

SmartRunner Explorer

Light section sensor for
high-precision profile
capture

Manual



With regard to the supply of products, the current issue of the following document is applicable:
The General Terms of Delivery for Products and Services of the Electrical Industry, published
by the Central Association of the Electrical Industry (Zentralverband Elektrotechnik und Elek-
troindustrie (ZVEI) e.V.) in its most recent version as well as the supplementary clause:
"Expanded reservation of proprietorship"

Worldwide

Pepperl+Fuchs Group
Lilienthalstr. 200
68307 Mannheim
Germany
Phone: +49 621 776 - 0
E-mail: info@de.pepperl-fuchs.com

North American Headquarters

Pepperl+Fuchs Inc.
1600 Enterprise Parkway
Twinsburg, Ohio 44087
USA
Phone: +1 330 425-3555
E-mail: sales@us.pepperl-fuchs.com

Asia Headquarters

Pepperl+Fuchs Pte. Ltd.
P+F Building
18 Ayer Rajah Crescent
Singapore 139942
Phone: +65 6779-9091
E-mail: sales@sg.pepperl-fuchs.com
<https://www.pepperl-fuchs.com>

1	Introduction	5
1.1	Content of this Document.....	5
1.2	Target Group, Personnel	5
1.3	Symbols Used	6
2	Product Specifications	7
2.1	Use and Application	7
2.2	Hazards of Laser Radiation	7
2.3	Dimensions.....	8
2.4	Displays and Operating Elements.....	9
2.5	Interfaces and Connections	11
2.6	Network Interface	12
2.7	Software Interface.....	13
2.8	Accessories.....	13
2.8.1	Voltage Supply.....	13
2.8.2	Network Cable	14
3	Installation	15
3.1	Storage and Disposal	15
3.2	Preparation	15
3.3	Detection range.....	16
3.4	Mounting the Sensor	16
3.5	Connecting the Sensor	18
3.6	Setting up Windows Network Communication Between the Sensor and a PC/Laptop19	
4	Commissioning	22
5	Configuration	23
5.1	VsxProtocolDriver.....	23
5.2	Configuration Parameters.....	37
6	Vision Configurator Software	43
6.1	Connecting to Vision Configurator	43
6.2	Application Window Structure.....	45

6.3	Menu Bar	47
6.3.1	File Menu	47
6.3.2	View Menu	47
6.3.3	Sensor Menu	48
6.3.4	Image Menu	48
6.3.5	Administration Menu	49
6.3.6	Help Menu	49
6.4	Toolbar	49
6.5	Device Data	50
6.6	Image and Line Display	51
6.7	Configuration window	55
6.7.1	Sensor Information Tab	55
6.7.2	Common Tab	56
6.7.3	Explorer Tab	57
7	Maintenance and Repair	59
7.1	Servicing	59
7.2	Repair	59
8	Troubleshooting	60
8.1	What to Do in Case of a Fault	60

1 Introduction

1.1 Content of this Document

This document contains information required to use the product in the relevant phases of the product life cycle. This may include information on the following:

- Product identification
- Delivery, transport, and storage
- Mounting and installation
- Commissioning and operation
- Maintenance and repair
- Troubleshooting
- Dismounting
- Disposal



Note

For full information on the product, refer to the further documentation on the Internet at www.pepperl-fuchs.com.



Note

For specific device information such as the year of construction, scan the QR code on the device. As an alternative, enter the serial number in the serial number search at www.pepperl-fuchs.com.

The documentation comprises the following parts:

- This document
- Datasheet

In addition, the documentation may comprise the following parts, if applicable:

- EU-type examination certificate
- EU declaration of conformity
- Attestation of conformity
- Certificates
- Control drawings
- Instruction manual
- Functional safety manual
- Other documents

1.2 Target Group, Personnel

Responsibility for planning, assembly, commissioning, operation, maintenance, and dismantling lies with the plant operator.

Only appropriately trained and qualified personnel may carry out mounting, installation, commissioning, operation, maintenance, and dismantling of the product. The personnel must have read and understood the instruction manual and the further documentation.

Prior to using the product make yourself familiar with it. Read the document carefully.

1.3 Symbols Used

This document contains symbols for the identification of warning messages and of informative messages.

Warning Messages

You will find warning messages, whenever dangers may arise from your actions. It is mandatory that you observe these warning messages for your personal safety and in order to avoid property damage.

Depending on the risk level, the warning messages are displayed in descending order as follows:



Danger!

This symbol indicates an imminent danger.

Non-observance will result in personal injury or death.



Warning!

This symbol indicates a possible fault or danger.

Non-observance may cause personal injury or serious property damage.



Caution!

This symbol indicates a possible fault.

Non-observance could interrupt the device and any connected systems and plants, or result in their complete failure.

Informative Symbols



Note

This symbol brings important information to your attention.



Action

1. This symbol indicates a paragraph with instructions. You are prompted to perform an action or a sequence of actions.

2 Product Specifications

2.1 Use and Application

This manual applies to SmartRunner Explorer light section sensors (hereinafter referred to as "Sensors"). The sensor is based on SmartRunner technology and combines the light section method for detecting height profiles with a 2-D vision sensor. Based on this unique technology, the sensor is able to output both the height profile in global coordinates and 2-D flat images.

A transmitter optic is used to project a laser line onto an object as part of the light section method. This is detected by a camera at a specific angle. A height profile is determined using the triangulation principle. This can be output via the Ethernet-TCP/IP interface and conveniently integrated into the relevant program via the programming interface supplied.

SmartRunner Explorer consequently allows the flexible implementation of numerous applications.

Design of the Sensor

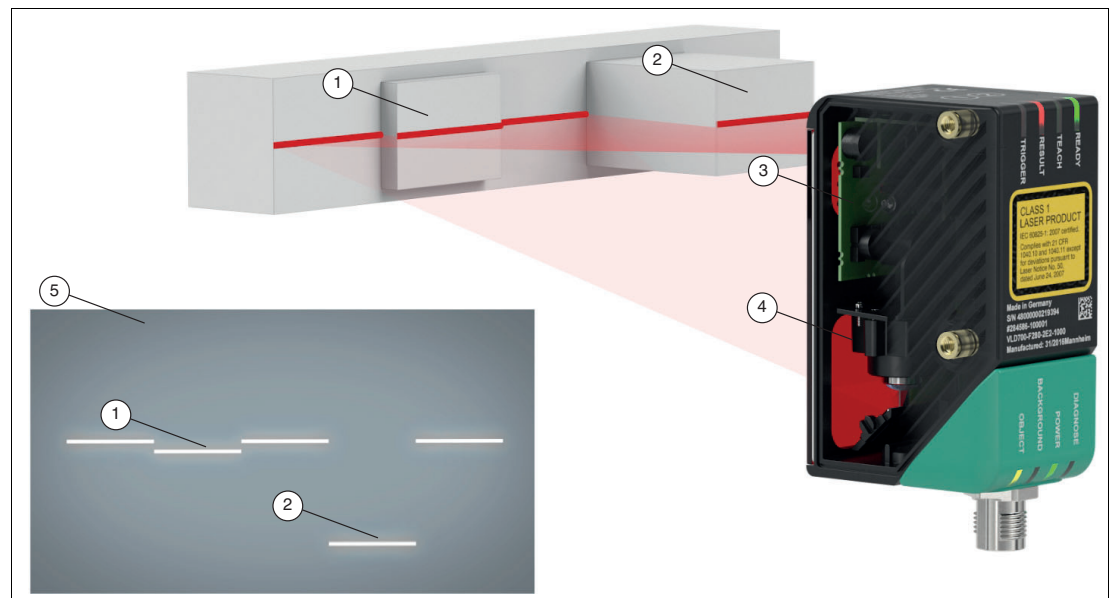


Figure 2.1 Overview of components and measurement result

- 1 Flat profile
- 2 Raised profile
- 3 Transmitter unit (laser for height profile and LEDs for surface lighting)
- 4 Camera
- 5 Height profile on the image sensor (measuring result)

2.2 Hazards of Laser Radiation

This section describes the contents and location of the warning label.

The sensor used corresponds to the safety standard IEC 60825-1:2014 for a laser class 1 product. In addition, the US regulation 21 CFR 1040.10 and 1040.11 is fulfilled except for **Laser Notice No. 56** dated May 8, 2019.



Warning!

Class 1 laser light

The laser light can be an irritant, especially in a dark environment. Do not point lasers at people!

Never look into the laser beam port if the sensor is operating.

Maintenance and repairs must be carried out by authorized service personnel only!

Install the device so that the warning is clearly visible and legible.

Do not remove the sensor's protective cover.

The warning label is fixed to the back of the housing as shown in the following figure.

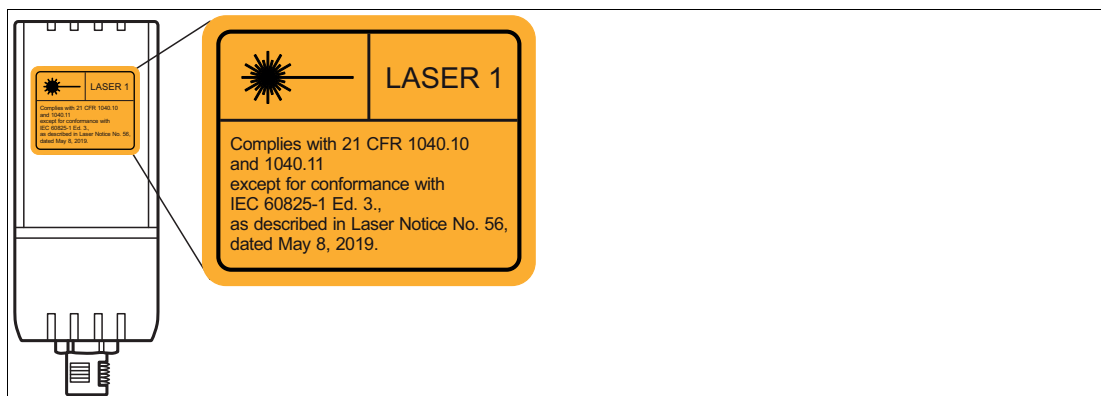


Figure 2.2 Laser radiation warning message

2.3 Dimensions

The devices in the SmartRunner series have the following identical housing dimensions.

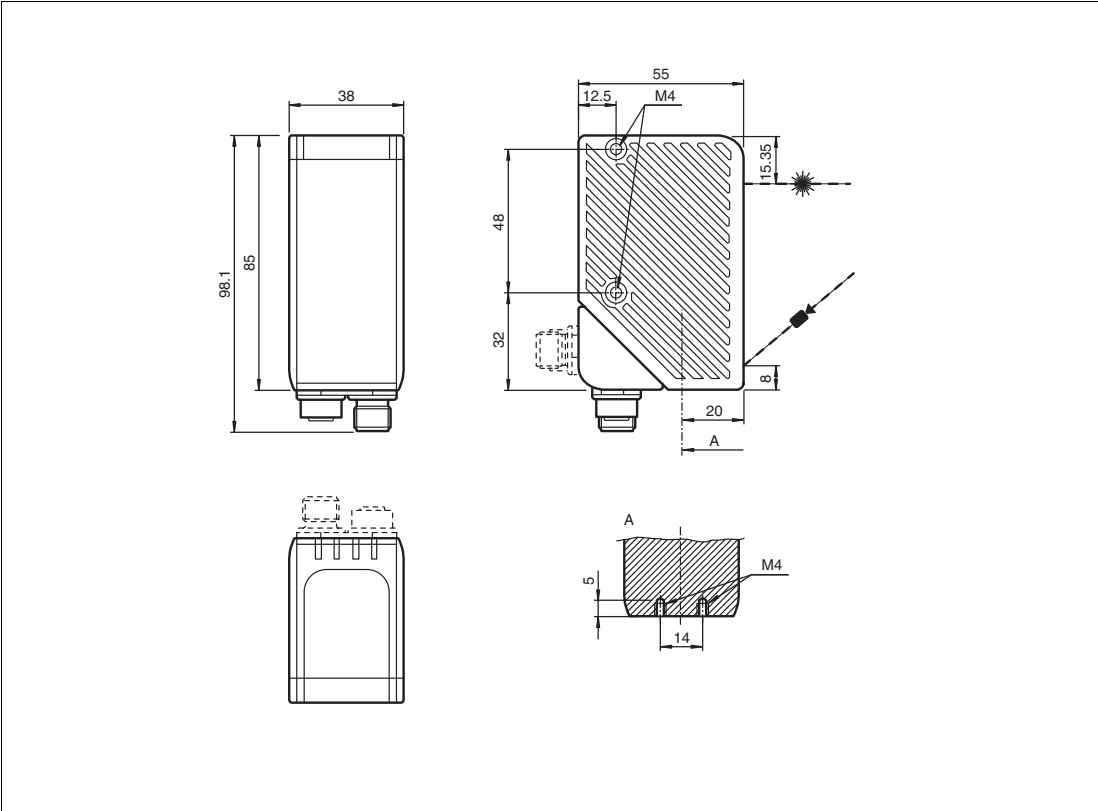


Figure 2.3 Dimensions of the SmartRunner series

2.4 Displays and Operating Elements

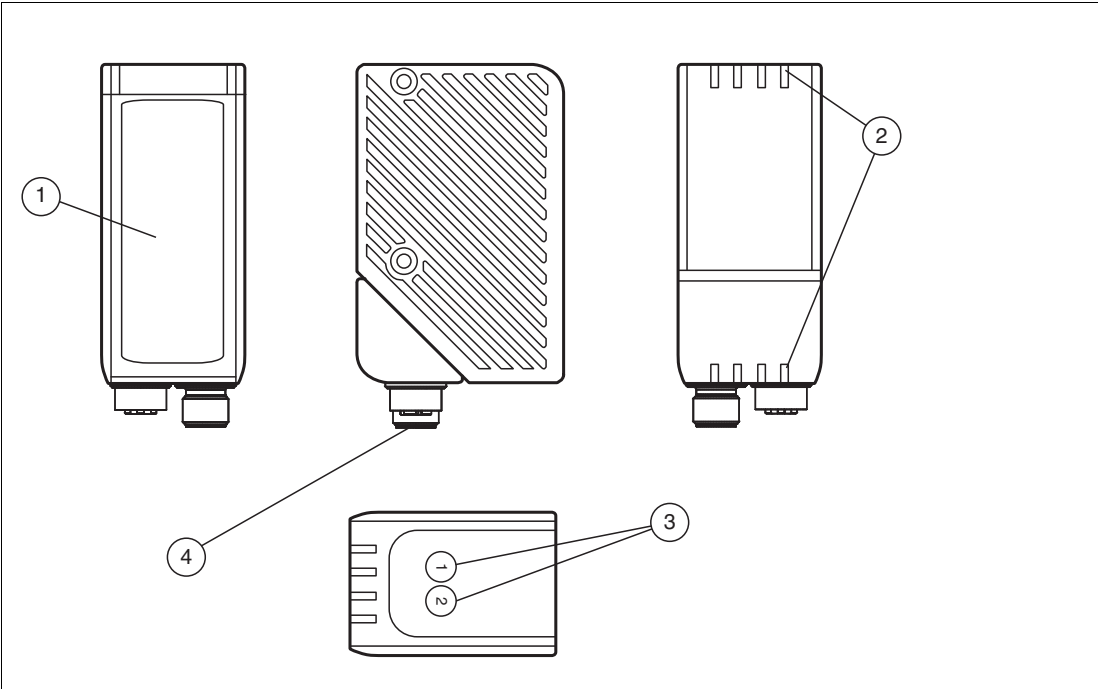


Figure 2.4 Overview of displays and operating elements

2022-07

Position	Designation	Function
1	Protective cover for transmitter optics	Used to protect against damage and contamination
2	LEDs	The functional description for the LEDs can be found in the table below.
3	Function keys in Presentation mode	<ul style="list-style-type: none"> Function key 1: Triggers an evaluation Function key 2: When pressed and held for less than two seconds, activates the teach-in process. When pressed and held for longer than two seconds, activates Code Card mode
	Function keys in Runtime mode	<ul style="list-style-type: none"> Function key 1: No function Function key 2: When pressed and held for longer than two seconds, activates Code Card mode
4	Connections	24 V DC + IO (voltage supply, inputs, and outputs) LAN (Network)

Description of the LEDs

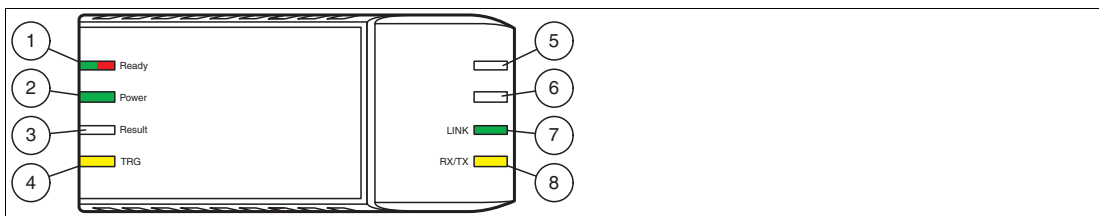


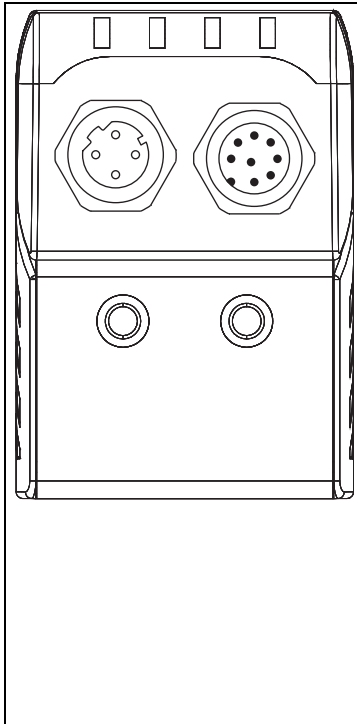
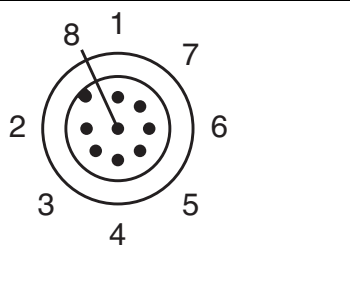
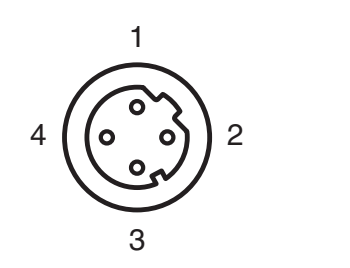
Figure 2.5 Overview of LEDs

LED	Designation	Color	State	Meaning
1	READY	● (red)	On	Illuminates red if a sensor fault is present and after an update. Flashes red during the update process. NOTE: Do not disconnect the power supply during the update!
		● (green)	On	Lights up green if the sensor is ready for operation.
		⚡ (green)	Flashing	Flashes green if the sensor is in configuration mode.
2	POWER	● (green)	On	Lights up as soon as voltage is present.
3	RESULT	-	-	Reserved
4	TRIGGER	● (yellow)	On	Lights up yellow if the hardware trigger signal is activated.
5	-	-	-	-
6	-	-	-	-
7	LINK	● (green)	On	Lights up green if the sensor is connected to the Ethernet.
		● (yellow)	On	Lights up yellow if the sensor is not connected to the Ethernet.
8	RX/TX	⚡ (yellow)	Flickers	Flickers yellow when data communication is active.

2.5 Interfaces and Connections

The following connections are located on the sensor.

Connection Assignment

	24 V DC + IO (voltage supply, inputs, and outputs)	
	M12 plug, A-coded, 8-pin	
		<ol style="list-style-type: none"> 1. IN trigger 2. + UB 3. nc 4. nc 5. nc 6. nc 7. GND
	LAN (Network)	
M12 socket, D-coded, 4-pin		
	<ol style="list-style-type: none"> 1. TX+ Ethernet 2. RX+ Ethernet 3. TX- Ethernet 4. TX- Ethernet 	



Tip

The corner of the housing with the connections can be rotated. To ensure simple cabling, you can turn the connector plug in a different direction depending on the assembly position.

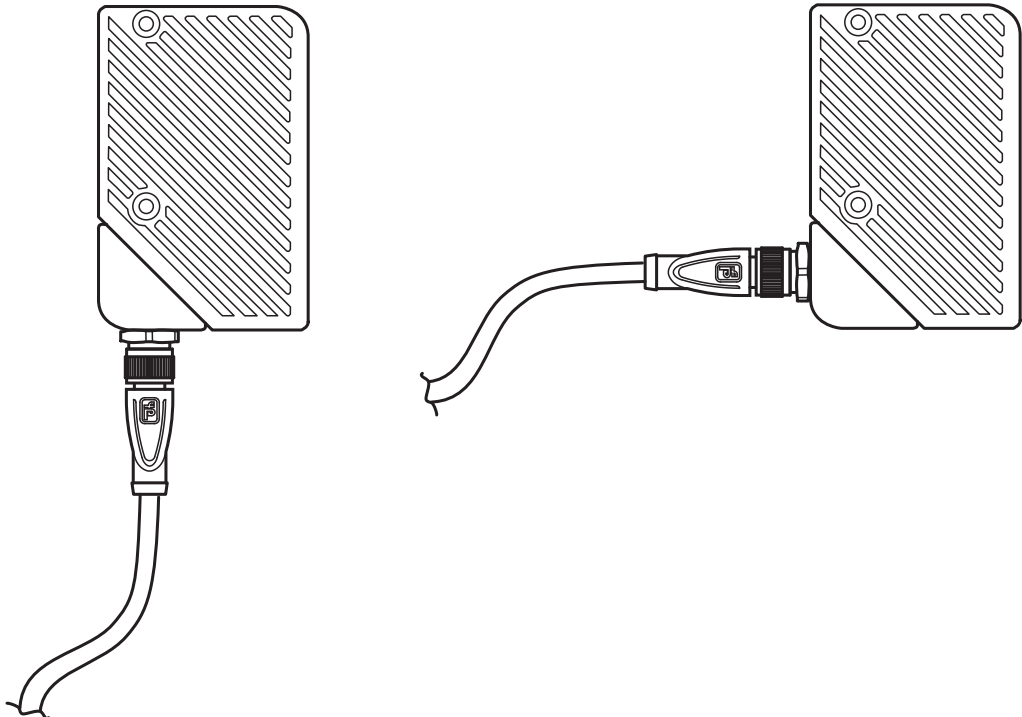


Figure 2.6 M12 connector plug

2.6 Network Interface

The network interface transfers the following data.

- Control commands (start, stop, operating mode)
- Firmware updates (data transfer from the network host to the sensor)
- Result data such as scans and images (data transfer from the sensor to the network host)

XML-embedded strings are used to transfer parameters and status requests. The transfer of data (scans, images) and firmware occurs as binary data.

The data is communicated over TCP/IP via port 50005.

The factory default IP address is 192.168.2.3.

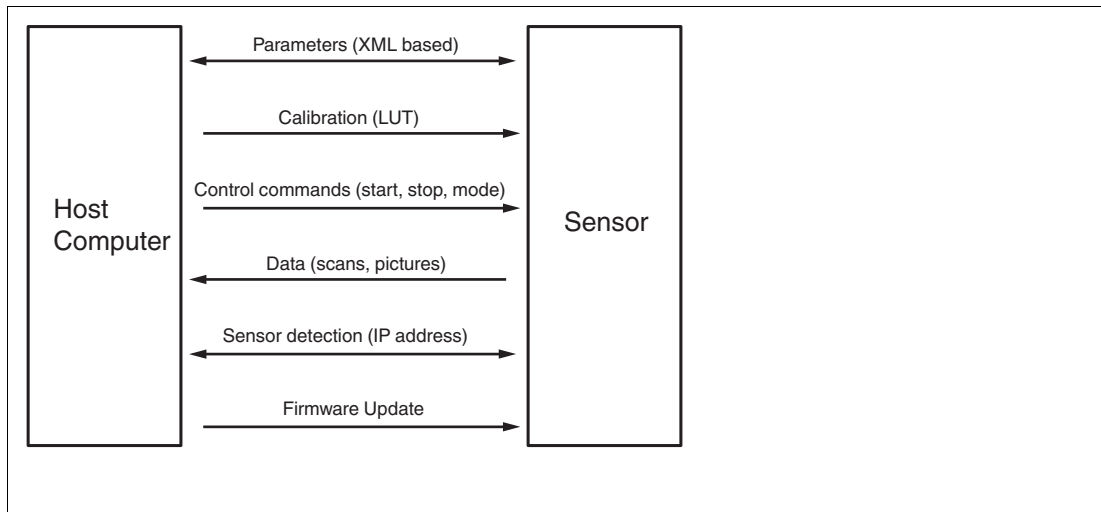


Figure 2.7 Network interface

2.7 Software Interface

SmartRunner Explorer has a VSX programming interface. The driver `VsxProtocolDriver` provides full access to the input and output data of the sensor and facilitates integration in a C#-based programming environment. The driver connects to the sensor and handles the communication in accordance with the communication protocol. The user can access functions for setting parameters on the sensor, retrieving parameter values from the sensor, and saving and loading entire parameter sets both locally and on the sensor. The user can also receive sensor images. Each function also contains an error object from which information can be obtained in the event of an error in the function.

A **.NET 5.0-based** software interface is provided for easy integration with PC software. This software interface takes the form of DLLs and handles the communication with the sensor. Integrate the DLLs into the programming environment and run the programming lines indicated.

Examples relate to the Visual Studio 2019 programming environment and to the C# programming language.

Note

Integrating the NuGet

In order to use the DLL, the NuGet must be integrated. This can be done in Visual Studio by the NuGet package manager, for example. The DLL can be found in the software folder on the SmartRunner Explorer product page of Pepperl and Fuchs. The NuGet is located in the ZIP file stored in this folder under the project folder ext.

Note

More information on configuration see chapter 5.1.

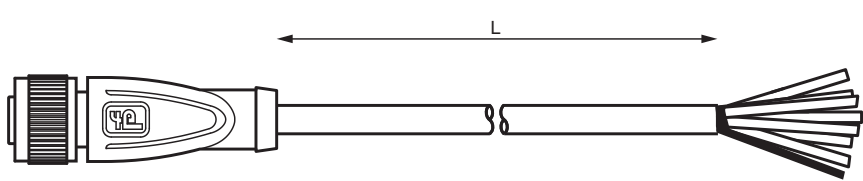
2.8 Accessories

Various accessories are available.

2.8.1 Voltage Supply

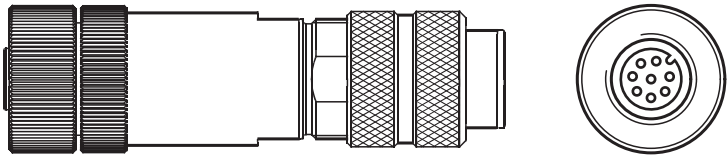
Use the following double-ended cordset to connect the voltage supply, inputs, and outputs to the sensor.

Field-Attachable Female Connector

Model number			
V19-G-2M-PUR-ABG	8-pin M12 socket, straight	L = 2 m	Open cable end with stranded conductors
V19-G-5M-PUR-ABG	8-pin M12 socket, straight	L = 5 m	Open cable end with stranded conductors
V19-G-10M-PUR-ABG	8-pin M12 socket, straight	L = 10 m	Open cable end with stranded conductors

Other lengths on request.

Field-Attachable M12 Socket

Model number			
V19-G-ABG-PG9	<ul style="list-style-type: none"> • 8-pin M12 socket, straight • Screw terminals for max. 0.75 mm² PG9 cable gland • Cable diameter: 5 ... 8 mm 		

2.8.2 Network Cable

The sensor is connected to the network using an M12 plug.

Designation	Description
V45-G	RJ45 network plug, field attachable
V1S-G	M12 plug, 4-pin, field attachable
V1SD-G-2M-PUR-ABG-V45X-G	Cordset, RJ45 network plug with M12 plug, crossed, 4-pin
V1SD-G-2M-PUR-ABG-V45-G	Cordset, RJ45 network plug with M12 plug, 4-pin

3 Installation

3.1 Storage and Disposal

Keep the original packaging. Always store and transport the device in the original packaging.

Store the device in a clean and dry environment. The permitted ambient conditions must be considered, see datasheet.

The device, built-in components, packaging, and any batteries contained within must be disposed in compliance with the applicable laws and guidelines of the respective country.

3.2 Preparation



Unpacking the Device

1. Check the packaging and contents for damage.
↳ In the event of damage, inform the shipping company and notify the supplier.
2. Check the package contents against your order and the shipping documents to ensure that all items are present and correct.
↳ Should you have any questions, direct them to Pepperl+Fuchs.
3. Retain the original packaging in case the device is to be stored or shipped again at a later date.

3.3 Detection range

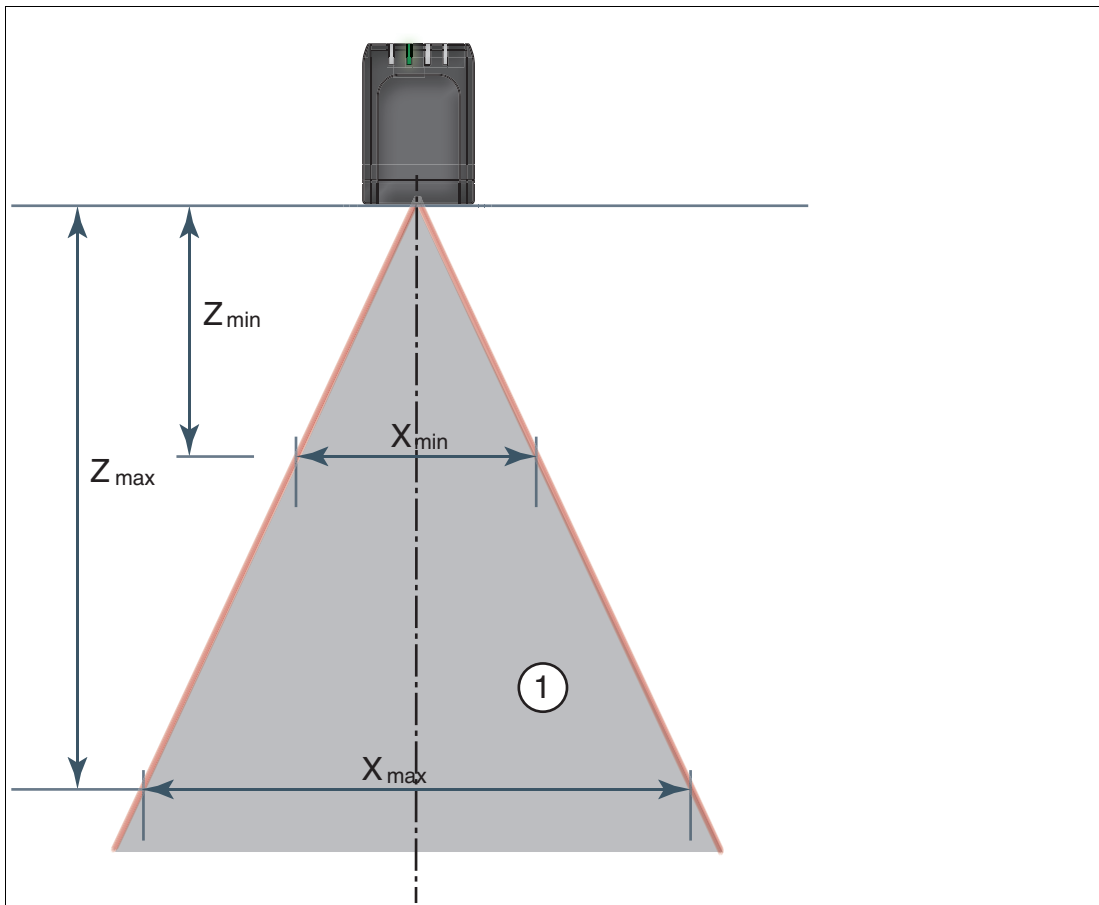


Figure 3.1 Detection range

1 Field of view

Note the detection range of the SmartRunner Matcher when planning your plant. You will find more information on the detection range in the respective data sheet of the sensor.



Note

The smallest possible resolution in the X and Z direction increases on a linear basis to the distance Z to the sensor.

3.4 Mounting the Sensor



Note

Mounting an optical device

- Do not aim the sensor at the sun.
- Protect the sensor from direct long-term exposure to sun.
- Prevent condensation from forming by not exposing the sensor to any major fluctuations in temperature.
- Do not expose the sensor to the effects of any aggressive chemicals.
- Keep the lenses and reflector of the device clean. Clean with a soft cloth, using standard commercial glass cleaner if necessary.

We recommend to clean the optical surface and to check screw fittings and electrical connections at regular intervals.

The operating distance differs depending on the sensor. The correct operating distance can be found in the datasheet for the sensor to be installed.

The following two figures show the orientation of the sensor under extraneous light:

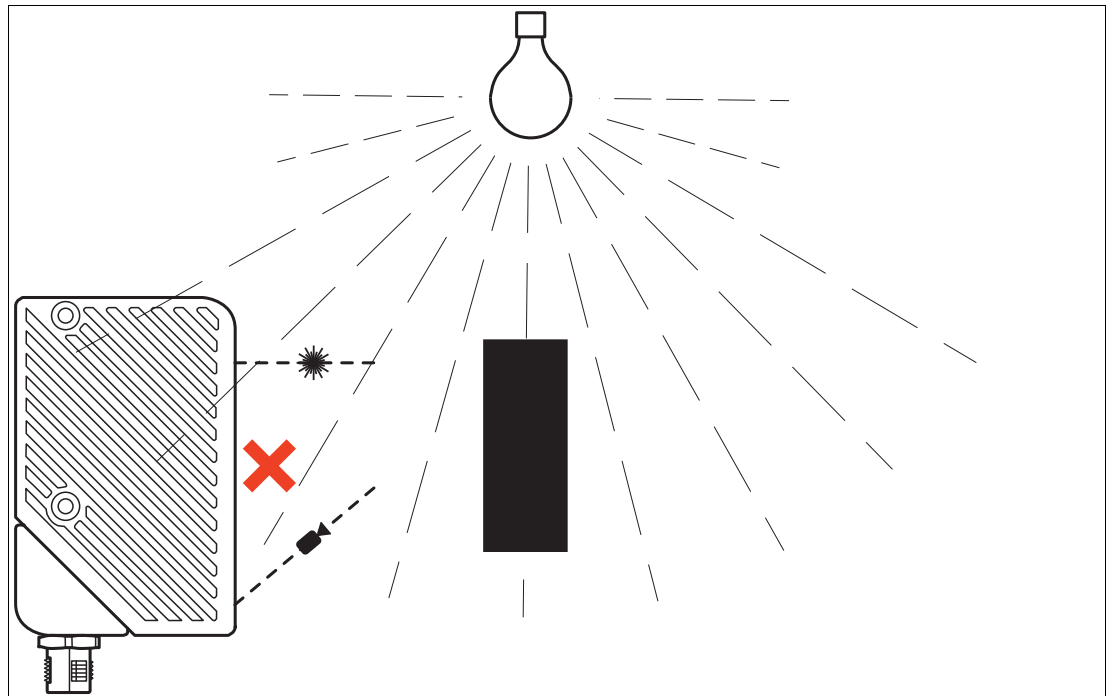


Figure 3.2 Incorrect orientation

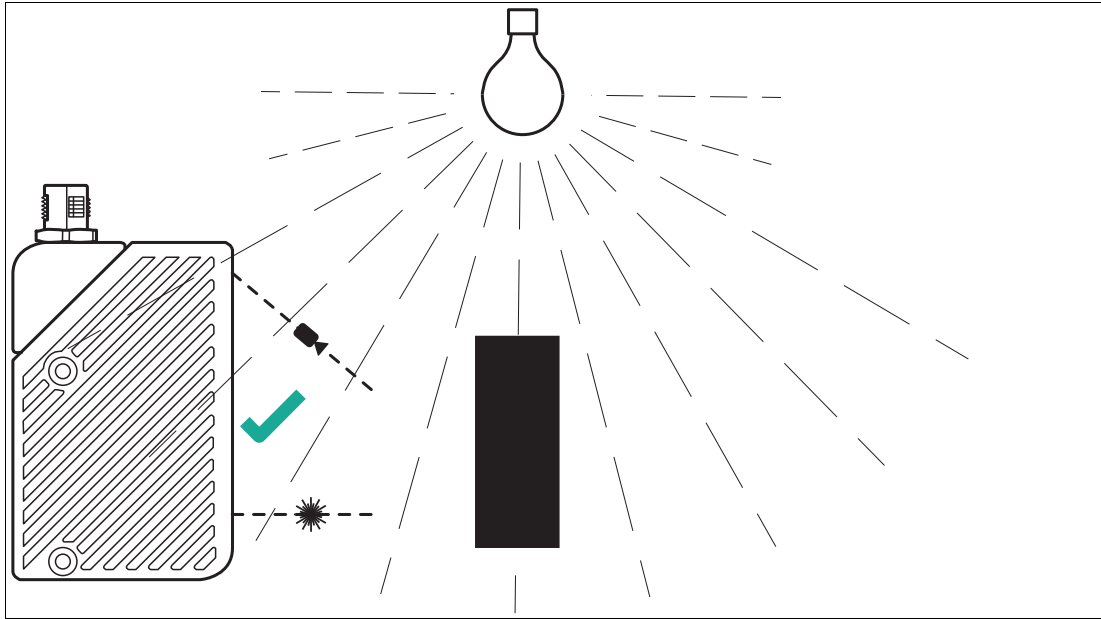


Figure 3.3 Correct orientation

The surface must be level to prevent the housing from becoming misaligned when the fittings are tightened. We advise securing the screws with spring disks to prevent the sensor becoming misaligned. Following installation of the sensor, ensure that there is still sufficient space to connect the connection cable to the sensor



Caution!

Damage to the equipment caused by improper installation!

Device components can be damaged if the permissible screw-in depths and the maximum permissible tightening torque is exceeded.

Note that the threads on the bottom of the housing are not thru-holes.

Observe the maximum permissible screw-in depth to avoid damaging the device or mounting incorrectly.

Never exceed the maximum permissible tightening speed of the fixing screws. The maximum tightening torque of the mounting screws must not exceed 2 Nm.

Mounting the Housing

The device has 2 M4 threads on the base and on both sides of the housing to allow easy installation of the sensor in your plant. This means there are 3 different ways to mount the sensor in your plant.

- One-sided lateral mounting with M4 screws: You can mount the housing on its right-hand or left-hand side using the 2 M4 threaded sleeves. The maximum screw-in depth of the M4 screws is 8 mm.
- Continuous lateral mounting with M3 screws: M4 threaded sleeves are designed in such a manner that M3 screws pass all the way through the housing. Use 2 sufficiently long M3 screws with 2 lock nuts to mount the device in the plant
- Mounting on the underside of the device with M4 screws: You can use the 2 threaded sleeves to mount the housing on the underside of the device. The maximum screw-in depth of the M4 screws is 5 mm.

Positioning the Sensor

When positioning the sensor, ensure that the camera's field of vision is not obscured by the objects being scanned.

3.5 Connecting the Sensor



Caution!

Damage to the device

Connecting an alternating current or excessive supply voltage can damage the device or cause the device to malfunction.

Electrical connections with reversed polarity can damage the device or cause the device to malfunction.

Connect the device to direct current (DC). Ensure that the supply voltage rating is within the specified device range. Ensure that the connecting wires on the female cordset are connected correctly.



Connecting the Supply Voltage

To supply voltage to the sensor, proceed as follows:

1. Connect the 8-pin M12 socket on the power cable to the plug located on the side of the housing.
2. Screw the union nut onto the connector until it reaches the end stop.

↳ This ensures that the power cable cannot be pulled out inadvertently.



Note

Shielding Cables

The shielding of connection lines is required to suppress electromagnetic interference. Establishing a low-resistance or low-impedance connection with the protective conductor or equipotential bonding circuit is an especially important factor in ensuring that these interference currents do not become a source of interference themselves. Only use connection lines with braid. Avoid connection lines with film shield because this would increase the line capacities. The shielding is integrated at both ends, i.e., in the switch cabinet **and** at the sensor. The grounding terminal available as an accessory allows easy integration in the equipotential bonding circuit.

In exceptional cases, the shielding of a connection at one end may be more favorable if:

- An equipotential bonding cable is not laid or cannot be laid.
- A film shield is used.

The following points relating to shielding must be noted:

- Use metal cable clips that cover large areas of the shielding.
- Place the cable shield onto the equipotential bonding rail immediately on entering the switch cabinet.
- Direct the protective grounding connections to a common point in a star configuration.
- The conductor cross section used for grounding should be as large as possible.



Establishing a Network Connection

To establish a network connection, proceed as follows:

1. Use a network cable that has an RJ45 network connector on one side and a 4-pin M12 plug on the other. Insert the 4-pin M12 plug into the socket on the side of the sensor.
2. When delivered, the sensor has a fixed IP address. To facilitate communication within the network, you must configure your network. You can find the configuration data in the network configuration overview.

**Note****Documenting the Network Configuration**

The sensor communicates with the connected machine control system using the TCP/IP protocol. To ensure proper communication, you must record all the changes made to the network configuration.

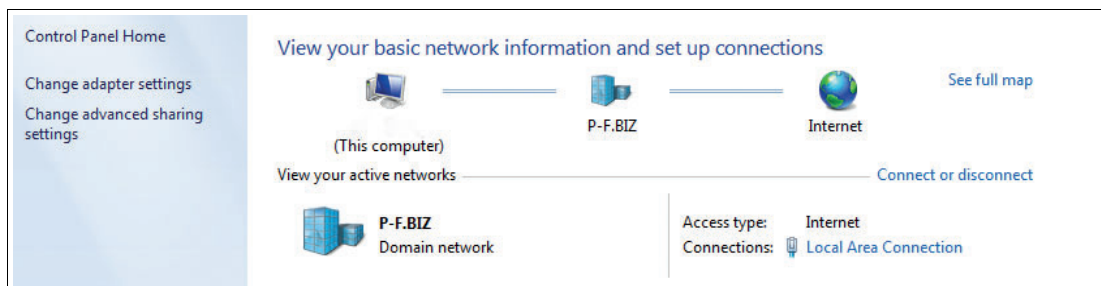
3.6**Setting up Windows Network Communication Between the Sensor and a PC/Laptop**

When the sensor is delivered, it has a fixed IP address: **192.168.2.3**. To enable communication within the network, the network settings of your PC/laptop must be synchronized with the sensor and may need to be adjusted. To do so, proceed as follows:

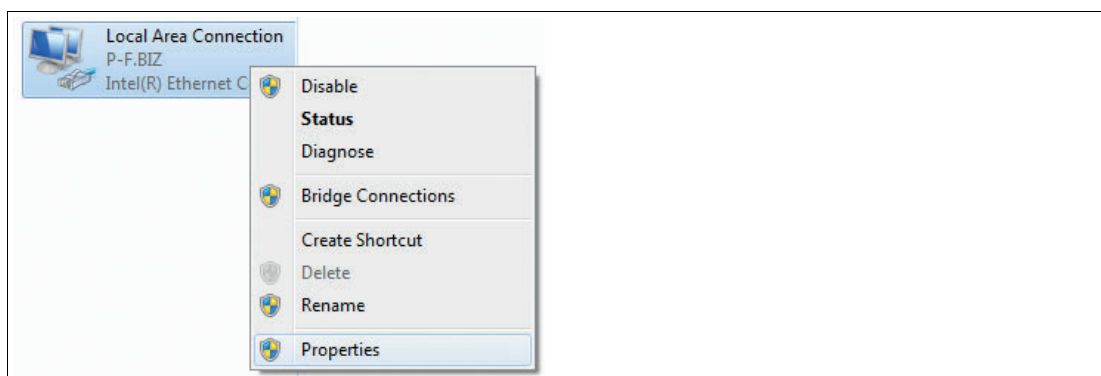
**Setting the IP Address**

The following section describes how to check the network connection settings of your Windows PC and adapt them accordingly. The illustrations in this description were created using Windows 7. The description below also applies to later versions of Windows.

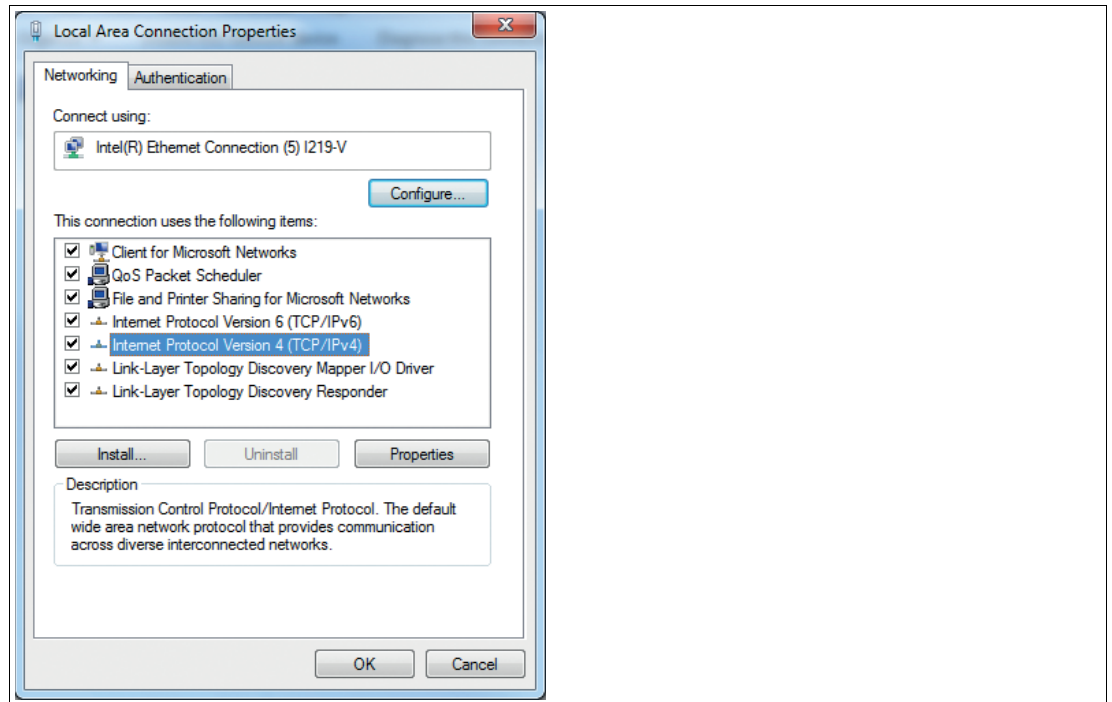
1. Click the Windows **"Start"** button.
2. Select **"Control Panel > Network and Sharing Center."**
3. Now click **"Change adapter settings."**



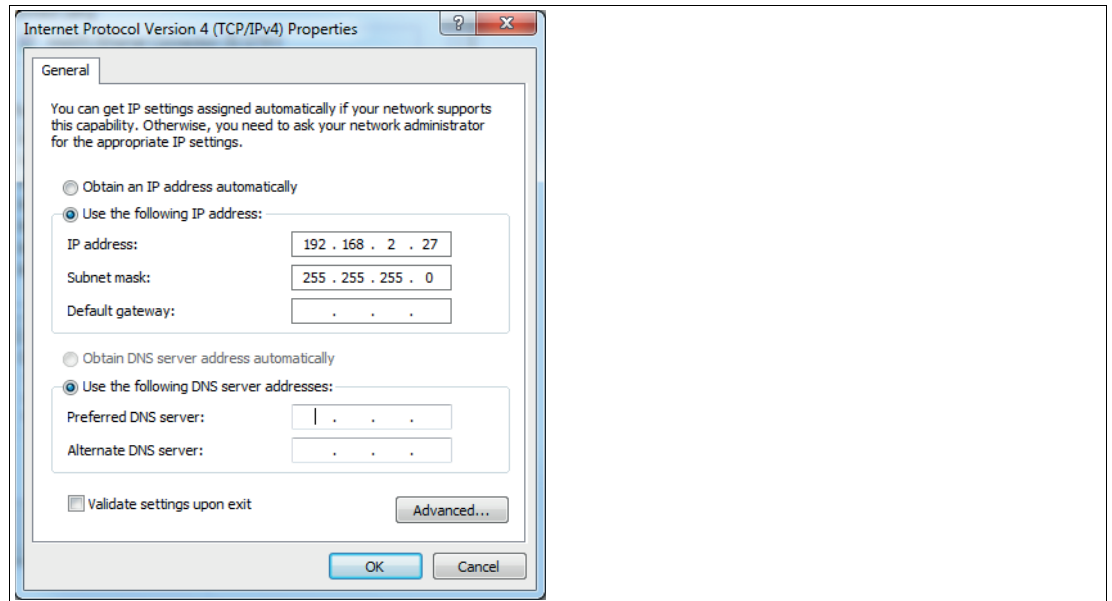
4. Select the required connection and right-click on your selection. In the selection window, select **Properties**.



5. Double-click **"Internet Protocol Version 4 (TCP/IPv4)."**



↳ The **Properties** window for the TCP/IP protocol opens.



6. Select the **"General"** tab.
7. Select the input function **"Use the following IP address."**
8. Enter the IP address of the sensor, but only the first three segments of the IP address. The last segment must be different from the IP address of the sensor.
9. In this example, enter the following IP address and subnet mask:
 - **IP address: 192.168.2.27**
 - **Subnet mask: 255.255.255.0**
10. Click **OK** and click **Cancel** in the next dialog.

↳ This completes the network configuration so that the device can be used.



Note

Changes to the network settings of the PC/laptop require advanced user rights. If necessary, consult with your administrator.

4 Commissioning

Overview of the commissioning steps:

1. Set the operating distance, see chapter 3.3
2. Mount the sensor, see chapter 3.4
3. Connect the sensor, see chapter 3.5
4. Set up Windows network communication between the sensor and a PC/laptop, see chapter 3.6
5. Configure the sensor using the "Vision Configurator" configuration software, see chapter 6

5 Configuration

5.1 VsxProtocolDriver

General

The driver VsxProtocolDriver provides full access to the input and output data of the sensor and facilitates integration in a C#-based programming environment. The driver connects to the sensor and handles communication in accordance with the communication protocol. The user can access functions for setting parameters on the sensor, retrieving parameter values from the sensor, and saving and loading entire parameter sets both locally and on the sensor. The user can also receive sensor images. Each function also contains an error object from which information can be obtained in the event of an error in the function.

The driver is implemented in C# and requires .NET 5.0 or higher.

The functions of the driver can be used **synchronously** or **asynchronously**. For this, the required instance must be created using the Init function of the respective classes. The VsxProtocolDriver class provides the asynchronous driver. The VsxProtocolDriverSync class provides the synchronous interface.

For clarity, this manual only describes the most important functions and variables. The SDK contains additional functions that are used for other Pepperl+Fuchs vision sensors. These functions are described in the corresponding product manual. There are several declaration options for some functions. In the following, the preferred functions are marked in bold.



Note

Integrating the NuGet

In order to use the SDK, the NuGet must be integrated. This can be done in Visual Studio using the NuGet package manager, for example. The SDK can be found on the product page of the corresponding sensor from Pepperl+Fuchs in the software folder. The NuGet is located in the ZIP file stored in this folder under the project folder ext.

Synchronous and Asynchronous Functions

The auxiliary classes used in the parameters are described below.

Static Functions

UDP Broadcast

The static function returns a list of the Vsx devices that are found on the network via a UDP broadcast.

Asynchronous Function

```
public static Task<(bool Succ, List<Device> DeviceList, Error Error-  
Desc)> UdpDeviceList()
```

Synchronous Function

```
public static (bool Succ, List<Device> DeviceList, Error ErrorDesc)  
UdpDeviceList()
```

Possible error IDs: VSX_DRIVER_CONNECTION_ERROR

Configure Network Settings via UDP

Network settings on the device are changed via UDP.

Asynchronous Function

```
public static Task<(bool Succ, Error ErrorDesc)> SetNetworkSettings-
ViaUdp(string macAddress, string ipAddress, string networkMask,
string gateway)
```

Synchronous Function

```
public static (bool Succ, Error ErrorDesc) SetNetworkSettingsVi-
aUdp(string macAddress, string ipAddress, string networkMask, string
gateway)
```

Possible error IDs: None

TCP/IP Connection

Initializes a new driver instance that can be used to communicate with the device via TCP/IP. While the IP address must be specified, the default VSXPORT = 50005 can be used.

Asynchronous Function

```
public static VsxProtocolDriver Init(string ipAddress, int port =
VSXPORT, string pluginName = "")
```

Synchronous Function

```
public static VsxProtocolDriverSync Init(string ipAddress, int port
=VSXPORT, string pluginName = "")
```

Save an IVsxMessage

An IVsxMessage is a general interface for data exchanged between the PC and the sensor.

Asynchronous Function

```
public static (bool Succ, Error ErrorDesc) SaveData(string filename,
IVsxMessage message)
```

Synchronous Function

```
public static (bool Succ, Error ErrorDesc) SaveData(string filename,
IVsxMessage message)
```

Possible error IDs: VSX_DRIVER_DATA_ERROR, VSX_DRIVER_INVALID_DATA_ERROR, VSX_DRIVER_SAVE_FILE_ERROR

Non-Static Functions

Establish a Connection to the Device

Establishes a connection to the device using the parameters set via Init. CONNECTION_TIMEOUT_MS = 1000 can be used as the timeout for opening the connection. A connection to the device must be established in order to use non-static features.

Asynchronous Function


```
public Task<(bool Succ, Error ErrorDesc)> Connect(int timeout = CONNECTION_TIMEOUT_MS)
```

Synchronous Function

```
public (bool Succ, Error ErrorDesc) Connect(int timeout = VsxProtocolDriver.CONNECTION_TIMEOUT_MS)
```

Possible error IDs: VSX_DRIVER_CONNECTION_ERROR

Disconnect from the Device

Disconnects from the device.

Asynchronous Function

```
public Task<(bool Succ, Error ErrorDesc)> Disconnect()
```

Synchronous Function

```
public (bool Succ, Error ErrorDesc) Disconnect()
```

Possible error IDs: None

Re-establish the Connection with Transferred Parameters

Disconnects the device and re-establishes the connection using the transferred parameters. These functions can be used if the connection parameters need to be changed at runtime.

Asynchronous Function

```
public Task<(bool Succ, Error ErrorDesc)> ReConnect(string ipAddress, int port = VSXPORT)
```

Synchronous Function

```
public (bool Succ, Error ErrorDesc) ReConnect(string ipAddress, int port = VSXPORT)
```

Asynchronous Function

```
public Task<(bool Succ, Error ErrorDesc)> ReConnect(string serialPort, int baudrate, TheSensor.ConnectionType connectionType)
```

Synchronous Function

```
public (bool Succ, Error ErrorDesc) ReConnect(string serialPort, int baudrate, TheSensor.ConnectionType connectionType)
```

Possible error IDs: VSX_DRIVER_CONNECTION_ERROR

Device Information

Transmits information about the device (such as MAC address, etc.).

Asynchronous Function

```
public Task<(bool Succ, Device CurrentDevice, Error ErrorDesc)> Get-  
CurrentDeviceInformation()
```

Synchronous Function

```
public (bool Succ, Device CurrentDevice, Error ErrorDesc) GetCurrent-  
DeviceInformation()
```

Possible error IDs: VSX_DRIVER_CONNECTION_ERROR

Feature List

Transmits the list of features available on the device.

Asynchronous Function

```
public Task<(bool Succ, float XmlVersion, Hashtable FeatureList,  
Error ErrorDesc)> GetFeatureList()
```

Synchronous Function

```
public (bool Succ, float XmlVersion, Hashtable FeatureList, Error  
ErrorDesc) GetFeatureList()
```

Possible error IDs: VSX_DRIVER_CONNECTION_ERROR

Parameter List incl. Detailed Information and Current Values

Transmits a list of all parameters available on the device, including detailed information and their current values.

Asynchronous Function

```
public Task<(bool Succ, List<Parameter> ParameterList, Error Error-  
Desc)> GetParameterList()
```

Synchronous Function

```
public (bool Succ, List<Parameter> ParameterList, Error ErrorDesc)  
GetParameterList()
```

Possible error IDs: VSX_DRIVER_GENERAL_ERROR, VSX_DRIVER_DATA_ERROR, VSX-
_DRIVER_CONNECTION_ERROR

Transmit the Value of a Single Device Parameter

Transmits the value of a single device parameter.

Asynchronous Function

```
public Task<(bool Succ, object parameterValue, Error ErrorDesc)>  
GetSingleParameterValue(Parameter parameter)
```

```
public Task<(bool Succ, object parameterValue, Error ErrorDesc)>  
GetSingleParameterValue(string parameterId)
```

```
public Task<(bool Succ, object parameterValue, Error ErrorDesc)>  
GetSingleParameterValue(ushort settingsVersion, ushort configVer-  
sion, string configId, string parameterId)
```

Synchronous Function

```
public (bool Succ, object parameterValue, Error ErrorDesc) GetSin-  
gleParameterValue(Parameter parameter)
```

```
public (bool Succ, object parameterValue, Error ErrorDesc) GetSin-  
gleParameterValue(string parameterId)
```

```
public (bool Succ, object parameterValue, Error ErrorDesc) GetSin-  
gleParameterValue(ushort settingsVersion, ushort configVersion,  
string configId, string parameterId)
```

Possible error IDs: VSX_DRIVER_CONNECTION_ERROR, VSX_DRIVER_DATA_ERROR

Set the Value of a Single Device Parameter

Sets the value of a single parameter on the device. The parameter is defined by transferring the function parameters in the same way as for the function *GetSingleParameterValue*. In addition, the desired value is transferred.

Asynchronous Function

```
public Task<(bool Succ, List<Parameter> DependendParameters, Error  
ErrorDesc)> SetSingleParameterValue(Parameter parameter, object  
value)
```

```
public Task<(bool Succ, List<Parameter> DependendParameters, Error  
ErrorDesc)> SetSingleParameterValue(string parameterId, object  
value)
```

```
public Task<(bool Succ, List<Parameter> DependendParameters, Error  
ErrorDesc)> SetSingleParameterValue(ushort settingsVersion, ushort  
configVersion, string configId, string parameterId, object value)
```

Synchronous Function

```
public (bool Succ, List<Parameter> DependendParameters, Error Error-
Desc) SetSingleParameterValue(Parameter parameter, object value)
```

```
public (bool Succ, List<Parameter> DependendParameters, Error Error-
Desc) SetSingleParameterValue(string parameterId, object value)
```

```
public (bool Succ, List<Parameter> DependendParameters, Error Error-
Desc) SetSingleParameterValue(ushort settingsVersion, ushort con-
figVersion, string configId, string parameterId, object value)
```

Possible error IDs: VSX_DRIVER_CONNECTION_ERROR, VSX_DRIVER_DATA_ERROR

Change Network Settings

Changes the network settings on the device. The connection to the device is terminated and must be re-established using the "Connect" function ().

Asynchronous Function

```
public Task<(bool Succ, Error ErrorDesc)> SetNetworkSettings(string
ipAddress, string networkMask, string gateway)
```

Synchronous Function

```
public (bool Succ, Error ErrorDesc) SetNetworkSettings(string ipAd-
dress, string networkMask, string gateway)
```

Possible error IDs: VSX_DRIVER_CONNECTION_ERROR

Send the Firmware File to the Device

Sends the firmware file under the specified path and file name to the device. The current status can be read out via *FirmwareStateChannelReader* while the update is in progress.

Asynchronous Function

```
public Task<(bool Succ, Error ErrorDesc)> SendFirmware(string file-
Name)
```

Synchronous Function

```
public (bool Succ, Error ErrorDesc) SendFirmware(string fileName)
```

Possible error IDs: VSX_DRIVER_CONNECTION_ERROR, VSX_DRIVER_DEVICE_ERROR

Read the Parameter Set

Reads the current parameter set from the device and saves it under the specified path and file name.

Asynchronous Function

```
public Task<(bool Succ, Error ErrorDesc)> DownloadParameterSet(string
destinationFileName)
```

Synchronous Function

```
public (bool Succ, Error ErrorDesc) DownloadParameterSet(string destinationFileName)
```

Possible error IDs: VSX_DRIVER_SAVE_FILE_ERROR

Load Parameter Set

Loads a parameter set stored under the specified path and file name and sends it to the device.

Asynchronous Function

```
public Task<(bool Succ, Error ErrorDesc)> UploadParameterSet(string sourceFileName)
```

Synchronous Function

```
public (bool Succ, Error ErrorDesc) UploadParameterSet(string sourceFileName)
```

Possible error IDs: VSX_DRIVER_CONNECTION_ERROR

Save the Parameter Set

Saves the current parameter settings on the device. The specified values are set each time the device is started.

All parameter data is initially stored on the device in a volatile state and can be reset by a restart or by *LoadParameterSetOnDevice*. Only *SaveParameterSetOnDevice* permanently saves the parameters.

Asynchronous Function

```
public Task<(bool Succ, Error ErrorDesc)> SaveParameterSetOnDevice()
```

Synchronous Function

```
public (bool Succ, Error ErrorDesc) SaveParameterSetOnDevice()
```

Possible error IDs: VSX_DRIVER_CONNECTION_ERROR

Load Parameter Settings

Loads the parameter settings saved via *SaveParameterSetOnDevice* on the device. The parameters then have the previously saved values. A current parameter list is transmitted.

Asynchronous Function

```
public Task<(bool Succ, List<Parameter> ParameterList, Error ErrorDesc)> LoadParameterSetOnDevice()
```

Synchronous Function

```
public (bool Succ, List<Parameter> ParameterList, Error ErrorDesc) LoadParameterSetOnDevice()
```

Possible error IDs: VSX_DRIVER_GENERAL_ERROR, VSX_DRIVER_DATA_ERROR, VSX_DRIVER_CONNECTION_ERROR

Load Factory Settings

Loads the factory settings of all parameters on the device. A current parameter list is transmitted.

Asynchronous Function

```
public Task<(bool Succ, List<Parameter> ParameterList, Error ErrorDesc)> LoadDefaultParameterSetFromDevice()
```

Synchronous Function

```
public (bool Succ, List<Parameter> ParameterList, Error ErrorDesc) LoadDefaultParameterSetFromDevice()
```

Possible error IDs: VSX_DRIVER_GENERAL_ERROR, VSX_DRIVER_DATA_ERROR, VSX_DRIVER_CONNECTION_ERROR

Read Out Sensor Data

Starts reading out sensor data. The sensor data for each trigger is packaged in *VsxDynamicContainer*. The data can be called up using the *GetDynamicContainer* function. The *bufferSize* specifies how many containers can be buffered by the driver, the *startCondition* specifies the container from which buffering should be performed, and the *strategy* determines what should happen if the buffer is full. DROP_OLDEST indicates that the oldest stored container is discarded, while DROP_WRITE means the latest container received is discarded.

Asynchronous Function

```
public void ResetDynamicContainerGrabber(int bufferSize, PF.Foundation.Communication.Vsx.Strategy strategy = PF.Foundation.Communication.Vsx.Strategy.DROP_OLDEST)
```

Synchronous Function

```
public void ResetDynamicContainerGrabber(int bufferSize, PF.Foundation.Communication.Vsx.Strategy strategy = PF.Foundation.Communication.Vsx.Strategy.DROP_OLDEST)
```

Possible error IDs: None

Output of the Oldest Buffered DynamicContainer

Returns the oldest buffered DynamicContainer (). This function returns an error after *timeoutMs* milliseconds if no new DynamicContainer is available by then.

Asynchronous Function

```
public Task<(bool Succ, IVsxDynamicContainer Container, int NumberOfDiscardedItems, Error ErrorDesc)> GetDynamicContainer(int timeoutMs = System.Threading.Timeout.Infinite)
```

Synchronous Function

```
public (bool Succ, IVsxDynamicContainer Container, int NumberOfDiscardedItems, Error ErrorDesc) GetDynamicContainer(int timeoutMs = Timeout.Infinite)
```

Possible error IDs: VSX_DRIVER_INIT_ERROR, VSX_DRIVER_TIMEOUT_ERROR

Read Out Log Data

Restarts the readout of log data. The log data can be called up using the *GetLogMessage* function. The *bufferSize* specifies how many log data messages can be buffered by the driver. The *typeMask* specifies which log data types should be transferred from the device, and the *strategy* specifies what should happen if the buffer is full. DROP_OLDEST indicates that the oldest stored log message is discarded, while DROP_WRITE means the latest log message received is discarded.

Asynchronous Function

```
public void ResetLogMessageGrabber(int bufferSize, int typeMask, PF.Foundation.Communication.Vsx.Strategy strategy = PF.Foundation.Communication.Vsx.Strategy.DROP_OLDEST)
```

Synchronous Function

```
public void ResetLogMessageGrabber(int bufferSize, int typeMask, PF.Foundation.Communication.Vsx.Strategy strategy = PF.Foundation.Communication.Vsx.Strategy.DROP_OLDEST)
```

Possible error IDs: None

Output of the Oldest Buffered LogMessage

Returns the oldest buffered *LogMessage* (). This function returns an error after *timeoutMs* milliseconds if no new DynamicContainer is available by then.

Asynchronous Function

```
public Task<(bool Succ, VsxLogMessage LogMessage, int NumberOfDiscardedItems, Error ErrorDesc)> GetLogMessage(int timeoutMs = System.Threading.Timeout.Infinite)
```

Synchronous Function

```
public (bool Succ, VsxLogMessage LogMessage, int NumberOfDiscardedItems, Error ErrorDesc) GetLogMessage(int timeoutMs = Timeout.Infinite)
```

Possible error IDs: VSX_DRIVER_INIT_ERROR, VSX_DRIVER_TIMEOUT_ERROR

Properties

Information about Firmware Updates

The *FirmwareStateChannelReader* provides information about the current status of the firmware update

```
public ChannelReader<FirmwareState> FirmwareStateChannelReader
```

Connection Status

Indicates the connection status.

```
public bool Connected { get; }
```

Timeout

Timeout specifies (in ms) how long to wait for a response to a request sent to the device. Depending on the connection type, the default value is `DEFAULT_ETHERNET_TIMEOUT_MS` or `DEFAULT_SERIAL_TIMEOUT_MS`.

```
public int WaitTimeout { get; set; }
```

Number of Discarded DynamicContainers or LogMessages

Specifies the number of DynamicContainers or LogMessages discarded since the last reset grabber, once there was no more space in the buffer.

DynamicContainer

```
public int MissingContainerFramesCounter { get; }
```

LogMessages

```
public int MissingLogMessagesCounter { get; }
```

Number of Buffered DynamicContainers or LogMessages

Specifies how many DynamicContainers or LogMessages can be buffered by the driver.

DynamicContainer

```
public int DynamicContainerQueueSize { get; }
```

LogMessages

```
public int LogMessageQueueSize { get; }
```

Which DynamicContainers or LogMessages Are Discarded

Specifies which DynamicContainers or LogMessages should be discarded if the buffer is full. `DROP_OLDEST` discards the oldest stored container or log message, while `DROP_WRITE` discards the most recently received.

DynamicContainer

```
public Strategy DynamicContainerGrabberStrategy { get; }
```

LogMessages

```
public Strategy LogMessageGrabberStrategy { get; }
```

Events

Loss of Connection

The event is triggered as soon as the driver detects a loss of connection to the device. The parameters are the IP address of the previously connected device and a message indicating why the connection was lost. This event is only triggered if a TCP/IP connection is used.

```
public event Action<string, string> OnDisconnect
```


Auxiliary Classes

Device Information (PF.Types.Device)

Contains information about the currently connected device.

```
public string PhysicalAddress;  
  
public int PhysicalPort;  
  
public string IpAddress;  
  
public string NetworkMask;  
  
public string Gateway;  
  
public string MacAddress;  
  
public string Identifier;  
  
public string FirmwareVersion;  
  
public string SensorType;
```

Error Information (PF.Types.Error)

Contains information about an error that has occurred.

```
public ErrorId Id;  
  
public string Tag;  
  
public string Message;
```

The Following Error IDs Are Possible

```
VSX_DRIVER_NO_ERROR = 0x0,  
  
VSX_DRIVER_INIT_ERROR = -0x1,  
  
VSX_DRIVER_TIMEOUT_ERROR = -0x2,  
  
VSX_DRIVER_SAVE_FILE_ERROR = -0x3,  
  
VSX_DRIVER_DATA_ERROR = -0x4,
```

```
VSX_DRIVER_CONNECTION_ERROR = -0x5,
```

```
VSX_DRIVER_INVALID_DATA_ERROR = -0x6,
```

```
VSX_DRIVER_DEVICE_ERROR = -0x7,
```

```
VSX_DRIVER_GENERAL_ERROR = -0x1000
```

Parameters (PF.VsxDriver.Parameter)

Contains information about a device parameter. Important properties here are the details of the version and IDs, which are required for setting a parameter. Another property is Value, which provides the current parameter value. Not every property is used for every parameter.

```
ushort settingsVersion;  
  
ushort configVersion;  
  
string configId;  
  
string parameterId;  
  
string name;  
  
Vsx.ParameterTypes type;  
  
Vsx.ValueTypes valueType;  
  
bool enable;  
  
bool visible;  
  
object min;  
  
object max;  
  
object step;  
  
string userLevel;  
  
object value;  
  
object defaultValue;  
  
string unit;
```

2022-07

```
List<ItemTuple> items;
```

Information about the Firmware Update (PF.Foundation.Communication.Vsx.Device.FirmwareState)

Contains information about the current status of a firmware update that is running.

```
public int Id;
```

```
public string Tag;
```

```
public string Message;
```

List of IVsxMessages (PF.Foundation.Communication.Vsx.Message.VsxDynamicContainerMessage: IVsxMessage)

Contains a list of IVsxMessages, which in turn contain data sent by the device. The messages it contains are identified in the list using a string. The possible messages are device-specific. Example of a VsxDynamicContainerMessage.

```
public bool ContainsMessage(string tag)
```

```
public IVsxMessage GetMessage(string tag)
```

Image Data (PF.Foundation.Communication.Vsx.Message.VsxImageData2Message: IVsxMessage)

Contains image data of a specific image. Depending on whether the individual image values are bytes or floats, they are stored in the respective ImageData or ImageDataFloats array.

```
public ImageData2Format;
```

```
public int Width;
```

```
public int Height;
```

```
public int LinePitch;
```

```
public long FrameCounter;
```

```
public double CoordinateScale;
```

```
public double CoordinateOffset;
```

```
public double AxisMin;
```

```
public double AxisMax;
```

```
public double InvalidDataValue;
```

```
public byte[] ImageData;
```

```
public float[] ImageDataFloats;
```

PF.Foundation.Communication.Vsx.Message.VsxTransformationMessage: IVsxMessage

```
public double TranslationTX;
```

```
public double TranslationTY;
```

```
public double TranslationTZ;
```

```
public double QuaternionQ0;
```

```
public double QuaternionQ1;
```

```
public double QuaternionQ2;
```

```
public double QuaternionQ3;
```

SmartRunner

DynamicContainer

A DynamicContainer received by the sensor can contain the following messages:

TagName	?Type?	Status	Description
"Image"	ImageData[Mono8]	Optional	Image data
"Line"	LineMessage	Optional	Line data

Structure of the LineMessage Type

Main Property

Line

Returns a list of points. Each point consists of image and world coordinates, and quality information. In detail, the point information looks as follows:

The LineMessage has the following interface:

- ContentValue: Reserved for future use
- ImageCoordinate: Image coordinates in subpixels
- Intensity: Value that correlates with the grayscale value point
- LineID: Always 0 in single-line systems
- Quality: Point quality as a percentage
- SegmentID: Line segment number if supported, otherwise 0
- Valid: True if point is valid
- WorldCoordinate: World coordinates in mm

Additional Properties (Meta-Information)

Meta-information provides additional information about the data.

MinX

Minimum possible X value of the device family, usually negative in μm .

MaxX

Maximum possible X value of the device family, usually positive in μm .

MinZ

Minimum possible Z value of the device family, positive in μm .

MaxZ

Maximum possible Z value of the device family, positive in μm .

FrameCounter

Current frame counter, related data has the same counter.

ScaleXYZ, ScaleC

Scaling factors for the internal calculation of the line data.

Width

Maximum number of possible line points.

Format

Reserved for future use.

Status

Reserved for future use.

Lines

Returns a list of lines. Usually, the list contains only one line. Each line consists of a list of points, the points of the first line can be called directly via Line.

5.2 Configuration Parameters

Before the sensor can be used in the system, it must be parameterized. The parameters used for the sensor are listed below.

The "read (r)/write (w)" column indicates whether the feature is readable ("r"), writable ("w"), or both ("r/w").

Config ID	Parameter ID	Parameter name	Values	read (r)/ write (w)	Description
Toolbar	0x640001	Trigger Laser	0.1	-/w	Perform manual trigger.
	0x640002	LED trigger	0.1	-/w	Perform LED trigger. NOTE: If auto trigger mode is enabled, a line image is output via the "LED trigger."
	GetImage	Get Image	0.1	-/w	Resend last image.
	0x6D0001	Teach	0.1	-/w	Automatically adjust exposure time with next trigger.

Config ID	Parameter ID	Parameter name	Values	read (r)/ write (w)	Description
General	0xCC0001	Custom Name	Char[32]	r/w	User-defined name
	0x510001	AutoTrigger	0.1	r/w	Deactivation/activation of cyclical image capture.
	0x680001	Exposure time	1 .. 10000	r/w	Sets the manual exposure time. To set the exposure time manually, the "Use manual exposure time" (see "Illumination Menu Item" on page 56) function must be activated. Increasing the value increases the exposure time and therefore the image brightness. Values below 1000 μ s are suitable in most cases.
	0xBE0001	Use manual exposure time	0.1	r/w	When enabled, the manually set exposure time is used. If this box is not checked, the exposure time is regulated automatically during the "teach-in" process
	0x100001	Flash time	1 .. 10000	r/w	Exposure time of used flash light when triggering LEDs in μ s (see 0x640002)
	0x9F0001	Object contrast	1 .. 100	r/w	Contrast threshold for detecting laser line on object in % (default value: 45)
	0xC90001	ROI min X	-3200 ... 3200	r/w	The lowest value on the x-axis of the ROI (in mm).
	0xC90002	ROI max X	-3200 ... 3200	r/w	The highest value on the x-axis of the ROI (in mm).
	0xC90003	ROI min Z	1 ... 3200	r/w	The lowest value on the z-axis of the ROI (in mm).
	0xC90004	ROI max Z	1 ... 3200	r/w	The highest value on the z-axis of the ROI (in mm).
	ImageTransferActive	Transfer images	0.1	r/w	Automatically transfer images. NOTE: may increase analysis time.

Examples of the Application of Configuration Parameters

The following declarations are used in these examples:

```
public static class Sr2DParameterIdentifier
{
    public static string Name_0x010001 = "0x010001";
    public static string Homepage_0x020001 = "0x020001";
    public static string Productname_0x030001 = "0x030001";
    public static string Version_0x070001 = "0x070001";
    public static string Firmware_0xC30001 = "0xC30001";
    public static string Autotrigger_0x510001 = "0x510001";
    public static string Exposuretime_0x680001 = "0x680001";
    public static string Usemanualexposuretime_0xBE0001 = "0xBE0001";
    public static string Flashtime_0x100001 = "0x100001";
    public static string Presentationmode_0xFD0001 = "0xFD0001";
    public static string Objectcontrast_0x9F0001 = "0x9F0001";
}
```

```

        public static string      HighPositionResolution_0x280001 = "0x280001";
        public static string      ROIMinX_0xC90001 = "0xC90001";
        public static string      ROIMaxX_0xC90002 = "0xC90002";
        public static string      ROIMinZ_0xC90003 = "0xC90003";
        public static string      ROIMaxZ_0xC90004 = "0xC90004";
        public static string      Triggerlaser_0x640001 = "0x640001";
        public static string      TriggerLED_0x640002 = "0x640002";
        public static string      Getimage_GetImage = "GetImage";
        public static string      Teach_0x6D0001 = "0x6D0001";
        public static string      CustomName_0xCC0001 = "0xCC0001";
        public static string      ImageTransferActive = "ImageTransferActive";
    }

```

Setting the Auto Trigger Mode:

```
var setParamResult = await _vsxProtocolDriver.SetSingleParameterValue(1, 1, "General",
Sr2DParameterIdentifier.Autotrigger_0x510001, "1");
```

Note

In the current version, the first two parameters (SettingsVersion and ConfigVersion) are always set to 1.

Executing an Area Image Capture Using LED Trigger

```
var setParamResult = await _vsxProtocolDriver.SetSingleParameterValue(1, 1, "Toolbar",
Sr2DParameterIdentifier.TriggerLED_0x640002, "1");
```

Switching on Image Transfer

```
var ret3 = await _vsxProtocolDriver.SetSingleParameterValue(1, 1, "General", Sr2DParameterIdentifier.ImageTransferActive, "1");
```

Example Program

The example program "ExampleSr2D.cs" is located in the "VSX-Interface libraries..." package. It is intended to make the initial steps of integrating the SmartRunner Explorer easier. The following steps are performed in this example program:

- TCP/IP connection
- Command to start permanent trigger
- Line data is saved in a container with each trigger
- A specific number (number must be determined) of containers can be stored temporarily in the API
- Retrieve line data from container (polling)
- Example access for line data



Examples of Using the Programming Interface

1. Establishing the TCP/IP Connection

First, all available devices are listed via UDP and an attempt is made to connect to the first device. If no device was found via UDP, an attempt is made to establish a connection via a fixed IP address (in this case the default IP).

```
//First discover devices via UDP and use the IP of the first device found
var sensors = await VsxProtocolDriver.UdpDeviceList();
if (sensors.Succ && sensors.DeviceList.Count > 0)
{
    // create a new VsxProtocolDriver instance
    _vsxProtocolDriver = VsxProtocolDriver.Init(sensors.DeviceList[0].IpAd-
dress);
}
else //use fixed IP address
{
    // create a new VsxProtocolDriver instance with fixed IP address
    _vsxProtocolDriver = VsxProtocolDriver.Init("192.168.2.3");
}
// Connect with device
var connRes =await _vsxProtocolDriver.Connect();
```

2. Output of Line Data

Wait for the container to be filled. The time for this can be limited to 500 ms, as in this case. This timeout period should be set depending on the data to be received. If there is only one line, about 40 ms is sufficient. A check is then made as to whether the container exists and whether it contains data — in this case, whether it contains lines. The data of the line is then transferred to the singleLine variable.

```
// try to get a data container within 500 ms
var dataltem = await _vsxProtocolDriver.GetDynamicContainer(500);
// dataltem.Container now contains an image/result/line package
if (dataltem.Succ && (dataltem.Container !=null))
{
    // Check if there is a line message contained
    if (dataltem.Container.ContainsMessage("Line"))
    {
        var lineMessage = dataltem.Container.GetMessage("Line") as Vsx-
LineMessage;
        // Get line from package
        var singleLine = lineMessage.Line;
    }
}
}
```



Note

For the data structure of the "line," .

3. Access to a Line Coordinate

To do this, the container must first be collected, as in example 2. Any line coordinate can then be accessed via the following program line. In this case, it is the first coordinate.

```
Coordinate coord = singleLine[0];
```

4. Output of Area Image Data

Wait for the first container. The time for this can be limited to 500 ms, as in this case. This time-out period can be set depending on the data to be received. A check is then made as to whether the container exists and whether it contains an image. If it contains an image, it is saved to the BMP variable and the image is stored in png format in the Temp directory.

```
// try to get a data container within 500 ms
var dataItem = await_vsxProtocolDriver.GetDynamicContainer(500);
    // Check if dataItem.Container now exists
    if (dataItem.Succ && (dataItem.Container != null))
    {
        // Check if there is an Image message contained
        if (dataItem.Container.ContainsMessage("Image"))
        {
            var Img = dataItem.Container.GetMessage("Image") as VsxImage-
            DataRegionMessage;
            var Bmp = Img.Image;
            Bmp.Save("C:/Temp/Bild.png");
        }
    }
```

6 Vision Configurator Software

The sensor is commissioned and operated using the Vision Configurator software.

The Vision Configurator software makes it easy to operate the sensor with its user-friendly interface. Standard functions include making connections to the sensor, specifying the operating parameters, saving data sets, and displaying data and error diagnostics.



Note

The following user roles are predefined with different authorizations in the Vision Configurator.

User Rights and Password

User rights	Description	Password
Default	View all information Sensor configuration Create users at same or lower level	A password is not required
User	View all information Sensor configuration Create users at same or lower level	User
Admin	View all information Sensor configuration	Request the admin password from Pepperl+Fuchs

Table 6.1 The users have different access and administration rights depending on the respective user role.



Establishing a Network Connection

To establish a network connection with the sensor, proceed as follows:

1. Supply the sensor with power.
2. Start the Vision Configurator software.
3. Enter your user name and password.



Note

Additional steps for user-defined installation and installation of additional components are described in the Vision Configurator manual. The Vision Configurator manual can be found online at www.pepperl-fuchs.com.

6.1 Connecting to Vision Configurator



Connecting Sensors

Connected sensors can be searched for using the "Auto detect" function (1). To do so, proceed as follows:

1. Ensure that the sensor and the PC/laptop are ready for operation and that there is an Ethernet connection.
2. Click on the "Auto detect (TCP/IP only)" button (1) to search for connected sensors.

↳ If a sensor (2) is detected, it appears in the overview window.

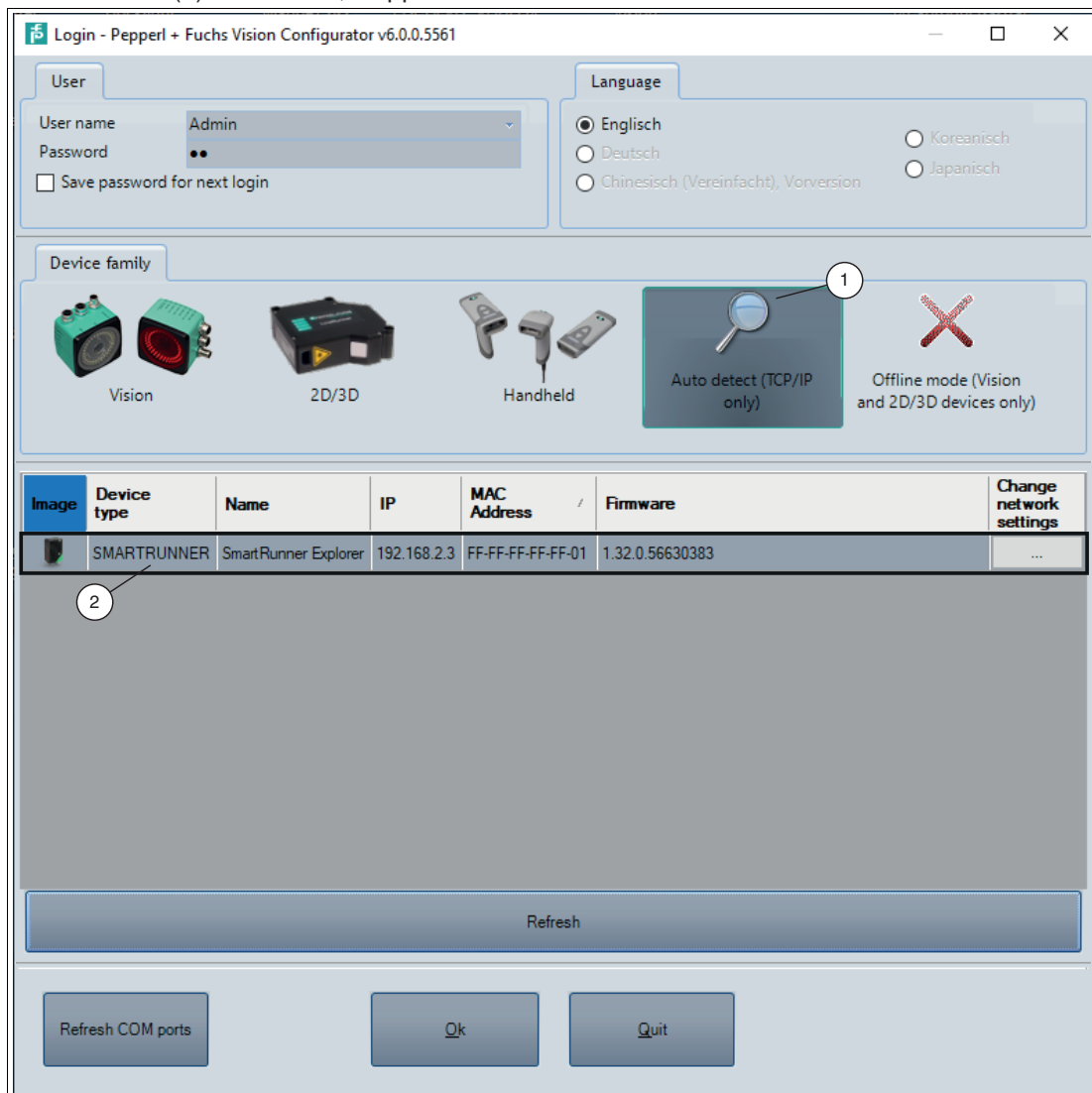


Figure 6.1 "Auto Detect" in Vision Configurator

3. Click on the "OK" button.
↳ The sensor (2) is shown on the home screen of all sensors found.
4. Select the **2-D/3-D** button (1) under the "Device family" tab on the home screen.

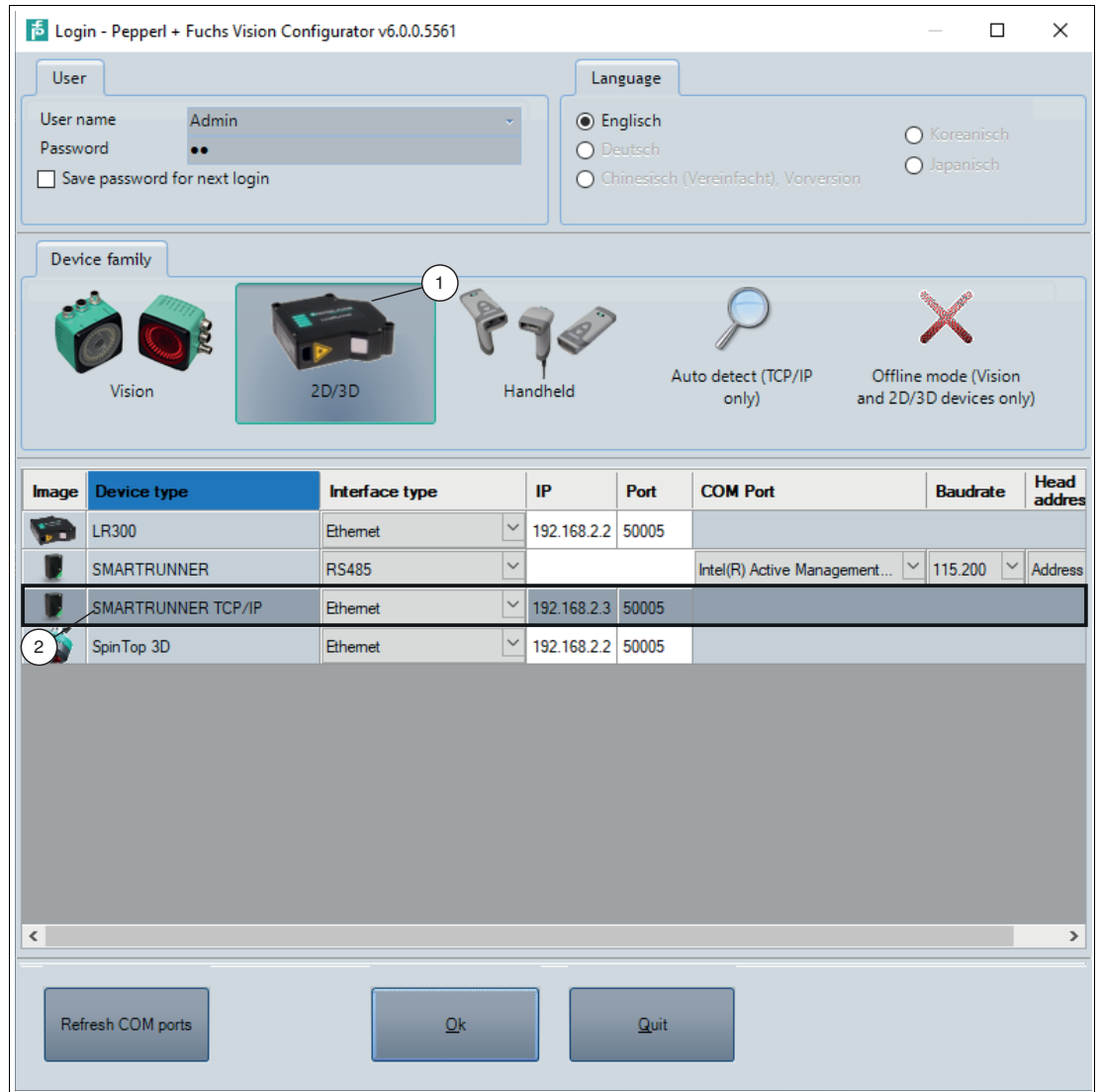


Figure 6.2 Home screen

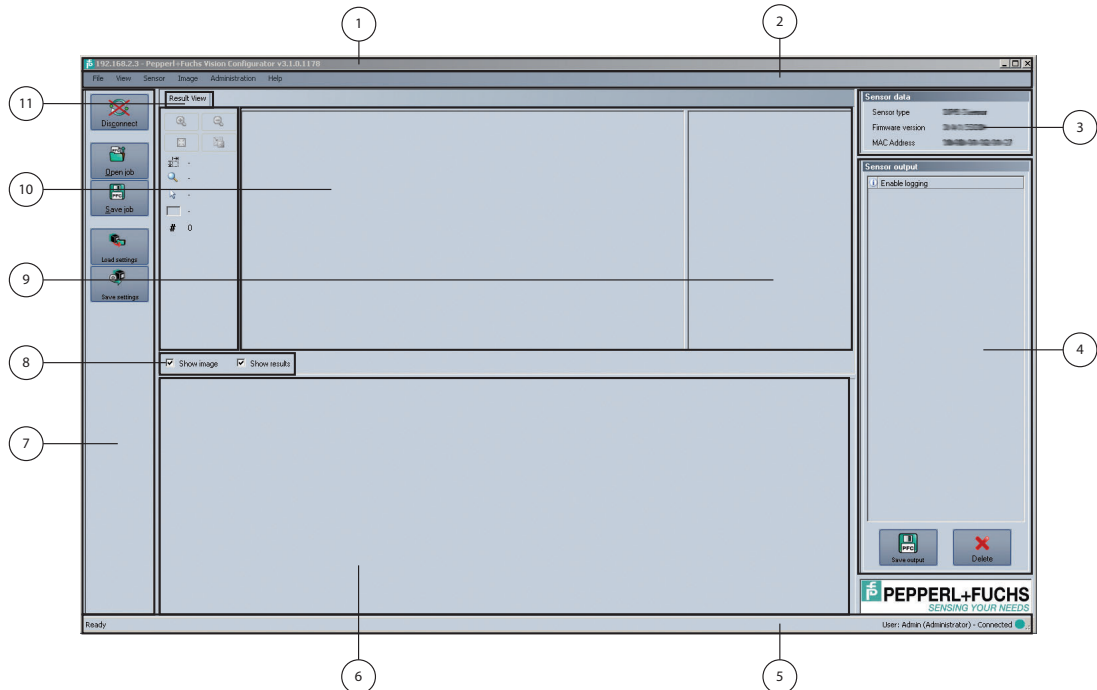
5. Select the sensor "SMARTRUNNER TCP/IP" (2) from the overview window.
 6. Click on the "OK" button.
- ↳ The sensor is connected and appears in the application window.

6.2 Application Window Structure

The application screen opens after you log in.

Note

The individual functions depend on the type of sensor connected and the current authorization level, so they are not always all visible.



The software is designed to be similar to most Windows applications.

1	Title bar	<ul style="list-style-type: none"> Shows the IP address, the software name, and the version number Contains the Minimize/Maximize/Close buttons
2	Menu bar	<ul style="list-style-type: none"> Displays all the menus in the program Provides an overview and helps with navigation
3	Sensor data screen	<ul style="list-style-type: none"> Displays data for the connected sensor
4	Sensor output screen	<ul style="list-style-type: none"> Shows the log display
5	Status bar	<ul style="list-style-type: none"> Displays status information about the application
6	Configuration window	<ul style="list-style-type: none"> Contains the sensor-specific parameters that you can set
7	Toolbar	<ul style="list-style-type: none"> Contains icon buttons as an extension to the menu
8	Check boxes	<ul style="list-style-type: none"> Show images: Enables or disables the image display Show results: Enables or disables the results area
9	Results area	<ul style="list-style-type: none"> Displays results from the sensor A varying number of tabs can be displayed depending on which sensor is connected This field can be enabled or disabled via Show results

10	Image display	<ul style="list-style-type: none"> • Displays the images captured or stored in the error memory • This field can be enabled or disabled via Show images
11	Tab	Displays information about the current image and the pixel under the mouse pointer. The following items are displayed: <ul style="list-style-type: none"> • Image size • Zoom level • Mouse position in image coordinates • Current grayscale value • Image number

6.3 Menu Bar

The menu bar contains a list of menu items. The functionality depends on the type of sensor that is connected and the permissions of the user logged in.

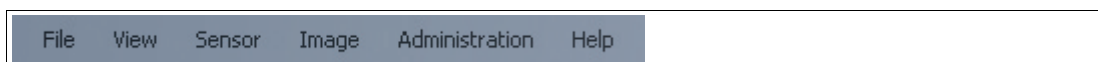


Figure 6.3 Menu Bar

6.3.1 File Menu

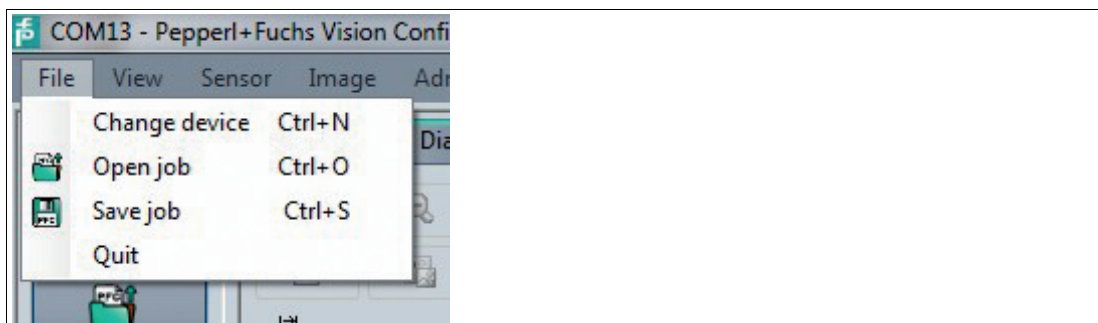


Figure 6.4 File Menu

Change device	Disconnects the device and returns to the Login dialog.
Open job	Loads a sensor configuration stored on the PC.
Save job	Saves the current sensor configuration on the PC.
Quit	Terminates the program.

Table 6.2 File Menu

6.3.2 View Menu

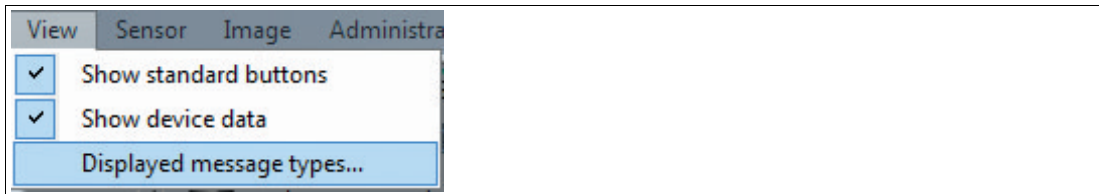


Figure 6.5 View Menu

Show standard buttons	Toggles the display of the buttons in the bar on the left on and off.
Show device data	Hides the display of the sensor data in the top right of the screen.
Displayed message types...	Opens a selection window in which the following display windows can be activated or deactivated: Info, Result OK, Result not OK, Warning, Error, Critical, Assert.

Table 6.3 View menu

6.3.3 Sensor Menu

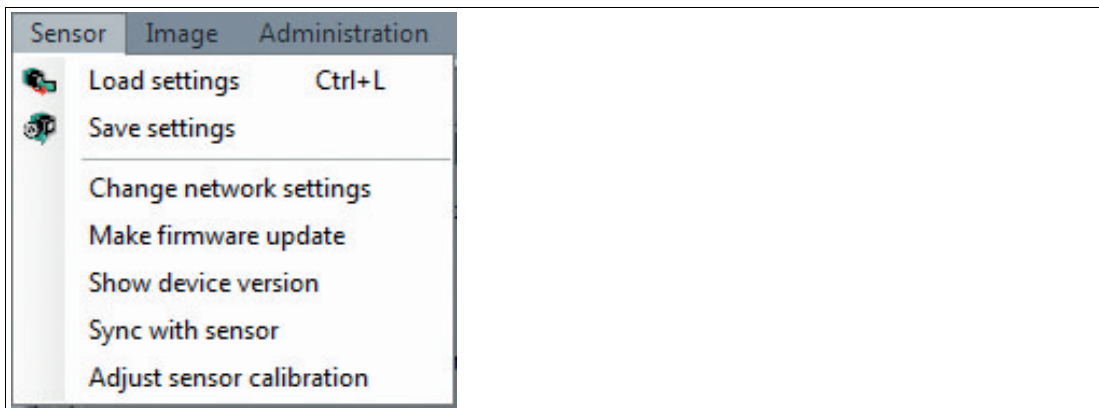


Figure 6.6 Sensor menu

Load settings	Loads the saved settings from the sensor
Save settings	Saves the settings to the sensor
Change network settings	Change the network settings. The settings window allows you to set the IP address, subnet mask, gateway address, and DHCP
Make firmware update	Performs firmware updates. This command should be used by experienced users only
Show device version	Displays the device version
Sync with sensor	Synchronization with the sensor
Adjust sensor calibration	Adjust the sensor calibration

Table 6.4 Sensor menu

Note

Firmware Update

Once you have upgraded the firmware and **Update complete** is displayed, restart the sensor.



6.3.4 Image Menu



Figure 6.7 Image menu

Load imagefile	Loads the image file
Open image folder	Opens the folder in which images are currently saved
Save image	Saves the image currently displayed on the PC
Copy image to clipboard	Loads an image file to the clipboard
Upload image to device	Uploads an image to the device
Show graphic	Switches display data sent from the sensor on and off in the image.

Table 6.5 Image menu

6.3.5 Administration Menu

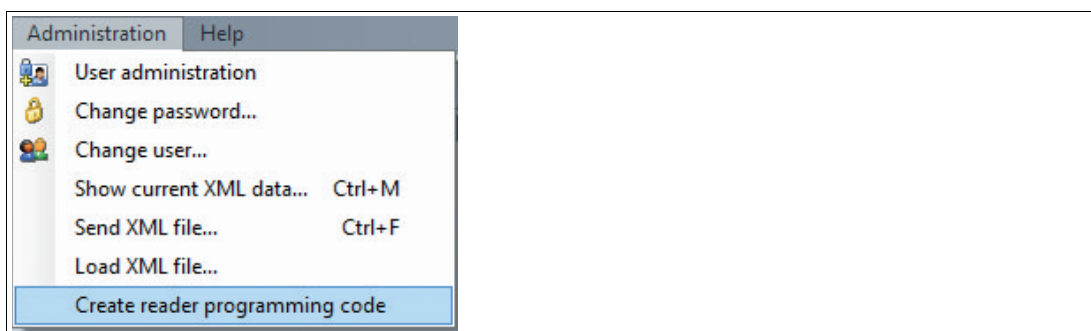


Figure 6.8 Administration menu

User administration	Opens a window that shows all currently created users at the same authorization level or lower. New users at the same authorization level or lower can also be created and deleted here. In addition, a user password can be reset to the default password for the relevant user level.
Change password	Changes the current user's password.
Change user	The login screen opens and a different user and/or sensor can be selected.
Send XML file...	Saves the XML data on a computer.
Load XML file...	Loads XML data from a computer.
Create reader programming code	Creates a reader programming code

Table 6.6 Administration menu

6.3.6 Help Menu

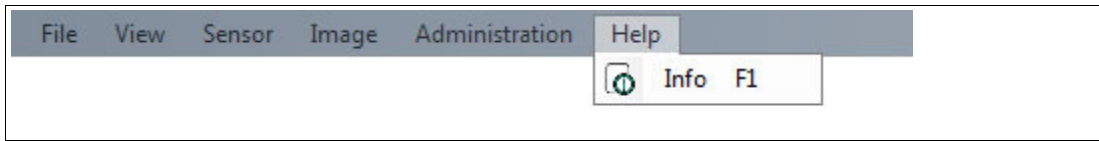




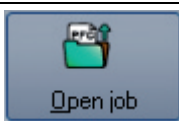
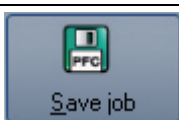



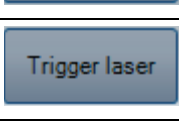
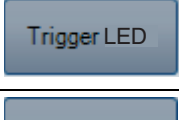
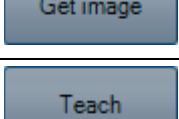
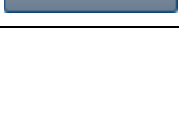
Figure 6.9 Help menu

Info	Displays information about Vision Configurator.
-------------	---

Table 6.7 Help menu

6.4 Toolbar

The toolbar can be used to select various functions.

 Connect	Selecting the Connect button establishes a connection between the PC and the sensor.
 Disconnect	The connection between the PC and the sensor is disconnected.
 Open job	Opens a saved setting.
 Save job	Saves the settings made.
 Load settings	Settings are read out from the sensor.
 Save settings	All settings made are saved on the sensor.
 Reset	Reset to default settings.
 Trigger laser	Perform laser trigger to take a line image.
 Trigger LED	Perform LED trigger to take an area image. NOTE: With autotrigger activated, a line image is output with "Trigger LED," not an area image.
 Get image	Loads the current sensor image.
 Teach	Teaches the exposure time.

2022-07

6.5 Device Data

The "Device data" area shows the connected device type, firmware version, and MAC ID.



Figure 6.10 Device data

6.6 Image and Line Display

Pictures can be analyzed in the image and line display. By analyzing the recorded profile form, it is possible to use the measurement result to make a qualitative assessment. This enables intrusive reflections to be identified and eliminated. There is a relationship between the exposure time and image blur. A correct exposure is dependent on the brightness of the profile and the incident amount of light. An exposure time that is too short leads to underexposed (too dark) images, an exposure time that is too long to overexposed images.

There are various options available to you to display and correct recorded data to avoid errors during recording.

Result View - Line Display

You can open the currently recorded image including height profile under the **Result View** tab. To do this, click on the **Trigger laser** buttons in the toolbar.

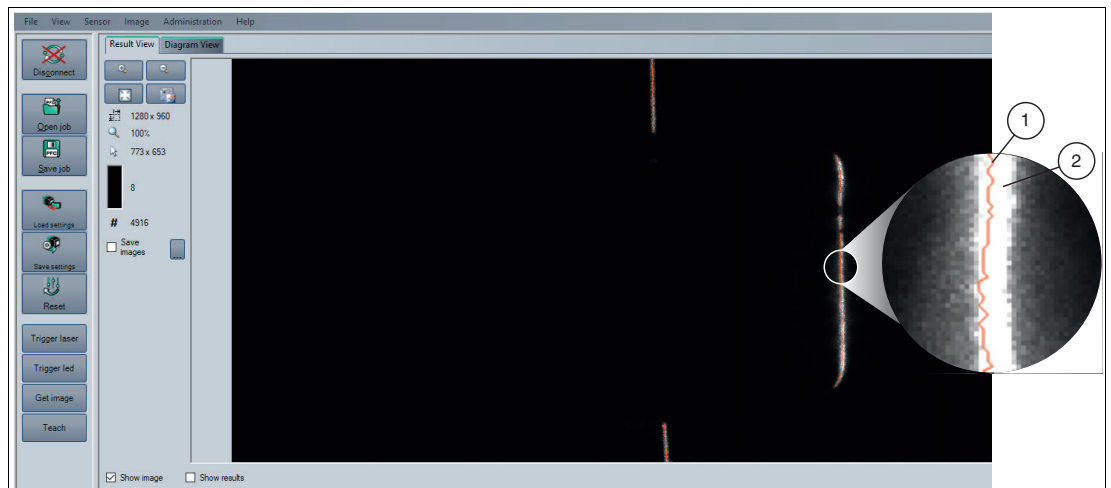


Figure 6.11 Line display

- 1 The line found is shown in red.
- 2 The laser light is shown in white.

The following context menu appears when you right-click the recorded image:



Figure 6.12 Result View context menu screen

Designation	Function
Load image file...	Loads a sensor image. You can select the sensor image.
Open image folder	Opens the storage location
Copy image to clipboard	Copies image to the clipboard
Save image	Saves the displayed sensor image

Result View - Area Image

You can open the currently recorded image including height profile under the **Result View** tab. To do this, click on the **Trigger LED** buttons in the toolbar.

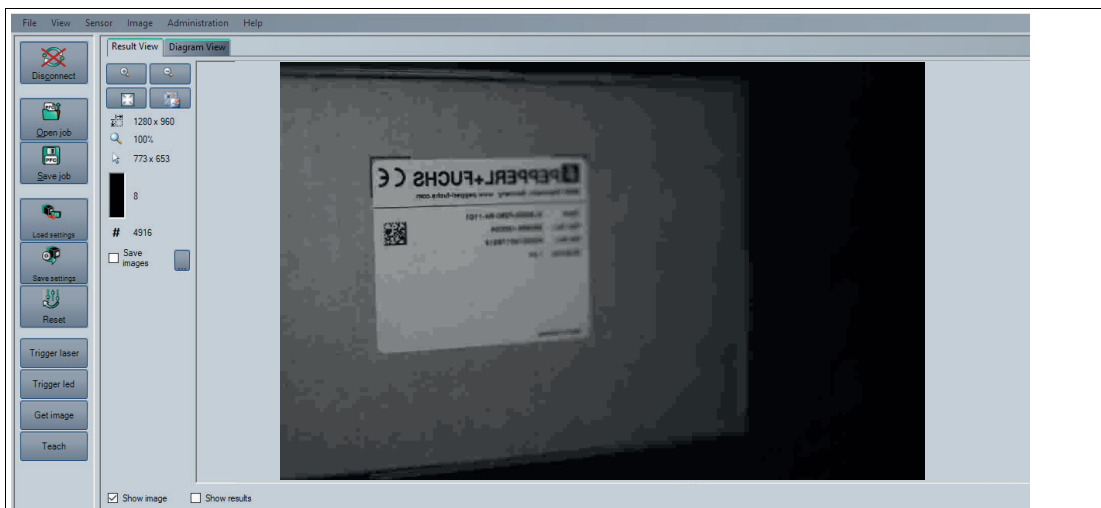


Figure 6.13 Area image



Note

Note that the "Image Transfer active" (see "Image Menu Item" on page 56) function is activated to show the area image.

The following context menu appears when you right-click the recorded image:

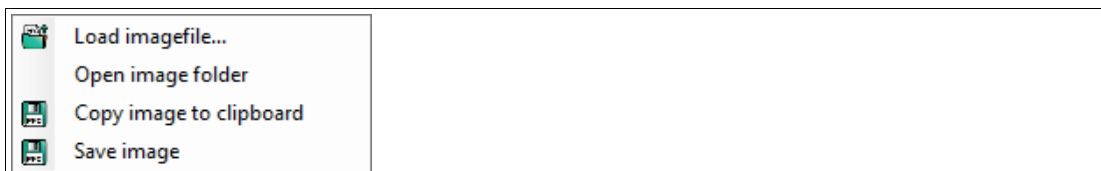


Figure 6.14 Result View context menu screen

Designation	Function
Load image file...	Loads a sensor image. You can select the sensor image.
Open image folder	Opens the storage location
Copy image to clipboard	Copies image to the clipboard
Save image	Saves the displayed sensor image

Result View - Toolbar Menu

The toolbar is located on the left side under the **Result View** tab. The toolbar contains several useful functions that are used to further process recorded images. The following functions are available.

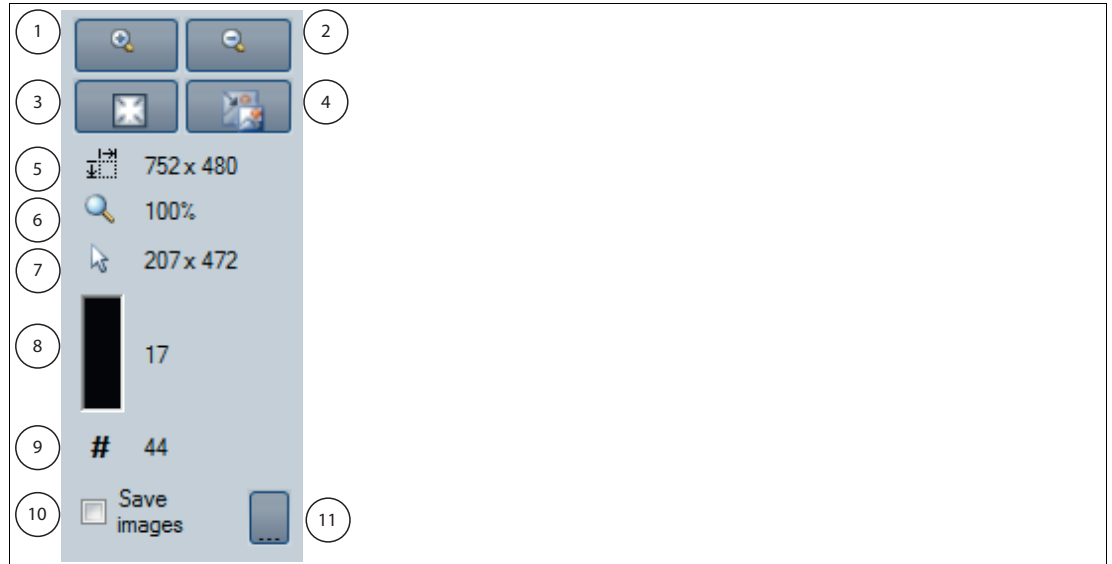


Figure 6.15 Toolbar

Position	Designation	Function
1	Magnifier +	Zoom in
2	Magnifier -	Zoom out
3	Fit to window	Fit image size to the window
4	Original size	Set original image size
5	Size details	Information field for image size
6	Zoom factor	Information field for zoom factor. A zoom factor of 100 % is the original image size
7	Position details	Shows the position of the mouse cursor
8	Gray scale value details	Gray scale value details for the pixel indicated by the mouse cursor
9	Image counter	Displays the current image number
10	Save image	Saves image following transfer
11	Select path	Select path on the storage medium

Diagram View

Under the **Diagram View** tab you can open the graphical view of the result data (height profile). To do this, you must first select whether you want to open the point view (checkmark out) or the line view (checkmark in) via the **Interpolate points** check box. Then click on the **Trigger laser** buttons in the toolbar. The graphical view of the height profile is displayed under **Diagram View**.

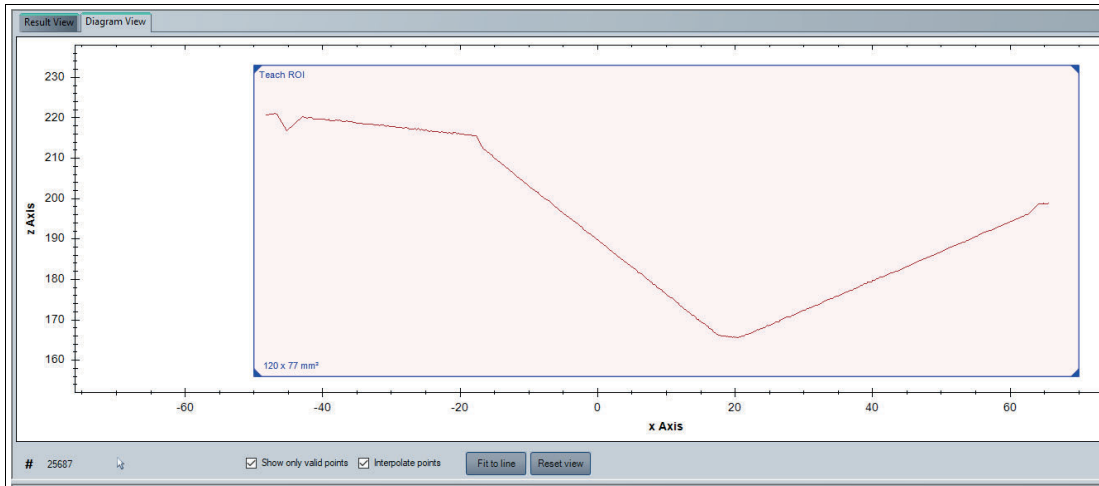


Figure 6.16 Diagram View - line view

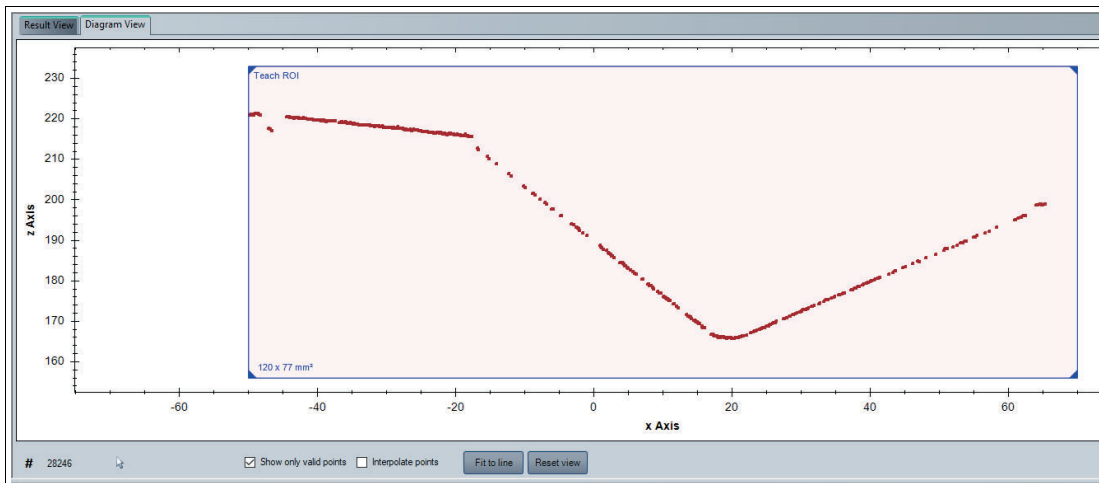


Figure 6.17 Diagram View - point view

The following context menu appears when you right-click the graphic:



Figure 6.18 Diagram View context menu screen

Designation	Function
Copy	Copies diagram into working memory
Save Image As...	Saves diagram to hard drive
Page Setup...	Page setup for print function
Print...	Print diagram
Show Point Values	Shows the values of the discrete line points in world coordinates [mm] as tooltip.
Un-Zoom	Undo the last zoom action
Undo All Zoom/Pan	Undo all the zoom and pan actions
Set Scale to Default	Scales the measure using the line data

Diagram View - Toolbar

The toolbar is located below the diagram view. The toolbar contains several useful functions that are used to further process the diagrams. The following functions are available.

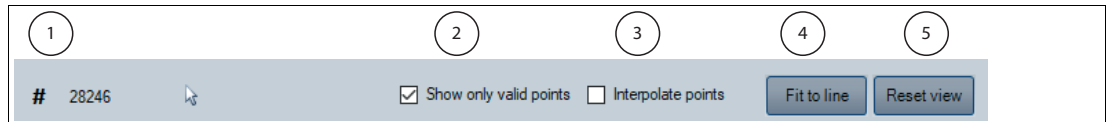


Figure 6.19 Diagram View - Toolbar

Position	Designation	Function
1	Grayscale value	Grayscale value of the pixel
2	Show only valid points	Shows all available points
3	Interpolate points	<ul style="list-style-type: none"> Function selected: Line view Function deselected: Point view
4	Fit to line	Entire visible line in field of view
5	Reset view	Resets to the original view

6.7 Configuration window

Various parameters are specified in the configuration window. The individual parameters depend on the current authorization level and are, therefore, not always all visible. Some features are available in different variants only. Depending on the parameters set, some fields will be grayed out.

6.7.1 Sensor Information Tab

The **Sensor information** tab contains the **Sensor information** menu items. The **Sensor information** menu item allows you to view more detailed information on the sensor. .

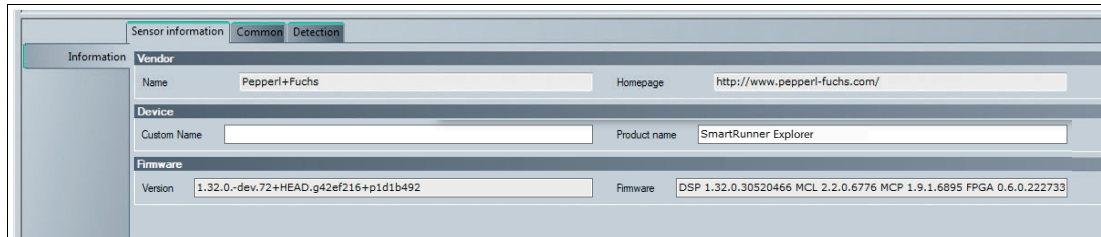


Figure 6.20 Sensor Information Tab

Vendor	Name	Manufacturer
	Homepage	Manufacturer homepage
Device	Custom name	User-defined name
	Product name	Product designation
Firmware	Version	Firmware version
	Firmware	Firmware designation

6.7.2 Common Tab

Five menu items are available under the **Common** tab. The purpose of this section is to present the menu items in detail.

Illumination Menu Item

You can adjust the sensor's exposure under the **Illumination** menu item.

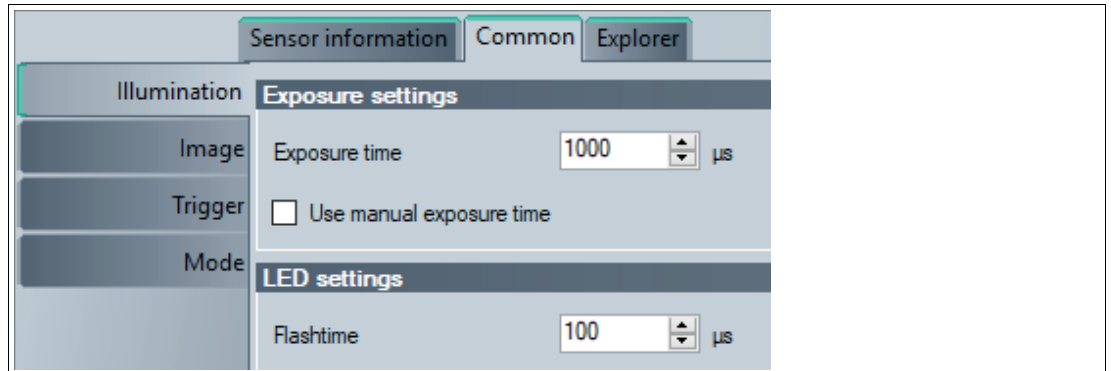


Figure 6.21 Illumination menu item

Designation	Function
Exposure time	Setting the manual exposure time. The "Use manual exposure time" function must be activated to manually adjust the exposure time. Increasing the value increases the exposure time and therefore the image brightness. Values below 1000 µs are suitable in most cases
Use manual exposure time	Function activated: the set exposure time is used. Function disabled: the exposure time is controlled automatically (Teach > Trigger laser). On the right side of the window, the automatically set exposure time is displayed under "Device output".
Flashtime	Sets the exposure time of the LED lighting in µs.

Image Menu Item

Under the **Image** menu item, you can adjust the image capture.

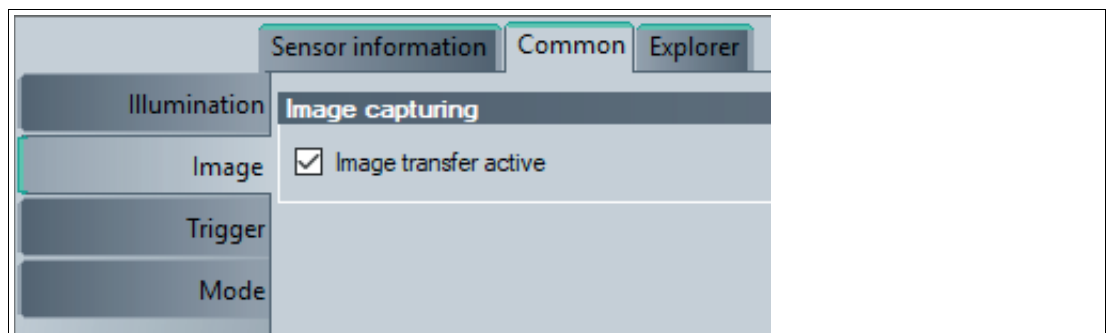


Figure 6.22 Image menu item

Designation	Function
Image transfer active	This function transfers images automatically with each image capture. If the function is disabled, no images are output in the "Result View."

Trigger Menu Item

You can enable or disable the autotrigger under the **Trigger** menu item.



Figure 6.23 Trigger menu item

Designation	Function
Autotrigger	Checking the box activates a cyclically recurring trigger. The autotrigger must be activated in "Presentation mode". NOTE: With autotrigger activated, a line image is output with "Trigger LED," not an area image.

Mode Menu Item

Under the **Mode** menu item, you can enable or disable "Presentation mode" and "function keys 1 and 2". "Presentation mode" and "function keys 1 and 2" are activated if checked and deactivated if unchecked.

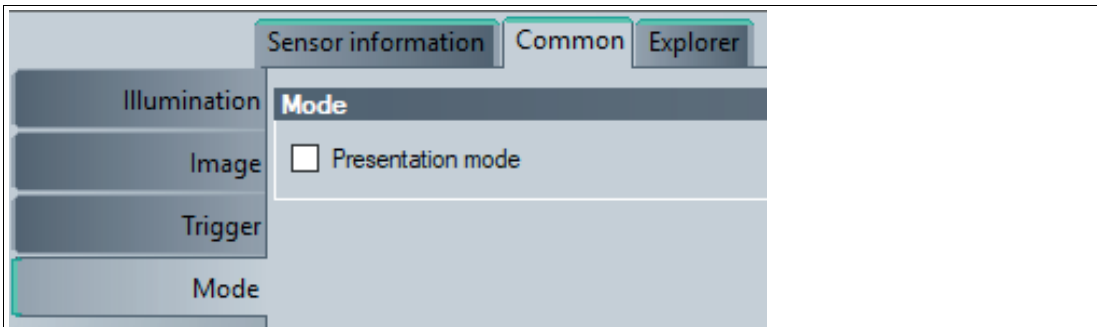


Figure 6.24 Mode menu item

Designation	Function
Presentation mode	Operating mode for presentation or testing without the assistance of a PC

6.7.3 Explorer Tab

Two menu items are available under the **Matcher** tab. The purpose of this section is to present the menu items in detail.

Line Menu Item

Under the **Line** menu item, you can adjust the x and z tolerance range for the sensor. The tolerance range determines by how much the height profile may move within the evaluation range and still be recognized.

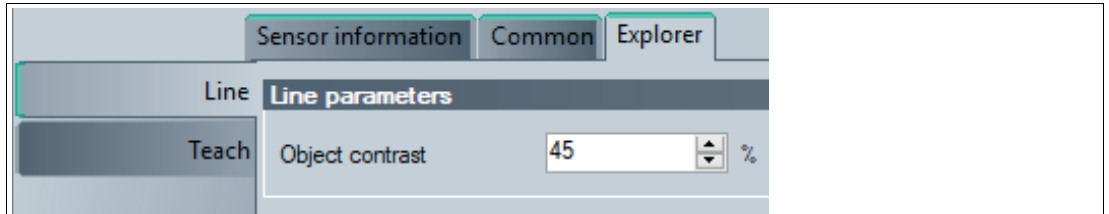


Figure 6.25 Match menu Item

Designation	Function
Object contrast	Contrast threshold used to detect the laser line on the object.

Teach Menu Item

The **Teach** menu item is used to adjust the exposure time. The required "Teach ROI" teach-in range is adjusted using the line profile under the "Diagram View" tab. The coordinates of the x and z axis are shown in the display field below the graphic.

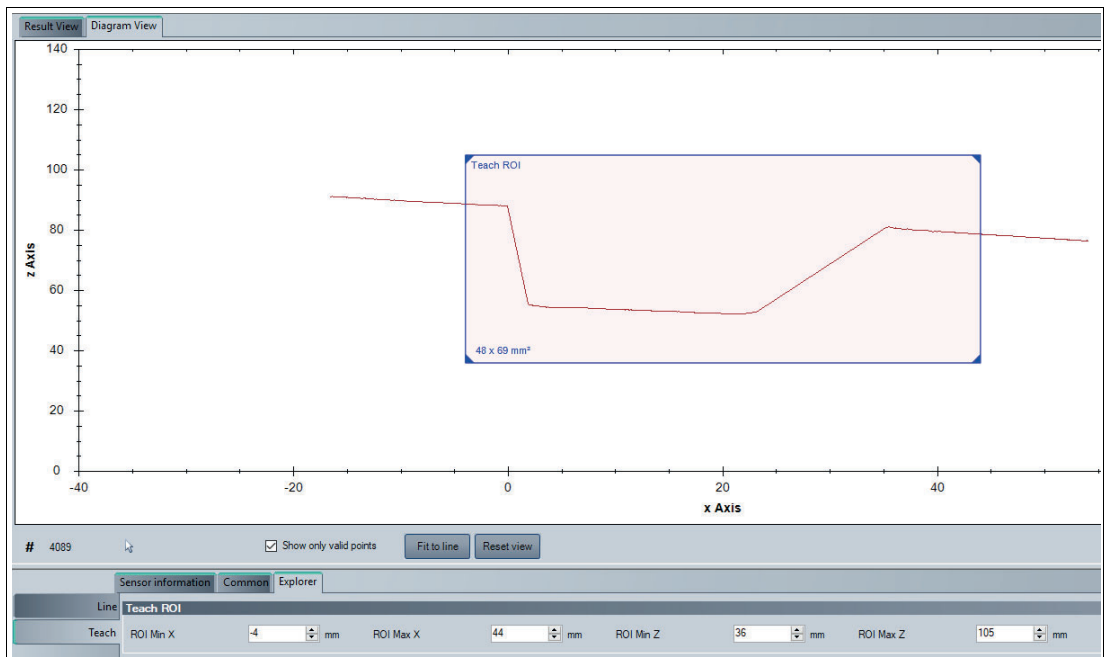


Figure 6.26 Teach Menu Item

Designation	Function
ROI Min X	The smallest value on the x axis
ROI Max X	The largest value on the x axis
ROI Min Z	The smallest value on the z axis
ROI Max Z	The largest value on the z axis

7 Maintenance and Repair

7.1 Servicing



Danger!

Danger to life due to electrical current!

Contact with live parts causes immediate danger to life.

- Allow only qualified electricians to carry out work on the electrical installation.
 - Switch off the power supply before carrying out servicing, cleaning, and repairs, and prevent the supply from being switched on again.
 - Keep the live parts free from moisture.
-

The device is maintenance-free. To get the best possible performance out of your device, keep the optical unit on the device clean, and clean it when necessary.

Observe the following instructions when cleaning:

- Do not touch the optical unit with your fingers.
- Do not immerse the device in water. Do not spray the device with water or other liquids.
- Do not use abrasive agents to clean the surface of the device.
- Use a cotton or paper cloth moistened (not soaked) with water or isopropyl alcohol.
- Remove any residual alcohol using a cotton or paper cloth moistened (not soaked) with distilled water.
- Wipe the device surfaces dry using a lint-free cloth.

7.2 Repair

The device must not be repaired, changed, or manipulated. In case of failure, always replace the device with an original device.

8 Troubleshooting

8.1 What to Do in Case of a Fault

Before you have the device repaired, perform the following actions:

Checklist

Fault	Cause	Remedy
"Power" LED does not light up	The power supply is switched off	Check whether there is a reason why the power supply is switched off (installation or maintenance work, etc.). Switch on the power supply if necessary.
	Wiring fault in the splitter or switch cabinet, cable break	Check the wiring carefully and repair any faults with the wiring. Check the cables to ensure that they are functioning properly.
Control panel receiving no measurement data	Connection cable not connected	Connect the connection cable.
	Incorrect connection cable used	Use the appropriate connection cable only.
Measurement object not recognized	Reflections	Avoid reflections
	Foreign exposure	Avoid foreign exposure
	Exposure time control	Set exposure (see chapter 6.7.2)
	Teach-in range set incorrectly	Set teach-in range (see chapter 6.7.3)
Measurement errors	Surfaces with pronounced scored structure and reflective surfaces	Improved arrangement of sensor components to the measurement object
	Temperature change in the sensor	Allow sensor to warm up for around 15 minutes before the measuring process is started.
	Incorrect distance to the measuring object	Note distance values
	Housing incorrectly mounted	Install housing correctly (see chapter 3.4)
Presentation mode not working	Presentation mode not activated	Enable Presentation mode and Autotrigger and confirm using "Save settings"
No connection to the sensor	Alternating-current voltage or supply voltage too high	Connect sensor to direct-current voltage (DC) only. Ensure that the supply voltage level is within the specified sensor range.

- If none of the above remedies the problem, please contact our service center. Please have the error images and the version number of the firmware available. The firmware version number can be found at the top right of the user interface.

Your automation, our passion.

Explosion Protection

- Intrinsic Safety Barriers
- Signal Conditioners
- FieldConnex® Fieldbus
- Remote I/O Systems
- Electrical Ex Equipment
- Purge and Pressurization
- Industrial HMI
- Mobile Computing and Communications
- HART Interface Solutions
- Surge Protection
- Wireless Solutions
- Level Measurement

Industrial Sensors

- Proximity Sensors
- Photoelectric Sensors
- Industrial Vision
- Ultrasonic Sensors
- Rotary Encoders
- Positioning Systems
- Inclination and Acceleration Sensors
- Fieldbus Modules
- AS-Interface
- Identification Systems
- Displays and Signal Processing
- Connectivity

Pepperl+Fuchs Quality

Download our latest policy here:

www.pepperl-fuchs.com/quality

