MICRO-EPSILON

Operating Instructions

eddyNCDT 3020

ES-U1-C-CAx/mB0	ES-S1-C-CAx/mB0	ES-S2-C-CAx/mB0	ES-U3-T-CAx/mB0	ES-U6-C-CAx/mB0
ES-U1-T-CAx/mB0	ES-U2-C-CAx/mB0	ES-U3-C-CAx/mB0	ES-S4-C-CAx/mB0	ES-U8-C-CAx/mB0

Non-contact compact displacement measuring system based on eddy currents

MICRO-EPSILON MESSTECHNIK
GmbH & Co. KG
Koenigbacher Str. 15

94496 Ortenburg / Germany

Tel: +49 (0) 8542 / 168-0
Fax: +49 (0) 8542 / 168-90
info@micro-epsilon.com
https://www.micro-epsilon.com

Contents

1 Safety

1.1 Symbols used

System operation assumes knowledge of the operating instructions.

The following symbols are used in these operating instructions:

⚠ CAUTION	Indicates a hazardous situation which, if not avoided, may result in minor or moderate injury.
NOTICE	Indicates a situation that may result in property damage if not avoided.
►	Indicates a user action.
i	Indicates a tip for users.
`Measurement`	Indicates hardware or a software button/menu.

1.2 Warnings

⚠ CAUTION

Connect the power supply according to the regulations for electrical equipment.

- Risk of injury
- Damage to or destruction of the sensor and/or the controller

NOTICE

Avoid shocks and impacts to the sensor and the controller.

- Damage to or destruction of the sensor and/or the controller

The supply voltage must not exceed the specified limits.

- Damage to or destruction of the sensor and/or the controller

Protect the sensor cable against damage.

- Destruction of the sensor
- Failure of the measuring device

1.3 Notes on product marking

1.3.1 CE marking

The following apply to the product:

- Directive 2014/30/EU ("EMC")
- Directive 2011/65/EU ("RoHS")

Products which carry the CE marking satisfy the requirements of the EU Directives cited and the relevant applicable harmonized European standards (EN).

The product is designed for use in industrial and laboratory environments.

The EU Declaration of Conformity and the technical documentation are available to the responsible authorities according to the EU Directives.

1.3.2 UKCA marking

The following apply to the product:

- SI 2016 No. 1091 ("EMC")
- SI 2016 No. 1101 ("Low Voltage")
- SI 2012 No. 3032 ("RoHS")

Products which carry the UKCA marking satisfy the requirements of the directives cited and the relevant applicable harmonized standards.

The product is designed for use in industrial and laboratory environments.

The UKCA Declaration of Conformity and the technical documentation are available to the responsible authorities according to the UKCA Directives.

1.4 Intended use

The DT3020 is used for

- Displacement, distance and thickness measurements
- Position measurement of parts or machine components

The measuring system is designed for use in industrial and laboratory applications.

► The measuring system must be used in such a way that no persons are endangered or machines are damaged in the event of malfunction or total failure of the sensor.

► Take additional precautions for safety and damage prevention in case of safety-related applications.

1.5 Proper environment

Protection class:	
- Sensor, sensor cable:	IP68 (connected)
- Controller:	IP67 (plugged)
Temperature range Sensor, sensor cable:	
- Operation:	-20 ... +180 °C, applies for ES-U1, ES-U1-T -20 ... +200 °C
- Storage:	-20 ... +180 °C, applies for ES-U1, ES-U1-T -50 ... +200 °C
Temperature range Controller:	
- Operation:	-20 … 105 °C (non-condensing)
- Storage:	-20 … 105 °C (non-condensing)
Humidity:	5 % RH ... 95 % RH (non-condensing)
Ambient pressure:	Atmospheric pressure

2 Functional principle, technical data

2.1 Field of application

The non-contact compact displacement measuring system is designed for industrial use in production lines, for machine monitoring and for measuring and testing during inline quality assurance.

2.2 Measuring principle

The eddyNCDT (Non-Contacting Displacement Transducers) displacement measuring system operates without contact based on eddy currents. It is used for measurements on objects made of electrically conductive materials, which may have ferromagnetic and non-ferromagnetic properties. A high frequency alternating current is passed through a coil installed in a sensor housing. The electromagnetic field of the coil induces eddy currents in the conducting measuring object, whereby the resulting impedance of the coil changes. This change of impedance causes an electrical signal which is proportional to the distance between the measuring object and the sensor. An electronic compensation method reduces temperature-dependent measurement errors to a minimum.

2.3 Structure of the complete measuring system

The non-contact single channel displacement measuring system consists of:

- Sensor
- Sensor cable
- Connection cables
- Controller
- Factory calibration

i The components are calibrated with one another at the factory. The assignment of sensor and controller determines the serial number. The coordinated components and serial numbers are denoted by labels on the side of the controller.

eddyNCDT 3020 controller and sensors

2.4 Term definitions, analog output displacement

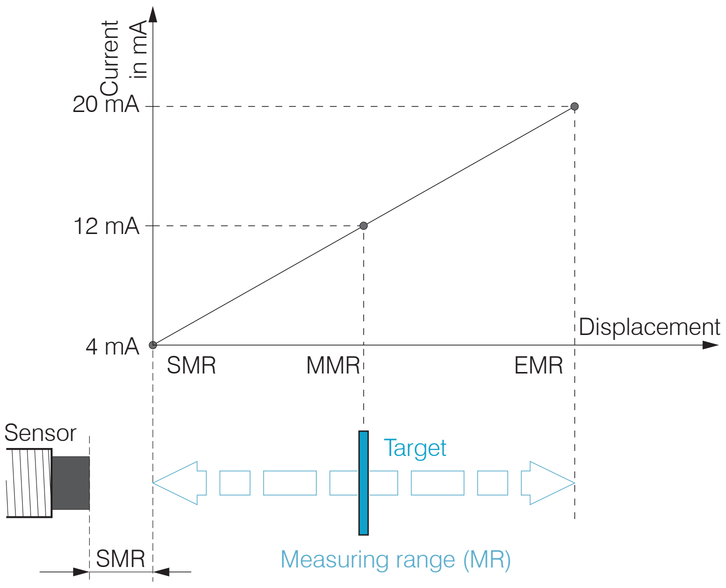


Fig. 2.1: Measuring range and output signal measuring system

SMR	Start of measuring range Minimum distance between the front sensor face and the target, sensor-specific
MR	Mid of measuring range
EMR	End of measuring range (start of measuring range + measuring range) Maximum distance between the front sensor face and the target
MR	Measuring range

2.5 Technical data

2.5.1 DT3020

Model		DT3020
Resolution [1]	Static	0.004 % FSO
	Dynamic	0.01 % FSO
Frequency response (-3dB) [2]		9 adjustable stages: 10 Hz ... 5 kHz
Measuring rate	Analog output	80 kSa/s
	Digital output	10 kSa/s
Linearity [3]		< ±0.2 % FSO
Temperature stability [4]		< 0.025 % FSO / K
Temperature compensation		10 ... 105 °C
Target material [5]		Steel, aluminum
No. of characteristic curves		1
Supply voltage		12 ... 32 VDC
Power consumption		< 1.7 W
Digital interface [6]		RS485 / USB / Ethernet / EtherCAT / PROFINET / EtherNet/IP
Analog output		4 ... 20 mA (max. 500 Ω load, freely scalable 0 ... 20 mA)

[1] FSO = Full Scale Output, RMS noise relates to the mid of the measuring range, static: 20 Hz, dynamic: 5 kHz
[2] Factory setting 5 kHz
[3] Value valid with 3-point linearization
[4] Value valid in the temperature-compensated range
[5] Steel: St37 1.0037; Aluminum: AlMg3 3.3535
[6] Connection via an interface module is required for USB, Ethernet, EtherCAT, PROFINET and EtherNet/IP

Model		DT3020
Switching output		Selectable: NPN, PNP, push-pull
Connection		Sensor: plug connector triaxial socket; supply/signal: 8-pole M12 connector
Mounting		Through-bores (Ø 4.4 mm)
Temperature range	Storage	-20 … 105 °C (non-condensing)
	Operation	-20 … 105 °C (non-condensing)
Shock (DIN EN 60068-2-27)		15 g / 6 ms in 3 axes, 2 directions and 1000 shocks each
Vibration (DIN EN 60068-2-6)		5 g / 10 … 500 Hz in 3 axes, 2 directions and 10 cycles each
Protection class (DIN EN 60529)		IP67 (plugged)
Material		Aluminum die-cast
Weight		approx. 190 g
Control and indicator elements [7]		Configurable via sensorTOOL software: 3-point linearization, scaling of the analog output, filter & averaging, interface selection

2.5.2 DT3020 sensors

Model		ES-U1	ES-U1-T	ES-S1	ES-U2	ES-S2
Measuring range		1 mm		1 mm	2 mm	2 mm
Start of measuring range		0.1 mm		0.1 mm	0.2 mm	0.2 mm
Resolution [8]		0.04 µm		0.04 µm	0.08 µm	0.08 µm
Linearity [9]		< ±2 µm		< ±2 µm	< ±4 µm	< ±4 µm
Temperature stability [10]		< 0.25 µm / K		< 0.25 µm / K	< 0.5 µm / K	< 0.5 µm / K
Temperature compensation		10 … 180 °C				
Sensor type		unshielded		shielded	unshielded	shielded
Recommended target size (flat)		Ø 18 mm		Ø 12 mm	Ø 24 mm	Ø 18 mm
Connection [11]		integrated cable, axial, standard length 3 m; 1 m, 6 m, 9 m optional				
Mounting		Screw connection (M6x0.5)	Clamp mounting (Ø 6 mm)	Screw connection (M8x1)	Screw connection (M8x1)	Screw connection (M12x1)
Temperature range	Storage	-20 … 180 °C		-20 … 200 °C		
	Operation	-20 … 180 °C		-20 … 200 °C		
Pressure resistance		20 bar (front & rear)				
Shock (DIN EN 60068-2-27)		15 g / 6 ms in 3 axes, 2 directions and 1000 shocks each				
Vibration (DIN EN 60068-2-6)		15 g / 49.85 ... 2000 Hz in 3 axes ±3 mm / 10 ... 49.85 Hz in 3 axes				
Protection class (DIN EN 60529)		IP68 (plugged)				
Material		Stainless steel and plastic				
Weight [12]		approx. 2.4 g		approx. 2.4 g	approx. 4.7 g	approx. 11 g

Model	ES-U3	ES-U3-T	ES-S4	ES-U6	ES-U8
Measuring range	3 mm		4 mm	6 mm	8 mm
Start of measuring range	0.3 mm		0.4 mm	0.6 mm	0.8 mm
Resolution [8]	0.12 µm		0.16 µm	0.24 µm	0.32 µm
Linearity [9]	< ±6 µm		< ±8 µm	< ±12 µm	< ±16 µm
Temperature stability [10]	< 0.75 µm / K		< 1 µm / K	< 1.5 µm / K	< 2 µm / K

[7] Access to sensorTOOL requires connection to PC via an interface module

[8] RMS value of the signal noise, static (20 Hz); valid for operation with DT3020; related to the nominal measuring range and in the mid of the measuring range; in the compensated temperature range

[9] Valid for operation with DT3020 and 3-point linearization; related to the nominal measuring range

[10] Valid for operation with DT3020; related to the nominal measuring range and in the mid of the measuring range; in the compensated temperature range

[11] Length tolerance cable: nominal value 0 % / + 30 %

[12] Weight of sensor without nuts, without cable

Model		ES-U3	ES-U3-T	ES-S4	ES-U6	ES-U8
Temperature compensation		10 … 180 °C				
Sensor type		unshielded		shielded	unshielded	unshielded
Recommended target size (flat)		Ø 36 mm		Ø 27 mm	Ø 54 mm	Ø 72 mm
Connection [11]		integrated cable, axial, standard length 3 m; 1 m, 6 m, 9 m optional				
Mounting		Screw connection (M12x1)	Clamp mounting (Ø 12 mm)	Screw connection (M18x1)	Screw connection (M18x1)	Screw connection (M24x1.5)
Temperature range	Storage	-20 … 200 °C				
	Operation	-20 … 200 °C				
Pressure resistance		20 bar (front & rear)				
Shock (DIN EN 60068-2-27)		15 g / 6 ms in 3 axes, 2 directions and 1000 shocks each				
Vibration (DIN EN 60068-2-6)		15 g / 49.85 ... 2000 Hz in 3 axes ±3 mm / 10 ... 49.85 Hz in 3 axes				
Protection class (DIN EN 60529)		IP68 (plugged)				
Material		Stainless steel and plastic				
Weight [12]		approx. 12 g		approx. 30 g	approx. 33 g	approx. 62 g

[11] Length tolerance cable: nominal value 0 % / + 30 %

[12] Weight of sensor without nuts, without cable

3 Delivery

3.1 Unpacking, included in delivery

- 1 sensor incl. sensor cable
- 1 controller incl. factory calibration
- 1 test report
- 1 setup guide

► Carefully remove the components of the measuring system from the packaging and ensure that the goods are forwarded in such a way that no damage can occur.

► Check the delivery for completeness and shipping damage immediately after unpacking.

► If there is damage or parts are missing, immediately contact the manufacturer or supplier.

Return of packaging

Micro-Epsilon Messtechnik GmbH & Co. KG offers customers the opportunity to return the packaging of products purchased from Micro-Epsilon by prior arrangement so that it can be reused or recycled.

To arrange the return of packaging, for questions about the costs and / or the exact return procedure, please contact us directly at

info@micro-epsilon.de

3.2 Storage

Temperature range:	-20 … 105 °C (non-condensing)
- Sensors	-20 ... +180 °C, applies for ES-U1, ES-U1-T -20 ... +200 °C
- Controller	-20 … 105 °C (non-condensing)
Humidity:	5 % RH ... 95 % RH (non-condensing)

4 Installation and mounting

4.1 General

No sharp or heavy objects should be allowed to affect the sensor, supply and output cables.

Note	A damaged cable cannot be repaired. Tension on the cable is not permitted!

4.1.1 Models

The eddyNCDT measuring system is used with shielded or unshielded sensors.

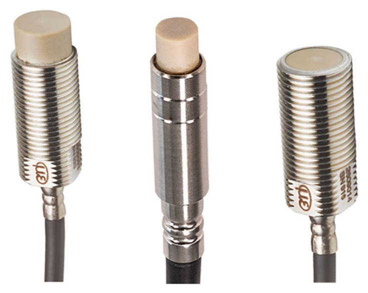


Fig. 4.1: Unshielded sensors with thread (left), without thread (center), shielded sensors (right)

Unshielded sensors

- Type designation: ES-Ux
- Structure: The sensor cap with embedded coil consists of electrically non-conductive materials.

Note	Metal parts that are nearby in the radial direction, such as a target, can influence and distort the measurement result. Bear this in mind when choosing the material for the sensor mounting.

Shielded sensors

- Type designation: ES-Sx
- Structure: The sensor is surrounded by a metal housing with mounting thread up to the front surface.
  The sensor is thus shielded from the influence of radially nearby metal parts.

4.1.2 Start of measuring range

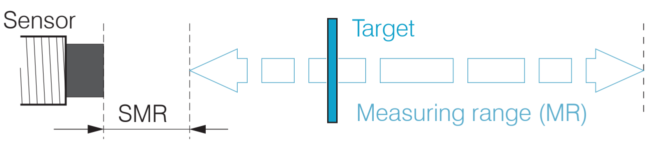


Fig. 4.2: Start of measuring range (SMR), minimum distance between the front surface of the sensor and the target.

Each sensor must have a minimum distance from the measuring object. This prevents unreliable measurements caused by the sensor pressing against the target and mechanically destroying the sensor/target. In the standard factory calibration, the start of the measuring range is 10 % of the nominal measuring range.

4.2 Sensor installation situation

The measurement behavior of eddy current displacement sensors can be affected by a metal holder. Make sure to mount the sensor according to the sensor type used.

4.2.1 Standard mounting

The sensors protrude over the metal holder. The installation scenario depicted is used for factory calibration of the sensors at Micro-Epsilon. The technical sensor data correspond to standard installation conditions. If you want to achieve the values indicated in the data sheet, we recommend to install the sensor in the same way as it was during calibration.

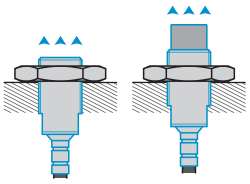


Fig. 4.3: eddyNCDT standard mounting

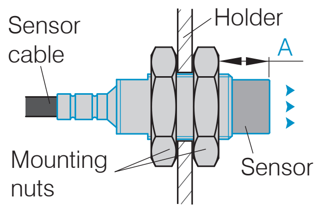
Sensors with threads

► Insert the sensor through the hole in the sensor holder.
► Screw the sensor in place.
► To do so, screw the mounting nuts supplied with the sensor onto the threads protruding from the holder on both sides.
► Tighten the mounting nuts carefully in order to avoid damage, especially to small sensors.

Note	The standard mounting method for the sensor should be used, since you will achieve the best measurement results with this method! Maintain the same position of the sensor relative to the holder during calibration as during measurement. Deviating installation scenarios can be partially compensated by means of field linearization.

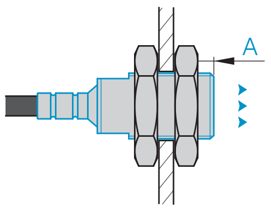
During the factory-calibration of the sensors, the sensor front face is in a defined distance A from the mounting nut. Consider this distance A for the application in order to achieve maximum linearity.

Unshielded sensor with threads, standard mounting

Unshielded sensor with threads, standard mounting	Sensor	Dimension A
	ES-U1	8 mm
	ES-U2	8 mm
	ES-U3	10 mm
	ES-U6	20.4 mm
	ES-U8	24.6 mm

Tab. 4.1:

Shielded sensor with threads, standard mounting

Shielded sensor with threads, standard mounting	Sensor	Dimension A
	ES-S1	4 mm
	ES-S2	4 mm
	ES-S4	4 mm

Tab. 4.2:

Sensors for clamping without thread

► Mount sensors without thread preferably with a circumferential clamping. You can alternatively mount the sensors with a set screw.

Circumferential clamping with clamping collet

This type of sensor mounting ensures the highest level of reliability because the sensor's cylindrical housing is clamped over a relatively large area. It is imperative in complex installation environments such as machines and production plants.

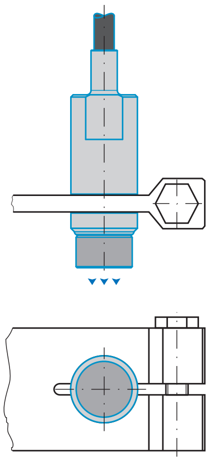


Fig. 4.4: Use a clamping collet

Radial spot clamping with set screw

This simple type of fixture is only recommended for installation locations that are free of impact and vibration. Spot clamping can be done using a set screw or compression screw

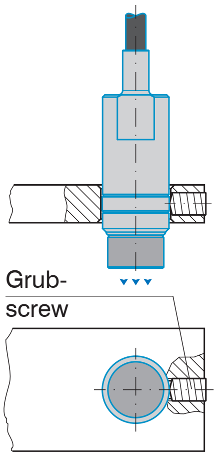
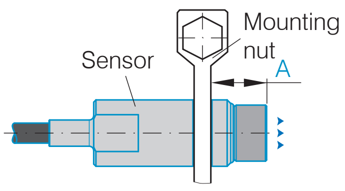


Fig. 4.5: Mounting with set screw

NOTICE
► Make sure not to damage the sensor housing!

Distance between sensor front face and the sensor bracket without thread (standard mounting)

During the factory-calibration of the sensors, the sensor front face is in a defined distance A from the mounting nut. Consider this distance A for the application in order to achieve maximum linearity.



Sensor	Dimension A
ES-U1-T	7 mm
ES-U3-T	10 mm

4.2.2 Flush Mounting Method

Flush mounting does not correspond to factory calibration. Micro-Epsilon recommends to carry out a 3-point field linearization.

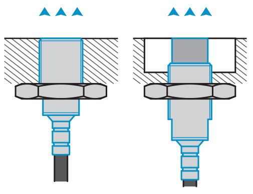


Fig. 4.6: Flush mounting

i Linearize the measuring system, if possible, when it is exactly arranged (in the same way as it will be arranged later during the measurement process).

Sensors with threads

► Mount shielded or unshielded sensors flush in the sensor holder made of insulating material (plastic, ceramic, etc.).

► Mount unshielded sensors flush in the metal sensor holder. Make sure that there is a recess on the holder that is three times the diameter of the sensor.

► Mount shielded sensors flush in the metal sensor holder.

► In all mounting scenarios, screw the sensors into the threaded bore and lock them in place using the mounting nut.

► Tighten them carefully in order to avoid damage, especially to small sensors.

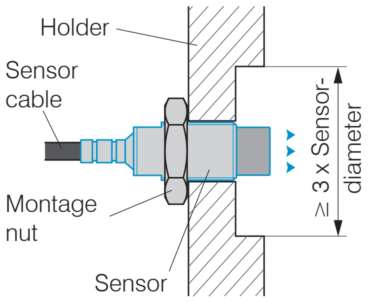


Fig. 4.7: Flush mounting of an unshielded sensor in a metal holder with recess

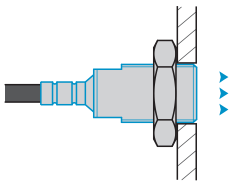


Fig. 4.8: Flush mounting of a shielded sensor in a metal holder

4.3 Measurement setup, operating multiple sensors

Eddy current sensors generate magnetic fields, which can overlap when the sensors are placed too close together (known as cross-talk). There are two solutions to avoid this:

- Mounting with sufficient minimum distance
- Mounting sensors with different frequencies, LF (low frequency) and HF (high frequency)

Fig. 4.9: Sensor ES-Ux, unshielded

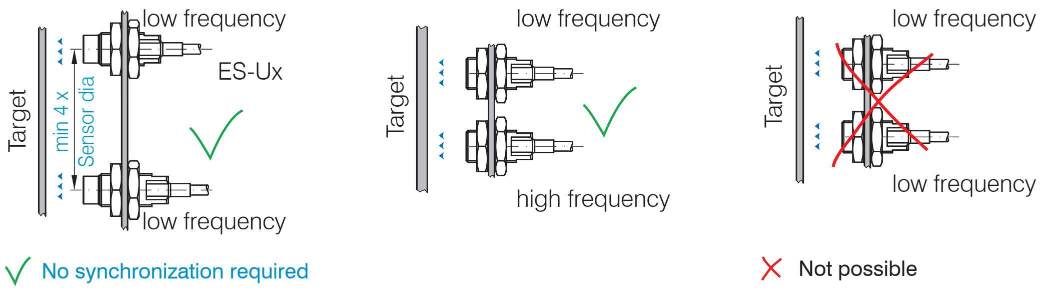
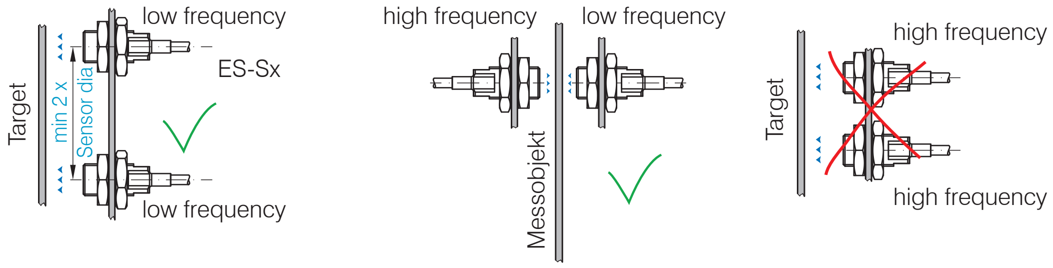


Fig. 4.10: Sensor ES-Sx, shielded



For the simultaneous operation of several measuring systems, these can be supplied with a frequency separation (LF/HF). The frequency separation enables multi-channel operation without mutual influence. This function makes a synchronization superfluous.

If there are more than two sensors, the alternating sequence LF-HF-LF-HF- ... or HF-LF-HF-LF- ... must be observed.

The choice of LF or HF sensors only affects the frequency of the electric field and has no effect on the accuracy, max. frequency response or measuring rate of the controller.

4.4 Dimensional drawings of sensors

Measurement direction

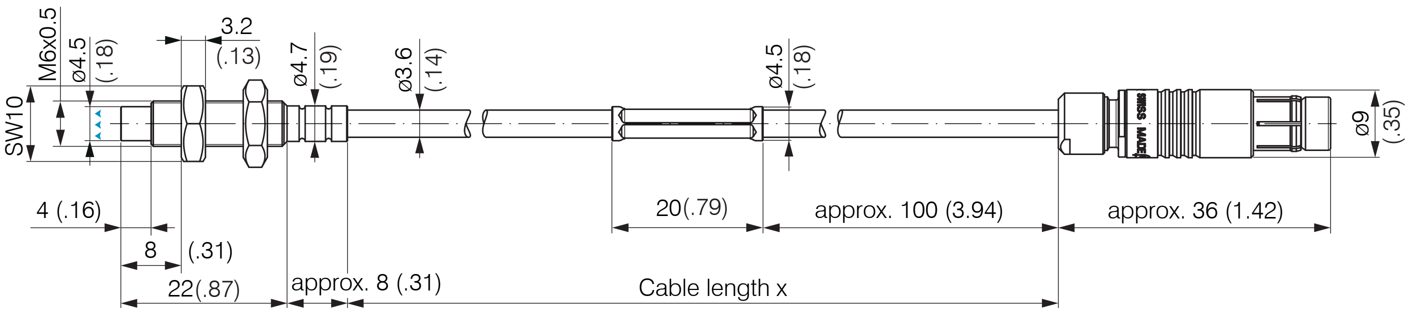


Fig. 4.11: Dimensional drawing for ES-U1-C-CAx/mB0 sensors, dimensions in mm

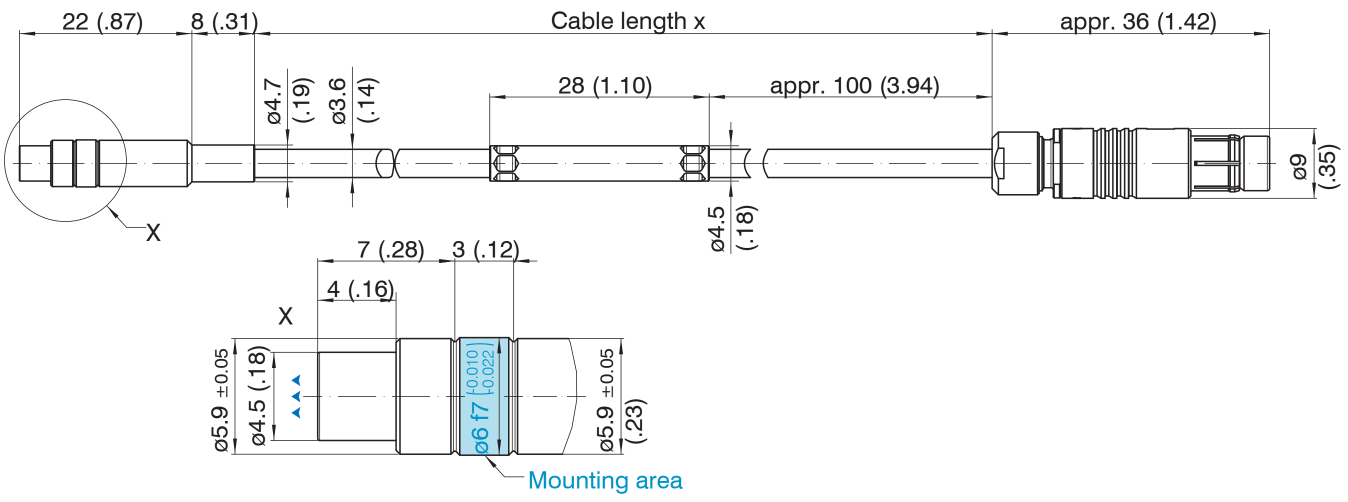


Fig. 4.12: Dimensional drawing for ES-U1-T-CAx/mB0 sensors, dimensions in mm

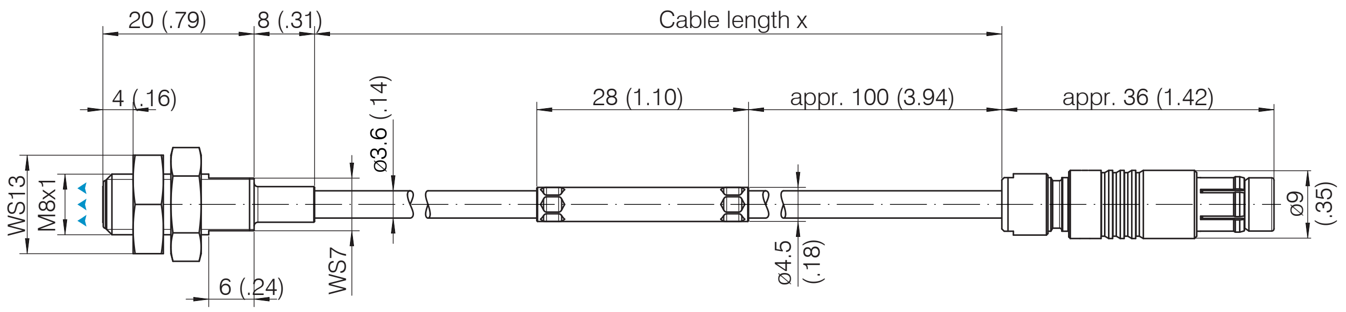


Fig. 4.13: Dimensional drawing for ES-S1-C-CAx/mB0 sensors, dimensions in mm

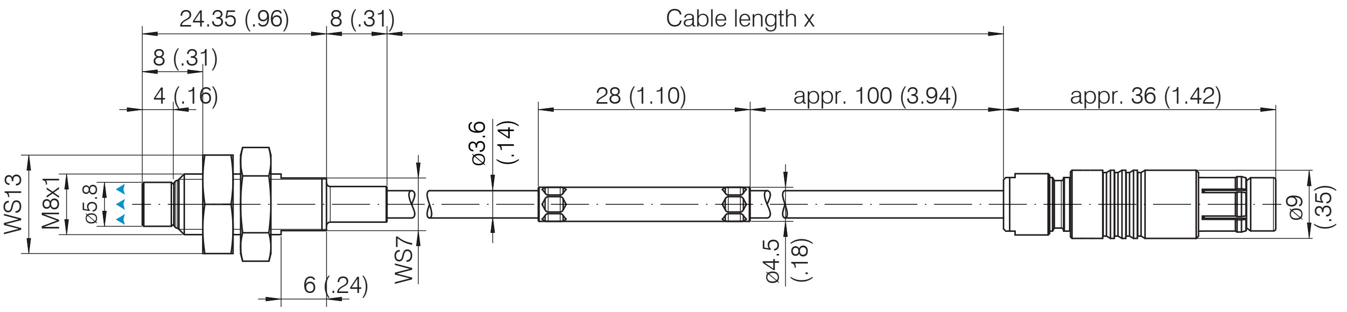


Fig. 4.14: Dimensional drawing for ES-U2-C-CAx/mB0 sensors, dimensions in mm

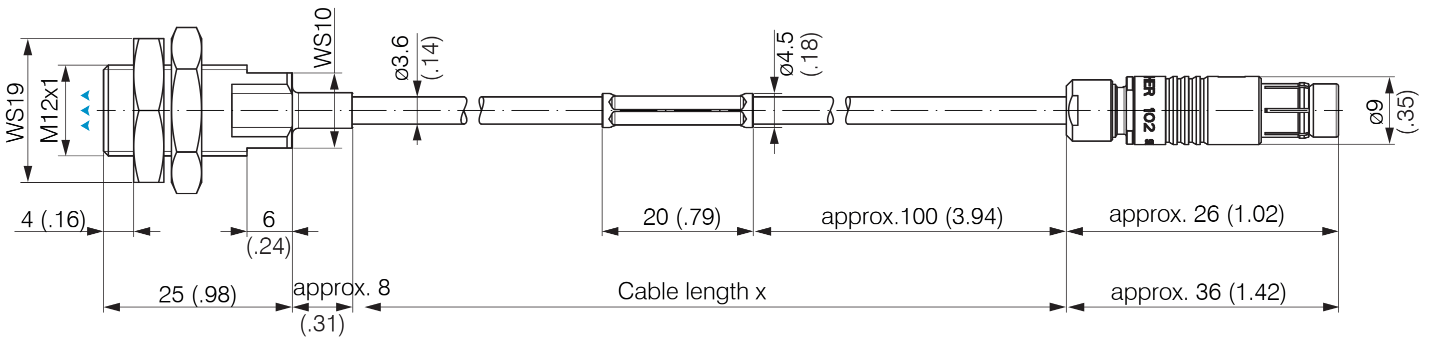


Fig. 4.15: Dimensional drawing for ES-S2-C-CAx/mB0 sensors, dimensions in mm

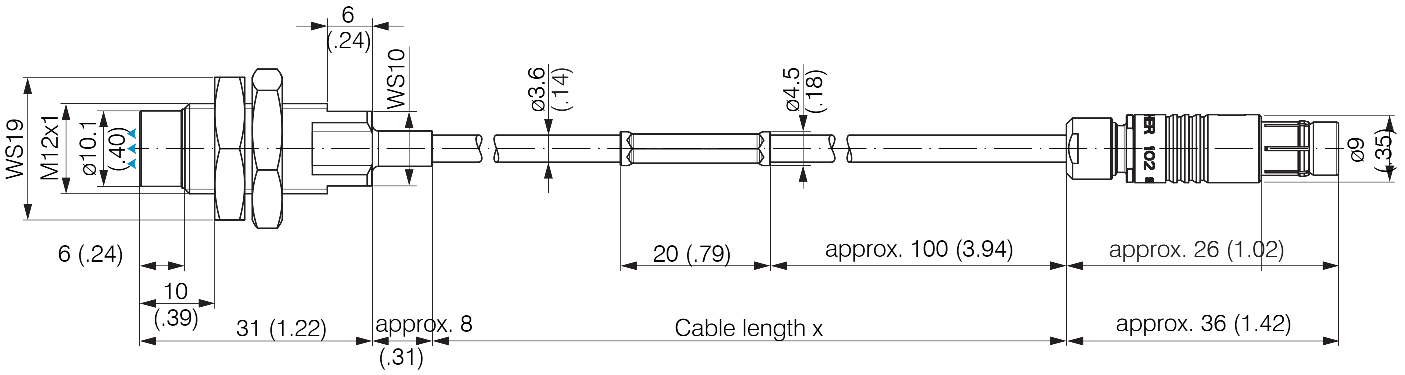


Fig. 4.16: Dimensional drawing for ES-U3-C-CAx/mB0 sensors, dimensions in mm

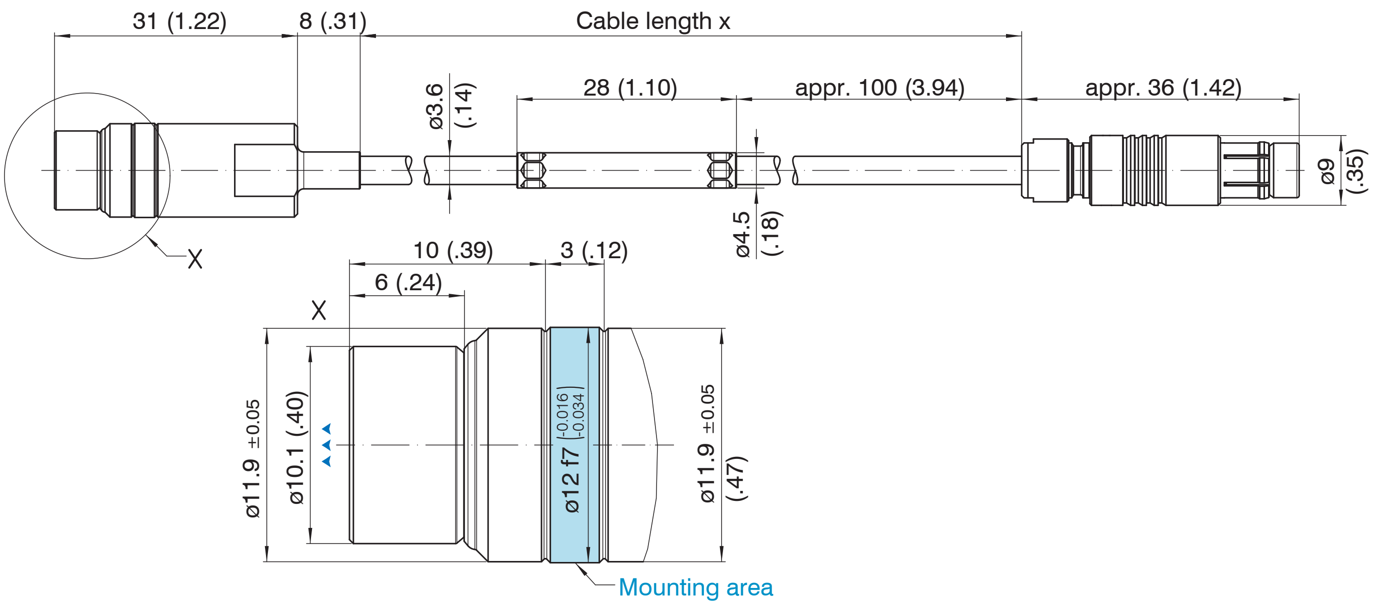


Fig. 4.17: Dimensional drawing for ES-U3-T-CAx/mB0 sensors, dimensions in mm

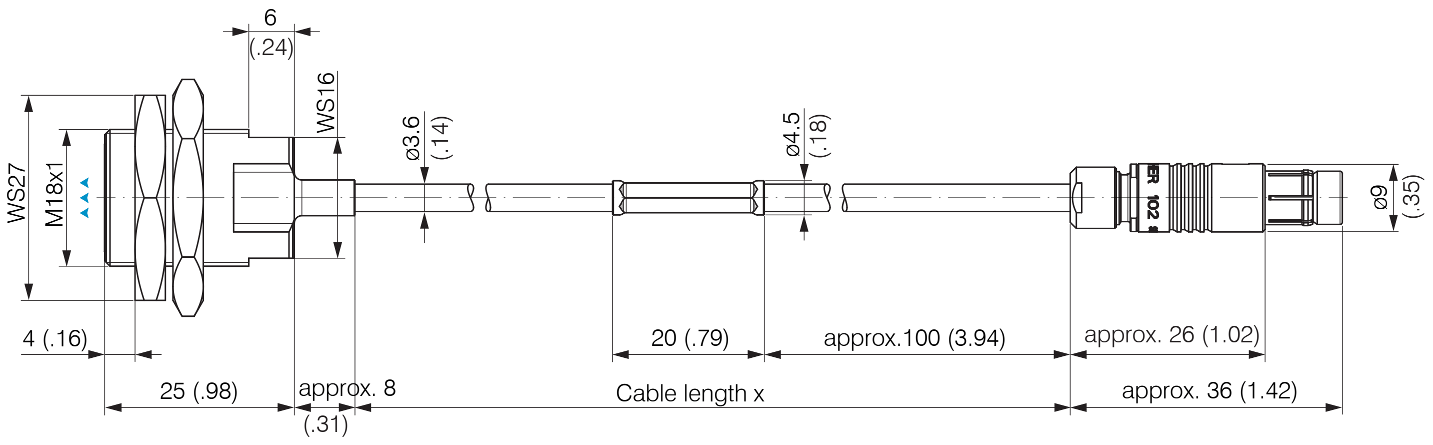


Fig. 4.18: Dimensional drawing for ES-S4-C-CAx/mB0 sensors, dimensions in mm

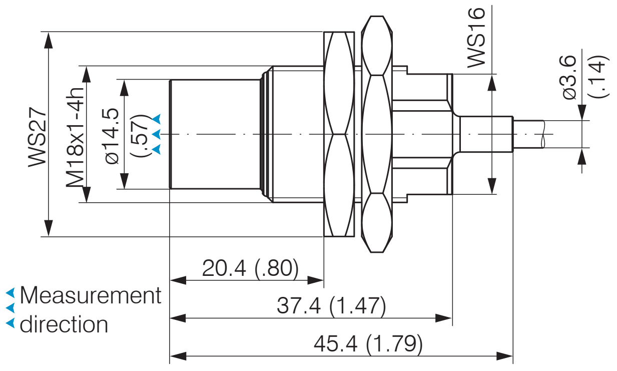


Fig. 4.19: Dimensional drawing for ES-U6-C-CAx/mB0 sensors

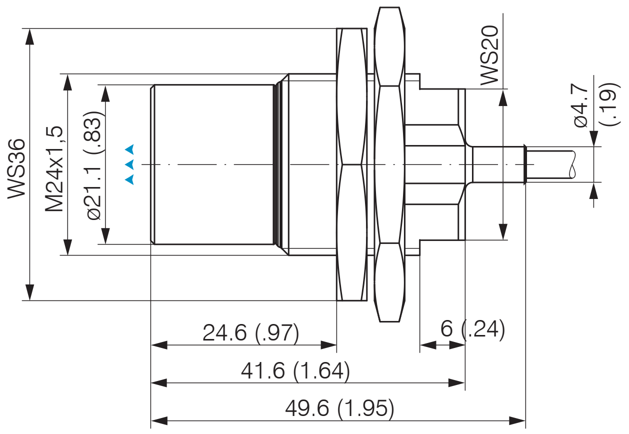


Fig. 4.20: Dimensional drawing for ES-U8-C-CAx/mB0 sensors

4.5 Sensor cable

► Do not kink the cable. Observe the minimum bending radii.

Cable ø 3.6 mm		
ES-U1-C-CAx/mB0 ES-S1-C-CAx/mB0 ES-U1-T-CAx/mB0 ES-U2-C-CAx/mB0 ES-S2-C-CAx/mB0 ES-U3-C-CAx/mB0 ES-U3-T-CAx/mB0 ES-S4-C-CAx/mB0 ES-U6-C-CAx/mB0 ES-U8-C-CAx/mB0	fixed installation, static	27 mm
	Dynamic	54 mm

Tab. 4.3: Minimum bending radii of the sensors/sensor cables

► Lay the sensor cable in such a way that no sharp or heavy objects affect the cable sheath.
► Connect the sensor cable to the controller.

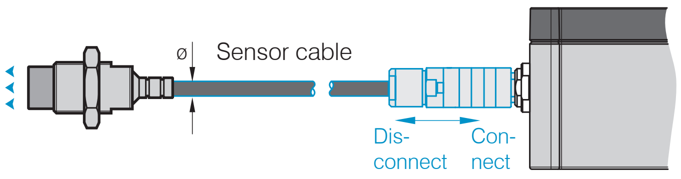


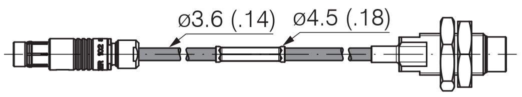
Fig. 4.21: eddyNCDT sensor cable

To release the screw connection, grip the plug connectors by the ribbed grips (outer sleeves) and pull them out straight.

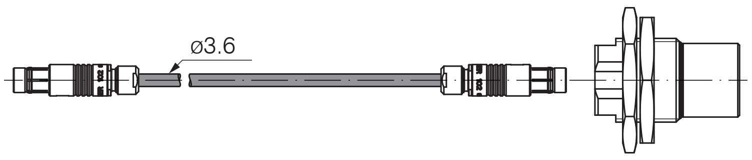
Note	Merely pulling on the cable and clamping nuts will lock the plug connectors and will not release the connection. Therefore, avoid excessive pull on the cables. Do not shorten the sensor cable. A shortened or damaged sensor cable cannot be repaired. Recalibration is required!

► Check the plug connections for a tight fit.

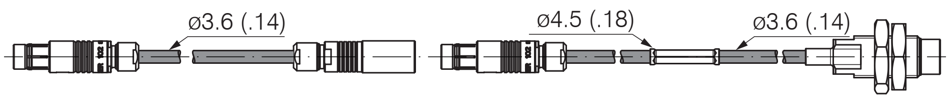
Sensors with integrated cable: cable type ES-xx-x-CAx/mB0



Sensors with socket: cable type EC-x/mB0/mB0



Extension cable: cable type ECE-x/fB0/mB0



Tab. 4.4: Connection cable for DT3020 sensors

Sensors and cables that are to be plugged into the DT3020 controller are designated by “mB0” in their name. See also model names, see Chap. 17.3

4.6 Dimensional drawing of controller

The DT3020 controller is installed in a stable aluminum housing. The controller is coordinated with the accompanying sensor and sensor cable at the factory. The controller has two through-holes measuring 4.4 mm in diameter for secure mounting.

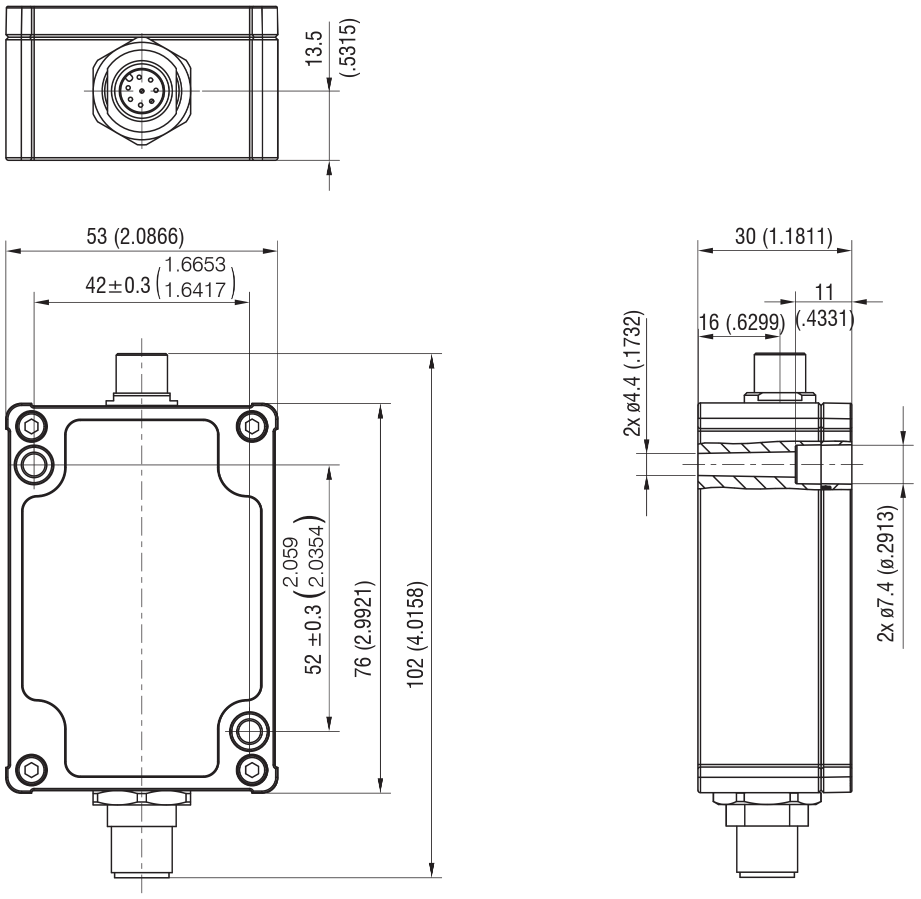


Fig. 4.22: Dimensional drawing for controller DT3020, dimensions in mm

4.7 Minimum target size

The relative size of the target object compared with the sensor affects the linearity and slope deviation for eddy current sensors.

If the required object minimum size cannot be complied with, the following aspects must be taken into account for a sufficiently high linearity:

- The size of the target must not change.
- The target must not be moved laterally in relation to the front sensor face.

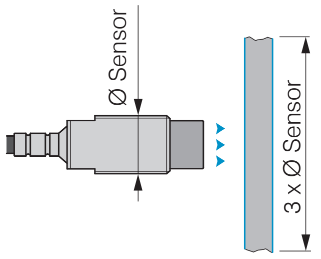


Fig. 4.23: Minimum target size, unshielded sensors

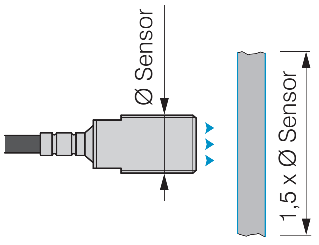


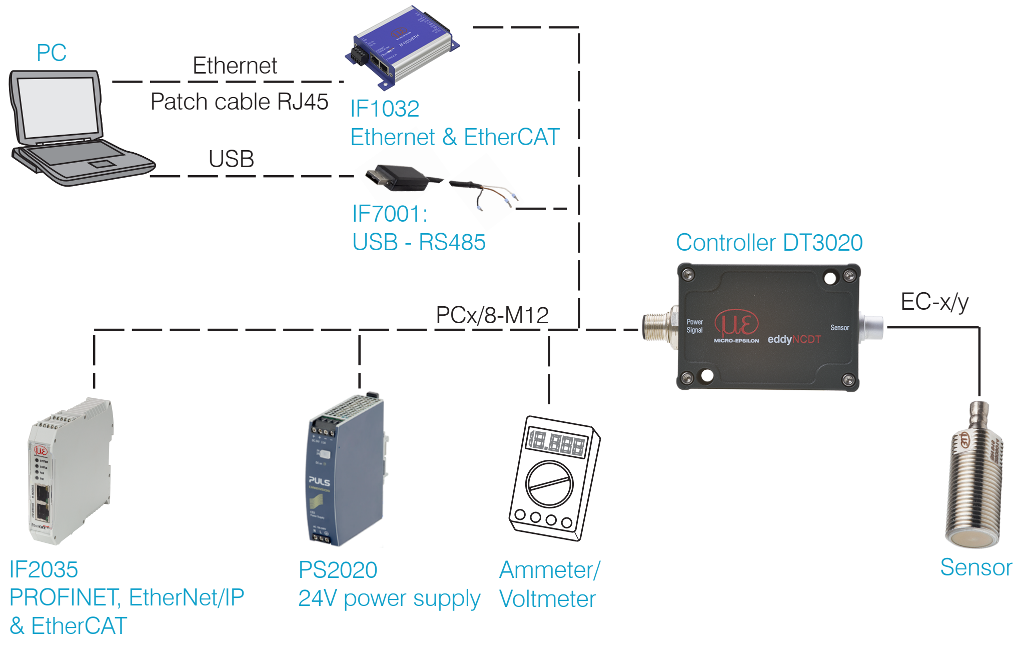
Fig. 4.24: Minimum target size, shielded sensors

Successful calibration is a pre-requisite for minimal linearity errors. In order to achieve an optimal result, Micro-Epsilon recommends a linearity calibration on the corresponding measuring object. A change of the measuring object size has significant effects on the quality of the measurement results.

4.8 Electrical connections

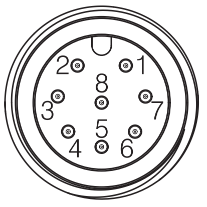
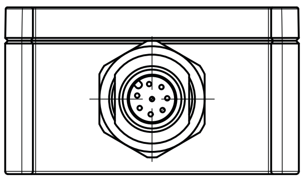
4.8.1 Connection possibilities

Power supply and signal output are on the front of the controller.



Optional accessories can be found here, see Chap. 17.1

4.8.2 Pin assignment

DT3020			
Pin	Description	Wire color PCx/8-M12	
1	NC	White	 *Fig. 4.25: Pin side, 8-pin housing plug*
2	12 ... 30 VDC	Brown	
3	Switching output	Green	
4	RS485 A / +	Yellow	 *Fig. 4.26: Supply and analog output, controller, 8-pin plug, M12 A-coded*
5	RS485 B / -	Gray	
6	GND (displacement)	Pink	
7	GND (supply)	Blue	
8	Analog output	Red	

Tab. 4.5: Pin assignment and color codes

The PCx/8-M12 is a ready-packaged supply and output cable that can be ordered as an optional accessory in the lengths 3 m, 5 m or 10 m. The two ground connections GND (supply) and GND (displacement) are connected to one another internally. The outputs are resistant to short circuits. A 5 m long cable is available in a drag chain-compatible design up to +105 °C, see Chap. 17.1

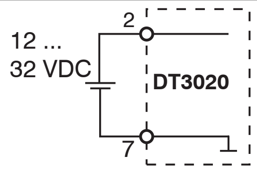
4.8.3 Supply voltage

Nominal value: 24 V DC (12 ... 32 V, power consumption < 1.7 W).

► Only turn on the power supply after wiring has been completed.

► Connect the inputs "2" and "7" on the controller to a 24 V power supply.

Voltage supply only for measuring devices, not to be used for drives or similar sources of impulse interference at the same time. Micro-Epsilon recommends using an optional available power supply unit PS2020 for the controller, see Chap. 17.1

	Controller - pin	PCx/8-M12 - color	Power supply
	2	Brown	V_+
	7	Blue	GND_{Supply}

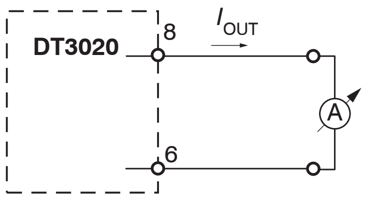
Tab. 4.6: Supply voltage connection

4.8.4 Analog output, displacement

In the delivery state, the controller supplies a current output of 4 ... 20 mA.

The current output can be scaled from 0 ... 20 mA, see Chap. 8.2.1

► Connect the outputs 8 (red) and 6 (pink) on the controller to the measuring device.

	Controller	
	8-pin M12 cable plug	Wire color PCx/8-M12
	I_{Out} (pin 8)	Red
	$GND_{Displacement}$ (pin 6)	Pink

Tab. 4.7: Wiring for current output

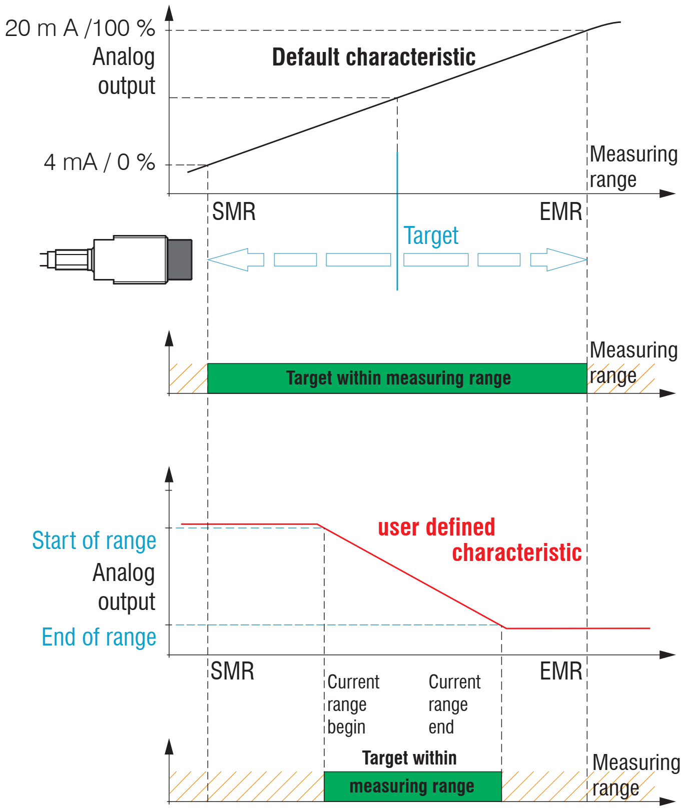


Fig. 4.27: Standard and user-defined characteristic curves

4.8.5 Switching output

The limit value output is deactivated in the delivery state and can be configured, for example, via the `sensorTOOL` software., see Chap. 8.3

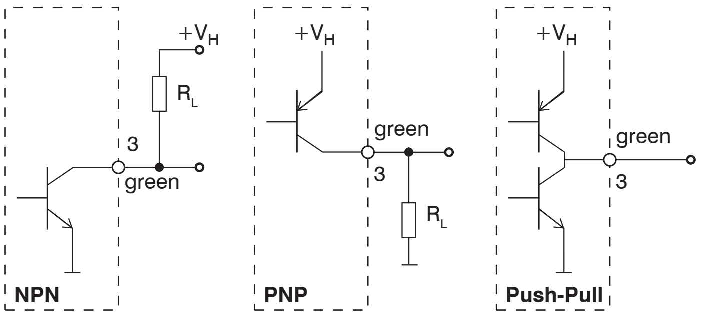


Fig. 4.28: Basic circuit of the switching output

The switching behavior (NPN normally closed, NPN normally open, PNP normally closed, PNP normally open, push-pull, push-pull negated), see Chap. 8.3.4 of both switching outputs depends on the DT3020 setting. The switching output is protected against:

- polarity reversal
- Overload: The status bit 7, see Chap. 10.5 is set to 1. The switch opens for 14.8 ms and closes again for 0.2 ms. Once the overload state has been rectified, the switch works normally again. The switch-on and switch-off cycles are used to determine whether the overload state still applies or has been rectified.
- Excessive temperature: The status bit 7, see Chap. 10.5 is set to 1 and the switch opens. Once the temperature falls below a particular threshold, the switch works normally again.

The voltage levels for the switching states *high* and *low* vary depending on the connected load. The following must apply for the voltage V_H at a pull-up resistor: $V_H \leq 32$ V.

Output current	U_{low}	U_{high}
100 mA[13]	< 1.5 V	> +V_H - 1.5 V
<=50 mA	< 0.5 V	> +V_H - 0.5 V

If the controller is supplied with 24 V and the upper switch is closed. The output voltage of the switching output at 50 mA load current is, in this case, at least 24 V - 0.5 V = 23.5 V and at 100 mA load current is at least 24 V - 1.5 V = 22.5 V.

Note	Shut-off at overload of > 140 mA. The status bit 7 is set to 1, see Chap. 10.5, the output switches to the *Overload* state.

[13] corresponds to saturation at I_{max}

5 Commissioning via sensorTOOL software

5.1 Description, download

sensorTOOL is a piece of software that allows you to set the controller and visualize and document measurements.

You can download the sensorTOOL software via the following link: https://www.micro-epsilon.de/service/software-sensor-integration/sensortool/

► Connect the DT3020 controller via a free USB port on your PC to an RS485 converter and connect the power supply to the DT3020. Alternatively, you can establish a connection with the optional IF1032 interface module via Ethernet., see Chap. 4.8.1

5.2 Testing the measuring system setup

- Is the sensor suitable for the application scenario (target material)?
- Are the sensor, sensor cable length and controller matched to one another (type and serial number)?
- Is the sensor connected? Are the cable connections tight?

5.3 Positioning the target

► Position the target within the sensor measuring range. The value for the start of the measuring range (SMR) depends on the sensor, but by default is 10% of the measuring range. You can find the relevant value in the technical data for the sensor. If the measuring range is restricted by the user, new values are produced for SMR, MMR and EMR, among other things.

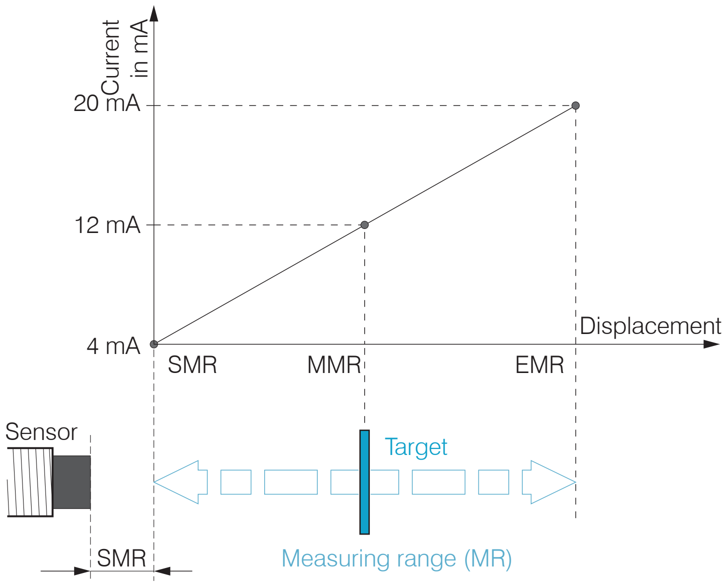
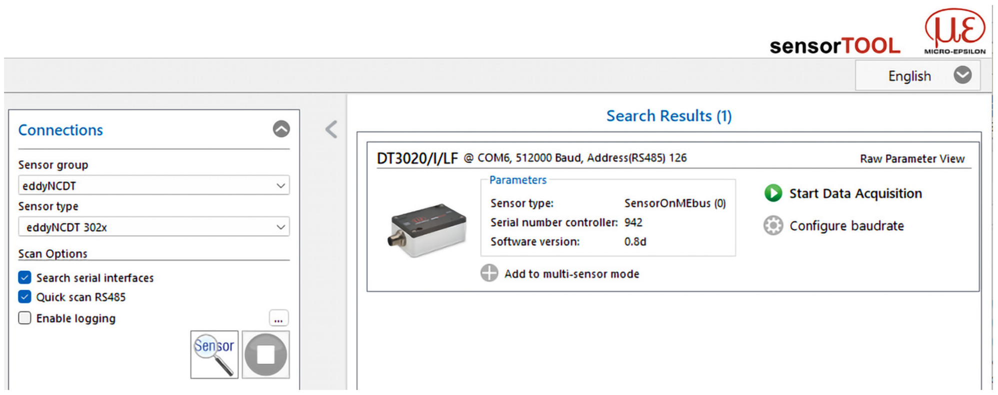


Fig. 5.1: Factory-scaling of analog output

5.4 Single-sensor mode

► Start the `sensorTOOL` program.



DT3020/I/LF @ COM6, 512000 Baud, Address(RS485) 126

Parameters

Sensor type:	SensorOnMEbus (0)
Serial number controller:	942
Software version:	0.8d

Fig. 5.2: First interactive site after calling the sensorTOOL program

► Set the *eddyNCDT* sensor group and the *eddyNCDT 3020* sensor type in the drop-down menus.

► Select the connected sensor.

► Check the box `Search serial interfaces`.

► If only 1 controller is operated on the bus, check the box next to `Quick scan RS485`.

If several controllers are operated on the bus, the check mark for `Quick scan RS485` must be deactivated. In this case, the search takes longer, as the program scans the entire ME bus address range.

► Click on the `Sensor` button with the magnifying glass icon in order to start the search.

► All available channels will now be displayed in the `Search Results (x)` overview.

► Further menus can now be called up using the `Start Data Acquisition` and `Configure baudrate` buttons.

► Now select the desired sensor from the search results.

5.5 Multi-sensor mode

The `sensorTOOL` software also offers the option of outputting the data from multiple DT302x controllers.

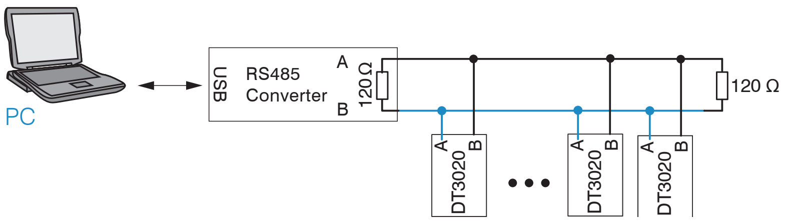


Fig. 5.3: Connecting multiple DT3020 controllers via an RS485 interface.

A 120 Ω terminating resistor is required between the connection lines A and B of the RS485 interface at the start and end of the RS485 bus. No terminating resistor for the RS485 line is installed in the DT3020. It is therefore permissible to connect multiple controllers to a bus cable.

To output the data of multiple bus participants in one graph, please proceed as follows:

Search for the controller in `sensorTOOL`; see *Single Sensor Mode*.

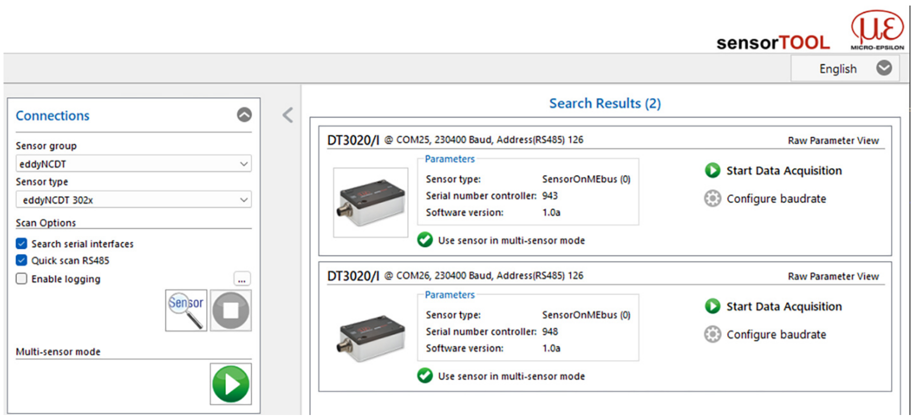
► If you have not already done so, configure each individual channel. Then return to the first interactive page after calling up `sensorTOOL` (search results).

► Then start the `Multi-sensor mode`.

► Then activate the individual check boxes of the relevant channels.

► Start the data acquisition by clicking on the green Play icon on the left-hand side.

Multi-sensor mode also works with multiple USBs on RS485 converters on a PC.



DT3020/I @ COM25, 230400 Baud, Address(RS485) 126

Sensor type:	SensorOnMEbus (0)
Serial number controller:	943
Software version:	1.0a

DT3020/I @ COM26, 230400 Baud, Address(RS485) 126

Sensor type:	SensorOnMEbus (0)
Serial number controller:	948
Software version:	1.0a

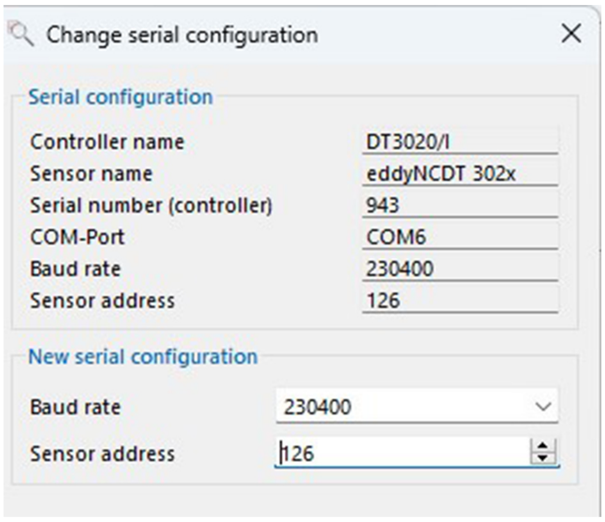
Fig. 5.4: Multi-sensor mode in the sensorTOOL

5.6 Setting baud rate

5.6.1 Configuring the baud rate

► To view the current configuration of the serial interface and change it if necessary, click on the `Configure baudrate` button on the right.

The `Change serial configuration` window then opens, see figure. Here you can change the baud rate and assign a new address for the device.



Controller name	DT3020/I
Sensor name	eddyNCDT 302x
Serial number (controller)	943
COM-Port	COM6
Baud rate	230400
Sensor address	126

Baud rate	230400
Sensor address	126

Fig. 5.5: Configuration of baud rate

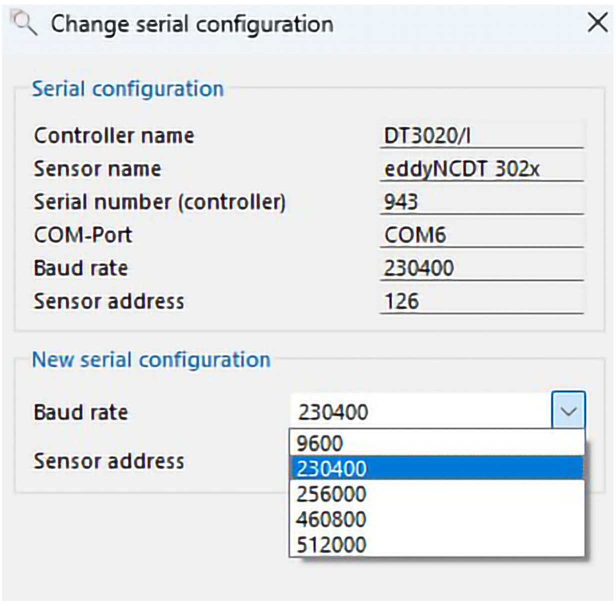
5.6.2 Changing the baud rate

The baud rate can be selected from a drop-down menu in the `Change serial configuration` window.

The DT3020 supports the following baud rates, see figure:

- 9600 bps
- 230400 bps
- 256000 bps
- 460800 bps
- 512000 bps

Byte frame: 1 start bit, 8 data bits, 1 parity bit (parity = even), 1 stop bit


Change serial configuration

Serial configuration

Controller name	DT3020/I
Sensor name	eddyNCDT 302x
Serial number (controller)	943
COM-Port	COM6
Baud rate	230400
Sensor address	126

New serial configuration

Baud rate: 230400

Sensor address

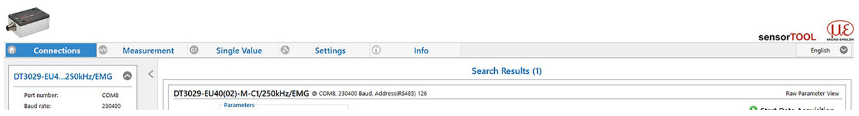
Fig. 5.6: Setting the baud rate

5.7 Menu navigation

A video tutorial is available for the general operation of the Micro-Epsilon software `sensorTOOL`: https://www.micro-epsilon.de/service/software-sensorintegration/sensortool/

► Select the menu item `Start Data Acquisition`.

This will take you to the main menu for the program with the selection bar `Connections, Data acquisition, Single value, Settings` and `Info`.



DT3029-EU4...250kHz/EMG

Port number: COM8

Baud rate: 230400

DT3029-EU40(02)-M-C1/250kHz/EMG COM8, 230400 Baud, Address(RS485) 126

Fig. 5.7: eddyNCDT 3020 menu sensorTOOL 120x13 TIF

`Connections`

Takes you back to the `sensorTOOL` start page with the selection `Start Data Acquisition` and `Configure baud rate`.

`Data acquisition`

Start and stop of the data acquisition, visualization of the measurement as a signal curve, shows controller and sensor temperature, allows settings to be made for signal processing, CSV output, InfluxDB output, TCP/UDP output and reference time setup, see Chap. 6

`Value (single)`

Start and stop of the data acquisition, shows digital measurement value, controller and sensor temperature, allows settings to be made for signal processing, CSV output, InfluxDB output, TCP/UDP output and reference time setup, see Chap. 7

`Settings`

- `Field linearization` with scalable measuring range and analog and distance values and setting of field linearization, see Chap. 8.2
- `Switching output` with detector, limit value type and time condition settings, see Chap. 8.3
- `Filters and measuring rates` with settings for filters and averaging, data transmission RS485 and data transmission diagnostics, see Chap. 8.5.1
- `Status bits` with temperature warning thresholds, see Chap. 8.4

`Info`

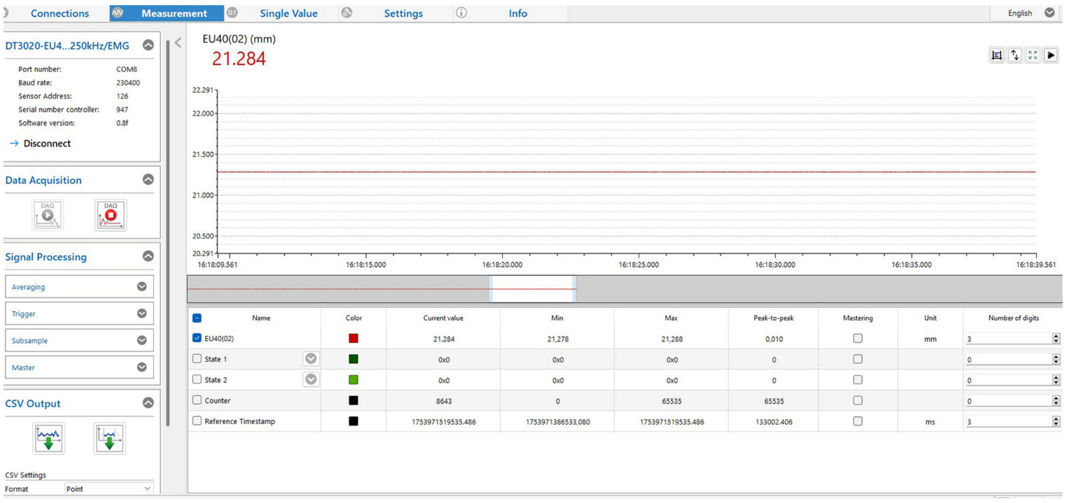
Controller, sensor, diagnostics and characteristic curve values are displayed under the menu item `Info`

6 Data acquisition in the sensorTOOL

Note	Settings that are made in the sections below only affect processing via the sensorTOOL software. Use the menu item `Settings` in the main program menu to render settings effective directly on the controller and so that the controller provides the measurement values taking into account previous settings. For more information, see Chap. 8

6.1 Data acquisition

Click on the `Start Data Acquisition` button or on the controller symbol to make further settings and start data acquisition.



Port number: COM8
Baud rate: 230400
Sensor Address: 126
Serial number controller: 947
Software version: 0.8f

Format: Point

EU40(02) (mm)
21.284

Name	Color	Current value	Min	Max	Peak-to-peak	Mastering	Unit	Number of digits
☑ EU40(02)	🟥	21.284	21,278	21,288	0,010	☐	mm	3
☐ State 1	🟩	0x0	0x0	0x0	0	☐		0
☐ State 2	🟩	0x0	0x0	0x0	0	☐		0
☐ Counter	⬛	8643	0	65535	65535	☐		0
☐ Reference Timestamp	⬛	1753971519535.486	1753971386533,080	1753971519535.486	133002.406	☐	ms	3

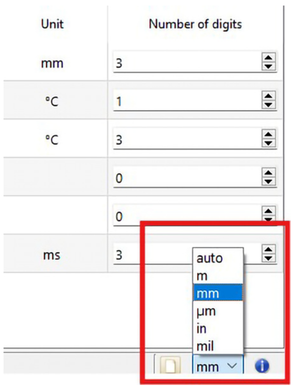
► Before the first data acquisition, select the desired parameters via the menu item `Settings`.

The selection menu in the top right corner makes the following settings possible:

Symbol	Description
⊞	Activates a second Y-axis as soon as distance channels and other signals are displayed. Non-distance channels are shifted to a different coordinate system in order to ensure optimal automatic scaling.
⇅	Inverts the scaling of the Y-axis.
⤢	Resets the Y-scale to the original setting (e.g. after zoom).
▶	The current signal curve is displayed.

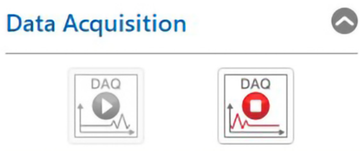
Changing units

The selection menu in the bottom right corner changes the unit for the distance values.



Unit	Number of digits
mm	3
°C	1
°C	3
	0
	0
ms	3

Starting and stopping data acquisition



	Recording is restarted when you press this button. The previously paused recording is lost.
	Recording is stopped when you click this button.

Tab. 6.1:

Disconnecting

When you click on the `Disconnect` button, the menu jumps back to the `sensorTOOL` start page.

6.2 Signal processing

Setting for signal processing in `sensorTOOL`. These settings do not influence the settings in the DT3020 controller and do not change how the measurement values are processed in the controller.

You can select the following options for signal processing:

- `Averaging`
- `Trigger`
- `Subsampling: Sample-based`
- `Master`

6.2.1 Moving mean

Moving mean

The arithmetic average M_{mov} is calculated and output using the selectable filter width N of consecutive measurement values. Each new measured value is added, and the first (oldest) value is removed from the averaging (from the window).

$M_{mov} = \frac{\sum_{k=1}^{N} MV(k)}{N}$	MV = measurement value
	N = averaging number
	k = continuous index (in the window)
	M_{mov} = mean value or output value

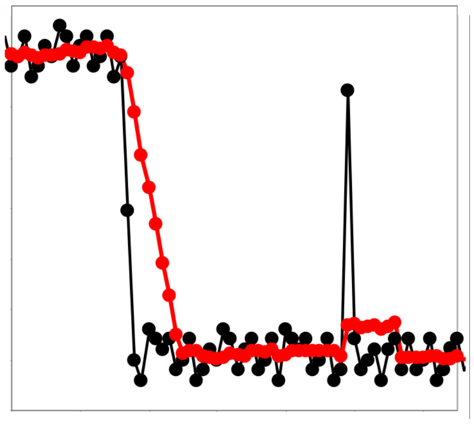
This produces short settling times in case of measurement jumps.

Example: N=4

$$\ldots\ 0,\ 1,\ \underbrace{2,\ 2,\ 1,\ 3}_{\downarrow} \qquad \ldots\ 1,\ 2,\ \underbrace{2,\ 1,\ 3,\ 4}_{\downarrow} \qquad \text{Measured values}$$

$$\frac{2,\ 2,\ 1,\ 3}{4} = M_{\text{mov}}(n) \qquad \frac{2,\ 1,\ 3,\ 4}{4} = M_{\text{mov}}(n+1) \qquad \text{Output value}$$

Fig. 6.1: Moving mean formula 2 examples 116x16



Application tips

- Smoothing of measured values
- The effect can be finely measured in comparison to the recursive averaging.
- With uniform noise of the measured values without spikes.
- Also suitable for measured value jumps at relatively short settling times.

— Signal without averaging

— Signal with averaging

Tab. 6.2:

6.2.2 Moving median

A median value is formed from a preselected number of measured values.

When a median value is formed, the incoming measurement values are resorted after each measurement.

The median splits a list of values into two halves. It can be determined as follows:

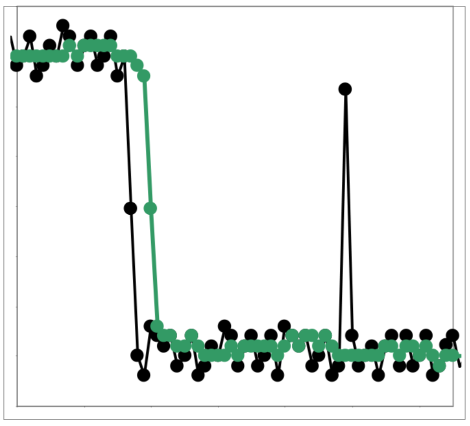
- All values are sorted (in ascending order).
- If the number of values is odd, the middle number is the median.
- If the number of values is even, the median is calculated as an arithmetic mean of the two numbers in the middle.

This means that individual interference pulses can be suppressed. However, smoothing of the measurement curves is not very strong.

Example: median from measurement values

$\ldots\ 0\ \ 1\ \ \underline{2\ \ 4\ \ 5\ \ 1\ \ 3} \rightarrow$ Sorted measurements: $1\ \ 2\ \ \boxed{3}\ \ 4\ \ 5$ Median$_{(n)} = 3$

$\ldots\ 1\ \ 2\ \ \underline{4\ \ 5\ \ 1\ \ 3\ \ 5} \rightarrow$ Sorted measurements: $1\ \ 3\ \ \boxed{4}\ \ 5\ \ 5$ Median$_{(n+1)} = 4$



Application tips

- The measured value curve is not smoothed to a great extent; it primarily eliminates spikes
- Suppresses individual interference pulses
- In short, strong signal peaks (spikes)
- Also suitable for edge jumps (only minor influence)
- Further averaging can be used after the median filter

— Signal without averaging

— Signal with averaging

Tab. 6.3: Median, N = 7

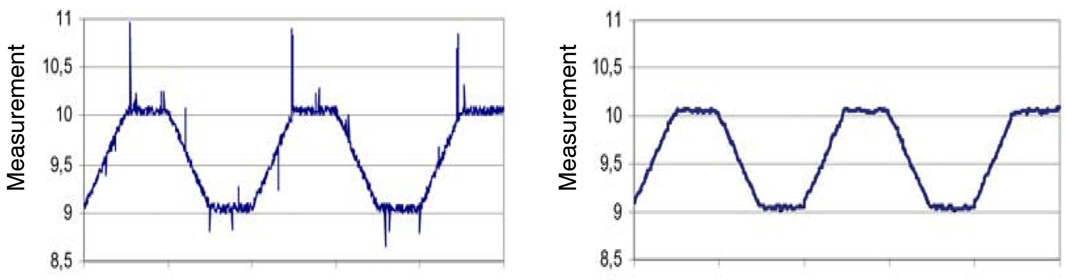


Fig. 6.2: Signal profile without median (left), with median N = 9 (right)

6.2.3 Recursive average

Each new measurement value *MW*(n) is weighted and added to (n-1) times the previous average value.

Formula:

$$M_{rec}(n) = \frac{MW_{(n)} + (N\text{-}1) \times M_{rec\,(n-1)}}{N}$$

N = averaging number

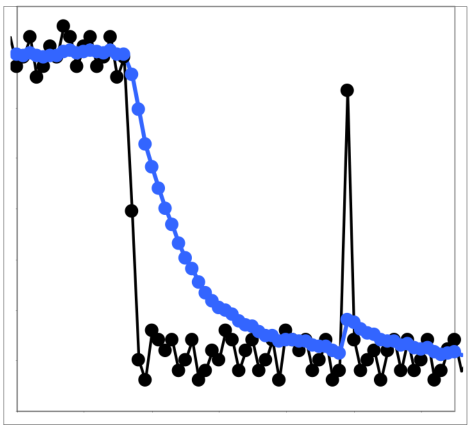
n = measured value index

MW = measurement value

M_{rec} = mean value or output value

Tab. 6.4:

Recursive averaging allows for very strong smoothing of the measurements, however it requires long response times for measurement jumps. The recursive average value shows low-pass behavior.



Application tips

- Permits a high degree of smoothing of the measured values. Long settling times for measured value jumps (low-pass behavior).
- Strong smoothing of noise without large spikes.
- For static measurements, to smooth the signal noise particularly strongly.
- To eliminate structures, e.g., parts with uniform groove structures, knurled turned parts or coarsely milled parts.
- Unsuitable for highly dynamic measurements.

—— Signal without averaging

—— Signal with averaging

Tab. 6.5:

6.2.4 Trigger, subsampling, master

Require a selection		Require the specification of a value
Trigger	Disabled	Deactivated; basic settings
	Continuous	Manual trigger: a measurement value is recorded when the `Trigger` button is pressed.
	One-shot (sample-based)	Sample settable; records the set number of samples in each case when the `Trigger` button is pressed; the more samples there are, the longer the curve
	One-shot (time-based)	Milliseconds settable; records measurement values for the set time period when the `Trigger` button is pressed.

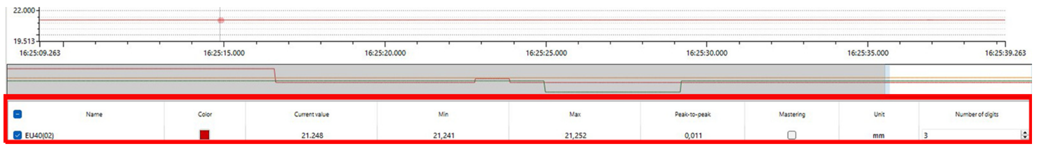
Subsampling	Deactivated	Deactivated; basic settings: Every measurement value is recorded.
	Sample-based	Number of samples is adjustable; every xth measurement is recorded.
	Time-based	Time-based: A measurement value is recorded within the set time interval.
Master	Master now	Sets the master, see figure.
	Resetting	Resets the master.

Tab. 6.6: Selection options for signal processing

Master: Supplementary function

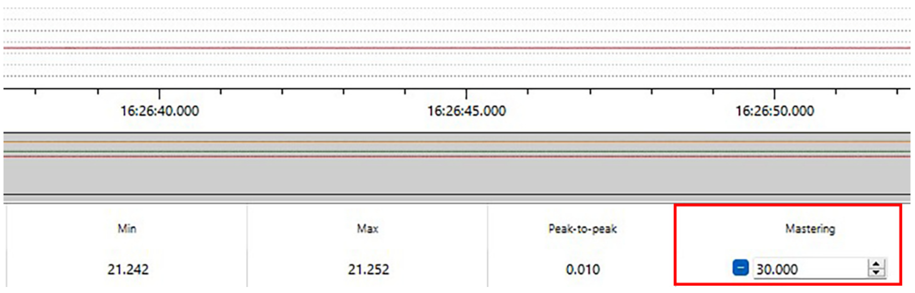
The respective current measured value is set to a default value/master value via the mastering function. All changes to the value are displayed relative to this default value.

In the `sensorTOOL` software, a list showing the connected controller(s) is displayed under the measurement signal.



	Name	Color	Current value	Min	Max	Peak-to-peak	Mastering	Unit	Number of digits
✓	EU40(02)	🟥	21.248	21,241	21,252	0,011	☐	mm	3

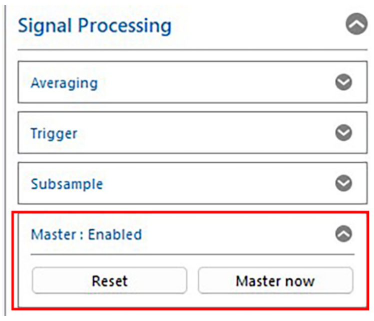
Here, the field must first be activated under `Mastering` and the desired master value entered.



Min	Max	Peak-to-peak	Mastering
21.242	21.252	0.010	30.000

Then you can click on the `Master now` button on the left-hand side in the `Signal processing` menu bar.

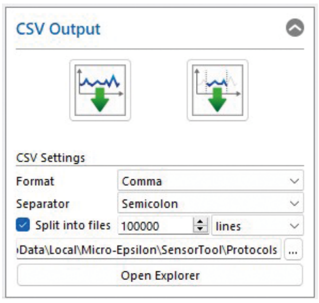
The action was successful if `Master: Enabled` is displayed.



The stored master value can be removed with the `Reset` button.

6.3 Recording and saving a measurement

During data acquisition, the measurement data is only displayed and not automatically saved on the PC. In the side menu under `CSV Output`, you can start transmitting data into a `*.CSV` file or only save the currently visible area from the time graph. In both cases, all data points of the selected interval contained in `sensorTOOL` are written to the specified file.



Format	Comma
Separator	Semicolon
✓ Split into files	100000 lines
\Data\Local\Micro-Epsilon\SensorTool\Protocols	...

Fig. 6.3: View sensorTOOL CSV Output

	Data acquisition into a *CSV file is started when you press this button. To stop and save the recording, click on the icon again.
	You can save the currently visible range in the graph by clicking on this button.

Tab. 6.7: Record and save measurement

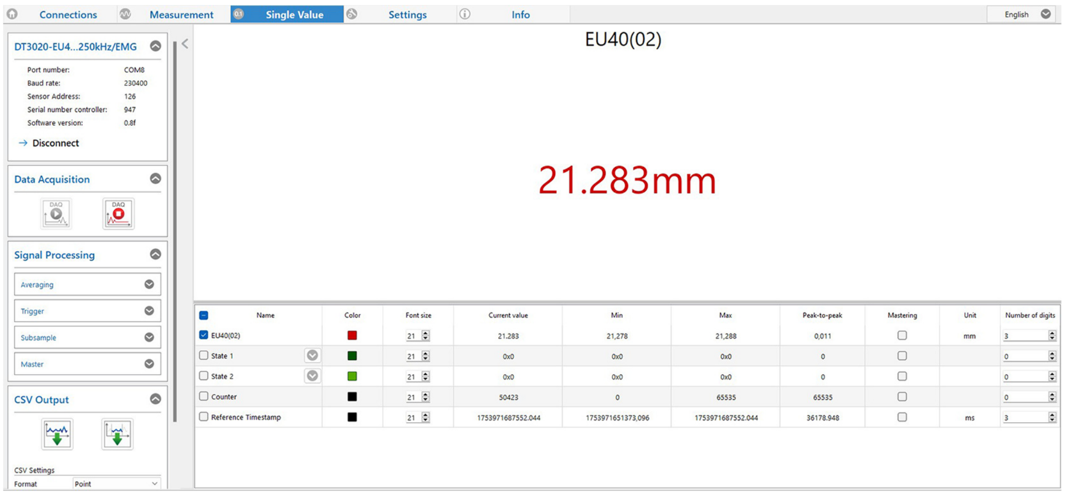
You can make further settings under `CSV settings`.

CSV output	CSV settings	*Format*	*Point / comma*	
		Separator	*Comma / semicolon / tab*	
		Split data	*value*	*lines / MB / min / hourly / time point / DAQ-Start*

With `Open Explorer`, the previously selected path opens in Explorer, where you can view the recorded measurement results.

7 Single value in sensorTOOL

The respective values are output individually here.



Port number:	COM8
Baud rate:	230400
Sensor Address:	126
Serial number controller:	947
Software version:	0.8f

Format: Point

EU40(02)

21.283mm

Name	Color	Font size	Current value	Min	Max	Peak-to-peak	Mastering	Unit	Number of digits
☑ EU40(02)	🟥	21	21.283	21,278	21,288	0,011	☐	mm	3
☐ State 1	🟩	21	0x0	0x0	0x0	0	☐		0
☐ State 2	🟩	21	0x0	0x0	0x0	0	☐		0
☐ Counter	⬛	21	50423	0	65535	65535	☐		0
☐ Reference Timestamp	⬛	21	1753971687552.044	1753971651373,096	1753971687552.044	36178.948	☐	ms	3

Fig. 7.1: Single value sensorTOOL

You have the following setting options:

Color	Setting for the color in which the relevant value is to be displayed.
Font size	Setting for the font size for the relevant value.
Decimal places	Number of decimal places for the measurement value that are displayed and saved in the CSV file
Mastering	Default value for the mastering function under signal processing.

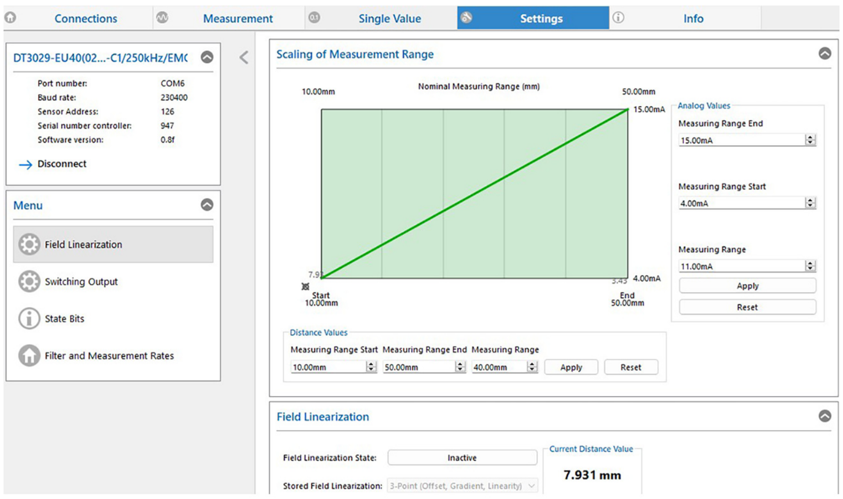
`Signal processing`, `Recording and saving measurement`, see Chap. 6.2 and, see Chap. 6.3

The individual states can be displayed by clicking on the arrow symbol next to the *State 1* signal., see Chap. 10.5

8 Settings on the controller

Note — The settings in this area are saved directly on the controller, which provides measurement values taking into account the settings made here. These settings thus not only have an effect on the data processing by the software, but also on the processing via interfaces and the analog output.
In contrast, settings under `Data acquisition` and `Single value` only affect the processing by the software. They are not saved on the controller.

8.1 General information on the controller settings



DT3029-EU40(02...-C1/250kHz/EM(

Port number:	COM6
Baud rate:	230400
Sensor Address:	126
Serial number controller:	947
Software version:	0.8f

Scaling of Measurement Range

Nominal Measuring Range (mm): 10.00mm – 50.00mm; 15.00mA; 4.00mA; 7.9; 3.43; Start 10.00mm; End 50.00mm

Analog Values

Measuring Range End	15.00mA
Measuring Range Start	4.00mA
Measuring Range	11.00mA

Distance Values

Measuring Range Start	Measuring Range End	Measuring Range
10.00mm	50.00mm	40.00mm

Field Linearization

Field Linearization State:	Inactive
Stored Field Linearization:	3-Point (Offset, Gradient, Linearity)

Current Distance Value: 7.931 mm

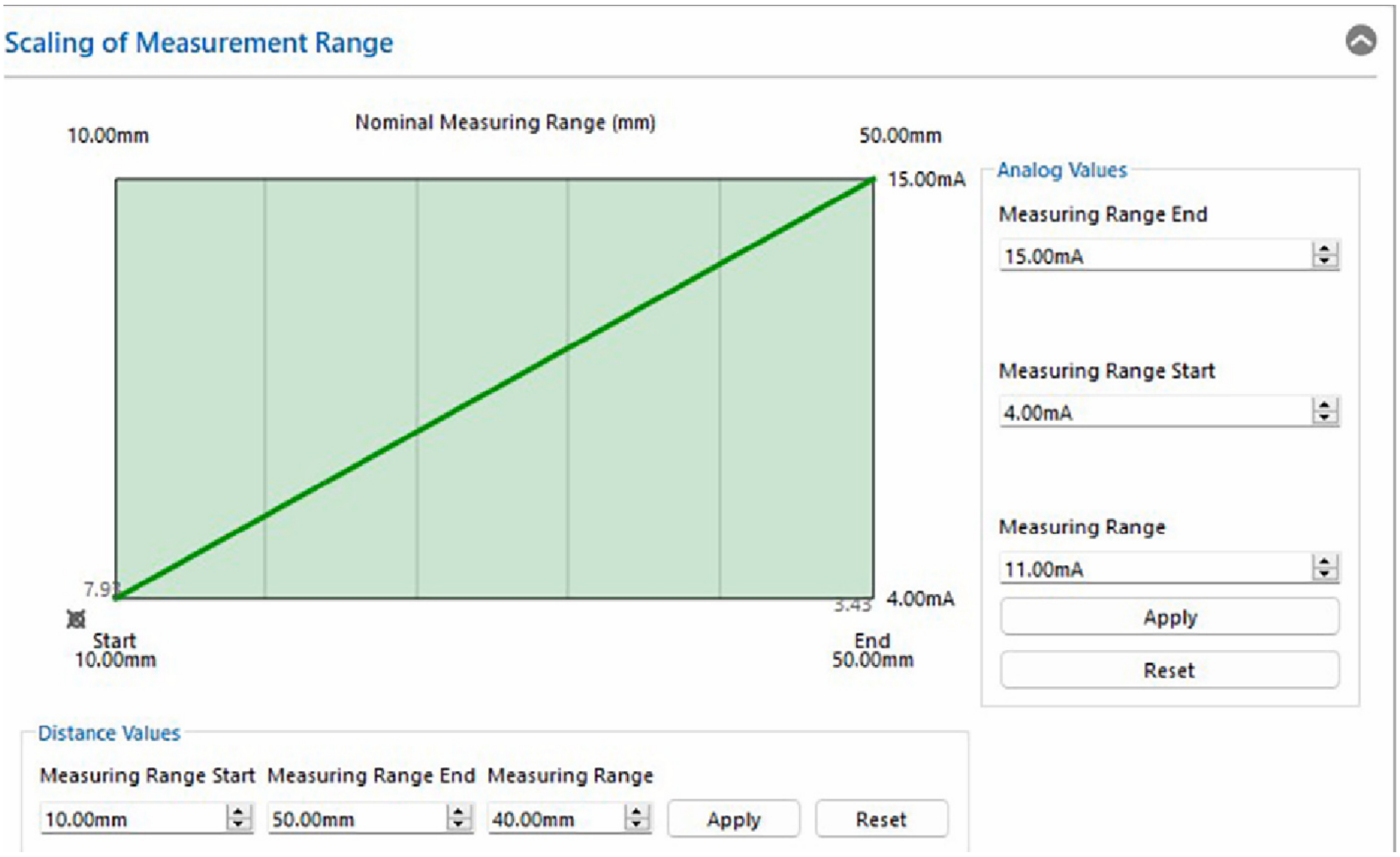
Fig. 8.1: sensorTOOL settings

Here you can make settings for `Field linearization`, `Switching output`, `Status bits` and `Filters and measuring rates`.

8.2 Field linearization

8.2.1 Scalable measuring range

You can set the measuring range of the eddyNCDT 3020 with the fields `Start of measuring range`, `End of measuring range` and `Measuring range` and/or via the analog current output as well as the manual scaling of the current range between 0 and 20 mA.



Analog Values:

Measuring Range End: 15.00mA

Measuring Range Start: 4.00mA

Measuring Range: 11.00mA

Distance Values:

Measuring Range Start	Measuring Range End	Measuring Range
10.00mm	50.00mm	40.00mm

Fig. 8.2: sensorTOOL scalable measuring range

This allows you to optimally set the sensor to the values to be measured. An adjusted measuring range produces a higher resolution. The following parameters are used here:

- **Start of measuring range (SMR):** Lower limit of the measurement value range to be displayed
- **End of measuring range (EMR):** Upper limit of the measurement value range to be displayed
- **Measuring range (MR):** Actual measuring range between the start of measuring range and end of measuring range.

If the SMR is changed, the EMR remains the same and the MR is shifted accordingly. If the EMR is changed, the SMR remains the same and the MR is shifted accordingly. If the MR is changed, the SMR remains fixed but the EMR is recalculated. Measuring range = Difference between start of measuring range and end of measuring range

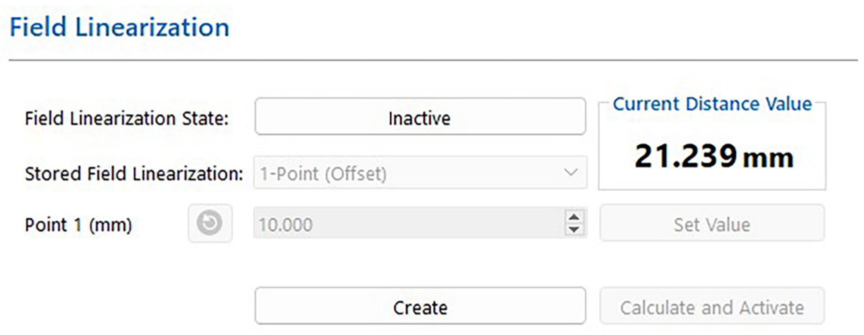
Once you have changed the values as needed, confirm them with `Apply`. If you want to undo the change, click on `Reset`.

Note	If the scaling of the range is smaller than the range actually used, this can result in measurement values outside of the defined range not being displayed or being misinterpreted.

8.2.2 Field linearization

Every eddyNCDT sensor is calibrated at the factory under standardized installation conditions (target material, measuring range). The installation conditions also include the fastening method, position of the nuts and surrounding materials. Deviations from the standard installation situation can compromise the linearity. Field linearization or special tuning in the factory may counteract this effect.

Note	Allow the measuring system to warm up for about 30 to 60 minutes before linearization.



Field Linearization

Field Linearization State: Inactive

Current Distance Value: 21.239 mm

Stored Field Linearization: 1-Point (Offset)

Point 1 (mm): 10.000

Fig. 8.3: sensorTOOL field linearization

Note — If `Inactive` is selected under `Field linearization status`, the sensor measures according to the factory-set values.
If you select `Active`, the sensor measures based on the last correctly performed field linearization.

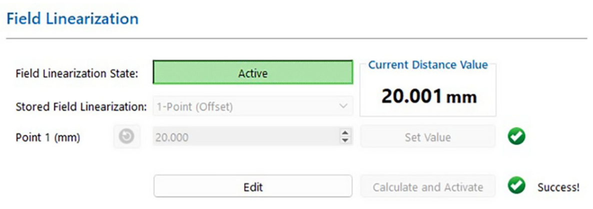
Function	Correction	Field of application
1-point	Zero offset	Fast zero point correction
2-point	Zero point shift + slope	Linear deviations
3-point	Zero point shift + slope + linearity	Non-linear sensor behavior

Tab. 8.1: Linearization functions

Set the respective distances between the sensor and target. Enter the physically produced distance value in each case under `Point 1/2/3`. Place

Field linearization	Point 1	Point 2	Point 3
1-point field linearization	close to the start of the measuring range	-	-
2-point field linearization	close to the start of the measuring range	close to the end of the measuring range	-
3-point field linearization	close to the start of the measuring range	close to the mid of the measuring range	close to the end of the measuring range

8.2.2.1 1-point (zero offset)



Field Linearization

Field Linearization State: Active

Current Distance Value: 20.001 mm

Stored Field Linearization: 1-Point (Offset)

Point 1 (mm): 20.000

Success!

Fig. 8.4: sensorTOOL 1-point (zero offset)

The system is factory-linearized, the mechanical zero point in the installed state should be redefined.

A measured sensor value is compared with a known reference distance (this can be freely defined within the sensor measuring range). The difference is saved as a fixed correction value (zero offset) and applied to all measurement values.

Example: The sensor shows a distance of 2.10 mm instead of 2.00 mm → an offset of 0.10 mm is subtracted.

This function is applied if the sensor characteristic is already linear and a shift only occurs due to mounting effects, for example.

Proceed as follows:

► Select `Edit`. The fields can now be edited.

► Select `1-point`.

► Enter the correct reference value at `Point 1 (mm)`.

- Click on `Set value` and then on `Calculate and activate`.

The green checkmarks indicate whether the action was successful.

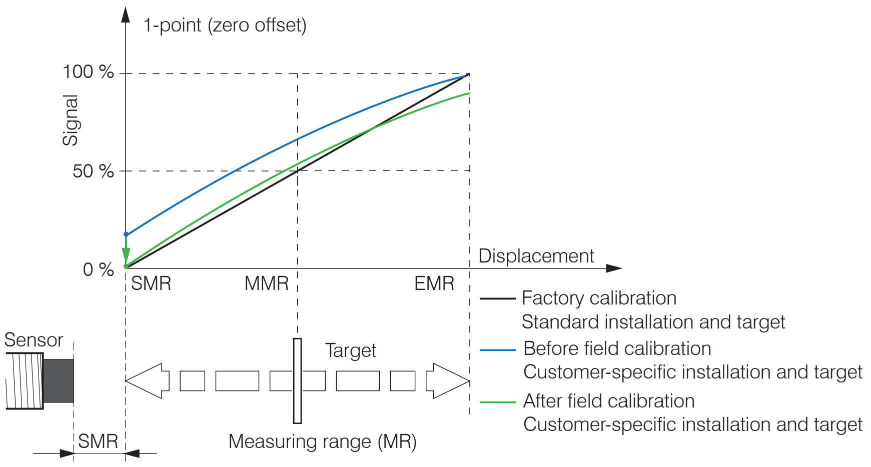
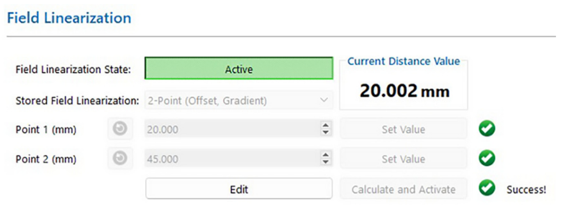


Fig. 8.5: 1-point (zero offset) sketch

> Note
>
> This symbol restores only the preset default values in the display. The values stored on the sensor for a previous field linearization are not overwritten. To reset these values, you must carry out a new field linearization.

8.2.2.2 2-point (zero offset, sensitivity)



Field Linearization

Field Linearization State:	Active	Current Distance Value	
Stored Field Linearization:	2-Point (Offset, Gradient)	20.002 mm	
Point 1 (mm)	20.000	Set Value	✓
Point 2 (mm)	45.000	Set Value	✓
	Edit	Calculate and Activate	✓ Success!

Fig. 8.6: sensorTOOL 2-point (zero offset, sensitivity)

The system is linearized and should be adapted to the measurement environment in the machine.

Two measured points are compared with known reference distances. Based on this, the software calculates a linear function (straight line) that adjusts the zero offset and sensitivity.

This function produces a much more precise measurement over the entire range provided that the sensor behavior remains linear.

Proceed as follows:

- Select `Edit`. The fields can now be edited.
- Select `2-point`.
- Enter the correct reference values at `Point 1 (mm)` and `Point 2 (mm)`.
- Click on `Set value` and then on `Calculate and activate`.

The green checkmarks indicate whether the action was successful.

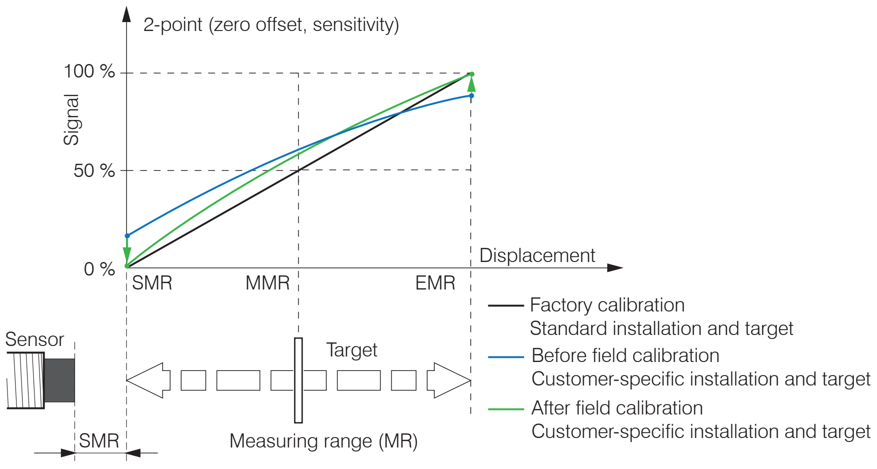
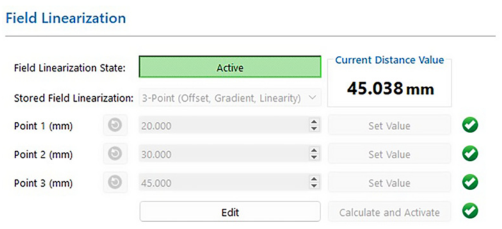


Fig. 8.7: 2-point (zero offset, sensitivity) sketch

8.2.2.3 3-point (zero offset, sensitivity, linearity)



Field Linearization

Field Linearization State:	Active	Current Distance Value 45.038 mm	
Stored Field Linearization:	3-Point (Offset, Gradient, Linearity)		
Point 1 (mm)	20.000	Set Value	✓
Point 2 (mm)	30.000	Set Value	✓
Point 3 (mm)	45.000	Set Value	✓
	Edit	Calculate and Activate	✓

Fig. 8.8: sensorTOOL 3-point (zero offset, sensitivity, linearity)

If the user changes the sensor or target geometry or if, for example, the material properties change, field linearization must be performed before the measurement. Here, use the following if possible:

- The original sensor assembly to be used in the measurement.
- The original target to be used in the measurement.

This function achieves the highest accuracy over the entire measuring range – even in the case of complex target materi-als or an unfavorable installation position.

Proceed as follows:

► Select `Edit`. The fields can now be edited.

► Select `3-point`.

► Enter the correct reference values at `Point 1 (mm)`, `Point 2 (mm)` and `Point 3 (mm)`.

► Click on `Set value` and then on `Calculate and activate`.

The green checkmarks indicate whether the action was successful.

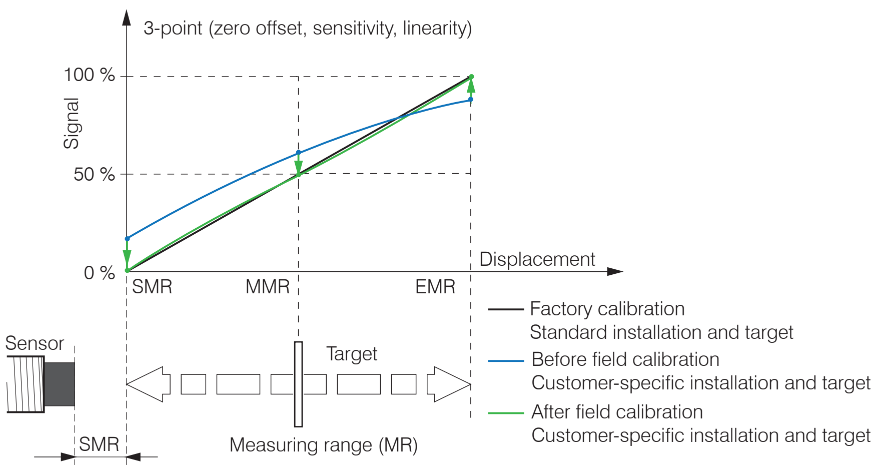


Fig. 8.9: 3-point (zero offset, sensitivity, linearity) sketch

8.3 Switching output: limit value monitoring

By configuring the limit value monitoring, the switching output can be activated or deactivated depending on the measured distance.

The sequence for the signal processing is shown in the following figure:

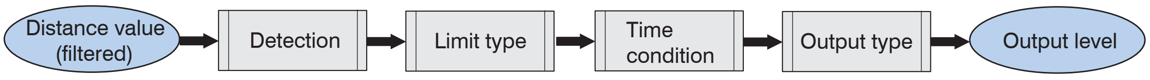
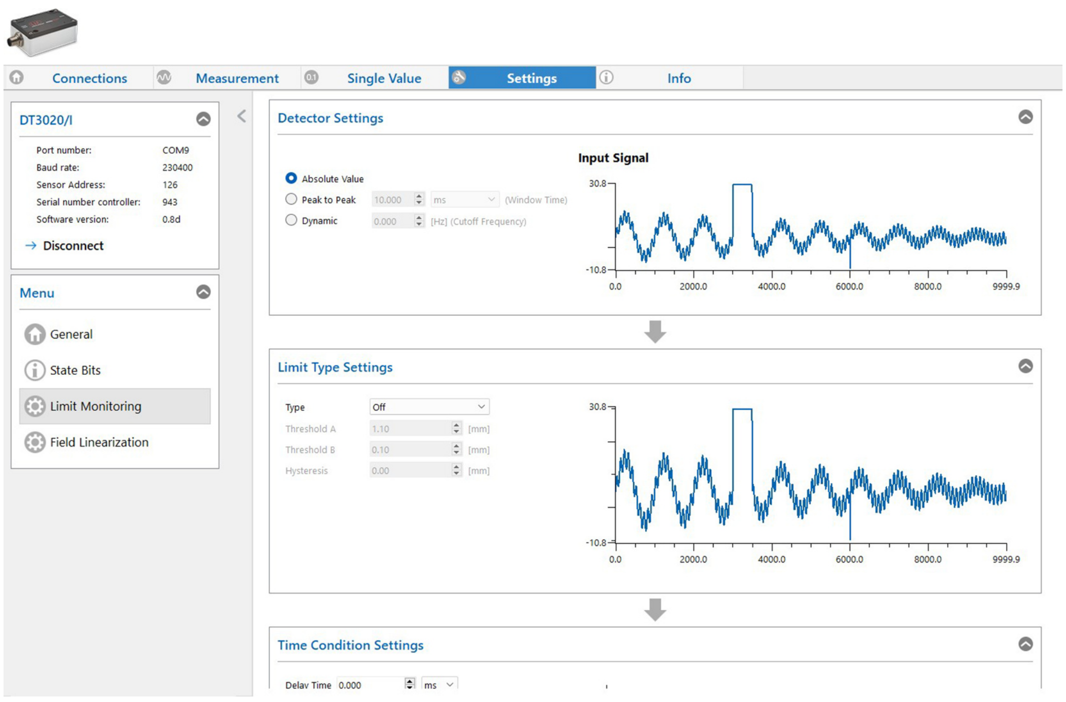


Fig. 8.10: Limit value monitoring signal processing



DT3020/I

Port number:	COM9
Baud rate:	230400
Sensor Address:	126
Serial number controller:	943
Software version:	0.8d

Detector Settings

Absolute Value [✓]
Peak to Peak 10.000 ms (Window Time)
Dynamic 0.000 [Hz] (Cutoff Frequency)

Limit Type Settings

Type	Off	
Threshold A	1.10	[mm]
Threshold B	0.10	[mm]
Hysteresis	0.00	[mm]

Time Condition Settings

Delay Time 0.000 ms

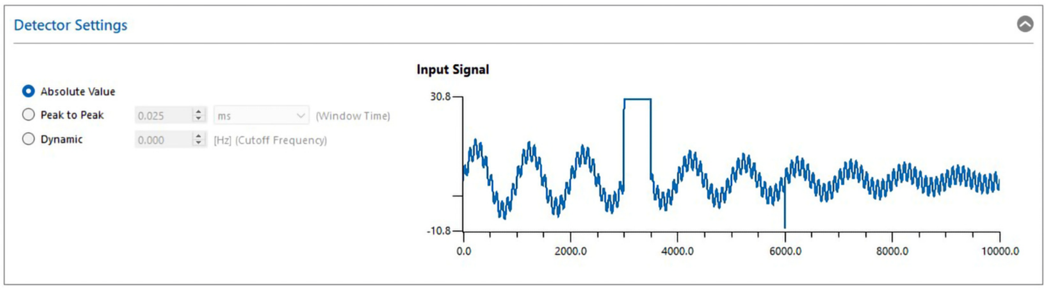
Fig. 8.11: sensorTOOL limit value monitoring

8.3.1 Detector settings

8.3.1.1 Absolute value

With the absolute value, there is no change to the distance value in the detector.

The measured distance value at the detector input corresponds to the distance value at the output.



Detector Settings

Absolute Value [✓]
Peak to Peak 0.025 ms (Window Time)
Dynamic 0.000 [Hz] (Cutoff Frequency)

Fig. 8.12: sensorTOOL detector absolute value

8.3.1.2 Peak to peak

When “Peak to peak” is selected, the difference between the maximum distance value and minimum distance value recorded within the window time t_{pkpk} is formed in the detector. Thus a new value is only formed in time intervals of window time t_{pkpk}. The signal is thus only applied at the output after a timespan of 1x *tpkpk*.

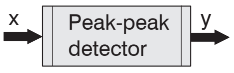


Fig. 8.13: Example of the peak-to-peak detector

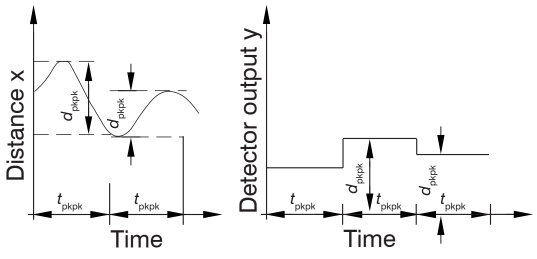


Fig. 8.14:

8.3.1.3 Dynamic

The detector “Dynamic” is a 1st order high-pass filter, the -3 dB frequency response of which is entered.

The figure below shows an example of how the detector filter works. The input signal consists of two sinusoidal signals with 0.1 Hz (period duration 10 s) and 4 Hz (period duration 0.25 s) and a direct component. A frequency response of 1 Hz is set for the detector. The direct component is fully suppressed. The low-frequency sinusoidal signal is barely visible at the output, whereas the high-frequency sinusoidal signal passes through the filter practically unchanged.

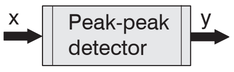


Fig. 8.15: Example of Dynamic detector (f_g ≈ 1 Hz)

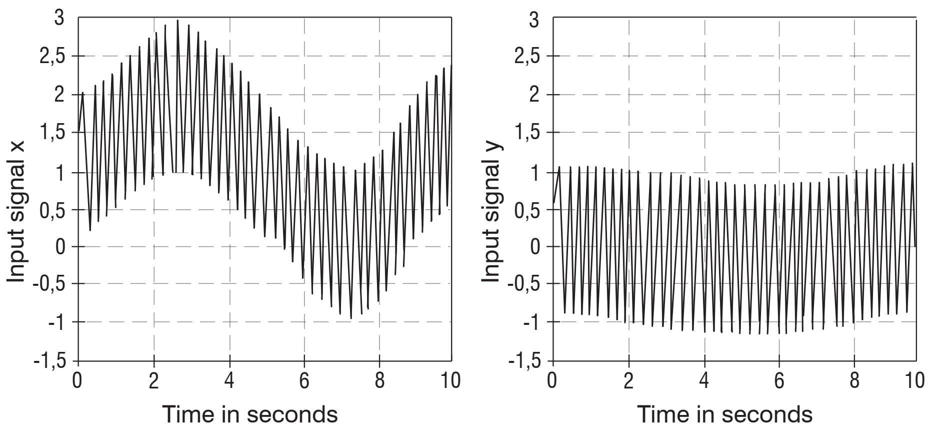


Fig. 8.16: Example of frequency suppression using the detector setting: Dynamic (f_g = 1 Hz)

In the following, a 0.1 Hz sinusoidal signal is superimposed with a short pulse of 0.1 s. The frequency response is again set to 1 Hz. The slower sinusoidal signal is suppressed. This makes the short pulse at the output stand out better.

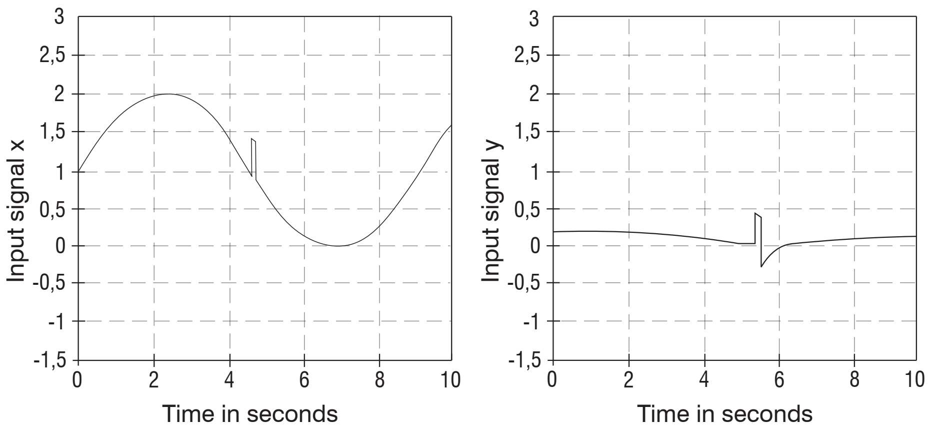


Fig. 8.17: Example of pulse detection using the detector setting: Dynamic (f_g = 1 Hz)

8.3.2 Limit value type settings

There are two basic types for the limit value test: *Limit* and *Window*. The “Limit” type checks for a limit value A. Here, a distinction is made as to whether the value is above or below the threshold value. This allows the result to be inverted.

If “Window” is selected, it is checked whether the threshold value is within or outside a defined window. Two limit values A and B that define this window are set for this check. Here, too, the results for selection within and outside are exactly inverted in relation to each other.

A hysteresis can also be specified for each limit value. In addition to the threshold value, this hysteresis must also be exceeded or not reached for the change from 1 to 0.

For the limit value type = Off, no limit value test is carried out, i.e. the status bit 6, see Chap. 10.5 is not set either.

- 0 = OK
- 1 = NOK

8.3.2.1 Limit value monitoring

The distance signal is checked with regard to whether a limit value has been exceeded or not reached depending on the setting. The distance value threshold to be checked is specified.

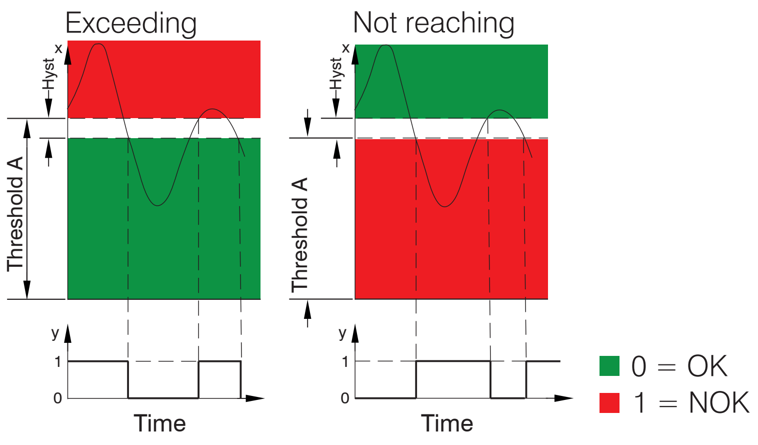


Fig. 8.18: Example for limit value type switching threshold exceeded and not reached

8.3.2.2 Windows

It is checked whether the value of the distance signal is within or outside a distance range. The range is defined by the two values limit value A and limit value B. Which of the two values is larger is irrelevant.

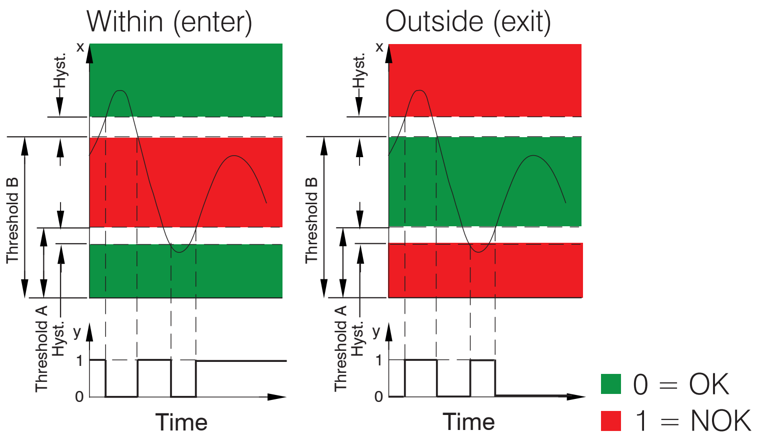


Fig. 8.19: Example of limit value type for window enter and exit

8.3.2.3 Hysteresis

A hysteresis can also be specified for each limit value. In addition to the threshold value, this hysteresis must also be exceeded or not reached for the change from 1 to 0. The graph below shows the result of an evaluation of a signal with the same threshold value but hysteresis of different sizes.

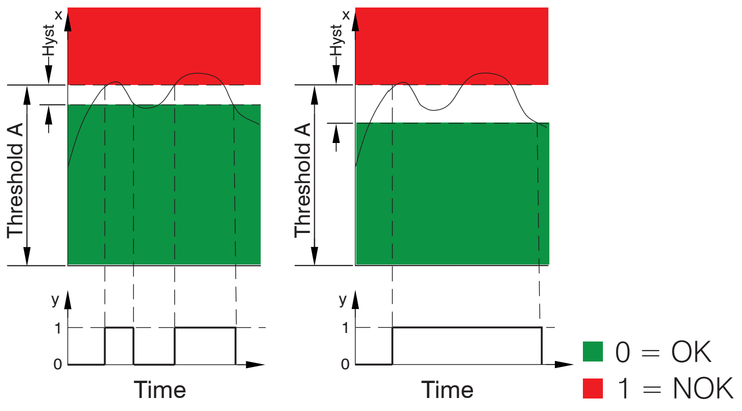


Fig. 8.20: Example of the effect of the hysteresis based on an example signal

8.3.3 Time condition settings

During the limit value test, the output can be delayed by specifying a time. This means, for example, that a limit value must be exceeded for at least as long as the specified delay time in order to change the output.

In addition, a minimum hold time as to how long the output should remain active after switching can be defined.

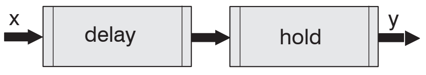


Fig. 8.21: Sequence for applying delay and hold to the limit value monitoring

The delay is first applied to the input signal. The limit value must therefore be exceeded for at least as long as the delay time in order to change the output and for the signal for the specified hold time to remain at 1 (switch in active state).

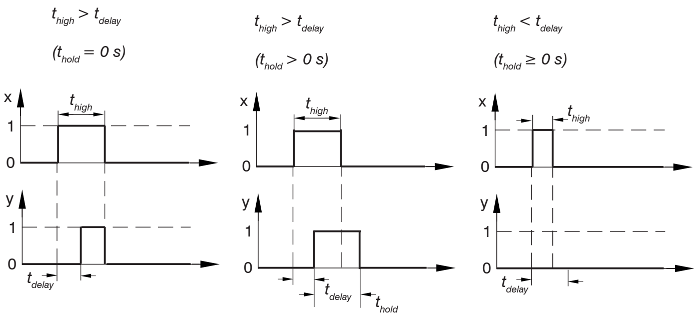


Fig. 8.22: Example for delay time > 0

The hold time indicates how long the signal remains at least at 1. If, for example, the limit value is exceeded for longer than the hold time, the output remains active exactly for this time. The hold time thus only has an effect if the limit value is exceeded for less than the specified hold time.

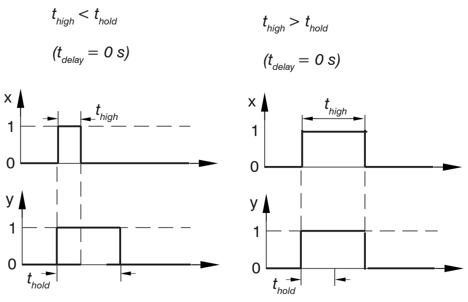
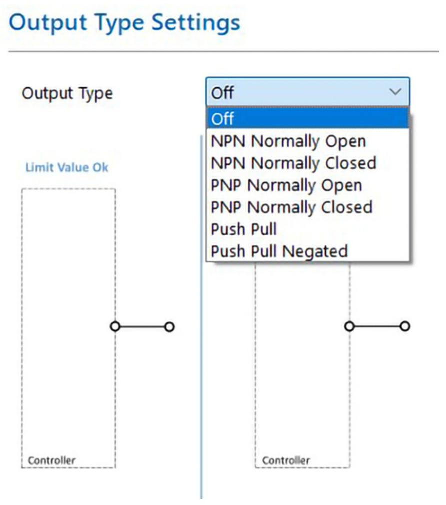


Fig. 8.23: Example for hold time > 0 (delay time = 0)

8.3.4 Output type settings

Selecting the output type determines the switching behavior of the switching output depending on the result of the limit value test. Closed means that the “GND switch” connects the switching output to GND, or, in the case of *Switch* V_+ the switching output is connected to the operating voltage of the DT3020.

If the output type *Off* is selected, the output is always at high impedance regardless of the other settings and the result of the limit value test. However, the limit value test still takes place and the status bit 6 changes the status depending on the result.


Output Type Settings

Output Type: Off

Fig. 8.24: sensorTOOL output type settings

Schematic diagram showing the output configuration

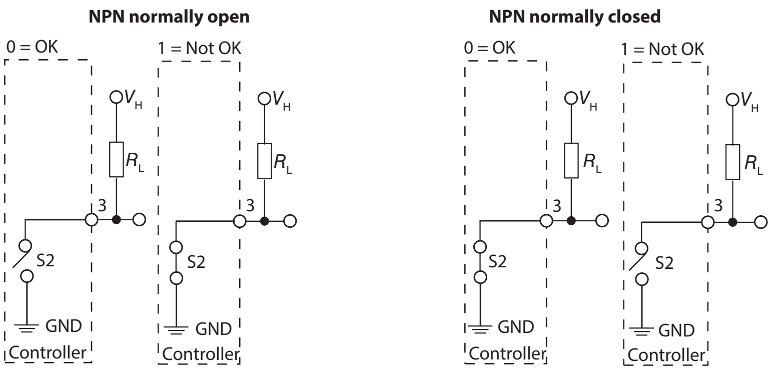


Fig. 8.25: Output type NPN

An RL resistor must be connected to a voltage (VH) by the customer.

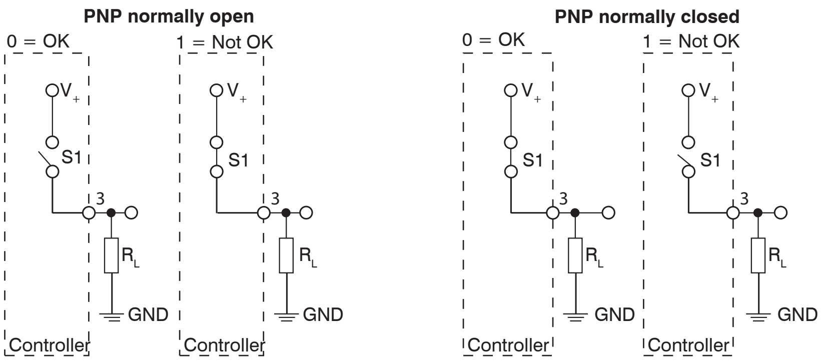


Fig. 8.26: Output type PNP

An RL resistor must be connected to GND by the customer.

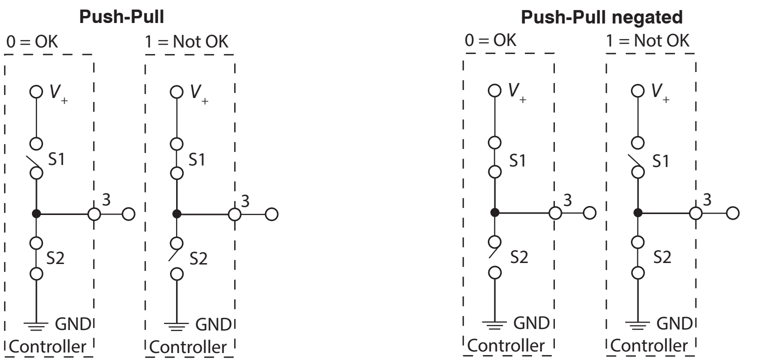


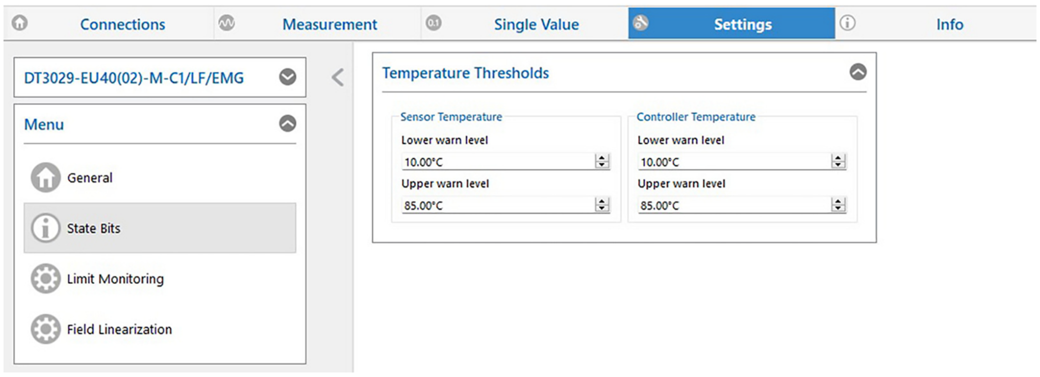
Fig. 8.27: Output type push-pull

Logical output value of the limit value test	0 = Limit value OK		1 = Limit value NOT OK	
Limit value type	Switch GND S2	Switch V_+ S1	Switch GND S2	Switch V_+ S1
Switching output deactivated, "Off"	open	open	open	open
NPN normally open	open	open	closed	open
NPN normally closed	closed	open	open	open
PNP normally closed	open	open	open	closed
PNP normally open	open	closed	open	open
Push-pull	closed	open	open	closed
Push-pull negated	open	closed	closed	open

Tab. 8.2: Truth table for the digital output, depending on the selected output type

8.4 *Status bits: Temperature warning thresholds*

The `Temperature warning thresholds` can be set via the menu on the left-hand side under the item `Status bits`. The temperature of the sensor and controller is monitored here. When a threshold is exceeded or not reached, this is output in the form of a status via the digital interface and as a warning message in `sensorTOOL`. When the temperature warning thresholds are exceeded, this cannot be displayed at the switching output.



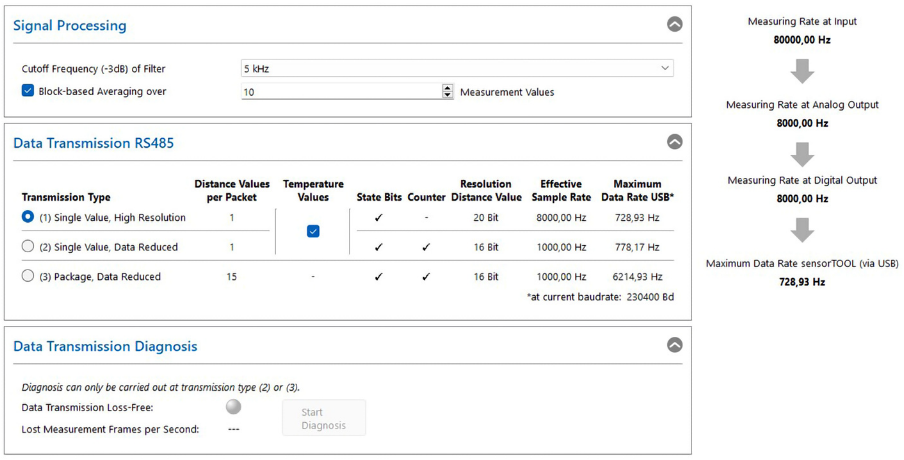
Sensor Temperature
Lower warn level: 10.00°C
Upper warn level: 85.00°C

Controller Temperature
Lower warn level: 10.00°C
Upper warn level: 85.00°C

Fig. 8.28: Setting the temperature warning thresholds

The status of the warning threshold can be displayed in `sensorTOOL` under `Single value`., see Chap. 7

8.5 Filters and measuring rates



Signal Processing

Cutoff Frequency (-3dB) of Filter: 5 kHz

[✓] Block-based Averaging over 10 Measurement Values

Data Transmission RS485

Transmission Type	Distance Values per Packet	Temperature Values	State Bits	Counter	Resolution Distance Value	Effective Sample Rate	Maximum Data Rate USB*
● (1) Single Value, High Resolution	1	✓	✓	-	20 Bit	8000,00 Hz	728,93 Hz
○ (2) Single Value, Data Reduced	1		✓	✓	16 Bit	1000,00 Hz	778,17 Hz
○ (3) Package, Data Reduced	15	-	✓	✓	16 Bit	1000,00 Hz	6214,93 Hz

*at current baudrate: 230400 Bd

Data Transmission Diagnosis

Diagnosis can only be carried out at transmission type (2) or (3).

Data Transmission Loss-Free:

Lost Measurement Frames per Second: ---

Measuring Rate at Input 80000,00 Hz

Measuring Rate at Analog Output 8000,00 Hz

Measuring Rate at Digital Output 8000,00 Hz

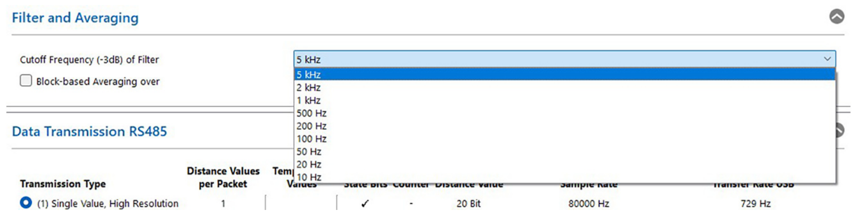
Maximum Data Rate sensorTOOL (via USB) 728,93 Hz

► Select the menu item `Filters and measuring rates` on the left-hand side.

8.5.1 Filters and averaging in the DT3020

Frequency response

The -3 dB frequency response of a low-pass filter can be selected by selecting `Frequency response (-3 dB) of the filter`. This limits the bandwidth of the distance signal, which is output at the analog output, the digital measurement values and the limit value test.



Filter and Averaging

Cutoff Frequency (-3dB) of Filter: 5 kHz

[ ] Block-based Averaging over

Data Transmission RS485

Transmission Type	Distance Values per Packet		State Bits	Counter	Distance Value	Sample Rate	Transfer Rate USB
● (1) Single Value, High Resolution	1		✓	-	20 Bit	80000 Hz	729 Hz

Fig. 8.29: Frequency response of the filter

Values within a range from 10 Hz to 5 kHz can be selected for the frequency response -3 dB. This setting influences the measurement values output with analog and digital means and the measurement values used for the limit value test. The filter setting does not affect the internal data rate for the signal processing (analog output and limit value test). However, an individual reduction in the data rate for the output can be set for each filter setting. In addition to the bandwidth, the filter setting also influences the delay of the output of measurement values and the limit value test.

Filter setting -3 dB frequency response	t_{delay} Delay	t_{rise} Rise time
5 kHz	0.13 ms	0.07 ms
2 kHz	0.32 ms	0.19 ms
1 kHz	0.54 ms	0.38 ms
500 Hz	0.98 ms	0.76 ms
200 Hz	2.3 ms	1.9 ms
100 Hz	4.6 ms	3.8 ms
50 Hz	9.0 ms	7.6 ms
20 Hz	22.3 ms	19 ms
10 Hz	44.6 ms	38 ms

Tab. 8.3: Typical values for the signal delay and rise time depending on the selected bandwidth

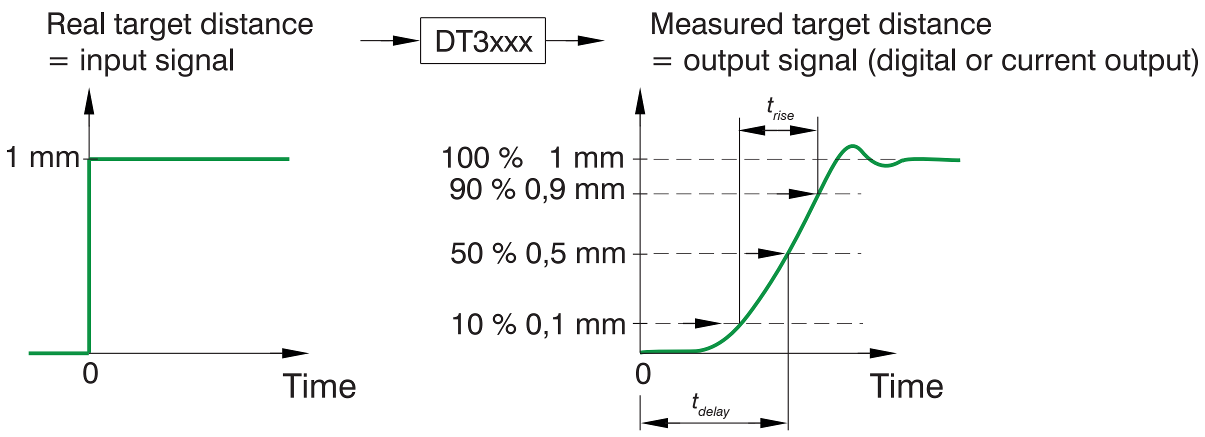


Fig. 8.30: Filter effect sketch

The percentages in the sketch above relate to the signal jump.

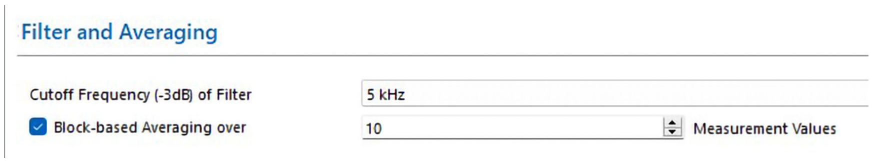
Example of signal jump from 0.5 to 0.8 mm:

- 0 % = 0.5 mm
- 10 % = 0.53 mm
- 50 % = 0.65 mm
- 90 % = 0.77 mm
- 100 % = 0.8 mm

Block-based averaging over x measurement values

The calculation of a block-wise arithmetic mean value for the measured distance values can additionally be activated. The averaging limits the bandwidth and thus the noise of the measurement signal and also reduces the data rate of the measurement values. The block-wise averaging also affects the analog output, the digitally output measurement values and the limit value test.

With a setting of 10, a mean value is calculated over 10 measurement values. The measuring rate is thus reduced from 80 kSa/s to 8 kSa/s.



Filter and Averaging

Cutoff Frequency (-3dB) of Filter	5 kHz	
✓ Block-based Averaging over	10	Measurement Values

Fig. 8.31: Block-based averaging

In the DT3020, it is possible to perform averaging over a maximum of 4096 values.

If the mean value is calculated over *N* values, a new mean value is only produced after *N* values. This means that the data rate is reduced by the factor *N*.

$$M(j) = \frac{1}{N}\sum_{k=0}^{N-1} d(j * N + k)$$

d(k) = kth distance measurement value
N = Number of averaging values
M(j) = jth mean value

Fig. 8.32: Formula for arithmetic mean value

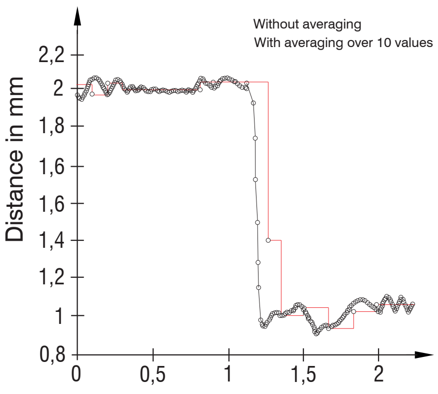
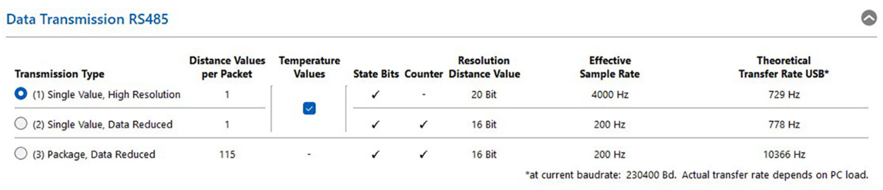


Fig. 8.33: Signal with averaging over 10 values

8.5.2 Data transmission RS485

In the `Data transmission RS485` drop-down menu, it is selected what measurement data is to be transmitted and displayed in sensorTOOL and in what format it is to be transmitted.



Data Transmission RS485

Transmission Type	Distance Values per Packet	Temperature Values	State Bits	Counter	Resolution Distance Value	Effective Sample Rate	Theoretical Transfer Rate USB*
[✓ (1) Single Value, High Resolution]	1	☑	✓	-	20 Bit	4000 Hz	729 Hz
(2) Single Value, Data Reduced	1		✓	✓	16 Bit	200 Hz	778 Hz
(3) Package, Data Reduced	115	-	✓	✓	16 Bit	200 Hz	10366 Hz

*at current baudrate: 230400 Bd. Actual transfer rate depends on PC load.

Fig. 8.34: sensorTOOL data transmission

Transmission method	Distance values per package	Temperature values	Status bits	Counter	Resolution distance value
Individual, high resolution	1	x (optional)	x	-	20
Individual, data reduced	1	x (optional)	x	x	16
Packet, data reduced	115	-	x	x	16

Tab. 8.4: Factory setting for subsampling depending on the bandwidth

The `Effective sampling rate` depends on the selected settings. The `Theoretical transfer rate USB` depends on the current baud rate, the real transfer rate depends on the PC usage.

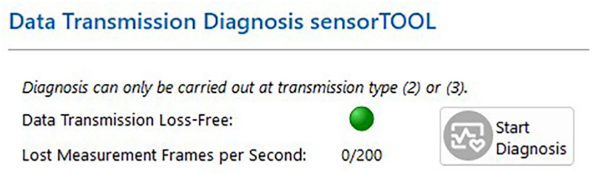
The highest data rate is achieved with the setting *Packet, data reduced*. To achieve the highest transfer rate, the maximum block size of 115 distance values per packet is set. This setting is possible via the config parameter 50067, see Chap. 10

8.5.3 Data transmission diagnostics sensorTOOL

To execute this function, click on the `Start diagnosis` button.

The data transmission diagnostics only relate to the transmission types `(2) Single, data reduced` and `(3) Packet, data reduced`. With this function, you can check whether measurement value frames have been lost during the data output. Loss-free transmission is indicated with a green dot, moderate data loss (<1% of the measurement values) with a yellow dot and critical data loss with a red dot (the quality of the measurement on the `sensorTOOL` is not ensured). You will get a message informing you of how many frames were lost.

In the event of a yellow or red dot, apply additional settings if necessary, see Chap. 8.5.1.



Data Transmission Diagnosis sensorTOOL

Diagnosis can only be carried out at transmission type (2) or (3).

Data Transmission Loss-Free: ●

Lost Measurement Frames per Second: 0/200

Fig. 8.35: sensorTOOL data transmission diagnostics

9 Info in the sensorTOOL

In this view, you can get more information on the connected system, e.g. controller, sensor, diagnostic and characteristic curve values.

The information can also be copied to the clipboard, the system can be reset to factory settings and wait times for the ME-Bus import can be set.

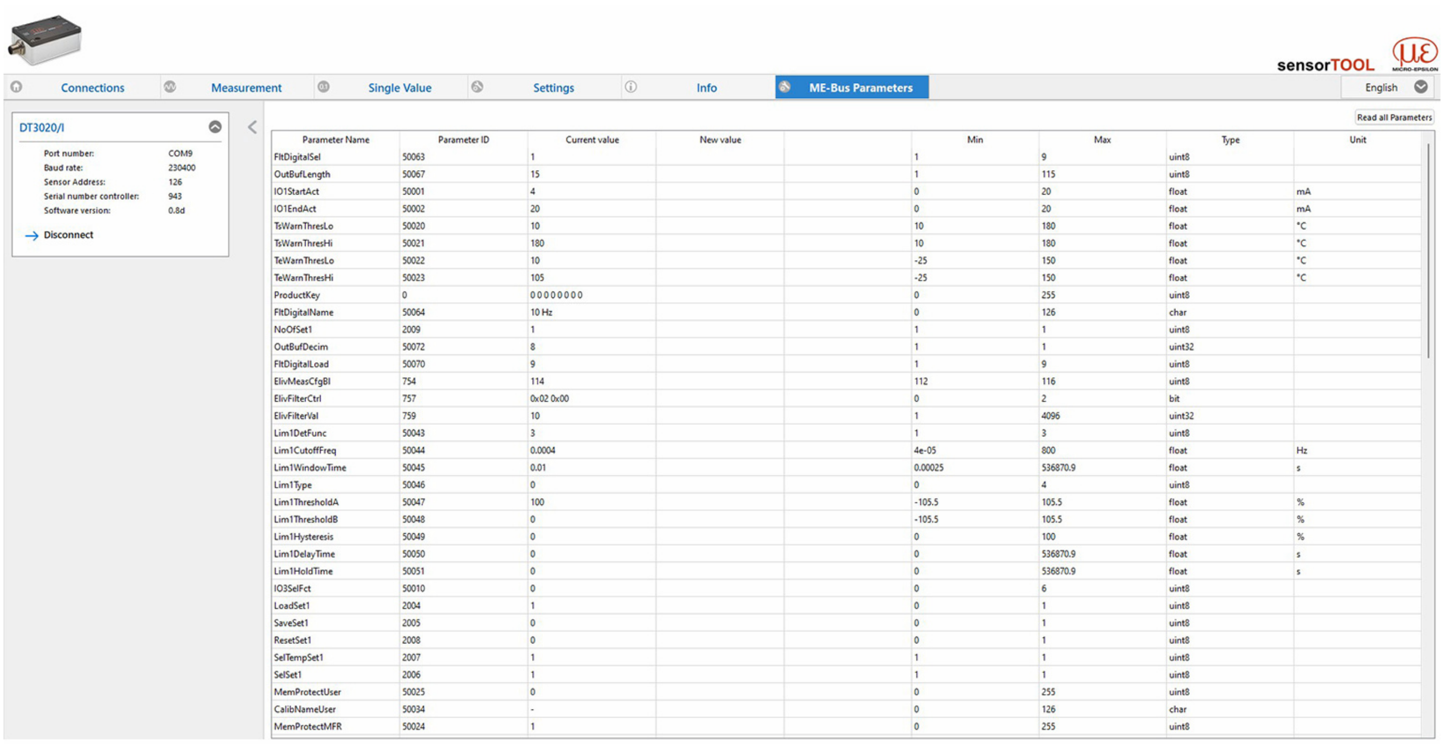
	Clicking the `Copy to clipboard` button copies the information and settings for the selected controller to the clipboard.
	By confirming the `Factory settings` button, you can restore the factory state. All deactivated channels are reactivated, and special channel-related settings are reset. A window opens with a table showing all parameters, with the reset parameters highlighted and the old and new values displayed.

10 Config parameters

You can access the config parameters in `sensorTOOL`.

You can also access the config parameters via the IF2035 interface module and via the MEDAQLib software and driver library (SDK), see Chap. 11. The config parameters are relevant for operating the DT3020, as they can be used to adjust settings in the device.

The additional menu item *ME-Bus parameters* can be activated with the key combination `CTRL+SHIFT+F9` in `sensor-TOOL`.



Port number:	COM9
Baud rate:	230400
Sensor Address:	126
Serial number controller:	943
Software version:	0.8d

Parameter Name	Parameter ID	Current value	New value	Min	Max	Type	Unit
FltDigitalSel	50063	1		1	9	uint8	
OutBufLength	50067	15		1	115	uint8	
IO1StartAct	50001	4		0	20	float	mA
IO1EndAct	50002	20		0	20	float	mA
TsWarnThresLo	50020	10		10	180	float	°C
TsWarnThresHi	50021	180		10	180	float	°C
TeWarnThresLo	50022	10		-25	150	float	°C
TeWarnThresHi	50023	105		-25	150	float	°C
ProductKey	0	00000000		0	255	uint8	
FltDigitalName	50064	10 Hz		0	126	char	
NoOfSet1	2009	1		1	1	uint8	
OutBufDecim	50072	8		1	1	uint32	
FltDigitalLoad	50070	9		1	9	uint8	
ElivMeasCfgBl	754	114		112	116	uint8	
ElivFilterCtrl	757	0x02 0x00		0	2	bit	
ElivFilterVal	759	10		1	4096	uint32	
Lim1DetFunc	50043	3		1	3	uint8	
Lim1CutoffFreq	50044	0.0004		4e-05	800	float	Hz
Lim1WindowTime	50045	0.01		0.00025	536870.9	float	s
Lim1Type	50046	0		0	4	uint8	
Lim1ThresholdA	50047	100		-105.5	105.5	float	%
Lim1ThresholdB	50048	0		-105.5	105.5	float	%
Lim1Hysteresis	50049	0		0	100	float	%
Lim1DelayTime	50050	0		0	536870.9	float	s
Lim1HoldTime	50051	0		0	536870.9	float	s
IO3SelFct	50010	0		0	6	uint8	
LoadSet1	2004	1		0	1	uint8	
SaveSet1	2005	0		0	1	uint8	
ResetSet1	2008	0		0	1	uint8	
SelTempSet1	2007	1		1	1	uint8	
SelSet1	2006	1		1	1	uint8	
MemProtectUser	50025	0		0	255	uint8	
CalibNameUser	50034	-		0	126	char	
MemProtectMFR	50024	1		0	255	uint8	

Fig. 10.1: ME-Bus parameters

You can read out the sensor parameters using the `Read all parameters` button in the top right.

You can specify values for the respective parameters via the `New value` field and send them to the sensor using the command `Send`

10.1 Perform field linearization with config parameters

Field linearization can be performed in the DT3020 controller with 1, 2 or 3 points.

Field linearization process

Write the number *n* for the field linearization to be performed into the parameter ID 50035

n	=	Number of points used for a field linearization (1 for 1-point, 2 for 2-point and 3 for 3-point field linearization.
x	=	The distance *d* relates to the measuring point (1 ... n).
d	=	Distance between sensor and target at measuring point *x*. A nominal value and an actual value therefore need to be entered; see following process description. These values must be within the measuring range.

- Set the desired distance *d* between the sensor and the target for point *x*.
- Write the target distance *d* just set in mm into the parameter ID 50040.
- Write the number of point *x* into the parameter ID 50038.
- Repeat steps 1 to 3 for all points *x* = 1 ... *n*.
- Start the calculation of the *n*-point field linearization when all points have been set. For this purpose, set the value *1* in the parameter ID 50039.

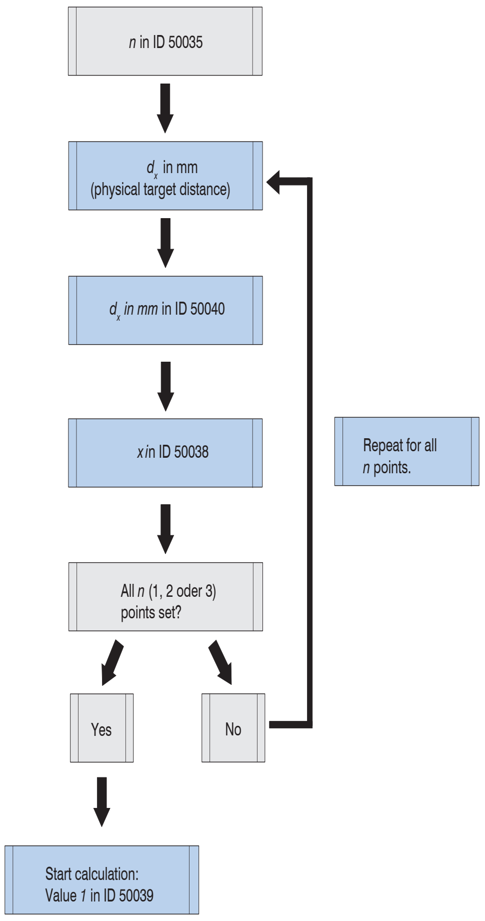


Fig. 10.2: Perform field linearization via config parameters

The status of the calculation can then be read from the parameter ID 50041.

If 0 is read, the field calibration was performed successfully and saved. Otherwise, the read value is interpreted as a bit field. The meaning of the individual bits is given in the table below.

Bit no.	Meaning 0	Meaning 1	Solution
0	No error	Set distance values identical	Set different distances between the sensor and target for each point (1/2/3).
1	No error	Distance values not strictly monotonic.	Repeat the field linearization. Make sure that the real, physical distance between the front of the sensor and the target matches the entered values d in ParamID 50040. Maintain this distance while the value *x* is being written into the ParamID 50038.
2	No error	Distance values in limitation	Values are outside the sensor measuring range. Set values that are within the measuring range of the relevant sensor.
3	No error	Invalid number (NaN) contained	
4	No error	Invalid type in ParamID 50035	Write valid values: 1, 2 or 3
5	No error	Field calibration could not be saved because the write protection is set in ParamID 50025.	Disable the write protection in ParamID 50025.

Tab. 10.1: Table for interpreting the value ID 50041

Note The field calibration can be enabled with parameter ID 50068 = 0 and disabled with = 1. The characteristic curve can only be switched on if a valid field calibration is stored. A valid field calibration remains saved even if it is disabled.

For information purposes, the type of the field linearization currently saved in the current characteristic curve is displayed in the parameter ID 50036:

0	=	No field linearization saved.
1	=	1-point field linearization
2	=	2-point field linearization
3	=	3-point field linearization

10.2 Signal processing: subsampling

This setting is possible via the config parameter ID50072. It reduces the data rate at which the digital interface outputs the data. The setting has no effect on the analog output or limit value monitoring.

This only relates to the data rates of the measurement values in the output formats `Individual, data reduced` and `Packet, data reduced`. The data rate for `Individual, data reduced` is not influenced by this setting.

The measuring rate of the DT3020 is 80 kSa/s. At a set subsampling of 8, only every 8th value is output by the DT3020. The values read via the digital interface therefore have a data rate of 10 kSa/s.

Filter type	Subsampling	Output data rate data-reduced
5 kHz	8	10 kSa/s
2 kHz	20	4 kSa/s
1 kHz	40	2 kSa/s
500 Hz	80	1 kSa/s
200 Hz	80	1 kSa/s
100 Hz	80	1 kSa/s
50 Hz	160	500 Sa/s
20 Hz	400	200 Sa/s
10 Hz	800	100 Sa/s

Tab. 10.2: Factory setting for subsampling depending on the bandwidth

The subsampling setting only applies to the following data formats:

- `Individual, data reduced`
- `Packet, data reduced`

The setting is saved individually in the DT3020 for every filter type. A maximum value of from 1 to 65535 can be set.

10.3 List of available config parameters

Param ID	Data type	Name DE	Unit	Description DE	Access	min	max
Frequency response distance measurement							
50063	uint8	Filter selection		Selection of the digital filter. The selection is activated and saved immediately.	rw	1	9
50070	uint8	Load digital filter name		Loads the name of the digital filter with this number into the parameter 50064	rw	1	9
50064	char[32]	Filter name		Contains the name of the filter loaded with 50070 (not of the currently active filter)	ro	0	126
Field linearization							
50035	uint8	Set field linearization type		Sets the type of field linearization to be carried out 1 = 1-point / offset 2 = 2-point 3 = 3-point	rw	1	3

Param ID	Data type	Name DE	Unit	Description DE	Access	min	max
50036	uint8	Read field linearization type		Type of saved field linearization 0 = No field linearization available 1 = 1-point / offset 2 = 2-point 3 = 3-point	ro	0	3
50037	uint8	Read distance value support point		Loads distance value of point 1 to 3 into parameter 50040	rw	1	3
50038	uint8	Set distance value support point		When writing the number, the point of the field calibration is set to the value pair consisting of the current measured distance value and FieldLinValue.	rw	1	3
50039	uint8	Save field linearization		0 = No action 1 = Calculation of the field calibration is started. If this is successful, the calibration is saved and also activated.	ro	0	1
50040	float	Distance support point	mm	Reads the value loaded with 50038. Writes the value with 50038 as a support point of the field calibration.	rw	-3.40E+38	3.40E +38
50041	uint16	Error codes of the field linearization		Result of the calculation of the field calibration after writing 1 in 50039. 0 = Field calibration successfully performed. Individual bits represent errors during the calibration. Bit 0 = 0x0001 set distance values identical Bit 1 = 0x0002 internal distance values not strictly monotone Bit 2 = 0x0004 internal distance values in limitation Bit 3 = 0x0008 contains an invalid number (NaN) Bit 4 = 0x0010 invalid type in FieldLinTypeWr (ParamID 50035) Bit 5 = 0x0020 could not be saved because the write protection is set in ParamID 50025.	ro	0	65535
50068	uint8	Activate field linearization		0 = Deactivate field linearization 1 = Activate field linearization	rw	0	1
Limit switch							
50010	uint8	Switching output type		Selects the function of the switching output 0 = Off 1 = NPN normally open 2 = NPN normally closed 3 = PNP normally open 4 = PNP normally closed 5 = PushPull 6 = PushPull negated	rw	0	6
50051	float	Hold time	s	The limit switch remains active at least for this time after responding.	rw	0	3.40E +38
50043	uint8	Detector		1 = Absolute value 2 = PeakToPeak block-wise over the time period set in 50045 3 = Dynamically filtered high pass, with frequency response set in 50044	rw	1	3
50044	float	High pass frequency response	Hz	Frequency response of 1st order high-pass filter for the detector “Dynamic”	rw	0	3.40E +38
50045	float	Observation time	s	Observation time for determining the peak-to-peak value for the detector “PeakToPeak”	rw	0	3.40E +38

Param ID	Data type	Name DE	Unit	Description DE	Access	min	max
50046	uint8	Limit value type		0 = Limit value test switched off 1 = Limit value exceeded 2 = Limit value not reached 3 = Window exit 4 = Window enter	rw	0	4
50047	float	Limit value	%	Limit value against which the measured distance value is checked.	rw	-105.5	105.5
50048	float	Limit value B	%	Second value that determines the window for the limit value types “Window exit” and “Window enter” in combination with the limit value A 50047.	rw	-105.5	105.5
50049	float	Hysteresis	%	Hysteresis of the limit value test	rw	0	100
50050	float	Response delay	s	The limit value switch responds with a delay at this time.	rw	0	3.40E +38
2000	uint16	Config parameter version		Version of the general parameters = 2	ro	1	65535
2002	uint16	Parameter group identification		Identification number for eddy current product group = 2	ro	1	65535
2003	uint16	Parameter group version		Version of the group-specific parameters = 1	ro	1	65535
Characteristic curve management							
2004	uint16	Load characteristic curve		Loads the information from the characteristic curve with the specified number into the memory. There is only one characteristic curve memory space for DT3020.	rw	0	1
2006	uint8	Permanent characteristic curve selection		Selection of the currently active characteristic curve with the specified number. This selection is permanently saved and the characteristic curve is thus also active during the next start-up.	rw	1	1
2007	uint8	Temporary characteristic curve selection		Temporary selection of the currently active characteristic curve. After a restart, the selection is lost and the characteristic curve selected with ParamID 2006 is active again.	rw	1	1
2005	uint8	Save characteristic curve		Saves the changes in the characteristic curve with the specified number. Relates to the two parameters 50025 (user write protection) an 50034 (user characteristic curve designation).	rw	0	1
2008	uint8	Characteristic curve reset		0 = No action 1 = Reset characteristic curve to factory setting (field linearization is deleted, measuring ranges are reset to nominal values, ...)	rw	0	1
2009	uint8	Number of characteristic curve memory spaces		Number of characteristic curve memory spaces available in the device	ro	1	1
50024	uint8	Write protection factory characteristic curve		0 = Characteristic curve is not write-protected 1 = Characteristic curve is write-protected by manufacturer (factory characteristic curve)	ro	0	255
50025	uint8	Write protection user		0 = full access 1 = delete protection 2 = read-only If ParamID 50024 = 1, then 0 and 1 are synonymous, as a factory characteristic curve cannot be deleted.	rw	0	255
Information about the device and sensor							
166	int32	Sensor article number		Article number of the eddy current sensor	ro	0	2.15E +09

Param ID	Data type	Name DE	Unit	Description DE	Access	min	max
310	float	Sensor temperature nominal measuring range	°C	Factory setting for the measuring range of the sensor temperature.	ro	-3.40E+38	3.40E +38
342	float	Sensor temperature nominal offset	°C	Factory setting for the offset of the sensor temperature.	ro	-3.40E+38	3.40E +38
311	float	Controller temperature nominal measuring range	°C	Factory setting for the measuring range of the controller temperature	ro	-3.40E+38	3.40E +38
343	float	Controller temperature nominal offset	°C	Factory setting for the offset of the controller temperature	ro	-3.40E+38	3.40E +38
646	int32	1. Sensor cable article number		Article number of the sensor cable to be used	ro	0,00E+00	2.15E +09
647	int32	2. Sensor cable article number		Sensor cable EC-	ro	0,00E+00	2.15E +09
648	int32	3. Sensor cable article number		Extension cable ECE-	ro	0,00E+00	2.15E +09
649	int32	4. Sensor cable article number		or adapter cable ECA-	ro	0,00E+00	2.15E +09
650	int32	5. Sensor cable article number		-1 means that the cable is not used.	ro	0,00E+00	2.15E +09
651	int32	6. Sensor cable article number			ro	0,00E+00	2.15E +09
198	int32	Sensor option number		Option number of the eddy current sensor	ro	0,00E+00	2.15E +09
662	int32	1. Sensor cable option number		Option number of the sensor cable to be used	ro	0,00E+00	2.15E +09
663	int32	2. Sensor cable option number		Sensor cable EC-	ro	0,00E+00	2.15E +09
664	int32	3. Sensor cable option number		Extension cable ECE-	ro	0,00E+00	2.15E +09
665	int32	4. Sensor cable option number		or adapter cable ECA-	ro	0,00E+00	2.15E +09
666	int32	5. Sensor cable option number		-1 means that the cable is not used.		0,00E+00	2.15E +09
667	int32	6. Sensor cable option number			ro	0,00E+00	2.15E +09
262	int32	Sensor serial number		Serial number of the eddy current sensor	ro	0,00E+00	2.15E +09
694	int32	1. Sensor cable serial number		Serial number of the sensor cable to be used	ro	0,00E+00	2.15E +09
695	int32	2. Sensor cable serial number		Sensor cable EC-	ro	0,00E+00	2.15E +09
696	int32	3. Sensor cable serial number		Extension cable ECE-	ro	0,00E+00	2.15E +09
697	int32	4. Sensor cable serial number		or adapter cable ECA-	ro	0,00E+00	2.15E +09
698	int32	5. Sensor cable serial number		-1 means that the cable is not used.		0,00E+00	2.15E +09
699	int32	6. Sensor cable serial number			ro		2.15E +09
50026	uint8	Calibration date Day		Date of factory calibration - day	ro	1	31
50027	uint8	Calibration date Month		Date of factory calibration - month	ro	1	12

Param ID	Data type	Name DE	Unit	Description DE	Access	min	max
50028	uint8	Calibration date Year		Date of factory calibration - year	ro	0	255
50029	int32	Calibration article number		Article number of the factory calibration	ro	0	2.15E +09
50030	int32	Calibration option number		Option number of the factory calibration	ro	0,00E+00	2.15E +09
50031	int32	Calibration serial number		Unique, consecutive serial number of the factory calibration	ro	0,00E+00	2.15E +09
50032	int32	Calibration identification number		Unique identification number of the calibration type (contains target, measuring range, temperature compensation range, ...)	ro	0,00E+00	2.15E +09
50033	uint8	Characteristic curve valid		0 = No valid calibration saved 1 = Valid calibration saved	ro	0,00E+00	1
454	char[32]	Sensor name		Name of the eddy current sensor used	ro	0	126
50034	char[32]	Characteristic curve user description		User-writable text with max. 32 characters	rw	0	126
50069	char[32]	Description of field calibration		User-writable text with max. 32 characters	ro	0,00E+00	126
50071	float	Total length of the sensor cable	mm	Total length of all cables used for the calibration (integrated sensor cable + sensor cable EC- + extension cable ECE- + adapter cable ECA-)	ro	1.18E-38	3.40E +38
422	char[32]	Target material		Description of target material against which the sensor is calibrated.	ro	0,00E+00	1.26E +02
294	float	Nominal measuring range distance	mm	Factory setting for the measuring range of the distance	ro	-3.40E+38	3.40E +38
358	float	Current measuring range distance	mm	Currently used measuring range of the distance	ro	-3.40E+38	3.40E +38
326	float	Nominal offset distance	mm	Factory setting for the offset of the distance	ro	-3.40E+38	3.40E +38
390	float	Current offset distance	mm	Currently used offset of the distance	ro	-3.40E+38	3.40E +38
Block-wise averaging							
757	bit[2]	Averaging status		0 = Averaging off 2 = Averaging on	rw	0	2
759	uint32	Averaging number of values		Number of measurement values [1 ... 4096] over which a block-wise arithmetic mean is formed.	rw	1	4096
Scale measuring range							
50001	float	Current start of the output range	mA	Current output current of the current output at SMR	rw	4	20
50002	float	Current end of the output range	mA	Current output current of the current output at EMR	rw	4	20
50061	float	User measuring range	mm	User-adjustable measuring range of the distance measurement	rw	0	2
500062	float	User offset	mm	User-adjustable offset of the distance measurement	rw	0.2	2.2
50003	float	Factory setting start of the output range	mA	Factory setting for the output current at the current output at SMR	ro	4	20
50004	float	Factory setting end of the output range	mA	Factory setting for the output current at the current output at EMR	ro	4	20
Digital output of measurement values							

Param ID	Data type	Name DE	Unit	Description DE	Access	min	max
754	uint8	Output data format, see Chap. 10.4.1		0x70 Multiple distances from FIFO, state, counter 0x71 Single high resolution distance, sensor temp., electronic temp., state 0x72 Single high resolution distance, state 0x73 Single distance from FIFO, state, counter 0x74 Single distance from FIFO, sensor temp., electronic temp, state, counter	rw	112	116
2016	float	Output Data Rate	Hz	Output data rate at the digital interface	ro	0	4.29E +09
50067	uint8	Output buffer length		Maximum number of measurement values saved in the FIFO output buffer. This setting is only used if the output format 754 is set to 0x70 and the measurement values are output in blocks.	rw	1	115
50072	uint32	Output buffer data reduction		Only every nth measured distance value is transferred to the output buffer. Output data rate = measurement data rate/x. This setting is only used if the output format 754 is set to 0x70, 0x73 or 0x74.	rw	1	65535
50042	uint16	System state		Bit 0 = 0x0001 measurement signal outside the measuring range Bit 1 = 0x0002 no sensor present / breakage of the sensor cable Bit 2 = 0x0004 sensor temperature outside the limits (factory default) Bit 3 = 0x0008 electronics temperature outside the limits (factory default) Bit 4 = 0x0010 sensor temperature outside the user limits (50020, 50021) Bit 5 = 0x0020 electronics temperature outside the user limits (50022, 50023) Bit 6 = 0x0040 measurement value outside the limits of the limit value check	ro	0	65535
0	uint8[8]				rw	0	255
Warning level for the sensor and controller temperature							
50020	float	Sensor temperature lower warning threshold	°C	User-settable lower warning thresholds for the sensor temperature. If the sensor temperature is lower, the corresponding status bit is set in 50042.	rw	10	180
50021	float	Sensor temperature upper warning threshold	°C	User-settable upper warning thresholds for the sensor temperature. If the sensor temperature is higher, the corresponding status bit is set in 50042.	rw	10	180
50022	float	Controller temperature lower warning threshold	°C	User-settable lower warning thresholds for the controller temperature. If the controller temperature is lower, the corresponding status bit is set in 50042.	rw	-25	150
50023	float	Controller temperature upper warning threshold	°C	User-settable upper warning thresholds for the controller temperature. If the controller temperature is higher, the corresponding status bit is set in 50042.	rw	-25	150

10.4 Digital values

10.4.1 Output data formats

Overview of the data formats that can be set in the config parameter 754.

Parameter ID754[14]	Block no.[15]	Format
0x70	0x80	Multiple distances from FIFO, state, counter
0x71	0x81	Single high resolution distance, sensor temp., electronic temp., state
0x72	0x82	Single high resolution distance, state
0x73	0x83	Single distance from FIFO, state, counter
0x74	0x84	Single distance from FIFO, sensor temp., electronic temp, state, counter

Data format block 0x80 (Config Parameter ID754 = 0x70)

Data type	Name	Description
uint8	system state	State of the DT3020, see Chap. 10.5
uint8	filler	always zero
uint16	measurement counter	Measurement counter value of the first (oldest) measurement in the data package.
uint8	number values	Number of measurement values *n* received
uint8	filler	always zero
int16	distance[0]	
...		
....		
int16	distance[n-1]	

Data format block 0x81 (Config Parameter ID754 = 0x71)

Data type	Name	Description
uint8	system state	State of the DT3020, see Chap. 10.5
uint8	filler	always zero
uint8	filler	always zero
uint8	filler	always zero
int32	distance	high resolution distance
uint16	sensor temperature	
uint16	electronic temperature	

Data format block 0x82 (Config Parameter ID754 = 0x72)

Data type	Name	Description
uint8	system state	State of the DT3020, see Chap. 10.5
uint8	filler	always zero
uint8	filler	always zero
uint8	filler	always zero
int32	distance	high resolution distance

Data format block 0x83 (Config Parameter ID754 = 0x73)

Data type	Name	Description
uint8	system state	State of the DT3020, see Chap. 10.5
uint8	filler	always zero
uint16	measurement counter	The internal measurement counter is incremented with every sample.
int16	distance	distance

Data format block 0x84 (Config Parameter ID754 = 0x74)

[14] The config parameter ID can be used to change the output block for the cyclical data exchange.

[15] Read with MEDAQLib command Get_AlternateMeasure or ME bus command Read block.

Data type	Name	Description
uint8	system state	State of the DT3020, see Chap. 10.5
uint8	filler	always zero
uint16	measurement counter	The internal measurement counter is incremented with every sample.
int16	distance	distance
uint16	sensor temperature	
uint16	electronic temperature	

Calculation of the distance and temperature values from the received measurement values.

10.4.2 Calculation of the distance value

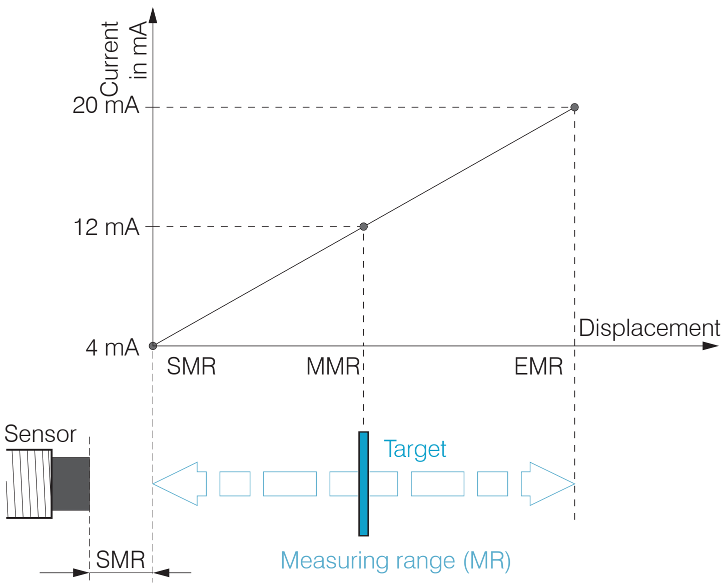
The measuring range of the sensor can also be read out from the config parameters.

Term definitions

- MR: measuring range
- SMR: Start of measuring range
- MMR: Mid of measuring range
- EMR: End of measuring range

Distances with SMR correspond to the distance between the front of the sensor and the target.

Distances without SMR do not take account of the so-called offset at the start of the measuring range.



MR in mm = parameter ID 358

SMR in mm = parameter-ID 390

Distance with SMR: SMR = SMR; MMR = SMR + MR/2; EMR = SMR + MR

Distance without SMR: SMR = 0; MMR = MR/2; EMR = MR

Calculating of digital distance values for a U3 sensor with a measuring range of 3 mm. (SMR = 0.3 mm, EMR = 3.3 mm).

10.4.2.1 int16 distance (Config Parameter 754 = 0x70, 0x73, 0x74)

	Distance in digits	Distance in mm	
		with SMR	without SMR
SMR	-29696	0.3	0
MR	0	1.8	1.5
EMR	29696	3.3	3

Conversion of the digital values

Distance with SMR: $$d = SMR + \frac{MMR}{2} * \left(1 + \frac{x}{29696}\right)$$

Distance without SMR: $$d = \frac{MR}{2} * \left(1 + \frac{x}{29696}\right)$$

10.4.2.2 int32 distance (Config parameter 754 = 0x71, 0x72)

	Distance in digits	Distance in mm	
		with SMR	without SMR
SMR	-475136	0.3	0
MR	0	1.8	1.5
EMR	475136	3.3	3

Conversion of the digital values

Distance with SMR: $$d = SMR + \frac{MR}{2} * \left(1 + \frac{x}{475136}\right)$$

Distance without SMR: $$d = \frac{MR}{2} * \left(1 + \frac{x}{475136}\right)$$

10.4.3 Calculation of the sensor and controller temperature

Conversion of the digital values

Temperature: $$T = \frac{(x_s - 12800)}{128}\ °C = \left(\frac{x_s}{128} - 100\right)\ °C$$

$$x_s = [0 \ldots 64512]$$

$$T = [-100\ °C \ldots + 404\ °C]$$

Temperature of the sensor

MR in °C = parameter ID 310 is 504 °C

SMR in °C = parameter ID 342 is -100 °C

EMR = SMR + MR = 404 °C

Temperature of the controller

MR in °C = parameter ID 311 is 504 °C

SMR in °C = parameter ID 343 is -100 °C

EMR = SMR + MR = 404 °C

10.5 State meaning

The values in State 1 indicate error states and the state of the limit value test. State 2 is not currently used and is always 0.

The value of the status byte State 1 should be interpreted as a bit field, i.e.:

- ► convert the displayed value into binary notation
- ► The status of the relevant status bit can be found in the table for interpreting *State 1*.

Table for interpreting State 1

Bit no.	Meaning 0	Meaning 1
0	Measurement signal in the measuring range	Measurement signal outside the measuring range
1	Sensor available	Broken sensor cable / no sensor available
2	Temperature of the sensor within the limits set at the factory	Temperature of the sensor outside the limits set at the factory

3	Temperature of the controller within the limits set at the factory	Temperature of the controller outside the limits set at the factory
4	Temperature of the sensor within the limits set at the factory	Temperature of the sensor outside the limits set at the factory
5	Temperature of the controller within the user-defined warning limits	Temperature of the controller outside the user-defined warning limits
6	Limit value OK (limit value is complied with)	Limit value NOT OK (limit value is not complied with)
7	Status of the switching output: OK	Status of the switching output: NOT OK (overload or excessive temperature)

Example for interpreting State 1

- Displayed value: State 1 = 5
- Value 5 in binary notation: 5 ≙ 00000101
- Bit no. 0 and bit no. 2 are set.
- From table: Measurement signal outside the measuring range and sensor temperature critical.

The status byte *State 2* always has the value 0 in the DT3020 and is therefore irrelevant.

11 Software support with MEDAQLib

The MEDAQLib software and driver library (SDK) connects Micro-Epsilon sensors quickly and easily to your software. These can be downloaded via the Micro-Epsilon website https://www.micro-epsilon.de/fileadmin/download/software/MEDAQLib.zip.

The MEDAQLib is delivered as a Dynamic Link Library (DLL). Thanks to their uniform C interface, integration and use in various programming languages is possible, e.g. C#, C++, VisualBasic, LabView, MATLAB and Python.

You can find software examples for common programing languages in the appendix , see Chap. 17.5

Supported ME bus sensor commands

Command	Supported	Comments
Logout	No	
Login	No	
Get_UserLevel	No	
Set_Password	No	
Set_Samplerate	No	The sample rate cannot be changed directly. A change is only possible via the configuration parameters, see Chap. 10.3 of the interface.
Get_Samplerate	Yes	
Set_Trigger	No	
Get_Trigger	No	
Set_Averaging	Yes	
Get_Averaging	Yes	Averaging type = (0;2), averaging value = (0; 4096)
Get_Measure	Yes	
Get_AlternateMeasure	Yes	
Set_ContinuousMode	No	
Get_ContinuousMode	Yes	
Set_Range	Yes	
Test_Baudrate	Yes	
Set_Baudrate	Yes	
Get_Baudrate	Yes	
Set_SensorAddress	Yes	
Get_SensorInfo	Yes	
Get_ChannelInfo	Yes	
Get_ChannelInfos	Yes	
Get_ControllerInfo	Yes	
Get_DiagnosticInfo	Yes	
Get_ConfigDescription	Yes	
Set_ConfigParameter	Yes	Detailed description, see Chap. 10
Get_ConfigParameter	Yes	Detailed description, see Chap. 10
Read_AllBlocks	Yes	

11.1 Reading and writing config parameters

Chapter 8 Config Parameters contains a list with configuration parameters. These configuration parameters can be read and written with MEDAQLib. `Get_ConfigParameter` is used for the reading and `Set_ConfigParameter` is used for the writing. To identify the parameter, the Config Parameter ID is used which must be set in MEDAQLib before writing or reading with the parameter `SP_ParameterID`. The MEDAQLib parameter `SP_ParameterName` is then not used and therefore does not need to be set.

The value of a config parameter read with the command `Get_ConfigParameter` is contained as text in the MEDAQLib parameter `SA_Value`.

To write a parameter with `Set_ConfigParameter`, it must first be written into the MEDAQLib parameter `SP_Value` as text.

11.2 Reading measurements

The measurement values read by the DT3020 controller with the MEDAQLib command `Get_Measure` can be changed via the output data format with the config parameter ID 754., see Chap. 10.4.1

It is also possible to read the other measurement values using the MEDAQLib command `Get_AlternateMeasure` without having to change the output format. The command for reading the alternative output blocks is longer. Therefore, the command `Get_Measure` should always be used for the maximum data rate. The command `Get_AlternateMeasure` can be used, for example, to read temperature values while distance values are transferred in blocks from the FIFO using the command `Get_Measure`.

The block number specified in the parameter `SP_AlternateBlockIdx` is also specified in the table , see Chap. 10.3.

11.3 Field linearization

Field linearization with MEDAQLib takes place via the config parameters. The procedure for this is described here, see Chap. 10.1.

12 Diagnostic messages

The error and warning messages are output in `sensorTOOL` under `Info` at `Diagnostic information`, or the status log can be called up via the flashing symbol in the bottom right corner. The IF2035 Profinet interface contains the index 0x2213 for outputting these messages, and the error codes can be read out from here as well.

12.1 Error codes

Code	Description
E16	ParamId 759: Number of samples for averaging is too large.
E17	ParamId 759: Number of samples for averaging is less than 1.
E18	ParamId 757: Invalid filter type written.
E19	ParamId 754: Invalid output data format specified.
E20	ParamId 50067: Length of the output buffer outside the valid range.
E21	ParamId 50063: Filter configuration to be set is not available.
E22	ParamId 2004, 2007, 2005, 2008: More than one characteristic curve management command was written at the same time.
E23	ParamId 2004: Characteristic curve data cannot be loaded.
E24	ParamId 2005: Characteristic curve data cannot be saved because there is not enough memory.
E25	ParamId 2005: Saving the characteristic curve is only possible if a characteristic curve was previously loaded with 2004.
E26	ParamId 2005: Saving the characteristic curve is not possible. The currently loaded characteristic curve is invalid.
E27	ParamId 2005: It is not possible to save the characteristic curve to the specified memory location as it is already occupied. The characteristic curve stored there must first be deleted.
E28	ParamId 2008: Specified characteristic curve cannot be reset.
E29	ParamId 2008: Specified characteristic curve is currently active and the field characteristic curve is write-protected.
E30	PramId 2008: Characteristic curve is write-protected.
E31	ParamId 50025: Value for write protection is invalid.
E32	ParamId 2007: Specified characteristic curve memory location outside the valid range.
E33	ParamId 2007: Specified characteristic curve does not exist or is incorrect.
E34	ParamId 2006: Specified characteristic curve memory location outside the valid range.
E35	ParamId 2006: Specified characteristic curve does not exist or is incorrect.
E36	ParamId 50039: The type of field linearization set with ParamId 50035 is invalid.
E37	ParamId 50035: Type of field linearization is invalid.
E38	ParamId 50068: Invalid value written. The status of the field linearization is not changed.
E39	ParamId 50039: Field linearization could not be saved because the write protection is set in 50025.
E40	ParamId 50068: Field linearization cannot be activated/deactivated because the write protection is set in 50025.
E41	ParamId 50068: Field linearization cannot be activated because there is no valid field linearization in the characteristic curve.
E42	ParamId 50070: Filter is not available.
E45	ParamId 50072: Decimation factor is outside the valid range.

12.2 Warnings

Code	Description/cause for user
W1	ParamId 2008: The field characteristic curve of the currently active characteristic curve was deleted.
W2	ParamId 2005: User write protection has been set in ParamId 50025. As a result, parts of the characteristic curve could not be saved.
W3	ParamId 50025: Invalid value written for the user write protection. Current status of the write protection was therefore not changed.
W4	ParamId 50025, 50034: Reading of characteristic curve information without first loading the information from a valid characteristic curve with ParamId 2004.
W5	Reading of characteristic curve information without first loading the information from a valid characteristic curve with ParamId 2004.
W13	ParamId 50037: No valid field calibration available or the currently valid field calibration contains fewer points.
W14	ParamId 50071: Reading of characteristic curve information without first loading the information from a valid characteristic curve with ParamId 2004.

13 Troubleshooting

Error	Reason and solution
Output signal in positive or negative saturation, depends on the scaling of the analog output.	• Target outside the measuring range • Cable and/or sensor not connected • Sensor is defective (e.g. interruption, short circuit) • Cable is defective
	Please observe the information in sensorTOOL. Replace the cable and/or sensor.
Output signal oscillates at a low frequency in multichannel operation.	• Mutual influence due to interferences
	Please observe the information on sensor arrangement with LF and HF bands; see the chapter “Measurement setup, operating multiple sensors”
No change to output signal.	• Check the supply voltage. • Check the assignment of sensor type and cable length. • Check the sensor and cable.

14 Disclaimer

All components of the device have been checked and tested for functionality in the factory. However, should any defects occur despite careful quality control, these shall be reported immediately to Micro-Epsilon or to your distributor / retailer.

Micro-Epsilon undertakes no liability whatsoever for damage, loss or costs caused by or related in any way to the product, in particular consequential damage, e.g., due to

- non-observance of these instructions/this manual,
- improper use or improper handling (in particular due to improper installation, commissioning, operation and maintenance) of the product,
- repairs or modifications by third parties,
- the use of force or other handling by unqualified persons.

This limitation of liability also applies to defects resulting from normal wear and tear (e.g., to wearing parts) and in the event of non-compliance with the specified maintenance intervals (if applicable).

Micro-Epsilon is exclusively responsible for repairs. It is not permitted to make unauthorized structural and / or technical modifications or alterations to the product. In the interest of further development, Micro-Epsilon reserves the right to modify the design or the firmware.

In addition, the General Terms of Business of Micro-Epsilon shall apply, which can be accessed under Legal details | Micro-Epsilon https://www.micro-epsilon.com/legal-details/.

15 Service, repair

If the measuring system is defective:

- If possible, save the current sensor settings in a parameter set to reload them into the controller after the repair.
- Please send us the affected parts for repair or exchange.

If the cause of a fault cannot be clearly identified, please send the entire system including sensor and all cables to:

MICRO-EPSILON MESSTECHNIK
GmbH & Co. KG
Koenigbacher Str. 15
94496 Ortenburg / Germany

Tel: +49 (0) 8542 / 168-0
Fax: +49 (0) 8542 / 168-90
info@micro-epsilon.com
www.micro-epsilon.com/contact/worldwide/
https://www.micro-epsilon.com

16 Decommissioning, disposal

In order to avoid the release of environmentally harmful substances and to ensure the reuse of valuable raw materials, we draw your attention to the following regulations and obligations:

- Remove all cables from the sensor and/or controller.
- Dispose of the sensor and/or the controller, its components and accessories, as well as the packaging materials in compliance with the applicable country-specific waste treatment and disposal regulations of the region of use.
- You are obliged to comply with all relevant national laws and regulations.

For Germany / the EU, the following (disposal) instructions apply in particular:

- Waste equipment marked with a crossed garbage can must not be disposed of with normal industrial waste (e.g. residual waste can or the yellow recycling bin) and must be disposed of separately. This avoids hazards to the environment due to incorrect disposal and ensures proper recycling of the old appliances.

- A list of national laws and contacts in the EU member states can be found at https://ec.europa.eu/environment/topics/waste-and-recycling/waste-electrical-and-electronic-equipment-weee_en. Here you can inform yourself about the respective national collection and return points.

- Old devices can also be returned for disposal to Micro-Epsilon at the address given in the legal details at https://www.micro-epsilon.com/legal-details.

- We would like to point out that you are responsible for deleting the measurement-specific and personal data on the old devices to be disposed of.

- Under the registration number WEEE-Reg.-Nr. DE28605721, we are registered at the foundation Elektro-Altgeräte Register, Nordostpark 72, 90411 Nuremberg, as a manufacturer of electrical and/or electronic equipment.

17 Appendix

17.1 Optional accessories

PS2020

Power supply unit for DIN rail mounting
Input 230 VAC, output 24 VDC/2.5 A

IF1032/ETH

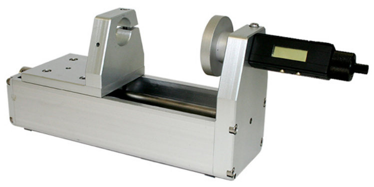
Interface module

- Ethernet/EtherCAT
- 1x RS485 (ME-internal protocol)
- 2x analog-in (14 bit, max. 4 ksps), voltage
- 1x analog-in, (14 bit, max. 4 ksps), current
- Inputs for supply voltage
- Trigger input
- EtherCAT synchronization output
- Output for sensor power supply

IF2035-EtherCAT
IF2035-PROFINET
IF2035-EtherNet/IP

Interface module for connection to EtherCAT, PROFINET or EtherNet/IP of a Micro-Epsilon sensor with RS485 or RS422 interface; DIN rail housing, incl. device description file for software integration in the PLC

MC25D



Digital micrometer for sensor calibration, adjustment range 0 - 25 mm, adjustable zero for all sensors

Vacuum feedthrough eddy/fB0/fB0/triax

Plug system for vacuum applications. Compatible with all common eddyNCDT products

- Application as a wall duct
- Pluggable version
- Mounting thread M9x0.5 / length 39 mm
- Max. wall thickness for mounting is 22 mm
- Maximum leakage rate <10^-8 mbar l/s

IF7001

- Single-channel USB/RS485 converter
- 2 x 0.14 mm^2, shielded, PVC, ø 4 mm ±0.2 mm
- Operating temperature: 0 ... 70 °C
- Protection class: IP 20

PCx/8-M12 PVC

- Power supply and signal cable
- M12 socket, straight, 8-pin, shielded
- Standard cable: 29011159 (3m)
- Optional cables: 29011141 (5m), 29011058 (10 m)
- Not for use with drag cable
- PVC cable: -25 °C ... +105 °C
- Socket: -25 °C ... +90 °C

PC10/8-M12 TPE	• Power supply and signal cable • M12 socket, straight, 8-pin, shielded • 29011285 (10m) • 29011059 (15m) • Drag chain-compatible TPE cable • •
PC5/8-M12/105 PUR	• Power supply and signal cable • M12 socket, straight, 8-pin, shielded • 29011506 (5m) • Drag chain-compatible PUR cable • Movably routed: -20 °C ... +105 °C • Immovably routed: -40 °C ... +105 °C • Drag chain: -20 °C ... +60 °C

17.2 Sensor model name

Eddy Sensor			Measuring range 1 / 2 / 3 / 4 / 6 / 8 mm				S = male connector C = integrated cable		Cable length [m]			A = mini B = normal C = large			Option
ES	-	S	3	-	C	-	S	A	2,0	/	m	B	0	/	▯
		S = shielded U = unshielded			C = cylindric F = flat T = Clamping flange			A = axial R = radial			m = male f = female OE = open ends		0 = straight 90 = right angle		

17.3 Sensor cable model name

								Controller side				
EC = eddy cable ECE = eddy cable extension								m = male f = female OE = open ends	A = mini B = normal C = large	0 = straight 90 = right angle		
EC	-	3,0	/	m	A	90	/	f	B	0	/	D3,9
		Nominal cable length [m]		m = male f = female OE = open ends	A = mini B = normal C = large	0 = straight 90 = right angle						D = nominal cable diameter [mm]
				Sensor side								

17.4 Stability against interference

Eddy current sensors from Micro-Epsilon are characterized by a high resistance to interference according to EN61000-4-6 (wired) and EN61000-4-3 (high-frequency electromagnetic fields). Typical sources of interference, such as grid applications, are far removed from the carrier frequency of the oscillator.

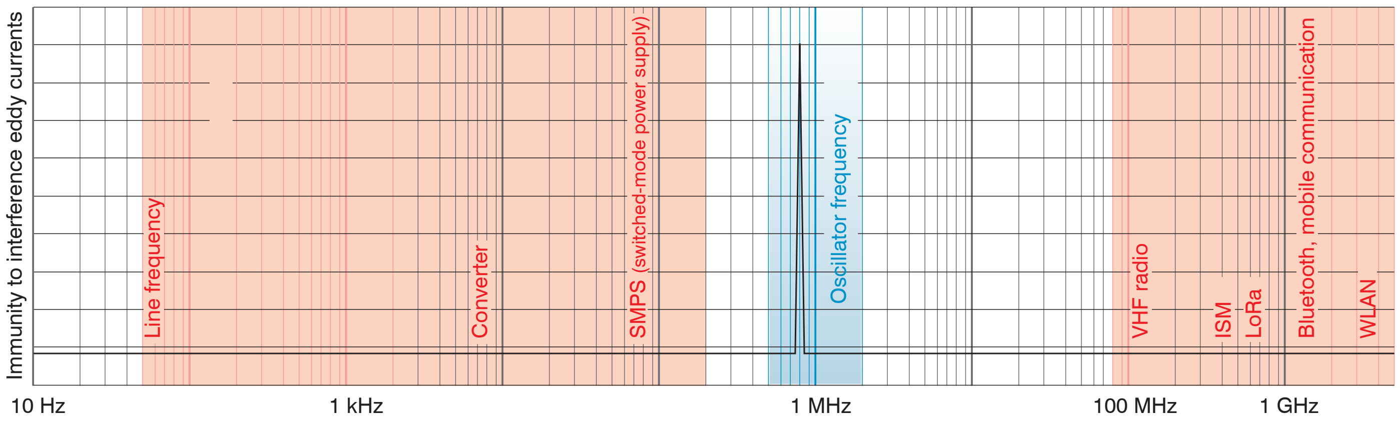


Fig. 17.1: Sensitivity of eddy current sensors to interference

17.5 Software examples for common programing languages

The following examples read the name, serial number of the DT3020 and the description of the measurement values. Then some measurement values are read from the DT3020.

Python

➡ Kopieren Sie die beiden Dateien `MEDAQLib.dll` und `MEDAQLib.py` aus dem Unterverzeichnis `Snippets/Python` im MEDAQLib-Installationsverzeichnis in das gleiche Verzeichnis wie den Python-Quellcode.

```
#
# This is a very simple sample following MEDAQLib.pdf section 4 Using MEDAQLib
#
# Please adjust to your setup (interface card and sensor used)
#

from MEDAQLib import MEDAQLib, ME_SENSOR, ERR_CODE
import time

number_of_reads = 10;

# Tell MEDAQLib about sensor type to be used
MEDAQLib_object = MEDAQLib.CreateSensorInstByName ("MEBus")



# Tell MEDAQLib about interface to be used
MEDAQLib_object.SetParameterString("IP_Interface", "RS232")
MEDAQLib_object.SetParameterString("IP_Port", "COM4")
MEDAQLib_object.SetParameterInt("IP_SensorAddress", 126)
MEDAQLib_object.SetParameterInt("IP_Baudrate", 230400)

# Enable Logfile writing
# MEDAQLib_object.SetParameterInt("IP_EnableLogging", 1)

# Try to open communication to sensor via interface specified
MEDAQLib_object.OpenSensor()
if MEDAQLib_object.GetLastError() != ERR_CODE.ERR_NOERROR:
    raise RuntimeError("OpenSensor: " + MEDAQLib_object.GetError())

MEDAQLib_object.ExecSCmd("Get_ControllerInfo")
if MEDAQLib_object.GetLastError() != ERR_CODE.ERR_NOERROR:
    raise RuntimeError("Get_ControllerInfo: " + MEDAQLib_object.GetError())

controller_name = MEDAQLib_object.GetParameterString("SA_ControllerName")
serial_number = MEDAQLib_object.GetParameterString("SA_SerialNumber")
print(f"Controller Name: {controller_name}")
print(f"Controller Serial Number: {serial_number}")

MEDAQLib_object.ExecSCmd("Get_TransmittedDataInfo")
if MEDAQLib_object.GetLastError() != ERR_CODE.ERR_NOERROR:
    raise RuntimeError("Get_TransmittedDataInfo: " + MEDAQLib_object.GetError())

number_of_channels = MEDAQLib_object.GetParameterInt("IA_ValuesPerFrame")
```

```
if number_of_channels == 0:
    raise RuntimeError("No data channels available")

for i in range(1, number_of_channels+1):
    index = MEDAQLib_object.GetParameterInt("IA_Index"+str(i))
    raw_name = MEDAQLib_object.GetParameterString("IA_Raw_Name"+str(i))
    scaled_name = MEDAQLib_object.GetParameterString("IA_Scaled_Name"+str(i))
    raw_unit = MEDAQLib_object.GetParameterString("IA_Raw_Unit"+str(i))
    scaled_unit = MEDAQLib_object.GetParameterString("IA_Scaled_Unit"+str(i))
    raw_range_min = MEDAQLib_object.GetParameterDouble("IA_Raw_RangeMin"+str(i))
    scaled_range_min = MEDAQLib_object.GetParameterDouble("IA_Scaled_RangeMin"+str(i))
    raw_range_max = MEDAQLib_object.GetParameterDouble("IA_Raw_RangeMax"+str(i))
    scaled_range_max = MEDAQLib_object.GetParameterDouble("IA_Scaled_RangeMax"+str(i))
    print(f"{index}: {raw_name} [{raw_range_min} .. {raw_range_max} {raw_unit}], " \
        f"{scaled_name} in {scaled_unit} [{scaled_range_min} .. {scaled_range_max}]")


print(f"Read {number_of_reads} measurements from {number_of_channels} channels ...")
# If no error then try to acquire data
if MEDAQLib_object.GetLastError() == ERR_CODE.ERR_NOERROR:
    for num_read in range(number_of_reads):
        # Sleep for 10 ms
        time.sleep(0.01)
        # Ask sensor for new data
        MEDAQLib_object.ExecSCmd("Get_Measure")
        # Check whether there is enough data to read in
        currently_available = MEDAQLib_object.DataAvail()
        # Check if DataAvail causes an Error
        if (MEDAQLib_object.GetLastError() != ERR_CODE.ERR_NOERROR):
            print(MEDAQLib_object.GetError())
        # If data is available?
        if currently_available >= number_of_channels:
            # Transfer/Move data from MEDAQLib internal buffer to own buffer
            transfered_data = MEDAQLib_object.TransferData(currently_available)
            # Check if TransferData causes an error
            if MEDAQLib_object.GetLastError() == ERR_CODE.ERR_NOERROR:
                # contains original values form sensor
                raw_data = transfered_data[0]
                # contains scaled data values
                scaled_data = transfered_data[1]
                # get number of data values received,
                # should be equal to currently_available
                nr_values_transfered = transfered_data[2]
                # output raw and scaled value of very first measurement
                for j in range(0,nr_values_transfered,number_of_channels):
                    print(scaled_data[j:j+number_of_channels], sep=', ')
                # do your computation on data ....
            else:
                # Print TransferData error
                print(MEDAQLib_object.GetError())
else:
    # Print OpenSensor Error
    print(MEDAQLib_object.GetError())

# Closing down by closing interface and releasing sensor instance
MEDAQLib_object.CloseSensor()
MEDAQLib_object.ReleaseSensorInstance()
```

C#

➡ Kopieren Sie die beiden Dateien `MEDAQLib.dll` und `MEDAQLib.Net.dll` aus dem Unterverzeichnis `Release` im MEDAQLib-Installationsverzeichnis in das gleiche Verzeichnis wie den C# Code.

```
using System;
using System.Diagnostics;
using MicroEpsilon; // MEDAQLib

namespace C_Sharp_Example
{
    class Program
    {
        static ERR_CODE Error(string location, ref MEDAQLib sensor)
        {
            string errText = "";
            ERR_CODE err = sensor.GetError(ref errText);
            Console.WriteLine(location + " returned error: " + errText);
            Console.WriteLine("Demo failed, press any key ...)");
            Console.ReadKey(true);
            return err;
        }

        static int sValsPerFrame = 0;

        static string StrWithIndex(string name, int index)
        {
            return name + index.ToString();
        }

        static ERR_CODE GetControllerInfo(ref MEDAQLib sensor)
        {
            string controllerName = "", controllerSerialNumber = "";

            if (sensor.ExecSCmd("Get_ControllerInfo") != ERR_CODE.ERR_NOERROR)
                return Error("Get_ControllerInfo", ref sensor);

            sensor.GetParameterString("SA_ControllerName", ref controllerName);
            sensor.GetParameterString("SA_SerialNumber", ref controllerSerialNumber);

            Console.WriteLine("Controller Name: {0}", controllerName);
            Console.WriteLine("Controller Serial Number: {0}", controllerSerialNumber);

            return ERR_CODE.ERR_NOERROR;
        }

        static ERR_CODE GetTransmittedDataInfo(ref MEDAQLib sensor)
        {
            int maxValsPerFrame = 0, maxOutputIndex = 0;

            if (sensor.ExecSCmdGetInt("Get_TransmittedDataInfo", "IA_ValuesPerFrame",
                    ref sValsPerFrame) != ERR_CODE.ERR_NOERROR)
                return Error("Get_TransmittedDataInfo", ref sensor);

            sensor.GetParameterInt("IA_MaxValuesPerFrame", ref maxValsPerFrame);
            sensor.GetParameterInt("IA_MaxOutputIndex", ref maxOutputIndex);
            Console.WriteLine("Sensor transmits {0} of {1} possible values," +
                "maximum output index is {2}",
                sValsPerFrame, maxValsPerFrame, maxOutputIndex);

            for (int i = 0; i < sValsPerFrame; i++)
            {
                int index = 0;
                double rawRangeMin = 0.0, rawRangeMax = 0.0;
                double scaledRangeMin = 0.0, scaledRangeMax = 0.0;
                string rawName = "", scaledName = "", rawUnit = "", scaledUnit = "";
                sensor.GetParameterString(
                  StrWithIndex("IA_Raw_Name", i + 1), ref rawName);
                sensor.GetParameterString(
                  StrWithIndex("IA_Scaled_Name", i + 1), ref scaledName);
                sensor.GetParameterString(
                  StrWithIndex("IA_Raw_Unit", i + 1), ref rawUnit);
                sensor.GetParameterString(
                  StrWithIndex("IA_Scaled_Unit", i + 1), ref scaledUnit);
                sensor.GetParameterInt(
                  StrWithIndex("IA_Index", i + 1), ref index);
                sensor.GetParameterDouble(
                  StrWithIndex("IA_Raw_RangeMin", i + 1), ref rawRangeMin);
                sensor.GetParameterDouble(
                  StrWithIndex("IA_Scaled_RangeMin", i + 1), ref scaledRangeMin);
                sensor.GetParameterDouble(
                  StrWithIndex("IA_Raw_RangeMax", i + 1), ref rawRangeMax);
                sensor.GetParameterDouble(
                  StrWithIndex("IA_Scaled_RangeMax", i + 1), ref scaledRangeMax);
                Console.WriteLine(
                    " {0,2}: {1} [{2} .. {3} {4}], {5} in {8} [" +
                    "{6} .. {7}" +
                    "]",
                    index, rawName, rawRangeMin, rawRangeMax, rawUnit, scaledName,
                    scaledRangeMin, scaledRangeMax, scaledUnit
                );
                Console.WriteLine(" {0,2}: {1} [{2} .. {3} {4}], {5} in {8} " +
                    "[{6} .. {7}]", index, rawName, rawRangeMin, rawRangeMax, rawUnit,
                    scaledName, scaledRangeMin, scaledRangeMax, scaledUnit);
            }

            return ERR_CODE.ERR_NOERROR;
        }
```

```
static ERR_CODE TransferData(ref MEDAQLib sensor)
{
    Console.WriteLine("Transfer data ...");

    while (!Console.KeyAvailable)
    {
        System.Threading.Thread.Sleep(10);
        sensor.ExecSCmd("Get_Measure");

        int avail = 0;
        if (sensor.DataAvail(ref avail) != ERR_CODE.ERR_NOERROR)
            return Error("DataAvail", ref sensor);

        int[] rawData = new int[avail];
        double[] scaledData = new double[avail];
        int read = 0;
        if (sensor.TransferData(rawData, scaledData, avail, ref read)
            != ERR_CODE.ERR_NOERROR)
                return Error("TransferData", ref sensor);

        int num_values = read/sValsPerFrame;
        for (int i = 0; i < num_values; i++)
        {
            Console.Write("{0:F3}", scaledData[i*sValsPerFrame]);
            for (int j = 1; j < sValsPerFrame; j++)
            {
              Console.Write(", {0:F3}", scaledData[i*sValsPerFrame+j]);
            }
            Console.WriteLine("");
        }
    }
    Console.ReadKey(true);
    Console.WriteLine("");

    return ERR_CODE.ERR_NOERROR;
}

static void Main(string[] args)
{
    Console.WriteLine("Start Demo...");

    MEDAQLib sensor = new MEDAQLib("ME-Bus");
    sensor.SetParameterString("IP_Interface", "RS232");
    sensor.SetParameterString("IP_Port", "COM4");
    sensor.SetParameterInt("IP_Baudrate", 230400);
    sensor.SetParameterInt("IP_SensorAddress", 126);
```

```
            // Enables logging of additional debugging information to TXT file
            //sensor.SetParameterInt("IP_EnableLogging", 1);

            if (sensor.OpenSensor() != ERR_CODE.ERR_NOERROR)
            {
                Error("OpenSensor", ref sensor);
                return;
            }

            if (GetControllerInfo(ref sensor) != ERR_CODE.ERR_NOERROR)
                return;

            if (GetTransmittedDataInfo(ref sensor) != ERR_CODE.ERR_NOERROR)
                return;

            if (sValsPerFrame == 0)
            {
                Console.WriteLine("No data channels available");
                Console.WriteLine("Demo failed, press any key ...)");
                Console.ReadKey(true);
                return;
            }

            if (TransferData(ref sensor) != ERR_CODE.ERR_NOERROR)
                return;

            Console.WriteLine("Demo successfully finished, press any key ...");
            Console.ReadKey(true);
        }
    }
}
```

MATLAB

➡ Kopieren Sie die beiden Dateien `MEDAQLib.h` und `Release-x64\MEDAQLib.dll` aus dem MEDAQLib-Installationsverzeichnis in das gleiche Verzeichnis wie das MATLAB-Skript.

```
%%DT3020_READ Example for reading one measurement value from DT3020
clear;
medaqlib_install_dir = 'C:\Program Files (x86)\MEDAQLib';
max_str_length = 32;
max_err_length = uint32(1024); % Reserved maximum length for error messages

number_of_reads = 10; % Number read requests to the sensor

%% Load MEDAQLib
if ~isfile('MEDAQLib.dll')
    copyfile(fullfile(medaqlib_install_dir, 'Release-x64', 'MEDAQLib.dll'), '.');
end
if ~isfile('MEDAQLib.h')
    copyfile(fullfile(medaqlib_install_dir, 'MEDAQLib.h'), '.');
end
if ~libisloaded('medaqlib')
    [notfound, warnings] = loadlibrary('MEDAQLib', 'MEDAQLib.h', 'alias', 'medaqlib');
end

try

    %% Tell MEDAQLib about sensor type to be used.
    h_sensor = uint32(calllib('medaqlib', 'CreateSensorInstByName', 'MEbus'));

    %% Tell MEDAQLib about interface to be used
    calllib('medaqlib', 'SetParameterString', h_sensor, 'IP_Interface', 'RS232');
    calllib('medaqlib', 'SetParameterString', h_sensor, 'IP_Port', 'COM4');
    calllib('medaqlib', 'SetParameterInt', h_sensor, 'IP_SensorAddress', 126);
    calllib('medaqlib', 'SetParameterInt', h_sensor, 'IP_Baudrate', 230400);

    %% Enable Logfile writing
    calllib('medaqlib', 'SetParameterInt', h_sensor, 'IP_EnableLogging', 1);

    %% Try to open communication to sensor via interface specified
    err = calllib('medaqlib', 'OpenSensor', h_sensor);
    if ~strcmp(err, 'ERR_NOERROR')
        error('Unable to open Sensor %s', err);
    end
catch ME
    unloadlibrary('medaqlib');
    rethrow(ME);
end

try

    %% Read information about connected controller
```

```
err = calllib('medaqlib', 'ExecSCmd', h_sensor, 'Get_ControllerInfo');
assert(strcmp(err, 'ERR_NOERROR'), 'medaqlib:ExecSCmd', 'Get_ControllerInfo');

% Display controller name
[~, param_name, controller_name, ~] = calllib('medaqlib', ...
    'GetParameterString', h_sensor, 'SA_ControllerName', ...
    blanks(max_str_length), libpointer('uint32Ptr', max_str_length));
fprintf('%s = %s\n', param_name, controller_name);

% Display controller serial number
[~, param_name, controller_serial_number, ~] = calllib('medaqlib', ...
    'GetParameterString', h_sensor, 'SA_SerialNumber', ...
    blanks(max_str_length), libpointer('uint32Ptr', max_str_length));
fprintf('%s = %s\n', param_name, controller_serial_number);

%% Read information about the transmitted data
err = calllib('medaqlib', 'ExecSCmd', h_sensor, 'Get_TransmittedDataInfo');
assert(strcmp(err, 'ERR_NOERROR'), 'medaqlib:ExecSCmd', 'Get_TransmittedDataInfo');
[~, ~, channel_count] = calllib('medaqlib', 'GetParameterInt', h_sensor, ...
    'IA_ValuesPerFrame', libpointer('int32Ptr', 0));

disp("Read "+string(number_of_reads)+" measurements from "+ ...
    channel_count+" channels ...");
if channel_count == 0
    error('No data channels available');
end

channel_names = cell(1, channel_count);
for k = 1:channel_count
    [~, ~, index] = calllib('medaqlib', 'GetParameterInt', h_sensor, ...
        sprintf('IA_Index%d', k), libpointer('int32Ptr', 0));
    [~, ~, scaled_name, ~] = calllib('medaqlib', 'GetParameterString', ...
        h_sensor, sprintf('IA_Scaled_Name%d', k), blanks(max_str_length), ...
        libpointer('uint32Ptr', max_str_length));
    [~, ~, scaled_unit, len] = calllib('medaqlib', 'GetParameterBinary', ...
        h_sensor, sprintf('IA_Scaled_Unit%d', k), ...
        zeros(1, max_str_length, 'uint8'), ...
        libpointer('uint32Ptr', max_str_length));
    if len > 0
        scaled_unit = char(scaled_unit(1:len));
    else
        scaled_unit = '';
    end
    [~, ~, scaled_range_min] = calllib('medaqlib', 'GetParameterDouble', ...
        h_sensor, sprintf('IA_Scaled_RangeMin%d', k), libpointer('doublePtr', 0));
    [~, ~, scaled_range_max] = calllib('medaqlib', 'GetParameterDouble', ...
        h_sensor, sprintf('IA_Scaled_RangeMax%d', k), libpointer('doublePtr', 0));
    disp(string(index)+": "+scaled_name+" ["+string(scaled_range_min) ...
            +" .. "+string(scaled_range_max)+"]");
        channel_names{k} = strip(strrep(scaled_name, '(scaled)', ''), 'both');
    end
    clear str;
    disp(strjoin(channel_names, ', '));

    %% Read measurement value from sensor
    for i = 1:number_of_reads
        % Ask sensor for new data
        err = calllib('medaqlib', 'ExecSCmd', h_sensor, 'Get_Measure');
        assert(strcmp(err, 'ERR_NOERROR'), 'medaqlib:ExecSCmd', 'Get_Measure');

        % Check whether there is enough data to read in
        [~, currently_available] = calllib('medaqlib', 'DataAvail', h_sensor, ...
            libpointer('int32Ptr', 1));

        % If data is available?
        if currently_available >= channel_count
            % Transfer/Move data from MEDAQLib‘s internal buffer to own buffer
            raw_data = libpointer('int32Ptr', zeros(1, currently_available));
            scaled_data = libpointer('doublePtr', zeros(1, currently_available));

            [err, raw_data, scaled_data, num_read] = calllib('medaqlib', ...
                'TransferData', h_sensor, raw_data, scaled_data, ...
                currently_available, libpointer('int32Ptr', 1));
            assert(strcmp(err, 'ERR_NOERROR'), 'medaqlib:TransferData', '');

            data_read = reshape(scaled_data, channel_count, []).';
            for k = 1:size(data_read, 1)
                disp(strjoin(string(data_read(k, :)), ', '));
            end
        else
            error('No data available');
        end
    end
    calllib('medaqlib', 'CloseSensor', h_sensor);
    calllib('medaqlib', 'ReleaseSensorInstance', h_sensor);
    unloadlibrary('medaqlib');
catch ME
    s = split(ME.identifier, ':');
    if strcmp(s{1}, 'medaqlib')
        [~, error_txt] = calllib('medaqlib', 'GetError', h_sensor, ...
            blanks(max_err_length), max_err_length);
        if isempty(error_txt)
            error('%s: %s', ME.identifier, ME.message);
        else
            error('%s: %s MEDAQLib "%s"', ME.identifier, ME.message, error_txt);
        end
    end
    calllib('medaqlib', 'CloseSensor', h_sensor);
    calllib('medaqlib', 'ReleaseSensorInstance', h_sensor);
    unloadlibrary('medaqlib');
    rethrow(ME);
end
```