Your Global Automation Partner

TURCK

# TN-UHF-...-LNX
# UHF Reader

Instructions for Use

# Contents

# 1 About these Instructions

These instructions for use describe the structure, functions and the use of the product and will help you to operate the product as intended. Read these instructions carefully before using the product. This is to avoid possible damage to persons, property or the device. Retain the instructions for future use during the service life of the product. If the product is passed on, pass on these instructions as well.

## 1.1 Target groups

These instructions are aimed at qualified personal and must be carefully read by anyone mounting, commissioning, operating, maintaining, dismantling or disposing of the device.

## 1.2 Explanation of symbols used

The following symbols are used in these instructions:

| | |
|---|---|
|  | **DANGER**<br>DANGER indicates a dangerous situation with high risk of death or severe injury if not avoided. |
|  | **WARNING**<br>WARNING indicates a dangerous situation with medium risk of death or severe injury if not avoided. |
|  | **CAUTION**<br>CAUTION indicates a dangerous situation of medium risk which may result in minor or moderate injury if not avoided. |
|  | **NOTICE**<br>NOTICE indicates a situation which may lead to property damage if not avoided. |
|  | **NOTE**<br>NOTE indicates tips, recommendations and useful information on specific actions and facts. The notes simplify your work and help you to avoid additional work. |
| ▶ | **CALL TO ACTION**<br>This symbol denotes actions that the user must carry out. |
| ⇨ | **RESULTS OF ACTION**<br>This symbol denotes relevant results of actions. |

## 1.3 Other documents

Besides this document the following material can be found on the Internet at www.turck.com:

- Data sheet
- Approvals
- Configuration manual

## 1.4 Naming convention

Read/write devices are called “read/write heads” for the HF range and “readers” for the UHF range. Common synonyms for “data carriers” are “tags”, “transponders” and “mobile data memory”.

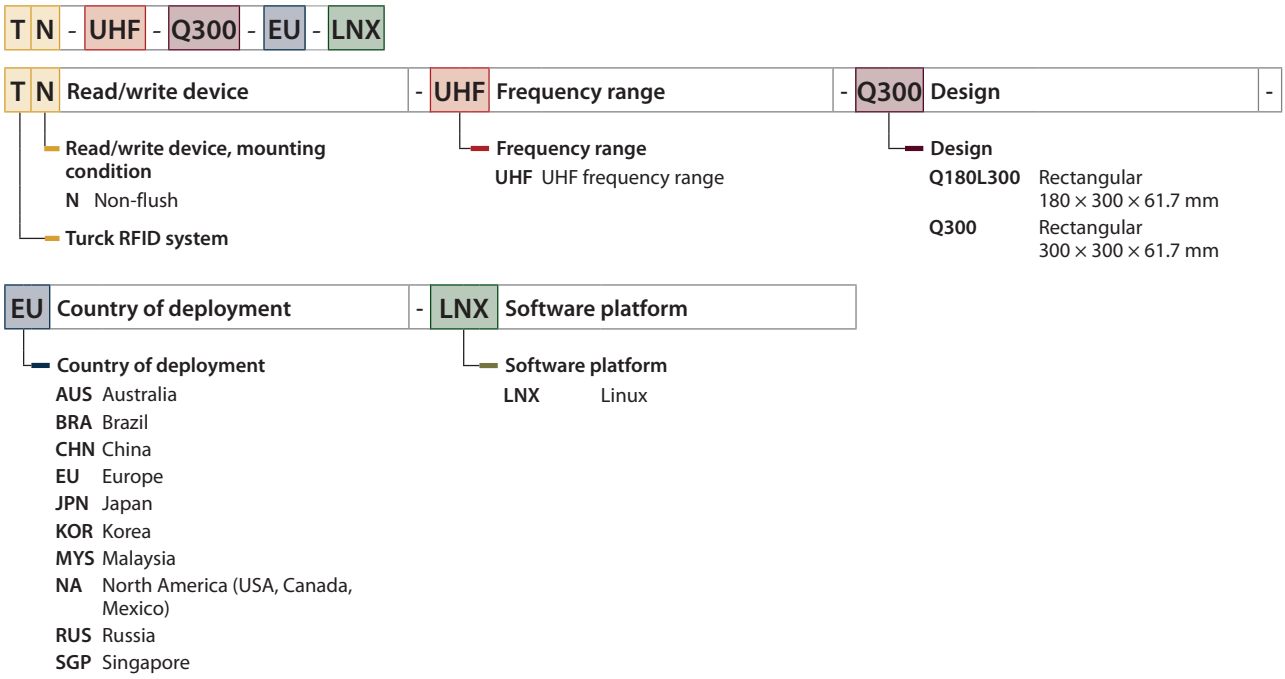
## 1.5 Feedback about these instructions

We make every effort to ensure that these instructions are as informative and as clear as possible. If you have any suggestions for improving the design or if some information is missing in the document, please send your suggestions to techdoc@turck.com.

# 2 Notes on the Product

## 2.1 Product identification

These instructions apply to the following UHF readers:




## 2.2 Scope of delivery

The scope of delivery includes:

- UHF reader
- Wall bracket (metal rail)
- Quick Start Guide

## 2.3 Legal requirements

The devices are subject to the following EU directives:

- 2014/30/EU (electromagnetic compatibility)
- 2011/65/EU (RoHS Directive)
- 2014/53/EU (RED Directive)

## 2.4 Turck service

Turck supports you with your projects, from initial analysis to the commissioning of your application. The Turck product database under www.turck.com contains software tools for programming, configuration or commissioning, data sheets and CAD files in numerous export formats.

The contact details of Turck subsidiaries worldwide can be found on p. [▷ 104].

# 3 For Your Safety

The product is designed according to state-of-the-art technology. However, residual risks still exist. Observe the following warnings and safety notices to prevent damage to persons and property. Turck accepts no liability for damage caused by failure to observe these warning and safety notices.

## 3.1 Intended use

These devices are designed solely for use in industrial areas.

The readers with an integrated RFID interface are used for contactless data exchange with the BL ident tags in the Turck UHF RFID system. The following table shows the operating frequency of the devices:

| Type code | Operating frequency | Application range |
|---|---|---|
| TN-UHF-…-AUS-LNX | 920…926 MHz | Australia, New Zealand |
| TN-UHF-…-BRA-LNX | 915…928 MHz | Brazil |
| TN-UHF-…-CHN-LNX | 920.5…924.5 MHz | China |
| TN-UHF-…-EU-LNX | 865…868 MHz | Europe, Turkey, India |
| TN-UHF-…-JPN-LNX | 916.7…920.9 MHz | Japan |
| TN-UHF-…-KOR-LNX | 917…920.8 MHz | Korea |
| TN-UHF-…-MYS-LNX | 919…923 MHz | Malaysia |
| TN-UHF-…-NA-LNX | 902…928 MHz | North America (USA, Canada, Mexico) |
| TN-UHF-…-RUS-LNX | 866…868 MHz | Russia |
| TN-UHF-…-SGP-LNX | 920…925 MHz | Singapore |

These devices may only be started up under the following conditions:

- The particular frequency range is permissible for the use of UHF-RFID.
- The operating frequency range of the devices is compliant with the permissible UHF RFID range of the region.
- A valid certification and/or approval is available for the region of use.

The integrated RFID interface enables the readers to communicate directly via TCP/IP with higher-level systems such as ERP systems. Read data is sent to the higher-level system via the device.

Four configurable digital channels are also provided for connecting digital sensors and actuators.

The devices may only be used as described in these instructions. Any other use is not in accordance with the intended use. Turck accepts no liability for any resulting damage.

## 3.2 General safety instructions

- The device only meets the EMC requirements for industrial areas and is not suitable for use in residential areas.
- The device may only be assembled, installed, operated, parameterized and maintained by professionally-trained personnel.
- The device may only be used in accordance with applicable national and international regulations, standards and laws.
- Any extended stay within the area of radiation of UHF readers may be harmful to health. Observe a minimum distance of > 0.35 m from the actively radiating surface of the UHF reader.
- The radiation of the UHF readers may have an adverse effect on the operation of electrically controlled medical equipment. Keep an additional distance from active radiation sources up to the maximum transmission distance.
- Change the default password of the integrated web server after the first login. Turck recommends using a secure password.

# 4 Product Description

The devices are designed with an aluminum housing and degree of protection IP67. The active face is made out of plastic. Devices are available with an integrated antenna (Q300) or for connecting external antennas (Q180). Both device variants are suitable for connecting up to four external passive UHF RFID antennas.

The terminals for the Ethernet and for digital I/Os are M12 sockets. The device has an M12 plug connector for connecting the power supply. Terminals are provided for up to four external antennas.

## 4.1 Device overview

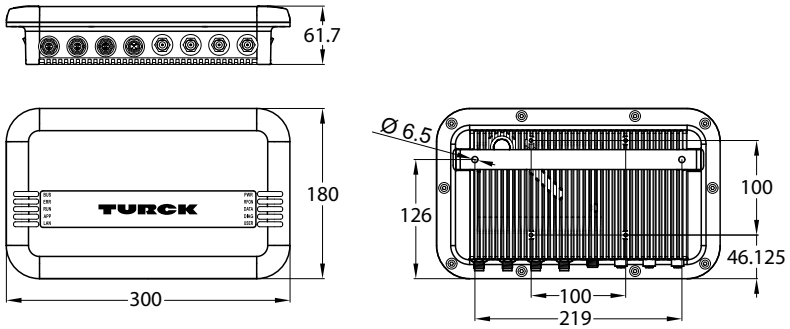



Fig. 1: Dimensions – TN-UHF-Q180L300…

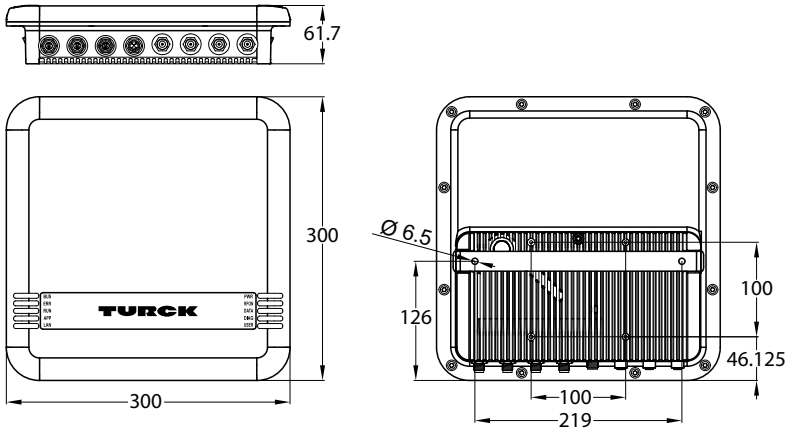



Fig. 2: Dimensions – TN-UHF-Q300…

### 4.1.1 Indication elements

The device has the following LED indicators:

- Power supply
- Group and bus errors
- Status
- Diagnostics

An audible alarm can also be set using software tools.

## 4.2 Properties and features

- TCP/IP
- Freely programmable Ethernet-based reader based on Linux
- Programming languages C, C++, NodeJS, Python
- Software components: SSH, SFTP, HTTP, IBTP, MTXP, DHCP, SNTP, Node.js 6.9.5 (LTS), Python 3.x
- Implementation of the protocol required
- 2 W (ERP) maximum output power
- 4 RP-TNC terminals for passive external UHF RFID antennas
- 4 configurable digital channels as 2 A PNP inputs and/or outputs
- 10 Mbps/100 Mbps transfer rate
- Integrated web server
- LEDs and diagnostics

## 4.3 Operating principle

The readers are used for contactless data exchange with tags. For this the controller sends commands and data via the interface to the reader and receives the corresponding response data from the reader. The reading of the IDs of all RFID tags in the read area or the writing of an RFID tag with a specific production date are examples of typical commands. To communicate with the tag, the data of the reader is coded and transferred via an electromagnetic field, which at the same time supplies the tags with power.

A reader contains a transmitter and a receiver, an interface to the interface module and a coupling element (coil and dipole antenna) for communicating with the tag. Electromagnetic wave propagation is used for the transmission between reader and tag on devices for the UHF range.

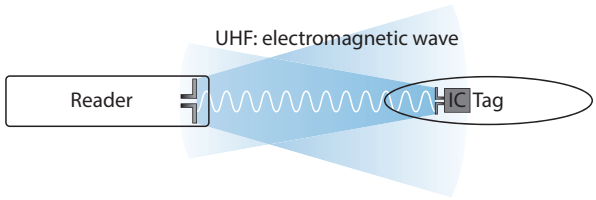



Fig. 3: Operating principle of UHF-RFID

The antenna of the reader generates electromagnetic waves. This produces a transmission window as a so-called air interface in which the data exchange with the tag takes place. The size of the transmission window depends on the combination of readers and tags, as well as on the relevant environmental conditions.

Each reader can communicate with a number of tags. This requires the reader and the tag to operate in the same frequency range. Depending on the power and frequency used, the device ranges vary from a few millimeters up to several meters. The specified maximum read/write distances only represent typical values under laboratory conditions without allowing for the effect of materials. The achievable distances may vary due to component tolerances, the mounting situation in the application, ambient conditions and the effect of materials (particularly metal and liquids).

## 4.4 Functions and operating modes

The devices operate with an integrated or external antenna (TN-UHF-Q300…) or only with an external antenna (TN-UHF-Q180L300…). The devices enable passive UHF tags to be read or written in single and multitag operation. For this the devices form a transmission zone that varies in size and range according to the tags used and the operating conditions of the application. Refer to the data sheets for the applicable maximum read/write distances. The devices can be fully tested, configured and parameterized from a PC using the specified software tools.

The Linux operating system enables the device functions to be programmed with C, C++, NodeJS or Python. It is also possible to integrate middleware functions on the device.

Sensors and actuators can be connected to the configurable digital channels. Up to four 3-wire PNP sensors or two PNP DC actuators with a maximum output current of 2 A per output can be connected. An external power supply is required in order to use the digital channels as outputs.

## 4.5 Technical accessories

Accessories for mounting, connecting and parameterizing can be found in product database under **www.turck.com**. The accessories are not part of the scope of delivery.

# 5 Installing

The device is provided with a bracket in accordance with VESA 100 × 100 for mounting. The device is provided with four M4 threaded holes spaced 100 mm apart (horizontally and vertically). The maximum length of the screws is 8 mm plus the thickness of the VESA bracket. The devices can be mounted in any position.

- ▶ Fasten the device with four M4 screws to a bracket in accordance with VESA 100 × 100.

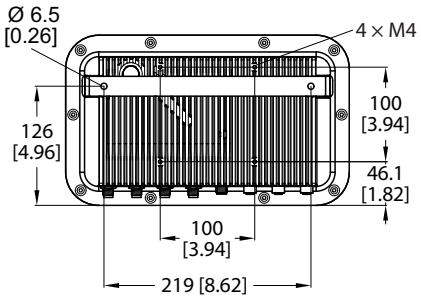



Fig. 4: Rear view – TN-UHF-Q180…

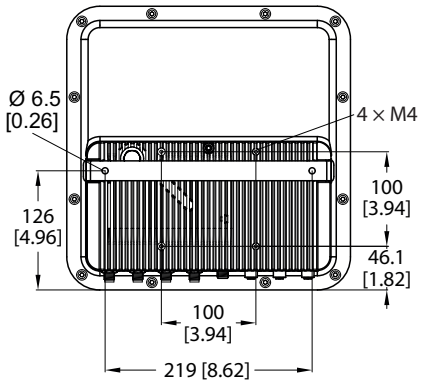



Fig. 5: Rear view – TN-UHF-Q300…

# 6 Connecting

## 6.1 Connecting devices to Ethernet

The device is provided with a 4-pin M12 female connector for connecting the device to an Ethernet system.

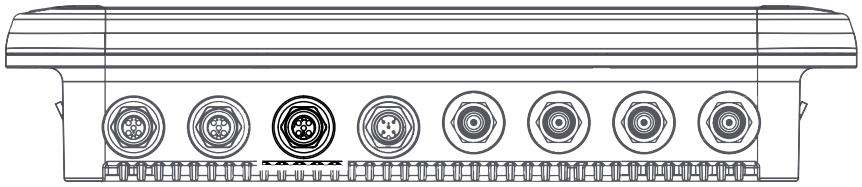


Fig. 6: M12 Ethernet connector

- ▶ Connect the device to Ethernet as per the following pin assignment (max. tightening torque 0.8 Nm).

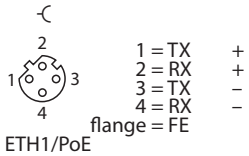



Fig. 7: Pin assignment of the Ethernet connections

**NOTE**
With PoE transfer the power supply via PoE Mode A with 4-wire cables.

## 6.2 Connecting the power supply

The device is provided with a 5-pin M12 plug connectors for connecting the power supply.

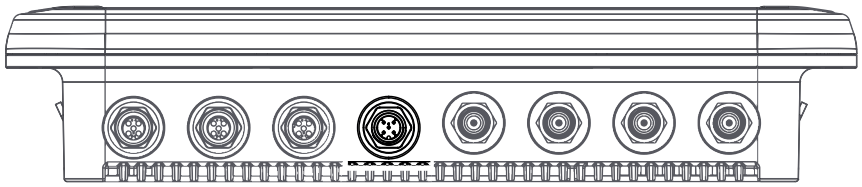


Fig. 8: M12 plug connector for connecting the power supply

- ▶ Connect the device to the power supply as per the following pin assignment (max. tightening torque 0.8 Nm).

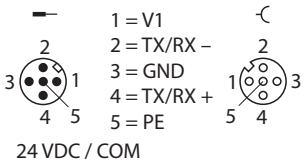



Fig. 9: Pin assignment of the power supply terminals

## 6.3 Connecting digital sensors and actuators

The device has two 5-pin M12 plug connectors for connecting digital sensors and actuators.

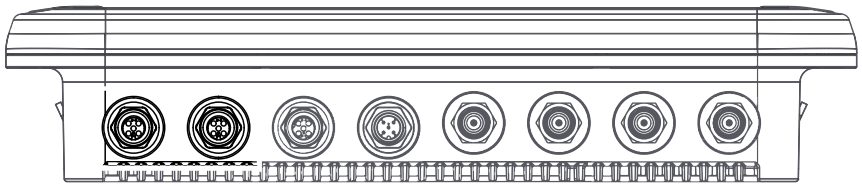


Fig. 10: M12 plug connectors for connecting digital sensors and actuators

**NOTE**
When operating via PoE (Power over Ethernet) the digital channels cannot be used as outputs.

- ▶ Connect sensors and actuators to the device as per the following pin assignment (max. tightening torque 0.8 Nm).

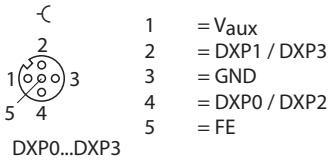



Fig. 11: Connections for digital sensors and actuators – pin assignment

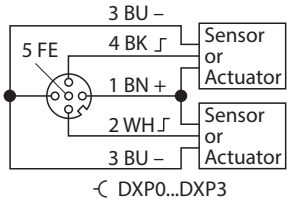



Fig. 12: Connections for digital sensors and actuators – wiring diagram

## 6.4 Connecting external antennas

The device is provided with four RP-TNC sockets for connecting up to four external antennas. The input impedance is 50 Ω.

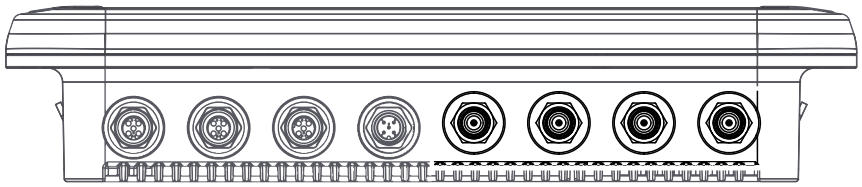


Fig. 13: RP-TNC sockets for connecting external antennas

- ▶ Connect external antennas with an RP-TNC antenna cable to the device (max. tightening torque 0.8 Nm).

# 7 Commissioning

The Linux operating system enables the device functions to be programmed with C, C++, NodeJS or Python.

Additional software tools (e.g. PuTTY) are required to access the device via the console. A file exchange between the device and a PC can be used, e.g. via WinSCP. The following login data is stored on the device by default:

User: user

Password: password

**NOTE**
The reader log is not implemented by default. The protocol must be implemented by the user.

## 7.1 Parameterizing readers with the DTM

The UHF settings of the device can be assigned additional parameters via the DTM.

All the required Turck software components can be downloaded via the Turck Software Manager. The Turck Software Manager is available free of charge from www.turck.com.

**NOTE**
The parameterization function up to firmware version V2.0.39.3937 is only available in English. All parameters are written in the DTM.

The individual readers are available in different variants. When a connection is made to a connected reader, the DTM automatically detects the relevant device and deactivates menu items that are not supported. The connection cannot be established if a different variant than set in the project tree is connected.

**NOTE**
Adjustable parameters are indicated in the DTM with green arrows. Fixed parameters are indicated with gray arrows.

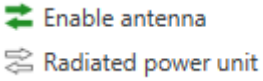



Fig. 14: DTM – example of adjustable and fixed parameters

### Requirements for extended parameter setting

- PACTware must be installed.
- The DTM for UHF readers must be installed.
- The device must be connected to a PC via the serial interface.

### 7.1.1 Connecting the device with the PC

- Open PACTware.
- Right-click **Host PC** in the project tree.
- Click **Add device**.
- Add **RS485 RFID**.
- Confirm selection with **OK**.

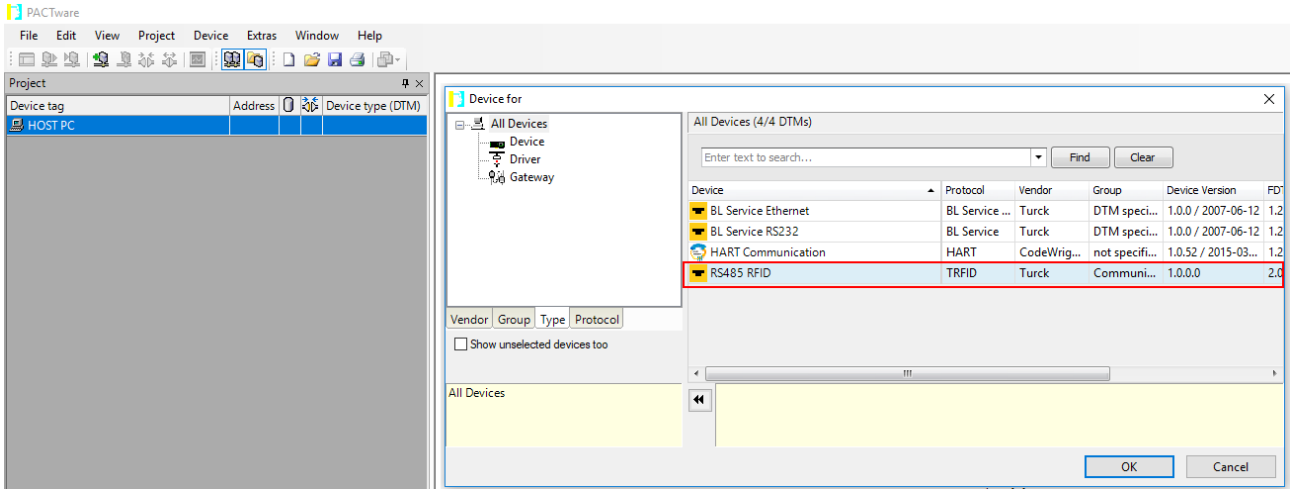



Fig. 15: Selecting RS485 RFID

- Right-click the Ethernet adapter.
- Start the **Topology scan**.

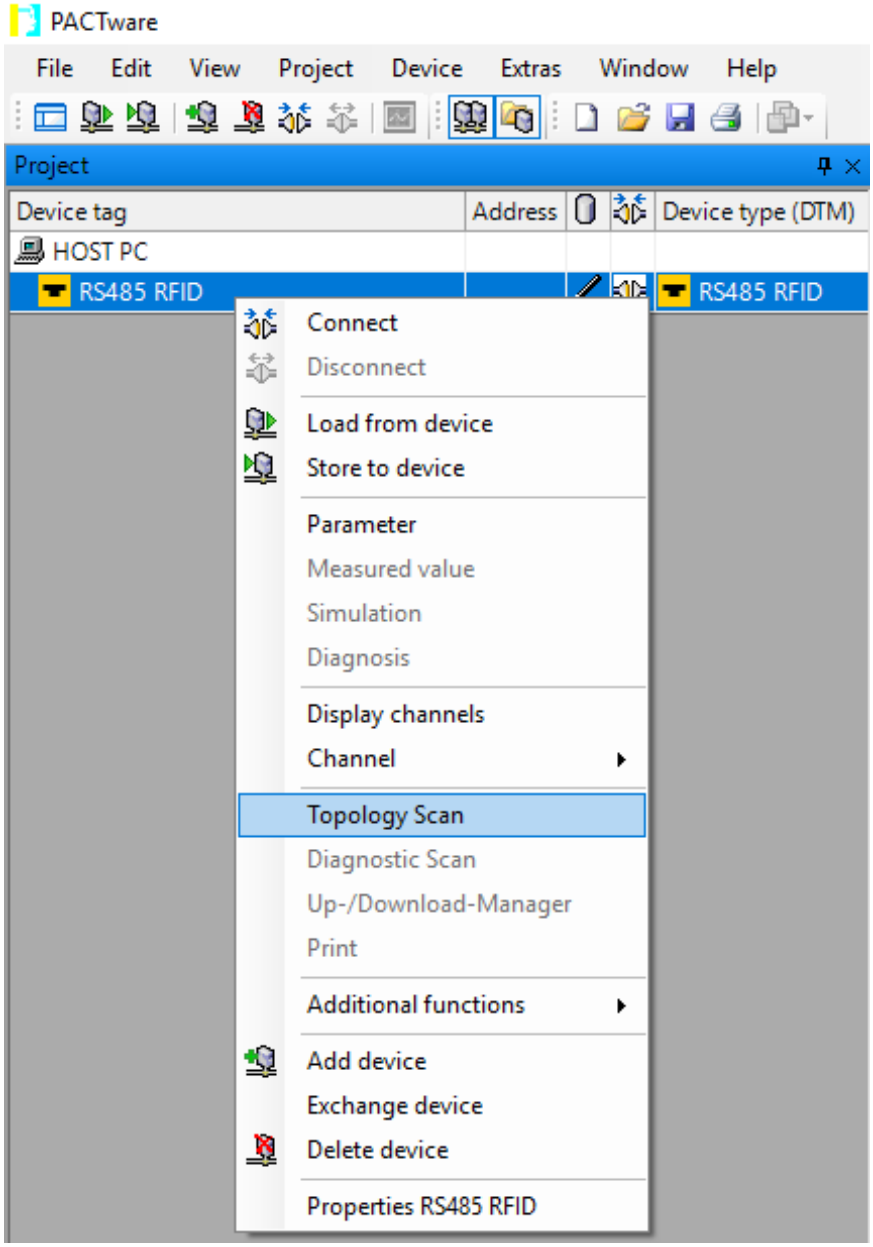



Fig. 16: Starting the Topology scan

The connected devices are automatically detected and added to the project tree.

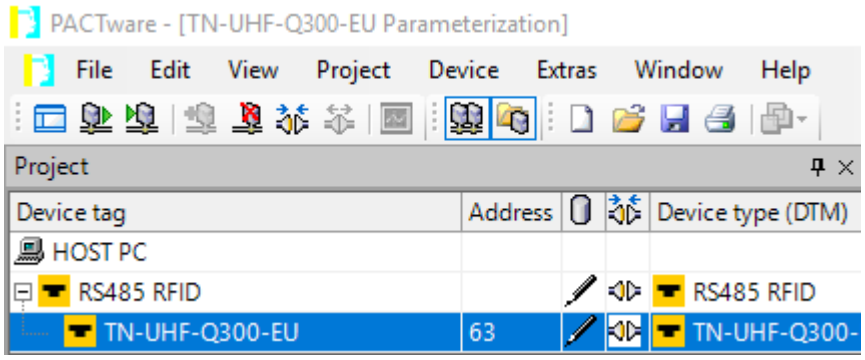



Fig. 17: Project tree


Hans Turck GmbH & Co. KG | T +49 208 4952-0 | F +49 208 4952-264 | more@turck.com | www.turck.com

### 7.1.2 Starting extended reader parameterization

- Right-click the device.
- Start the parameter setting: Choose **Parameterization** or **Online Parameterization**. The device must be connected to the PC for the **Online Parameterization**.

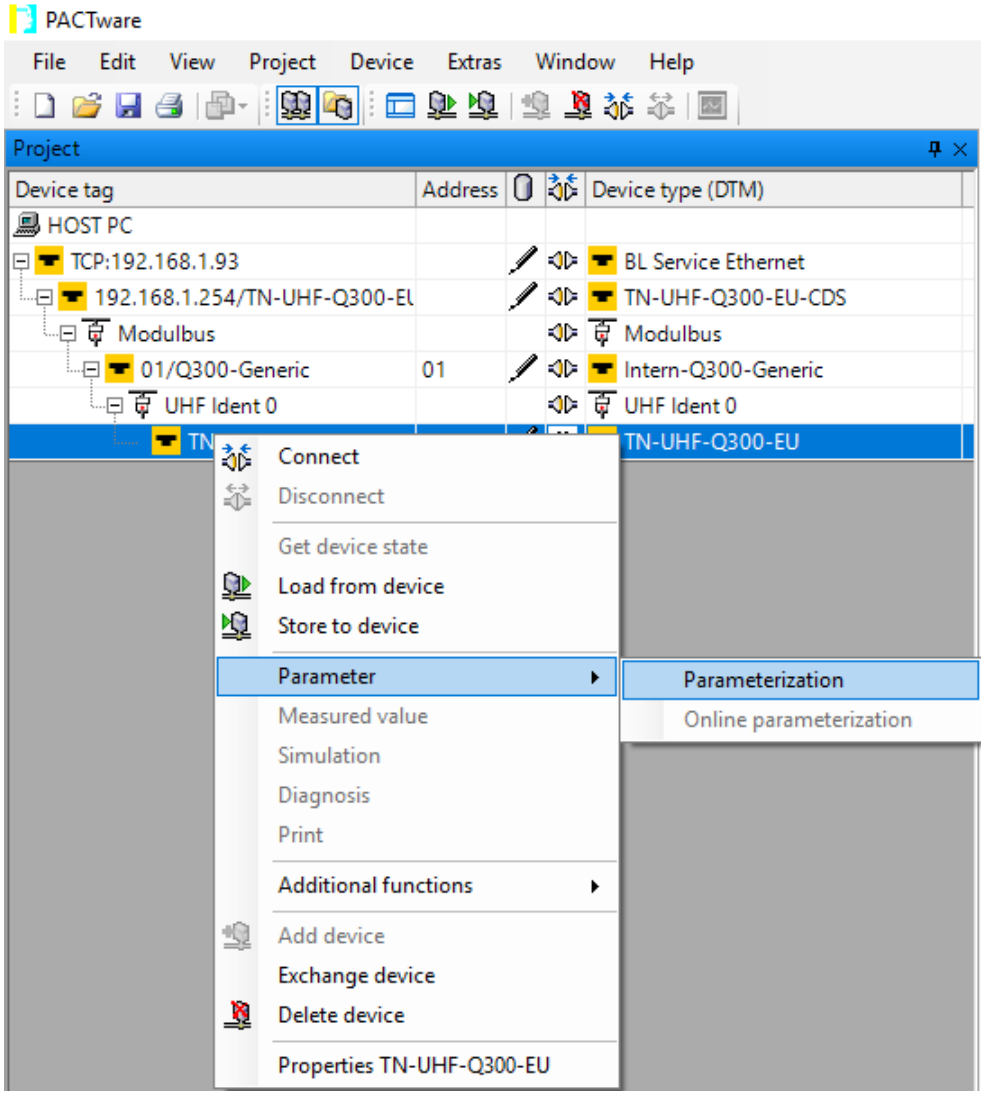



Fig. 18: Starting the parameterization

### 7.1.3 DTM main menu – overview

The main menu provides the following functions:

| Icon | Function | Description |
|---|---|---|
|  | Switch display | Shows the information bar for the connected device and DTM version at the top of the screen. |
|  | Help about DTM | Starts the DTM help. |
|  | Help about device | Opens the data sheet of the connected reader. |
|  | Activating and deactivating Expert mode | Opens the drop-down menu to select the access level. The following access levels are available:<br>■ Basic (default setting)<br>■ Advanced<br>■ Administrator (password-protected) |
|  | Channel-wise display | Toggles the view between standard display and channel-wise display. |
|  | Load data from database | Loads previously stored parameters from the database (e.g. an existing project). |
|  | Store data in database | Transfers the current reader parameters to the database of the current project. |
|  | Read data from device | Reads the set parameters from the device. |
|  | Transmit data to device | Transfers the set parameters to the device. |
|  | Compare displayed values with database | Compares the values displayed in the DTM with the values saved in the database. |
|  | CSV export current values | Exports the current values from the DTM to a CSV file. |

The following setup windows can be opened in tabs via the main menu:

- Basic setup
- Antenna
- Antenna configuration
- Communication
- EPC Class1 Gen2
- Post read filter
- Signaling

### 7.1.4 Choosing the access level

Three access levels are available for setting the device parameters. Different parameters can be set depending on the access level.

**NOTE**
Modifications made in the **Administrator** access level can result in serious changes to operation. The **Administrator** access level is therefore only available for Turck service technicians. All relevant settings for the successful parameter setting of an application are available in the **Advanced** access level.

| Access level | Description | Initial password |
|---|---|---|
| Basic | Basic access for configuration and commissioning | Not required |
| Advanced | Extended access, e.g. for applications | Not required |
| Administrator | Administrator access for critical security or wireless settings | Required |

The current access level is displayed in the top right screen area of the DTM.


Fig. 19: Display of the access level

### 7.1.5 Setting multiplex operation

In multiplex operation, several antennas can be controlled or switched on in sequence. The example below shows the activation of the antennas in sequence. The multiplex operation can consist of up to 16 sequences and can be used, for example, for gate applications.

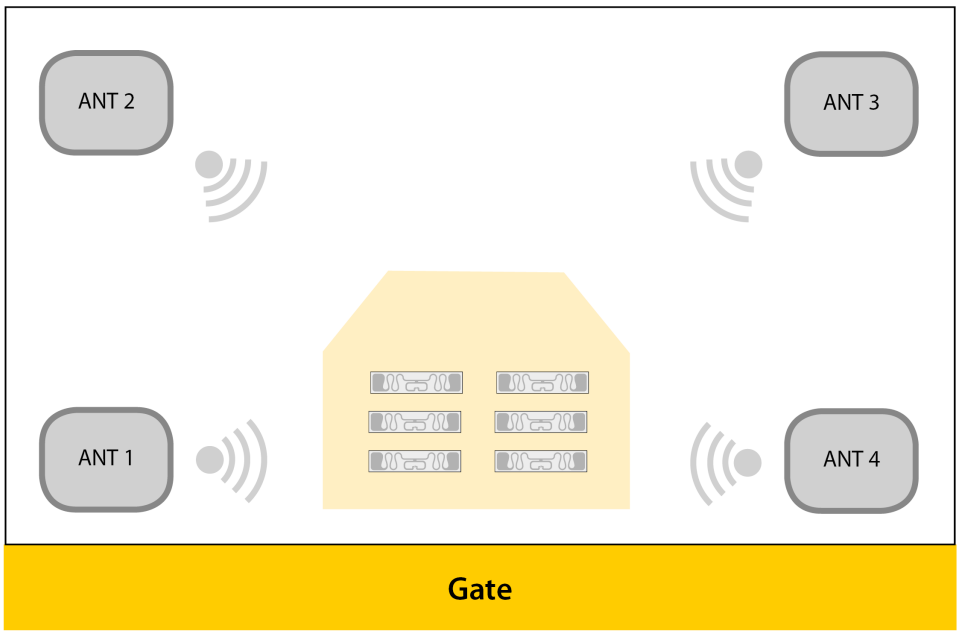



Fig. 20: Gate application – schematic representation


Hans Turck GmbH & Co. KG | T +49 208 4952-0 | F +49 208 4952-264 | more@turck.com | www.turck.com

## Configuring multiplex operation – example

- Choose the **Antenna** tab in the main menu.
- At **Antenna** → **Antenna multiplexing** → **Number of entries** enter the number of antennas.

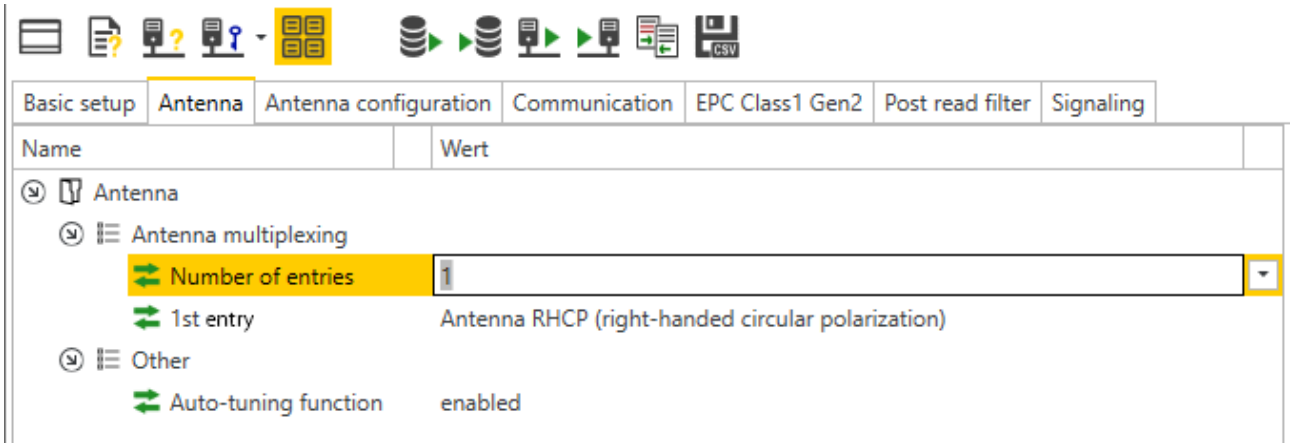



Fig. 21: Entering the number of antennas

- Assign antennas with functions (e.g. internal antenna: RHCP, LHCP, external antenna)

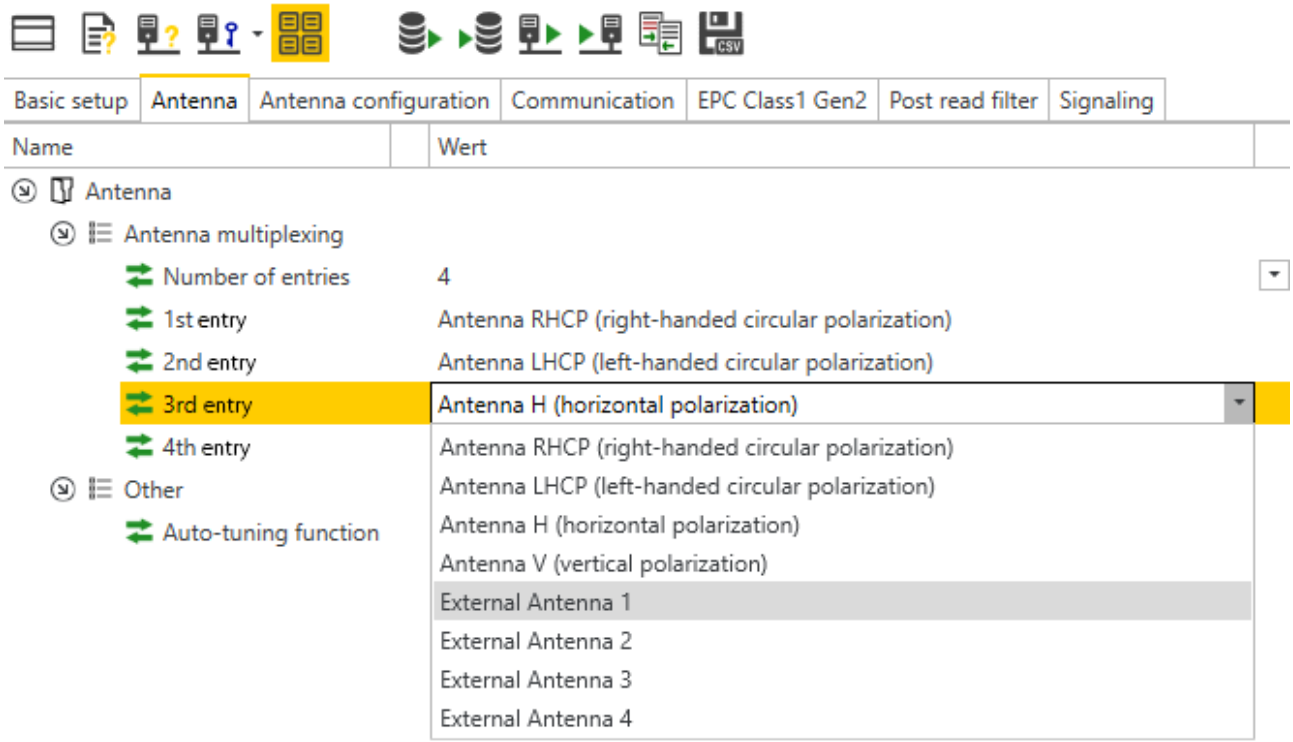



Fig. 22: Example: setting multiplex operation

- ▶ Click **Accept** to save the settings.
- ▶ For all antennas used set at **Antenna configuration** → **Maximal transmit time** the time in which the particular antenna is to remain active and stay switched on.

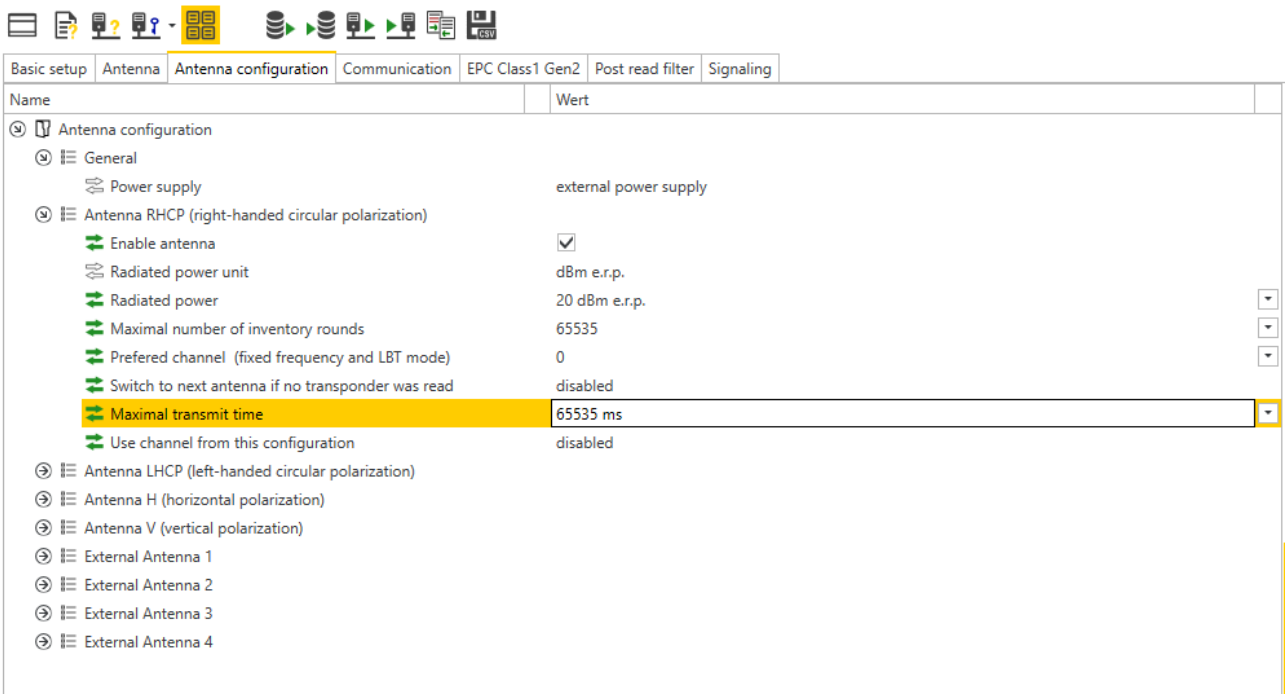



Fig. 23: Setting the maximum transmit time

### 7.1.6 Setting antenna power

The antenna power of the reader can be set for the specific application. The radiated power for the integrated antenna can be entered directly in the DTM. The power must be calculated for external antennas.

The following parameters must be used to calculate the radiated power ($P_{ERP}$):

| | |
|---|---|
| $P_{cond}$ | Power to be output at the TNC socket of the reader |
| dB | Cable attenuation |
| $G_{HW}$ | Antenna gain of the external antenna |

**NOTE**
Refer to the data sheets of the components used for the cable attenuation and antenna gain.

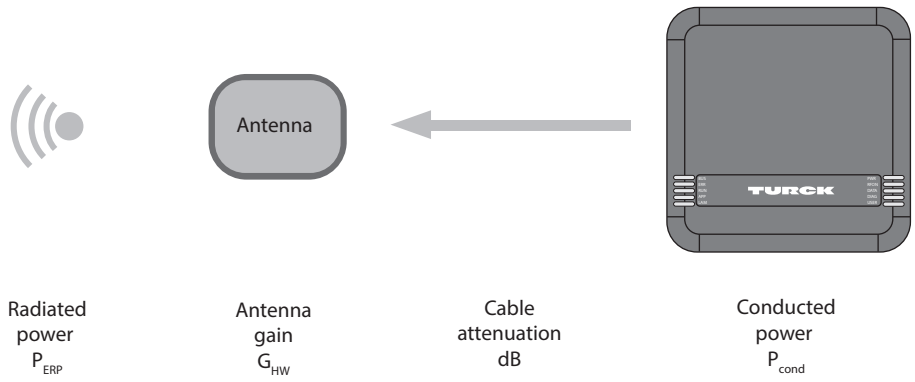



Fig. 24: Power calculation – relevant variables (schematic representation)

The power can be calculated with the following formula:

$P_{ERP} = G_{HW} - dB + P_{cond}$

#### Setting antenna power – restrictions of radio regulations

Some national regulations restrict the degree of freedom available for creating an RFID system. You as the operator are responsible for ensuring that regulations are observed.

- ETSI
  - Radiated power $P_{ERP}$: max. 33 dBm ERP
- FCC
  - Radiated power $P_{ERP}$: max. 36 dBm EIRP
  - $P_{cond}$: max. 30 dBm with antenna gain $G_{HW} \leq 6$ db

**NOTE**
The DTM indicates impermissible configurations with an exclamation mark. A transmission to the device is prevented.

## Calculating radiated power

The effective radiated power (ERP) is the power that is radiated from an antenna into free space. To make it possible to compare the technical properties of different antenna, the power specifications given are always in relation to a reference antenna.

- EIRP = equivalent isotropic radiated power (reference: isotropic antenna)
- ERP = effective radiated power (reference: with the length of λ/2)

The radiated power can be stated in watts or in dBm. The following table shows approximate values as a guide for converting between dBm and mW:

| dBm | mW | dBm | mW | dBm | mW | dBm | mW |
|---|---|---|---|---|---|---|---|
| 1 | 1.25 | 9 | 8 | 17 | 50 | 25 | 316 |
| 2 | 1.6 | 10 | 10 | 18 | 63 | 26 | 400 |
| 3 | 2 | 11 | 13 | 19 | 80 | 27 | 500 |
| 4 | 2.5 | 12 | 16 | 20 | 100 | 28 | 630 |
| 5 | 3 | 13 | 20 | 21 | 125 | 29 | 800 |
| 6 | 4 | 14 | 25 | 22 | 160 | 30 | 1000 |
| 7 | 5 | 15 | 32 | 23 | 200 | … | … |
| 8 | 6 | 16 | 40 | 24 | 250 | 33 | 2000 |

The formula for calculating the exact values is: **dBm = 10 × lg (P/1 mW)**

## Converting antenna gain

The antenna gain can be specified in the following units:

| | |
|---|---|
| dBd | Antenna gain in relation to a dipole |
| dBi | Antenna gain in relation to an isotropic radiator (linear) |
| dBic | Antenna gain in relation to an isotropic radiator (circular) |

The different units can be converted as follows:

- $G_{HW}$ = dBd
- $G_{HW}$ = dBi - 2.15
- $G_{HW}$ = dBic - 5.15

## Setting the power for external antennas via the DTM

When supplied via Power over Ethernet (PoE), the radiated power for the internal antenna is limited to 1 W. With external antennas 1 W of output power is provided at the TNC socket. The power supply type is set automatically via **Antenna configuration** → **Power supply** to the **external power supply** value.

- ▶ Set the radiated power via **Antenna configuration** → **Radiated power** (here: 33 dBm ERP).

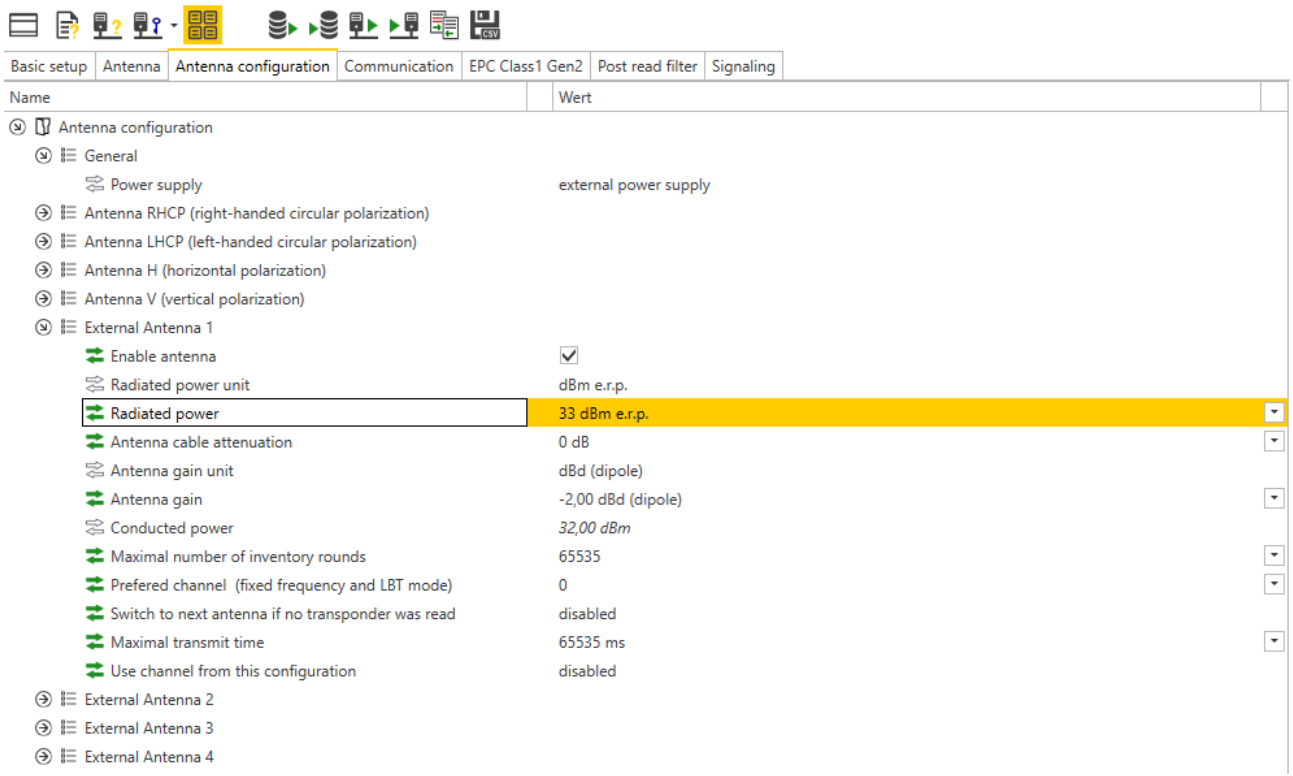



Fig. 25: Setting the radiated power

- Refer to the data sheet of the cable used for the cable attenuation.
- Enter the cable attenuation at **Antenna cable attenuation**.

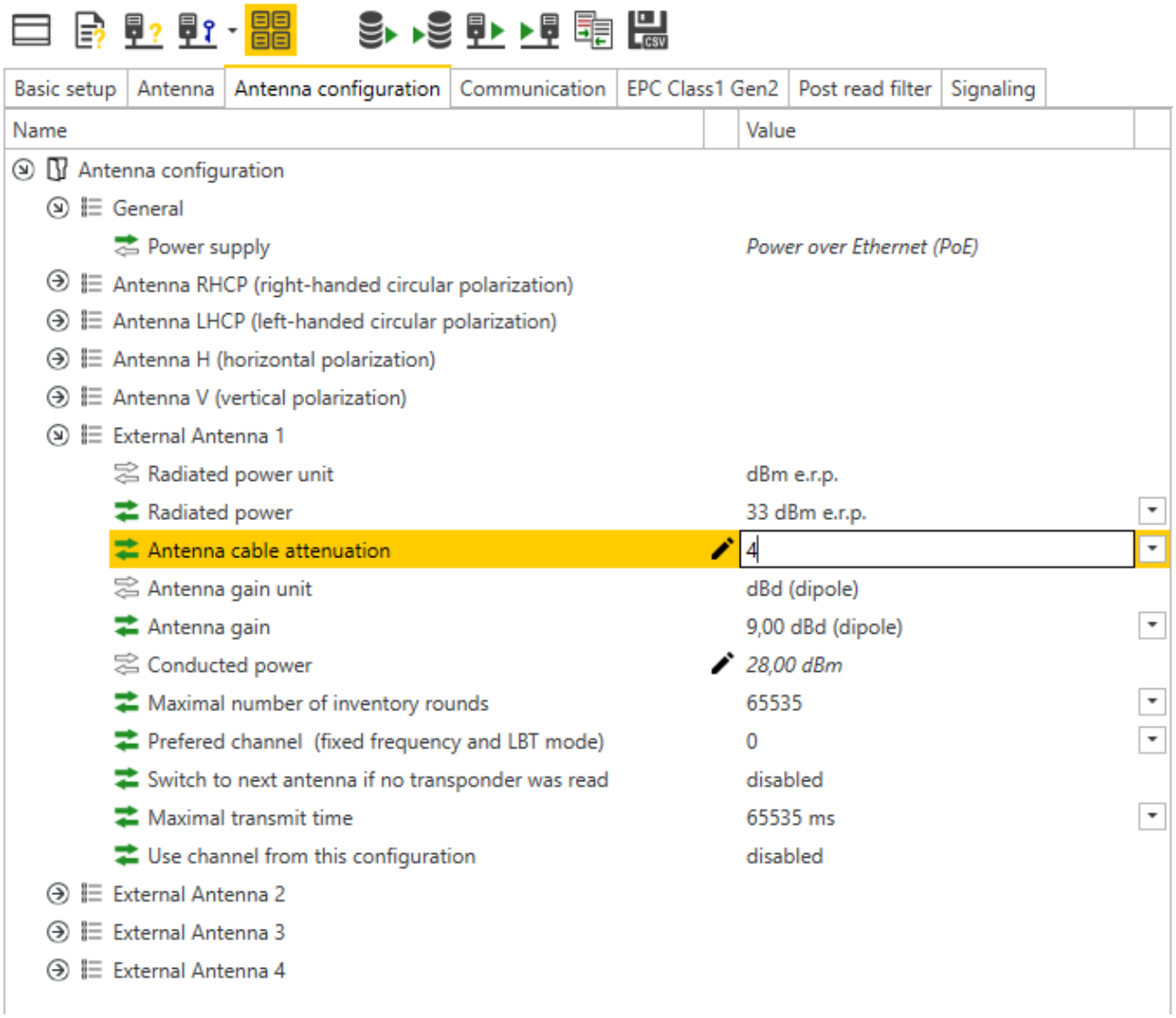



Fig. 26: DTM – entering the cable attenuation


Hans Turck GmbH & Co. KG | T +49 208 4952-0 | F +49 208 4952-264 | more@turck.com | www.turck.com

- Refer to the data sheet of the external antenna for the antenna gain.
- Set the unit for the antenna gain at **Antenna gain unit** (here: dBd).

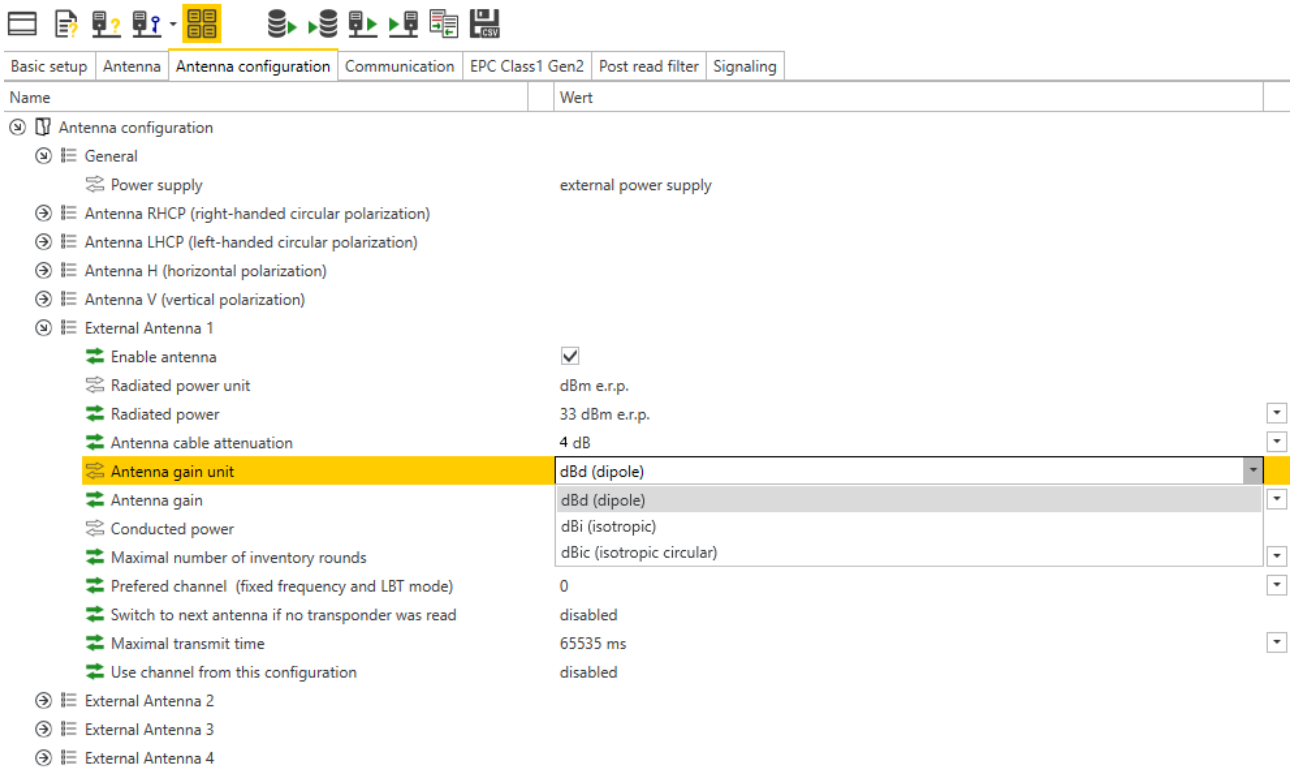



Fig. 27: Setting the unit for the antenna gain

▶ Set antenna gain at **Antenna gain** (here: 9.00).

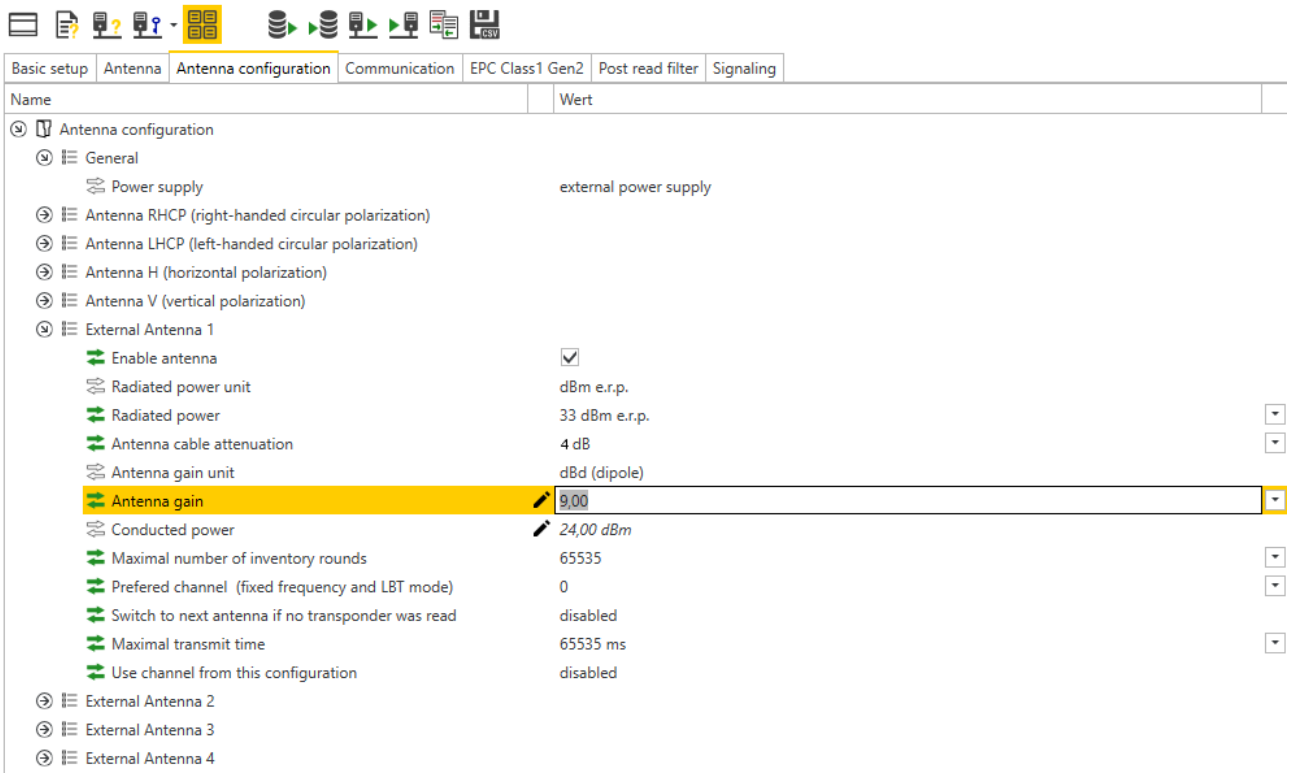



Fig. 28: Setting antenna gain

The power at the TNC socket ($P_{cond}$) is calculated automatically by the DTM and displayed at **Conducted power**.

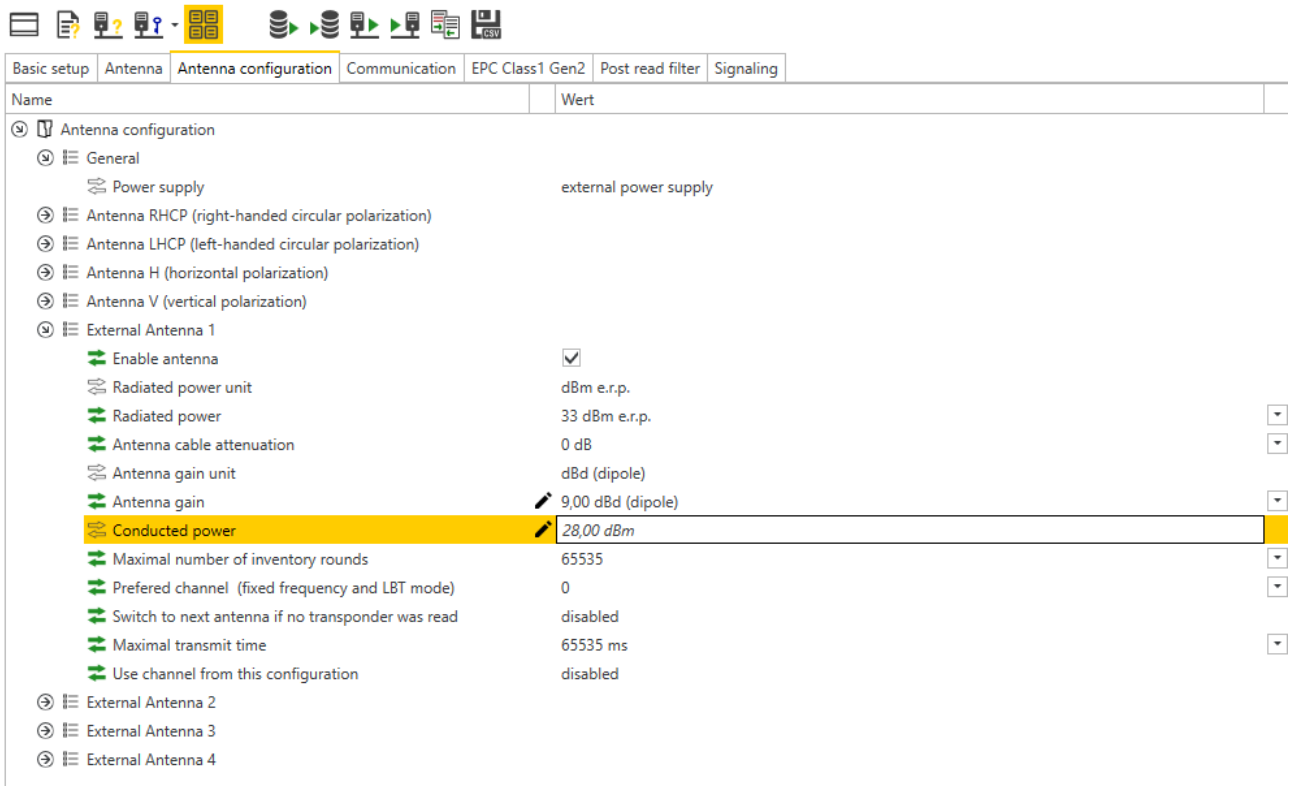



Fig. 29: Display of the power at the TNC socket

- ▶ Click **Accept** to save the settings.
- ▶ Set the power for each additional antenna separately.

### 7.1.7 Setting antenna polarization

The antenna polarization can be switched via the DTM. Switching the polarization makes it possible to change null spots caused by interference. The detection rate can be increased by switching the polarization. Polarization switching is suitable for example in single tag applications in particularly metallic environments.

The following graphics schematically illustrate the possibilities of antenna polarization.

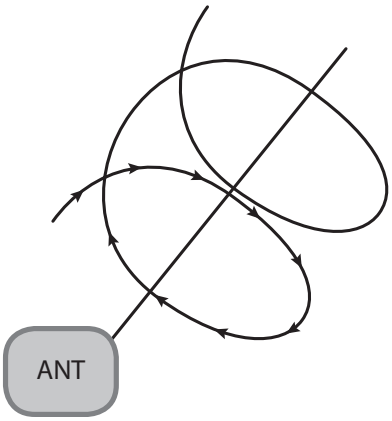



Fig. 30: Antenna polarization circular (RHCP)

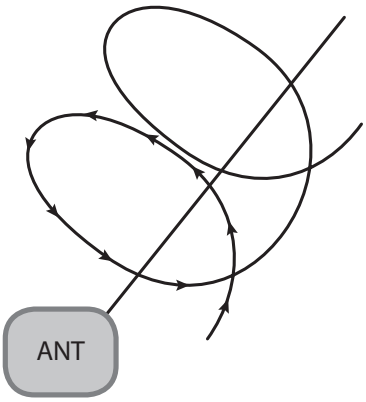



Fig. 31: Antenna polarization circular (LHCP)

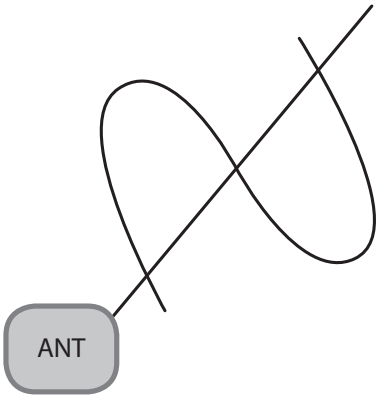



Fig. 32: Antenna polarization linear (vertical)

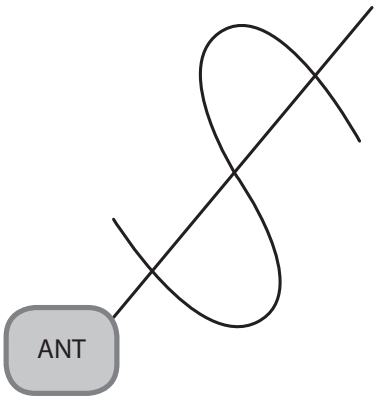



Fig. 33: Antenna polarization linear (horizontal)

## Switching antenna polarization

Polarization switching is activated in the DTM via the multiplex settings.

- Set at **Antenna** → **Number of entries** the value **2**.
- Set at **Antenna** → **1st entry** the value **Antenna LHCP**.
- Set at **Antenna** → **2nd entry** the value **Antenna RHCP**.

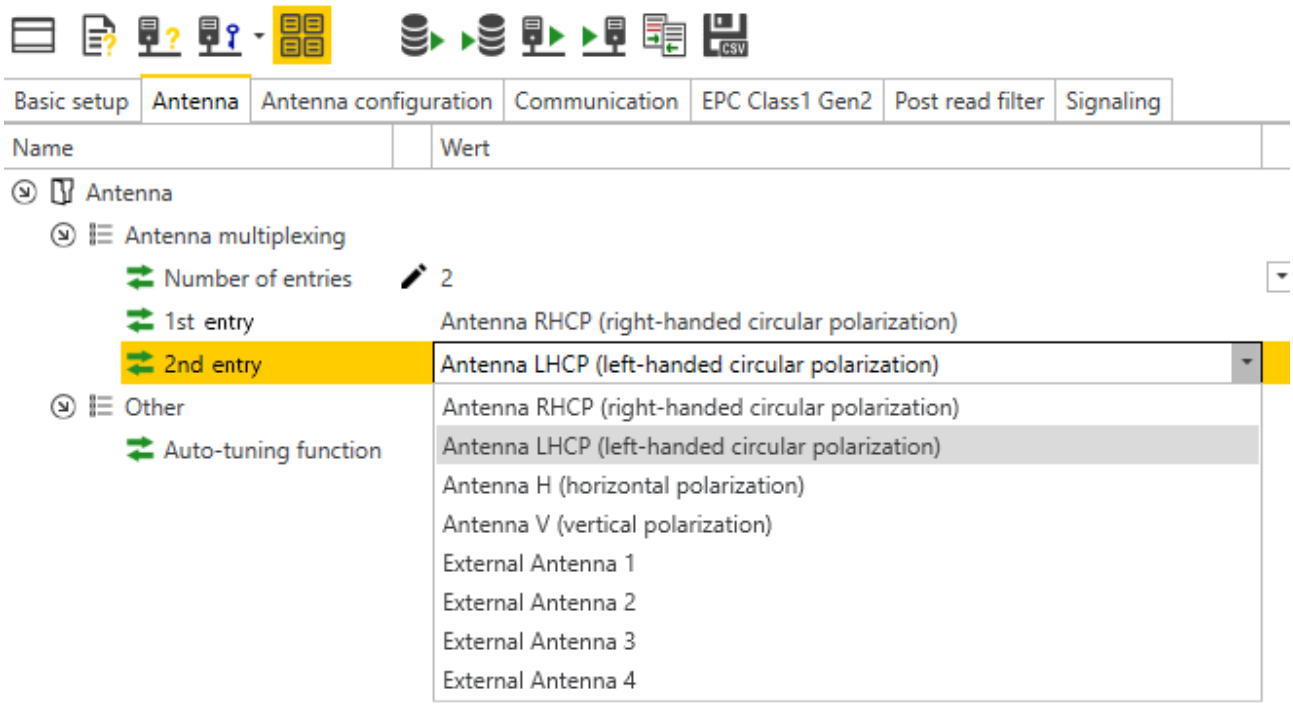



Fig. 34: Switching antenna polarization

- ▶ At **Antenna configuration** → **Maximal transmit time** set the time up to the polarization switch or activate the **Switch to next antenna if no transponder was read** option.
- ⇨ If the **Switch to next antenna if no transponder was read** option is activated, the reader automatically switches after an Inventory operation without reading to the next multiplex sequence (**Entry**).

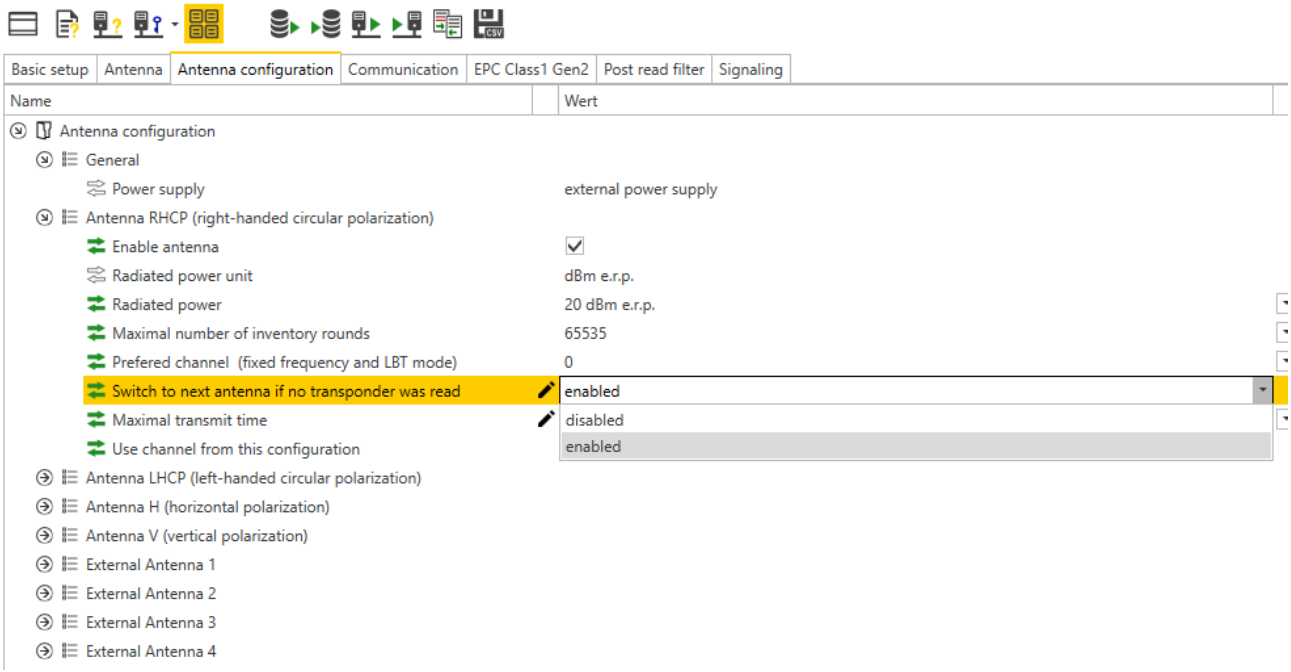



Fig. 35: Switching polarization automatically

### 7.1.8 Switching on presence sensing mode

In order to use the Continuous presence sensing mode command, the Presence sensing mode must be activated in the reader. In Continuous presence sensing mode, the readers are automatically switched on as soon as a tag is located in the detection range.

- ▶ Choose **Basic Setup** → **General** → **Device Mode** and set the **Presence sensing mode** option.

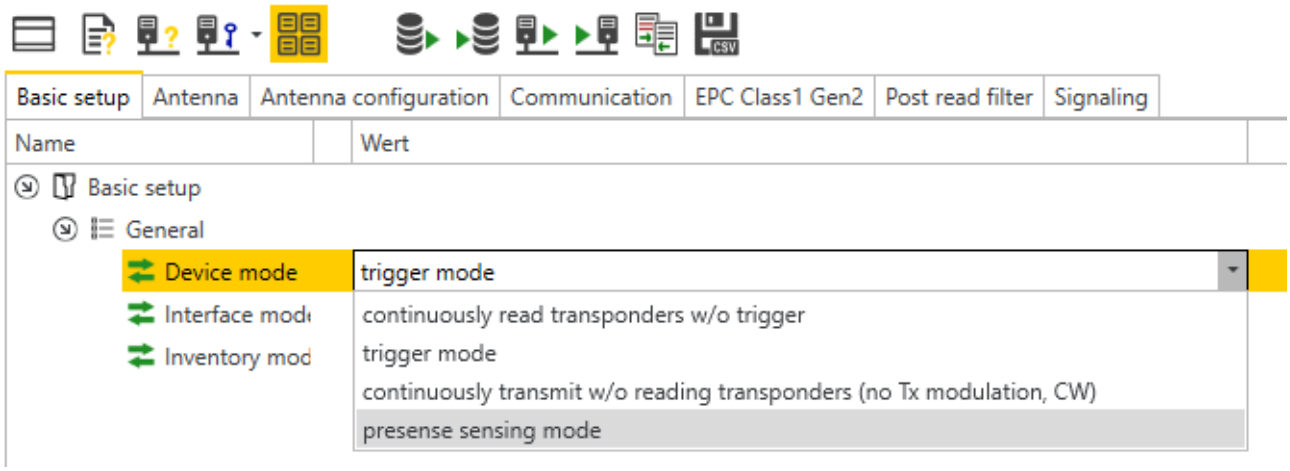



Fig. 36: Switching on presence sensing mode

The Advanced access level allows the **Tag data delay time** and **Carrier delay time** parameters to be set individually.

- **Tag data delay time**: Time in which the reader searches for a tag. If a tag is found, the field is switched on. In the Basic access level, the parameter is set by default to 100 ms.
- **Carrier delay time**: Time until the reader switches off the field after the last read operation. In the Basic access level, the parameter is set by default to 65535 ms.

**NOTE**
Report mode is recommended for the RFID test since the read tag information items appear in the RFID test window and do not have to be polled individually.

### 7.1.9 Transferring the RSSI value – communication

The **Communication** tab is used to set the parameters for the configuration of the deBus messages. All parameters and the settable values are written in the DTM.

#### Example: switch on RSSI transmission

- ▶ Switch on RSSI transmission: choose **Communication** → **Message data content** → **Transponder RSSI** and select the **enabled** option.
- ⇨ The RSSI value is displayed with the inventory in the read data.

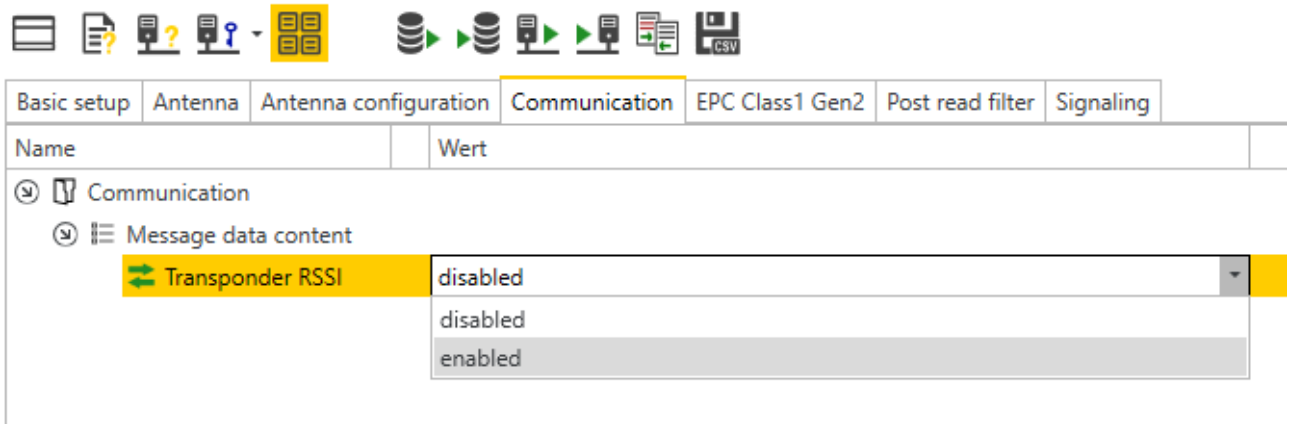



Fig. 37: Switching on RSSI transmission


Hans Turck GmbH & Co. KG | T +49 208 4952-0 | F +49 208 4952-264 | more@turck.com | www.turck.com

### 7.1.10 Setting the air interface parameters – EPC Class 1 Gen 2

The EPC Class1 Gen2 tab is used to set the **EPC Class1 Gen2** parameters for the air interface. The parameters set here are used if the reader performs an Inventory command. All parameters and the settable values are written in the DTM.

### 7.1.11 Setting the RSSI filter – post read filter

The **Post read filter** tab enables parameters to be set in order to filter event messages.

The set filters do not reduce the data traffic on the air interface and are not suitable for multi-tag applications with many tags or high passing speeds. All parameters and the settable values are written in the DTM.

#### Example: set the RSSI filter

An RSSI filter makes it possible to prevent unwanted read operations. All read operations with an RSSI outside of the set limit values are filtered out and not displayed.

- ▶ At **Post read filter** → **RSSI filter** switch on the RSSI filter.

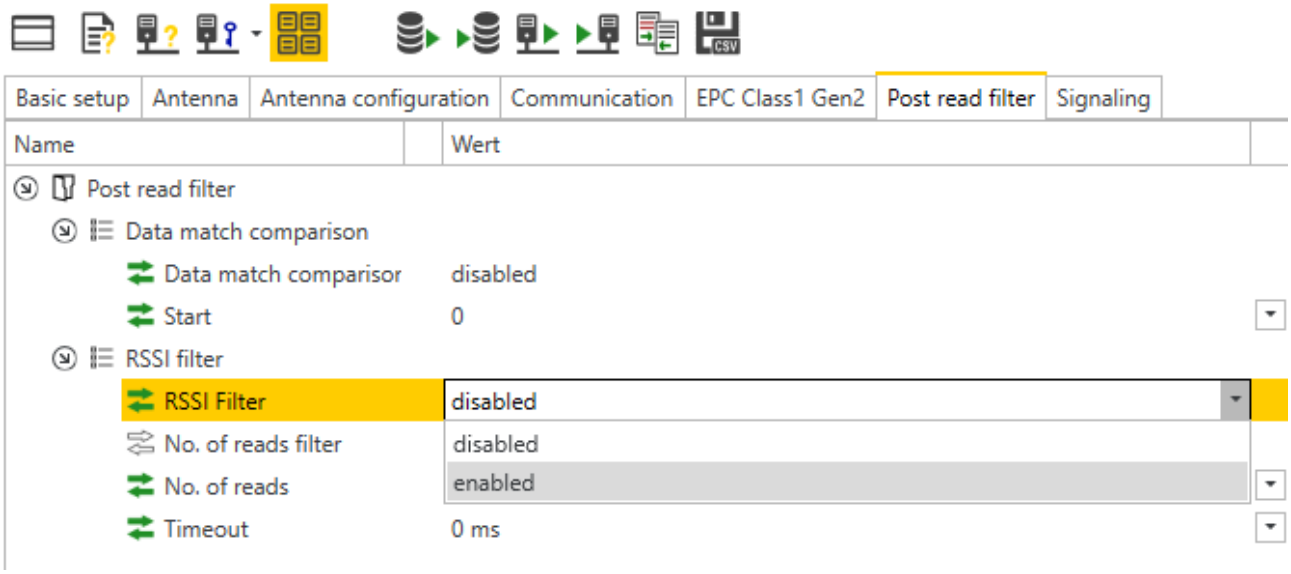



Fig. 38: Switching on the RSSI filter

- ▶ Set the limit values at **Post read filter** → **RSSI filter** → **Lower threshold**.
- ⇨ Example: all read operations below an RSSI value of -45 dBm are filtered out.

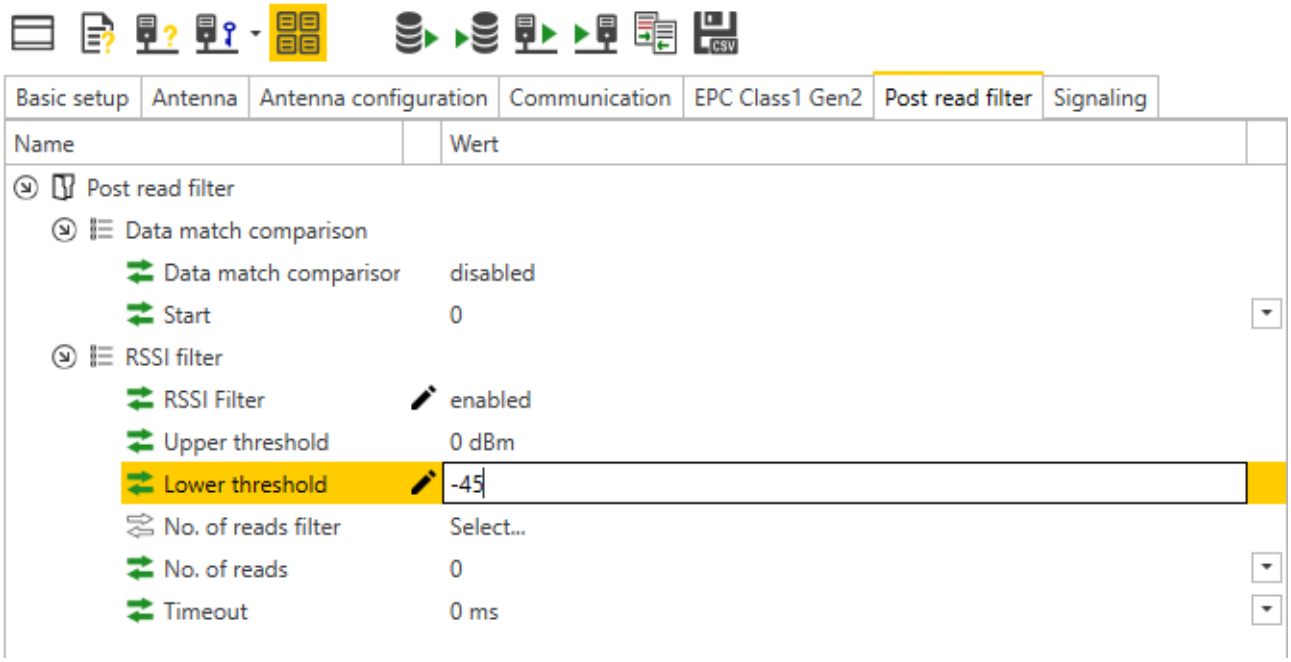



Fig. 39: Example – setting the limit value for RSSI

### 7.1.12 Setting LED indication – signaling

The **Signaling** tab enables the default settings for the USER LEDs to be edited. All parameters and the settable values are written in the DTM.

## 7.2 Testing readers with the DTM

The following functions can be executed via the RFID Test in the DTM:

- Display read data
- Displaying the log of the communication between host or PC and reader
- Logging of the interface communication between host or PC and reader
- Send user-specific deBus commands
- Write tags with a user-defined number
- Send tag-specific commands

### Requirements for the RFID test

- PACTware must be installed.
- The DTM for UHF readers must be installed.
- The connection between the reader and the PC must be established.
- A project must have been created in PACTware.

### 7.2.1 Starting the RFID Test

- Right-click the device in the project tree.
- In the context menu choose **Additional functions** → **RFID Test**.

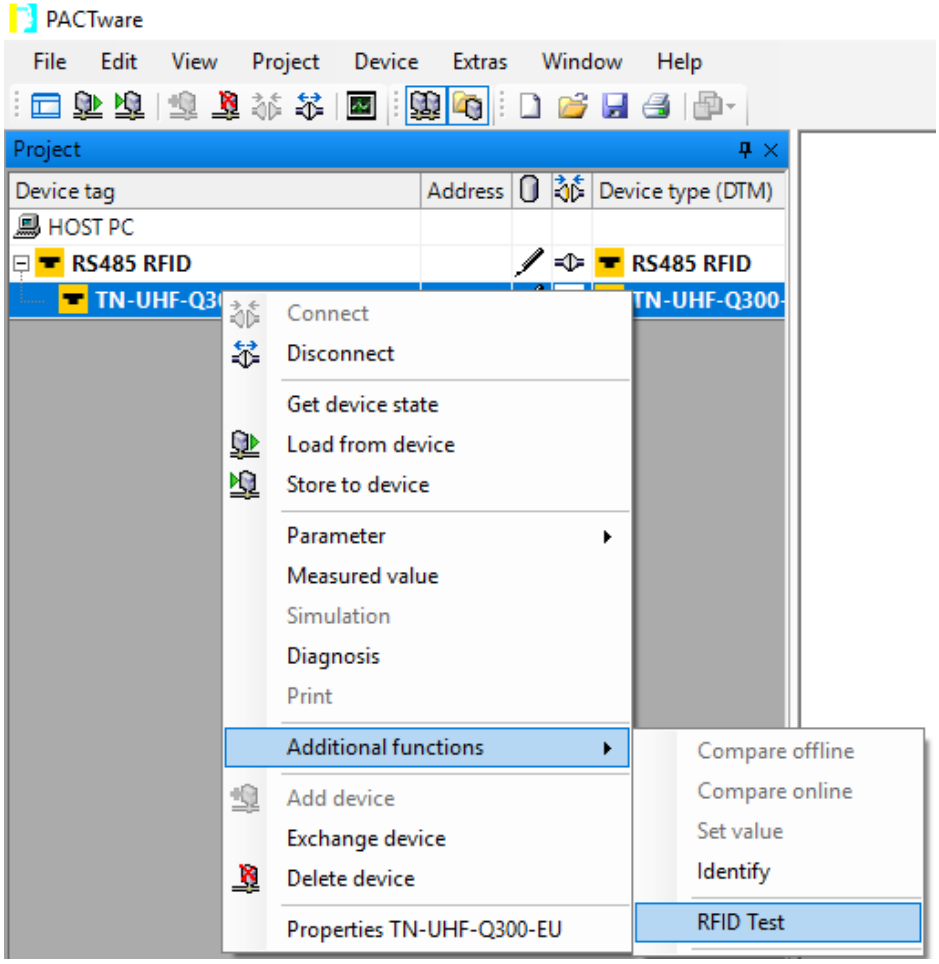



Fig. 40: Starting the RFID Test

### 7.2.2 Start window – overview

The **RFID Test** window consists of the following elements:

- Main menu
- Basic test
- Tag actions
- Reader Status
- Logger

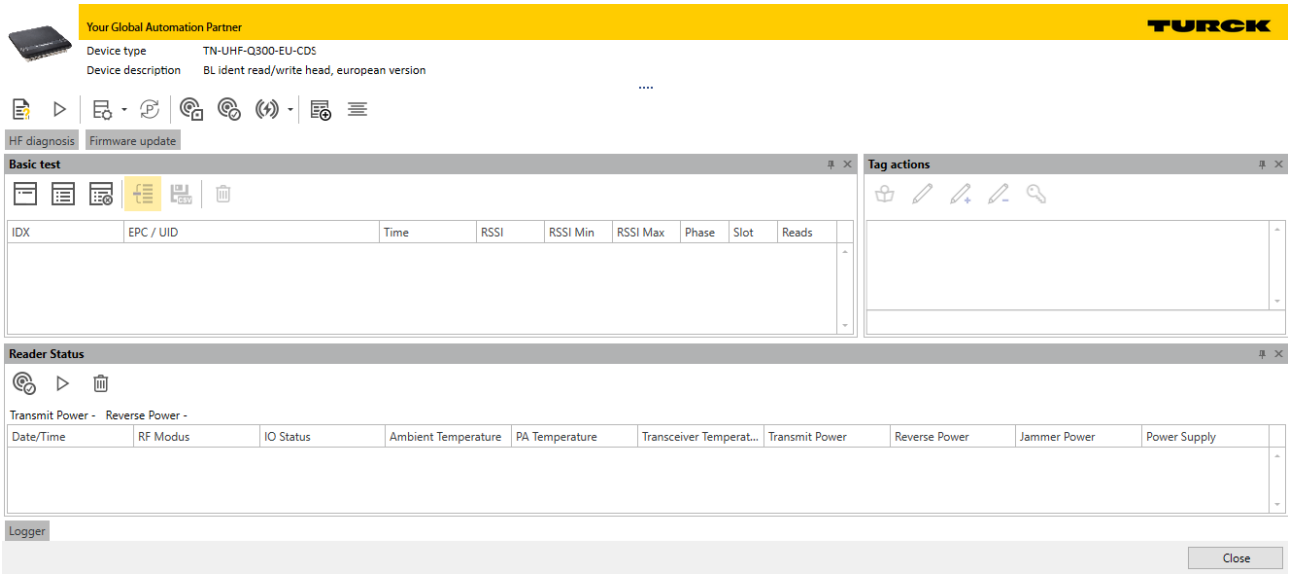



Fig. 41: RFID Test – overview of the start window


Hans Turck GmbH & Co. KG | T +49 208 4952-0 | F +49 208 4952-264 | more@turck.com | www.turck.com

### 7.2.3 RFID Test – main menu


Fig. 42: RFID Test – main menu

The main menu provides the following functions:

| Icon | Function | Description |
|---|---|---|
| | Help on the DTM | Starts the DTM help. |
| | Start trigger/ON<br>or<br>Stop trigger/OFF | Starts the trigger for command execution (standard view).<br><br>Ends the trigger for command execution ( displayed after clicking the start button). |
| | Configure message content | Displays the content to be transferred with a read operation. The following can be selected:<br>■ Phase<br>■ RSSI<br>■ Slot<br>■ Time |
| | Switch mode (report/polling) | Switches between Report mode (automatic read/write) and polling mode (read/write started through an explicit polling command). |
| | Reading the reader status | Calls the status of the reader and provides the information in the **Logger** window. |
| | Reading the reader version | Calls the following information from the reader and provides the information in the **Logger** window:<br>■ Hardware revision<br>■ Firmware status<br>■ Serial number |
| | Resetting the reader | Offers three ways of resetting the reader:<br>■ Voltage reset<br>■ Factory reset: Reset to factory settings<br>■ Reset the reader status<br><br>When resetting to factory settings, any modified transfer rate or RS485 address is not changed because the reader could not otherwise be addressed any longer. |
| | Set the current window layout as the default | Saves the individually set window layout. |
| | Reset window layout | Resets the window layout. |
| HF diagnosis | HF diagnosis | Opens the window for HF diagnostics. |
| Firmware update | Firmware update | Opens the window for the firmware update. |

### 7.2.4 RFID Test – basic test window

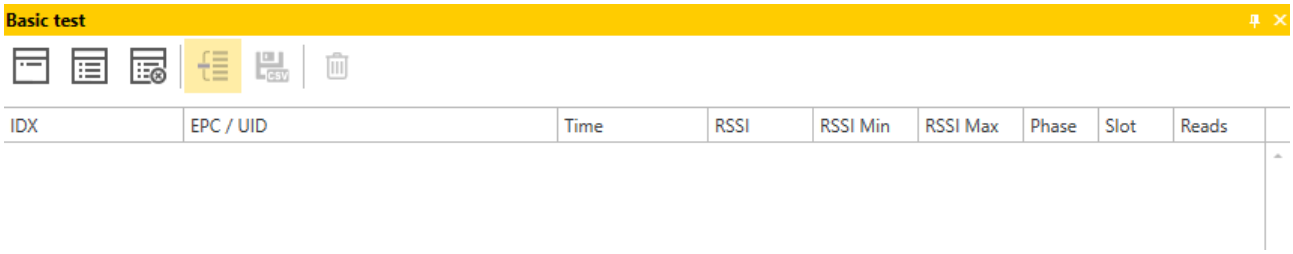



Fig. 43: RFID test – Basic test window

The following functions are available in the **Basic test** window:

| Icon | Function | Description |
|---|---|---|
| | Polling | Shows the first tag in the polling memory of the device in the tag list. The function is only available in polling mode. |
| | Poll all | Shows all tags in the polling memory of the device in the tag list. The function is only available in polling mode. |
| | Delete tags polled by the reader | Clears the polling memory of the reader. |
| | EPC grouping | Combines readings of tags with the same EPC. |
| | CSV export of the current data | Saves the tag list in CSV format. |
| | Delete tag list | Deletes the list of displayed tags. |

The queried data is displayed in the tag list. The content of the message can be set via the **Configure message content** function.

**NOTE**
If the polling memory of the reader is full, the ERR LED is lit red and indicates an internal error.

### 7.2.5 RFID Test – tag actions window

The functions in the **Tag actions** window are available if a tag is selected in the tag list of the **Basic test** window.

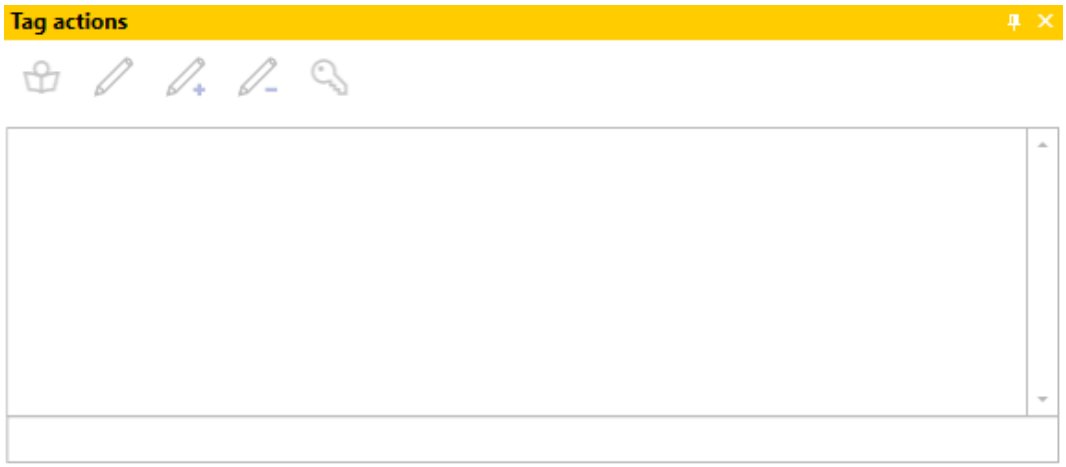



Fig. 44: RFID test – Tag actions window

The following functions are available in the **Tag actions** window:

| Icon | Function | Description |
|---|---|---|
| | Read tag memory | Starts the read operation.<br>The chip type is automatically displayed. One word is always read with the first read operation. The following parameters can be set for other read operations:<br>■ Memory bank (TID, EPC/UID, PC, access password or kill password)<br>■ Start word<br>■ Number of words<br><br>The read data is displayed in the **Data** area. |
| | Write tag memory | Starts the write operation.<br>The chip type is automatically displayed. The following parameters can be set for the write operations:<br>■ Memory bank (TID, EPC/UID, PC, access password or kill password)<br>■ Start word<br>■ Number of words<br><br>Data to be written is displayed in the **Data** area. |
| | Auto-increment | The EPC is incremented automatically by 1. |
| | Auto-decrement | The EPC is decremented automatically by 1. |
| | Activate and deactivate the access password | Switches the password for write or read access on or off. |

### Example: execute tag actions

- Position the tag within the detection range of the reader.
- Activate the trigger for the reader in the main menu.


Fig. 45: Main menu – activating trigger

- **Basic test** window: Execute polling command in order to display tag in the tag list.
- **Basic test** window: Select tag from the tag list.

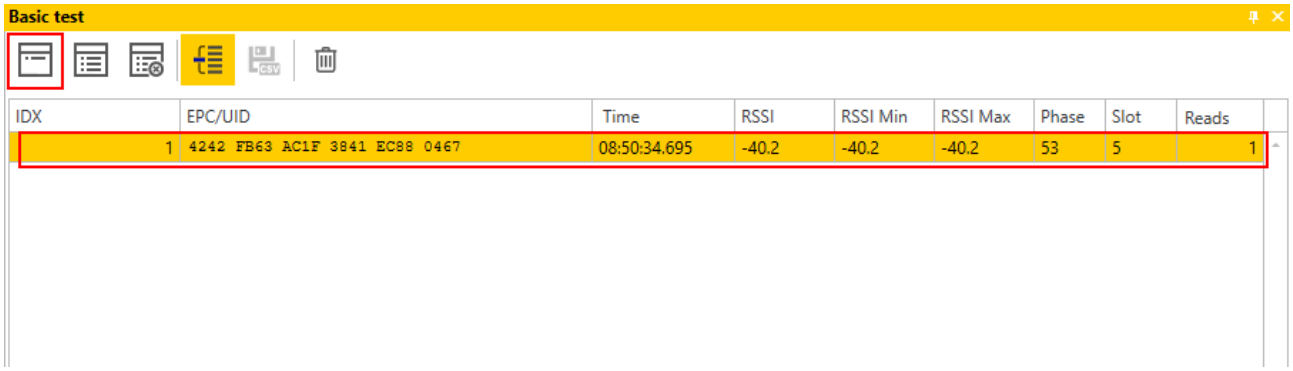



Fig. 46: Basic test – Tag selection

- **Tag actions** window: To read, select **Memory location**, **Start word** or **Word length** and click the appropriate icon.
- To write, enter values under **Data** and confirm with **OK**

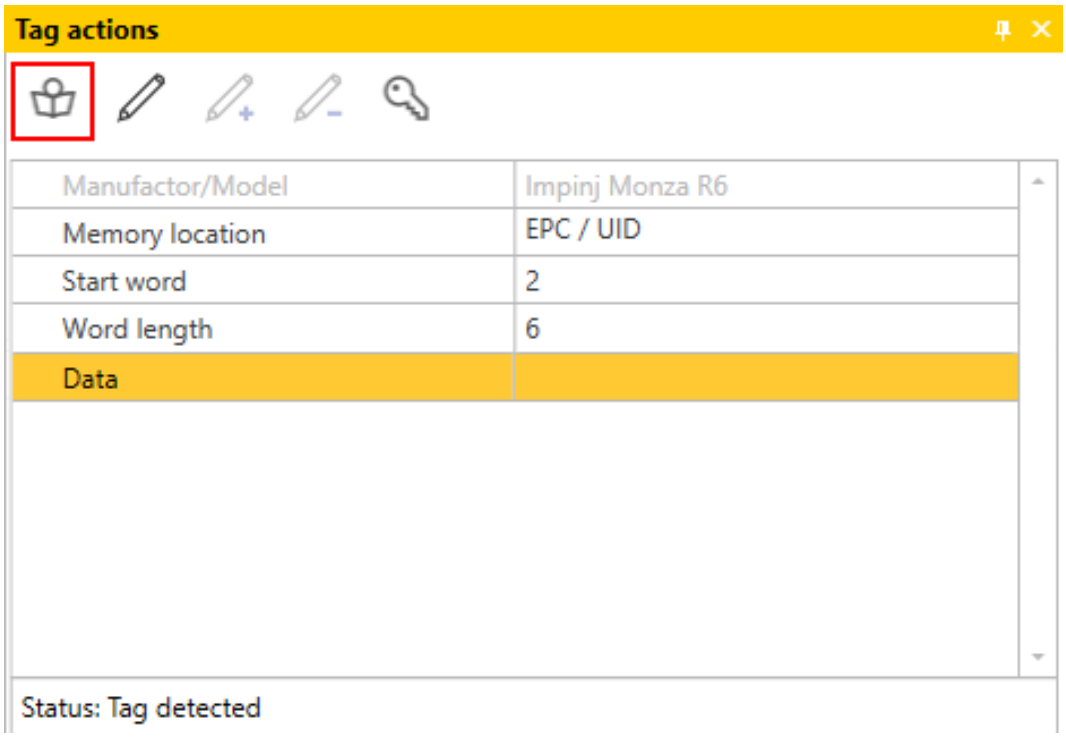



Fig. 47: Execute tag action (example: read)

⇨ Successful access is displayed via the status message at the bottom of the window.

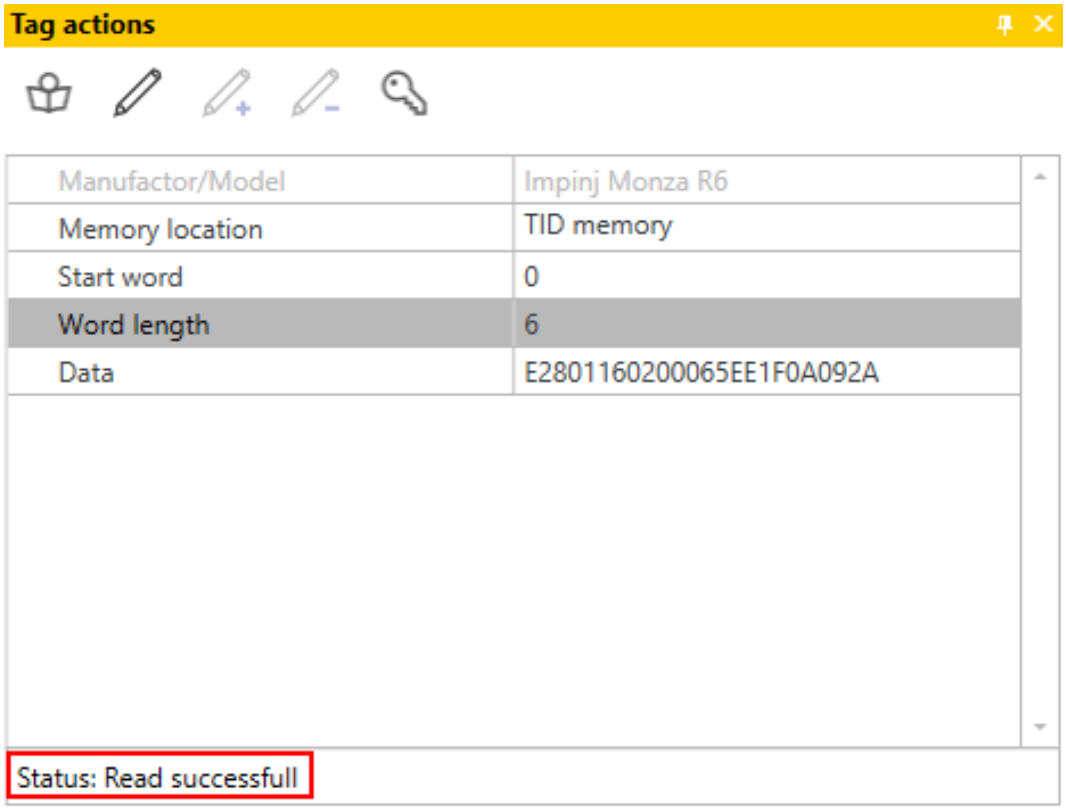



Fig. 48: Example: read successful


Hans Turck GmbH & Co. KG | T +49 208 4952-0 | F +49 208 4952-264 | more@turck.com | www.turck.com

### 7.2.6 RFID Test – logger window

The **Logger** window displays read/write information and error messages. The list can be cleared via the **Delete** button.

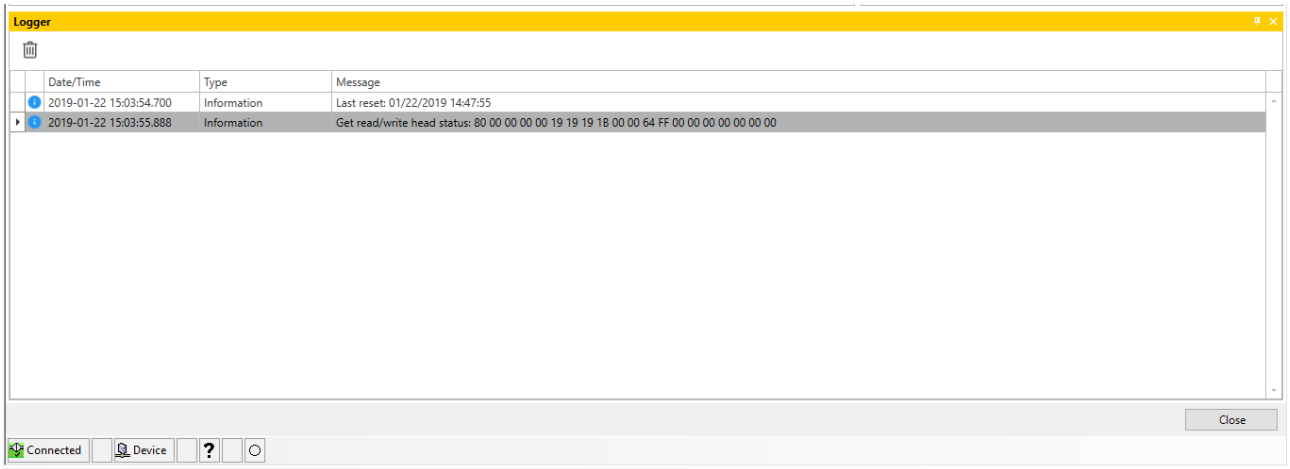



Fig. 49: Messages in the Logger window

### 7.2.7 HF diagnosis window

Interference frequencies affecting the respective channels are displayed in the **HF diagnosis** window.

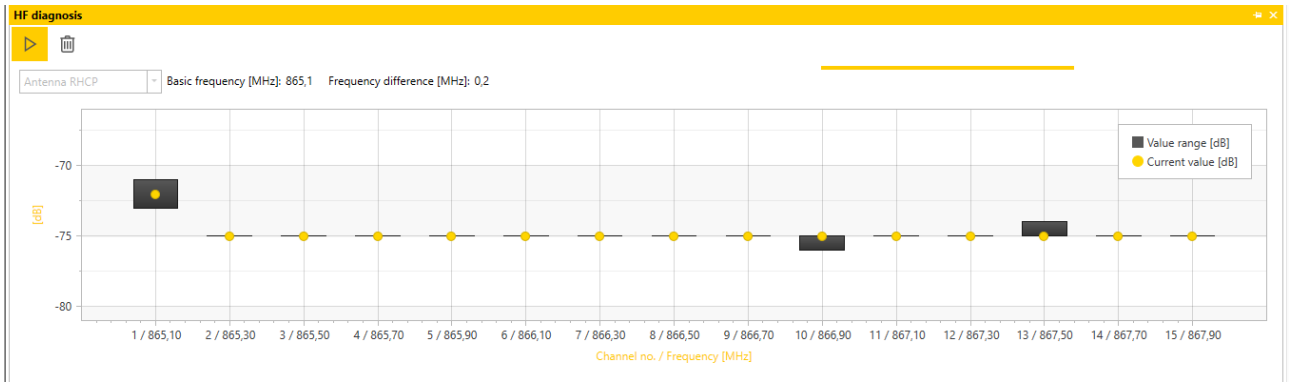



Fig. 50: HF diagnosis window

The following functions can be executed in the **HF diagnosis** window:

| Icon | Function | Description |
|---|---|---|
| ▷ | Start/stop HF diagnosis | Starts or closes the HF diagnosis. |
| 🗑 | Clear values | Deletes the displayed values. |

## 7.3 Querying device information with the DTM

The DTM provides access to hardware and software information as well as regulations on the connected device.

- ▶ Right-click the device in the project tree.
- ▶ Choose **Additional functions** → **Identify**.
- ⇨ The DTM shows the available information on the device according to the selected access level.

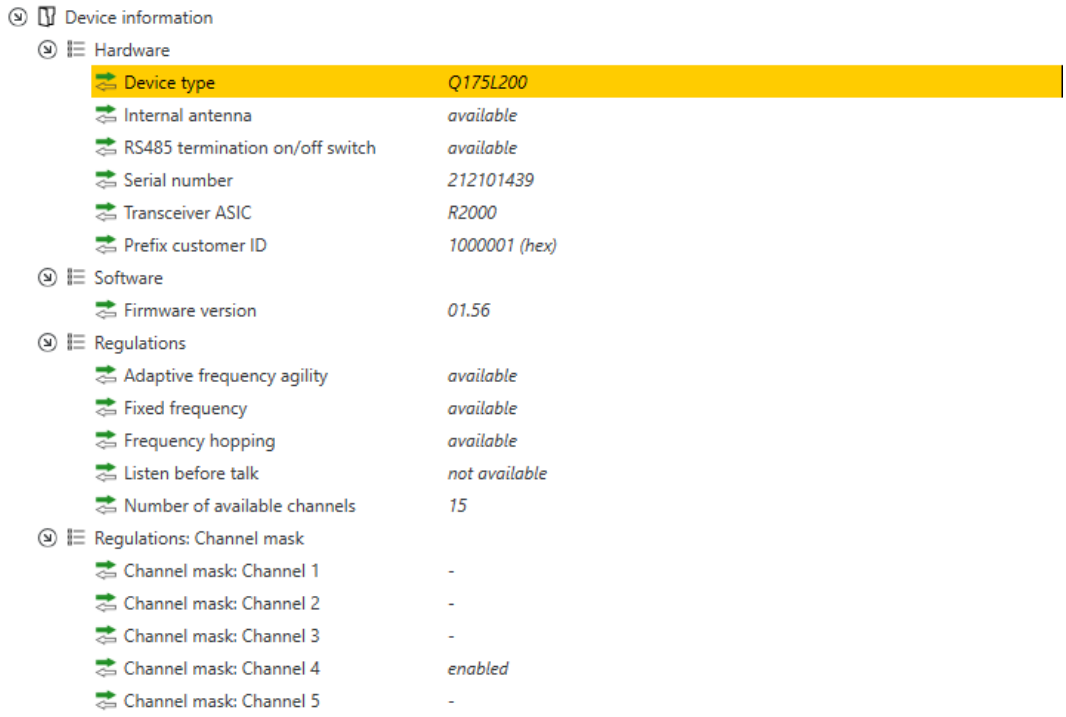



Fig. 51: Device information for TN865-Q175L200-H1147 in the Advanced access level

## 7.4 Setting the IP address

The IP address can be set via the Turck Service tool or via DHCP.

### 7.4.1 Setting the IP address via the Turck Service Tool

The device is factory set to IP address 192.168.1.254 The IP address can be set via the Turck Service Tool. The Turck Service Tool is available free of charge from www.turck.com.

- Connect the device to a PC via the Ethernet interface.
- Launch the Turck Service Tool.
- Click **Search** or press F5.

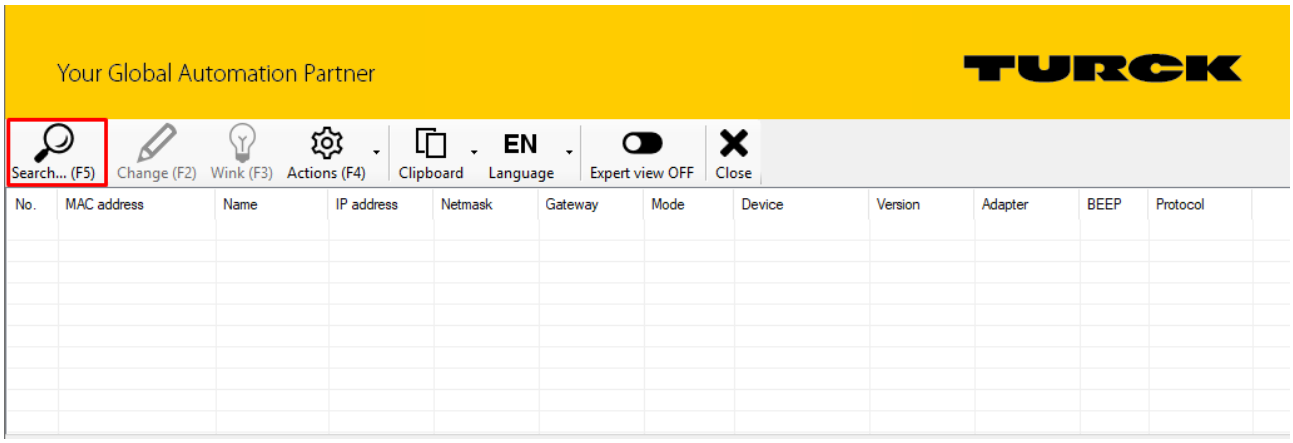



Fig. 52: Turck Service Tool – start screen

The Turck Service Tool displays the connected devices.

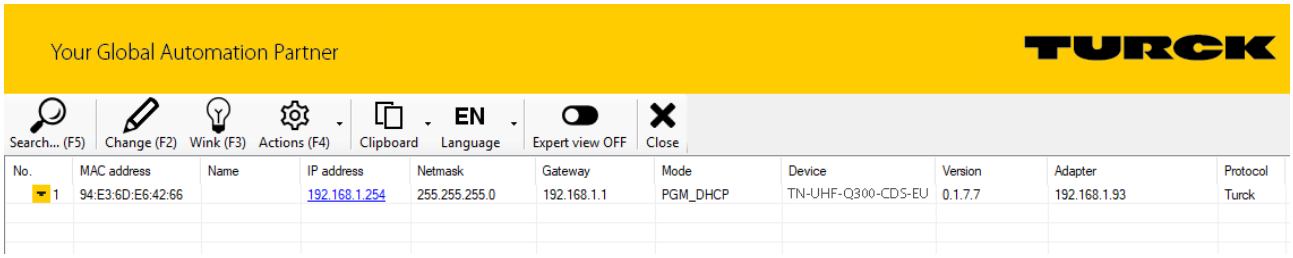



Fig. 53: Turck Service Tool – found devices

- Click the required device.
- Click **Change** or press [F2].

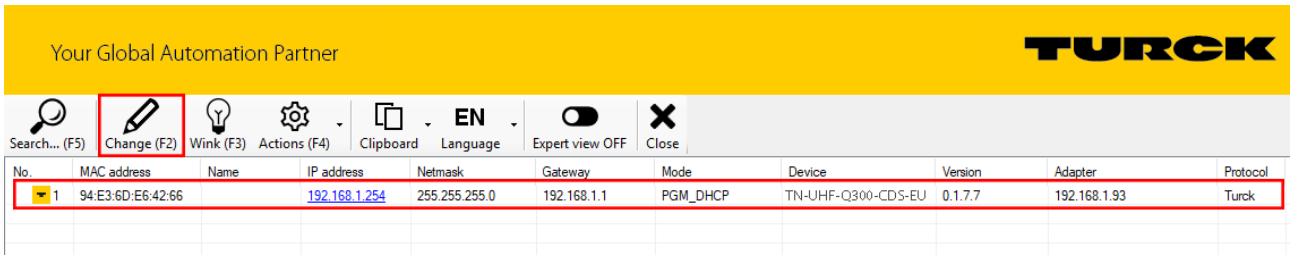



Fig. 54: Turck Service Tool – selecting the device to be addressed

**NOTE**
Clicking the IP address of the device opens the web server.

- Change the IP address and if necessary the network mask and gateway.
- Accept the changes by clicking **Set in device**.

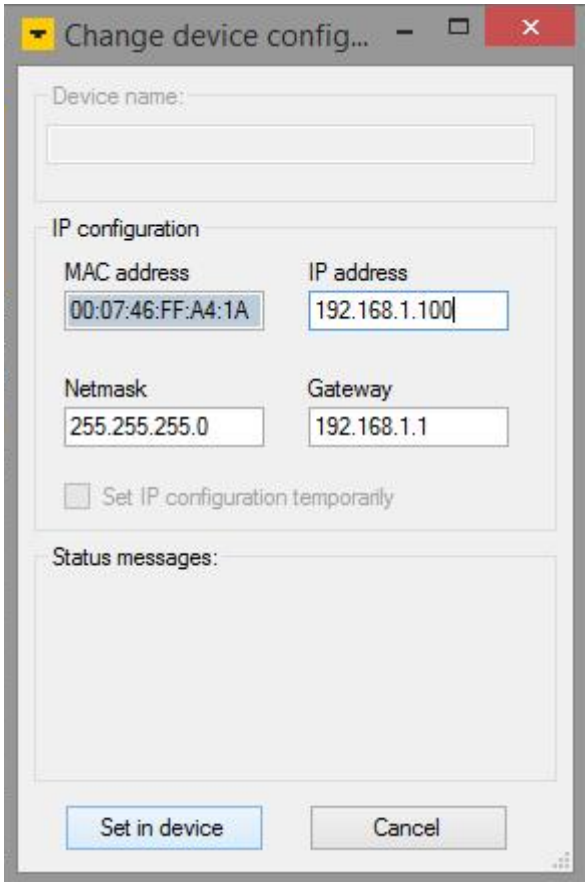



Fig. 55: Turck Service Tool – changing the device configuration

## 7.5 Programming RFID channels

The RFID channel is designed as a (`/dev/tty01` or `/dev/COM0`) serial interface.

### 7.5.1 Programming RFID channels with Python 3

The following examples illustrate the programming of the RFID interface with Python 3.

#### Example 1: using the “pySerial” module

```
import serial # from module pySerial


# open serial interface on port 0 and set a timout of 8 seconds
seri = serial.Serial("/dev/COM0", timeout=8)


# change settings
seri.baudrate = 115200 # set the baudrate of port COM0 to 115200
seri.parity ='N' # set no parity for port COM0
seri.bytesize = 7 # set the byte size for a sign to 7 for port
COM0
seri.stopbits = 1 # set stopbits to 1 for port COM0


seri.write(bytearray.fromhex("aa 07 07 49 00 41 23")) # writes a
bytestream


print(seri.readline()) # reads incoming message as ascii
```

Example 2: using the “periphery” module

```
from periphery import Serial


# Open /dev/COM0 with baudrate 115200, and defaults of 8N1, no
flow control
serial = Serial("/dev/COM0", 115200)


# write a bytestream serial.write(bytearray.fromhex("aa 07 07 49
00 41 23"))


# Read up to 128 bytes with 500ms timeout
buf = serial.read(128, 0.5)


print(buf)
print("read %d bytes: _%s_" % (len(buf), buf))


serial.close()
```

### 7.5.2 Programming RFID channels with C or C++

The following examples illustrate the programming of the RFID interface with Ansi C or C++.

```
#include <stdio.h>
#include <stdlib.h>
#include <termios.h>
#include <fcntl.h>

// initialize function (use extern for C++)
ssize_t write (int __fd, void *__buf, size_t __n) __wur;
ssize_t write (int __fd, const void *__buf, size_t __n) __wur;
int close (int __fd);

int main(void) {
     //choose Interface for connection
    const char *Path = "/dev/COM0";
    struct termios options;
    int fd, count, i ;
    unsigned char currentBuff[1];
    unsigned char InBuff[255];
    unsigned char *p_InBuff = InBuff;
    unsigned char Message[] = {0x0D}; //deExample command for "Get
status"
    if((fd = open((Path), O_RDWR | O_NOCTTY)) != -1)
    {
         // Set serial Interface
        tcgetattr (fd, &options);

        cfsetspeed(&options, B115200);
        options.c_cflags |= CS8|CSTOPB;
        options.c_flags &= ~(PARENB.PARODD)
        tcsetattr (fd, TCSANOW, &options);

        // write to Interface COM0
        if ((write(fd, Message, sizeof(Message))) == -1)
                {
                        printf("not able to write...");
                 }
```

```
        // read from Interface COM0
        count = 0;
        do
        {
                 if ((count += read(fd, currentBuff, 1)) == -1)
                        printf("can not read...");
                *p_InBuff = currentBuff[0];
                p_InBuff++;
        }while(currentBuff[0] != 0xfe);

        // print:
         p_InBuff -= count;
        printf("\nData count: %i",count+1);
        printf("\nValues: \n");
         for(i = 0; i <= count ; i++)
        {
                 printf("%.02x ", *p_InBuff ); p_InBuff++;
        }

        // close the Interface
        if((close(fd)) == -1)
        {
                 printf("\n can not close interface");
        }
    }
    else
        printf("can not open interface\n");
     return EXIT_SUCCESS;
}
```

## 7.6 Programming digital channels (DXP)

### 7.6.1 GPIOs of the DXP channels – overview

The digital I/O channels (DXP) can be programmed as inputs or outputs via the GPIOs. The GPIOs are located under the following path:
/sys/class/gpio/…

| Channel | Socket | Type | GPIO | Possible values |
|---|---|---|---|---|
| DXP0 | C0 | Input | 59 | 0: Input off (0V)<br>1: Input on (24V) |
| | | Output | 88 | 0: Output off (0V)<br>1: Output on (24V) |
| DXP1 | | Input | 60 | 0: Input off (0V)<br>1: Input on (24V) |
| | | Output | 89 | 0: Output off (0V)<br>1: Output on (24V) |
| DXP2 | C1 | Input | 57 | 0: Input off (0V)<br>1: Input on (24V) |
| | | Output | 86 | 0: Output off (0V)<br>1: Output on (24V) |
| DXP3 | | Input | 58 | 0: Input off (0V)<br>1: Input on (24V) |
| | | Output | 87 | 0: Output off (0V)<br>1: Output on (24V) |

#### Setting the switchable VAUX power supply

| Socket | Type | GPIO | Possible values |
|---|---|---|---|
| C0 | Output VAUX0 | 41 | 0: VAUX off<br>1: VAUX on |
| | Output VAUX1 | 40 | |

#### Setting the switchable VAUX power supply – diagnostics

| Socket | Type | GPIO | Possible values |
|---|---|---|---|
| C0 | Input VAUX0 | 56 | 0: VAUX error-free<br>1: Error or overvoltage on VAUX |
| | Input VAUX1 | 55 | |

### 7.6.2 Setting DXP functions via script

A script is installed on the device for setting the DXP channels. The script is located under the following path:

/usr/bin/dxp

The script can be used with the following syntax:

```
/usr/bin/dxp channel [value]
```

The following example sets the value for the DXP0 channel to ON.

```
/usr/bin/dxp 0 1
```

| Parameter | Possible values |
|---|---|
| DXP0…DXP3 | 1: Switch on channel<br>0: Switch off channel |

### 7.6.3 Programming DXP channels with Python 3

**NOTE**
The speed of the data transmission depends on the configured block size and the set transfer rate. The speed may possibly not be enough for time critical applications. To achieve faster data processing, the process can be set as a real-time process.

The following example shows the programming of the digital I/O channels with Python 3.

```
import sys
#GPIOs-> OUT: IN:
ports = ["88","59"]

# write GPIO:
try:
    # set direction to write DXP
    fo = open("/sys/class/gpio/gpio" + ports[0] +"/direction", "w")
    fo.write("out")
    fo.close()
    # write GPIO:
    f = open("/sys/class/gpio/gpio" + ports[0] +"/value", "w")
    f.write("1")
    f.close()
except:
    # export gpio if not done as yet
    f1 = open("/sys/class/gpio/export", "w")
    f1.write(ports[0]) f1.close()
    # set direction to write DXP
    fo = open("/sys/class/gpio/gpio" + ports[0] +"/direction", "w")
    fo.write("out")
    fo.close()
    # write GPIO:
    fw = open("/sys/class/gpio/gpio" + ports[0] +"/value", "w")
    fw.write("1")
    fw.close()
```

```
# read GPIO:
try:
    # set direction to read DXP
    fo = open("/sys/class/gpio/gpio" + ports[1] +"/direction", "w")
    fo.write("in")
    fo.close()
    # set active low to get the right value...

    fal.write("1")
    fal.close()
    # read DXP: fr = open("/sys/class/gpio/gpio" + ports[1] +"/
value", "r")
    val=fr.read()
    fr.close()
    print(val)
except:
    # export gpio if not done as yet
    f1 = open("/sys/class/gpio/export", "w")
    f1.write(ports[1])
    f1.close()
    # set direction to read DXP
    fo = open("/sys/class/gpio/gpio" + ports[1] +"/direction", "w")
    fo.write("in")
    fo.close()
    # set active low to get the right value...

    fal.write("1")
    fal.close()
    # read DXP:
    fr = open("/sys/class/gpio/gpio" + ports[1] +"/value", "r")
    val=fr.read()
    fr.close()
    print(val)
```

### 7.6.4 Programming DXP channels with Node.js

**NOTE**
The speed of the data transmission depends on the configured block size and the set transfer rate. The speed may possibly not be enough for time critical applications. To achieve faster data processing, the process can be set as a real-time process.

The following examples illustrate the programming of the digital I/O channels with Node.js. Further information on Node.js and the Node.js packages is provided at:

- **https://nodejs.org**
- **https://www.npmjs.com/**

```
// initialize the onoff box
const Gpio = require('onoff').Gpio;

function setGpioByInt(OUT, val) {
// switch from DXP to GPIO...
switch (OUT) {
    case 0:
        res = 88;
        break;
    case 1:
        res = 89;
        break;
    case 2:
        res = 86;
        break;
    case 3:
        res = 87;
        break;
}
        // initialize the GPIO just to write...

        // write the GPIO / DXP...

        console.log('set Gpio '+ res + ' to ' + val);
}
```

```
function getGpio(IN) {
// switch from DXP to GPIO...
switch (IN) {
     case "0":
        res = 59;
        break;
    case "1":
        res = 60;
        break;
    case "2":
        res = 57;
        break;
    case "3":
        res = 58;
        break;
}
    // initialize the GPIO just to read...

    # set active low to get the right value...

    // read the GPIO / DXP...

     ' + res);
    return res;
}
```

### 7.6.5 Programming DXP channels with C or C++

The following example shows the programming of the digital I/O channels with Ansi C/C++.

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>

// initialize function (use extern for C++)
int access(const char *pfad, int modus);

int main(void) {
     //choose DXP / GPIO for connection



    char input[2];
/*
    ================================================================
==============
    READ:

================================================================
==========*/

    {
        // file doesn't exist!
            // export gpio...
            if((fh = fopen("/sys/class/gpio/export", "w")) !=
            {
                fputs("59",fh);
                fclose(fh);
            }
            else
            {
                printf("failed on export to read...\n");
                printf("result: %i \n", (int)fh);
                return -1;
            }
```

```
    }
    // set direction to read...
    if((fh =fopen("/sys/class/gpio/gpio59/direction", "w")) !=
    {
        fputs("in",fh);
        fclose(fh);
    }
    else
    {
         printf("failed on setting direction to read...\n");
        return -1;
    }

    // set active low to read...

    {
        fputs("1",fh);
        fclose(fh);
    }
    else
    {
        printf("failed on setting active low ...\n");
        return -1;
    }
    // read GPIO...
    if((fh = fopen("/sys/class/gpio/gpio59/value", "r")) !=
    {
        fgets(input,2,fh);
        fclose(fh);
        printf("Value: %c\n", input[0]);
    }
     else
    {
        printf("failed on reading ...\n");
        return -1;
    }
/*
```

```
==========================================================================
==========
    WRITE:
    ==========================================================================
==============*/

    {
        // file doesn't exist
            // export gpio...
            if((fh = fopen("/sys/class/gpio/export", "w")) !=
            {
                fputs("88",fh);
                fclose(fh);
            }
            else
            {
                printf("failed on export to write...\n");
                printf("result: %i \n", (int)fh);
                return -1;
        }
    }
    // set direction to read...
    if((fh = fopen("/sys/class/gpio/gpio88/direction", "w")) !=
    {
            fputs("out",fh);
            fclose(fh);
    }
    else
    {
         printf("failed on setting direction to write...\n");
        return -1;
    }

    // write GPIO...
    if((fh = fopen("/sys/class/gpio/gpio88/value", "w")) !=
    {
        fputs((const char*)"1",fh);
```

```
            fclose(fh);
        }
         else
        {
            printf("failed on writing ...\n");
            return -1;
        }

}
```

## 7.7 Programming LED functions

### 7.7.1 LEDs – overview

The device is provided with three freely programmable LEDs. The LEDs can be programmed individually via read and write commands. The LEDs are mapped on the system under the following path: „/sys/class/leds/...“

| LED | Color | System name |
|---|---|---|
| APPL | Green | appl_green |
| | Red | appl_red |
| ERR | Green | err_green |
| | Red | err_red |
| RUN | Green | run_green |
| | Red | run_red |

If the red and green of an LED are switched on at the same time, the LED is orange.

**NOTE**
During an ongoing firmware update the RUN-LED is used by the system.

### 7.7.2 Setting LED functions via a script

A script is installed on the device for setting the LEDs. The script is located under the following path:

/TURCK/scripts/led.sh

The script can be used with the following syntax:

```
sh led.h led color [value]
```

The following example switches on the red APPL LED:

```
sh led.sh appl red 1
```

| LED | Possible color setting | Possible values |
|---|---|---|
| ERR | Green/red | 1: Switch on LED<br>0: Switch off LED |
| RUN | Green/red | 1: Switch on LED<br>0: Switch off LED |
| APPL | Green/red | 1: Switch on LED<br>0: Switch off LED |

### 7.7.3 Programming LED functions with Python 3

The following example shows the programming of the LED functions with Python 3:

```
import sys
import time
# write red LEDs:
fw = open("/sys/class/leds/run_red/brightness", "w")
fw.write("1")
fw.close()

fw = open("/sys/class/leds/appl_red/brightness", "w")
fw.write("1")
fw.close()

fw = open("/sys/class/leds/err_red/brightness", "w")
fw.write("1")
fw.close()

# Wait for 5 seconds
time.sleep(5)

# write green LEDs:
fw = open("/sys/class/leds/appl_green/brightness", "w")
fw.write("1")
fw.close()

fw = open("/sys/class/leds/err_green/brightness", "w")
fw.write("1")
fw.close()

fw = open("/sys/class/leds/run_green/brightness", "w")
fw.write("1")
fw.close()

# Wait for 5 seconds
time.sleep(5)
```

```
# clean red LEDs:
fw = open("/sys/class/leds/run_red/brightness", "w")
fw.write("0")
fw.close()

fw = open("/sys/class/leds/appl_red/brightness", "w")
fw.write("0")
fw.close()

fw = open("/sys/class/leds/err_red/brightness", "w")
fw.write("0")
fw.close()

# Wait for 5 seconds
time.sleep(5)

# clean green LEDs:
fw = open("/sys/class/leds/appl_green/brightness", "w")
fw.write("0")
fw.close()

fw = open("/sys/class/leds/err_green/brightness", "w")
fw.write("0")
fw.close()

fw = open("/sys/class/leds/run_green/brightness", "w")
fw.write("0")
fw.close()
```

### 7.7.4 Programming LED functions with Node.js

The following examples illustrate the programming of the LED functions with Node.js. Further information on Node.js and the Node.js packages is provided at:

- **https://nodejs.org**
- **https://www.npmjs.com/**

```
// initialize the onoff box
const Gpio = require('onoff').Gpio;

//initialize the leds which are free for the user

appl_green_led = new LED('appl_green');
appl_red_led = new LED('appl_red');
error_green_led = new LED('err_green');
error_red_led = new LED('err_red');
run_green_led = new LED('run_green');
run_red_led = new LED('run_red');
```

### 7.7.5 Programming LED functions with C or C++

The following example shows the programming of the LED functions with Ansi C/C++.

```
#include <stdio.h>
#include <stdlib.h>

// initialize function (use extern for C++)
char* strcpy(char* target, const char* source);
char* strcat(char* s1, const char* s2);

int main(void) {
   // LEDs for the customer:
   char *appl_green_led = "appl_green";
   char *appl_red_led = "appl_red";
   char *error_green_led = "err_green";
   char *error_red_led = "err_red";
   char *run_green_led = "run_green";
   char *run_red_led = "run_red";
   char *LED_FILE = "/sys/class/leds/";
   char *brightness = "/brightness";
   FILE *fh;

   char cur_Str[50] = {0};
   strcpy(cur_Str, LED_FILE);
   // take LED which will shine:
   strcat(cur_Str, run_red_led);
   strcat(cur_Str, brightness);

   //WRITE:
   printf("string to led: %s\n", cur_Str);

   // write LED...
   if((fh = fopen(cur_Str, "w")) != 0)
   {
      // write "1" to switch on and "0" to switch of LED
      fputs((const char*)"0",fh);
      fclose(fh);
```

```
    }
    else
    {
        printf("failed on writing ...\n");
        return -1;
    }
    return EXIT_SUCCESS;
}
```

## 7.8 Creating a C application

### Requirements

The following components are required to create a C application:

- Toolchain for Cortex A8
- C program

### Downloading a toolchain

The following toolchain is required for the Cortex A8 processor in order to cross compile a C program:

- OSELAS.toolchain-2014.12.0-arm-cortexa8-linux-gnueabihf-gcc-4.9.2-glibc-2.20-binutils-2.24-kernel-3.16-sanitized

The toolchain is available to download at **http://debian.pengutronix.de/debian**.

### Example: creating the C program

- ▶ Create the “hello.c” file.
- ▶ Copy the following text to the file:

```
// hello.c
#include <stdio.h>

int main() {
   printf("Hello World!\n");
   return 0;
}
```

- ▶ Create executable file with the following toolchain command:
  `/opt/OSELAS.Toolchain-2014.12.0/arm-cortexa8-linux-gnueabihf/gcc-4.9.2-glibc-2.20-binutils-2.24-kernel-3.16-sanitized/bin/arm-cortexa8-linux-gnueabihf-gcc -o helloExample hello.c`

## Example: creating a C program via a make file

The “make” service program automates the creation of executable files from source code. C programs can be compiled via “make”. This uses a make file which contains the rules for creating executable files.

The following example shows a simple make file:

```
all: helloExample


helloExample: hello.o

    /opt/OSELAS.Toolchain-2014.12.0/arm-cortexa8-linux-gnueabihf/
gcc-4.9.2-glibc-2.20-binutils-2.24-kernel-3.16-sanitized/bin/arm-
cortexa8-linux-gnueabihf-gcc -o helloExample hello.o


hello.o: hello.c

   /opt/OSELAS.Toolchain-2014.12.0/arm-cortexa8-linux-gnueabihf/
gcc-4.9.2-glibc-2.20-binutils-2.24-kernel-3.16-sanitized/bin/arm-
cortexa8-linux-gnueabihf-gcc -c hello.c


clean:

   rm hello.o helloExample

}
```

- ▶ Create a make file.
- ▶ Save a make file in the same folder as the C application.
- ▶ Execute the make file with the “make” command.
- ⇨ The C program is installed.

## 7.9 Starting the application automatically (autostart)

An application can be executed automatically with the autostart function after the RFID interface is started. For this a configuration file (unit file) must be created, written to the device and activated

### 7.9.1 Autostart – creating the configuration file (unit file)

- Create a unit file with the suffix "service".

Example: The ".setdxp.service" unit file starts a Node.js application, by which the DXP channels are triggered with every restart.

- Call up via "ExecStart" the application to be called every time the interface is restarted:
  `ExecStart=path_to_programm app/file`

A parameter can be transferred if required. Example: `ExecStart=path_to_programm app/file parameter`

Further information on unit configuration files is available at:
**https://www.freedesktop.org/software/systemd/man/systemd.service.html**

#### Example: autostart of an application with a parameter transfer

```
ExecStart=/usr/bin/node /home/user/ hello_GPIO.js webactive
```

### 7.9.2 Example: using the unit file

The following example downloads the Node.js file "hello_GPIO.js" and stores it at "/home/user":

```
[Unit]

Description= trigger the DXPs

#After=Service_that_must_run_before.service


[Service] Type=simpleExecStart=/usr/bin/node /home/user/
hello_GPIO.js


[Install]

WantedBy=multi-user.target
```

- Creating an example file "./etc/systemd/system/":
  `sudo touch /etc/systemd/system/setdxp.service`
- Open the created file:
  `sudo nano /etc/systemd/system/setdxp.service`
- Insert the source text shown above in the opened file.

### 7.9.3 Activating the unit file

After being created, the unit file must be activated via the systemctl command. Access rights to the root directory are required to activate. The .services file suffix is optional and can be omitted.

- Activate the unit file via the following command:
  `sudo systemctl enable setdxp.service`

The created symlink is:
`/etc/systemd/system/multi-user.target.wants/setdxp.service â /etc/systemd/system/setdxp.service`

#### Deactivating the unit file

- Deactivate the unit file with the following command:
  `sudo systemctl disable setdxp.service`

## 7.10 Managing access rights

The device supports the standard Linux user management. The access rights can be managed with the following standard Linux tools:

- adduser
- addgroup
- passwd

| User | Rights | Password |
|---|---|---|
| root | System administrator (all access rights) | turck |
| user | Restricted access rights and console rights | password |
| sftpuser | Access rights, SFTP rights in the directory /home | password |

## 7.11 Installing Python packages

Modules, libraries and other software can be configured via the BSP (Board Support Package) with the PTXdist distribution tool and loaded on the device. If packages are to be integrated in an existing firmware, they must be created beforehand with PTXdist. PTXdist is available for download at **https://www.pengutronix.de/de/software/ptxdist.html**.

The ipkg package manager (Itsy Package Management System) is installed on the device for integrating software packages. The ipkg package manager makes it possible to also install Python modules at a later time.

### 7.11.1 Example: installing the Python module

The following example explains the procedure for the installation of the Python sh module. The Python module will be integrated at a later time in an existing firmware.

#### Requirements

- PTXdist is installed on the Linux host system.
- The required Python module was downloaded (example: **https://amoffat.github.io/sh/**).

#### Example: installing the Python sh module

In order to create the Python sh module, a rule file must be created first.

- ▶ Create the rule file with the following command:
  ```
  $ ptxdist newpackage target
  ```
- ▶ Create interactive information on the package:Output:
  ```
  Output:
  ptxdist: creating a new 'target' package:
  ptxdist: enter package name...........:
  sh ptxdist: enter version number......:
  1.12.13 ptxdist: enter URL of basedir.:https://github.com/
  amoffat/sh/archive/
  ptxdist: enter suffix.................: tar.gz
  ptxdist: enter package author.........: Your Name <E-Mail>
  ptxdist: enter package section........: Python3


  generating rules/sh.make
  generating rules/sh.in
  ```

The sh.make and sh.in files are created automatically.

- If known, enter the key of the package as an SH_MD5 parameter in the sh.make file.
- Set the SH_CONF_TOOL parameter in the sh.make file to the appropriate tool (in this case: Python 3).
  ```
  SH_CONF_TOOL :=python3
  ```
- If a Python module has a separate subfolder: Create the subfolder in the target directory (in this case not required):
  ```
  @$(call install_copy, module, 0, 0, 0755, $(PYTHON3_SITEPACK-
  AGES)/foldername)
  ```
- In the #Target-Install area, specify the installation location of the Python module in the target system (Example: sh module):
  ```
  @for file in `find $(SH_PKGDIR)/usr/lib/python$(PYTHON3_MA-
  JORMINOR)/site-packages \
      ! -type d ! -name "*.py" -printf "%P\n"`; do \
    $(call install_copy, sh, 0, 0, 0644, -, \
      /usr/lib/python$(PYTHON3_MAJORMINOR)/site-packages/$
  $file); \
  done
  ```

Dependencies can be entered in the sh.in file. Python 3 must be available in the following example in order to install Python modules. The “setuptools” module must be available on the host system.

- Enter dependencies as follow:
  ```
  ## SECTION=python3
  config PYTHON_SH
    tristate
    select PYTHON3                  # Python 3 must be installed
    select HOST_PYTHON3_SETUPTOOLS # Setuptools must be in-
  stalled on the host
    prompt "sh"
    help
      FIXME
  ```
- Compile.

In order for the sh module to be created with the next build, the module must be selected in “menuconfig”:

- „menuconfig“ über folgenden Befehl öffnen:
  `ptxdist menuconfig`
- Navigate to the Python 3 modules via “Scripting Languages” → “python3 Extra Modules”.
- Select the sh module.
- Save the configuration.

Fig. 56: PTXdist – “Python 3 Extra Modules”

- Generate ipkg packages with the following command:
  `ptxdist go`

⇨ If no errors have occurred, the package with the sh module can be found at “platform-tben-lx-linux/packages/”:

```
$ ls platform-tben-lx-linux/packages/
...
python3_3.5.0_armhf.ipk
sh_1.12.14_armhf.ipk
...
```

- Copy the ipk file to the TBEN device (e.g. with scp):

```
scp ~/turck/TBEN-Lx-4RFID-8DXP-LNX/platform-tben-lx-linux/
packages/sh_1.12.14_armhf.ipk
root@Target-IP:/directory/of/your/choice/
```

- Log into the TBEN device in order to install the “sh_1.12.14_armhf.ipk” package.
- Call up the ipkg manager to install the Python module:
  `ipkg -force-depends install sh_1.12.14_armhf.ipk`

⇨ The module is available in the Python Interpreter.

# 8 Operation

## 8.1 LEDs

The device has the following LED indicators:

- Power supply
- Group and bus errors
- Status
- Diagnostics

| PWR LED | Meaning |
|---|---|
| Off | No power supply |
| Green | Power supply error-free |
| Yellow | Undervoltage within tolerance range |
| Red | Undervoltage outside of tolerance range |

| RFON LED | Meaning |
|---|---|
| Off | Wireless field deactivated |
| Green | Wireless field activated |

| DATA LED | Meaning |
|---|---|
| Off | No tag in the field, no data transfer |
| Yellow flashing | Tag in the field, data transfer via the air interface |

| DIAG LED | Meaning |
|---|---|
| Off | No error |
| Red | Error |

The following multicolor LEDs are freely programmable. The tables below describe the default display functions.

| DXP LEDs (digital channels, LEDs DXP0...3) | | |
|---|---|---|
| **LED green** | **LED red** | **Meaning** |
| Off | Off | No I/O signal present |
| Lit | Off | I/O signal present |
| Off | Lit | Overload at output |
| Flashing | Flashing | Overload of the auxiliary voltage |

| APPL LED | Meaning |
|---|---|
| Flashing white | Wink command active |


Hans Turck GmbH & Co. KG | T +49 208 4952-0 | F +49 208 4952-264 | more@turck.com | www.turck.com

# 9 Troubleshooting

If the device does not work as expected, proceed as follows:

- Exclude environmental disturbances.
- Check the connections of the device for errors.
- Check device for parameterization errors.

If the malfunction persists, the device is faulty. In this case, decommission the device and replace it with a new device of the same type.

## 9.1 Rectifying errors

Errors are displayed by an ERR LED lit red on the device.

### Calling error messages in the DTM and rectifying them

**NOTE**
Contact Turck if the error persists after the reader is reset.

- Right-click the device in the project tree.
- Select **Diagnostics** in the context menu.

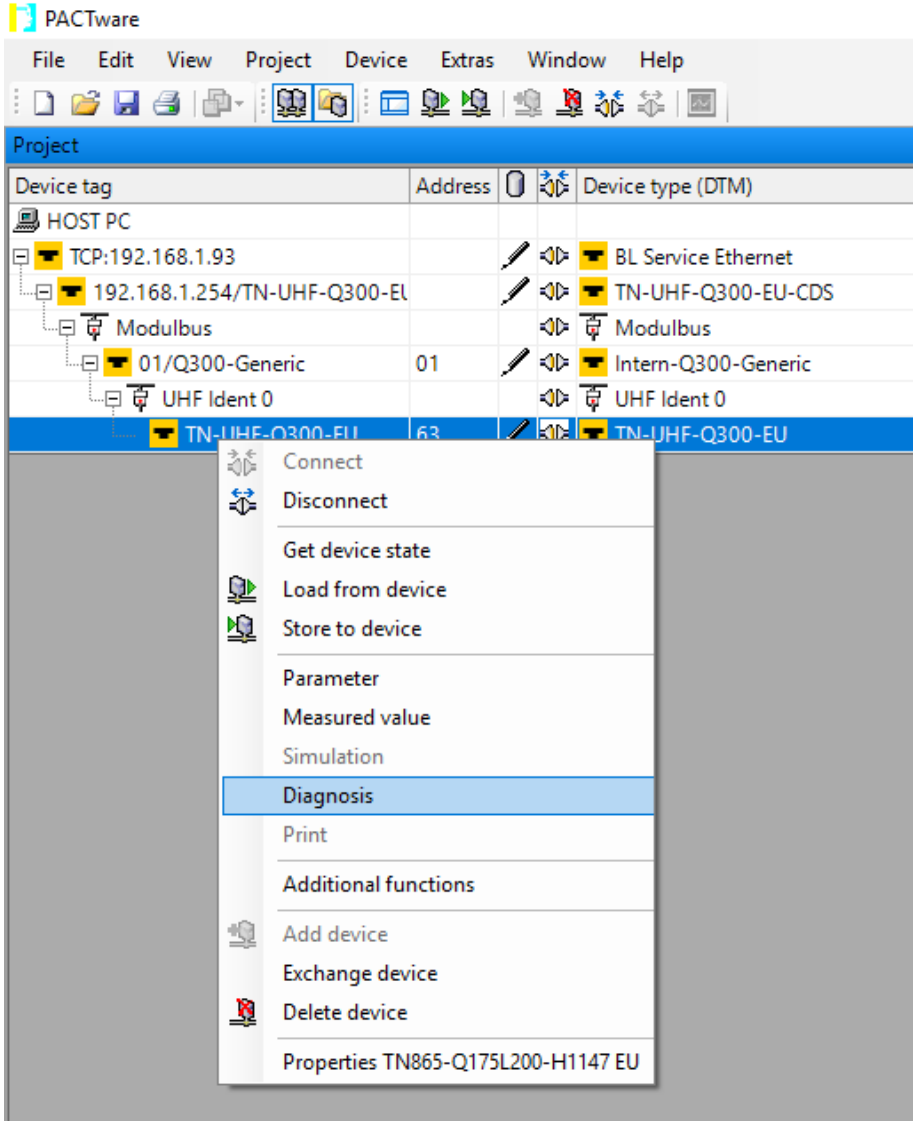



Fig. 57: Project tree – starting diagnosis


Hans Turck GmbH & Co. KG | T +49 208 4952-0 | F +49 208 4952-264 | more@turck.com | www.turck.com

⇨ The diagnostic window is shown in the DTM.

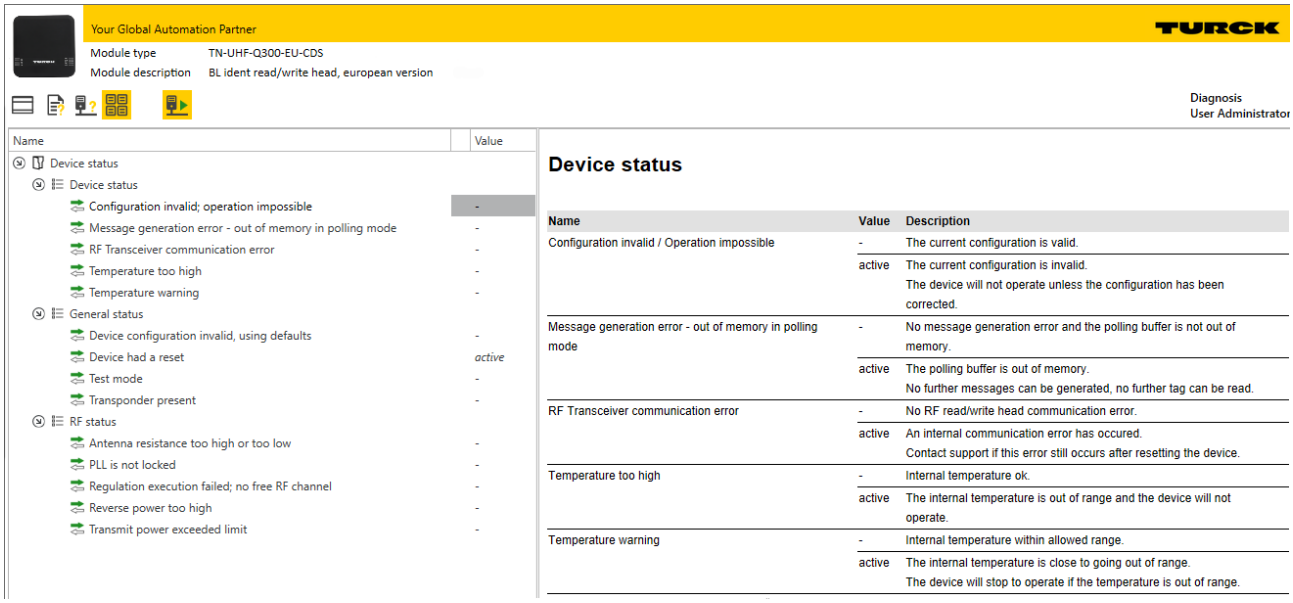



Fig. 58: DTM – diagnosis

Rectifying error messages:

- ▶ Click the **Reset read/write head** button in the RFID test main menu.
- ▶ Select **Reset the read/write head** in the drop-down menu.

⇨ The reader is reset.

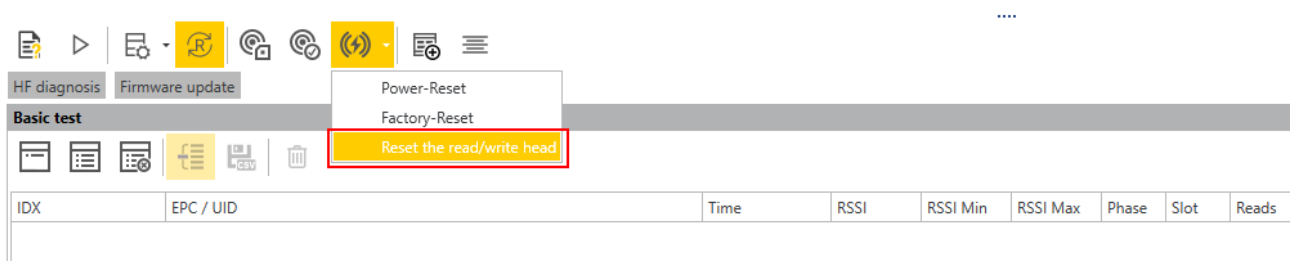



Fig. 59: DTM – resetting read/write head

# 10 Maintenance

## 10.1 Carrying out a Linux update via the web server

The Linux version of the device can be updated via the web server. The current firmware version can be downloaded free of charge from www.turck.com.

**NOTICE**
Interruption of the power supply during the firmware update
**Risk of device damage due to faulty firmware update**
- ▶ Do not interrupt the power supply during the firmware update.
- ▶ During the firmware update do not reset the power supply.

- ▶ Connect the device to a PC via the Ethernet interface.
- ▶ Call the web server of the device: Enter the IP address of the device in the address line of the browser or start the web server via the Turck Service Tool.
- ▶ Click **Software** in the web server.

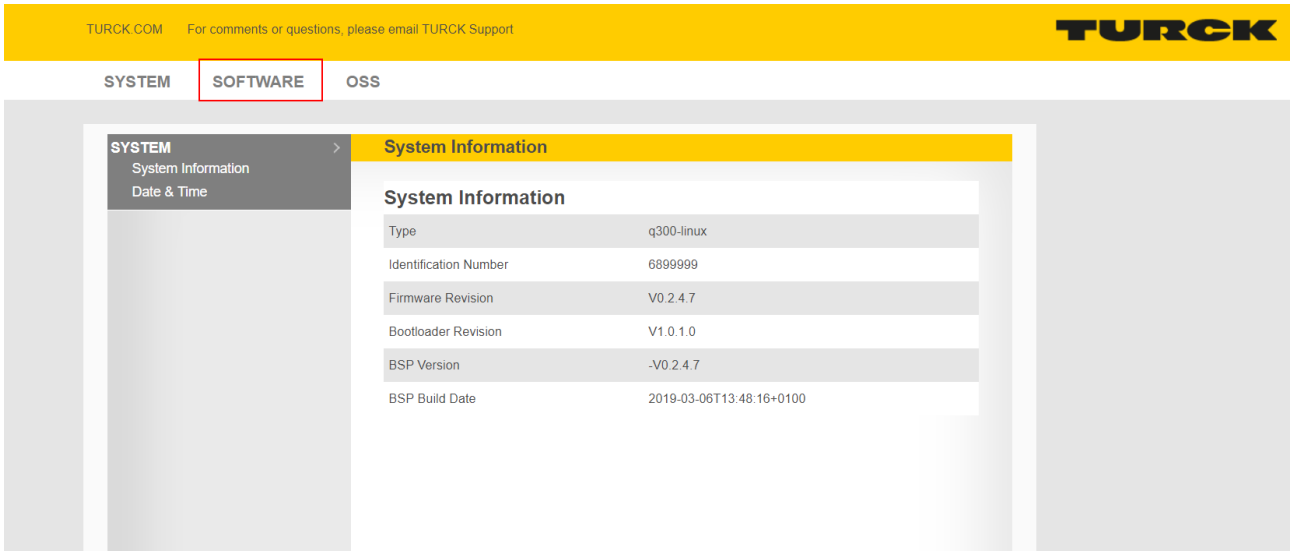



Fig. 60: Web server – selecting the software

▶ Click **File Upload** → **Select File**.

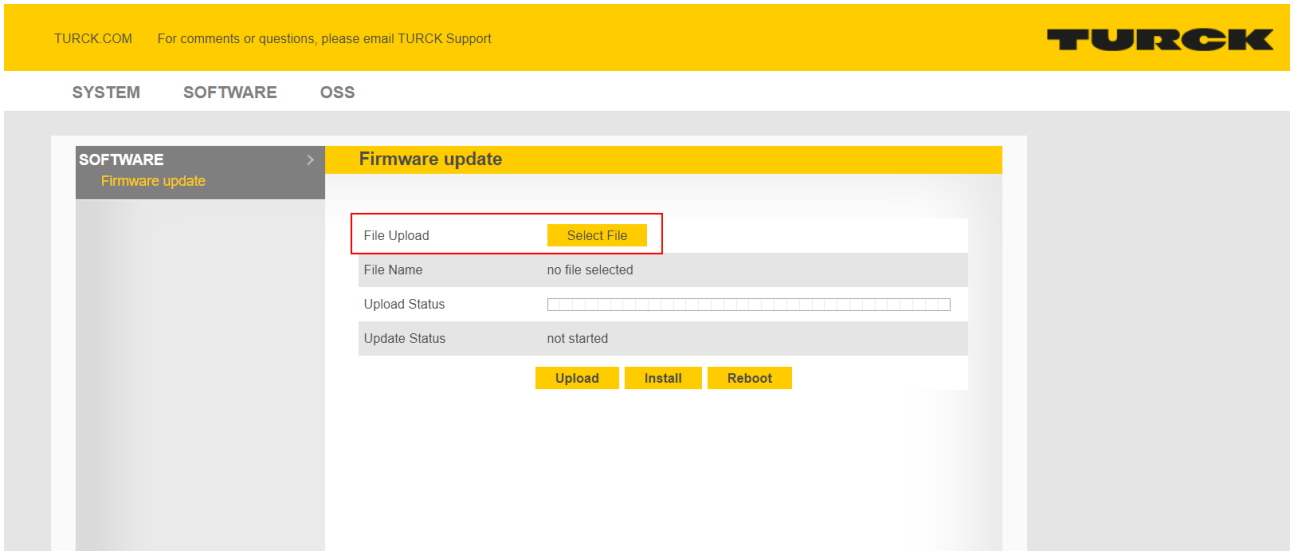



Fig. 61: Web server – upload function

▶ Select the update file on the host PC.

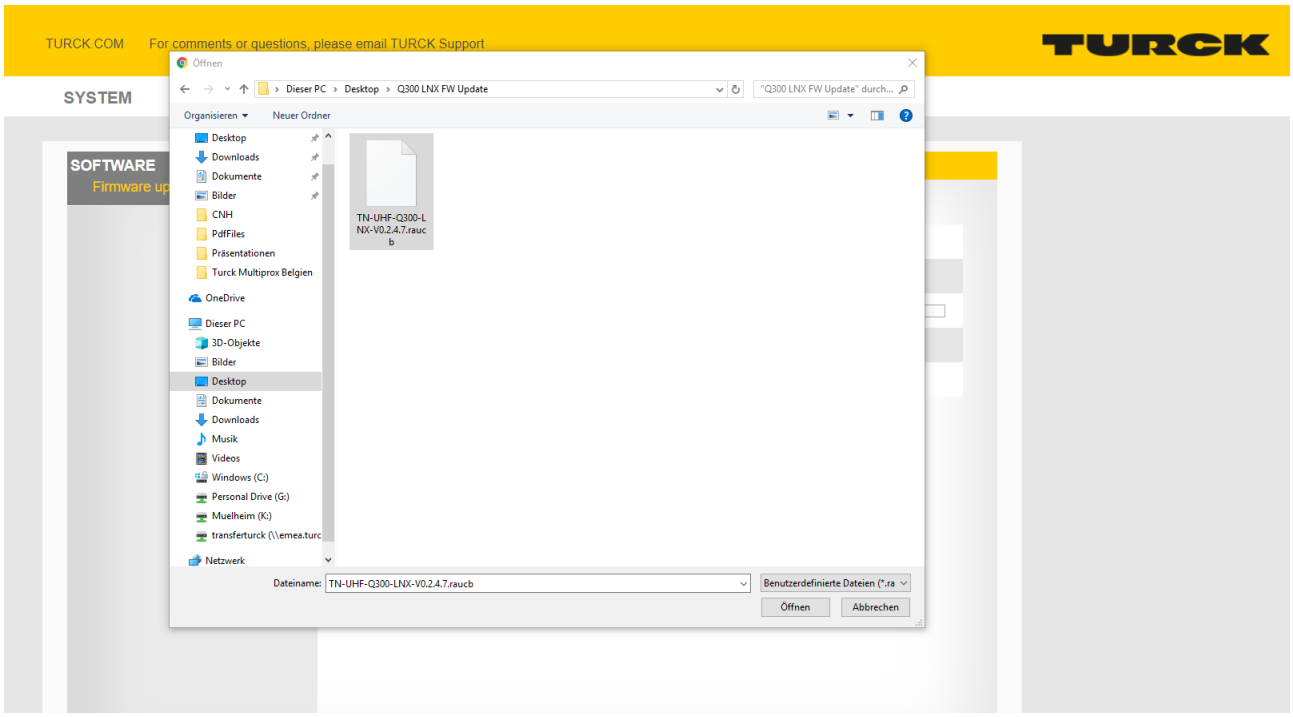



Fig. 62: Selecting the update file

▶ Load the update file on the device: Click **Upload**.

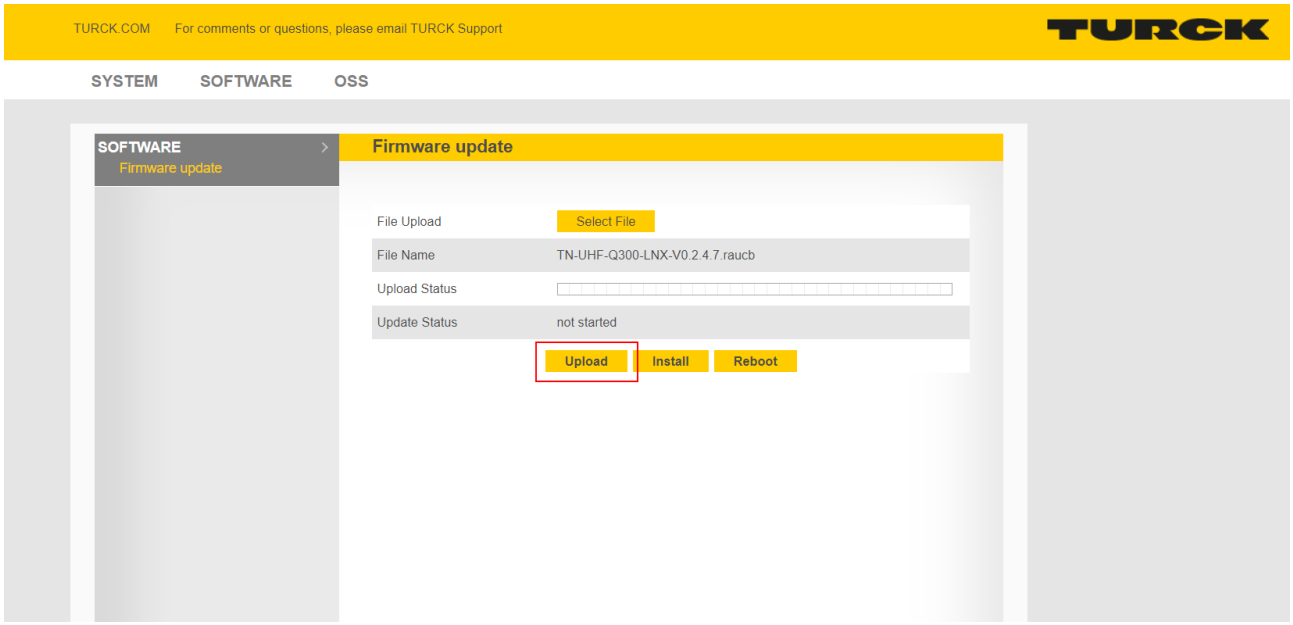



Fig. 63: Web server – loading the update file on the device

▶ Install the update file: Click **Install**.

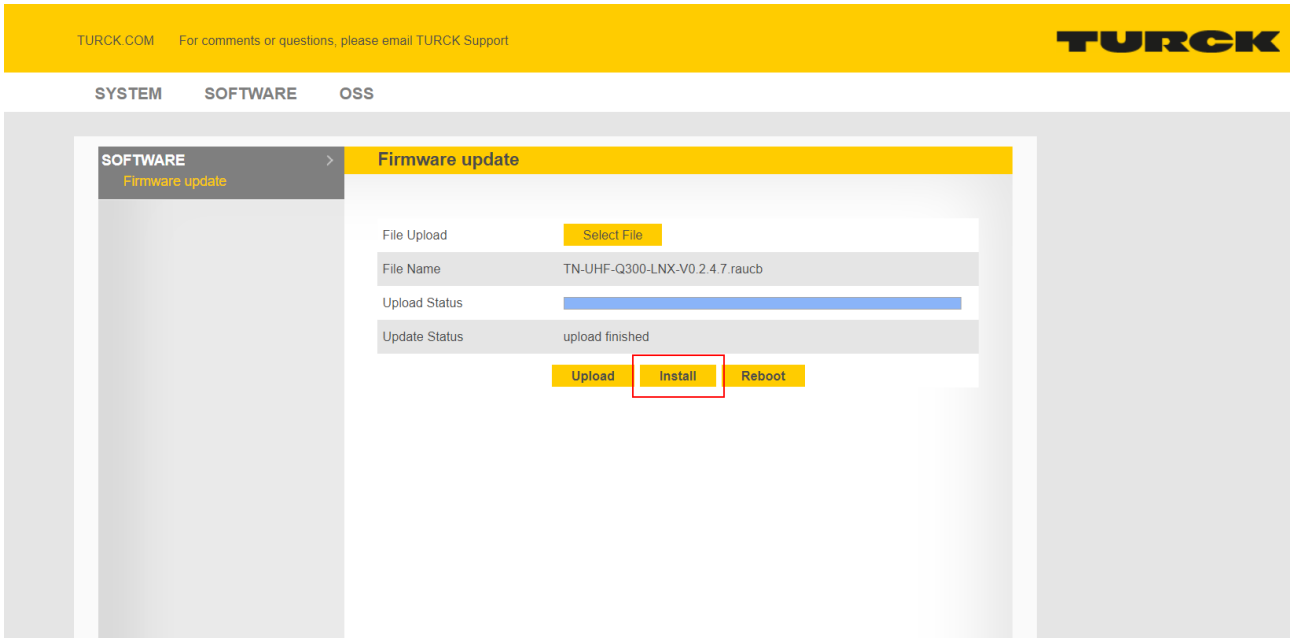



Fig. 64: Web server – installing the update file

▶ Restart the device to complete the update: Click **Reboot**.

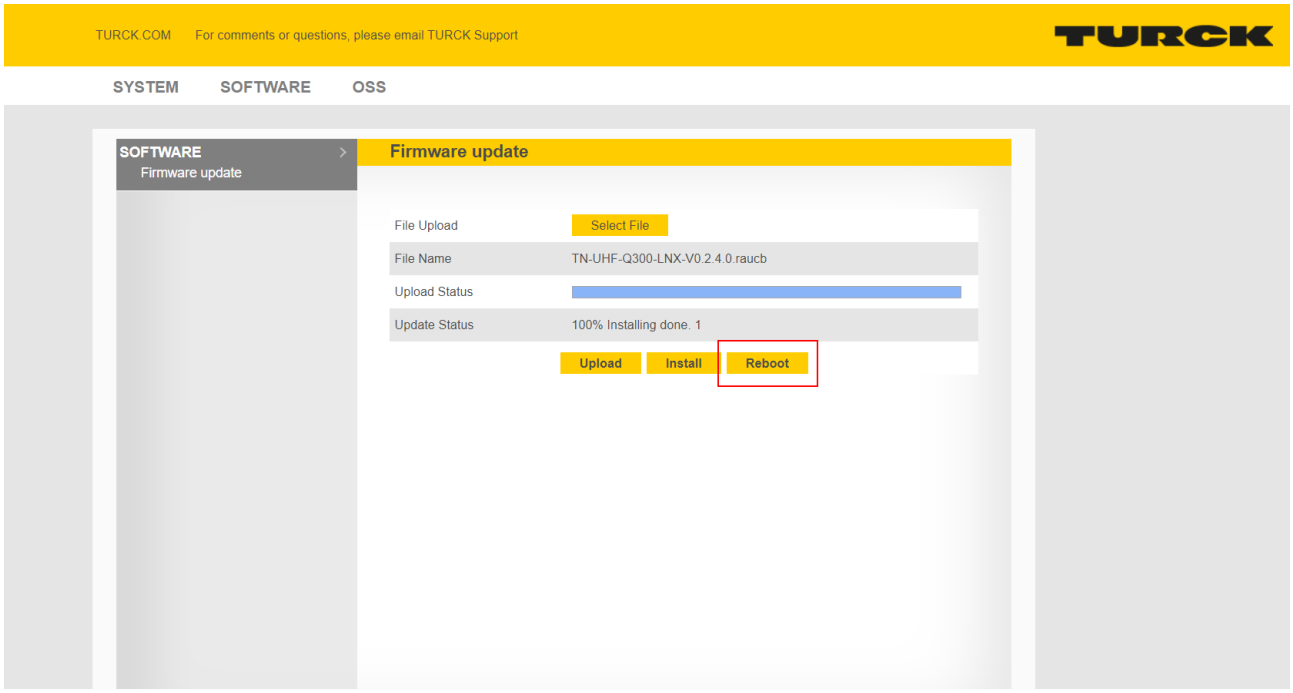



Fig. 65: Restarting the device

## 10.2 Carrying out a UHF update via the DTM

The firmware for the UHF functions of the device can be updated via FDT/DTM. The PACTware FDT frame application, the DTM for the device and the latest firmware are available as a free download from www.turck.com.

**NOTICE**
Interruption of the power supply during the firmware update
**Risk of device damage due to faulty firmware update**
- ▶ Do not interrupt the power supply during the firmware update.
- ▶ During the firmware update do not reset the power supply.

### Example: updating the firmware with the PACTware FDT frame application

- ▶ Launch PACTware.
- ▶ Right-click HOST PC → **Add device**.

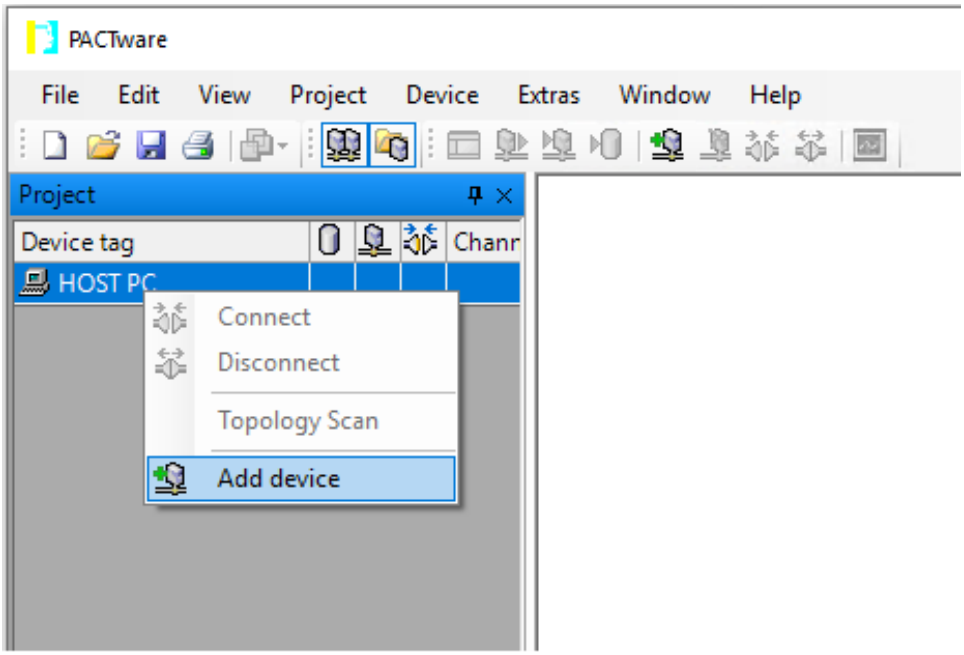



Fig. 66: Adding a device in PACTware

▶ Select **RS485 RFID** and confirm with **OK**.

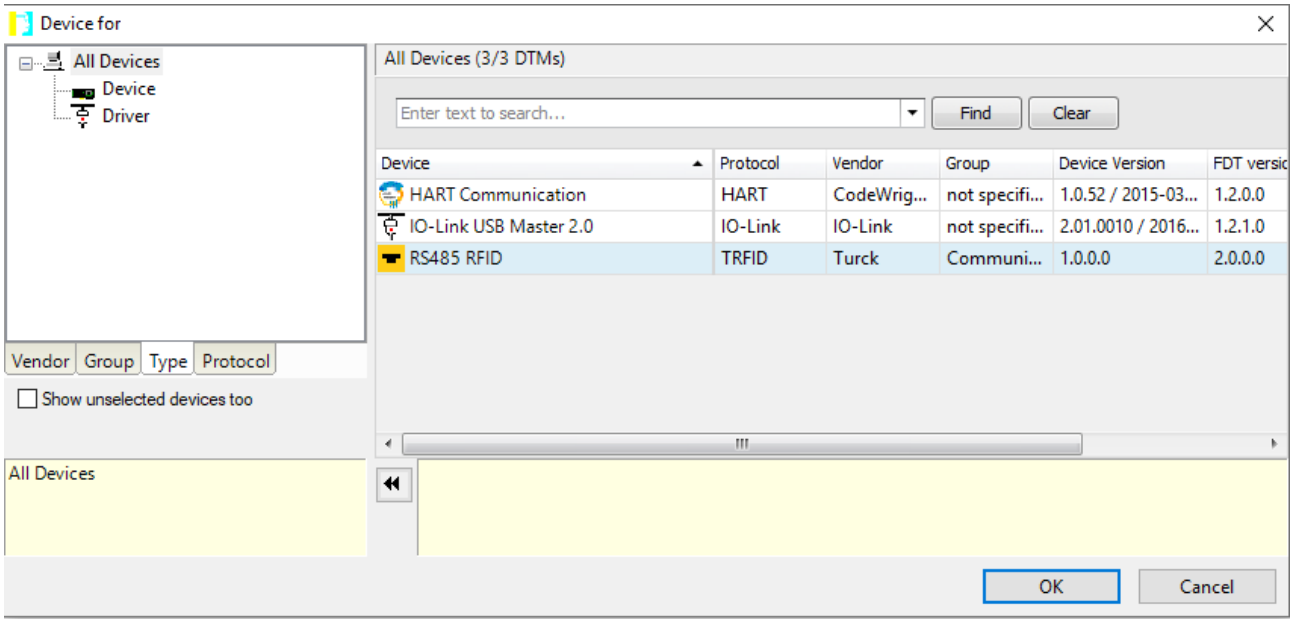



Fig. 67: Selecting the RS485 interface

- ▶ Right-click **RS485 RFID**.
- ▶ Click **Add device** in the context menu.

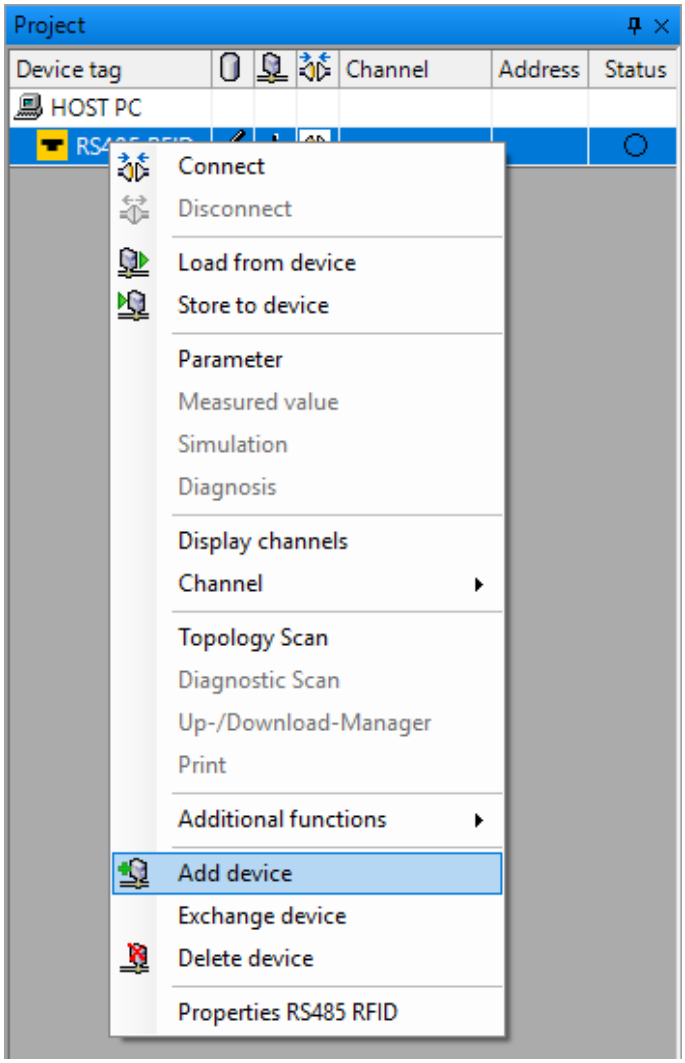



Fig. 68: Add reader

- Select the required reader.

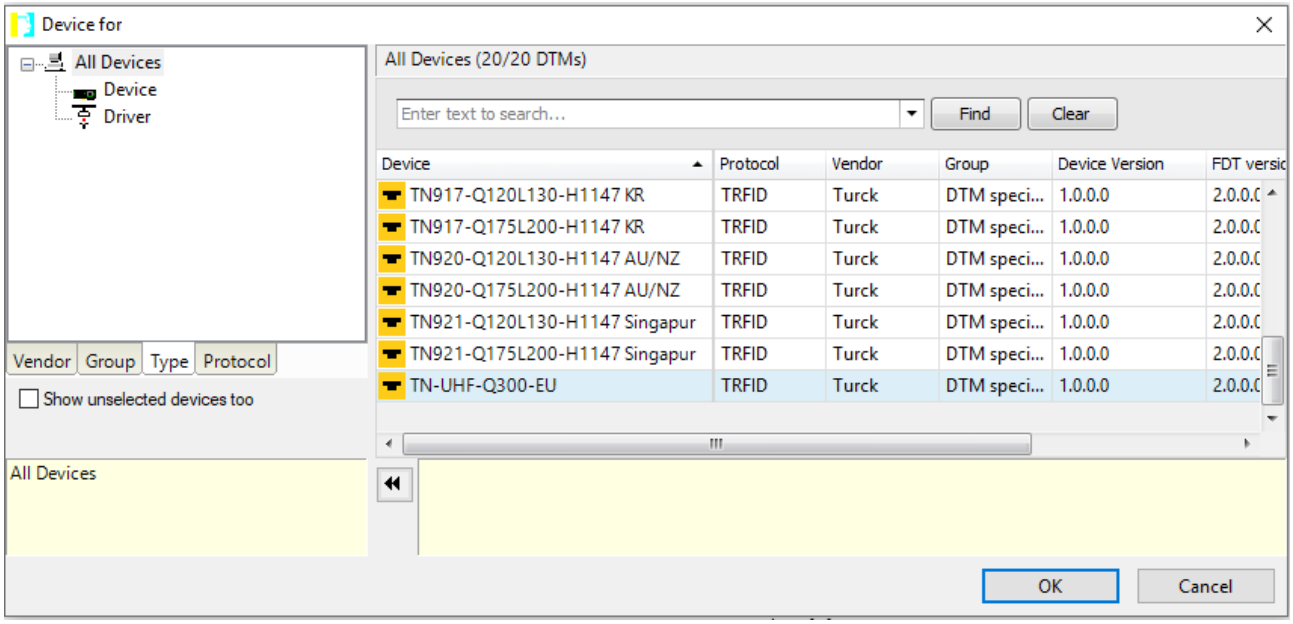



Fig. 69: Example – selecting the reader

- Right-click **TN-UHF-Q300-EU**.
- Establish connection to the host PC: Click **Establish connection** in the context menu.

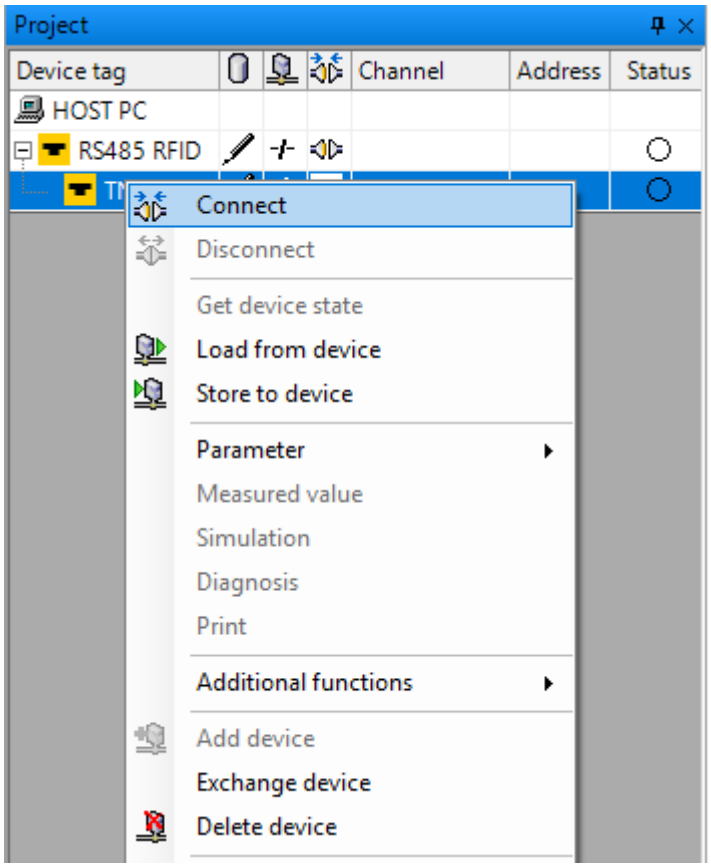



Fig. 70: Establishing a connection

- Right-click **TN-UHF-Q300-EU**.
- In the context menu choose **Other Functions** → **RFID Test**.

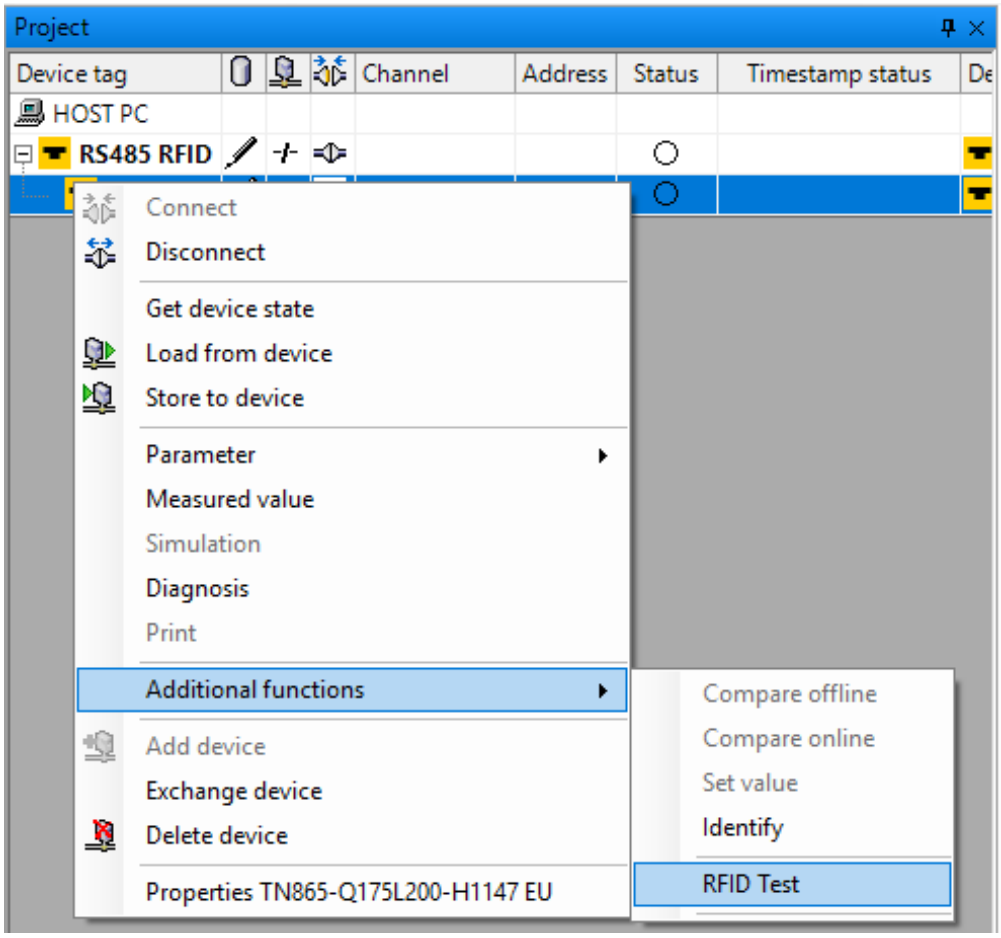



Fig. 71: Starting the RFID test

- Click **Firmware update**.

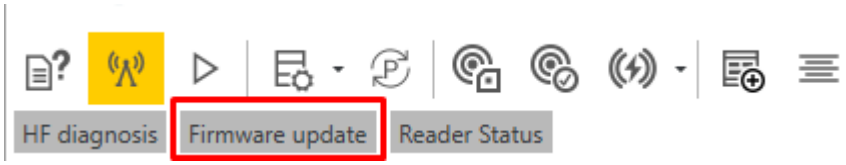



Fig. 72: Firmware update – opening the menu

▶ Click the **Select BL File** button.

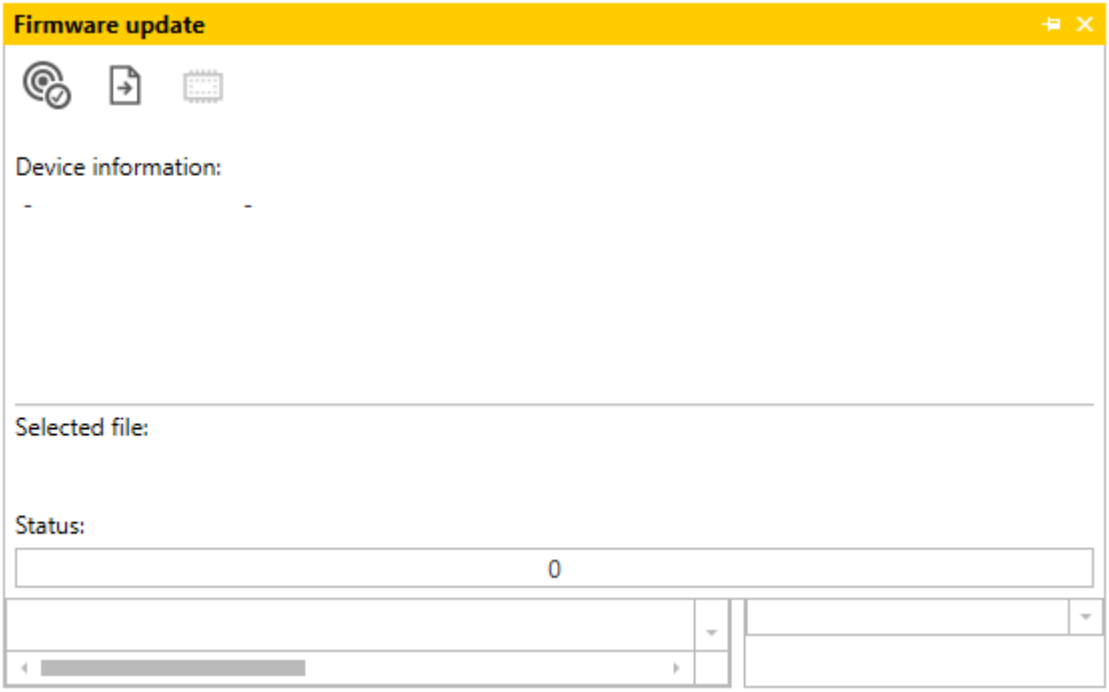



Fig. 73: Button – selecting a BL file

▶ Select the update file from the local memory location.

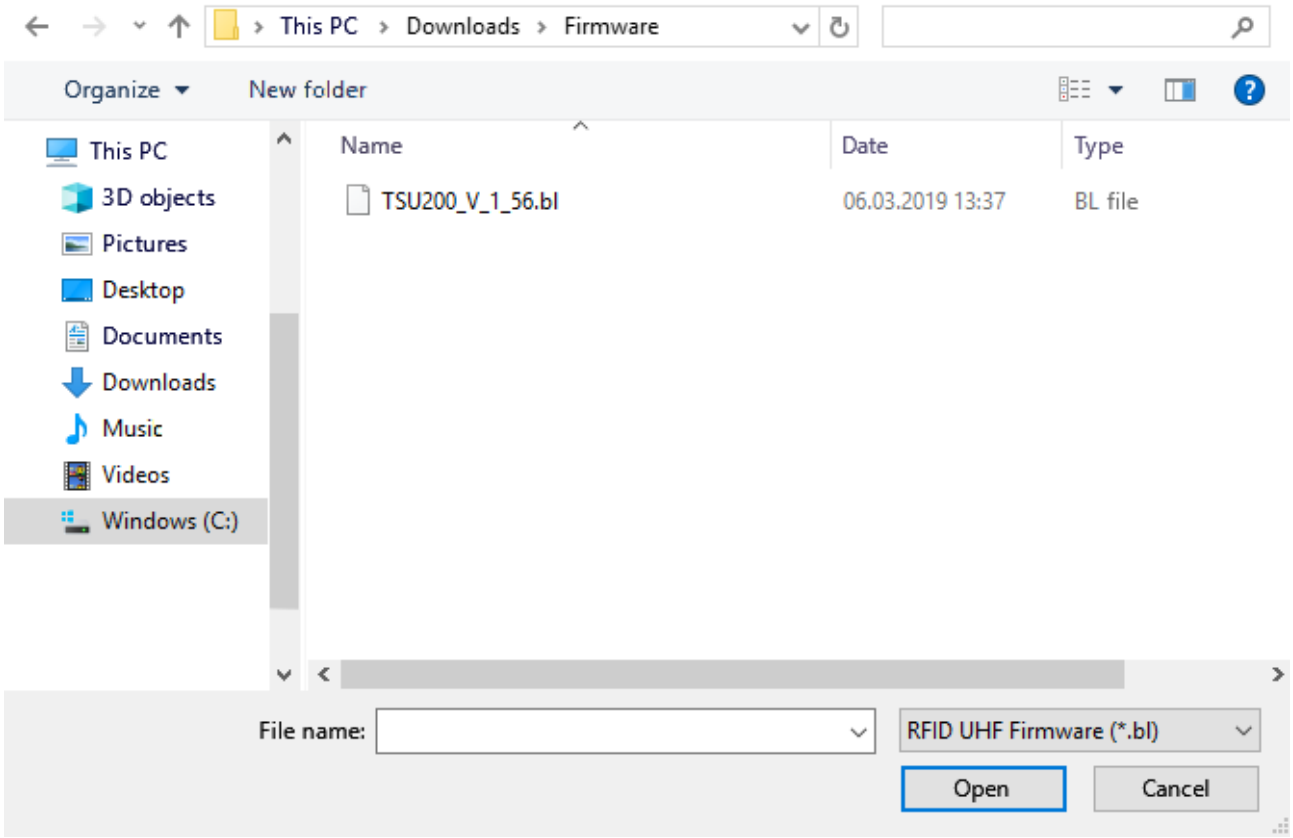



Fig. 74: Local storage of the file


Hans Turck GmbH & Co. KG | T +49 208 4952-0 | F +49 208 4952-264 | more@turck.com | www.turck.com

▶ Start the update: click **Firmware update**.

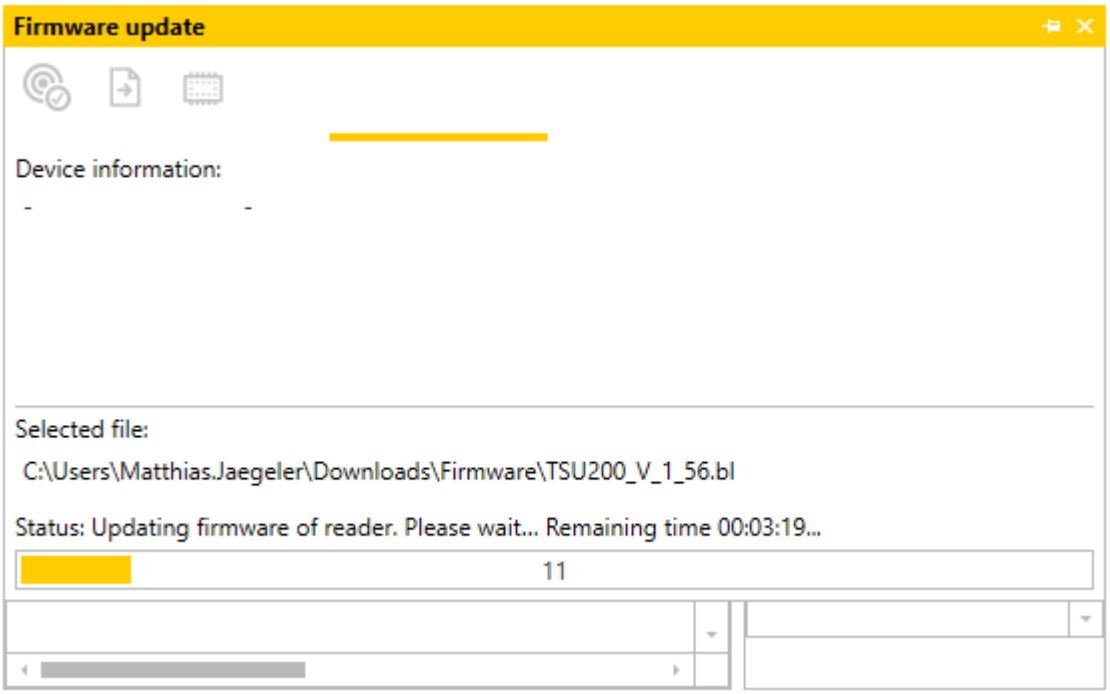



Fig. 75: Starting the firmware update

A completed firmware update is indicated by a status message and a yellow progress bar in the DTM.

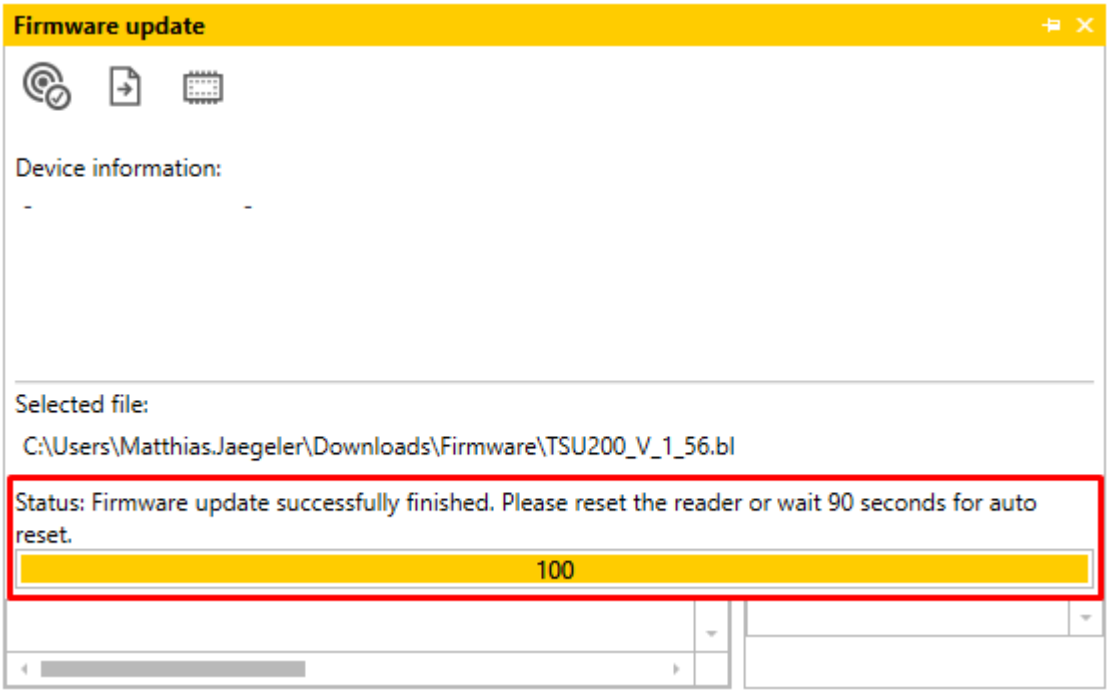



Fig. 76: Successful firmware update

▶ Carry out a reader reset to complete the firmware update.

# 11 Repair

The device must not be repaired by the user. The device must be decommissioned if it is faulty. Observe our return acceptance conditions when returning the device to Turck.

## 11.1 Returning devices

Returns to Turck can only be accepted if the device has been equipped with a Decontamination declaration enclosed. The decontamination declaration can be downloaded from
**https://www.turck.de/en/retoure-service-6079.php**
and must be completely filled in, and affixed securely and weather-proof to the outside of the packaging.

# 12 Disposal

The devices must be disposed of correctly and must not be included in general household garbage.

# 13 Technical Data

| Technical Data | |
|---|---|
| **Electrical data** | |
| Operating voltage | 18…30 VDC |
| DC rated operational current | ≤ 1000 mA |
| Data transmission | Electromagnetic AC field |
| Wireless communication and protocol standards | ISO 18000-6C<br>EN 302208<br>EPCglobal Gen 2 |
| Antenna polarization | Circular/linear, adjustable |
| Antenna half power beam width | 65° |
| Output function | Read/write |
| **Mechanical data** | |
| Mounting requirement | Non-flush |
| Ambient temperature | -20…+50 °C |
| Dimensions | 300 × 300 × 61.7 mm |
| Housing material | Aluminum, AL, silver |
| Material of active face | Fiber glass reinforced polyamide, PA6-GF30, black |
| Vibration resistance | 55 Hz (1 mm) |
| Shock resistance | 30 g (11 ms) |
| Protection class | IP67 |
| No. of channels | 4 |
| Electrical connection | RP-TNC |
| Input impedance | 50 Ω |
| **System description** | |
| Processor | ARM Cortex A8, 32-bit, 800 MHz |
| ROM memory | 512 MB Flash |
| RAM memory | 512 MB DDR3 |
| **System data** | |
| Ethernet transfer rate | 10 Mbit/s / 100 Mbit/s |
| Ethernet connection technology | 1 × M12, 4-pin, D-coded |
| **Digital inputs** | |
| No. of channels | 4 |
| Connection technology of inputs | M12, 5-pin |
| Input type | PNP |
| Switch threshold | EN 61131-2 Type 3, PNP |
| Signal voltage Low signal | < 5 V |
| Signal voltage High signal | > 11 V |
| Signal current Low signal | <1.5 mA |
| Signal current High signal | > 2 mA |
| Type of input diagnostics | Channel diagnostics |

| Technical Data | |
|---|---|
| **Digital outputs** | |
| No. of channels | 4 |
| Connection technology of outputs | M12, 5-pin |
| Output type | PNP |
| Type of output diagnostics | Channel diagnostics |

# 14 EU Declaration of Conformity

Hans Turck GmbH & Co. KG hereby declares that wireless system read/write heads of type TN-UHF-Q…L…-EU… comply with directive 2014/53/EU. The complete text of the EU declaration of conformity can be obtained from the following Internet address: **www.turck.com**

# 15 Turck Subsidiaries - Contact Information

| | |
|---|---|
| **Germany** | Hans Turck GmbH & Co. KG<br>Witzlebenstraße 7, 45472 Mülheim an der Ruhr<br>**www.turck.de** |
| **Australia** | Turck Australia Pty Ltd<br>Building 4, 19-25 Duerdin Street, Notting Hill, 3168 Victoria<br>**www.turck.com.au** |
| **Belgium** | TURCK MULTIPROX<br>Lion d'Orweg 12, B-9300 Aalst<br>**www.multiprox.be** |
| **Brazil** | Turck do Brasil Automação Ltda.<br>Rua Anjo Custódio Nr. 42, Jardim Anália Franco, CEP 03358-040 São Paulo<br>**www.turck.com.br** |
| **China** | Turck (Tianjin) Sensor Co. Ltd.<br>18,4th Xinghuazhi Road, Xiqing Economic Development Area, 300381 Tianjin<br>**www.turck.com.cn** |
| **France** | TURCK BANNER S.A.S.<br>11 rue de Courtalin Bat C, Magny Le Hongre, F-77703 MARNE LA VALLEE Cedex 4<br>**www.turckbanner.fr** |
| **Great Britain** | TURCK BANNER LIMITED<br>Blenheim House, Hurricane Way, GB-SS11 8YT Wickford, Essex<br>**www.turckbanner.co.uk** |
| **India** | TURCK India Automation Pvt. Ltd.<br>401-403 Aurum Avenue, Survey. No 109 /4, Near Cummins Complex, Baner-Balewadi Link Rd., 411045 Pune - Maharashtra<br>**www.turck.co.in** |
| **Italy** | TURCK BANNER S.R.L.<br>Via San Domenico 5, IT-20008 Bareggio (MI)<br>**www.turckbanner.it** |
| **Japan** | TURCK Japan Corporation<br>Syuuhou Bldg. 6F, 2-13-12, Kanda-Sudacho, Chiyoda-ku, 101-0041 Tokyo<br>**www.turck.jp** |
| **Canada** | Turck Canada Inc.<br>140 Duffield Drive, CDN-Markham, Ontario L6G 1B5<br>**www.turck.ca** |
| **Korea** | Turck Korea Co, Ltd.<br>B-509 Gwangmyeong Technopark, 60 Haan-ro, Gwangmyeong-si, 14322 Gyeonggi-Do<br>**www.turck.kr** |
| **Malaysia** | Turck Banner Malaysia Sdn Bhd<br>Unit A-23A-08, Tower A, Pinnacle Petaling Jaya, Jalan Utara C, 46200 Petaling Jaya Selangor<br>**www.turckbanner.my** |

| | |
|---|---|
| **Mexico** | Turck Comercial, S. de RL de CV<br>Blvd. Campestre No. 100, Parque Industrial SERVER, C.P. 25350 Arteaga, Coahuila<br>**www.turck.com.mx** |
| **Netherlands** | Turck B. V.<br>Ruiterlaan 7, NL-8019 BN Zwolle<br>**www.turck.nl** |
| **Austria** | Turck GmbH<br>Graumanngasse 7/A5-1, A-1150 Wien<br>**www.turck.at** |
| **Poland** | TURCK sp.z.o.o.<br>Wroclawska 115, PL-45-836 Opole<br>**www.turck.pl** |
| **Romania** | Turck Automation Romania SRL<br>Str. Siriului nr. 6-8, Sector 1, RO-014354 Bucuresti<br>**www.turck.ro** |
| **Russian Federation** | TURCK RUS OOO<br>2-nd Pryadilnaya Street, 1, 105037 Moscow<br>**www.turck.ru** |
| **Sweden** | Turck Sweden Office<br>Fabriksstråket 9, 433 76 Jonsered<br>**www.turck.se** |
| **Singapore** | TURCK BANNER Singapore Pte. Ltd.<br>25 International Business Park, #04-75/77 (West Wing) German Centre, 609916 Singapore<br>**www.turckbanner.sg** |
| **South Africa** | Turck Banner (Pty) Ltd<br>Boeing Road East, Bedfordview, ZA-2007 Johannesburg<br>**www.turckbanner.co.za** |
| **Czech Republic** | TURCK s.r.o.<br>Na Brne 2065, CZ-500 06 Hradec Králové<br>**www.turck.cz** |
| **Turkey** | Turck Otomasyon Ticaret Limited Sirketi<br>Inönü mah. Kayisdagi c., Yesil Konak Evleri No: 178, A Blok D:4, 34755 Kadiköy/ Istanbul<br>**www.turck.com.tr** |
| **Hungary** | TURCK Hungary kft.<br>Árpád fejedelem útja 26-28., Óbuda Gate, 2. em., H-1023 Budapest<br>**www.turck.hu** |
| **USA** | Turck Inc.<br>3000 Campus Drive, USA-MN 55441 Minneapolis<br>**www.turck.us** |

TURCK

Over 30 subsidiaries and over
60 representations worldwide!

100003067 | 2021/05

www.turck.com