Manual for C2 sample O3D3xx M03023

Version 1.1

Content

1. Primary note

This document is used for the quick bring up of a C2 sample O3D3xx (high resolution camera) of the company ifm syntron gmbh.
For detailed information the contact person is your ifm sales contact.

2. Safety instructions

Please read this manual before activating the unit.
The violation of application advice or technical data could lead to personal or property damage.
These units are development prototypes (Prototype C2 samples). Later releases of the O3D3xx (including its interfaces) could be different to the current version.
For this sample release no safety certification exists nor will be applied for. This will change for later releases.

Additional to this document a deviation list exists which refers to the maturity level of the C2 samples. Please consider it as part of the manual as well.

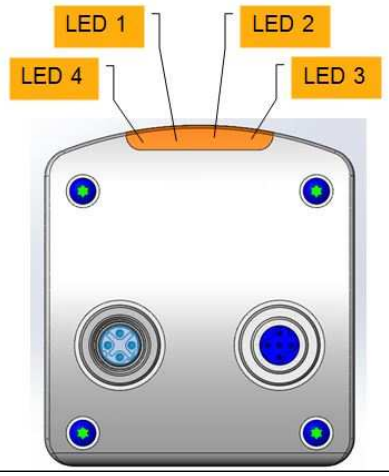
Operating the camera at high frame rates with high exposure times will lead to high temperatures. It is the customer's responsibility to provide adequate cooling.
The maximum recommended settings for medium dynamic range without external cooling are:

- 2ms exposure time at 15Hz frame rate
- 5ms exposure time at 5Hz frame rate

When operating, the camera emits pulsed infrared LED light of nominally 850nm wavelength from the upper and lower front windows. Do not stare into those windows.

3. Control and indicating element

3.1. LED display



Process	LED status
Startup of the unit	Sequence flashing lights (LED 1-3)
Standard operation	LED 1 (green) is active
Ethernet Connection	LED 4 flashing

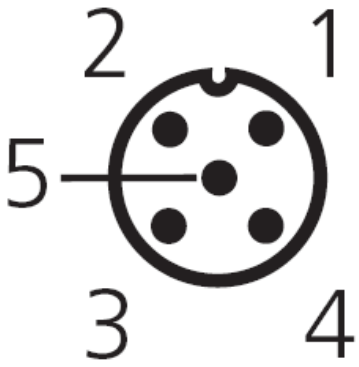
4. Manual for mounting

4.1. Electric connection

ATTENTION! The unit must be installed by a electrically qualified person. The plant must be volt-free before installing the unit.

The unit has to be connected with the crossover-cable to the PC.
Supply the unit with power over the power supply.

4.2. Pin configuration



Power supply (right)	
Pin 1	V++ (18-30V)
Pin 2	n.a.
Pin 3	GND
Pin 4	n.a.
Pin 5	n.a.
Ethernet plug (left)	

5. Network communication between the O3D3XX sample and a PC

Attention: The C” sample O3D3xx and the PC have to be in the same IP-Address area.

5.1. Networkparameter

	Networkaddress	Stationaddress
C2 sample O3D3xx	192.168.0.	69 (Can be changed if necessary)
	=	≠
PC	192.168.0.	e.g. 60

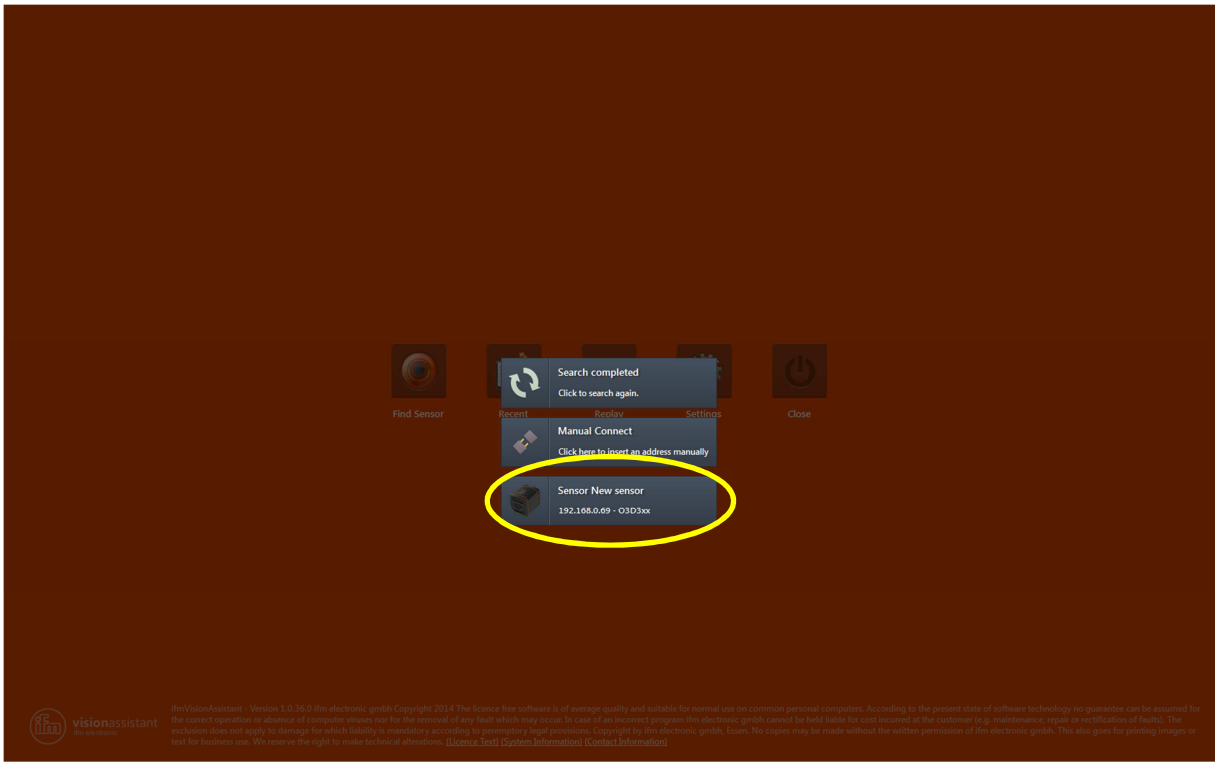
5.2. Connection between software “ifmVisionAssistant” (GUI) and the camera

To connect the camera and the GUI, the software has to establish communication. The following steps will show a first connection and a edit of an application.

5.2.1. “Find Sensor”

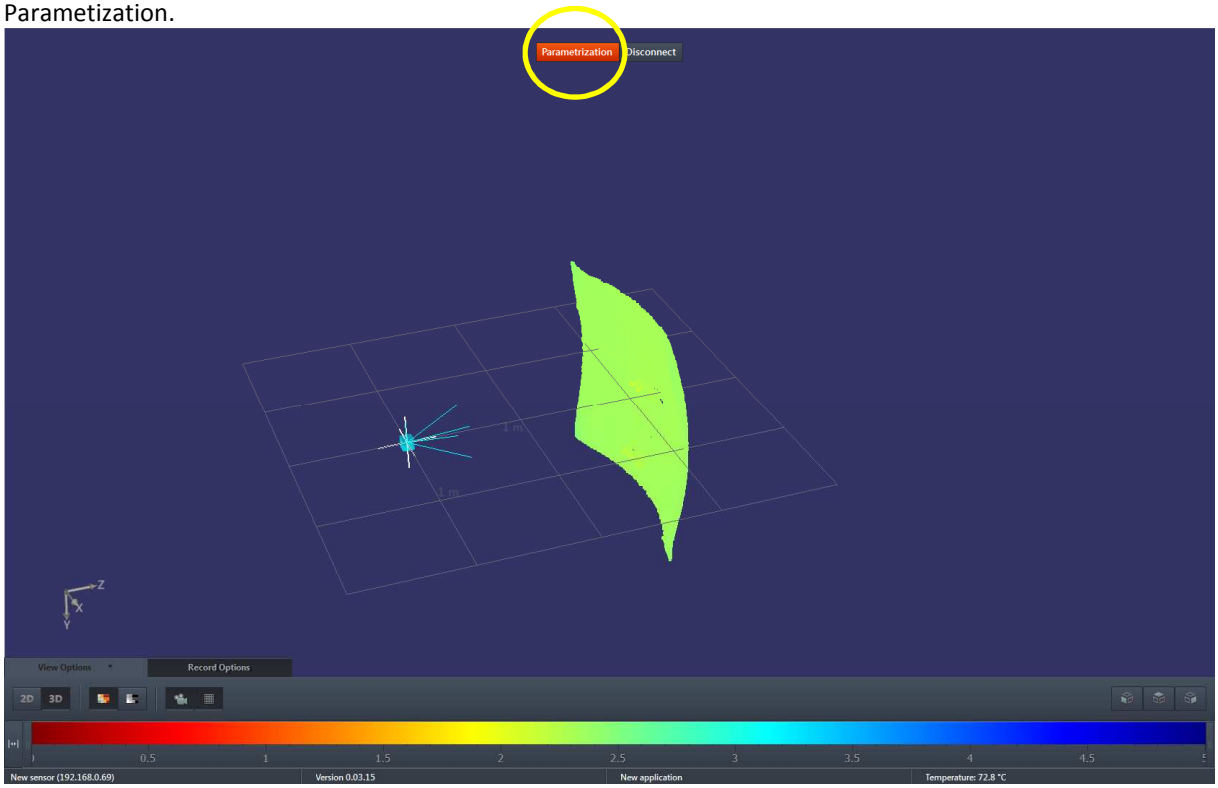
The software will at first search automatically the camera in the network over UDP. If no camera was found the user can type in the IP-Address manually.

A found camera will be listed after the search process. By clicking on it the software will connect to the camera.



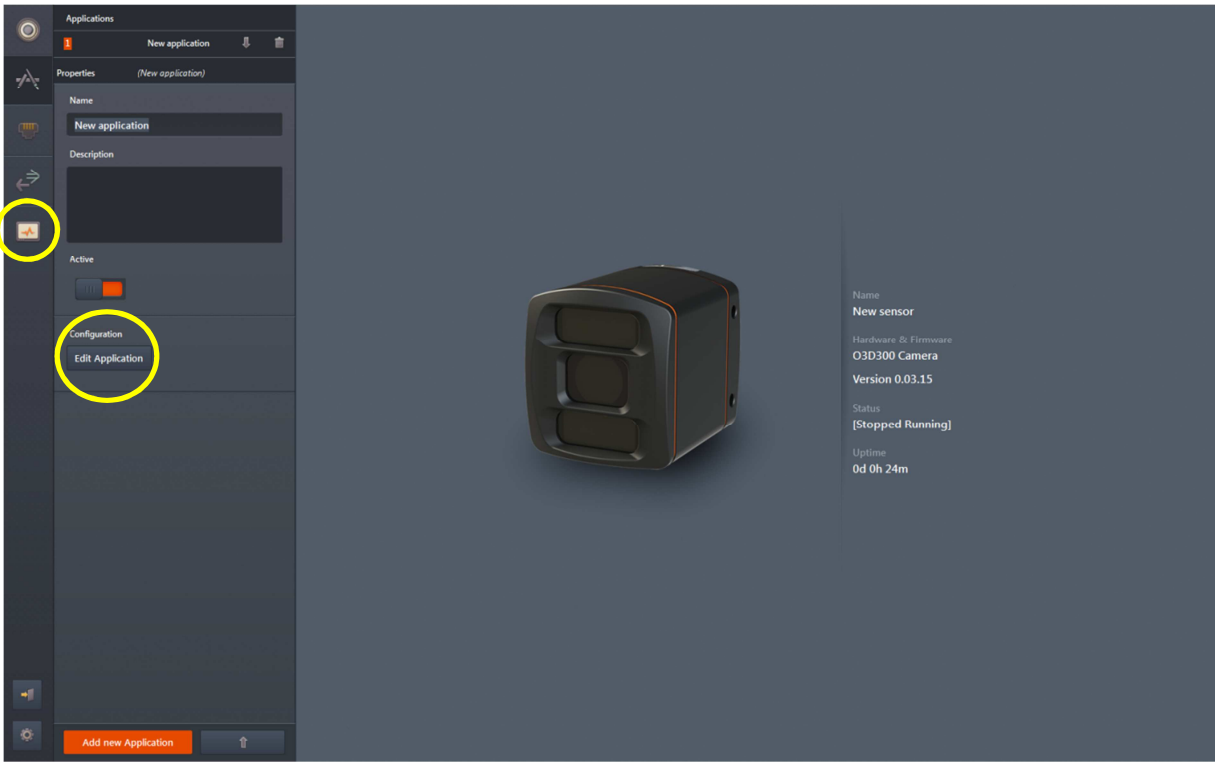
5.2.2. “Monitor mode”

After the connection is established the “ifmVisionAssistant” will switch to the Monitor mode. In this mode the user can see the actual image and activated application. To edit the saved configuration click on Parametization.



5.2.3. “Edit Application”

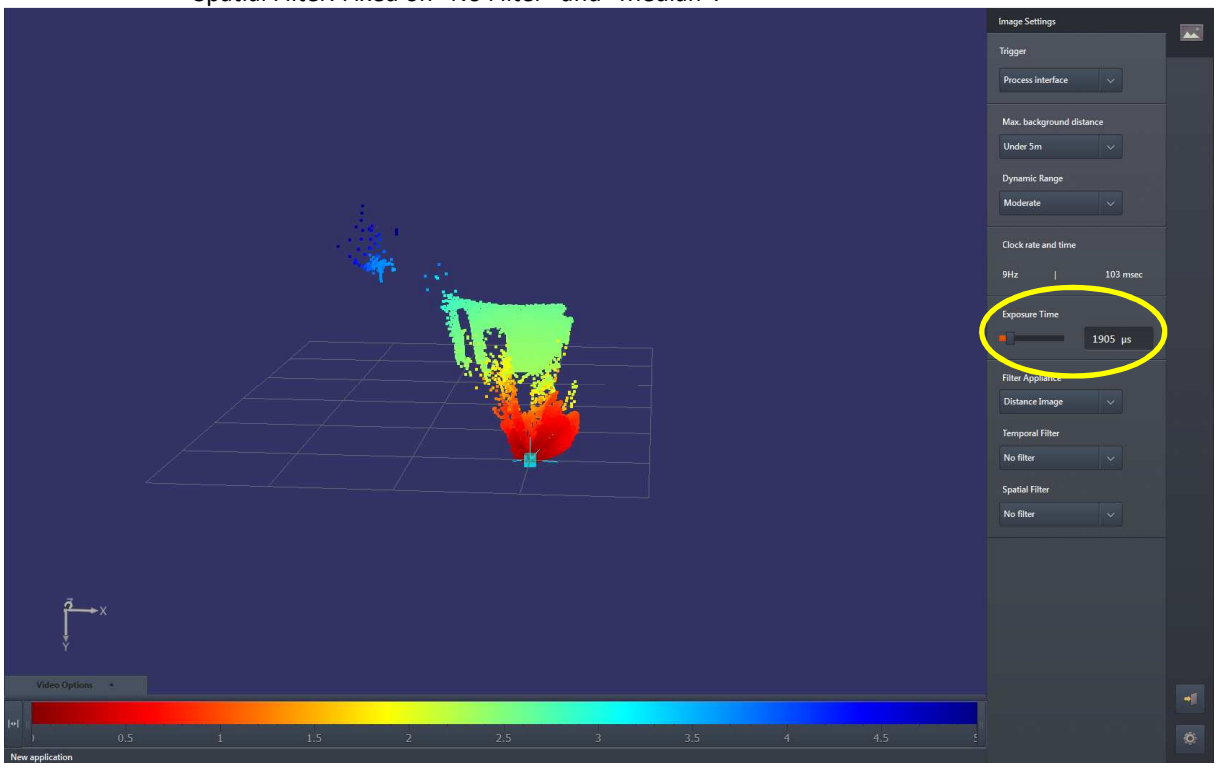
With “Edit Application” the actual marked Application will be configured. To come back to the “Monitor mode” use the button on the left side called “Monitor”



5.2.4. “Configurate the application”

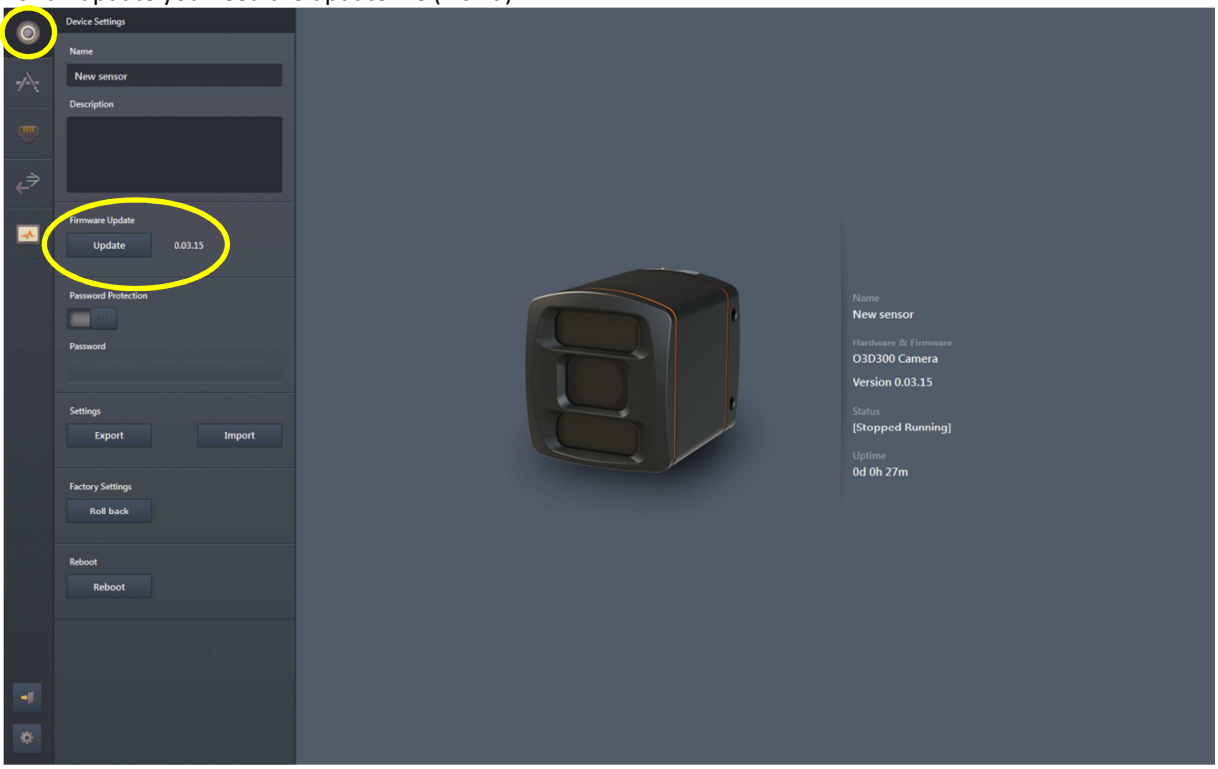
In this mode the application can be changed.

- Trigger: It is possible to switch the triggertype from ”Process Interface” (Software) to “Continuous”.
- Dynamic Range: Switch between:
  - Low: Single exposure time.
  - Moderate: Dual exposure time. The ratio of the exposure times is fixed at 40:1. The setting controls the longer exposure time.
  - High: Triple exposure time. In the “High” mode it is not possible to change the exposure time.
- Max. Framerate: Please see 2. Safety instructions for operating the camera at high framerates. The actual displayed framerate is not necessarily the real framerate.
- Exposure Time: Changes the illumination time/power. To get a better image.
- Filter Appliance: Fixed on Distance Image
- Temporal Filter: Will be supported in future versions
- Spatial Filter: Fixed on “No Filter” and “Median”.

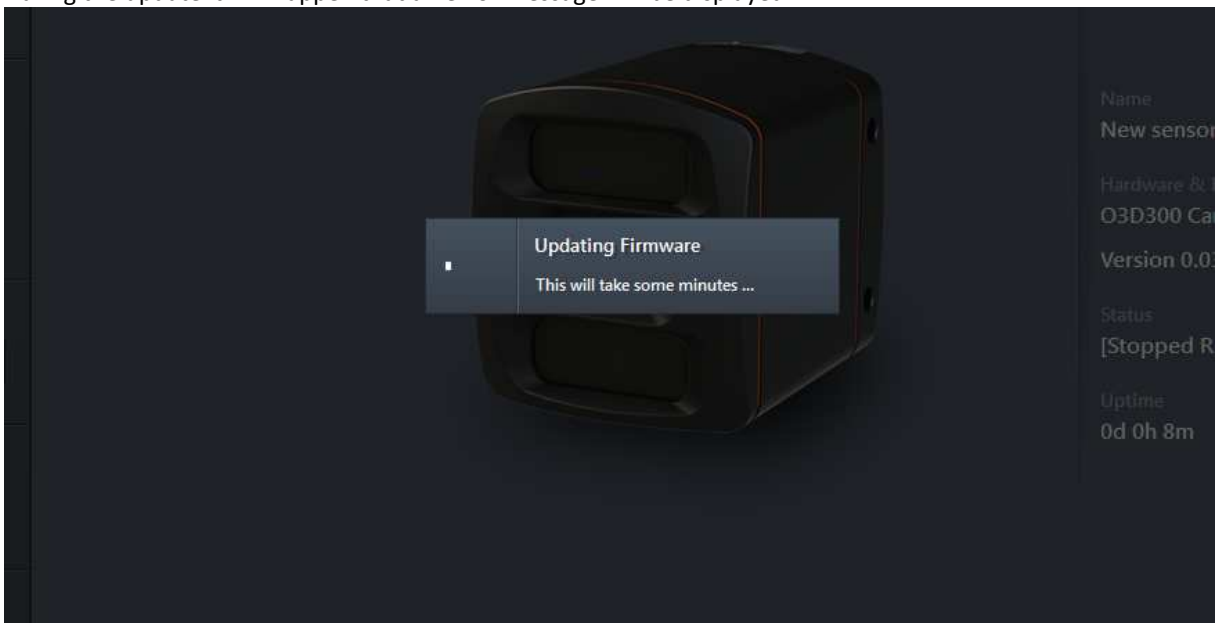


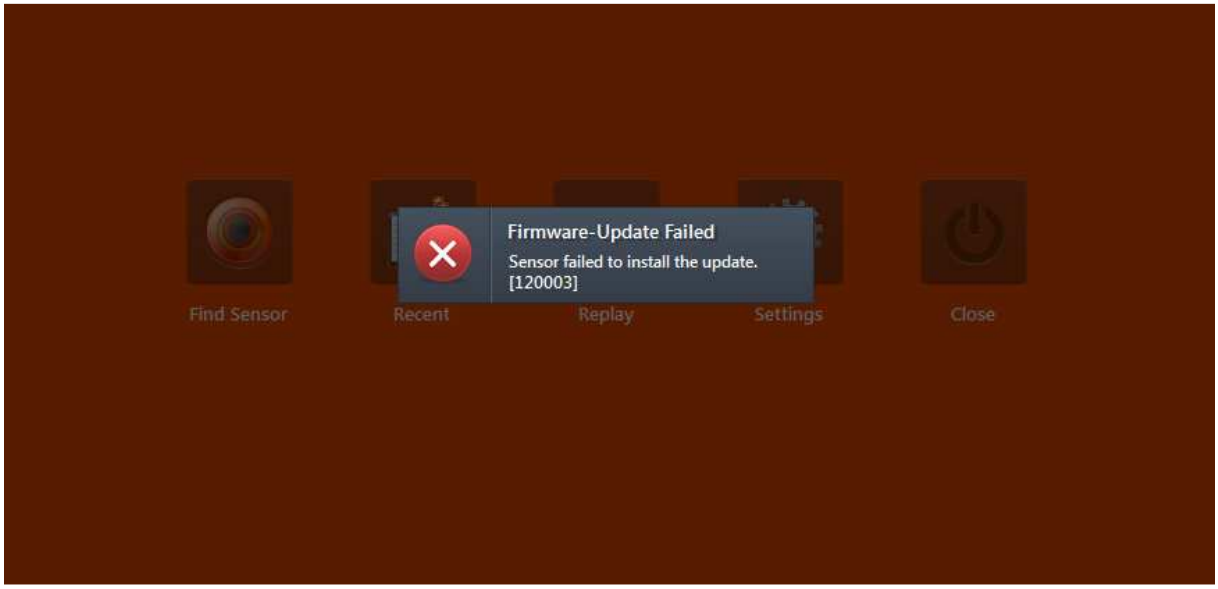
5.2.5. Device Settings

It is possible to upgrade the camera with an update file over the button “Update”.
For an update you need the update file (*.swu).



During the update it will happen that an error message will be displayed.



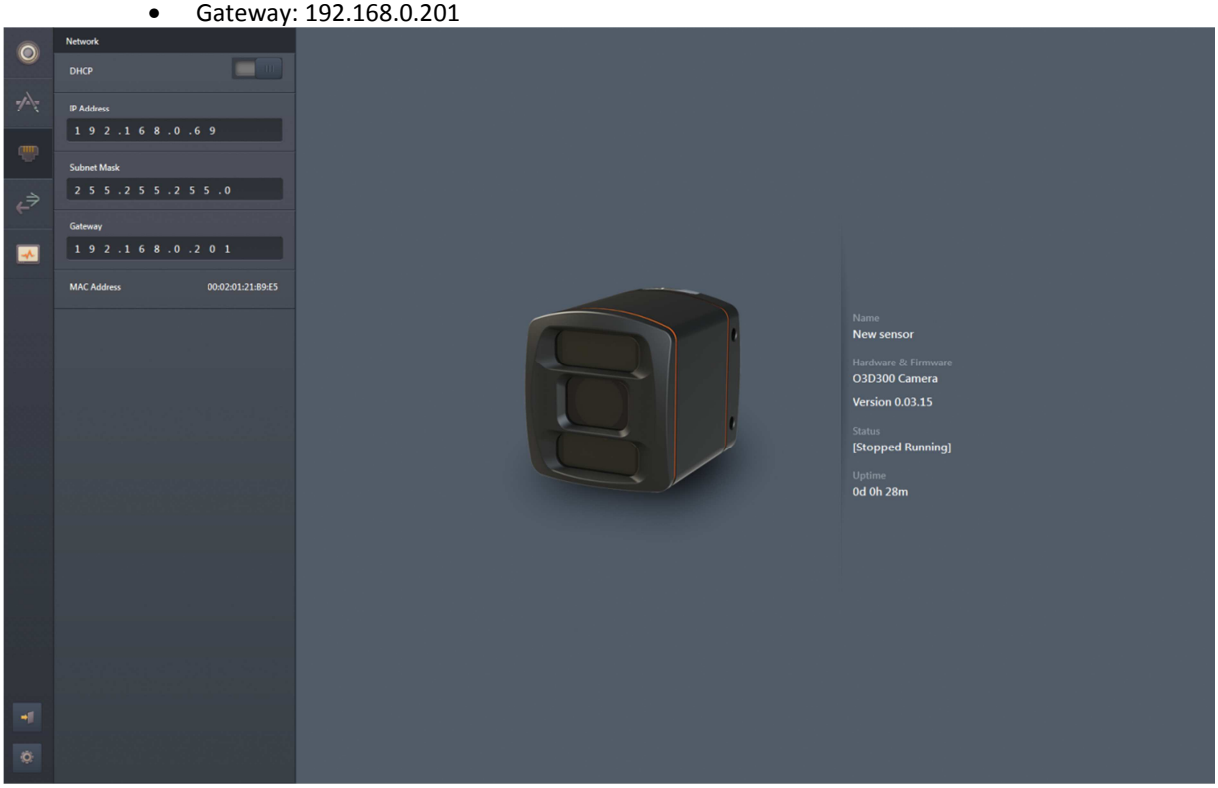


Wait 2 min. after the error message until you try to connect the GUI to the camera. Don't install the firmware again, boot the camera into the recovery mode. This will start the camera with the new firmware version.

5.2.6. Network settings

It is possible to change the Networkparameters.
Default parameter:

- IP Address: 192.168.0.69
- Subnet Mask: 255.255.255.0
- Gateway: 192.168.0.201



6. XML-RPC Interface

The O3D3xx can be configured by the XML-RPC interface. Note: general information about XML-RPC is found on the website http://xmlrpc.scripting.com/spec

To send a command over the XML-RPC interface the command has to be on a special layout. In this command, linefeeds and carriage returns are essential.

Several commands will use different URL in the XML-RPC header.

6.1. Sample XML-RPC command

All following XML-RPC commands will have this type of layout:

```
POST /RPC2 HTTP/1.0
User-Agent: Frontier/5.1.2 (WinNT)
Host: betty.userland.com
Content-Type: text/xml
Content-length: 181

<?xml version="1.0"?>

<methodCall>

<methodName>examples.getStateName</methodName>

<params>

<param>

<value><i4>41</i4></value>

</param>

</params>

</methodCall
```

Following example contains one command of the O3D3xx C2-sample:

```
POST /api/rpc/v1/com.ifm.efector/
User-Agent: Frontier/5.1.2 (WinNT)
Host: 192.168.0.69
Content-Type: text/xml
Content-length: 94


<?xml version="1.0"?>

<methodCall>

<methodName>getParameter</methodName>

</methodCall>
```

6.2. Main-Object:

Object-URI: /api/rpc/v1/com.ifm.efector/

This is the main-object of RPC, it allows to access some basic information and contains methods for activating edit-mode.

6.3. SessionObject

Object-URI e.g.: /api/rpc/v1/com.ifm.efector/session_d21c80db5bc1069932fbb9a3bd841d0b/

6.4. EditMode-Object

Object-URI e.g.: /api/rpc/v1/com.ifm.efector/session_d21c80db5bc1069932fbb9a3bd841d0b/edit/

This object is only available if the device is in edit-OperatingMode. Index of Applications must be between 1 and 32. The device must only support 32 applications and the indexes must start at 1.

6.5. DeviceConfig-Object

Object-URI e.g.: /api/rpc/v1/com.ifm.efector/session_d21c80db5bc1069932fbb9a3bd841d0b/edit/device/

6.6. Device/NetworkConfig-Object

Object-URI e.g.:
/api/rpc/v1/com.ifm.efector/session_d21c80db5bc1069932fbb9a3bd841d0b/edit/device/network/

6.7. ApplicationConfig-Object (editable application)

Object-URI e.g.: /api/rpc/v1/com.ifm.efector/session_d21c80db5bc1069932fbb9a3bd841d0b/edit/application/

6.8. App./ImagerConfig-Object (O3D3xx)

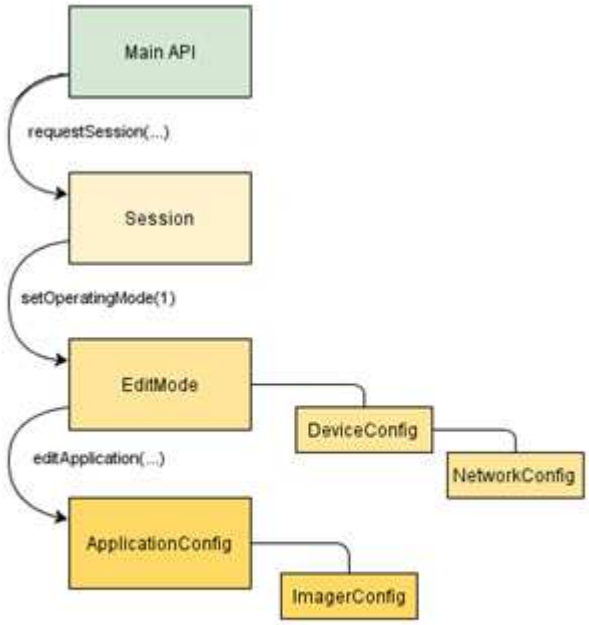
Object-URI e.g.:
/api/rpc/v1/com.ifm.efector/session_d21c80db5bc1069932fbb9a3bd841d0b/edit/application/imager_001/

As there is only one imger-config on O3D3xx, the ID must be fixed to "001". Data of this object is persistent saved, when calling "save" on ApplicationConfig-object. The imager config RPC-object has multiple sub-types. Only parameters relevant for a specific type are available while it is active. They are based on frequency (extending the distance) and integration-intervals (extending the measure-details).

type-names, based on GUI-draft (under 5 meter->single Frequency, up to 30 meter-> double Freq., more than 30 Meter -> 3Freq.):

under5m_low
under5m_moderate
under5m_high

6.9. Visual description of the XML-RPC objects



Note: It could be necessary to send heartbeats so that there will be no session-timeout.

7. Process Interface

7.1. Sending commands

For sending commands over the process interface, the commands have to be send with a special protocol and as strings. This protocol conforms to the version 3 of the O2V/O2D products.

Structure of the protocol:

<Ticket><length>CR LF <Ticket><contents>CR LF

Abbreviation	Description	ASCII code (dec)
CR	Carriage Return	13
LF	Linefeed	10
< >	Marking of a placeholder (e.g. <code> is a placeholder for code)	
[ ]	Optional argument (possible but not required)	

<contents> is the command to the device (e.g. trigger the unit).

<ticket> is a character string of 4 digits 0-9, to be interpreted as decimal number. If a message with a specific ticket is sent to the device, its reply will contain the same ticket

<length> is a character string beginning with the letter 'L' followed by 9 digits to be interpreted as decimal number. This figure indicates the length of the following data (<ticket><contents>CR LF) in bytes.

Version	input format	output format
V1	<Content>CR LF	as input
V2	<Ticket><Content>CR LF	as input
V3	<Ticket><Length>CR+LF<Ticket><Content>CR LF	as input
V4	<Content>CR LF	<length>CR LF<Content>CR LF

The default protocolversion is „V3“. Do not use the other versions for the C2 sample.

7.2. Receiving images

For receiving the image data there has to be a TCP/IP socket communication established. This communication works on the Port 50010. After opening the socket communication the O3D3XX sample will automatically (if the unit is in free run mode) send the image data through this socket to the TCP/IP client (PC).

PCIC output per frame. The following data shall be submitted in this sequence:

Component	Content
Ticket and length information	Please see chapter 0 7.1. Sending commands
Ticket	„0000“
Start sequence	String "star" (4 bytes)
normalized amplitude image output format: 16 Bit Unsigned Integer.	1 image
distance image output format: 16 Bit Unsigned Integer. Unit is mm.	1 image
x-image output format: 16 Bit Signed Integer. Unit ist mm.	1 image
y-image output format: 16 Bit Signed Integer. Unit ist mm.	1 image
z-image output format: 16 Bit Signed Integer. Unit ist mm.	1 image
Confidence image output format: 8 Bit Unsigned Integer	1 image
Diagnostic data	
Stop sequence	String "stop" (4 bytes)
Ticket finish	<CR><LF>

Diagnostic data output has the following structure:

Illumination Temperature	32-bit signed int	4 bytes
Frontend Temperature 1	32-bit signed int	4 bytes
Frontend Temperature 2	32-bit signed int	4 bytes
i.mx6 Temperature	32-bit signed int	4 bytes
processing time in ms	32-bit unsigned integer	4 bytes

All temperature values have the unit 0.1 °C, invalid temperatures have the value 0x7FFF (32767).

Illumination Temperature and Frontend Temperature 2 are not measured in C2 sample. Therefor these temperatures have the value 0x7FFF (32767).

7.3. Image data

For every image there will be a separate chunk. The chunk is a part of the response frame data from the process interface. The image data layout of the response is separated to these points:

The header of each chunk contains different kind of information. This information is separated into Bytes. The information contains e.g. the kind of image which will be in the “PIXEL_DATA” and the size of the chunk.

Chunk type:

Offset	Name	Description	Size [Byte]
0x0000	CHUNK_TYPE	Defines the type of the chunk. For each distinct chunk type a own type has to be defined.	4
0x0004	CHUNK_SIZE	Size of the whole image chunck in bytes. After this count of bytes the next chunk starts.	4
0x0008	HEADER_SIZE	Number of bytes starting from 0x0000 until PIXEL_DATA	4
0x000C	HEADER_VERSION	Version number of the header	4
0x0010	IMAGE_WIDTH	Image width in pixel	4
0x0014	IMAGE_HEIGTH	Image height in pixel	4
0x0018	PIXEL_FORMAT	Pixel-Format	4
0x001C	TIME_STAMP	Timestamp in uS	4
0x0020	FRAME_COUNT	Frame count according to algorithm output	4
0x0024	PIXEL_DATA	The pixel data in the given type and dimension of the image. Padded to 4-Byte boundary.	

Pixel format:

Constant	Value	Description
FORMAT_8U	0	**8**bit unsigned integer
FORMAT_8S	1	**8**bit signed integer
FORMAT_16U	2	**16**bit unsigned integer
FORMAT_16S	3	**16**bit signed integer
FORMAT_32U	4	**32**bit unsigned integer
FORMAT_32S	5	**32**bit signed integer
FORMAT_32F	6	**32**bit floating point number
FORMAT_64U	7	**64**bit unsigned integer
FORMAT_64F	8	64bit floating point number

7.4. Additional information for image data

Image type	Unit	Invalid pixel	Data interpretation
Distance	Milimeter	0.0 mm	Each pixel of the distance matrix denotes the distance between the optical center of the camera and appropriate object in the scene
Amplitude	Digits	0.0	Each pixel of the amplitude matrix denotes the amount of light which is reflected by the appropriate object. Due to the double exposure time the lack of normalization may lead to inhomogeneous amplitude image impression
Intensity	Digits	0.0	Each pixel of the intensity matrix denotes the sum of light reflected by the appropriate object. Due to the double exposure time the lack of normalization may lead to inhomogeneous intensity image impression
Confidence	Digits		See table bellow
X,Y,Z	Milimeter		The X,Y,Z matrices denote the appropriate component of the Cartesian coordinate of a PMD 3D measurement. +X: to the right +Y: down +Z: forward

Further information for the Confidence image

Bit	Value	Description
0	1 = pixel invalid	**Pixel invalid** (NV) The pixel is invalid. To determine, whether a pixel is valid or not only this bit needs to be checked. For a reason why the bit is invalid the other confidence bits may be checked.
1	1 = pixel saturated	**Pixel is saturated** (SA) Contributes to pixel validity: yes
2	1 = bad A-B symmetry	**A-B pixel symmetry** (SY) The A-B symmetry value of the four phase measurements is above threshold. Remark: This symmetry value is used to detect motion artefacts. Noise (e.g. due to strong ambient light or very short integration times) or PMD interference may also contribute. Contributes to pixel validity: yes
3	1 = amplitude below minimum amplitude threshold	**Amplitude limits** (AM) The amplitude value is below minimum amplitude threshold. Contributes to pixel validity: yes
4+5	Bit 5, Bit 4 0 0 unused 0 1 shortest exposure time (only used in 3 exposure mode) 1 0 middle exposure time in 3 exposure mode, short exposure in double exposure mode 1 1 longest exposure time (always 1 in single exposure mode)	**Exposure time indicator** The two bits indicate, which exposure time was used in a multiple exposure measurement. Contributes to pixel validity: no
6	1= motion artefact compensated	**Not implemented for C2 sample O3D3xx**
7	1 = pixel suspect/defect	**Suspect pixel** (SU) This pixel has been marked as "suspect" or "defect" and values have been replaced by interpolated values from the surrounding. Contributes to pixel validity: no

8. Set up PMDSDK2

8.1. Purpose

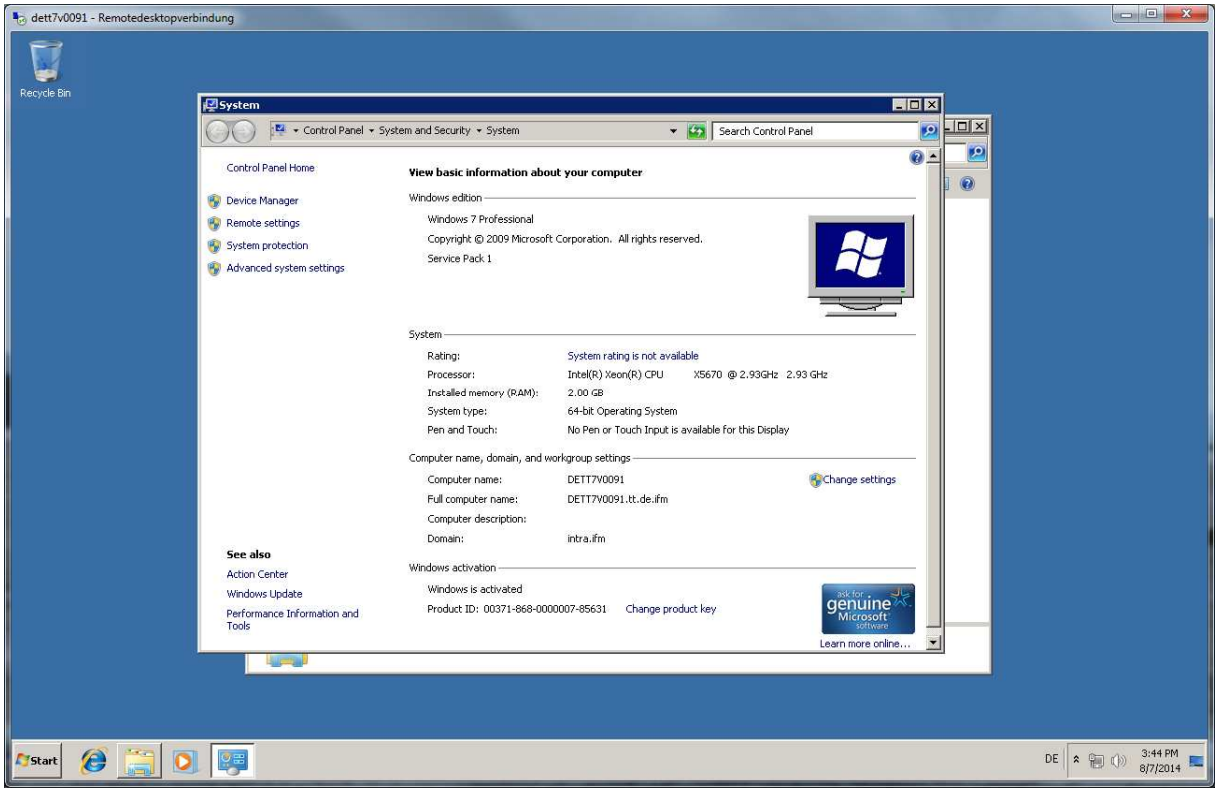
The purpose of the PMDSDK2 is to provide an environment for a code example of the interface of the O3D3xx. To visualize results the PC-Software "LightVis" can be used. "LightVis" originally has been developed by **pmd**technologies. The PMDSDK2 provides sample code to demonstrate the O3D interface. However, ifm electronic gmbh excludes liability for demonstrational sample codes.

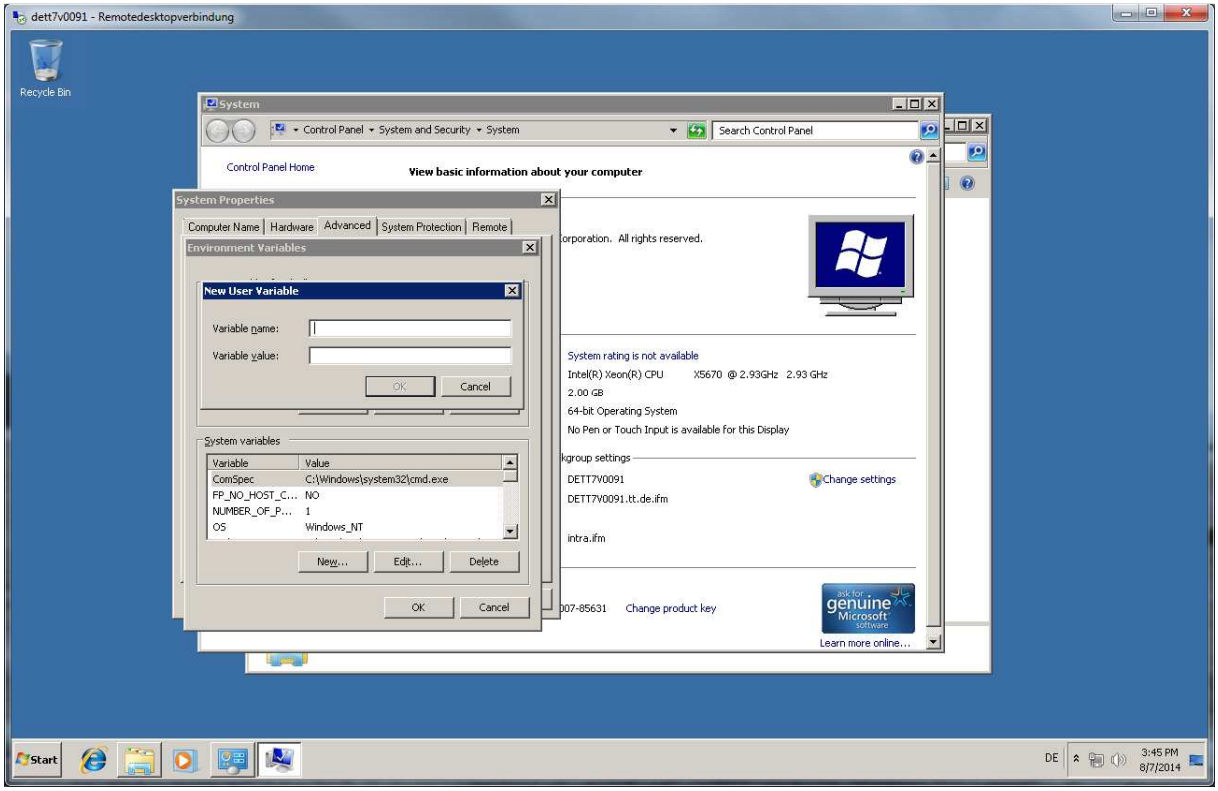
All sample codes are provided in the PMDSDK2. To set up the SDK follow these steps:

The following software has to be installed on the PC before you can build the library
1) CMake (version 2.8.9 or higher from http://cmake.org/cmake/resources/software.html)
When the software asks you for the path use all users

2)Boost (version 1.54.0 or higher from http://www.boost.org/users/download/)

- after download, go to the boost root folder over with the console
- start "bootstrap"
- start "b2.exe --with-regex --with-date_time --with-system --with-thread" (copy/paste this command, files are found at "stage/lib")
- add system variable to boost root directory: e.g.: BOOST_ROOT = "D:\boost_1_54_0\" (see pictures bellow)




dett7v0091 - Remotedesktopverbindung
Recycle Bin
System
Control Panel
System and Security
System
Search Control Panel
Control Panel Home
View basic information about your computer
System Properties
Computer Name
Hardware
Advanced
System Protection
Remote
Environment Variables
New User Variable
Variable name:
Variable value:
OK
Cancel
System variables
Variable
Value
ComSpec
C:\Windows\system32\cmd.exe
FP_NO_HOST_C...
NO
NUMBER_OF_P...
1
OS
Windows_NT
New...
Edit...
Delete
OK
Cancel
orporation. All rights reserved.
System rating is not available
Intel(R) Xeon(R) CPU X5670 @ 2.93GHz 2.93 GHz
2.00 GB
64-bit Operating System
No Pen or Touch Input is available for this Display
kgroup settings
DETT7V0091
Change settings
DETT7V0091.tt.de.ifm
intra.ifm
007-85631
Change product key
ask for genuine Microsoft software
Learn more online...
Start
DE
3:45 PM
8/7/2014

The following software is needed to build the Doxygen documentation -
1) Doxygen (version 1.8.7 or higher from http://www.stack.nl/~dimitri/doxygen/download.html)

2) Graphviz (version 2.38 or higher from http://www.graphviz.org/Download.php)

3) Miktex (version 2.9 or higher from http://miktex.org/download)
this is required only if the documentation is as PDF required.

NOTE: We have built and tested the code with the versions mentioned above. Higher versions should also work seamlessly, but this is not guranteed.

Download the source code from ifm official site. Or use the provided ones.
The source code folder shall contain code related files for "O3D3xx plugin development" project.
It mainly consists of O3D3xx source and processing plugin, along with dependent code and third party tools, test suites etc.

The folder structure looks like this:
Base Folder

- “processing_plugin” (the files needed to build the processing plugin)
- “source_plugin” (the files needed to build the source plugin)
- “samples” (all the sample codes)
- “tests” (all the unit test codes)
- “third_party” (all the third party code)

8.2. Process for building the Source Code:

- Open the CMake GUI (Start Menu -> All Programs -> CMake 2.8 -> CMake (cmake-gui)).
- Type into "Where is the Source code", the point to the location of the source code (only the path to the folder /code).
- In the "Where to build the binaries" area, point to the location where you want to build the binaries.
- It is recommended that a separate folder called as "build" (this name can be changed)be created in a separate location on the PC for this purpose.
- Click on "Configure". Select the Visual Studio (VS) compiler version that you have installed on your PC. Please note the following mapping to be used:

  VS 2008 --> Visual Studio 9
  VS 2010 --> Visual Studio 10
  VS 2012 --> Visual Studio 11
  VS 2013 --> Visual Studio 12

There are a couple of options in CMake which can be selected if required:

O3D3XX_SAMPLE_CODES_BUILD_ALL	This option if selected (default ON),generates the sample code project
O3D3XX_UNIT_TESTS_BUILD_ALL	This option if selected (default ON), generates the unit test and Gtest projects.
O3D3XX_DOXYGEN_DOCUMENTATION_GENERATE	If selected this option (default OFF), generates the doxygen documentation. For this option, please have a look at the software required for doxygen described in the section above.
O3D3XX_DOXYGEN_DOCUMENTATION_CHM	If selected this option (default ON), generates the doxygen documentation. For this option, please have a look at the software required for doxygen described in the section above.
O3D3XX_DOXYGEN_DOCUMENTATION_PDF	If selected this option (default OFF), generates the doxygen documentation. For this option, please have a look at the software required for doxygen described in the section above.

- Click on "Generate".
- This will build a VS solution in the "build" folder.
- Open the VS Solution in VS.
- In VS is now a list “pf projects” which are created as –

  ALL_BUILD - CMake project; do not touch
  ZERO_CHECK - CMake project; do not touch
  gtest - part of the gtest library
  gtest_main - part of the gtest library
  libxmlrpc - part of the open source XML-RPC library
  libxmlrpc_client - part of the open source XML-RPC library
  libxmlrpc_util - part of the open source XML-RPC library
  libxmlrpc_xmlparse - part of the open source XML-RPC library
  libxmlrpc_xmltok - part of the open source XML-RPC library
  O3D3xxCamera - Source plugin for the O3D3xx Camera
  O3D3xxProc - Processing plugin for the O3D3xx Camera
  SampleO3D3xxCamera - Sample code for using the O3D3xxCamera Class directly

- You are now ready to build all the projects.

- After successful compilation of code,
  i. If the doxygen documentation generation is selected, the technical reference document named "O3D3xx_Plugin.chm" and / or "O3D3xx_Plugin.pdf" shall be available at, "build/docs".
  ii. The generated binaries shall be available at, "build/bin/<configuration_folder>".
  <configuration_folder> denotes the configuration by which you build the solution eg: Debug, Release etc.

NOTE:
For compiling boost library, make sure that your environment variables contain the path for following binaries.
i."cl.exe" - Generally found at "$(WIN_DIR)\Program Files\Microsoft Visual Studio 9.0\VC\bin\" folder.
ii."mspdb80.dll" - Generally found at "$(WIN_DIR)\Program Files\Common Files\microsoft shared\VSA\9.0\VsaEnv" folder.

8.3. Sample C code - shows usage of class functions via C code.

User application can directly access O3D3xx plugin class functions to connect/communicate to the camera. A sample C/C++ code is given to demonstrate the usage of O3D3xx plugin class functions.

8.3.1. Steps to use sample code:

- If your camera ip is 192.168.0.69, xmlrpc port is 80 and pcic port is 50010, then directly go to step 4.
- In solution window, open file "o3d3xx_camera_sample_usage_code.cpp" listed under "SampleO3D3xxCamera" project.
- Steps to change default camera ip address, xmlrpc port number and pcic port number
- If your camera ip address is not 192.168.0.69, then find following line of code:
  #define IP_ADDRESS_OF_CAMERA "192.168.0.69"
  then replace “192.168.0.69” with your camera's IP-Address.
- If your xmlrpc port number is not 80, then find following line of code:
  #define XMLRPC_PORT 80
  then replace "80" with your xmlrpc port number.
- If your pcic port number is not 50010, then find following line of code,
  #define PCIC_PORT 50010
  then replace "50010" with your pcic port number. After this save the solution.
- Compile the solution.
- After successfull compilation of project, the executable for sample C code shall be generated by name "SampleO3D3xxCamera.exe" at "build/bin/<configuration_folder>".<configuration_folder> denotes the configuration by which you build the solution. It could be Debug, Release etc.
- Double click at the executable. It shall demonstrate the configuration of the camera and receiving frames and image data.

9. XML-RPC command references

9.1. Main-Object

"getParameter"

Method Name	getParameter
Description	Getter for the device-global parameters
Input Parameters	Name of a device-parameter :string
Output Parameters	Value of the requested parameter :string

"getAllParameters"

Method Name	getAllParameters
Description	Getter for the parameters described here: This is an additional getter outside of Edit-Sessions, so it is possible to read device informations without login.
Input Parameters	none
Output Parameters	1. Struct (name contains parameter-name, value the stringified parameter-value)

"getSWVersion"

Method Name	getSWVersion
Description	Returns version-information of all software componenents
Input Parameters	none
Output Parameters	1. Struct of strings (e.g. { "IFM_Software": "0.01.07", "Frontend": "01.05.02", ... }) *mandatory keys: "IFM_Software" "Linux" "Main_Application" "Diagnostic_Controller" "Algorithm_Version" "Calibration_Version" "Calibration_Device"

"getHWInfo"

Method Name	getHWInfo
Description	Returns hardware-information of all components
Input Parameters	none
Output Parameters	Struct of strings (e.g. { "MACAddress": "00:02:01:40:06:C9", "Frontend": "#!01_F340_001_...", ... }) *mandatory keys: "MACAddress" "Connector" "Diagnose" "Frontend" "Illumination" "Mainboard"

"getApplicationList"

Method Name	getApplicationList
Description	Delivers basic information of all Application stored on the device. This should be available before password-session, so the CombiGUI could display Sensor-screen before login.
Input Parameters	none
Output Parameters	1. Array of structs (Index: int, Id: int, Name: string, Description: string)

"requestSession"

Method Name	requestSession
Description	Request a session-object for access to the configuration and for changing device operating-mode. This should block parallel editing and allows to put editing behind password. The ID could optionally be defined from the external system, but it must be the defined format (32char "hex"). If it is called with only one parameter, the device will generate a SessionID. The session will start with a default timeout("SessionTimeout" device-parameter), the timeout can be extended by calling "heartbeat". The device will stay in RUN-mode. If password is disabled on the device, the value given as password-parameter is ignored.
Input Parameters	1. Password: string 2. SessionID: string (optional)
Output Parameters	1. SessionID: string

"reboot"

Method Name	reboot
Description	Reboot system, parameter defines which mode/system will be booted
Input Parameters	1. type of system that should be booted after shutdown :int 0: productive-mode 1: recovery-mode
Output Parameters	1. empty-string (compatibility for classic XmlRPC-Client)

"systemCommand"

Method Name	systemCommand
Description	Performs a generic command on the device.
Input Parameters	1. Command :string 2. Parameter :string
Output Parameters	1. Output :string

9.2. Session-Object

"heartbeat"

Method Name	heartbeat
Description	Extend the live time of edit-session If the given value is outside the range of "SessionTimeout", the saved default timeout will be used.
Input Parameters	1. requested timeout-interval till next heartbeat, in seconds :int
Output Parameters	1. the used timeout-interval, in seconds :int

"cancelSession

Method Name	cancelSession
Description	Explicit stopping this session If an application is still in edit-mode, it will implicit do the same as "stopEditingApplication".
Input Parameters	none
Output Parameters	1. empty-string (compatibility for classic XmlRPC-Client)

"exportConfig"

Method Name	exportConfig
Description	exports the whole configuration of the sensor-device
Input Parameters	none
Output Parameters	1. configuration as one data-blob :binary/base64

"importConfig"

Method Name	importConfig
Description	import whole configuration, with the option to skip specific parts
Input Parameters	1. configuration as one data-blob :binary/base64 2. flags which parts should be loaded: 0x0001: Include Globale-Configuration (Name, Description, Location, ...) 0x0002: Include Network-Configuration (IP, DHCP, ...) 0x0010: Include All Application-Configurations
Output Parameters	1. empty-string (compatibility for classic XmlRPC-Client)

"exportApplication"

Method Name	exportApplication
Description	exports one application-config
Input Parameters	1. Application Index
Output Parameters	1. application-config as one data-blob :binary/base64

"importApplication"

Method Name	importApplication
Description	imports an application-config and creates a new application with it. The name of the application should be based on the one stored in the exported-config. If the name should be unique, the sensor must generate an suffix in case of a naming conflict. The device will give a new ID, if there is an ID inside the config-data it must be ignored. The device will put the new application on the first free Index.
Input Parameters	1. application-config as one-data-blob: binary/base64
Output Parameters	1. Index of new application

"setOperatingMode"

Method Name	setOperatingMode
Description	Changes the operation mode of the device. Setting this to "edit" will enable the "EditMode"-object on RPC.
Input Parameters	1. mode :int 0: run mode 1: edit mode
Output Parameters	1. empty-string (compatibility for classic XmlRPC-Client)

9.3. EditMode-Object

"factoryReset"

Method Name	factoryReset
Description	sets all configurations back to "werkseinstellungen"
Input Parameters	none
Output Parameters	1. empty-string (compatibility for classic XmlRPC-Client)

Note: A factory reset will delete all applications which are saved on the camera.

"editApplication"

Method Name	editApplication
Description	Puts a specified Application into edit-status. This will attach an application-object to the RPC interface. The name of the object will be application independent. This does not change the "ActiveApplication"-parameter.
Input Parameters	1. Application index :int
Output Parameters	1. empty-string (compatibility for classic XmlRPC-Client)

"stopEditingApplication"

Method Name	stopEditingApplication
Description	Tells the device that editing this application was finished. Unsaved changed should be discard. HINT: The device must also call this implicit, when a edit-session timed out or was closed by "cancelSession".
Input Parameters	none
Output Parameters	1. empty-string (compatibility for classic XmlRPC-Client)

"createApplication"

Method Name	createApplication
Description	Creates an "empty" application. The embedded side should initialize all needed parameters and structures. Such an application might be in an non-activatable state, this means it could be saved on the device, but not set as active application.
Input Parameters	none
Output Parameters	1. Index of new application :int

"copyApplication"

Method Name	copyApplication
Description	Creates a new application by copying the configuration of another application. The device will generate an ID for the new application and put it on a free Index.
Input Parameters	1. Index of the application which should be copied :int
Output Parameters	1. Index of new application :int

"deleteApplication"

Method Name	deleteApplication
Description	Deletes the application form sensor If the deleted application was the active one, the sensor will have no active application anymore until the user picks one. (O2V-behavior)
Input Parameters	1. Index of application :int
Output Parameters	1. empty-string (compatibility for classic XmlRPC-Client)

"changeNameAndDescription" must be implemented in Edit-API

Method Name	changeNameAndDescription
Description	
Input Parameters	1. Application index :int 2. new name of the application: string(utf8, max. 64 character) 3. new description of the application: string(utf8, max. 500 character)
Output Parameters	1. empty-string (compatibility for classic XmlRPC-Client)

"moveApplications" must be implemented in Edit-API

Method Name	moveApplications
Description	Moves applications to other Index. There must be all applications in the new list, none of them duplicated and no Index used twice. The ID is a fixed value that stays the same as long as the application stays on the sensor. The Index could be changed and is used to address the application via PCIC, XML-RPC and Digital-IO.
Input Parameters	1. Array of structs (Id :int, Index :int)
Output Parameters	1. empty-string (compatibility for classic XmlRPC-Client)

9.4. DeviceConfig-Object

"activatePassword"

Method Name	activatePassword
Description	Set a password and activate it for the next edit-session. Making this change presistant requires to call "save" on the DeviceConfig.
Input Parameters	1. Password :string
Output Parameters	1. empty-string (compatibility for classic XmlRPC-Client)

"disablePassword"

Method Name	disablePassword
Description	Disables the password-protection. Making this change presistant requires to call "save" on the DeviceConfig.
Input Parameters	none
Output Parameters	1. empty-string (compatibility for classic XmlRPC-Client)

"save"

Method Name	save
Description	Store current configuration in persistent memory. If this is not called after changing device-parameters (via setParameter), changes will get lost on reboot.
Input Parameters	none
Output Parameters	1. empty-string (compatibility for classic XmlRPC-Client)

9.4.1 Parameters

Parameters of DeviceConfig
Methods for parameter access are defined here:

Parameter Name	Data Type	Description
Name	string (utf8)	User defined name of the device. (max. 64 characters)
Description	string (utf8)	User defined description of the device. (max. 500 characters)
ActiveApplication	int ***has limits**	Index of active Application This effects only RUN-mode: * defines the application active on startup (if static-application-switching is disabled) * contains the current active application (could also be changed via PCIC-command) * 0 means there is no application active (see SYRS 660606-3530)
PcicTcpPort	int	TCP/IP-Port for PCIC-connections.
PcicProtocolVersion	int ***has limits**	Sub Protocol of PCIC, see specification of PCIC:
IOLogicType	int ***has limits**	Defines logic-type of all digital-pins. Allowed values: 0: NPN 1: PNP
IODebouncing	bool	Applies to all inputs
IOExternApplicationSwitch	int ***has limits**	Allowed values: 0: off 1: static via I/O 2: pulse driven via I/O 3: pulse driven via trigger
SessionTimeout	int ***has limits**	number of seconds which a session stays before a call to "heartbeat"-method is needed
ExtrinsicCalibTransX	double	Extrinsic calibration, Transition in X direction
ExtrinsicCalibTransY	double	Extrinsic calibration, Transition in Y direction
ExtrinsicCalibTransZ	double	Extrinsic calibration, Transition in Z direction
ExtrinsicCalibRotX	double	Extrinsic calibration, Rotation around X-axis
ExtrinsicCalibRotY	double	Extrinsic calibration, Rotation around Y-axis
ExtrinsicCalibRotZ	double	Extrinsic calibration, Rotation around Z-axis
IPAddressConfig	int	**readonly**, The GUI requires to know if the device is on a Discovery-IP-address for multiple usecases. This information was extended to reflect all kinds of IP-address situations. Allowed values: 0: static (IP-address explicit defined inside the device) 1: DHCP (using a DHCP-server in the network) 2: LinkLocal (configured to DHCP, but no server which provided an address) 3: Discovery (changed by IP4Discovery mechanism)

PasswordActivated	bool	**readonly**, is true if the password-protection is enabled
OperatingMode	int	**readonly**, mode of device (RUN, EDIT) see "setOperatingMode" (the setter is outside of edit-mode, but inside of session)
DeviceType	string	**readonly**, Delivers a type description, unique by imager, evaluation-logic and device-interface Format could be like this: "[VendorID]:[TypeID]" e.g. "1:42"
ArticleNumber	string	**readonly**, Official catalog-number
ArticleStatus	string	**readonly**, Official two-letter status code
UpTime	double	**readonly**, Hours since last reboot
ImageTimestampReference	int	**readonly**, The image-data contains a timestamp (32bit int, µs) This should return the current timestamp as a reference to the images received.
TemperatureFront1	double	**readonly**, Temperature measured in the device, value is in degree Celsius. Measured by first sensor on imager board.
TemperatureFront2	double	**readonly**, Temperature measured in the device, value is in degree Celsius. Measured by second sensor on imager board.
TemperatureIMX6	double	**readonly**, Temperature measured in the device, value is in degree Celsius. Measured inside the main CPU.
TemperatureIllu	double	**readonly**, Temperature measured in the device, value is in degree Celsius. Measured on the illumination board.

***has limits**: parameters with this marker are listed in the reply of getAllParameterLimits-method

Default values of DeviceConfig parameters
The default values of the device configuration parameters are:

Parameter Name	Data Type	Default Value
Name	string (utf8)	"New sensor"
Description	string (utf8)	""
ActiveApplication	int	0
PcicTcpPort	int	50010
PcicProtocolVersion	int	3
IOLogicType	int	1
IODebouncing	bool	true
IOExternApplicationSwitch	int	0
SessionTimeout	int	30

ExtrinsicCalibTransX	double	0.0
ExtrinsicCalibTransY	double	0.0
ExtrinsicCalibTransZ	double	0.0
ExtrinsicCalibRotX	double	0.0
ExtrinsicCalibRotY	double	0.0
ExtrinsicCalibRotZ	double	0.0
IPAddressConfig	int	0
PasswordActivated	bool	false
OperatingMode	int	0

For all other DeviceConfig parameters there are no defined default values because they are either device-dependent (DeviceType, ArticleNumber, ArticleStatus) or volatile (UpTime, ImageTimestampReference).

Minimum and maximum values of DeviceConfig parameters
The minimum and maximum values of the device configuration parameters are:

Parameter Name	Minimum Value	Maximum Value
ActiveApplication	0	32
PcicProtocolVersion	1	4
IOLogicType	0	1
IOExternApplicationSwitch	0	3 (C2: 0)
SessionTimeout	5	300

9.5. Device/NetworkConfig-Object

"saveAndActivateConfig"

Method Name	saveAndActivateConfig
Description	Reinitialize the network interface so that it uses the configuration which was set by the other RPC-methods. There will be no XMLRPC-replay, because the network-interface is instantly reset. This must also close the session, as there is no clean way of closing it.
Input Parameters	none
Output Parameters	1. empty-string (compatibility for classic XmlRPC-Client)

9.6. ApplicationConfig-Object

"save"

Method Name	save
Description	stores current configuration in persistent memory. This should also be possible if the application is not yet in an "activatable" status.
Input Parameters	none
Output Parameters	1. empty-string (compatibility for classic XmlRPC-Client)

"forceTrigger"

Method Name	forceTrigger
Description	Do a software-trigger of current active application
Input Parameters	none
Output Parameters	1. empty-string (compatibility for classic XmlRPC-Client)

9.6.1. Parameters

Parameters of Application

Methods for parameter access are defined here:

Parameter Name	Data Type	Description
Name	string (utf8)	User defined name of the application. (max. 64 characters)
Description	string (utf8)	User defined description of the application. (max. 500 characters)
TriggerMode	int ***has limits**	Allowed values: 1: free run 2: process interface
PcicTcpResultOutputEnabled	bool	Allows to disable the automatic output of results via PCIC. If it is false, PCIC-commands could be used to access the data again.
PcicTcpResultSchema	string	The Schema defines which images and result-data will be send. It will also define the order of data-elements and additional separators. Contains single-enabling/-disabling of AmplitudeImage, IntensityImage, DistanceImage, XImage, YImage, ZImage, ConfidenceImage, DiagnosticData

***has limits**: parameters with this marker are listed in the reply of getAllParameterLimits-method

Default values of application parameters
The default values of application parameters are:

Parameter Name	Data Type	Default Value
Name	string (utf8)	"New application"
Description	string (utf8)	""
TriggerMode	int	1
PcicTcpResultOutputEnabled	bool	true
PcicTcpResultSchema	string	""

Minimum and maximum values of application parameters
The minimum and maximum values of application parameters are:

Parameter Name	Minimum Value	Maximum Value
TriggerMode	1	4 (C2: 2)

9.7. App./ImgagerConfig-Object

"changeType"

Method Name	changeType
Description	Changes the type of imager-configuration. This changes the set of available parameters and might also change available RPC-methods.
Input Parameters	1. type :string
Output Parameters	1. empty-string (compatibility for classic XmlRPC-Client)

"availableTypes"

Method Name	availableTypes
Description	Lists all available imager-configuration types.
Input Parameters	none
Output Parameters	1. Array of strings

9.7.1. Parameters

Parameters of all types of Application-ImagerConfig

Methods for parameter access are defined here:

Parameter Name	Data Type	Description
Type	string	**readonly**, Type of Imager-Configuration, see changeType method
FrameRate	double ***has limits**	Target frame rate in frames per second for free run mode.
ClippingLeft	int ***has limits**	Clipping-Area lower value in width-dimension
ClippingTop	int ***has limits**	Clipping-Area lower value in height-dimension
ClippingRight	int ***has limits**	Clipping-Area upper value in width-dimension
ClippingBottom	int ***has limits**	Clipping-Area upper value in height-dimension
ReduceMotionArtifacts	bool	enables a filtering for motion-artifacts. Not implemented for C2 samples.
SpatialFilterType	int ***has limits**	Allowed values: 0: off 1: Median-filter
AverageFilterNumPictures	int ***has limits**	Number of pictures to use for average filter Filter will be active if is unique 1.

***has limits**: parameters with this marker are listed in the reply of getAllParameterLimits-method

Default values of common ImagerConfig parameters. The default values of the common imager configuration parameters are:

Parameter Name	Data Type	Default Value
Type	string	"under5m_low"
FrameRate	double	5.0
ClippingLeft	int	0
ClippingTop	int	0
ClippingRight	int	175
ClippingBottom	int	131
ReduceMotionArtifacts	bool	false
SpatialFilterType	int	0
AverageFilterNumPictures	int	1

Minimum and maximum values of common ImagerConfig parameters
The minimum and maximum values of the common imager configuration parameters are:

Parameter Name	Minimum Value	Maximum Value
FrameRate	0.0167	100.0
ClippingLeft	0	175 (C2: 0)
ClippingTop	0	131 (C2: 0)
ClippingRight	0 (C2: 175)	175
ClippingBottom	0 (C2: 131)	131
SpatialFilterType	0	4 (C2: 1)
AverageFilterNumPictures	1	25 (C2: 1)

Parameters only in "under5m_low"-type of Application-ImagerConfig

Parameter Name	Data Type	Description
ExposureTime	int ***has limits**	Time for the Exposure

Default values of the "under5m_low" mode parameters

Parameter Name	Data Type	Default Value
ExposureTime	int	1905
Channel	int	0

Minimum and maximum values of the "under5m_low" mode parameters

Parameter Name	Minimum Value	Maximum Value
ExposureTime	0	17476
Channel	-1 (C2 0)	3 (C2 0)

Parameters only in "under5m_moderate"-type of Application-ImagerConfig

Parameter Name	Data Type	Description
ExposureTime	int ***has limits**	Time for the long Exposure The 2nd ExposureTime will be calculated based on the first one.

Default values of the "under5m_moderate" mode parameters

Parameter Name	Data Type	Default Value
ExposureTime	int	1905
Channel	int	0

Minimum and maximum values of the "under5m_moderate" mode parameters

Parameter Name	Minimum Value	Maximum Value
ExposureTime	0	17476
Channel	-1 (C2 0)	3 (C2 0)

10. Process Interface command reference

Note: All received messages which are send because of the following commands, will be send without "start"/"stop" at the beginning or ending of the string.

t command

command	t	
description	execute trigger and send process data asynchronously	
type	action	
reply	*	trigger was executed, the device captures an image and evaluates the result
	!	• device is busy with an evaluation • device is in an invalid state for this command, e.g. configuration mode • device is set to a different trigger source • no active application

l? command

command	l<image-ID>?	
description	request last image taken	
type	request	
reply	<length><image data>	
	!	• no image available • wrong ID
	?	• invalid command length
note	• <image-ID> 2 digits for the image type • <length> char string with exact 9 digits as decimal number for the image data size in bytes • <image data> image data	Valid image ID: ID • 01: Amplitude image • 02: not fully implemented • 03: Distance image • 04: Distance cartesian x image • 05: Distance cartesian y image • 06: Distance cartesian z image • 07: not fully implemented • 08: not fully implemented • 09: not fully implemented • 10: not fully implemented

		• 11: not fully implemented

p command

command	p<state>	
description	turns the PCIC output on or off, no images will be send from the camera in free run mode or as an answer of a trigger	
type	action	
reply	*	
	!	• <state> contains wrong value • device is in an invalid state for this command, e.g. configuration mode
	?	• invalid command length
note	• <state> 1 digit 0 deactivate 1 activate	at device restart the value configured within the application is essential for the output of data

V? command

command	V?	
description	request current protocol version	
type	request	
reply	<current version><emtpy><min version><emtpy><max version>	
note	• <current version> 2 digits for the current set version • <empty> space sign: 0x20 • <min/max version> 2 digits for the min and max available version to set	

v command

command	v<version>	
description	set the current protocoll verstion. The device configuration is not afected	
type	action	
reply	*	
	!	• invalid version
note	• <version> 2 digits for the protocol version	

Note: The default protocolversion is „V3“. Do not use the other versions for the C2 sample.

G? command

command	G?	
description	request device information	
type	request	
reply	<vendor><t><article number><t><name><t><location><t><description><t><ip><subnet mask><t><gateway><t><MAC><t><DHCP><t><port number>	
note	• <vendor> IFM ELECTRONIC • <t> Tabulator (0x09) • <article number> e.g. O3D300 • <name> UTF8 Unicocde string • <location> UTF8 Unicocde string • <description> UTF8 Unicocde string • <ip> IP-Adresse des Geräts als ASCII-Zeichenkette • <subnet mask> subnet mask of the device as ASCII • <gateway> Gateway of the device as ASCII • <MAC> MAC of the device as ASCII • <DHCP> ASCII string "0" for off and "1" for on	

H? command

command	H?	
description	returns a list with available commands	
type	request	
reply	H? - show this list t - execute Trigger T? - execute Trigger and wait for data I<image-id>? - get last image of defined type V? - get current protocol version v<version> - set protocol version G? - show device information	