



CE

Operating instructions

O3DC02

GB

80299431 / 00 01 / 2021



Contents



1	Preliminary note	4
1.1	Symbols used	4
1.2	Further documents	4
1.3	Legal and copyright information	4
1.4	Open source information	5
2	Safety instructions	6
3	Intended use	7
4	Items supplied	8
5	Accessories	9
6	Installation	10
6.1	Select installation location	10
6.2	Install device	11
6.3	Reduce the surface temperature of the device	11
6.4	Mounting accessories	11
7	Electrical connection	12
7.1	Wiring	13
7.1.1	Pin 1 / 3 (24 V / GND)	13
7.2	Wiring examples	13
7.2.1	Simple image capture	13
7.2.2	Install several devices in parallel	14
8	Operating and display elements	15
9	Set-up	16
9.1	Set parameters of the device	16
9.1.1	Object warning zones	17
9.1.2	Occupancy map	19
9.2	Detect objects	20
10	Maintenance, repair and disposal	22
10.1	Maintenance	22
10.2	Cleaning the housing surface	22
10.3	Repair	22
10.4	Disposal	22
10.5	Update firmware	22
10.6	Replace device	22
11	Approvals / standards	23
12	Appendix	24
12.1	Required ports	24
12.2	XML-RPC interface	24
12.2.1	Sample XML-RPC command	24
12.2.2	XML-RPC objects	25
12.3	XML-RPC command reference	26
12.3.1	Parameter API	26
12.3.2	Main object	27
12.3.3	Session object	29
12.3.4	Device config object	30
12.3.5	Device / Network config object	33
12.3.6	Device / Time config object	33
12.4	Process interface	34
12.4.1	Sending commands	35
12.4.2	Receiving images	36
12.4.3	Image data	37
12.4.4	Additional information for CONFIDENCE IMAGE	40
12.4.5	Configuration of PCIC output	41
12.5	Process interface command reference	45
12.5.1	a command (activate application)	45

12.5.2	A? command (occupancy of application list)	45
12.5.3	c command (upload PCIC output configuration)	46
12.5.4	C? command (retrieve current PCIC configuration).	46
12.5.5	E? command (request current error state))	46
12.5.6	G? command (request device information)	47
12.5.7	H? command (return a list of available commands)	47
12.5.8	I? command (request last image taken)	48
12.5.9	o command (set logic state of a ID).	48
12.5.10	O? command (request state of a ID)	49
12.5.11	p command (turn PCIC output on or off)	49
12.5.12	S? command (request current decoding statistics)	50
12.5.13	t command (execute asynchronous trigger)	50
12.5.14	T? command (execute synchronous trigger)	51
12.5.15	v command (set current protocol version)	51
12.5.16	V? command (request current protocol version)	51
12.6	Process interface commands ODS specific	52
12.6.1	Sending ego data to the device	52
12.6.2	Sending and retrieving zone configuration to the device	54
12.6.3	Switching the sensing state	55
12.7	Error codes	57
	Glossary	60

1 Preliminary note

You will find instructions, technical data, approvals and further information using the QR code on the unit / packaging or at www.ifm.com.

1.1 Symbols used

- ✓ Requirement
- ▶ Instructions
- ▷ Reaction, result
- [...] Designation of keys, buttons or indications
- Cross-reference
-  Important note
Non-compliance may result in malfunction or interference.
-  Information
Supplementary note

1.2 Further documents

- Quick reference guide
- Operating instructions
- Certificates



The documents can be downloaded at:
▷ www.ifm.com

1.3 Legal and copyright information

© All rights reserved by ifm electronic gmbh. No part of these instructions may be reproduced and used without the consent of ifm electronic gmbh.

All product names, pictures, companies or other brands used on our pages are the property of the respective rights owners.

- AS-i is the property of AS-International Association, (→ www.as-interface.net)
- CAN is the property of Robert Bosch GmbH, Germany (→ www.bosch.de)
- CAN is the property of CiA (CAN in Automation e.V.), Germany (→ www.can-cia.org)
- CODESYS™ is the property of CODESYS GmbH, Germany (→ www.codesys.com)
- DeviceNet™ is the property of ODVA™ (Open DeviceNet Vendor Association), USA (→ www.odva.org)
- EtherNet/IP® is the property of → ODVA™
- EtherCAT® is a registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany.
- IO-Link® is the property of PROFIBUS Nutzerorganisation e.V., Germany (→ www.io-link.com)
- ISOBUS is the property of AEF - Agricultural Industry Electronics Foundation e.V., Germany (→ www.aef-online.org)
- Microsoft® is the property of Microsoft Corporation, USA (→ www.microsoft.com)
- Modbus® is the property of Schneider Electric SE, France (→ www.schneider-electric.com)

- PROFIBUS® is the property of PROFIBUS Nutzerorganisation e.V., Germany (→ www.profibus.com)
- PROFINET® is the property of → PROFIBUS Nutzerorganisation e.V., Deutschland
- Windows® is the property of → Microsoft Corporation, USA

1.4 Open source information

This product can contain Free Software or Open Source Software from various software developers which is subject to the following licenses: General Public License version 1, version 2 and version 3 (General Public License version 3 in conjunction with the GNU Compiler Collection Runtime Library Exception version 3.1), Lesser General Public License version 2.1, Lesser General Public License version 3, Berkeley Software Distribution („This product includes software developed by the University of California, Berkeley and its contributors“), The Academic Free License version 2.1. For the components subject to the General Public License in their respective versions the following applies:

This program is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation. If version 1 applies to the software: either version 1 of the License or (at your option) any later version; if version 2 (or 2.1) applies to the software: either version 2 (or 2.1) of the License or (at your option) any later version; if version 3 applies to the software: either version 3 of the License or (at your option) any later version. The following disclaimer of the software developers applies to the software components that are subject to the General Public License or the Lesser General Public License in their respective versions: The Free Software is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License and the GNU Lesser General Public License for more details.

The responsibility of ifm electronic gmbh for ifm products, in the case of product-specific software, remains unaffected by the above disclaimer. Please note that the firmware for the ifm products is in some cases provided free of charge. The price of the ifm products has then to be paid for the respective device itself (hardware) and not for the firmware. For the latest information on the license agreement for your product please visit www.ifm.com

For binaries that are licensed under any version of the GNU General Public License (GPL) or the GNU LGPL you may obtain the complete corresponding source code of the GPL software from us by sending a written request to: opensource@ifm.com or to ifm electronic gmbh, Friedrichstraße 1, 45128 Essen, Germany.

We charge €30 for each request. Please write “source for product Y” in the memo line of your payment. Your request should include (i) the name of the covered binary, (ii) the name and the version number of the ifm product, (iii) your name and (iv) your return address.

This offer is valid to anyone in receipt of this information. This offer is valid for at least three years (from the date you received the GPL/LGPL covered code).

2 Safety instructions

- The unit described is a subcomponent for integration into a system.
 - The system architect is responsible for the safety of the system.
 - The system creator undertakes to perform a risk assessment and to create documentation in accordance with legal and normative requirements to be provided to the operator and user of the system. This documentation must contain all necessary information and safety instructions for the operator, the user and, if applicable, for any service personnel authorised by the architect of the system.
- Read this document before setting up the product and keep it during the entire service life.
- The product must be suitable for the corresponding applications and environmental conditions without any restrictions.
- Only use the product for its intended purpose (→ Intended use).
- If the operating instructions or the technical data are not adhered to, personal injury and/or damage to property may occur.
- The manufacturer assumes no liability or warranty for any consequences caused by tampering with the product or incorrect use by the operator.
- Installation, electrical connection, set-up, operation and maintenance of the product must be carried out by qualified personnel authorised by the machine operator.
- Protect units and cables against damage.

3 Intended use

The O3DCxx 3D sensor is an optical sensor measuring the distance between the sensor and the nearest surface point by point using the time-of-flight principle. The O3DCxx 3D sensor is hereinafter referred to as the device. The device illuminates the scene with an infrared light source and calculates the distance by means of the light reflected from the surface.

From the image data, process values are generated via internal image processing and compared to threshold values. The device cyclically signals whether there is an object in the geometrical zone defined by the user.

Because of the requirements for electromagnetic interference emissions, the device is intended for use in industrial environments. The device is not suitable for use in domestic areas.



Environmental conditions

- ▷ The device may only be used under the operating conditions specified in the data sheet.

4 Items supplied

- Device
- Spring washers (only for devices O3DC0x)
- Quick reference guide



▷ Data sheet and documentation: www.ifm.com

5 Accessories

The following accessories are needed for operation:

Articles	Description
E11950	Power supply cable for camera/sensor
E11898	M12 industrial Ethernet connection cable



▷ The ifm Vision Assistant software is available on the internet: www.ifm.com

6 Installation

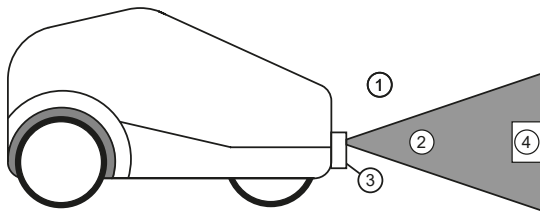


Fig. 1: Installed device with a detected object

1 Surroundings (non-detectable area)
3 Device

2 Detectable area
4 Object

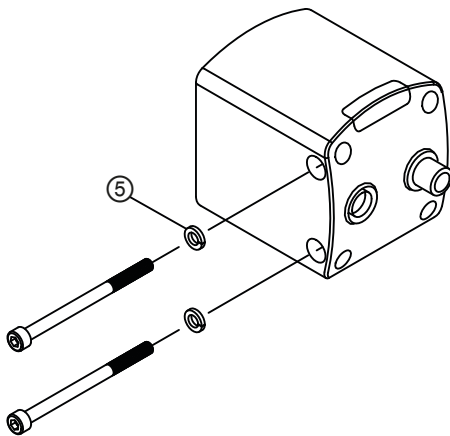


Fig. 2: Install device with spring washers

5 Spring washer

6.1 Select installation location

Observe the following instructions for the selection of the installation location:

- ▶ The object (4) is only detected if it is in the detectable area (2).
- ▷ The size of the detectable area is indicated in the data sheet of the device. The size additionally depends on the distance between the device (3) and the object (4). With increasing distance the detectable area becomes larger.
- ▶ Keep a distance of > 20 cm between the device and the floor.
- ▷ Different viewing angles can lead to an increase or reduction of the distance to the floor.
- ▶ Take tolerances into account when positioning the object.
- ▶ Keep the surroundings (1) free of objects.
- ▶ Do not install the device in heavily polluted areas.
- ▷ The lens of the device will get dirty.
- ▶ Avoid a transparent pane between the device (3) and the object (4).
- ▷ Even when using a very clean glass pane, some of the light is reflected.



Observe instructions

- ▷ Failure to observe the instructions will result in measurement errors.

6.2 Install device

Observe the following instructions when installing the device:

- ▶ Install the sensor using 2x M5 screws or mounting set.
- ▷ The bore dimensions for the M5 screws are indicated in the data sheet.
- ▷ The mounting set is available as accessory.
- ▶ Install the device (3) with the spring washers (5) with max. 3 Nm.
- ▶ Use strain reliefs for all cables connected to the device.



Corrosion in wet areas

- ▷ If the camera is permanently used in wet areas, the nut of the M12 industrial Ethernet connection cable (e.g. E11898) may corrode. Use a connection cable with a high-grade stainless steel nut for permanent use in wet areas.

6.3 Reduce the surface temperature of the device

During operation, the device may heat up. With the following measures the surface temperature of the device can be reduced:

- ▶ Install the device on heat-conductive metal parts.
- ▷ A large-surface contact of the device with metal parts increases heat dissipation (e.g. aluminium).
- ▶ Use a heat conductor when installing the device on metal parts.
- ▷ The heat-conductive effect is increased by means of the heat conductor. The heat conductor is available as accessory.
- ▶ Reduce obstructions around the device and the density of objects mounted near the device.
- ▷ Obstructions around the device and a high installation density may have a negative impact on convection (air movement).
- ▶ Mount one or two heat sinks on the device.
- ▷ The heat sinks increase the surface of the sensor, reducing the surface temperature. The heat sinks are available as accessories.

6.4 Mounting accessories

The following accessories are needed depending on the installation:

Articles	Description
E3D301	Smart Camera mounting set
E3D302	Smart Camera cooling element
E3D303	Smart Camera heat conductor
E3D304	2x Smart Camera cooling element



- ▷ Information about the accessories: www.ifm.com

7 Electrical connection

ATTENTION

High surface temperature

- ▷ Depending on the operating mode, the device may heat up. The difference between the device's surface temperature and the ambient temperature must not exceed 25 degrees. Take one or several of the following measures:
 - ▶ Adjust the operating mode.
 - ▶ Make sure the device is sufficiently cooled (e.g. with cooling element and heat conductor).
 - ▶ Use a contact protection.

ATTENTION

Electrical voltage

- ▷ The device must be connected by a qualified electrician.
- ▷ Device of protection class III (PC III).
- ▷ The electrical supply must only be made via PELV circuits.
- ▷ Electric supply must comply with UL61010-1, chap. 9.4 - Limited Energy.
- ▷ The separation of external circuits must comply with UL61010-2-201, Fig. 102.
 - ▶ Disconnect power before connecting the device.
 - ▶ For cable lengths > 30 m use an additional protection against surge voltages to IEC6100-4-5.

ATTENTION

Moisture penetration

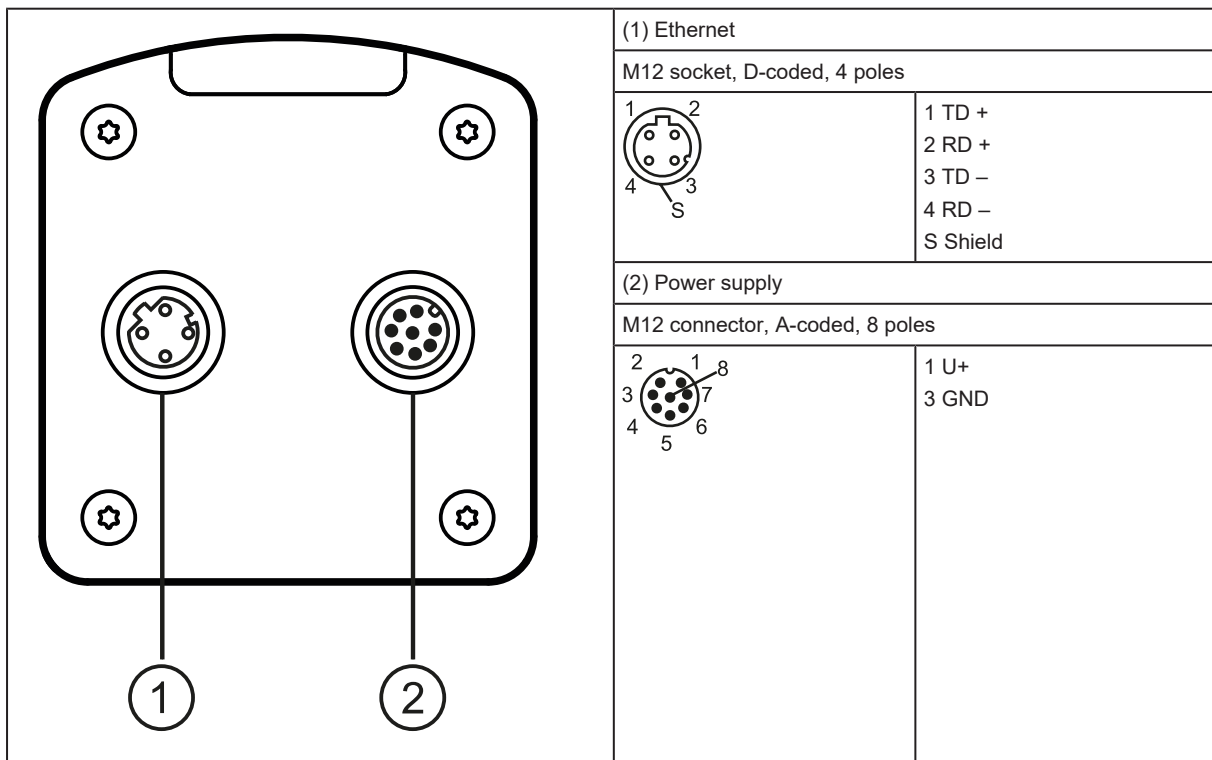
- ▷ The IP rating indicated in the data sheet is only guaranteed if the M12 connectors are firmly screwed.
- ▷ The device can be damaged by insufficiently tightened M12 connectors.
 - ▶ Firmly screw the M12 connectors to the device.



Scope of validity cULus

- ▷ Observe the minimum temperature rating of 70 °C of the cable to be connected to the field wiring terminals.

7.1 Wiring



GB

ATTENTION

Moisture penetration

- ▷ If the Ethernet connection is not used, moisture can enter the device through the unprotected Ethernet connection.
- ▶ Cover the unused Ethernet connection with the protective cap E73004.
- ▶ Tighten the protective cap applying 0.6..0.8 Nm.

7.1.1 Pin 1 / 3 (24 V / GND)

The permissible voltage range is indicated in the data sheet of the sensor.

7.2 Wiring examples

7.2.1 Simple image capture

The illustration shows the wiring of the device to the control computer of a vehicle.

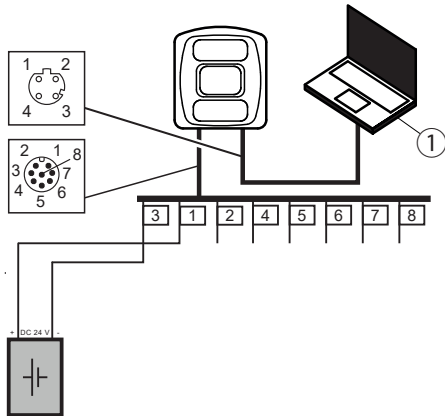


Fig. 3: Wiring example

1 Vehicle control computer

7.2.2 Install several devices in parallel

The illustration shows the wiring of 4 devices (C1 to C4) to the control computer of a vehicle.

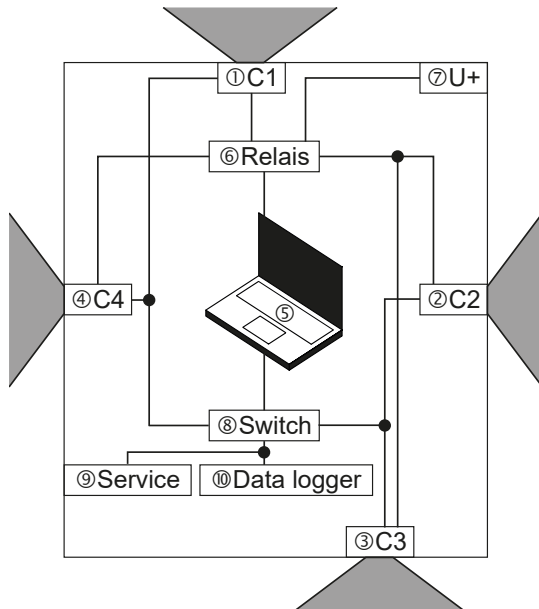


Fig. 4: Wiring example for several devices

- | | |
|----------------------------|----------------|
| 1 Device 1 | 2 Device 2 |
| 3 Device 3 | 4 Device 4 |
| 5 Vehicle control computer | 6 Relay |
| 7 Voltage supply | 8 Switch |
| 9 Service access | 10 Data logger |

Each device faces a different direction. Due to the specific arrangement of the devices, a partial 360° view of the vehicle is realised. The components must guarantee the necessary performance (for example, data of proper motion).



Trouble-free operation

- ▷ The O3DCxx 3D sensors do not influence each other during obstacle detection.

8 Operating and display elements

Via the LED indicators 1 - 4 the device signals the current operating status.

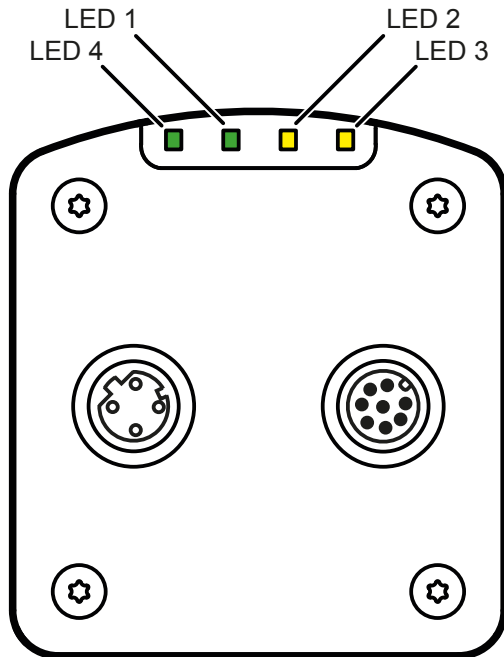




Fig. 5: Indicators of the device




LED 4 "Ethernet"	LED 1 "Power"	LED 2 "OUT 1"	LED 3 "OUT 2"	Description
	on			The device is ready for operation, supply voltage applied
	flashes at 0.5 Hz			No parameters set or parameter setting was not loaded into the device. On  Off
	flashes 2x at 0.5 Hz			The device is in the parameter setting mode. On  Off
		on		The device is ready (IDLE state) (→ Switching the sensing state □ 55).
		flashes at 10 Hz	flashes at 10 Hz	Obstacle detection is active (SENSING state) (→ Switching the sensing state □ 55).
on				Ethernet connected.
flashes				Ethernet transmitting data.
off				Ethernet not connected.
		flashes at 8 Hz	flashes at 8 Hz	The device signals an internal error.
		flashes at 2 Hz	flashes at 2 Hz	The device signals a recoverable error. The error information can be read via Ethernet.
		running light →		Device booting.
		running light ←		The device carries out a firmware update.

9 Set-up

After power on, the device is put into operation. After approx. 15 seconds, the device is in standby and ready to receive configuration commands. The indicators signal the current operating status.

9.1 Set parameters of the device

The parameters of the device are set with the ifm Vision Assistant software or via XML-RPC and process interface commands.

-  ▷ The ifm Vision Assistant software and the software manual are available on the internet: www.ifm.com
-  ▷ Third party libraries can simplify communication via XML-RPC (for example <https://docs.python.org/3/library/xmlrpc.html>)
-  ▷ The XML-RPC interface is only required for parameter setting. In normal operation, the XML-RPC interface is not necessary for communication between a controller and the device.

Individual IP address

When using multiple devices on one vehicle, the default IP address "192.168.0.69" needs to be changed. The XML-RPC object "device/networkconfig" is required for this (→ Device / Network config object [□ 33](#)). The parameter "StaticIPv4Address" is changed with the "setParameter" method (→ Parameter API [□ 26](#)). The new configuration is activated with the "saveAndActivateConfig" method.

NTP server configuration

The IP address of the NTP server is set with the XML-RPC object for time configuration "device/time" (→ Device / Time config object [□ 33](#)). First, the parameter "NTPServers" is set to the IP address of the NTP server with the "setParameter" method (→ Parameter API [□ 26](#)). Subsequently, the synchronisation is activated with the parameter "SynchronisationActivated". The new configuration is activated with the "saveAndActivateConfig" method. Finally, a restart of the device is required.

Extrinsic calibration

The coordinate systems of the device and vehicle must be calibrated to each other, as the object warning zones are specified in vehicle coordinates. The extrinsic calibration transforms the intrinsic coordinate system of the device into the coordinate system of the vehicle. The estimation is based on the observation of a calibration pattern whose position is given in vehicle coordinates.

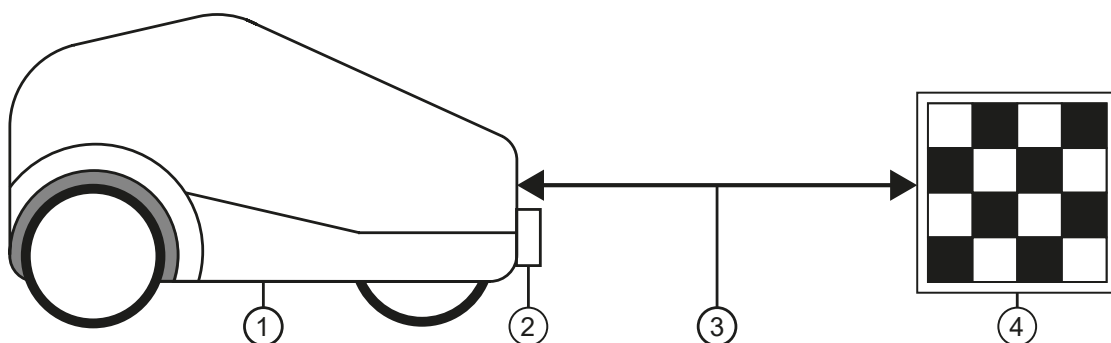


Fig. 6: Extrinsic calibration of the device

- | | |
|---|-----------------------|
| 1 Automated guided transport vehicle (FTF) | 2 Device |
| 3 Distance between device and calibration pattern | 4 Calibration pattern |

An 80x50 cm chequerboard pattern with 10x10 cm squares is suitable as a calibration pattern. The calibration pattern is placed on a flat plate at a distance of 75-105 cm from the device with a maximum uncertainty of the reference points of ± 5 mm.

With standard methods for camera calibration, a rotation and translation are extracted, which describe the correlation between the vehicle and device coordinate system. The parameters can be saved on the device via XML-RPC. For this, the XML-RPC object "DeviceConfig" is needed (→ Device config object ¶ 30). The corresponding parameters are indicated in the following table (→ Parameter API ¶ 26).

Parameter name	Unit	Description
ExtrinsicCalibRotX	[°]	Rotation around the x-axis
ExtrinsicCalibRotY	[°]	Rotation around the y-axis
ExtrinsicCalibRotZ	[°]	Rotation around the z-axis
ExtrinsicCalibTransX	[mm]	Translation in x-direction
ExtrinsicCalibTransY	[mm]	Translation in y-direction
ExtrinsicCalibTransZ	[mm]	Translation in z-direction

The new configuration is activated with the "save" method.

9.1.1 Object warning zones

The device has three independent object warning zones in which objects are detected and displayed. Each object warning zone is defined by a polygon with six corner points. A unique configuration ID is assigned for each object warning zone. The object warning zones are set with process interface commands (→ Process interface command reference ¶ 45) and sent to the device (→ Sending and retrieving zone configuration to the device ¶ 54).

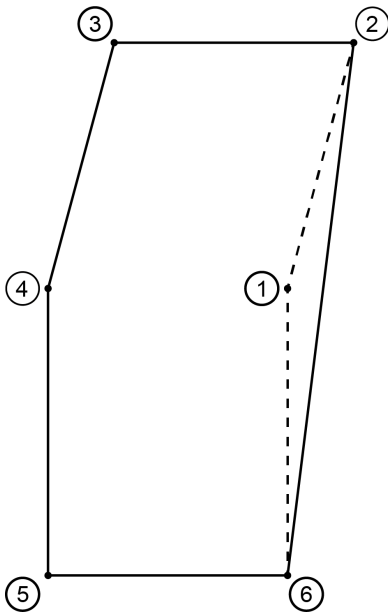


Fig. 7: Polygon with 6 corner points

The 6 corner points are given in vehicle coordinates. The object warning zone results from the convex hull of the 6 corner points, recognisable by the continuous lines in the polygon. 2 corner points can have the same coordinates.

There is a global height for all object warning zones. The 6 corner points and the global height result in a three-dimensional object warning zone. The device does not check the validity of the selected points in relation to the field of view of the device.

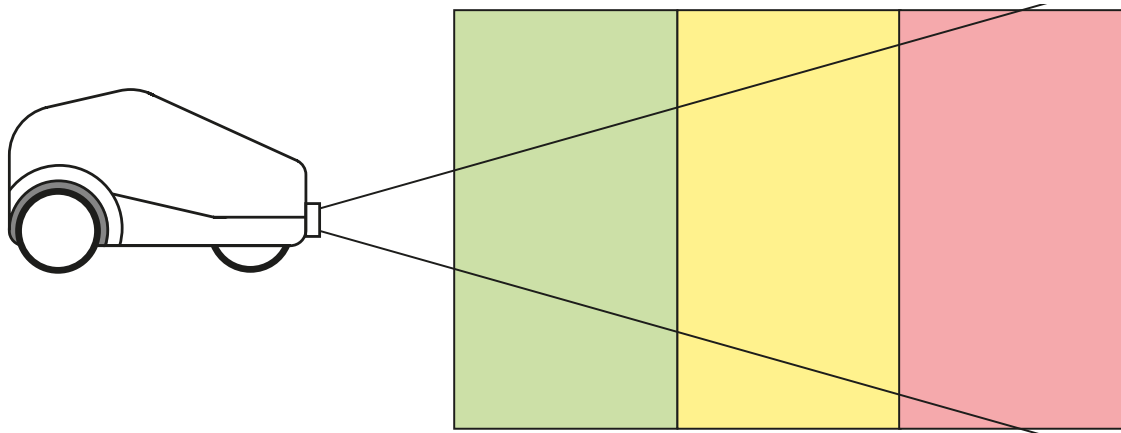


Fig. 8: Device with object warning zones

If an object warning zone is defined beyond the field of view of the device (or above/behind it), the device cannot detect objects in these areas. A minimum distance of the object warning zone to the device is not necessary.

With the 6 corner points and the 3 object warning zones, a curve-like object warning zone can be determined. The default configuration of the object warning zones is [0.0] for all corner points and zone identification parameters. Therefore, for the standard configuration of the object warning zones, an object is never detected and displayed in the zone assignment.

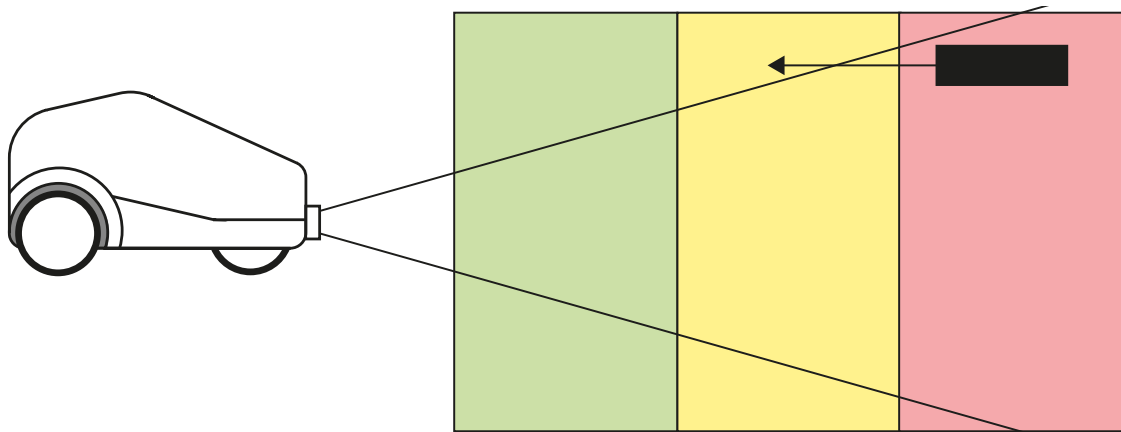


Fig. 9: Object moves out of the detection range

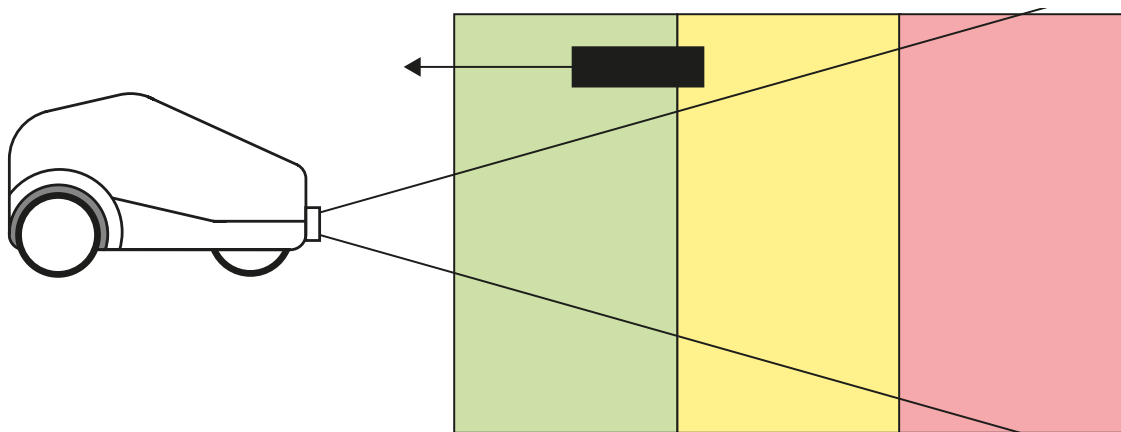


Fig. 10: Object is outside the detection range

ATTENTION

Objects are not recognised

- ▷ Objects outside the detection range are not detected by the device. For example, objects such as boxes directly in front of the vehicle are problematic if they are outside the detection range.
- ▶ Observe objects outside the detection range.



Object warning zones will not be saved

- ▷ The object warning zones are not saved in the device. After restarting the device, the object warning zones must be set again.

GB

9.1.2 Occupancy map

Independently of the object warning zones, the device can issue an occupancy map. The occupancy map covers a range of ± 5 m in x- and y-direction of the coordinates of the vehicle.

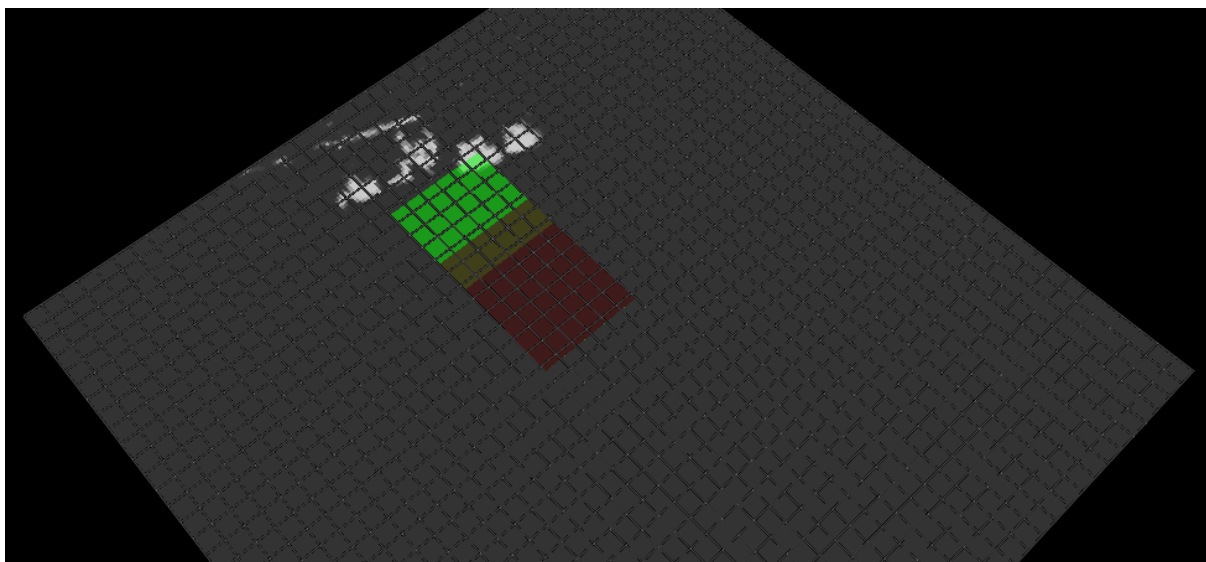


Fig. 11: Occupancy map with the grid of 200x200 cells

The occupancy map is divided into a grid of 200x200 cells. For each cell (5x5 cm), the probability of an obstacle is output at the position. The FTF with the mounted device is located at position 0.0 in the middle of the occupancy map.

Three object warning zones are set in the occupancy map in the area of the green, yellow and red cells. The white coloured cells have a non-zero value and an increased probability that there is an obstacle at the position. A cell can take values in the range "0" to "255". If the value is "0", an obstacle is most likely not present. If the value is "255", an obstacle is most likely present. The green object warning zone detects an obstacle and triggers accordingly.

The following figure shows the distance, amplitude and confidence image for the occupancy map shown above.



Fig. 12: Distance image (left), amplitude image (centre) and confidence image (right)

The output of the occupancy map is set with the process interface commands (→ Configuration of PCIC output ¶ 41). The associated image ID is "occupancy_map".

Example of a JSON string

```
{ "layouter": "flexible", "format": { "dataencoding": "ascii" }, "elements":
[ { "type": "string", "value": "star", "id": "start_string" }, { "type":
"blob", "id": "occupancy_map" }, { "type": "string", "value": "stop", "id":
"end_string" } ] }
```

The element "0" of the output matrix corresponds to the cell at x = -5 m and y = -5 m in the coordinates of the vehicle.

The element "199" of the output matrix corresponds to the cell at x = -5 m and y = +5 m in the coordinates of the vehicle.

The last element "39999" of the output matrix corresponds to the cell at x = +5 m and y = +5 m in the coordinates of the vehicle.

9.2 Detect objects

After the parameters have been set in the device, the detection of objects can be activated:

- ▶ start the device,
- ▶ provide the proper motion data,
- ▶ switch the device to detection mode.



Commands are not answered

- ▶ The device connects to the set NTP server after the restart. At this point, the ODS application is incompletely loaded. This allows a connection to the device, but commands are not answered or rejected. The device is only fully functional after the ODS application has been completely loaded.

Proper motion data

The proper motion is described with the following parameters:

Parameter name	Unit	Precision	Description
VelocityX	[m/s]	+ - 10 mm/s	Velocity in x-direction in the vehicle coordinate system
VelocityY	[m/s]	+ - 10 mm/s	Velocity in y-direction in the vehicle coordinate system
GearRate	[rad/s]	+ - 1°/s	Rotation in radian per time interval in the vehicle coordinate system
TimeStamp	[ns]	+5ms	Time stamp of the end position (synchronised for sensor and vehicle control system)

The device requires information on the vehicle's own movement at a constant repetition rate of 30 Hz.

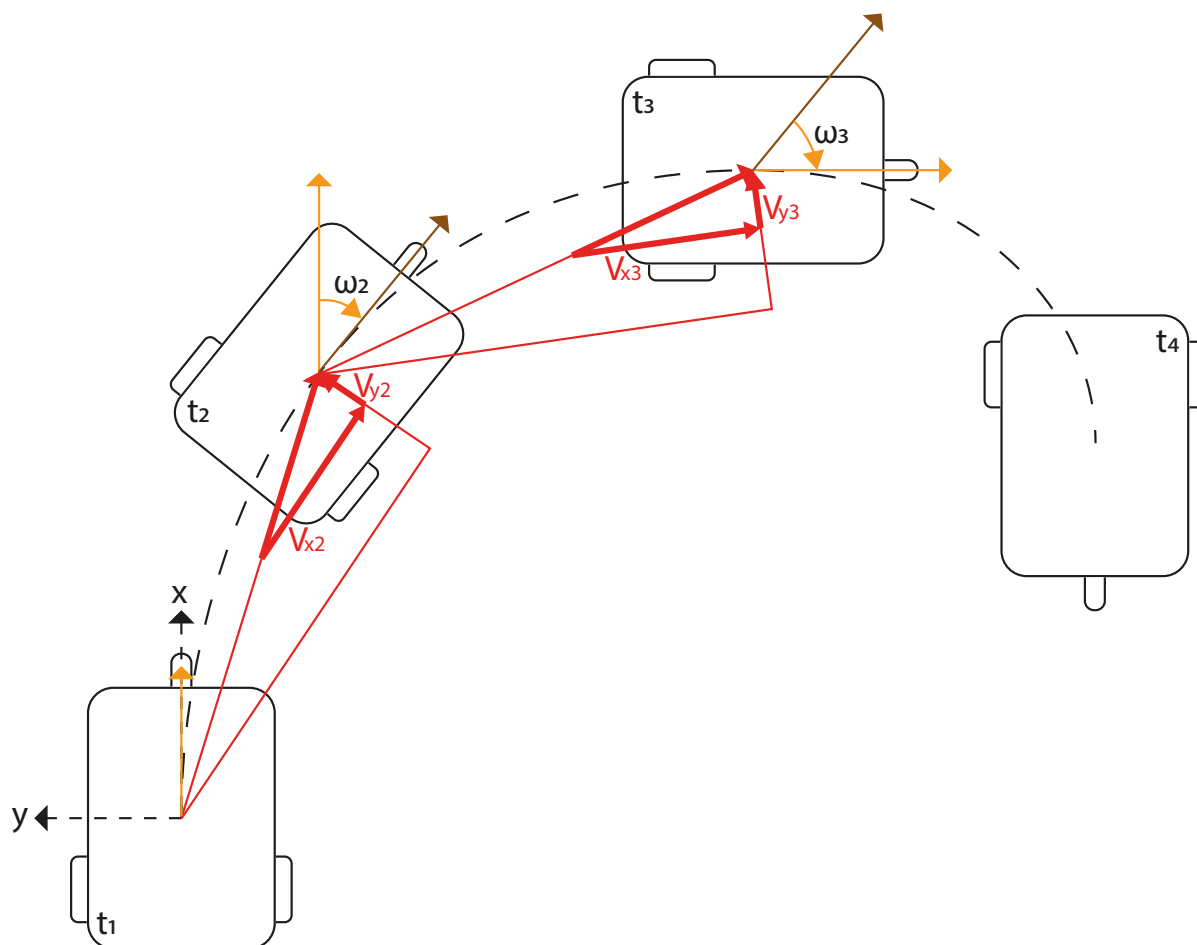


Fig. 13: Vehicle's own movement

The vehicle's own movement at time t_i ($i = 1, 2, 3, \dots$) is described by the velocity in x-direction v_{xi} and y-direction v_{yi} and by the yaw rate ω_i .

The proper motion data is sent to the device in the vehicle's coordinate system via process interface commands (→ Sending ego data to the device □ 52).

Device mode

The device can assume the mode "BOOT", "IDLE", "SENSING" or "ERROR".

The device is in "BOOT" mode during start-up.

The device is in "IDLE" mode after start-up. The unit does not emit light and no data is recorded to save energy. Device mode is intended for the vehicle when it is not moving.

In the "SENSING" device mode, obstacle detection is activated. The unit emits light, data is recorded, proper motion data is evaluated and zone occupancy information is output. Device mode is intended for the vehicle when it is moving.

The device is in "ERROR" mode in case of a fault.



- ▷ Process interface commands allow switching between the device modes "IDLE" and "SENSING" (→ Switching the sensing state □ 55).

10 Maintenance, repair and disposal

10.1 Maintenance

The unit is maintenance-free.

10.2 Cleaning the housing surface

- ▶ Disconnect the device.
- ▶ Clean the device from dirt using a soft, chemically untreated and dry cloth.



Micro-fibre cloths without chemical additives are recommended.

- ▶ In case of severe soiling, use a damp cloth.

10.3 Repair

- ▶ The device must only be repaired by the manufacturer.
- ▶ Observe the safety instructions.

10.4 Disposal

- ▶ Dispose of the unit in accordance with the national environmental regulations.

10.5 Update firmware

The firmware of the device can be updated using the ifm Vision Assistant software.



Data loss due to firmware update

- ▷ Some parameters saved in the sensor get lost when the firmware is updated. A backup of the parameters can be created with the ifm Vision Assistant.
 - ▶ Export parameters before updating the firmware.
 - ▶ Import parameters after updating the firmware.

10.6 Replace device



Data loss due to device replacement

- ▷ The parameters stored in the device are lost through the replacement. A backup of the parameters can be created with the ifm Vision Assistant.
 - ▶ Export parameters before updating the firmware.
 - ▶ Import parameters after updating the firmware.



Backing up the parameters has an additional benefit: With the backup, several devices can be quickly equipped with identical parameters.

11 Approvals / standards

Test standards and regulations (→ Technical data)

The EU declaration of conformity and approvals can be found at:
www.ifm.com

12 Appendix

12.1 Required ports

The following ports are required for the sensor configuration using XML-RPC and for receiving data on the process interface. They must not be blocked by a firewall or router.

- TCP/HTTP: 80
- TCP: 50010

If the ifm Vision Assistant is used, the following additional ports must also be available:

- UDP: 3321
- TCP/HTTP: 8080

It is possible to configure another port than 50010 for the process interface. If a different port is used, it must not be blocked either.

12.2 XML-RPC interface

In case the device is not configured by the ifm Vision Assistant, the XML-RPC interface can be used instead.



▷ General information about XML-RPC is found on the website <http://xmlrpc.com/spec.md>

To send a command via the XML-RPC interface, the command must have a special layout. Linefeeds and carriage returns are essential.



▷ Every command which is sent via the XML-RPC interface must end with carriage return `<CR>` and linefeed `<LF>`.

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

12.2.1 Sample XML-RPC command

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

```
POST /RPC3 HTTP/1.0<CR><LF>
User-Agent: Frontier/5.1.2 (WinNT)<CR><LF>
Host: betty.userland.com<CR><LF>
Content-Type: text/xml<CR><LF>
Content-length: 181<CR><LF>
<CR><LF>
<?xml version="1.0"?><CR><LF>
<methodCall><CR><LF>
<methodName>examples.getStateName</methodName><CR><LF>
<params><CR><LF>
<param><CR><LF>
<value><i4>41</i4></value><CR><LF>
</param><CR><LF>
</params><CR><LF>
</methodCall><CR><LF>
```


The following example contains one O3DCxx command:


```

POST /api/rpc/v1/com.ifm.efector/ HTTP/1.1 <CR><LF>
User-Agent: Frontier/5.1.2 (WinNT)<CR><LF>
Host: 192.168.0.69<CR><LF>
Content-Type: text/xml<CR><LF>
Content-length: 94<CR><LF>
<CR><LF>
<?xml version="1.0"?><CR><LF>
<methodCall><CR><LF>
<methodName>getParameter</methodName><CR><LF>
    
```

12.2.2 XML-RPC objects

To communicate and to configure the device via XML-RPC the XML-RPC commands have to use different XML-RPC objects. Different commands need different XML-RPC objects (→ XML-RPC command reference □ 26). The interface of O3DCxx is structured in an object-oriented way. Some of the objects are available all the time, others are only available after bringing the device into a special mode by calling a method on an already available object. This mechanism is used to create system requirements (e.g. password protection).

 ▷ It could be necessary to send heartbeats so that there will be no session timeout.

The following diagram should give an overview how objects are related to each other and which methods must be called to make others available:

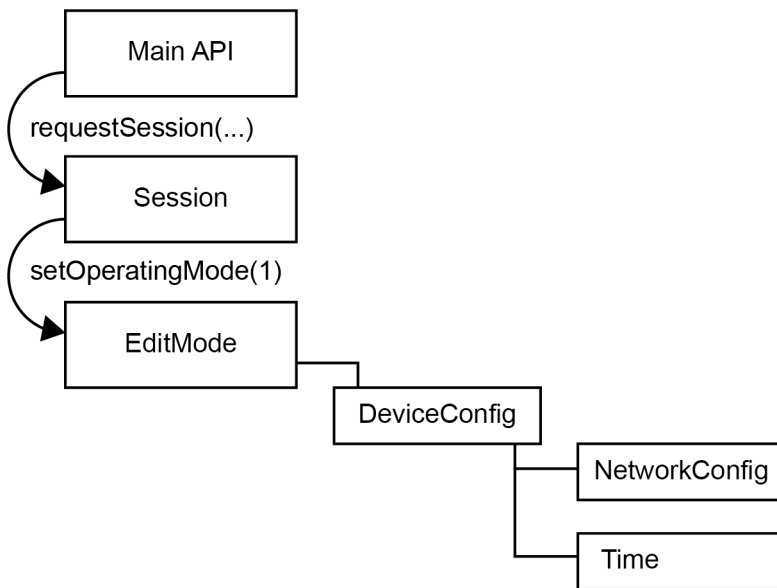


Fig. 14: XML-RPC Objects 1

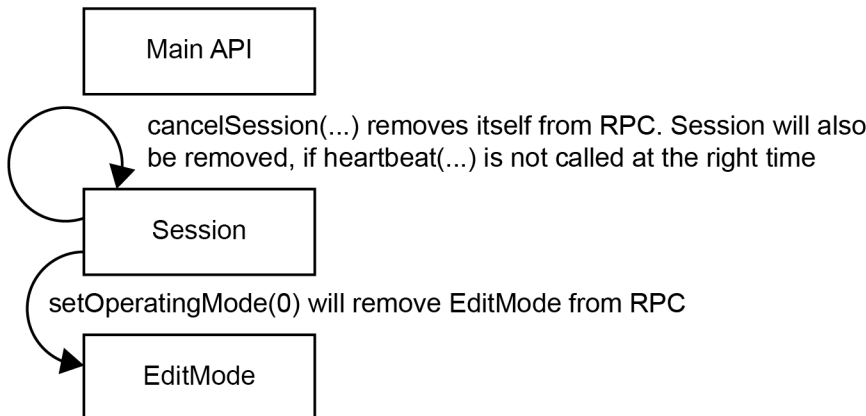


Fig. 15: XML-RPC Objects 2

Main object

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

This is the main object of RPC. It contains methods to open a session. The session contains methods for activating the edit mode. Most of its methods are only getters, because it should be possible to protect editing with a password.

Session object

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

The URL part "d21c80db5bc1069932fbb9a3bd841d0b" is the session ID. It is returned by the command "requestSession" of the main object. If the command "requestSession" is called without a user-defined session ID, which can be passed as a parameter, a random session ID is generated automatically.

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 the edit operating mode. The index of applications must be between 1 and 32. The device must only support 32 applications and the indexes must start at 1.

DeviceConfig object

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

Device/NetworkConfig object

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

Device/Time object

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

12.3 XML-RPC command reference

12.3.1 Parameter API

The parameters `setParameter`, `getParameter`, `getAllParameters` and `getAllParameterLimits` are implemented in the following RPC objects:

- Device
- Network
- Time

setParameter

Method Name	Description	Input parameter	Output parameter
<code>setParameter</code>	Sets a parameter to a specific value.	Name of parameter: string. New value: string.	Empty string (compatible with classic XmlRPC client).

getParameter

Method Name	Description	Input parameter	Output parameter
getParameter	Returns the current value of the parameter.	Name of parameter: string.	Value of parameter: string.

getAllParameter

Method Name	Description	Input parameter	Output parameter
getAllParameters	Returns all parameters of the object in one data structure.	None.	Struct (name contains the parameter name, value contains the stringified parameter value).

getAllParameterLimits

Method Name	Description	Input parameter	Output parameter
getAllParameterLimits	Returns limits of all numeric parameters, that have limits defined on the device.	None.	Struct of Structs (name in first struct is the parameter name, substructs contains: min :string, max :string) E.g.: {"ExposureTime1": { "min": "123", "max": "432" }, "ExposureTime2": { "min": "123", "max": "432" }}

Parameter string encoding

Non-string parameters must be encoded in the following format.

Type	Stringified
bool	"true" / "false" setParameter method also accepts "1" / "0". Getter methods must always return "true" / "false".
int	decimal (e.g. "-1234" / "1234") Values should be in the range of int32 (-2 ³¹ ...2 ³¹).
double	English floating-point notation (optional with exponent). E.g. "1.2", ".3", "4.5e6", "-7E-8", "-inf", "nan"



- ▷ Structured types (array or structs) can't be put into parameter storage in a general way. Encoding of arrays must specified on specific parameters.

12.3.2 Main object

getParameter

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

getAllParameters

Method Name	Description	Input parameter	Output parameter
getAllParameters	Getter for the parameters described here. This is an additional getter outside of edit sessions. It is possible to read device information without login.	None.	Struct (name contains the parameter name, value contains the stringified parameter value).

getSWVersion

Method Name	Description	Input parameter	Output parameter
getSWVersion	Returns version information of all software components.	None.	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	Description	Input parameter	Output parameter
getHWInfo	Returns hardware information of all components.	None.	Struct of strings, e.g. { "MACAddress": "00:02:01:40:06:C9", "Frontend": "#!01_F340_001_...", ... } Mandatory keys: "MACAddress" "Connector" "Diagnose" "Frontend" "Illumination" "Mainboard"

requestSession

Method Name	Description	Input parameter	Output parameter
requestSession	<p>Requests a session object for access to the configuration and for changing the device operating mode.</p> <p>This blocks parallel editing and allows protection of editing with a password.</p> <p>The ID could optionally be defined by the external system, but it must be the defined format (32char "hex").</p> <p>If it is called with only one parameter, the device will generate a session ID.</p> <p>The session will start with a default timeout ("SessionTimeout" device parameter). The timeout can be extended by calling "heartbeat".</p> <p>The device will stay in RUN mode.</p> <p>If password is disabled on the device, the value given as password parameter is ignored.</p>	Password: string Session ID: string (optional)	Session ID: string

**reboot**

Method Name	Description	Input parameter	Output parameter
reboot	Reboot system. Parameter defines which mode/system will be booted.	Type of system that should be booted after shutdown: int "0": Productive mode "1": Recovery mode	Output: string

systemCommand

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

12.3.3 Session object**heartbeat**

Method Name	Description	Input parameter	Output parameter
heartbeat	<p>Extends the lifetime of the edit session.</p> <p>If the given value is outside the range of "SessionTimeout", the saved default timeout will be used.</p>	Requested timeout interval till next heartbeat, in seconds: int	The used timeout interval, in seconds: int

cancelSession

Method Name	Description	Input parameter	Output parameter
cancelSession	<p>Explicit stop of this session.</p> <p>If an application is still in edit mode, it will implicitly do the same as "stopEditingApplication".</p>	None.	Empty string (compatible with classic XmlRPC client).

setOperatingMode

Method Name	Description	Input parameter	Output parameter
setOperatingMode	Changes the operating mode of the device. Setting this to "edit" will enable the "edit mode object" on RPC.	Mode: integer "0": Run mode "1": Edit mode	Empty string (compatible with classic XmlRPC client).

12.3.4 Device config object

activatePassword

Method Name	Description	Input parameter	Output parameter
activatePassword	Sets a password and activates it for the next edit session. Making this change persistently requires calling "save" on device config.	Password: string	Empty string (compatible with classic XmlRPC client).

save

Method Name	Description	Input parameter	Output parameter
save	Stores current configuration in persistent memory. If this is not called after changing device parameters (via setParameter), changes will get lost on reboot.	None.	Empty string (compatible with classic XmlRPC client).

Parameter of device config

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 applies only to 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 no application is active.
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

Parameter name	Data type	Description
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.
ServiceReportFailedBuffer	Int *) Has limits	Number of buffers reserved for failed results.
ServiceReportPassedBuffer	Int *) Has limits	Number of buffers reserved for passed results.
ExtrinsicCalibTransX	Double Unit: millimetres	Extrinsic calibration, transition in X direction.
ExtrinsicCalibTransY	Double Unit: millimetres	Extrinsic calibration, transition in Y direction.
ExtrinsicCalibTransZ	Double Unit: millimetres	Extrinsic calibration, transition in Z direction.
ExtrinsicCalibRotX	Double Unit: degrees	Extrinsic calibration, rotation around X axis.
ExtrinsicCalibRotY	Double Unit: degrees	Extrinsic calibration, rotation around Y axis.
ExtrinsicCalibRotZ	Double Unit: degrees	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 use cases. This information was extended to reflect all kinds of IP-address situations. Allowed values: "0": Static (IP address explicitly 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 the edit mode, but inside session).
DeviceType	String	readonly: Delivers a type description, unique by imager, evaluation logic and device interface.
ArticleNumber	String	readonly: Official catalogue number.
ArticleStatus	String	readonly: Official two-letter status code.
UpTime	Double	readonly: Hours since last reboot.
ImageTimestampReference	Int Unit: microseconds	readonly: This returns the current timestamp as a reference for the timestamps in the received images.
TemperatureFront1	Double Unit: celsius	Invalid temperature, the output is 3276.7.

Parameter name	Data type	Description
TemperatureFront2	Double Unit: celsius	Invalid temperature, the output is 3276.7.
TemperatureIllu	Double Unit: celsius	readonly: Temperature measured in the device. Measured on the illumination board.

*) Has limits: parameters with this marker are listed in the reply of getAllParameterLimits method.

Default values of device config parameters

Parameter name	Data type	Description
Name	String (utf8)	"New sensor"
Description	String (utf8)	""
ActiveApplication	Int *) Has limits	0
PcicTcpPort	Int	50010
PcicProtocolVersion	Int *) Has limits	3
IOLogicType	Int *) Has limits	1
IODebouncing	Bool	true
IOExternApplicationSwitch	Int *) Has limits	0
SessionTimeout	Int *) Has limits	30
ExtrinsicCalibTransX	Double Unit: millimetres	0.0
ExtrinsicCalibTransY	Double Unit: millimetres	0.0
ExtrinsicCalibTransZ	Double Unit: millimetres	0.0
ExtrinsicCalibRotX	Double Unit: degrees	0.0
ExtrinsicCalibRotY	Double Unit: degrees	0.0
ExtrinsicCalibRotZ	Double Unit: degrees	0.0
IPAddressConfig	Int	0
PasswordActivated	Bool	false
OperatingMode	Int	0
ServiceReportFailedBuffer	Int	15
ServiceReportPassedBuffer	Int	15

*) Has limits: parameters with this marker are listed in the reply of getAllParameterLimits method.

For all other device config 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 device config parameters

Parameter name	Minimum value	Maximum value
ActiveApplication	0	32

Parameter name	Minimum value	Maximum value
PcicProtocolVersion	1	4
IOLogicType	0	1
IOExternApplicationSwitch	0	3
SessionTimeout	5	300

12.3.5 Device / Network config object

saveAndActivateConfig

Method Name	Description	Input parameter	Output parameter
saveAndActivateConfig	Reinitialise the network interface so that it uses the configuration which was set by the other RPC methods. There will be no XMLRPC reply, because the network interface is instantly reset.	None.	Empty string (compatible with classic XmlRPC client).

Parameters of network config

Parameter name	Data type	Description
MACAddress	string (read-only)	MAC address of the device (format: "xx:xx:xx:xx:xx:xx").
UseDHCP	bool	Configure the IP interface via DHCP.
StaticIPv4Address	string	IPv4 address of the device (max. 15 characters). Only used if DHCP is disabled and if there is no temporary IP address set via discovery (format: "192.168.0.69").
StaticIPv4SubNetMask	string	IPv4 network mask of the device (max. 15 characters).
StaticIPv4Gateway	string	IPv4 gateway of the device (max. 15 characters).

Default values of network config parameters

Parameter name	Data type	Description
UseDHCP	bool	false
StaticIPv4Address	string	"192.168.0.69"
StaticIPv4SubNetMask	string	"255.255.255.0"
StaticIPv4Gateway	string	"192.168.0.201"

12.3.6 Device / Time config object

saveAndActivateConfig

Method Name	Description	Input parameter	Output parameter
saveAndActivateConfig	Save and immediately apply the current time configuration. This might lead to a jump in the system time.	None.	Empty string (compatible with classic XmlRPC client).

Parameters of time config

Parameter name	Data type	Description
NTPServers	string (UTF-8)	List of NTP servers which should be used for system time synchronization. The server entries must be separated with commas and whitespaces. Each server entry must be a numeric IPv4 address (e.g. "192.168.0.69").
WaitSyncTries	int	Number of attempts to initially synchronize with an NTP server. Each attempt waits 10 seconds for a server response.
SynchronizationActivated	bool	Enables synchronization of system time via NTP.
StartingSynchronization	bool (read-only)	Returns "true" during the initial NTP synchronization, which may result in a jump in the system time. Returns "false" afterwards.
Syncing	bool (read-only)	Returns "true" if the system clock is synchronized to an NTP server. This happens if at least one of the three most recent synchronization attempts were successful. At least 1 of the last 3 less significant bits in the "Reach" register must set to "1". Each measurement of reachability is performed every 64 s in standard conditions, but it may depend on current network traffic etc. (the pooling interval).
CurrentTime	int (read-only)	Returns the current system time (UTC, seconds since 1.1.1970)
Stats	string (read-only)	Returns a human readable string containing the synchronization state and accuracy for all NTP servers.
MaxNumberOfNTPServers	int (read-only)	Returns the maximum number of servers which can be added to the NTP client.

Default values of time config parameters

Parameter name	Data type	Description
NTPServers	string (UTF-8)	"
WaitSyncTries	int	2
SynchronizationActivated	bool	false

Minimum and maximum values of time config parameters

Parameter name	Minimum value	Maximum value
NTPServers	0	5
WaitSyncTries	1	60

12.4 Process interface

The process interface is used during the normal operation mode to get operational data from the device (e.g. 3D images, process values etc.). By default, the process interface uses port 50010. The port can be changed using the parameter "PcicTcpPort" of the "device config object" (→ Device config object □ 30).

12.4.1 Sending commands

For sending commands via the process interface the commands must be sent with a special protocol and as ASCII character strings. This protocol conforms to the version 3 of the O2V/O2D products.

Structure of the protocol

```
<Ticket><length>CR LF <Ticket><content>CR LF
```

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

Command	Description
<content>	It is the command to the device (e.g. trigger the unit).
<ticket>	It is a character string of 4 digits between 0-9. If a message with a specific ticket is sent to the device, it will reply with the same ticket.
<length>	It is a character string beginning with the letter 'L' followed by 9 digits. It indicates the length of the following data (<ticket><content>CR LF) in bytes.



- ▷ By using a ticket number the response can be safely assigned to a specific command. Otherwise, commands can only be sent one at a time while waiting for each response in between.
- ▷ A ticket number must be > 0999. Use a ticket number from the range 1000 - 9999.

Protocol versions

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 protocol version is "V3". It is recommended to use protocol version 3 for machine to machine communication. This is since only version 3 supports asynchronous messages and provides length information.

Ticket numbers for asynchronous messages

Ticket number	Description
0000	Asynchronous results
0001	Asynchronous error messages / codes
0010	Asynchronous notifications / message codes

Format of asynchronous notifications

The format of the asynchronous notifications is a combination of the unique message ID and a JSON formatted string containing the notification details: <unique message ID>:<JSON content>

Example for protocol version 3

```
<ticket=0010>L<length>CR LF<ticket=0010><unique message ID>:<JSON content>CR LF
```

Result

```
0010L000000045\r\n00100005000000:{"ID": 1034160761,"Index":1,"Name": "Pos 1"}\r\n
```

Explanation of the result

Command	Result
<ticket=0010>	0010
L<length>	L000000045
CR LF	\r\n
<ticket=0010>	0010
<unique message ID>	000500000
<JSON content>	{"ID": 1034160761,"Index":1,"Name": "Pos 1"}
CR LF	\r\n

Asynchronous message IDs

Asynchronous message ID	Description	Example	Description
000500000	Application changed	{"ID": 1034160761,"Index":1,"Name": "Pos 1","valid":true}	
000500001	Application is not valid	{"ID": 1034160761,"Index":1,"Name": "Pos 1","valid":false}	If a application exists on given index but it is invalid, the ID and Name are filled according to the application. If there is no application on given index, the application ID will contain 0 and the name an empty string "".
000500002	image acquisition finished	{}	This message signals the receiver, that the device has finished the image acquisition. This can be used for cascading multiple devices with a software trigger.

12.4.2 Receiving images

For receiving the image data, a TCP/IP socket communication is established. The default port number is 50010. The port number may differ based on the configuration. After opening the socket communication, the device needs to be configured by the c command (→ c command (upload PCIC output configuration) □ 46) to send data through this socket to the TCP/IP client (PC). For details on how to configure the PCIC output see Section (→ Configuration of PCIC output □ 41).

An exemplary configuration for the PCIC output per frame is given in the table.

Component	Content
Ticket and length information	T? command (→ T? command (execute synchronous trigger) □ 51)
Ticket	"0000"
Start sequence	String "star" (4 bytes)
Normalised amplitude image Output format: 16-bit unsigned integer	1 image
Distance image Output format: 16-bit unsigned integer. Unit: mm	1 image
X image Output format: 16-bit signed integer. Unit: mm	1 image

Component	Content
Y image Output format: 16-bit signed integer. Unit: mm	1 image
Z image Output format: 16-bit signed integer. Unit: mm	1 image
Confidence image Output format: 8-bit unsigned integer	1 image
Diagnostic data	
Stop sequence	String "stop" (4 bytes)
Ticket signature	<CR><LF>

12.4.3 Image data

For every image there will be a separate chunk. The chunk is part of the response frame data of the process interface.

The header of each chunk contains different kinds 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.

Offset	Name	Description	Size [byte]
0x0000	CHUNK_TYPE	Defines the type of the chunk. For each distinct chunk an own type is defined.	4
0x0004	CHUNK_SIZE	Size of the whole image chunk 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_HEIGHT	Image height in pixel	4
0x0018	PIXEL_FORMAT	Pixel format	4
0x001C	TIME_STAMP	Time stamp in microseconds (deprecated)	4
0x0020	FRAME_COUNT	Frame counter	4
0x0024	STATUS_CODE	Errors of the device	4
0x0028	TIME_STAMP_SEC	Time stamp in seconds	4
0x002C	TIME_STAMP_NSEC	Time stamp in nanoseconds	4
0x0030	PIXEL_DATA	The pixel data in the given type and dimension of the image. Padded to 4-byte boundary.	4

Available chunk types

Constant	Value	Description
RADIAL_DISTANCE_IMAGE	100	<p>Each pixel of the distance matrix denotes the ToF distance measured by the corresponding pixel or group of pixels of the imager. The distance value is corrected by the camera's calibration, excluding effects caused by multipath and multiple objects contributions (e.g. "flying pixels"). Reference point is the optical center of the camera inside the camera housing.</p> <p>Invalid PMD pixels (e.g. due to saturation) have a value of zero. Data type: 16-bit unsigned integer (little endian)</p> <p>Unit: millimeters</p>
NORM_AMPLITUDE_IMAGE	101	<p>Each pixel of the normalized amplitude image denotes the raw amplitude (see amplitude image below for further explanation), normalized to exposure time. Furthermore, vignetting effects are compensated. That means the darkening of pixels at the image border is corrected. The visual impression of this grayscale image is comparable to that of a common 2D camera.</p> <p>Invalid PMD pixels have an amplitude value of "0" (e.g. due to saturation).</p> <p>Data type: 16-bit unsigned integer</p>
AMPLITUDE_IMAGE	103	<p>Each pixel of the amplitude matrix denotes the amount of modulated light (i.e. the light from the camera's active illumination) which is reflected by the appropriate object. Higher values indicate higher PMD signal strengths and thus a lower amount of noise on the corresponding distance measurements. The amplitude value is directly derived from the PMD phase measurements without normalization to exposure time. In multiple exposure mode, the lack of normalization may lead (depending on the chosen exposure times) to inhomogeneous amplitude image impression, if a certain pixel is taken from the short exposure time and some of its neighbors are not.</p> <p>Invalid PMD pixels (e.g. due to saturation) have an amplitude value of 0.</p> <p>Data type: 16-bit unsigned integer</p>
GRAYSCALE_IMAGE	104	<p>Each pixel of the amplitude matrix denotes the amount of modulated light which is reflected by the appropriate object (i.e. the light from the camera's active illumination). Higher values indicate higher PMD signal strengths and thus a lower amount of noise on the corresponding distance measurements. The amplitude value is directly derived from the PMD phase measurements without normalization to exposure time.</p>

Constant	Value	Description
CARTESIAN_X_COMPONENT	200	The X matrix denotes the X component of the Cartesian coordinate of a PMD 3D measurement. The origin of the camera's coordinate system is in the middle of the lens' front glass, if the extrinsic parameters are all set to 0. Data type: 16-bit signed integer Unit: millimeters
CARTESIAN_Y_COMPONENT	201	The Y matrix denotes the Y component of the Cartesian coordinate of a PMD 3D measurement. The origin of the camera's coordinate system is in the middle of the lens' front glass, if the extrinsic parameters are all set to 0. Data type: 16-bit signed integer Unit: millimeters
CARTESIAN_Z_COMPONENT	202	The Z matrix denotes the Z component of the Cartesian coordinate of a PMD 3D measurement. The origin of the camera's coordinate system is in the middle of the lens' front glass, if the extrinsic parameters are all set to 0. Data type: 16-bit signed integer Unit: millimeters
CARTESIAN_ALL	203	CARTESIAN_X_COMPONENT, CARTESIAN_Y_COMPONENT, CARTESIAN_Z_COMPONENT
UNIT_VECTOR_ALL	223	The unit vector matrix contains 3 values [ex, ey, ez] for each PMD pixel, i.e. the data layout is [ex_1,ey_1,ez_1, ... ex_N, ey_N, ez_N], where N is the number of PMD pixels. Data type: 32-bit floating point number (3x per pixel)
CONFIDENCE_IMAGE	300	(Additional information for CONFIDENCE_IMAGE)
DIAGNOSTIC	302	(Receiving images)
JSON_DIAGNOSTIC	305	Items with JSON formatted diagnostic data is formatted like this: <pre>{ "AcquisitionDuration": 20.391, "EvaluationDuration": 37.728, "FrameDuration": 37.728, "FrameRate": 15.202, "TemperatureIllu": 52.9 }</pre> Unit for durations: milliseconds Unit for framerates: Hz Unit for temperature: °C
EXTRINSIC_CALIB	400	The transformation from one cartesian coordinate system to another is defined by 6 degrees of freedom vector (DOF): [trans_x, trans_y, trans_z, rot_x, rot_y, rot_z]. Let R be the product of the common "clockwise" 3D-rotation matrices: $R = R_x * R_y * R_z$ Data type: 32-bit floating point number (little endian) Unit for trans_x, trans_y, trans_z: millimetres Unit for rot_x, rot_y, rot_z: °
JSON_MODEL	500	Model data in JSON

Constant	Value	Description
MODEL_ROIMASK	501	ROI mask for internal debugging purposes
SNAPSHOT_IMAGE	600	Snapshot image
OCCUPANCY_MAP	602	Occupancy grid mapping representing probabilities of obstacles in grid cells. By default, the grid ranges from -5m to 5m in x- and y-direction in vehicle coordinates with grid dimensions 200x200. First element corresponds to [x=-5m, y=-5m] grid cell, element 199 to [x=-5m, y=+5m], and the last element to [x=+5m, y=+5m]. Data type: 8-bit unsigned integer, 0 representing probability zero, 255 representing probability 1.

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	64-bit floating point number
Reserved	9	N/A
FORMAT_32F_3	10	Vector with 3x32-bit floating point number

12.4.4 Additional information for CONFIDENCE IMAGE

Further information for the confidence image.

Bit	Value	Description
0	1 = pixel invalid	Pixel invalid The pixel is invalid. To determine whether a pixel is valid or not only this bit needs to be checked. The reason why the bit is invalid is recorded in the other confidence bits.
1	1 = pixel saturated	Pixel is saturated Contributes to pixel validity: yes
2	1 = bad A-B symmetry	A-B pixel symmetry 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

Bit	Value	Description
3	1 = amplitude below minimum amplitude threshold	Amplitude limits The amplitude value is below minimum amplitude threshold. Contributes to pixel validity: yes
4+5	Bit 5, bit 4 0 0 = unused 1 = shortest exposure time (only used in 3 exposure mode) 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 = pixel is clipped	Clipping box on 3D data If clipping is active this bit indicates that the pixel coordinates are outside the defined volume. Contributes to pixel validity: yes
7	1 = suspect/defective pixel	Suspect pixel This pixel has been marked as "suspect" or "defective" and values have been replaced by interpolated values from the surroundings. Contributes to pixel validity: no

12.4.5 Configuration of PCIC output

The user has the possibility to define his own PCIC output. This configuration is only valid for the current PCIC connection. It does not affect any other connection and will get lost after disconnecting.

For configuring the PCIC output a "flexible" layouter concept is used, represented by a JSON string. The format of the default configuration is as follows:

```
{
  "layouter": "flexible",
  "format": { "dataencoding": "ascii" },
  "elements": [
    { "type": "string", "value": "star", "id": "start_string" },
    { "type": "blob", "id": "normalized_amplitude_image" },
    { "type": "blob", "id": "x_image" },
    { "type": "blob", "id": "y_image" },
    { "type": "blob", "id": "z_image" },
    { "type": "blob", "id": "confidence_image" },
    { "type": "blob", "id": "diagnostic_data" },
    { "type": "string", "value": "stop", "id": "end_string" }
  ]
}
```

This string can be retrieved by the C? command, altered and sent back using the c command. The layout software has the following main object properties:

Name	Description	Details
layouter	Defines the basic data output format. So far only "flexible" is supported	Type: string
format	Defines format details, the definitions in the main object are the defaults for any of the following data elements (e.g. if it says dataencoding=binary, all data elements should be binary encoded instead of ASCII).	Type: object

Name	Description	Details
elements	List of data elements which must be written.	Type: array of objects

Actual data

The actual data is defined within the “elements” properties and may consist of these settings:

Name	Description	Details
type	Defines the type of data which must be written. The data might be stored in a different type (e.g. stored as integer but should be output as Float32) The type "records" will need some special handling.	Type: string
id	Defines an identifier for this data element. If there is no fixed value (property "value"), the data should be retrieved via id.	Type: string
value	Optional property for defining a fixed output value.	Type: any JSON value
format	Type-dependent option for fine-tuning the output format. E.g. cut an integer to less than 4 bytes.	Type: object

Available values for the type property

Type	Description
records	Defines that this element represents a list of records. If type is set to "records", there must be an "elements" property. The "elements" property defines which data should be written per record.
string	Data is written as string. Most of the time this will be used with "value" property to write fixed start, end or delimiter text. Text encoding should be UTF8 if there is nothing else specified in format properties.
float32	Data is written as floating point number. This has a lot of formatting options (at least with "flexible" layout software) See following section about format properties.
uint32	Data is written as integer. This has a lot of formatting options (at least with "flexible" layout software) See following section about format properties.
int32	Data is written as integer. This has a lot of formatting options (at least with "flexible" layout software) See following section about format properties.
uint16	Limits the output to two bytes in binary encoding, besides the binary limitation it acts like uint32.
int16	Limits the output to two bytes in binary encoding, besides the binary limitation it acts like int32.
uint8	Limits the output to one byte in binary encoding, besides the binary limitation it acts like uint32.

Type	Description
int8	Limits the output to one byte in binary encoding, besides the binary limitation it acts like int32.
blob	Data is written as a BLOB (byte by byte as if it came from the data provider). (Binary Large Object)

Depending on the desired data format the user may tune his output data with further "format" properties.

Common format properties

Format properties	Allowed values	Default
dataencoding	"ascii" or "binary" can be defined in top-level-object and overwritten by element objects.	"ascii"
scale	"float value with decimal separator" to scale the results for output byte width	1.0
offset	"float value with decimal separator"	0.0

Binary format properties

Format properties	Allowed values	Default
order	Little, big and network	Little

ASCII format properties

Format properties	Allowed values	Default
width	Output width. If the resulting value exceeds the width field the result will not be truncated.	0
fill	Fill character	" "
precision	Precision is the number of digits behind the decimalseparator.	6
displayformat	Fixed, scientific	Fixed
alignment	Left, right	Right
decimalseparator	7-bit characters for e.g. "."	."
base	Defines if the output should be: binary (2) octal (8) decimal (10) hexadecimal (16)	10

Example of a format configuration of the temperature element (id: temp_illu)

Illumination temperature like this "33,5 ___":

```
c00000226{ "layouter": "flexible", "format": { "dataencoding": "ascii" },
"elements": [ { "type": "float32", "id": "temp_illu", "format": { "width": 7,
"precision": 1, "fill": "_", "alignment": "left", "decimalseparator":
", " } } ] }
```

Illumination temperature as binary (16-bit integer, 1/10 °C):

```
c000000194{ "layouter": "flexible", "format": { "dataencoding": "ascii"},
"elements": [ { "type": "int16", "id": "temp_illu", "format": {"dataencoding":
"binary", "order": "network", "scale": 10 } } ] }
```

Illumination temperature in °F (e.g. "92.3 Fahrenheit"):

```
c000000227{ "layouter": "flexible", "format": { "dataencoding": "ascii" },
"elements": [ { "type": "float32", "id": "temp_illu", "format":
{ "precision":1, "scale": 1.8, "offset": 32 } }, { "type": "string", "value": "
Fahrenheit"} ] }
```

Available element IDs

ID	Description	Native data type
all_cartesian_vector_matrices	All Cartesian images (X+Y+Z) concatenated to one package	16-bit signed integer
all_unit_vector_matrices	Matrix of unit vectors. Each element consists of a 3-component vector [e_x, e_y, e_z]	Float32
amplitude_image	PMD raw amplitude image	16-bit unsigned integer
confidence_image	Confidence image	8-bit unsigned integer
distance_image	Radial distance image	16-bit unsigned integer unit: millimeters
occupancy_map	Values of occupancy map are representing probabilities of occurring obstacles in grid cells.	8-bit unsigned integer
evaltime	Evaluation time for current frame in milliseconds	32-bit unsigned integer
extrinsic_calibration	Extrinsic calibration, consisting of 3 translation parameters (unit: millimeters) and 3 angles (unit: degree): [t_x, t_y, t_z, alpha_x, alpha_y, alpha_z]	Float32
framerate	Current frame rate in Hz	Float32
normalized_amplitude_image	Normalized amplitude image	16-bit unsigned integer
temp_front1	Invalid temperature, the output is 3276.7	Float32, unit: °C
temp_illu	Temperature measured in the device while capturing this result Measured on the illumination board	Float32, unit: °C
x_image y_image z_image	Cartesian coordinates for each pixel Each dimension is a separate image	16-bit signed integer

With the f command changeable IDs

ID	Name	Description	Values
0000010000	OdsEgoData	ODS ego data (→ Detect objects □ 20)	Binary structure (→ Sending ego data to the device □ 52)
0000010001	ZoneConfiguration	configuration data used to define new observation zones of the ODS algorithm (→ Object warning zones □ 17)	Binary structure (→ Sending and retrieving zone configuration to the device □ 54)

ID	Name	Description	Values
0000010002	SensingState	switch the camera's sensing state to on or off (→ Detect objects □ 20)	0: off 1: on

12.5 Process interface command reference



▷ All received messages which are sent because of the following commands will be sent without "start"/"stop" at the beginning or ending of the string.

GB

12.5.1 a command (activate application)

Command	Description	Type	Reply			Note
a<application number>	Activates the selected application	Action	*	!	?	<application number> 2 digits for the application number as decimal value.
				The command is not supported. The error code 100000006 is propagated.	Invalid command length.	

12.5.2 A? command (occupancy of application list)

Command	Description	Type	Reply			Note
A?	Requests the occupancy of the application list.	Request	<amount><t><n umber active application><t> ... <number><t><n umber>	!	?	<amount> char string with 3 digits for the amount of applications saved on the device as decimal number. <t> tabulator (0x09). <number active application> 2 digits for the active application. <number> 2 digits for the application number.
				The command is not supported. The error code 100000006 is propagated.	Invalid command length.	The active application is repeated within the application list.

12.5.3 c command (upload PCIC output configuration)

Command	Description	Type	Reply			Note
c<length><configuration>	Uploads a PCIC output configuration lasting this session.	Action	*	!	?	<length> 9 digits as decimal value for the data length. <configuration> configuration data.
				Error in configuration. Wrong data length.	Invalid command length.	

12.5.4 C? command (retrieve current PCIC configuration)

Command	Description	Type	Reply		Note
C?	Retrieves the current PCIC configuration.	Request	<length><configuration>	?	<length> 8 digits as decimal value for the data length <configuration> configuration data
				Invalid command length.	

12.5.5 E? command (request current error state)

Command	Description	Type	Reply			Note
E?	Requests the current error state	Request	<code>	!	?	<code> Error code with 8 digits as a decimal value. It contains leading zeros.
				Invalid state (e.g. configuration mode).	Invalid command length.	

12.5.6 G? command (request device information)

Command	Description	Type	Reply	Note
G?	Requests device information	Request	<pre><vendor><t><article number><t> <name><t><location><t><description><t> ><ip> <subnet mask><t><gateway><t><MAC><t><DHC P><t><portnumber></pre>	<pre><vendor> IFM ELECTRONIC <t> Tabulator (0x09) <article number> e.g. O3D300 <name> UTF8 Unicode string <location> UTF8 Unicode string <description> UTF8 Unicode string <ip> IP address of the device as ASCII character sting, e.g. 192.168.0.96. <port number> port number of the XML-RPC. <subnet mask> subnet mask of the device as ASCII, e.g. 192.168.0.96 <gateway> gateway of the device as ASCII, e.g 192.168.0.96 <MAC> MAC adress of the device as ASCII, e.g. AA:AA:AA:AA:AA:AA <DHCP> ASCII string "0" for off and "1" for on.</pre>

GB

12.5.7 H? command (return a list of available commands)

Command	Description	Type	Reply
H?	Returns a list of available commands	Request	<pre>H? (show this list) T (execute Trigger) T? (execute Trigger and wait for data) o<io-id><io-state> (sets IO state) O<io-id>? (get IO state) I<image-id>? (get last image of defined type) A? (get application list) p<state> (activate / deactivate data output) a<application number> (set active application) E? (get last error) V? (get current protocol version) v<version> (sets protocol version) c<length of configuration file><configuration file> (configures process date formatting) C? (show current configuration) G? (show device information) S? (show statistics)</pre>

12.5.8 I? command (request last image taken)

Command	Description	Type	Reply			Note
I<image-ID>?	Request last image taken	Request	<length><image data>	!	?	<image-ID> 2 digits for the image type. <length> char string with exactly 9 digits as decimal number for the image data size in bytes. <image data> image data.
				The command is not supported. The error code 100000006 is propagated.	Invalid command length.	Valid image ID: 01 (amplitude image) 02 (normalised amplitude image) 03 (distance image) 04 (X image, distance information) 05 (Y image, distance information) 06 (Z image, distance information) 07 (confidence image, status information) 08 (extrinsic calibration) 09 (unit_vector_matrix_ex, ey, ez) 10 (last result output as formatted for this connection) 11 (all distance images: X, Y, and Z)

12.5.9 o command (set logic state of a ID)

Command	Description	Type	Reply			Note
o<IO-ID><IO-state>	Sets the logic state of a specific ID.	Action	*	!	?	<IO-ID> 2 digits for digital output: "01" IO1 "02" IO2 "03" IO3 <IO-state> 1 digit for the state: "0" logic state low. "1" logic state high.

Command	Description	Type	Reply			Note
				The command is not supported. The error code 100000006 is propagated.	Invalid command length.	

12.5.10 O? command (request state of a ID)

Command	Description	Type	Reply			Note
O<IO-ID>?	Requests the state of a specific ID.	Request	<IO-ID><IO-state>	!	?	<IO-ID> 2 digits for digital output: "01" IO1 "02" IO2 "03" IO3 <IO-state> 1 digit for the state: "0" logic state low. "1" logic state high.
				The command is not supported. The error code 100000006 is propagated.	Invalid command length.	

12.5.11 p command (turn PCIC output on or off)

Command	Description	Type	Reply			Note
p<state>	Turns the PCIC output on or off	Action	*	!	?	<state> 1 digit 0 (deactivates all asynchronous output) 1 (activates asynchronous result output) 2 (activates asynchronous error output) 3 (activates asynchronous error and data output) 4 (activates asynchronous notifications) 5 (activates asynchronous notifications and asynchronous result) 6 (activates asynchronous notifications and asynchronous error output) 7 (activates all outputs)

Command	Description	Type	Reply		Note
				The command is not supported. The error code 100000006 is propagated.	Invalid command length. On device restart the value configured within the application is essential for the output of data. This command can be executed in any device state. By default, the error codes will not be provided by the device.

12.5.12 S? command (request current decoding statistics)

Command	Description	Type	Reply		Note
S?	Requests current decoding statistics	Request	<number of results><t><number of positive decodings><t><number of false decodings>	!	<t> tabulator (0x09) <number of results> Images taken since application start. 10 digits decimal value with leading "0". <number of positive decodings> Number of decodings leading to a positive result. 10 digits decimal value with leading "0". <number of false decodings> Number of decodings leading to a negative result. 10 digits decimal value with leading "0".
				The command is not supported. The error code 100000006 is propagated.	

12.5.13 t command (execute asynchronous trigger)

Command	Description	Type	Reply	
t	Executes trigger. The result data is sent asynchronously.	Action	*	!
			Trigger was executed, the device captures an image and evaluates the result.	The command is not supported. The error code 100000006 is propagated.

12.5.14 T? command (execute synchronous trigger)

Command	Description	Type	Reply		Note
T?	Executes trigger. The result data is send synchronously	Request	Process data within the configured layout.	!	Result data can be sent via Ethernet/IP, PROFINET or TCP/IP.
			Trigger was executed, the device captures an image, evaluates the result and sends the process data.	The command is not supported. The error code 100000006 is propagated.	



12.5.15 v command (set current protocol version)

Command	Description	Type	Reply			Note
v<version>	Sets the current protocol version. The device configuration is not affected.	Action	*	!	?	<version> 2 digits for the protocol version.
				Invalid version.	Invalid command length.	Only protocol version 3 is supported (→ Sending commands <code>␣ 35</code>).



▷ The default protocol version is „V3“.

12.5.16 V? command (request current protocol version)

Command	Description	Type	Reply	Note
V?	Requests current protocol version.	Request	<current version><empty><min version><empty><max version>	<current version> 2 digits for the currently set version. <empty> space sign: 0x20 <min/max version> 2 digits for the available min and max version that can be set.

12.6 Process interface commands ODS specific

12.6.1 Sending ego data to the device

The "f-command" in combination with the fixed parameter identification number "10000" is used to send the vehicle's ego data to the device.

Command	Description	Type	Reply			Note
f<Parameter><reserved><ego data>	Set current ego data	Action	<current process data>	!	?	<Parameter> Id of parameter to be set. Fixed 5 bytes decimal ascii: "10000". <reserved> Fixed "#00001". <egodata > ODS ego data structure (see below). <current process data> Latest ODS result structure (see below).
				Invalid parameter ID or syntax error.	Invalid command length.	

ODS ego data structure

Name	Type	Note
EgoDataLength	uint32	Length of the following data: 20 bytes.
VelocityX	float32	Velocity in x direction [m/s].
VelocityY	float32	Velocity in y direction [m/s].
GearRate	float32	Gear rate [radian/s].
TimeStamp	uint64	Global NTP time stamp of the acquisition of the image data in nano seconds, expired after the Unix epoch.

ODS result structure

Name	Type	Note
ResultLength	uint32	Length of the following data.
CameraStatus	uint32	Current camera state: "0": IDLE state "1": SENSING state "3": ERROR state
CurrentError	uint32	Error code of the last occurred error (see table below). "0": no error
TimeStamp	uint64	Global NTP time stamp of the acquisition of the image data in nanoseconds, expired after the Unix epoch.
ResultZoneConfigID	uint32	Identification tag for last set zone.

Name	Type	Note
ResultZoneOccupancyState	uint32	Each bit defines the current state of the search zone (32-bit value): "0": free "1": occupied Bit 31 defines if the current result is valid or not. Example for zone 1 and 3 are occupied in sensing state and no errors: "100000000000000000000000000000101"



CurrentError

The "CurrentError" depends on the sensor state "CameraStatus". In case of the sensor being in ERROR state, the "CurrentError" corresponds to the sensor errors (→ Error codes □ 57). Suggested reactions to the errors are mentioned there.

While the sensor is in SENSING or IDLE state the "CurrentError" is used to reflect potential errors occurring in relation to the ODS algorithm. The algorithm errors are bitcoded. A non-zero-bit value signals an error as listed in the following table.

Bit	Description	Solution
0	No ego data.	Check ego data.
1	Ego data range check failed.	Check ego data.
2	Ego data unexpected timestamps.	Check synchronization with the NTP server on device and master controller.
3	Phase data unexpected timestamps.	Check synchronization with the NTP server on device and master controller.
4	Internal error 1.	Reboot the device. Contact support if it happens several times.
5	Internal error 2.	Reboot the device. Contact support if it happens several times.
6	Too many ego data packets received, list truncated (distance image).	Check frame rate of ego data.
16	Internal error 3.	Reboot the device. Contact support if it happens several times.
17	Too many ego data packets received, list truncated (object detection).	Check frame rate of ego data.
18	Default zone used (no zone set).	Set zone if desired.
19	Default extrinsic calibration used (not calibrated).	Calibrate the device.
20	Invalid zone configuration.	Check zone configuration.

Tab. 1: Possible errors

Example for sending ego data

Command for sending ego data:

```
9876L00000042\r
\n9876f10000#00001<EgoDataLength><VelocityX><VelocityY><GearRate><TimeStamp>\r
\n
```

Code	Explanation
9876	Ticket number [1000..9999] in ASCII.
L00000042	Length of the message after "\r\n" [Byte].

Code	Explanation
\r\n	Carriage return and Line feed in ASCII.
9876	Ticket number [1000..9999] in ASCII.
f1000#00001	Command in ASCII.
<EgoDataLength>	Ego data length value as binary UINT32 in little endian.
<VelocityX>	Velocity X value as binary FLOAT32 in little endian.
<VelocityY>	Velocity Y value as binary FLOAT32 in little endian.
<GearRate>	Gear rate value as binary FLOAT32 in little endian.
<TimeStamp>	Time stamp value as binary UINT64 in little endian.
\r\n	Carriage Return + Line Feed in ASCII.

Tab. 2: Explanation of code example

Command response:

```
9876L000000034\r
\n9876<ResultLength><CameraStatus><CurrentError><TimeStamp><ResultZoneConfigID>
<ResultZoneOccupancyState>\r\n
```

Code	Explanation
9876	Ticket number of the corresponding command.
L000000034	Length of the message after " \r\n " [Byte].
\r\n	Carriage return and Line feed in ASCII.
9876	Ticket number [1000..9999] in ASCII.
<ResultLength>	Result length as binary UINT32 in little endian.
<CameraStatus>	Camera status as binary UINT32 in little endian.
<CurrentError>	Current error as binary UINT32 in little endian.
<TimeStamp>	Time stamp as binary UINT64 in little endian.
<ResultZoneConfigID>	ResultZoneConfigID as binary UINT32 in little endian.
<ResultZoneOccupancyState>	ResultZoneOccupancyState as binary UINT32 in little endian.
\r\n	Carriage return and Line feed in ASCII.

Tab. 3: Explanation of code example

12.6.2 Sending and retrieving zone configuration to the device

The "f command" in combination with the fixed parameter identification number "10001" is used to set the zone configuration for the ODS application.

Command	Description	Type	Reply			Note
f<Parameter-ID> <reserved><zoneconfiguration>	Set the zone configuration to be applied.	Action	*	!	?	<Parameter-ID> Id of parameter to be set. Fixed 5 bytes decimal ascii: "10001". <reserved> fixed "#00001" <zoneconfiguration> ODS zone configuration structure (see below).

Command	Description	Type	Reply			Note
			Parameter successfully set	Invalid parameter ID or syntax error.	Invalid command length.	

ODS zone configuration structure

Name	Type	Note
ZoneConfigurationLength	uint32	Length of the following data.
zoneConfigID	uint32	Identification tag for the current zone. Valid values: "1-255". If there is no zone set by the user the value will be "0".
GlobalZoneHeight	float32	Max. Height [m].
Zone1Coordinates	12x float32	6 points in vehicle coordinate system [m]. [x1,y1;x2,y2;x3,y3;x4,y4;x5,y5;x6,y6]
Zone2Coordinates	12x float32	6 points in vehicle coordinate system [m]. [x1,y1;x2,y2;x3,y3;x4,y4;x5,y5;x6,y6]
Zone3Coordinates	12x float32	6 points in vehicle coordinate system [m]. [x1,y1;x2,y2;x3,y3;x4,y4;x5,y5;x6,y6]

The "F command" in combination with the fixed parameter identification number "10001" is used to retrieve the current zone configuration.

Command	Description	Type	Reply	Note
F<Parameter-ID>?	Get the current valid zone configuration.	Request	<zoneconfigurationdata >	<Parameter-ID> Id of parameter. Fixed 5 bytes decimal ASCII: "10001".
			ODS zone configuration structure (see above).	



▷ The command for sending the zone configuration is build up like the one for sending ego data (→ Sending ego data to the device □ 52).

12.6.3 Switching the sensing state

The "f command" in combination with the fixed parameter identification number "10002" is used to switch between IDLE and SENSING state.

Command	Description	Type	Reply			Note
f<Parameter-ID> <reserved><sensingstate>	Set the sensing state.	Action	*	!	?	<Parameter-ID> Id of parameter to be set. Fixed 5 bytes decimal ascii: "10002". <reserved> fixed "#00001" <sensingstate> Fixed 5 bytes signed decimal ascii padded with "0" and sign. "+00001": sensing state on (SENSING). "+00000": sensing state off (IDLE).
			Parameter successfully set	Invalid parameter ID or syntax error.	Invalid command length.	

Example for switching the SENSING state

Command for switching the device to SENSING state:

```
1234L000000024\r\n1234f10002#00001+00001\r\n
```

Command for switching the device to IDLE state:

```
1234L000000024\r\n1234f10002#00001+00000\r\n
```

Code	Explanation
1234	Ticket number [1000..9999] in ASCII.
L000000024	Length of the message after "\r\n" [Byte].
\r\n	Carriage return and Line feed in ASCII.
1234	Ticket number [1000..9999] in ASCII.
f10002#00001	Command in ASCII.
+00001	Command parameter in ASCII: +00000 or +00001.
\r\n	Carriage return and Line feed in ASCII.

Tab. 4: Explanation of code example

Command response after the status is successfully set:

```
1234L000000007\r\n1234*\r\n
```

1234	Ticket number [1000..9999] in ASCII.
L000000007	Length of the message after "\r\n" [Byte].
\r\n	Carriage return and Line feed in ASCII.
1234	Ticket number [1000..9999] in ASCII.
*	Command response in ASCII.
\r\n	Carriage return and Line feed in ASCII.

Tab. 5: Explanation of code example

12.7 Error codes

The p-command provides the error codes of the device. By default, the error codes are not provided.

Error code ID	Description	Solution
10000001	Maximum number of connections exceeded.	Close other opened connections (e.g. ifm Vision Assistant).
110001001	Boot timeout.	Reboot, call support if it happens several times.
110001002	Fatal software error.	Reboot, call support if it happens several times.
110001003	Unknown hardware.	Reboot, call support if it happens several times.
110001006	Trigger overrun.	Reboot, call support if it happens several times.
110002000	Short circuit on Ready for Trigger.	Check wiring.
110002001	Short circuit on OUT1.	Check wiring.
110002002	Short circuit on OUT2.	Check wiring.
110002003	Reverse feeding.	Check wiring.
110003000	Vled overvoltage.	Check the supply voltage.
110003001	Vled undervoltage.	Check the supply voltage.
110003002	Vmod overvoltage.	Check the supply voltage.
110003003	Vmod undervoltage.	Check the supply voltage.
110003004	Mainboard overvoltage.	Check the supply voltage.
110003005	Mainboard undervoltage.	Check the supply voltage.
110003006	Supply overvoltage.	Check the supply voltage.
110003007	Supply undervoltage.	Check the supply voltage.
110003008	VFEMon alarm.	Reboot, call support if it happens several times.
110003009	PMIC supply alarm.	Reboot, call support if it happens several times.
110004000	Illumination overtemperature.	Provide external cooling.
120000001	NTP server not available.	Check availability of NTP server and general network settings (e.g. Firewalls).
120000002	other NTP error.	Check availability of NTP server and general network settings (e.g. Firewalls).
130000001	ego data invalid.	Check ego data (→ Detect objects □ 20).
130000002	zone data invalid.	Check zone configuration (→ Set parameters of the device □ 16).
130000003	given sensor state is invalid.	Provide allowed values (→ Detect objects □ 20)

Tab. 6: Error codes

List of Figures

Fig. 1	Installed device with a detected object	10
Fig. 2	Install device with spring washers	10
Fig. 3	Wiring example	14
Fig. 4	Wiring example for several devices	14
Fig. 5	Indicators of the device	15
Fig. 6	Extrinsic calibration of the device	16
Fig. 7	Polygon with 6 corner points	17
Fig. 8	Device with object warning zones	18
Fig. 9	Object moves out of the detection range	18
Fig. 10	Object is outside the detection range	18
Fig. 11	Occupancy map with the grid of 200x200 cells	19
Fig. 12	Distance image (left), amplitude image (centre) and confidence image (right)	19
Fig. 13	Vehicle's own movement	21
Fig. 14	XML-RPC Objects 1	25
Fig. 15	XML-RPC Objects 2	25

List of tables

Tab. 1	Possible errors.....	53
Tab. 2	Explanation of code example.....	53
Tab. 3	Explanation of code example.....	54
Tab. 4	Explanation of code example.....	56
Tab. 5	Explanation of code example.....	56
Tab. 6	Error codes	57



Glossary

FTF (AGV)

An Automated Guided Vehicle (AGV) is an in-floor transport system with its own drive. The AGV is automatically controlled and guided without contact.

ODS

ODS stands for "Obstacle detection system".