iCube SDK_Open(int nCamIndex)

Purpose: opens the camera interface.

	Description
Parameters	int CamIndex
Return value	Int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera to open.

void iCube SDK_Close (int nCamIndex)

Purpose: closes the camera interface.

	Description
Parameters	int CamIndex
Return value	Int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera to close.

void iCube SDK_IsOpen (int nCamIndex)

Purpose: checks for open camera interfaces.

	Description
Parameters	int CamIndex
Return value	-

Return value:

int stat: ON: camera is open
 OFF: camera is not open

Parameters:

int nCamIndex: index of camera.

void iCube SDK_IsOpenEx (int nCamIndex)

Purpose: checks for open camera interfaces, also in other processes (programs).

	Description
Parameters	int CamIndex
Return value	-

Return value:

int stat: IC_SUCCESS: camera is open
 else: camera is not open

Parameters:

int nCamIndex: index of camera.

***int iCube SDK_SetCallback (int nCamIndex,int nMode,long
(CALLBACK *CallbackFunc), void* pCBContext)***

Purpose: sets the callback function.

Only necessary, if Callback-Flag in “**iCubeSDK_Start**” is true.

The structure of the callback-function has to be as follows:

long CALLBACK CallbackFunc(BYTE * pBuffer, long lBufferSize,PVOID pContext)

Parameters passed by the Callback-function:

BYTE * pBuffer: pointer to frame data.

long lBufferSize: size of the frame data buffer.

PVOID pContext: context parameter (see below).

	Description
Parameters	int CamIndex, int nMode, long (CALLBACK* CallbackFunc), void* pCBContext
Return value	Int stat

Return value:

int stat: IC_SUCCESS: success
else: error

Parameters:

int nCamIndex: index of camera.
int nMode: defines the grab mode.

CALLBACK_RAW:

Data-format, passed to the callback function:

- color-camera: 8bit/Pixel (Bayer-raw Data)
- monochrome -camera: 8bit/Pixel (Raw Data)

CALLBACK_RGB:

Data-format, passed to the callback function:

- color-camera: 24bit/Pixel (RGB24 Data)
- monochrome-camera: 24bit/Pixel (RGB24 Data)
- color order in callback data is BGR.

long* CallbackFunc: pointer to the callback function.

void* pCBContext: pointer to user defined argument, which is passed by the callback function.

int iCube SDK_SetCallbackEx (int nCamIndex,int nMode,long (CALLBACK *CallbackFunc), void* pCBContext)

Purpose: sets the event-callback function.

Only necessary, if Callback-Flag in “**iCubeSDK_Start**” is true.

The structure of the callback-function has to be as follows:

long CALLBACK CallbackFunc(int event_type, BYTE * pBuf, long lBufferSize, PVOID pContext)

Parameters passed by the Callback-function:

EVENT_NEW_FRAME	0	// new frame
EVENT_DEV_DISCONNECTED	1	// a camera is disconnected
EVENT_DEV_RECONNECTED	2	// a camera is reconnected
EVENT_USB_TRANSFER_FAILED	3	// a usb transaction error occurred during streaming

BYTE * pBuffer: pointer to frame data.

long lBufferSize: size of the frame data buffer.

PVOID pContext: context parameter (see below).

	Description
Parameters	int CamIndex, int nMode, long (CALLBACK* CallbackFunc), void* pCBContext
Return value	Int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera.
int nMode: defines the grab mode.

CALLBACK_RAW:

Data-format, passed to the callback function:

- color-camera: 8bit/Pixel (Bayer-raw Data)
- monochrome -camera: 8bit/Pixel (Raw Data)

CALLBACK_RGB:

Data-format, passed to the callback function:

- color-camera: 24bit/Pixel (RGB24 Data)
- monochrome-camera: 24bit/Pixel (RGB24 Data)
- color order in callback data is BGR.

long* CallbackFunc: pointer to the callback function.

void* pCBContext: pointer to user defined argument, which is passed by the callback function.

int iCube SDK_Start (int nCamIndex,HWND ImgHandle,bool Preview,bool Callback)

Purpose: starts the video stream

Linux: NETUSBCAM_Start uses only the parameter nCamIndex, because there is no internal preview functionality implemented.

	Description
Parameters	int CamIndex, HWND ImgHandle, bool Preview, bool CallBack
Return value	Int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera.
HWND ImgHandle: handle to preview window.
bool Preview: set/unset the use of a preview window. If preview mode is set and ImgHandle is NULL, a default preview window will be used.
bool Callback: set/unset the use of the callback function.

void iCube SDK_IsStarted (int nCamIndex)

Purpose: checks for open image stream.

	Description
Parameters	int CamIndex
Return value	-

Return value:

int stat: ON: image stream is open
 OFF: image stream is not open

Parameters:

int nCamIndex: index of camera.

int iCube SDK_Stop (int nCamIndex)

Purpose: stops the video stream.
 Closes the default preview window, if used.
 Stop resets the trigger status to stop

	Description
Parameters	int CamIndex
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera.

int iCube SDK_GetSize(int nCamIndex,int* pnXRes,int* pnYRes)

Purpose: get the current frame sizes.

	Description
Parameters	int CamIndex, int* pnXRes, int* pnYRes
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera.
 int* pnXRes: current width.
 int* pnYRes: current height.

int iCube SDK_GetName(int nCamIndex, char* Name)

Purpose: get the name of the selected camera

	Description
Parameters	int CamIndex, int* Name
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera.
 char* Name: name of the camera.

int iCube SDK_GetBrokenFrames(int nCamIndex, int* pnFrames)

Purpose: get the number of broken frames since the last start.
 The broken frames information can also be retrieved via the callback function, if the parameter REG_CALLBACK_BR_FRAMES is set to ON.
 In this case a buffersize of 0 indicates a badframe.

	Description
Parameters	int CamIndex, int* pnFrames
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera.
 int* pnFrames: the number of broken frames since the last start.

int iCube SDK_GetGoodFrames(int nCamIndex, int* pnFrames)

Purpose: get the number of good frames since the last start.

	Description
Parameters	int CamIndex, int* pnFrames
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera.
 int* pnFrames: the number of good frames since the last start.

int ICubeSDK_SetDisplayMode(int nCamIndex, int nMode, DISP_PROPERTY property)

Purpose: sets the display mode. This function has to be called before ICubeSDK_Start().

	Description
Parameters	int CamIndex, int nMode, DISP_PROPERTY property
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera.
 int nMode: DISPLAY_NORMAL displays the full frame
 DISPLAY_FIT_TO_WINDOW resizes the displayed frame to the size
 of the preview window
 DISPLAY_RECT resizes the displayed frame to the size
 given with DISP_PROPERTY

DISP_PROPERTY property: defines height, width and position of the displayed window, only
 used in DISPLAY_RECT-Mode
 (left / top; right / bottom)

Version functions:

void iCube SDK_GetVersion(int nType ,char* pVersion)

Purpose: get the SDK version

	Description
Parameters	int nType, char* pVersion
Return value	-

Return value:

none.

Parameters:

int Type: type of parameter to get. (See Parameter Definitions in API Header)

char* version: the version of the SDK dll

void iCube SDK_GetFWVersion(int nCamIndex,char* pVersion)

Purpose: get the firmware version

	Description
Parameters	int CamIndex, char* pVersion
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera.

char* version: the version of the camera firmware

int iCube SDK_GetSerialNum(int nCamIndex,char* pVersion)

Purpose: get the serial number of the camera

	Description
Parameters	int CamIndex, char* pVersion
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera

char* pVersion: SerialNumber

void iCube SDK_GetFPGAVersion(int nCamIndex, char* pVersion)

Purpose: get the camera FPGA firmware version.
 This functions returns IC_SUCCESS if the hardware uses a FPGA.

	Description
Parameters	int CamIndex, char* pVersion
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera.
char* version: the version of the camera fpga firmware

ROI functions:

int iCube SDK_SetResolution(int nCamIndex, ROI_PROPERTY *property)

Purpose: set the resolution and position of the Region of Interest (ROI).
enable/disable the ROI mode.

This function can be called anytime after “*iCubeSDK_Open*”.

This function has to be called after “*iCubeSDK_SetMode*”, if “*iCubeSDK_SetMode*” is used.

The ROI parameter must be dividable by 4.

	Description
Parameters	int CamIndex, ROI_PROPERTY *property
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
else: error

Parameters:

int nCamIndex: index of camera.
ROI_PROPERTY *property: pointer to the ROI_PROPERTY structure.

int iCube SDK_GetResolution(int nCamIndex, ROI_PROPERTY *property)

Purpose: get the resolution and position of the Region of Interest (ROI).
Get the enable/disable status of the ROI mode.

	Description
Parameters	int CamIndex, ROI_PROPERTY *property
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
else: error

Parameters:

int nCamIndex: index of camera.
ROI_PROPERTY *property: pointer to the ROI_PROPERTY structure.

int iCube SDK_GetResolutionRange(int nCamIndex, ROI_RANGE_PROPERTY *property)

Purpose: get the min/max resolution of the Region of Interest (ROI).

	Description
Parameters	int CamIndex, ROI_PROPERTY *property
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera.
 ROI_RANGE_PROPERTY *property: pointer to the ROI_RANGE_PROPERTY structure.

int iCube SDK_SetResolutionParam(int nCamIndex, HWND ImgHandle, IN BOOL Preview, IN BOOL Callback)

Purpose: starts the image stream of roi 2-4 in multi roi applications.
 Available only for 4133 and 4203 cameras.
 The correct roi ID (2-4) has to set before this function with
 ICubeSDK_SetCamParameter(nCamIndex, REG_ROI_ID, roi ID)
 It is not allowed to leave a gap in the roi settings (you can not set roi 2
 without setting roi 1).
 See 4133_MultiROI SDK Example.
 Linux: this function is not available with the linux api (see MultiROITest
 example).

	Description
Parameters	int CamIndex, ROI_PROPERTY *property
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera.
 HWND ImgHandle: handle to preview window.
 bool Preview: set/unset the use of a preview window. If preview mode is set and
 ImgHandle is NULL, a default preview window will be used.
 bool Callback: set/unset the use of the callback function.

Mode functions:

int iCube SDK_SetMode(int nCamIndex, int nMode)

Purpose: set the basic format (e.g. 640x480). This function has to be called before “iCubeSDK_Start”.

If **iCubeSDK_SetMode** is not used, the camera will start with the default mode (highest resolution).

	Description
Parameters	int CamIndex, int nMode
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera.

int nMode: this value sets the basis format (see table below)

Table 2: resolution

Mode	Resolution
0	320x240 QVGA
1	640x480 VGA
2	752x480 WVGA
3	800x600 SVGA
4	1024x768 XGA
5	1280x1024 SXGA
6	1600x1200 UXGA
7	2048x1536 QXGA
8	2592x1944 QSXGA
9	3840x2748 WQUXGA

int iCube SDK_GetMode(int nCamIndex,int* pMode)

Purpose: get the basic format.

	Description
Parameters	int CamIndex, int nMode
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera.
 int* pMode: current format.

int iCube SDK_GetModeList(int nCamIndex,int* pLength,int* pList)

Purpose: get the possible formats of the camera.

	Description
Parameters	int CamIndex, int* pLength, int* pList
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera.
 int* pLength: number of formats in list.
 int* pList: format list.

Bin Skip functions:

int iCube SDK_SetBinSkip(int nCamIndex,int nParameter,int nMode)

Purpose: set the camera into a skipping or binning mode.
 This function has to be called before "**iCubeSDK_Start**".
 This function has to be called after "**iCubeSDK_SetMode**", if "**iCubeSDK_SetMode**" is used.

The availability of BinSkip modes is camera dependent and can be identified with "**iCubeSDK_GetBinSkipList**".

	Description
Parameters	int CamIndex, int* nParameter, int* nMode
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera.
int nParameter: this value sets how many pixels should be skipped or binned
 (BIN_SKIP_OFF, BIN_SKIP_2ND_PIXEL or BIN_SKIP_4TH_PIXEL)
int nMode: MODE_SKIP: sets skipping mode
 MODE_BIN: set binning mode

int iCube SDK_GetBinSkip(int nCamIndex,int* pParameter,int nMode)

Purpose: get the current skipping or binning mode.

	Description
Parameters	int CamIndex, int* nParameter, int nMode
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
else: error

Parameters:

int nCamIndex: index of camera.
int* pParameter: (BIN_SKIP_OFF,BIN_SKIP_2ND_PIXEL or BIN_SKIP_4TH_PIXEL).
int nMode: MODE_SKIP: get current skipping mode
MODE_BIN: get current binning mode

int iCube SDK_GetBinSkipList(int nCamIndex,int nMode,int* pLength,int* pList)

Purpose: get the possible skipping or binning formats of the camera.

	Description
Parameters	int CamIndex, int nMode, int* pLength, int* pList
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
else: error

Parameters:

int nCamIndex: index of camera.
int nMode: MODE_SKIP: get skipping mode list.
MODE_BIN: get binning mode list.
int* pLength: number of bin/skip modes in list.
int* pList: bin list or skip list.
(BIN_SKIP_OFF,BIN_SKIP_2ND_PIXEL or BIN_SKIP_4TH_PIXEL).

Save functions:

int iCube SDK_SaveToFile(int nCamIndex, char * Name)

Purpose: saves a bitmap, jpg or tiff.

Linux: tiff is not supported by the linux api tiff can be used to save 16bit per color channel pictures (3iCube only). This is only possible, if REG_DATA_TRANSMISSION is set to 1. If the camera is in freerun mode, this function returns when the next available frame has been saved.

If the camera is in trigger mode, this function marks the next available frame to be saved. If the frame arrives (SetTrigger(TRIG_SW_DO)), it will be saved.

	Description
Parameters	int CamIndex, char* Name
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera.

char* Name: the name of the file, the extension sets the type of the saved image (bmp, jpg, tif).

int ICubeSDK_SaveAvi(int nCamIndex, char* Name, int nMode, int nTimeInSecs)

Purpose: saves an avi stream to the hard disk. If an avi stream is saved without the use of an preview window, a window_handle has also being provided with ICubeSDK_Start(). ICubeSDK_GetCamParameter(REG_AVI_STATE) get the avi stream save status (ON | OFF), whereat ON means saving is still in progress and OFF saving is finished.

	Description
Parameters	int nCamIndex, char* Name, int nMode, int nTimeInSecs
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera.
Char* name: path and name of the file, the extension must be avi
int nMode: currently, only AVI_DIB mode is supported
int nTimeInSecs: length of the saved stream in seconds

Trigger functions:

int iCube SDK_SetTrigger(int nCamIndex,int nMode)

Purpose: sets the Trigger mode. The behavior of this function is dependent from the parameter REG_SW_TRIG_MODE.

DELAYED_TRIGGER_RETURN:

SetTrigger(TRIG_SW_DO) returns when the last sensor line is in the camera-usb-chip. If the last sensor line is not arriving in time the return value will be an error.

IMMEDIATE_TRIGGER_RETURN:

SetTrigger(TRIG_SW_DO) returns immediately.

3iCube: IMMEDIATE_TRIGGER_RETURN is default and the only possible mode.

iCube: DELAYED_TRIGGER_RETURN is default, but it is more common to use IMMEDIATE_TRIGGER_RETURN.

	Description
Parameters	int nCamIndex, int nMode
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera.

Int nMode:

TRIG_SW_START:	starts the software trigger mode .
TRIG_SW_DO :	get one software triggered frame.
TRIG_HW_START:	starts the hardware trigger mode.
TRIG_STOP:	stops soft/hardware trigger mode.
TRIG_SW_START_2:	GRR mode (1500CU/BU >= FW x.4.11.x) (1300CU >= FW x.1.12.x)
TRIG_HW_START_2 :	GRR mode (1500CU/BU >= FW x.4.11.x) (1300CU >= FW x.1.12.x)

int iCube SDK_GetTrigger(int nCamIndex,int* nMode)

Purpose: gets the current trigger mode.

	Description
Parameters	int nCamIndex, int* nMode
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera.
 Int* nMode: current trigger mode.

Parameter functions:

int iCube SDK_SetCamParameter(int nCamIndex,int Type,unsigned long Value)

Purpose: set parameter value
 If ICubeSDK_SetCamParameter is used with multi roi applications, the correct roi ID (2-4) has to set before this function with ICubeSDK_SetCamParameter(nCamIndex,REG_ROI_ID, roi ID)
 See 4133_MultiROI SDK Example.

	Description
Parameters	int nCamIndex, int Type, unsigned long Value
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera.
int Type: type of parameter to set. (See Parameter Definitions in API Header)
unsigned long Value: value to set.

int iCube SDK_GetCamParameter(int nCamIndex,int Type,unsigned long *Value)

Purpose: get parameter value.
 If ICubeSDK_GetCamParameter is used with multi roi applications, the correct roi ID (2-4) has to set before this function with ICubeSDK_SetCamParameter(nCamIndex,REG_ROI_ID, roi ID)
 See 4133_MultiROI SDK Example.

	Description
Parameters	int nCamIndex, int Type, unsigned long* Value
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera.
int Type: type of parameter to get. (See Parameter Definitions in API Header)
unsigned long* Value: value to get.

int iCube SDK_GetCamParameterRange(int nCamIndex,int Type,PARAM_PROPERTY*property)

Purpose: get parameter min/max values, default value, auto, onepush and enabled information.

	Description
Parameters	int nCamIndex, int Type, PARAM_PROPERTY *property
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera.
 int Type: type of parameter to get. (See Parameter Definitions in API Header)
 PARAM_PROPERTY *property: pointer to PARAM_PROPERTY structure.

Exposure functions:

This functions can be used alternatively to parameter functions, for manipulating exposure only. Parameter value is in milli-seconds. Be aware that every change of clock and width will lead to different exposure values.

int iCube SDK_SetExposure(IN int nCamIndex, IN float Value)

Purpose: set Exposure time (Input)

	Description
Parameters	int CamIndex, float* Value
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera
 FLOAT Value: exposure time [ms]

int iCube SDK_GetExposure(IN int nCamIndex, OUT float Value)

Purpose: get Exposure time (Output)

	Description
Parameters	int CamIndex, float* Value
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera
 FLOAT Value: exposure time [ms]

int iCube SDK_GetExposureRange(IN int nCamIndex, PARAM_PROPERTY_f *property)

Purpose: get Exposure time Range (Output)

	Description
Parameters	int CamIndex, PARAM_PROPERTY_f *property
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera
 PARAM_PROPERTY_f *property: pointer to the PARAM_PROPERTY_f

Additional Parameter functions:

(Also useable with Exposure functions)

int iCube SDK_GetParamAuto(int nCamIndex,int Type,int* bAuto)

Purpose: check, if the parameter supports auto mode.

	Description
Parameters	int CamIndex, int Type, int* bAuto
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera.
 int Type: type of parameter. (See Parameter Definitions in API Header)
 int* bAuto: 1==supported, 0==unsupported.

int iCube SDK_SetParamAuto(int nCamIndex,int Type,int* bAuto)

Purpose: if auto mode is supported, set/unset auto mode of parameter.

	Description
Parameters	int CamIndex, int Type, int* bAuto
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera.
 int Type: type of parameter to set. (See Parameter Definitions in API Header)
 int* bAuto: 1==set auto, 0==unset auto.

int iCube SDK_SetParamDef(int nCamIndex,int Type)

Purpose: set parameter to default setting

	Description
Parameters	int CamIndex, int Type
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera.
 int Type: type of parameter to set. (See Parameter Definitions in API Header)

int iCube SDK_SetParamOnePush(int nCamIndex,int Type)

Purpose: if one push mode is supported, set/unset one push mode of parameter.
 (currently one push is used for white balance)

	Description
parameters	int CamIndex, int Type
Return value	int stat

Return value:

int stat: IC_SUCCESS: success
 else: error

Parameters:

int nCamIndex: index of camera.
 int Type: type of parameter to set. (See Parameter Definitions in API Header)

Error Codes

Table 3: Error codes

Name	value	Description
IC_SUCCESS	0	no error
IC_ERROR	1	unspecified error
IC_IF_NOT_OPEN	-1	camera-interface is not open
IC_WRONG_PARAM	-2	parameter is out of range
IC_OUT_OF_MEMORY	-3	memory could not be allocated
IC_ALREADY_DONE	-4	e.g. Interface already open
IC_WRONG_CLOCK_VAL	-5	wrong PLL value (more information on operation manual / camera specification)
IC_COM_LIB_INIT	-6	wrong library called
IC_NOT_IF_STARTED	-7	parameter not usable when video stream is started
IC_WRONG_ROI_ID	-8	wrong roi id number
IC_IF_NOT_ENABLED	-9	parameter not enabled
IC_COLOR_CAM_ONLY	-10	parameter is only for color cameras
IC_DRIVER_VERSION	-11	version mismatch (*.sys is not compatible to *.dll)
IC_D3D_INIT	-12	error d3d initialization
IC_BAD_POINTER	-13	bad file pointer
IC_ERROR_FILE_SIZE	-14	wrong file size ()
IC_RECONNECTION_ACTIVE	-15	camera is in reconnection mode (xact recover)
IC_USB_REQUEST_FAIL	-16	usb communication with the camera failed
IC_RESOURCE_IN_USE	-17	parameters is in use by another process
IC_DEVICE_GONE	-18	camera is not accessible
IC_DLL_MISMATCH	-19	dll versions are not compatible
IC_WRONG_FW_VERSION	-20	feature not supported by this firmware version
IC_NO_RGB_CALLBACK	-21	SaveToFile() needs either a preview or rgb callback enabled
IC_NO_USB30_CAMERA	-22	feature not supported by usb2.0 cameras
IC_ERR_FIX_RELATION	-23	serial number of fix registered device does not match
IC_CRC_CONFIG_DATA	-24	crc of config data is corrupt
IC_CONFIG_DATA	-25	config data is corrupt

DirectShow-Interfaces

Supported standard-DirectShow-Interfaces:

IID_IAMVideoProcAmp:

VideoProcAmp_Brightness

VideoProcAmp_Contrast

VideoProcAmp_Gamma

VideoProcAmp_Gain

IID_IAMVideoControl:

VideoControlFlag_FlipHorizontal

VideoControlFlag_FlipVertical

IID_IAMCameraControl:

CameraControl_Exposure

These are the interfaces for controlling camera parameters. Other implemented interfaces (e.g. IAMStreamConfig) are not shown here.

iCube-DirectShow-Interface:

With the ***iCube***-DirectShow-interface, it is possible to control all camera parameters, including Trigger-mode, ROI-mode and Bin/Skip-modes.

(In DirectShow, ROI-mode is, unlike to the SDK, a basic format, like 640x480)

For further documentation see DirectShow-SDK-Files (***iCube***Interface.h,***iCube***Interface.cpp).

iCube-DirectShow-setting

Video Control Parameters

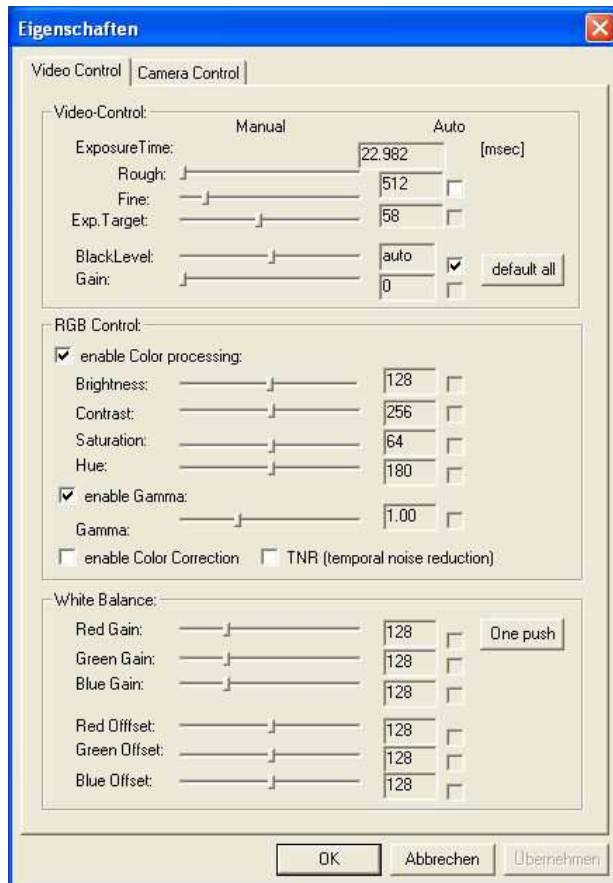


Figure 7: Video control parameters

Table 4: Video Control Parameters

Video Control	
Brightness	Eeprom
Contrast	Eeprom
Gamma	Eeprom
BlackLevel	Eeprom
BlackLevel Auto	Eeprom
Exposure Time	
Exposure Time Auto	Eeprom
Rough	Eeprom
Fine	Eeprom
Exp. Target	Eeprom
Gain	Eeprom
Default	Registry (default parameters)
Color Enhancement	
Color Enhancement enable	Eeprom
Saturation	Eeprom
White Balance	
White Balance	Eeprom
Red	Eeprom
Green	Eeprom
Blue	Eeprom
Red Offset	Eeprom
Green Offset	Eeprom
Blue Offset	Eeprom
One Push	not saved
Color correction enable	Eeprom
TNR enable	Eeprom

Camera Control Parameters

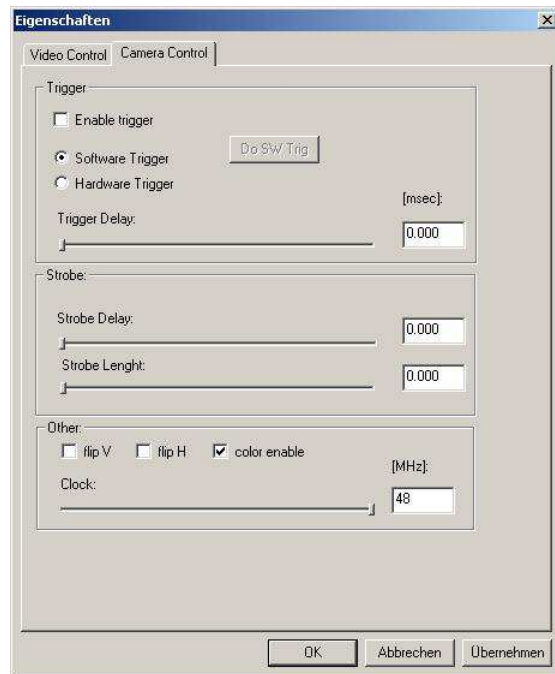


Figure 8: Camera control parameters

Table 5: Camera Control Parameters

Trigger	
Trigger enable	not saved
Software Trigger	not saved
Hardware Trigger	not saved
Trigger Delay	Eeprom
Push SW trigger	not saved
Strobe	
Strobe Delay	Eeprom
Strobe Length	Eeprom
Other	
flip V	Registry
flip H	Registry
color enable	Registry /RAW Data on/off
Clock	Registry

Video Stream Control Parameters

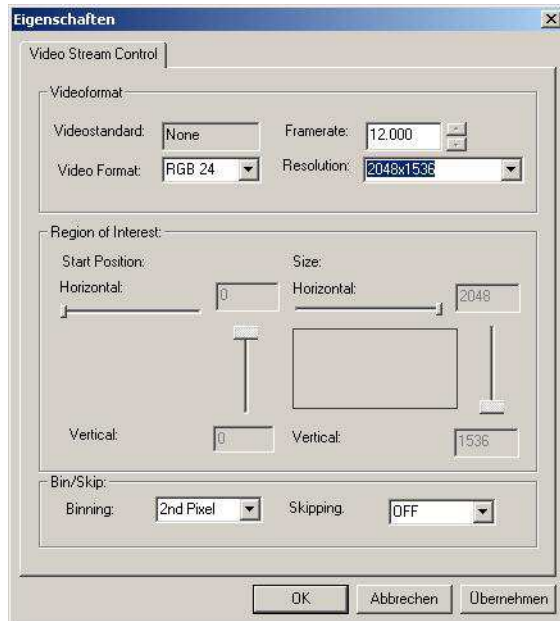


Figure 9: Video stream control parameters

Table 6: Video stream control parameters

Videoformat	
Videostandard	display only
Video Format	Registry
Framerate	calculation (PLL, H, Shutter)
Resolution	Registry
Region of Interest	
Start Position	
Horizontal	Registry
Vertical	Registry
Size	
Horizontal	Registry
Vertical	Registry
Bin/Skip	
Binning	Registry
Skipping	Registry

Technical Support

NET ensures the conformity of its product to be reliable and free from defects during manufacturing by testing all the cameras before release. However, unexpected problems and technical issues may come up due to the complexity of the product.

In case you require technical support, contact the agent near you or contact NET directly at the following locations:

Websites

Europe	www.net-gmbh.com
France	www.net-france-sas.fr
Italy	www.net-italia.it
USA	www.net-usa-inc.com
Asia	www.net-japan.com

Email

Europe	info@net-gmbh.com
France	info@net-france-sas.fr
Italy	info@net-italia.it
USA	info@net-usa-inc.com
Asia	info@net-japan.com

Phone

Europe	+49 8806 92 34-0
Italy	+39 030 5237 163
USA	+1 219 934 9042
Asia	+81 454 781 020

Fax

Europe	+49 8806 92 34-77
Italy	+39 030 5237 163
USA	+1 219 934 9047
Asia	+81 45 476 2423

In case of an RMA, you must first contact NET and obtain an RMA Number before sending the product to us. We are not responsible for any problems caused by not following the RMA procedure.

IMPRINT

NET New Electronic Technology GmbH

Address:

Lerchenberg 7
D-86923 Finning
Germany

Contact:

Phone: +49-88 06-92 34-0
Fax: +49-88 06-92 34-77
www.net-gmbh.com
E-mail: info@net-gmbh.com

VAT- ID: DE 811948278
Register Court: Augsburg HRB 18494

Copyright © 2014 NEW ELECTRONIC TECHNOLOGY GMBH

All rights reserved.