TURCK

Your Global Automation Partner

TN-UHF-...-LNX
UHF Reader

Instructions for Use

Contents

1 About these Instructions

These instructions describe the setup, functions and use of the product and help you to operate the product according to its intended purpose. Read these instructions carefully before using the product. This will prevent the risk of personal injury and damage to property. Keep these instructions safe 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

The following symbols are used in these instructions:

	**DANGER** DANGER indicates a hazardous situation with a high level of risk, which, if not avoided, will result in death or serious injury.
	**WARNING** WARNING indicates a hazardous situation with a medium level of risk, which, if not avoided, will result in death or serious injury.
	**CAUTION** CAUTION indicates a hazardous situation with a medium level of risk, which, if not avoided, will result in moderate or minor injury.
	**NOTICE** CAUTION indicates a situation which, if not avoided, may cause damage to property.
	**NOTE** NOTE indicates tips, recommendations and important information about special action steps and issues. The notes simplify your work and help you to avoid additional work.
▶	**MANDATORY ACTION** This symbol denotes actions that the user must carry out.
⇨	**RESULT OF ACTION** This symbol denotes the relevant results of an action.

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 in the HF are called “read/write heads” and “readers” in the UHF area. “Tag”, “transponder” and “mobile data memory” are common synonyms for “data carriers”.

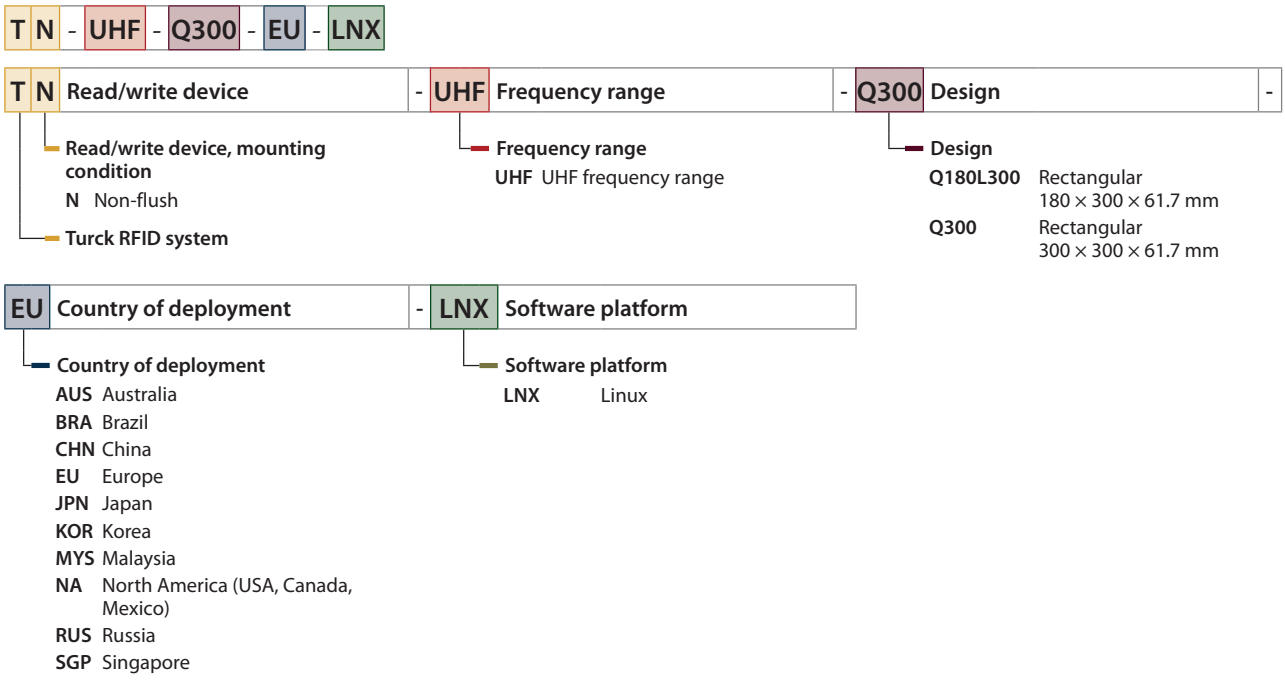
1.5 Feedback about these instructions

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 delivery consists of the following:

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

2.3 TURCK service

TURCK supports you in your projects — from the initial analysis right through to the commissioning of your application. The TURCK product database at www.turck.com offers you several software tools for programming, configuring or commissioning, as well as data sheets and CAD files in many export formats.

For the contact details of our branches worldwide, please see page [▶ 113].

3 For your safety

The product is designed according to state of the art technology. Residual hazards, however, still exist. Observe the following safety instructions and warnings in order to prevent danger to persons and property. TURCK accepts no liability for damage caused by failure to observe these safety instructions.

3.1 Intended use

The readers with an integrated RFID interface are used for contactless data exchange with the RFID 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 device must 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 meets the EMC requirements for the industrial areas. When used in residential areas, take measures to prevent radio frequency interference.
- The device must only be fitted, installed, operated, parameterized and maintained by trained and qualified personnel.
- Only use the device in compliance with the 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 the use of a secure password.

3.3 Notes on EU Directive 2014/53/EU (RED Directive)

For safe and proper use of the device, ensure the following physical and logical safety measures in accordance with DIN EN 18031-1 in the environment:

- Access control: Enable access to security-related data and settings only to authorized persons, devices and services. Especially protect cryptographic keys in the device.
- Authentication: Manage access to security-related data and settings through appropriate authentication mechanisms. This also includes the regular verification and adjustment of passwords and other authentication methods.
- Firmware management: Regularly check the availability of new firmware versions at www.turck.com and carry out updates promptly. Check the integrity of firmware updates by comparing them with the hash values provided on the TURCK website.
- Data protection and communication: Protect the data stored in the device for integrity and confidentiality. Secure communication with the device against manipulation, unauthorized access and listening in.
- Attack protection: Take measures to prevent successful replay, denial of service or brute force attacks.
- Vulnerability management: Ensure that known vulnerabilities cannot be exploited.
- Interface control: Only send valid and authorized data to the device interfaces.

All existing interfaces of the device are required for operation. Optional services are not available. An overview of the network interfaces and protocols can be found in the following table (unless described in the previous chapters):

Interface/protocol	Default state	Configurable
(S)NTP	On	Yes
HTTP	On	No
IBTP	On	No
MTXP	On	No
Debus via RS485	On	No
Digital I/Os	On	Yes
RFID wireless protocols (UHF)	On	No
DHCP	On	Yes
DNS	On	No
Service interface	On	No

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

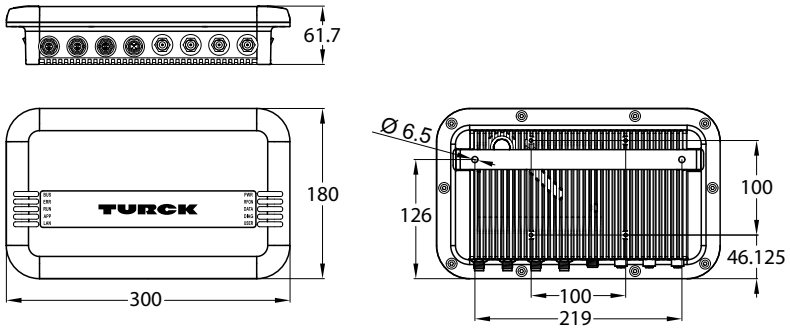


Fig. 1: Dimensions – TN-UHF-Q180L300…

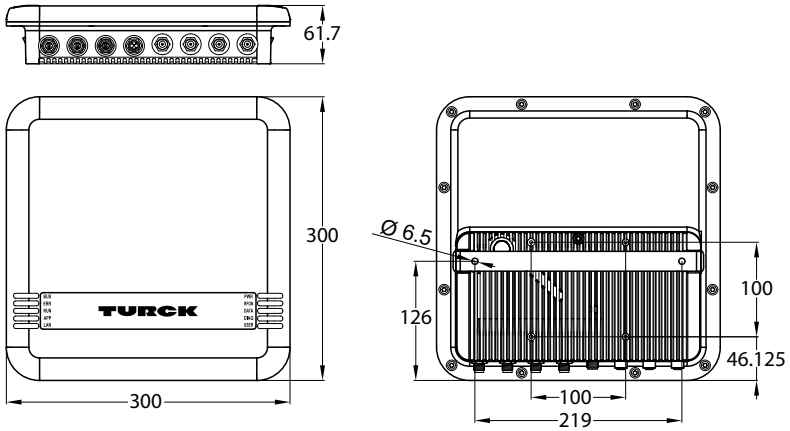


Fig. 2: Dimensions – TN-UHF-Q300…

4.1.1 Indication elements

The device has the following LED indicators:

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

An acoustic signal 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 and 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.

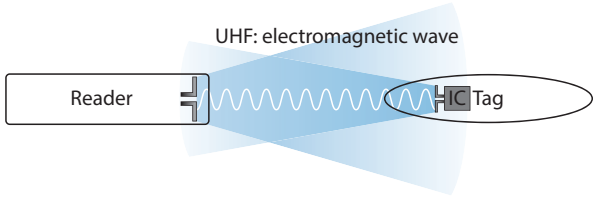


Fig. 3: Operating principle of UHF-RFID

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 their power and the frequency in use, the devices have a range of a few millimeters up to several meters. The specified maximum distance between the read/write heads represents values measured under laboratory conditions, free from any influences caused by surrounding materials. Attainable distances may vary due to component tolerances, mounting conditions, ambient conditions and influences caused by surrounding materials (especially 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.

4.4.1 Linux distribution — Software components

The Linux distribution of the device contains the following software components:

- SSH
- SFTP
- HTTP
- IBTP
- MTXP
- DHCP
- SNTP
- Node.js 6.9.5 (LTS)
- Python 3.x

4.5 Technical accessories

Optionally available accessories for mounting, connecting and parameter setting can be found in the TURCK product database at www.turck.com. Accessories are not supplied with the device.

5 Installing

The device is designed for mounting on a bracket based on the VESA 100 × 100 standard. 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.

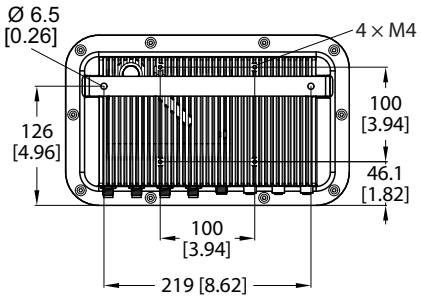


Fig. 4: Rear view – TN-UHF-Q180…

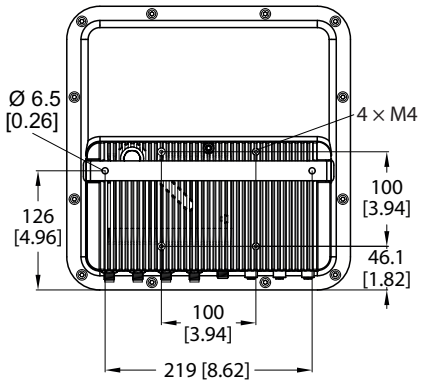


Fig. 5: Rear view – TN-UHF-Q300…

6 Connecting

6.1 Connecting devices to Ethernet

The device has a 4-pin M12 female connector for connection to an Ethernet system.

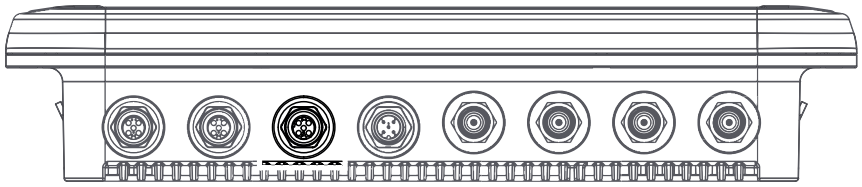


Fig. 6: M12 Ethernet connector

- ▶ Connect the device to Ethernet in accordance with the pin assignment below (max. tightening torque: 0.8 Nm).

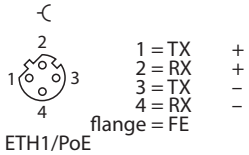


Fig. 7: Pin assignment for Ethernet connections

NOTE
With PoE, the supply voltage is transmitted via PoE Mode A with 4-wire cables.
The use of PoE and 24 VDC simultaneously is not supported.

6.2 Connecting the power supply

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

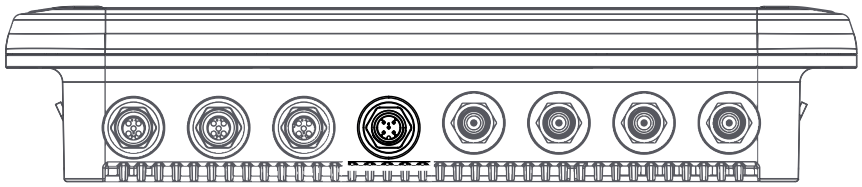


Fig. 8: M12 plug connector for connecting the power supply

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

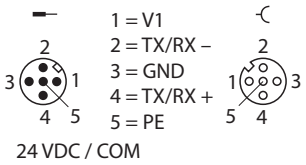


Fig. 9: Pin assignment of the power supply terminals

6.3 Connecting digital sensors and actuators

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

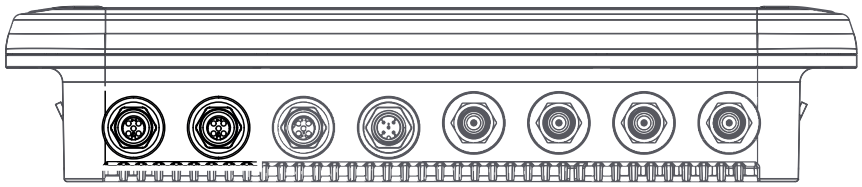


Fig. 10: M12 plug connectors for connecting digital sensors and actuators

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).

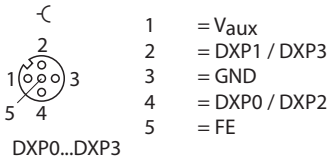


Fig. 11: Connections for digital sensors and actuators – pin assignment

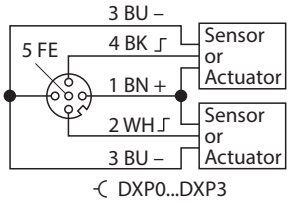


Fig. 12: Connections for digital sensors and actuators – wiring diagram

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.

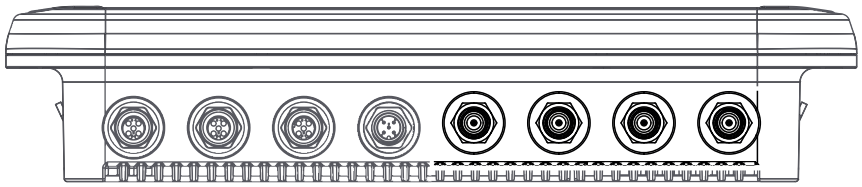


Fig. 13: RP-TNC sockets for connecting external antennas

- ▶ 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.

To access the device from the console, additional software tools are required (for example PuTTY). A program such as WinSCP can be used to transfer files between the device and a PC. 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 log must be implemented by the user. As of firmware version 1.2.5.0, the log is no longer implemented by the user as an RFID REST API is provided. To parameterize the reader, the REST API must be activated (see chapter [▶ 63]).

7.1 Parameterizing the reader using the web server

The integrated web server can be used to set the devices and send commands to the devices. In order to be able to open the web server with a PC, the device and the PC must be in the same IP network.

7.1.1 Opening a web server

The web server can be opened from a web browser or from the TURCK Automation Suite (TAS). Accessing the web server via TAS is described in the section entitled "Adjusting network settings."

7.1.2 Editing settings in the web server

A login is required to edit settings via the web server. The default password is "password".

NOTE
TURCK recommends changing the password after the first login for security reasons.

- Open the device's web server.
- Enter **Username** and **Password**.
- Click **Login**

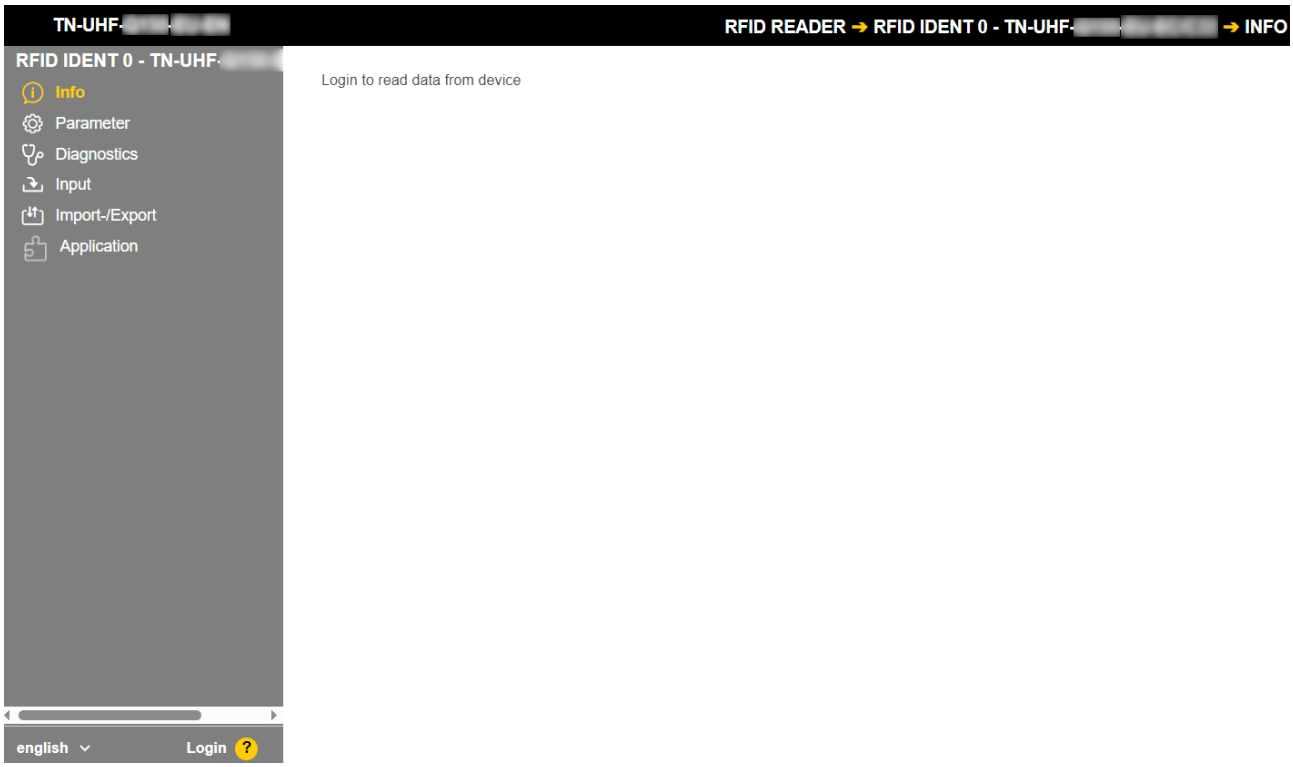


Fig. 14: Web server — login

- Change the password after you log in.

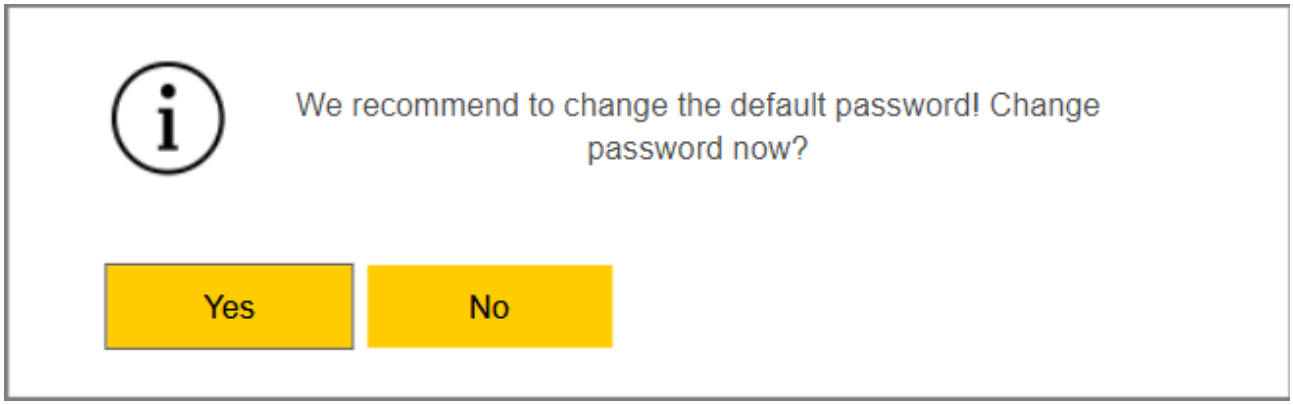


Fig. 15: Web server — password change dialog

After you log in, the home page is displayed with the device information.

▶ Click **RFID READER** to display and set the device parameters.

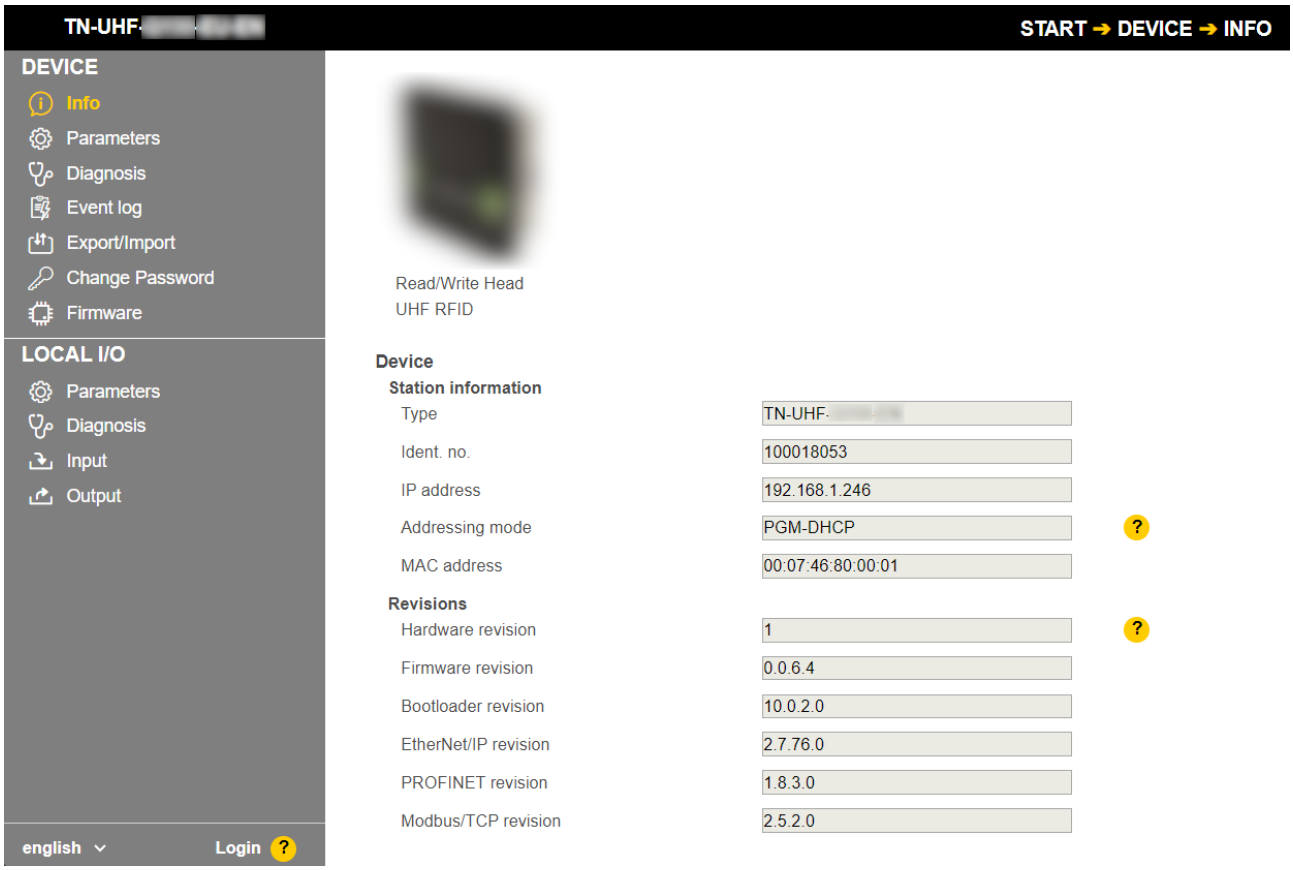


Fig. 16: Web server — RFID Reader — Info

▶ Click **Parameter** in the navigation bar on the left of the screen.

⇨ All parameters of the device are displayed.

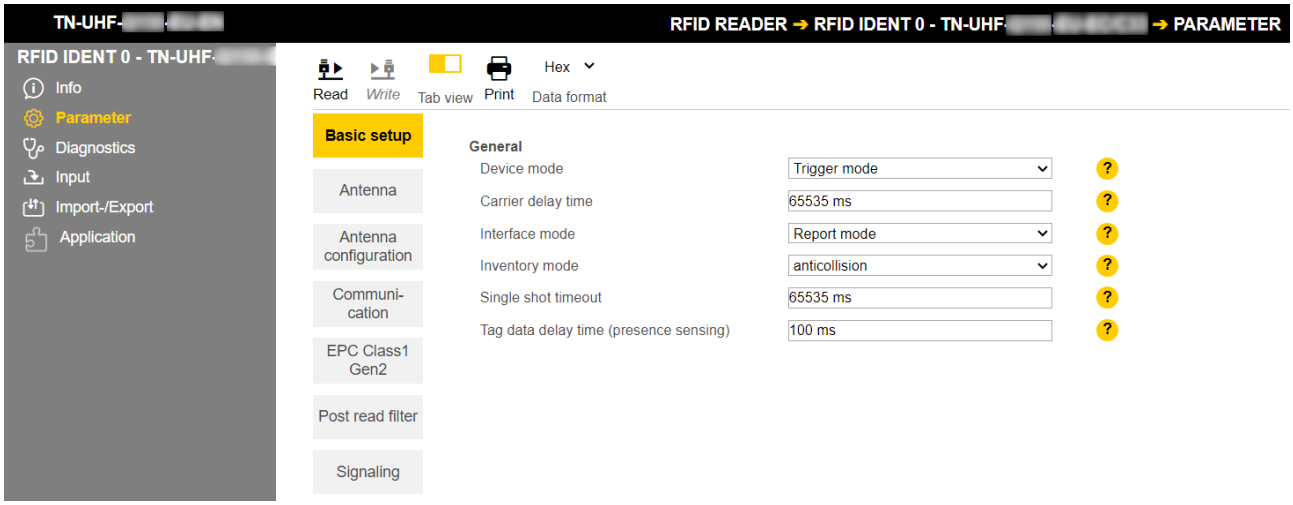


Fig. 17: Web server — RFID Reader — Parameter

The following setup windows can be called up:

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

▶ Set the parameters: Click **Write**.

NOTE
While a parameter is being set, the ERR LED lights up red and automatically turns green.

7.1.3 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.

A login is required to edit settings via the web server. The default password is "password".

NOTE
TURCK recommends changing the password after the first login for security reasons.

▶ Open the device's web server.
▶ Enter **Username** and **Password**.
▶ Click **Login**

Example: Configuring multiplex operation

▶ Select **RFID READER**.
▶ Select **Parameter**.

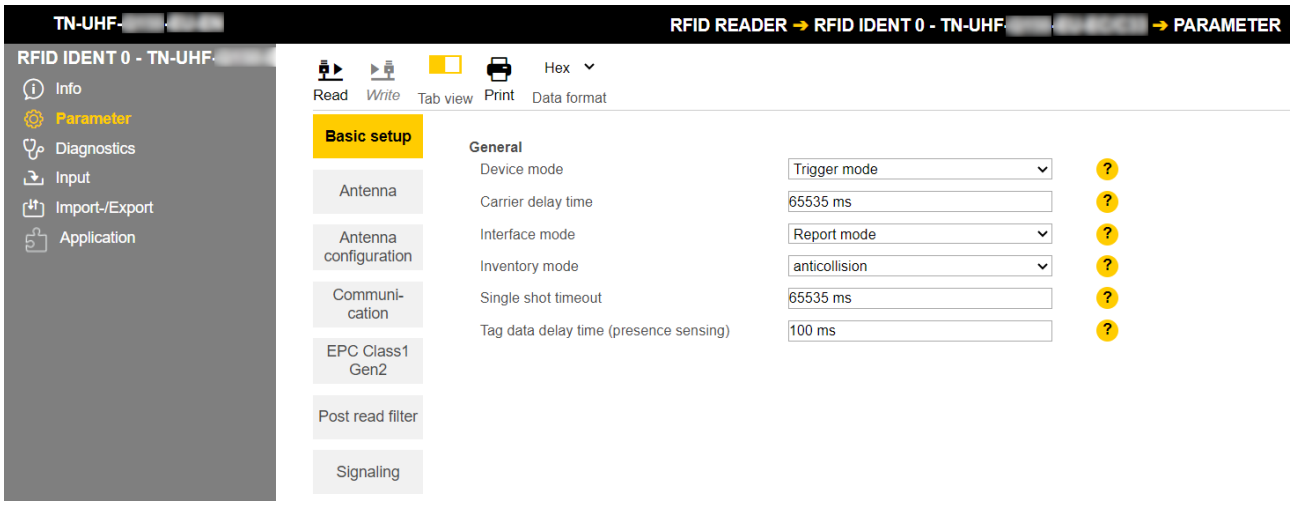


Fig. 18: RFID Reader — Parameter

▶ Select **Antenna**.

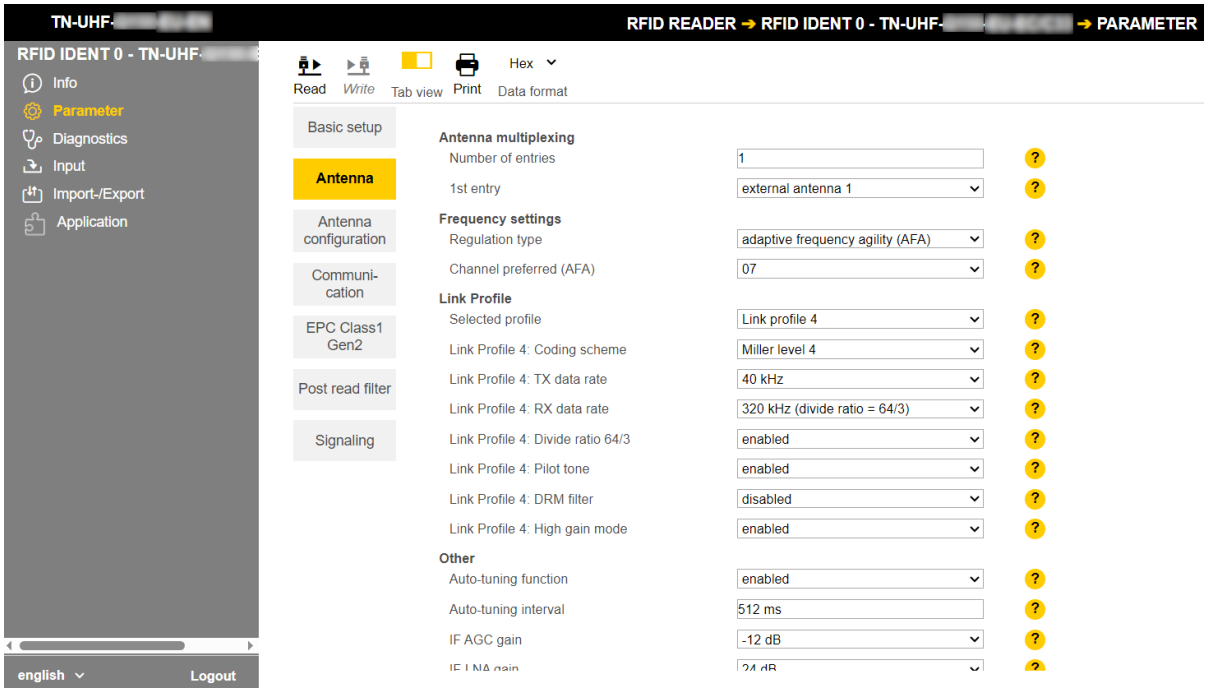


Fig. 19: RFID Reader — Parameter — Antenna

▶ Under **Antenna multiplexing**, enter the number of antennas in the **Number of entries** field.

▶ Select **Antenna configuration**.

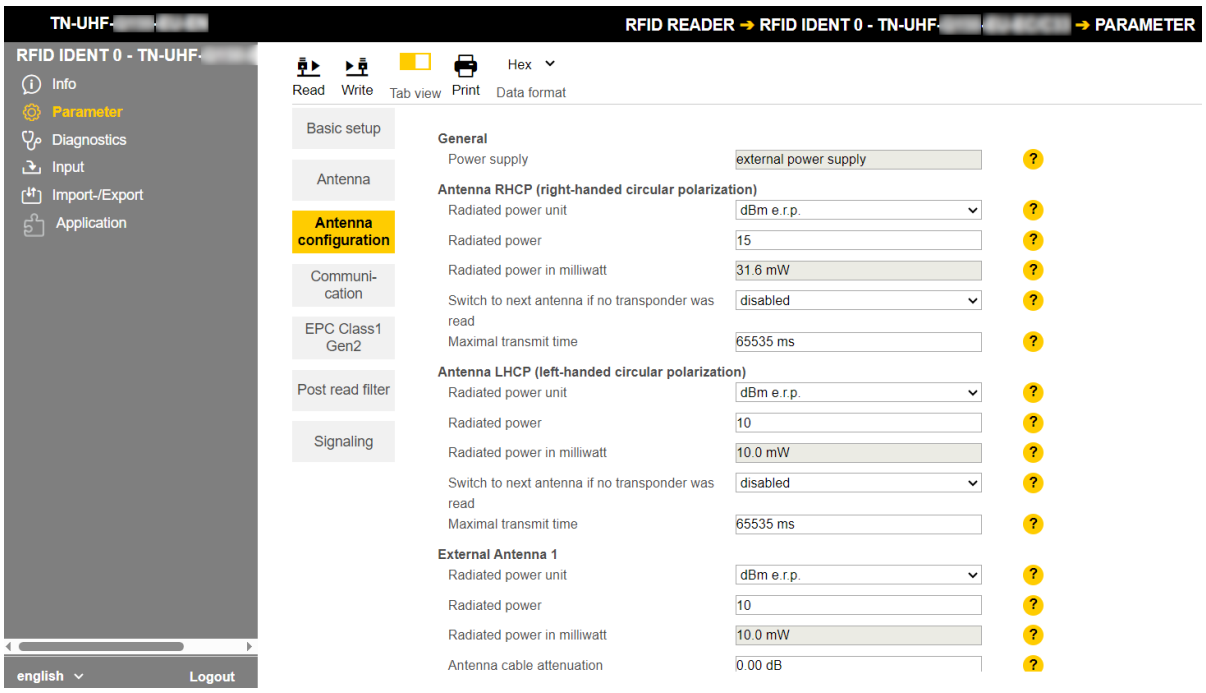


Fig. 20: RFID reader — Parameter — Antenna configuration

▶ For each antenna, enter in the **Maximum transmit time** field the amount of time for which that antenna should remain active.

7.1.4 Setting antenna power

The antenna power of the reader can be set for the specific application. The radiated power can be entered directly for the integrated antenna. 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 female connector 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.

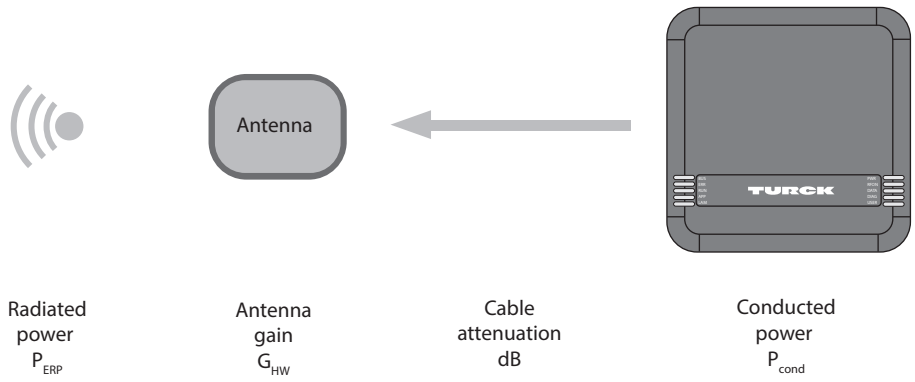


Fig. 21: Power calculation – Relevant variables (schematic representation)

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 web server uses an exclamation point to identify invalid configurations. 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

- Set the radiated power under **External Antenna 1** → **Radiated power** (here: 24 dBm e.r.p.).
- Refer to the data sheet of the cable used for the cable attenuation.
- Enter the cable attenuation at **Antenna cable attenuation**.
- 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).
- Set antenna gain at **Antenna gain** (here: 5.00).
  - ⇨ The power at the TNC female connector (P_{cond}) is calculated automatically and displayed under **Conducted power**.

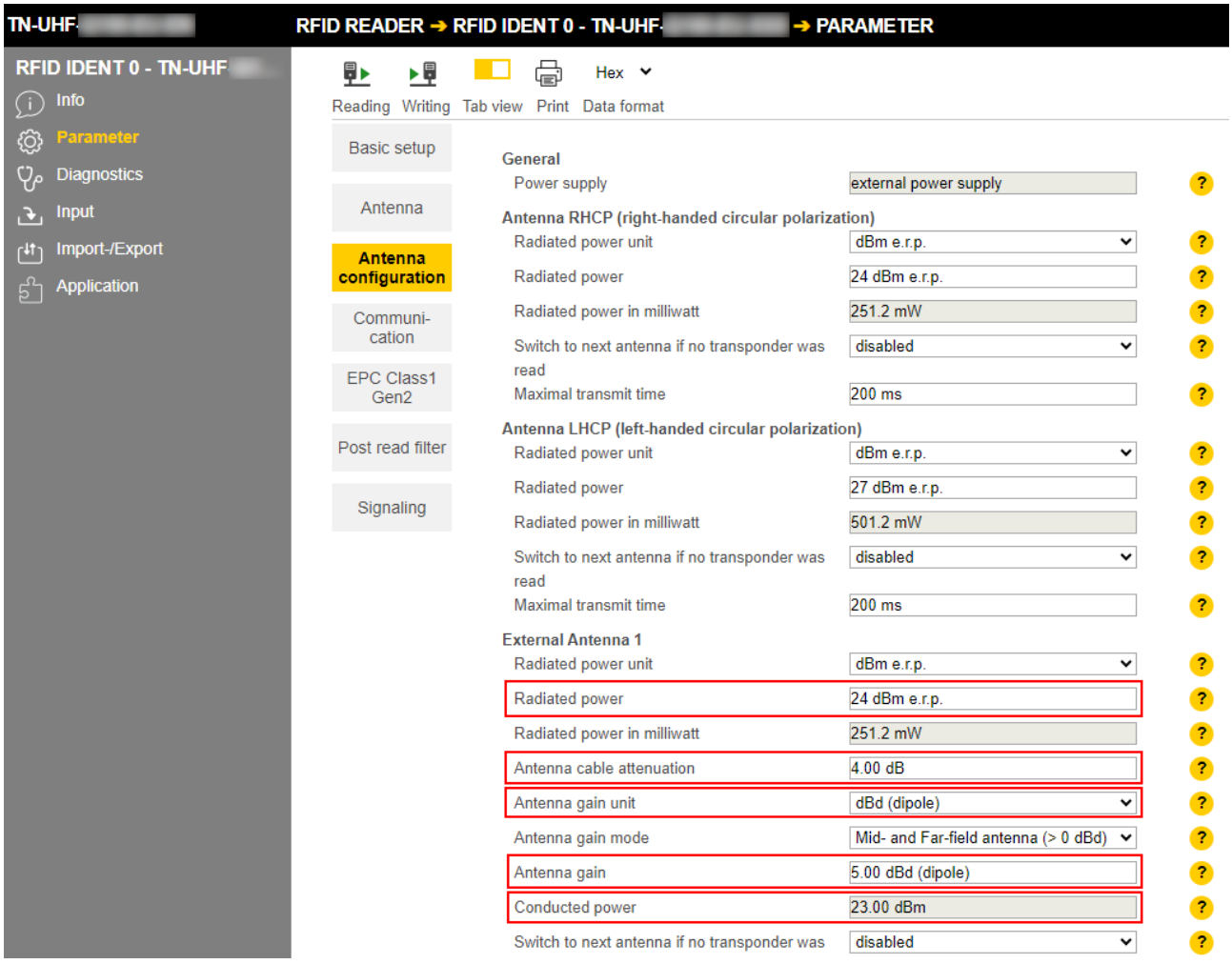


Fig. 22: Setting the antenna power

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

7.1.5 Setting antenna polarization

The antenna polarization can be set via the web server or via TAS. 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.

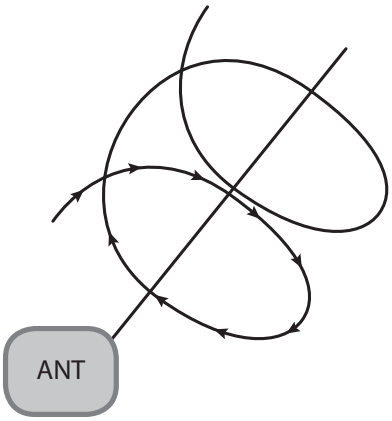


Fig. 23: Antenna polarization circular (RHCP)

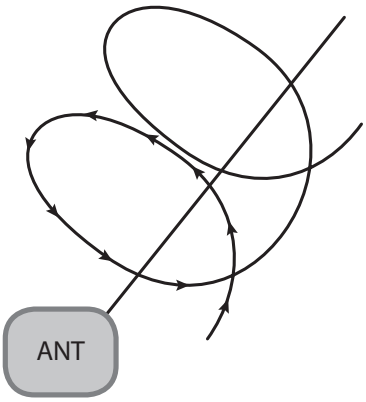


Fig. 24: Antenna polarization circular (LHCP)

Switching antenna polarization

Polarization switching is activated via the multiplex settings.

- At **Antenna** → **Number of entries**, set the value **2**.
- At **Antenna** → **1st entry**, set the value **antenna RHCP**.
- At **Antenna** → **2nd entry**, set the value **antenna LHCP**.

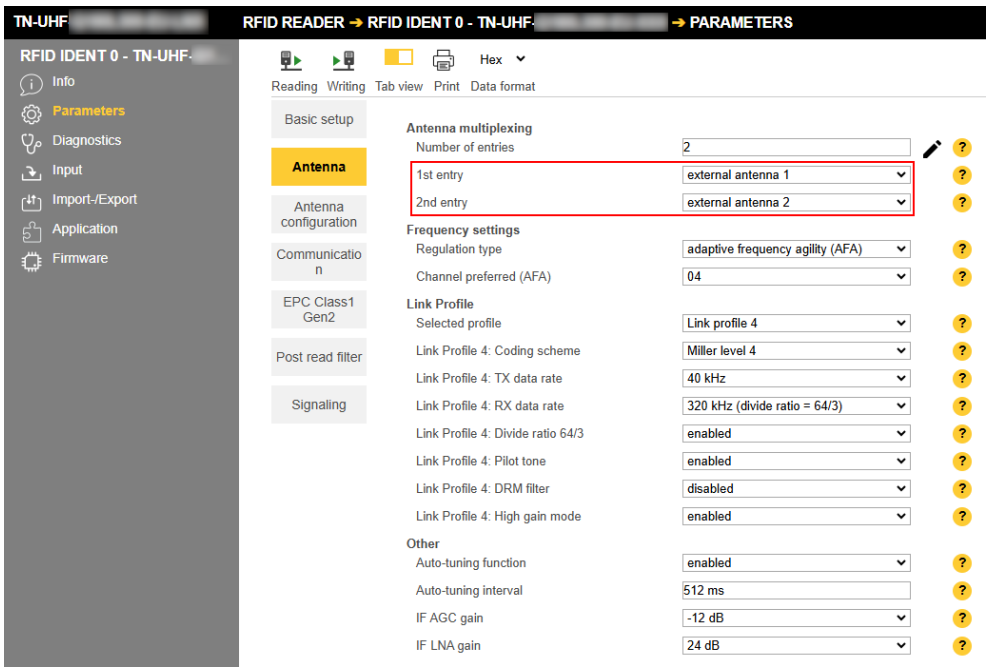


Fig. 25: Switching antenna polarization

- ▶ 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**).

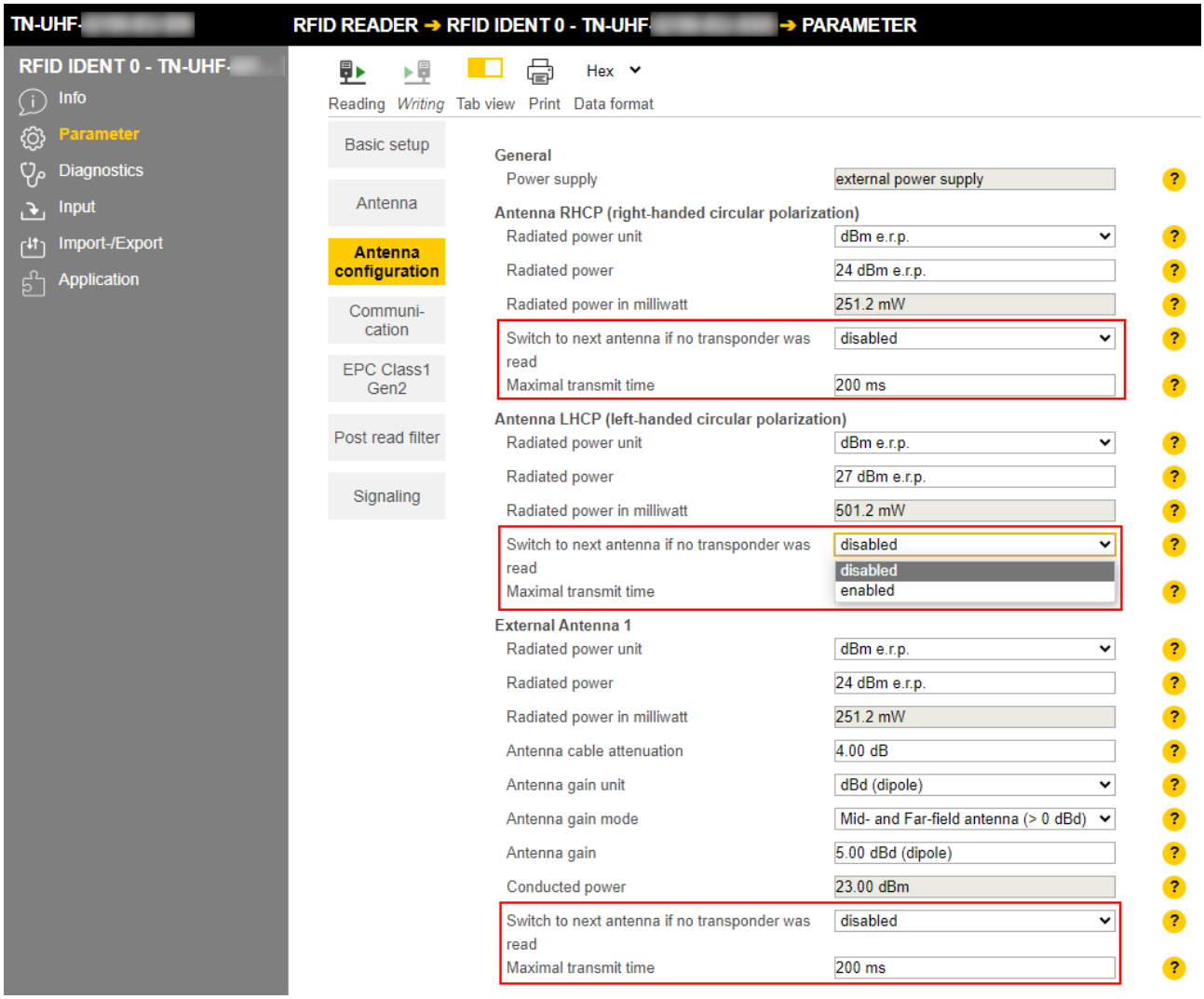


Fig. 26: Switching polarization automatically

7.1.6 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 the Presence sensing mode, the readers are automatically switched on as soon as a tag is located in the detection range.

▶ At **Basic setup** → **General** → **Device mode**, set the **Presence sensing mode** option.

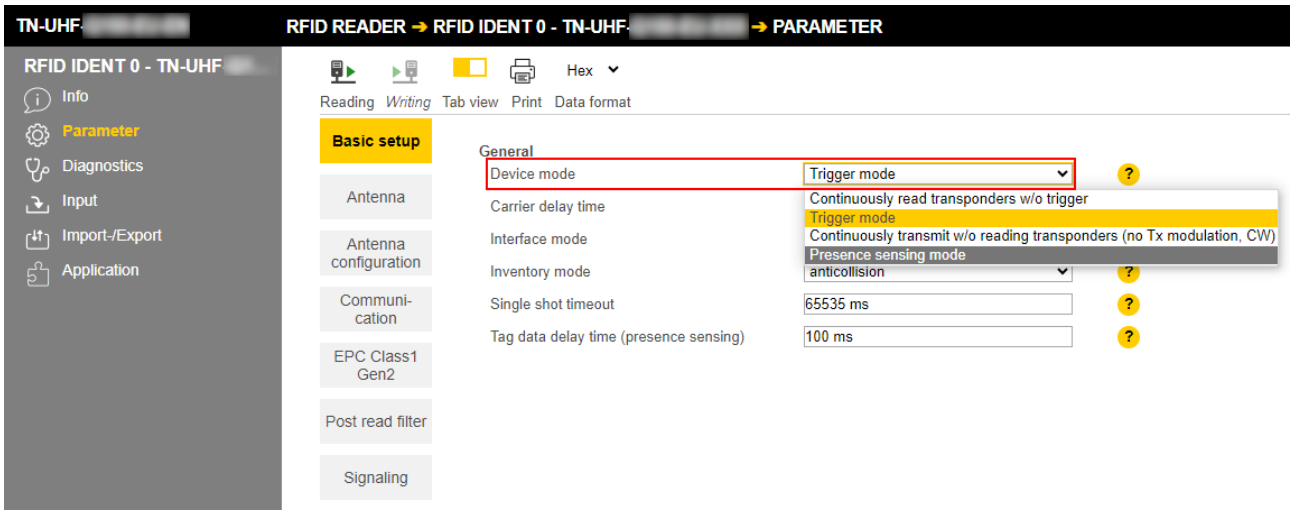


Fig. 27: Switching on presence sensing mode

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.

7.1.7 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 adjustable values are described in the web server.

Example: Switching on RSSI transmission

- ▶ Switch on RSSI transmission: At **Communication** → **Message data content** → **Transponder RSSI**, select the **enabled** option.

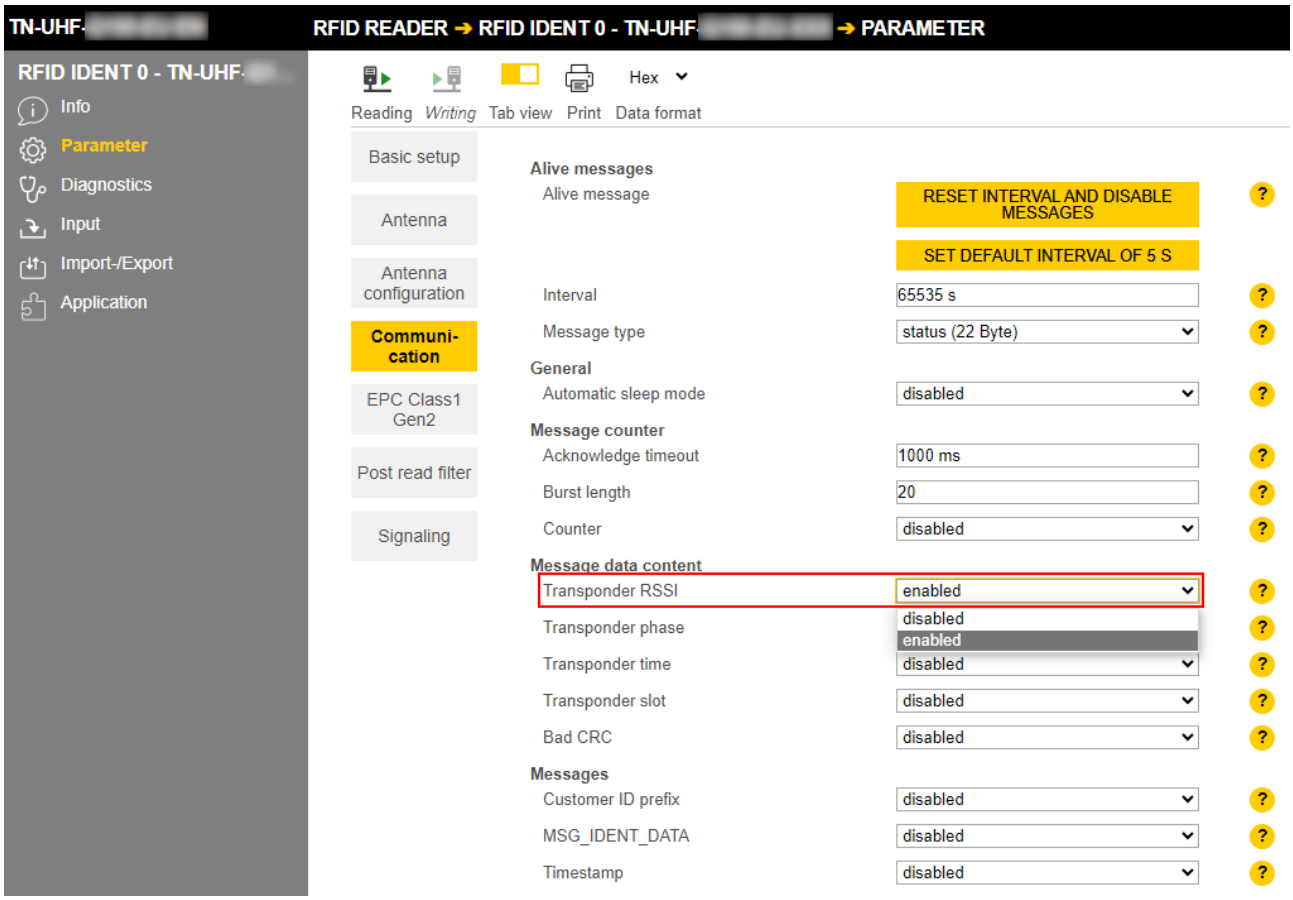


Fig. 28: Switching on RSSI transmission

⇨ The RSSI value is displayed with the inventory in the read data.

7.1.8 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 adjustable values are described in the web server.

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**, enable the RSSI Filter.
- Set the threshold at **Post read filter** → **RSSI filter** → **Lower threshold**.

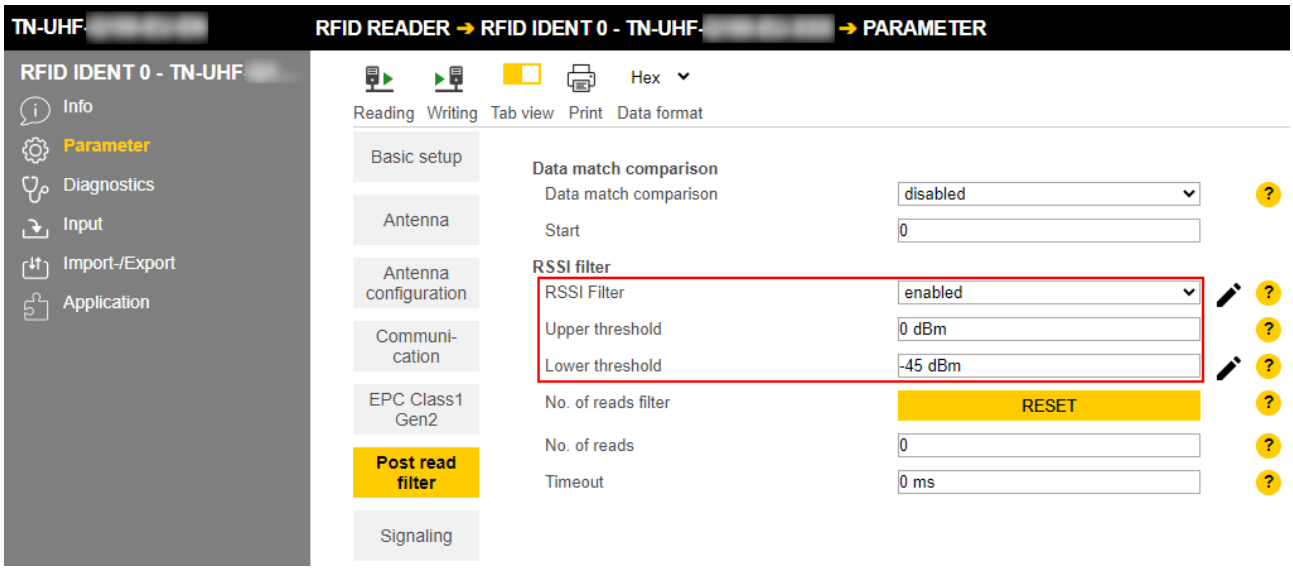


Fig. 29: Switching on the RSSI filter

⇨ Example: All read operations below an RSSI value of -45 dBm are filtered out.

7.2 Testing the reader using the web server

The **Application** function enables the devices to be tested with the web server.

▶ Click **RFID READER → Application**

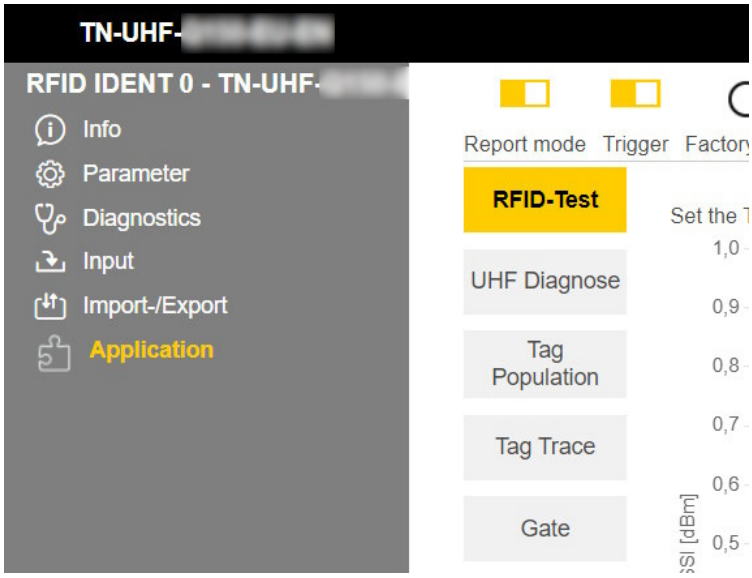


Fig. 30: Web server — RFID Reader — Application

The following items are available in the **Application** area: **RFID Test**, **UHF Diagnostics**, **Tag Population**, **Tag Trace** and **Gate**:

- RFID-Test: If the trigger is set to ON, the RF field is activated and tags can be read.
- UHF Diagnostics: The diagrams show the interference frequencies of all channels used.
- Tag Population: A tool to determine the radiated power from which all tags can be read.
- Tag Trace: Tool for reading individual tags with curve of signal strength over time.
- Gate: Tool for reading multiple tags (bulk reading)

RFID test allows EPC information from tags to be displayed and read out in single-tag and multi-tag mode. The received RSSI values are displayed as a curve in relation to time.

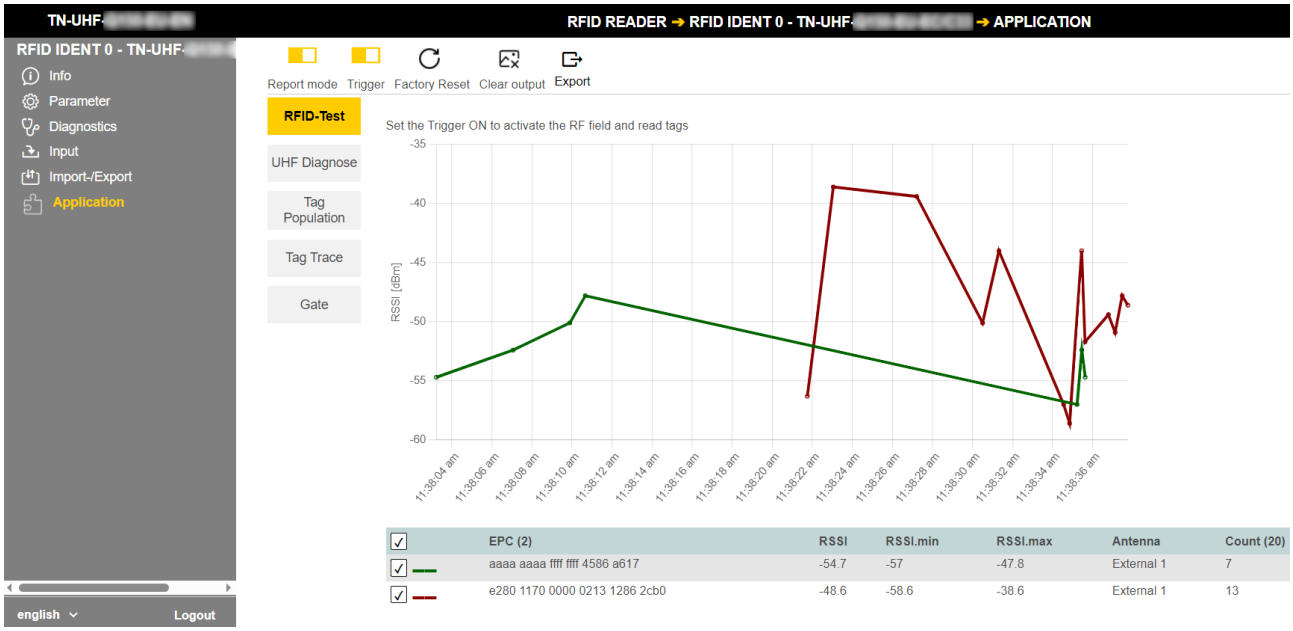


Fig. 31: Example RFID test

The **UHF diagnostics** display the current power level being received by the reader per channel.

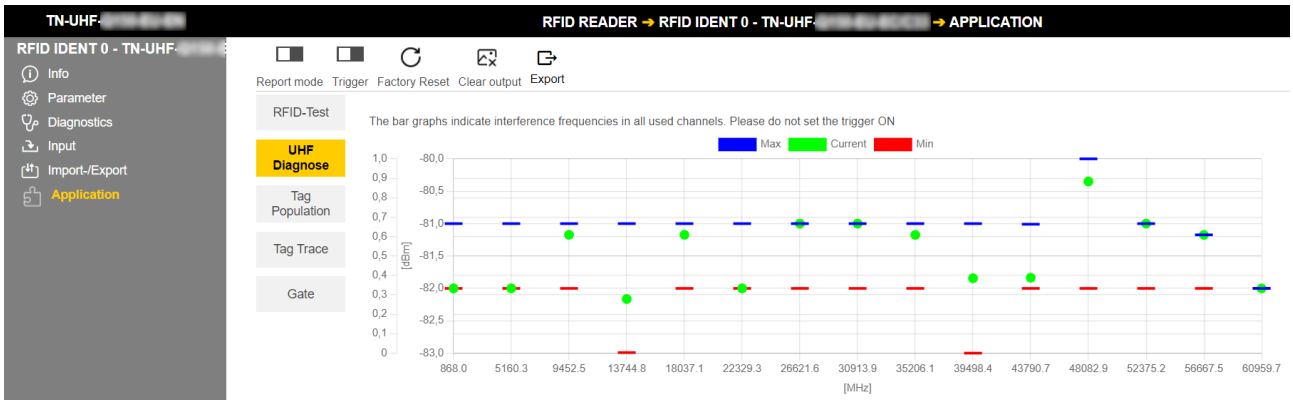


Fig. 32: Example UHF diagnostics

7.3 Adjusting network settings

7.3.1 Adjusting network settings via TAS (TURCK Automation Suite)

The device is factory set to IP address 192.168.1.254. The IP address can be set via TAS (TURCK Automation Suite). TAS is available free of charge at www.turck.com.

- ▶ Connect the device to a PC via the Ethernet interface.
- ▶ Open TAS.
- ▶ Click **Scan network**.

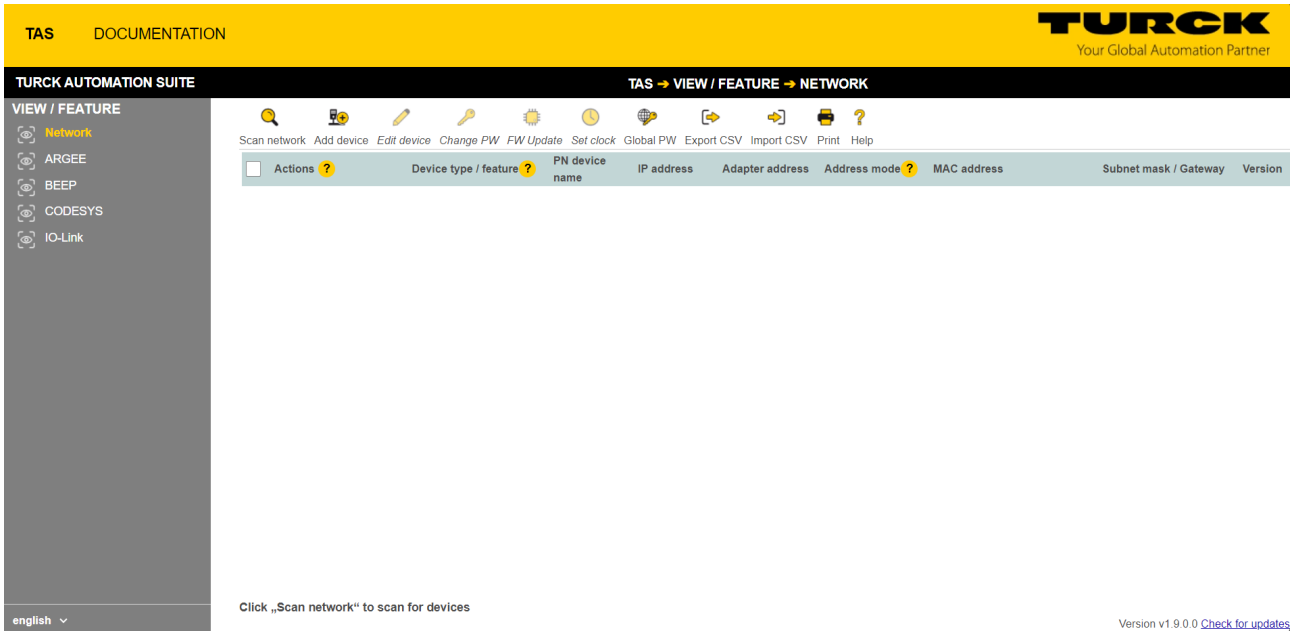


Fig. 33: TAS — home screen

⇨ TAS displays the connected devices.

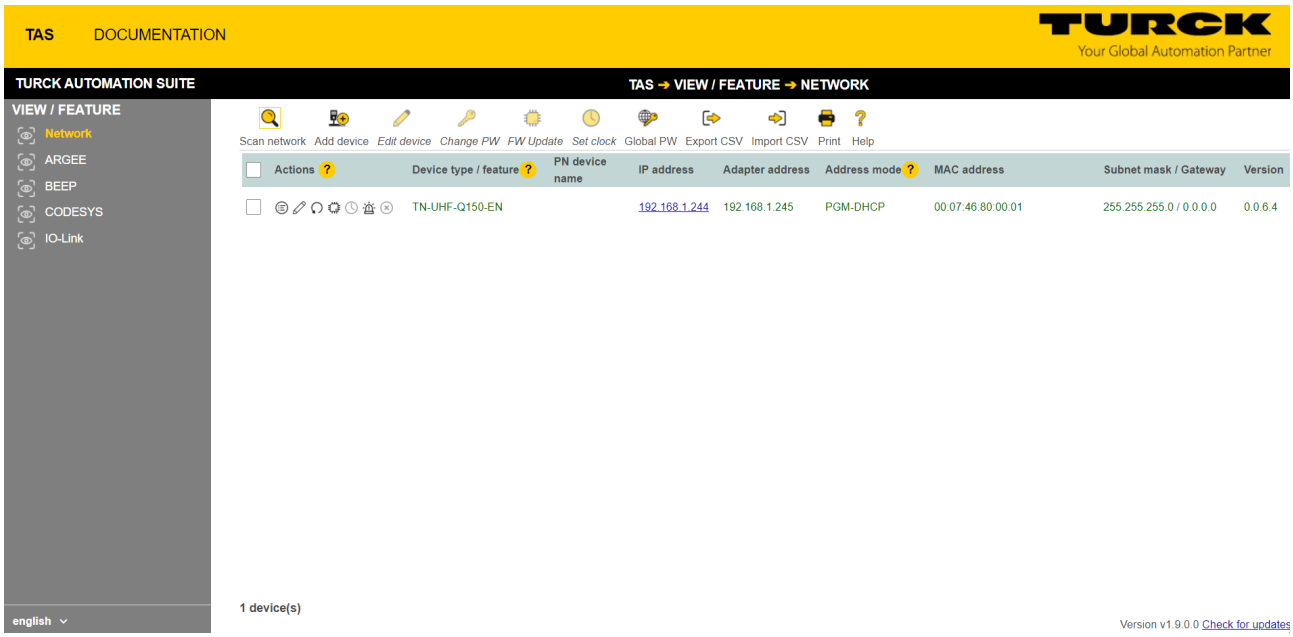


Fig. 34: TAS — found devices

- Select the required device (check the checkbox).
- Click **Edit device**.

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 NETWORK DATA**.

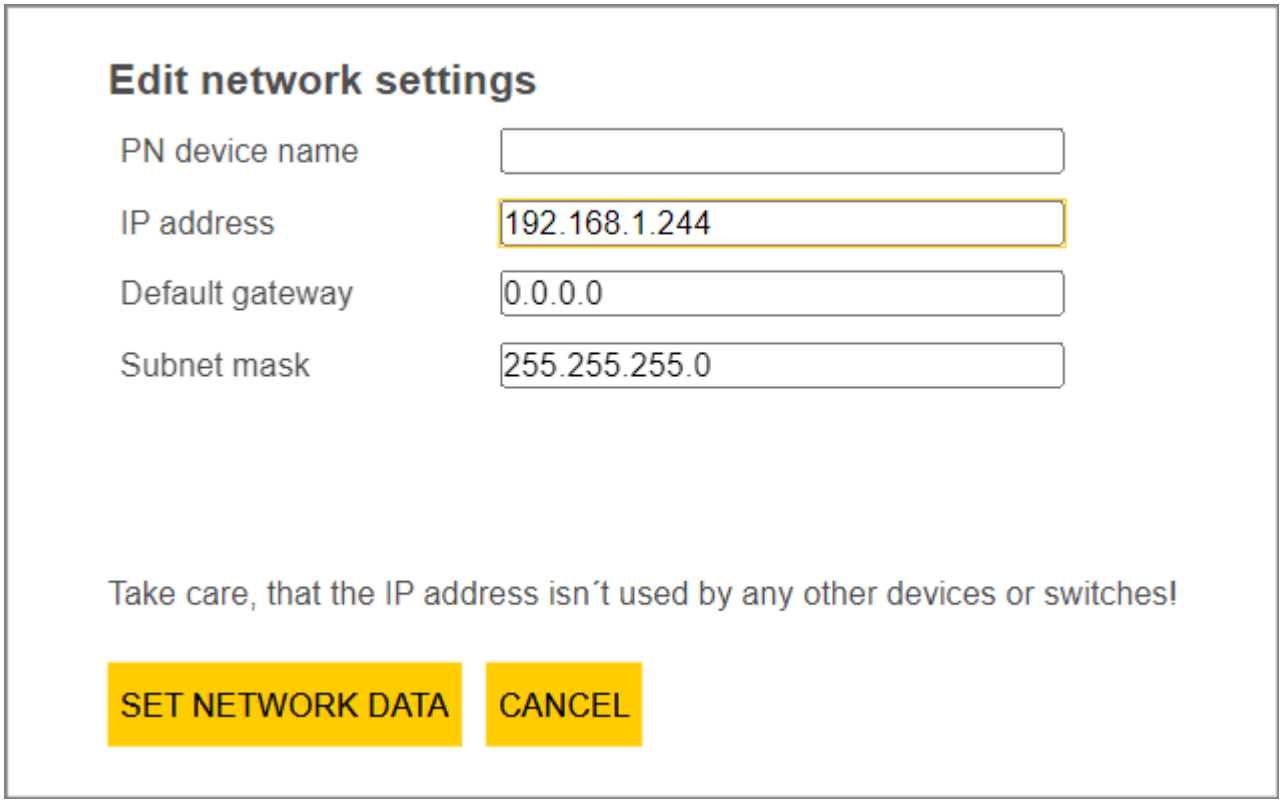


Fig. 35: TAS — edit network settings

7.3.2 Adjusting network settings via the web server

NOTE
The device must be in PGM mode in order to set the IP address via the web server.

- Open the web server.
- Log into the device as administrator.
- Click **Parameters → Network**.
- Change the IP address and, if necessary, also the subnet mask and default gateway.
- Write the new IP address, subnet mask and default gateway via **SET NETWORK CONFIGURATION** to the device.

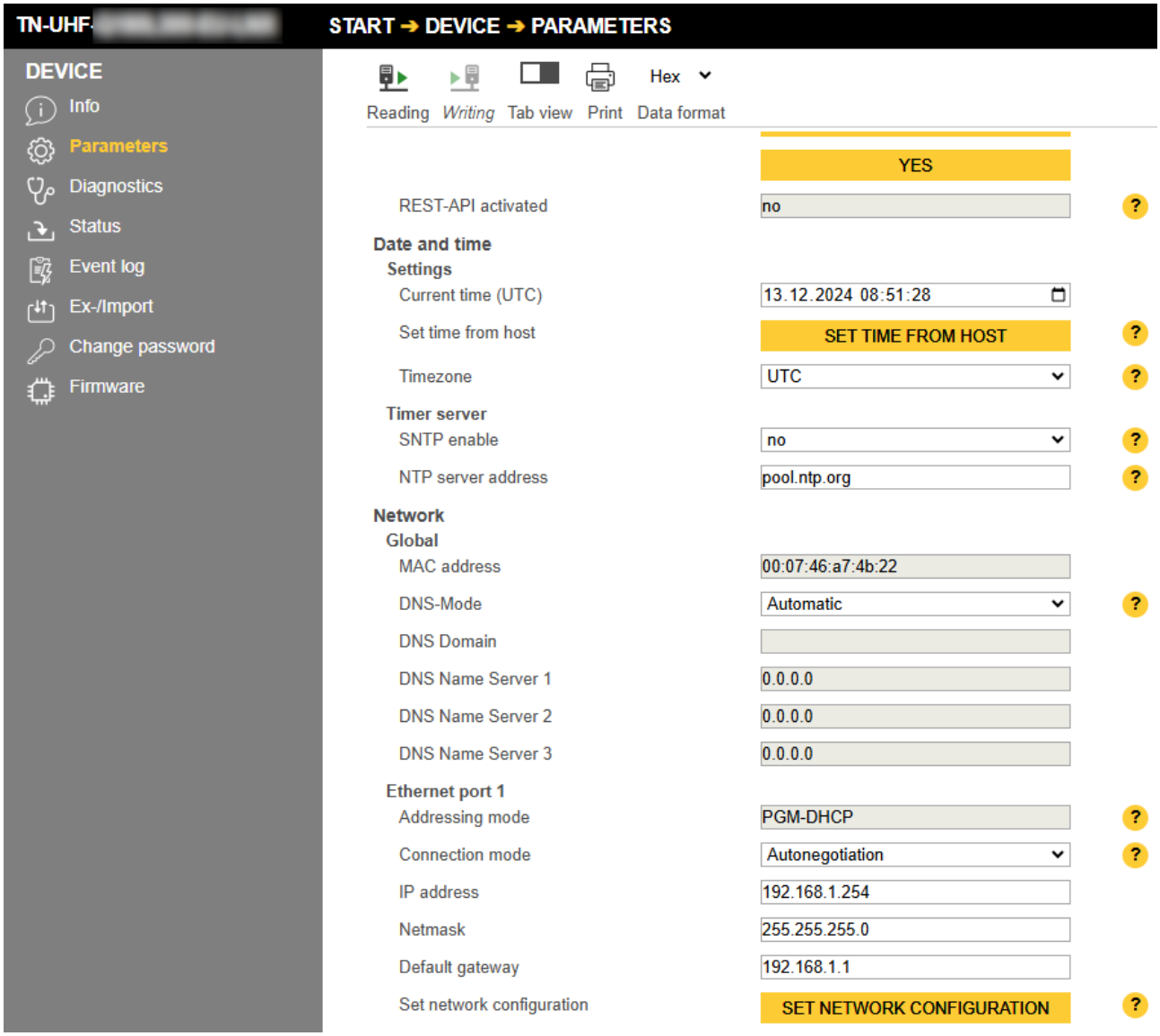


Fig. 36: Adjusting network settings via the web server

7.4 Programming RFID channels

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

7.4.1 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()
```

7.4.2 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;
}
```

7.5 Programming digital channels (DXP)

7.5.1 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) 1: Input on (24V)
		Output	88	0: Output off (0V) 1: Output on (24V)
DXP1		Input	60	0: Input off (0V) 1: Input on (24V)
		Output	89	0: Output off (0V) 1: Output on (24V)
DXP2	C1	Input	57	0: Input off (0V) 1: Input on (24V)
		Output	86	0: Output off (0V) 1: Output on (24V)
DXP3		Input	58	0: Input off (0V) 1: Input on (24V)
		Output	87	0: Output off (0V) 1: Output on (24V)

Setting the switchable VAUX power supply

Socket	Type	GPIO	Possible values
C0	Output VAUX0	41	0: VAUX off 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 1: Error or overvoltage on VAUX
	Input VAUX1	55	

7.5.2 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 0: Switch off channel

7.5.3 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)
```

7.5.4 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;
}
```

7.5.5 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;
        }

    }
```

7.6 Programming LED functions

7.6.1 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.

7.6.2 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 0: Switch off LED
RUN	Green/red	1: Switch on LED 0: Switch off LED
APPL	Green/red	1: Switch on LED 0: Switch off LED

7.6.3 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()
```

7.6.4 Programming LED functions with Node.js

The following examples show the programming of the LED functions with Node.js. For more information about Node.js and the Node.js packages, see:

- **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');
```

7.6.5 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* Ziel, const char* Quelle);
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;
}
```

7.7 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.

7.8 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

7.8.1 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
```

7.8.2 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.

7.8.3 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`

7.9 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

7.10 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.

7.10.1 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. 37: 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.

7.11 Using the REST API

The TN-UHF-…-LNX UHF reader contains an integrated REST API with which RFID functions can be executed directly. The GET and POST commands shown here are mapped, if necessary, with a REQUEST body in JSON data format.

NOTE
The REST API is included as of firmware version 1.2.5.0. For easier integration of the REST API, a .YAML file is available in the download area on the product page at www.turck.com.

7.11.1 Activating the REST API in the web server

Hardware used:

- TN-UHF-Q180L300-EU-LNX with IP address 192.168.1.20

NOTE
By default, the REST API is deactivated. After activation, a restart is necessary for the changes to become active.

- ▶ Open the web server of the device.
- ▶ Enter **user name** and **password**.
- ▶ Click **Login**.
  - ⇨ After you log in, the home page is displayed with the device information.
- ▶ Click **Parameters** in the navigation bar on the left of the screen.
- ▶ Under **Activate REST API**, select **yes**.
- ▶ Click **Write**.
- ▶ Restart device.

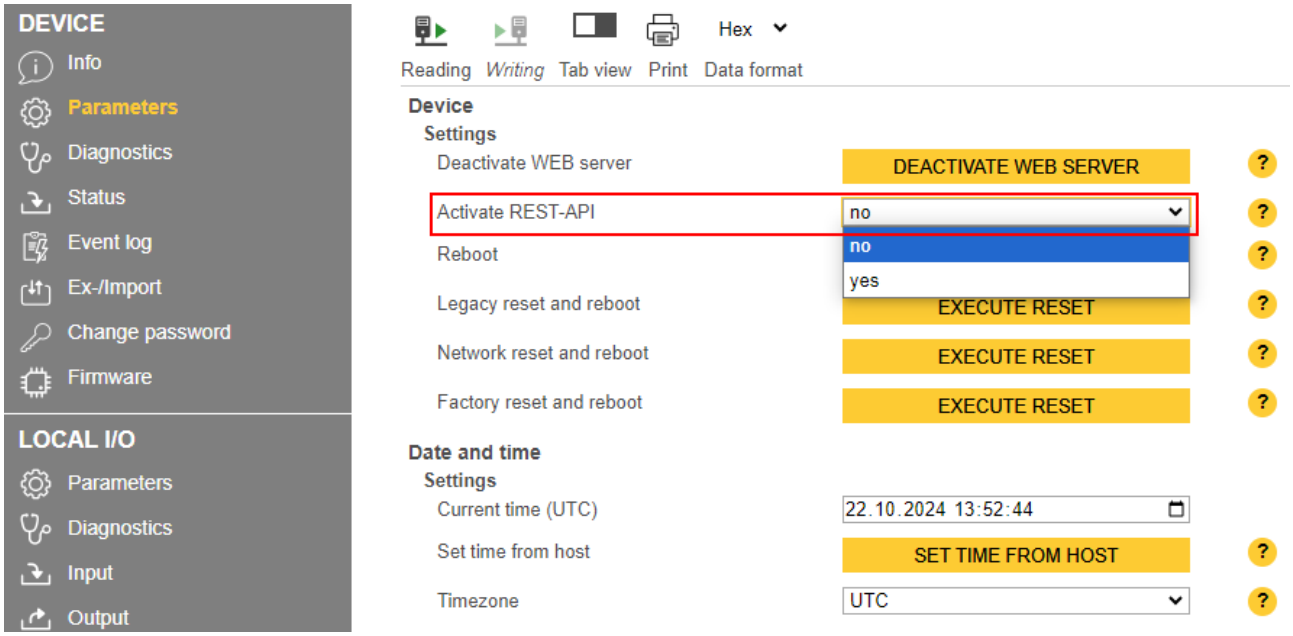


Fig. 38: Web server — activating the REST API

⇨ Device restarts, REST API is activated.

7.11.2 Overview of commands

Command	Function
Process an inventory	Executes an inventory command to read individual or multiple tags (bulk reading).
Read data from transponder	Reads data from any memory area of a UHF RFID tag.
Write data to transponder	Writes data to any memory area of a UHF RFID tag.
Write data to transponder and verify	Writes data to any memory area of a UHF RFID tag and then verifies the write command with an automatic read command.
Process a (universal RFID interface) request	Executes any command of the RFID-U interface.

7.11.3 Command: Process an inventory

URL

GET http://192.168.24.79:32760/api/v1/rfid/inventory?timeout=0

Parameter	Meaning
Timeout	Time in ms in which the command is to be executed. If a command is not executed within the specified time, the device outputs a fault signal. 0: No timeout, command remains active until the first tag has been read. 1: Command is executed once (if there is already a tag in the detection range). >1…65535: time in ms; UHF inventory: Command remains active for the entire specified time.

Overview of input data

Response	
Loop counter	See description of the input data
Response code	0x0001 (hex.), 1 (dec.)
Length	Length of the read data
Error code	See description of the input data
Tag in detection range	See description of the input data
Data (bytes) available	See description of the input data
Tag counter	Ascending
Write fragment no.	0
Read fragment no.	See description of the input data
Read data, byte 0…n	See example: UHF read data

Data format in UHF applications

The UHF read data is formatted by a header. The header has the following structure:

Type	Name	Meaning
uint8_t	Size	Data size
uint8_t	Block type	1: EPC etc. other values: reserved
uint8_t	Data [size]	EPC and read data

The size of EPC/RSSI etc. is dependent on the reader settings.

Example: UHF read data (header and EPC, grouping with RSSI activated)

Type	Name	Meaning
uint8_t	Size	16
uint8_t	Block type	1
uint8_t	Data [20]	uint8_t EPC [12] uint16_t RSSI [2] uint16_t Number of the antenna (LSB → MSB) [2] uint16_t Number of the read operations (LSB → MSB) [2]

Byte	Content	Meaning
0	Data size (EPC + number of read operations)	2 byte header
1	UHF memory range	
3…13	EPC	12 bytes EPC
14	LSB	2 bytes RSSI
15	MSB	
16	LSB	2 bytes Number of the antenna: ■ 0: RHCP ■ 1: LHCP ■ 2: Horizontal ■ 3: Vertical ■ 4: External 1 ■ 5: External 2 ■ 6: External 3 ■ 7: External 4
17	MSB	
18	LSB	2 bytes Number of read operations
19	MSB	

Reading out the RSSI value

The RSSI value is output in binary code in 2 bytes and corresponds to the two's complement of the output binary code. Mapped to a signed integer, the 2 bytes output correspond to ten times the actual RSSI value. Refer to the following table for an example of the RSSI value:

MSB…LSB (decimal)	MSB…LSB (binary)	Two's complement	RSSI (dBm)
252 253	11111100 11111101	-771	-77.1

7.11.4 Command: Read

URL

POST http://192.168.24.79:32760/api/v1/rfid/read

Request Body

```
{
"length": "<integer>",
"memoryArea": "<integer>",
"address": "<integer>",
"sizeOfUid": "<integer>",
"timeout": "<integer>",
"data": [
"<integer>",
"<integer>"
]
}
```

Overview of output data

Request	
Memory area (DOM)	0: Kill password 1: EPC 2: TID 3: USER area 4: Access password 5: PC (defines the response length of the EPC)
Length of EPC	The EPC size should be entered in bytes if a particular tag is to be read. The EPC must be defined in the write data (start byte: 0). The function of the length of the EPC depends on the command used. 0: No entry of an EPC for executing the command. Only one tag may be in the detection range of the read/write head. >0: EPC length of the tag that is to be read if an EPC is present in the write data.
Start address	Start address of the memory area on the tag that is to be read (specification in bytes)
Length	Length of the data to be read in bytes
Command timeout	See description of the output data
Write data, byte 0…(size of the EPC-1)	EPC of the tag to be read
Write data, byte (size of the EPC)…127	Not required

Response

```
{
"data": [
"<integer>",
"<integer>"
]
}
```

7.11.5 Command: Write

URL

POST http://192.168.24.79:32760/api/v1/rfid/write

Request Body

```
{
"data": [
"<integer>",
"<integer>"
],
"memoryArea": "<integer>",
"address": "<integer>",
"sizeOfUid": "<integer>",
"timeout": "<integer>"
}
```

Overview of output data

Request	
Memory area (DOM)	0: Kill password 1: EPC 2: TID 3: USER area 4: Access password 5: PC (defines the response length of the EPC)
Length of EPC	The EPC size should be entered in bytes if a particular tag is to be written. The EPC must be defined in the write data (start byte: 0). The function of the length of the EPC depends on the command used. 0: No entry of an EPC for executing the command. Only one tag may be in the detection range of the read/write head. >0: EPC length of the tag that is to be written to if an EPC is present in the write data.
Start address	Start address of the memory area on the tag that is to be written to (specification in bytes)
Length	Length of data to be written in bytes
Command timeout	See description of the output data
Write data, byte 0…(size of the EPC-1)	EPC of the tag to be written to
Write data, byte (size of the EPC)…127	Write data

Response


```
{
"data": [ ],
"errorCode": 0,
"length": 0,
"status": {
"code": 4,
"tagCounter": 1
}
}
```

7.11.6 Command: Write and verify

URL

POST http://192.168.24.79:32760/api/v1/rfid/writeandverify

Request Body

```
{
"data": [
"<integer>",
"<integer>"
],
"memoryArea": "<integer>",
"address": "<integer>",
"sizeOfUid": "<integer>",
"timeout": "<integer>"
}
```

Overview of output data

Request	
Memory area (DOM)	0: Kill password 1: EPC 2: TID 3: USER area 4: Access password 5: PC (defines the response length of the EPC)
Length of EPC	The EPC size should be entered in bytes if a particular tag is to be written. The EPC must be defined in the write data (start byte: 0). The function of the length of the EPC depends on the command used. 0: No entry of an EPC for executing the command. Only one tag may be in the detection range of the read/write head. >0: EPC length of the tag that is to be written to if an EPC is present in the write data.
Start address	Start address of the memory area on the tag that is to be written to (specified in bytes)
Length	Length of data to be written in bytes
Command timeout	See description of the output data
Write data, byte 0…(size of the EPC-1)	EPC of the tag to be written to
Write data, byte (size of the EPC)…127	Write data

Response


```
{
"data": [ ],
"errorCode": 0,
"length": 0,
"status": {
"code": 8,
"tagCounter": 1
}
}
```


7.11.7 Command: Process a (universal RFID interface) request

URL

POST http://192.168.24.79:32760/api/v1/rfid/command

Request Body


```
{
"commandCode": "<integer>",
"memoryArea": "<integer>",
"length": "<integer>",
"address": "<integer>",
"sizeOfUid": "<integer>",
"timeout": "<integer>",
"data": [
"<integer>",
"<integer>"
]
}
```

7.12 RFID channels — overview of the commands

RFID commands are initiated via the command code in the process output data of an RFID channel. The commands can be executed with or without a loop counter function. The loop counter must be set individually for each new command.

NOTE
After commands are executed without the loop counter function, the device must be reset to the Idle state before a new command is sent.

▶ After a command is executed, send an idle command to the device.

Command	Command code		Possible for	
	hex.	dec.	UHF compact	UHF extended
Idle	0x0000	0	x	x
Inventory	0x0001	1	x	x
Fast inventory	0x2001	8193	x	x
Read	0x0002	2	x	x
Fast read	0x2002	8194	x	x
Write	0x0004	4	x	x
Fast write	0x2004	8196	x	x
Write and verify	0x0008	8	x	x
Continuous mode	0x0010	16	–	x
Get data from buffer (Continuous mode)	0x0011	17	Max. 128 bytes	x
Get data from buffer with fast command processing (Continuous mode)	0x2011	8209	Max. 128 bytes	x
Continuous presence sensing mode	0x0020	32	–	x
End Continuous (presence sensing) mode	0x0012	18	–	x
Read/write head identification	0x0041	65	x	x
Direct read/write head command	0x0060	96	x	x
Direct read/write head command with fast command processing	0x2060	8288	x	x
Set tag password	0x0102	258	x	x
Set tag password with fast command processing	0x2102	8450	x	x
Set read/write head password	0x0100	256	x	x
Reset read/write head password	0x0101	257	x	x
Set tag protection	0x0103	259	x	x
Set tag protection with fast command processing	0x2103	8451	x	x
Set permanent lock (Lock)	0x0105	261	x	x
Set permanent lock with fast command processing	0x2105	8453	x	x
Tag info	0x0050	80	x	x

Command	Command code		Possible for	
	hex.	dec.	UHF compact	UHF extended
Tag info with fast command processing	0x2050	8272	x	x
Kill UHF tag	0x0200	512	x	x
Kill UHF tag with fast command processing	0x2200	8704	x	x
Restore UHF read/write head settings	0x1000	4096	x	x
Backup settings of the UHF read/write head	0x1001	4097	x	x
Query error/status of UHF read/write head	0x0042	66	x	x
Reset	0x8000	32768	x	x

7.12.1 Command: Idle

The **Idle** command switches the interface to idle mode. Command execution is canceled. The EPC is displayed if the reader is configured in Presence Sensing Mode via TAS or the web server.

Overview of output data

Request	
Loop counter	Not required
Command code	0x0000 (hex.), 0 (dec.)
Read/write head address	Not required
EPC length	Not required
Start address	Not required
Length	Not required
Command timeout	Not required
Write fragment no.	Not required
Read fragment no.	Not required
Write data	Not required

Overview of input data

Response	
Loop counter	See description of the input data
Response code	0x0000 (hex.), 0 (dec.)
Length	EPC length of the tag in the detection range
Error code	See description of the input data
Tag in detection range	See description of the input data
Data (bytes) available	See description of the input data
Tag counter	See description of the input data
Write fragment no.	Size of the fragment
Read fragment no.	Size of the fragment
Read data, byte 0…n	EPC of the tag in the detection range

7.12.2 Command: Inventory

The **Inventory** command triggers the reader to search for tags in the detection range and to read the EPC or, if activated in the UHF reader, the RSSI of the tags. The inventory command can be executed in single-tag mode and in multi-tag mode.

NOTE
The command code for rapid processing with the loop counter is 0x2001 (hex.) or 8193 (dec.).

Overview of output data

Request	
Loop counter	See description of the output data
Command code	0x0001 (hex.), 1 (dec.)
Read/write head address	See description of the output data
EPC length	Not required
Start address	1: Grouping of the EPCs active 0: Grouping of the EPCs inactive
Length	0: The actual length (bytes) of the transferred EPC is transferred with an inventory. > 0: EPC is output in full.
Command timeout	See description of the output data
Write fragment no.	0
Read fragment no.	See description of the output data
Write data	Not required

Overview of input data

Response	
Loop counter	See description of the input data
Response code	0x0001 (hex.), 1 (dec.)
Length	Length of the read data
Error code	See description of the input data
Tag in detection range	See description of the input data
Data (bytes) available	See description of the input data
Tag counter	Ascending
Write fragment no.	0
Read fragment no.	See description of the input data
Read data, byte 0…n	See example: UHF read data

Data format in UHF applications

The UHF read data is formatted by a header. The header has the following structure:

Type	Name	Meaning
uint8_t	Size	Data size
uint8_t	Block type	1: EPC etc. other values: reserved
uint8_t	Data [size]	EPC and read data

The size of EPC/RSSI etc. is dependent on the reader settings.

Reading out the RSSI value

The RSSI value is output in binary code in 2 bytes and corresponds to the two's complement of the output binary code. Mapped to a signed integer, the 2 bytes output correspond to ten times the actual RSSI value. Refer to the following table for an example of the RSSI value:

MSB…LSB (decimal)	MSB…LSB (binary)	Two's complement	RSSI (dBm)
252 253	11111100 11111101	-771	-77.1

Example: UHF read data (header and EPC, grouping with RSSI activated)

Type	Name	Meaning
uint8_t	Size	16
uint8_t	Block type	1
uint8_t	Data [20]	uint8_t EPC [12] uint16_t RSSI [2] uint16_t Number of the antenna (LSB → MSB) [2] uint16_t Number of the read operations (LSB → MSB) [2]

Byte	Content	Meaning
0	Data size (EPC + number of read operations)	2 byte header
1	UHF memory range	
3…13	EPC	12 bytes EPC
14	LSB	2 bytes RSSI
15	MSB	
16	LSB	2 bytes Number of the antenna: ■ 0: RHCP ■ 1: LHCP ■ 2: Horizontal ■ 3: Vertical ■ 4: External 1 ■ 5: External 2 ■ 6: External 3 ■ 7: External 4
17	MSB	
18	LSB	2 bytes Number of read operations
19	MSB	

7.12.3 Command: Read

The **Read** command is used by the reader to read data of tags in the detection range. 128 bytes are transferred by default in a read process. Larger data quantities can be transferred in fragments. If a specific EPC is specified, the reader reads the corresponding tags only. All other tags in the detection range are ignored in this case.

NOTE
The command code for rapid processing with the loop counter is 0x2002 (hex.) or 8194 (dec.).

Overview of output data

Request	
Loop counter	See description of the output data
Command code	0x0002 (hex.), 2 (dec.)
Memory area	See description of the output data
Read/write head address	See description of the output data
EPC length	The EPC size should be entered in bytes if a particular tag is to be read. The EPC must be defined in the write data (start byte: 0). The function of the EPC length depends on the command used. 0: No entry of a EPC for executing the command. Only one tag may be in the detection range of the read/write head. > 0: EPC length of the tag to be read if a EPC is present in the write data.
Start address	Start address of the memory area on the tag that is to be read (specification in bytes)
Length	Length of the data to be read in bytes
Command timeout	See description of the output data
Write fragment no.	0
Read fragment no.	See description of the output data
Write data, byte 0… (EPC size-1)	EPC of the tag to be read
Write data, byte (EPC size)…127	Not required

Overview of input data

Response	
Loop counter	See description of the input data
Response code	0x0002 (hex.), 2 (dec.)
Length	Length of the read data
Error code	See description of the input data
Tag in detection range	See description of the input data
Data (bytes) available	increases during command execution
Tag counter	See description of the input data
Write fragment no.	0
Read fragment no.	See description of the input data
Read data, byte 0…n	read data

7.12.4 Command: Write

The **Write** command is used by the reader to write data to tags in the detection range. 128 bytes are transferred in a write operation by default. Larger data quantities can be transferred in fragments. If a specific EPC is specified, the reader writes the corresponding tags only. All other tags in the detection range are ignored in this case.

NOTE

▶ With multi-tag applications, specify the EPC of the tag that is to be written.

NOTE

The command code for rapid processing with the loop counter is 0x2004 (hex.) or 8196 (dec.).

Overview of output data

Request	
Loop counter	See description of the output data
Command code	0x0004 (hex.), 4 (dec.)
Memory area	See description of the output data
Read/write head address	See description of the output data
EPC length	The EPC size should be entered in bytes if a particular tag is to be written. The EPC must be defined in the write data (start byte: 0). The function of the EPC length depends on the command used. 0: No entry of a EPC for executing the command. Only one tag may be in the detection range of the read/write head. > 0: EPC length of the tag to be written if a EPC is present in the write data.
Start address	Start address of the memory area on the destination tag (specified in bytes)
Length	Length of data to be written in bytes
Command timeout	See description of the output data
Write fragment no.	1: Using fragmentation 0: Do not use fragmentation
Read fragment no.	0
Write data, byte 0…(size of the EPC-1)	EPC of the tag to be written
Write data, byte (size of the EPC)...127	Write data

Overview of input data

Response	
Loop counter	See description of the input data
Response code	0x0004 (hex.), 4 (dec.)
Length	Length of the read data
Error code	See description of the input data
Tag in detection range	See description of the input data
Data (bytes) available	increases during command execution
Tag counter	See description of the input data
Write fragment no.	See description of the input data
Read fragment no.	0
Read data, byte 0...127	Not required

7.12.5 Command: Write and verify

The **Write with validation** command writes a number of bytes defined by the user. The data written is also sent back to the interface and validated. When writing, up to 128 bytes are transferred by default. Larger data quantities can be transferred in fragments. The data written is validated in the interface only, and not sent back to the controller. If the validation fails, a fault signal is output. If the command is processed without a fault signal, the data has been validated successfully.

NOTE

▶ With multi-tag applications, specify the EPC of the tag that is to be written.

NOTE

The command code for rapid processing with the loop counter is 0x2008 (hex.) or 8200 (dec.).

Overview of output data

Request	
Loop counter	See description of the output data
Command code	0x0008 (hex.), 8 (dec.)
Memory area	See description of the output data
Read/write head address	See description of the output data
EPC length	The EPC size should be entered in bytes if a particular tag is to be written. The EPC must be defined in the write data (start byte: 0). The function of the length of the EPC depends on the command used. 0: No entry of a EPC for executing the command. Only one tag can be located in the detection range of the reader. > 0: EPC length of the tag to be written if a EPC is present in the write data.
Start address	Start address of the memory area on the destination tag (specified in bytes)
Length	Length of data to be written in bytes
Command timeout	See description of the output data
Write fragment no.	1: Using fragmentation 0: Do not use fragmentation
Read fragment no.	0
Write data, byte 0… (EPC size-1)	Optional: EPC of the tag to be written
Write data, byte (EPC size)…127	Write data

Overview of input data

Response	
Loop counter	See description of the input data
Response code	0x0008 (hex.), 8 (dec.)
Length	Length of the read data
Error code	See description of the input data
Tag in detection range	See description of the input data
Data (bytes) available	increases during command execution
Tag counter	See description of the input data
Write fragment no.	See description of the input data
Read fragment no.	0
Read data, Byte 0…MIN(127, set length-1)	Not required

7.12.6 Command: Continuous mode

In Continuous Mode, a user-defined command is sent to the reader and stored in the reader. The write, read and inventory commands can be executed in continuous mode. The parameters for Continuous Mode must be set direct in the reader.

The command is executed continuously until the user stops continuous mode. Continuous mode can be stopped with a reset command.

NOTE
The reset command resets all read data.

Readers in Continuous Mode send all command-specific data to the interface. The data is stored in the FIFO memory of the interface and can be queried by the controller via the **Get Data from FIFO** command.

Commands in Continuous Mode are triggered if the reader detects a tag. If there is a tag in the detection range of the reader when starting Continuous Mode, the command sent in Continuous Mode is not executed until the next tag.

NOTE
In Continuous mode the **Tag in detection range** signal is not updated.
Start address and length cannot be changed during the execution of Continuous mode.
After a restart of continuous mode, all data of the continuous mode already running is deleted.

Overview of output data

Request	
Loop counter	See description of the output data
Command code	0x0010 (hex.), 16 (dec.)
Read/write head address	See description of the output data
EPC length	Not required
Start address	1: Grouping of the EPCs active (UHF inventory only) 0: Grouping of the EPCs inactive (UHF inventory only) >1: not defined
Length	Not required
Command timeout	Not required
Write fragment no.	0
Read fragment no.	See description of the output data
Write data	Not required

Overview of input data

Response	
Loop counter	See description of the input data
Response code	0x0010 (hex.), 16 (dec.)
Length	0
Error code	See description of the input data
Tag in detection range	See description of the input data
Data (bytes) available	increases during command execution
Tag counter	increases with each read or written EPC
Write fragment no.	0
Read fragment no.	See description of the input data
Read data	See description of the input data

7.12.7 Command: Get data from buffer (Continuous mode)

NOTE
The command code for fast processing with the loop counter is 0x2011 (hex.) or 8209 (dec.).

The **Get data from buffer (Continuous Mode)** command passes on data stored in the interface to the controller. The command is required to transfer read data to the controller in continuous mode or in continuous presence sensing mode. The data is transferred to the controller in fragments of up to 128 bytes. The size of the fragments can be set by the user. A EPC is not divided by fragment limits. If a EPC does not fit completely in a fragment, it is automatically moved to the next fragment.

NOTE
The **Get data from buffer** command does not end Continuous mode.

Overview of output data

Request	
Loop counter	See description of the output data
Command code	0x0011 (hex.), 17 (dec.)
Read/write head address	See description of the output data
EPC length	Not required
Start address	Not required
Length	Max. length of the data to be read by the device (≤ size of the data that the device has actually stored), entered in bytes
Command timeout	See description of the output data
Write fragment no.	0
Read fragment no.	See description of the output data
Write data	Not required

Overview of input data

Response	
Loop counter	See description of the input data
Response code	0x0011 (hex.), 17 (dec.)
Length	Length of the read data. The data is specified in complete blocks.
Error code	See description of the input data
Tag in detection range	See description of the input data
Data (bytes) available	is reduced automatically after the command execution
Tag counter	See description of the input data
Write fragment no.	0
Read fragment no.	See description of the input data
Read data	Read data

Data format in UHF applications

The UHF read data is formatted by a header. The header has the following structure:

Type	Name	Meaning
uint8_t	Size	Data size
uint8_t	Block type	1: EPC etc. other values: reserved
uint8_t	Data [size]	EPC and read data

The size of EPC/RSSI etc. is dependent on the reader settings.

7.12.8 Command: UHF continuous presence sensing mode

In Continuous Presence Sensing Mode, a user-defined command (write, read, inventory) is sent to the UHF reader and stored 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. The duration of the query interval and the duty cycle can be adapted in the settings of the UHF reader. The command is executed continuously until the user ends the Continuous presence sensing mode by executing a reset command.

NOTE
The reset command resets all read data.

Readers in Continuous Presence Sensing Mode send all command-specific data to the interface. The data is stored in the buffer of the interface and can be queried by the controller via the **Get data from buffer** command. In Continuous presence sensing mode the **Tag in detection range** signal is not permanently updated.

Overview of output data

Request	
Loop counter	See description of the output data
Command code	0x0020 (hex.), 32 (dec.)
Read/write head address	See description of the output data
Length of EPC	Not required
Start address	0: Grouping inactive 1: Grouping active >1: not defined
Length	Not required
Command timeout	Not required
Write fragment no.	0
Read fragment no.	See description of the output data
Write data	Not required

Overview of input data

Response	
Loop counter	See description of the input data
Response code	0x0020 (hex.), 32 (dec.)
Length	Not required
Error code	See description of the input data
Tag in detection range	See description of the input data
Data (bytes) available	increases during command execution
Tag counter	Increases with each read or written EPC
Write fragment no.	0
Read fragment no.	See description of the input data
Read data	See description of the input data

7.12.9 Command: End Continuous (presence sensing) mode

Continuous mode and presence sensing mode can be stopped via the **Shut down Continuous (presence sensing) mode** command. The data in the buffer of the interface is not deleted after the command is executed and can still be called up via the **Get data from buffer** command.

Overview of output data

Request	
Loop counter	See description of the output data
Command code	0x0012 (hex.), 18 (dec.)
Read/write head address	Not required
EPC length	Not required
Start address	Not required
Length	Not required
Command timeout	See description of the output data
Write fragment no.	0
Read fragment no.	See description of the output data
Write data	Not required

Overview of input data

Response	
Loop counter	See description of the input data
Response code	0x0012 (hex.), 18 (dec.)
Length	Not required
Error code	See description of the input data
Tag in detection range	See description of the input data
Data (bytes) available	See description of the input data
Tag counter	See description of the input data
Write fragment no.	0
Read fragment no.	See description of the input data
Read data	Not required

7.12.10 Command: Read/write head identification

The **Read/write head identification** command scans the following parameters of the connected reader:

- ID
- Serial number
- Hardware version
- Firmware version

The parameters are summarized in the reader in the identification record.

Overview of output data

Request	
Loop counter	See description of the output data
Command code	0x0041 (hex.), 65 (dec.)
Read/write head address	See description of the output data
EPC length	Not required
Start address	Start address in the identification record, specification in bytes
Length	Length of the data to be queried 0: Read full parameter set
Command timeout	Not required
Write fragment no.	Not required
Read fragment no.	See description of the output data
Write data	Not required

Overview of input data

Response	
Loop counter	See description of the input data
Response code	0x0041 (hex.), 65 (dec.)
Length	See description of the input data
Error code	See description of the input data
Tag in detection range	See description of the input data
Data (bytes) available	See description of the input data
Tag counter	increases with each read or written EPC
Write fragment no.	0
Read fragment no.	See description of the input data
Read data, byte 0...19	ID: ARRAY [0…19] of BYTE
Read data, byte 20...35	Serial number: ARRAY [0…15] of BYTE
Read data, byte 36...37	Hardware version: INT16 (Little Endian)
Read data, byte 38...41	Firmware status: ARRAY [0…] of BYTE: V (0x56), x, y, z (Vx.y.z)
Read data, byte 42...119	Not required

7.12.11 Direct read/write head command

NOTE
The command code for fast processing with the loop counter is 0x2060 (hex.) or 8288 (dec.).

Commands from the reader protocol can be sent direct to the reader via a direct command. The commands are defined and interpreted via specifications in the read and write data.

NOTE
The reader protocol is not part of this documentation and has to be requested from and specially released by TURCK. Questions on the reader protocol should be addressed to TURCK.

Overview of output data

Request	
Loop counter	See description of the output data
Command code	0x0060 (hex.), 96 (dec.)
Read/write head address	See description of the output data
EPC length	0
Start address	Not required
Length	Length of the description of the direct command in the write data, specification in bytes
Command timeout	See description of the output data
Write fragment no.	0
Read fragment no.	See description of the output data
Write data	Description of the direct command

Overview of input data

Response	
Loop counter	See description of the input data
Response code	0x0060 (hex.), 96 (dec.)
Length	Length of the description of the direct command in the write data
Error code	See description of the input data
Tag in detection range	See description of the input data
Data (bytes) available	See description of the input data
Tag counter	See description of the input data
Write fragment no.	0
Read fragment no.	See description of the input data
Read data	Response to the direct command

Example: Direct command in UHF applications (query reader version)

Request	
Loop counter	0
Command code	0x0060
Read/write head address	0
Length of EPC	0
Start address	0
Length	2
Command timeout	200
Write fragment no.	0
Read fragment no.	0
Write data	0x02 (CMD), 0x00 (application) — see debus protocol

Response	
Loop counter	0
Response code	0x0060
Length	12
Error code	0
Tag in detection range	0
Data (bytes) available	0
Tag counter	0
Write fragment no.	0
Read fragment no.	0
Read data	0x02, 0x00, 0x01, 0x02, 0x03, 0x04, 0x8B, 0x20, 0x00, 0x01, 0x00, 0x01

The read data can be interpreted via the debus protocol as follows:

MSG	**ERR**	**SNR0**	**SNR1**	**SNR2**	**SNR3**	**GTYP**	**VERS**	**HW**
0x02	0x00	0x01	0x02	0x03	0x04	0x8B 0x20	0x00 0x01	0x00 0x01

- Serial number: 0x01020304
- Device type: 0x208B
- Software version: v1.00
- Hardware version: v1.00

7.12.12 Command: Set tag password

NOTE
The command code for fast processing with the loop counter is 0x2102 (hex.) or 8450 (dec.).

The **Set tag password** command sets a password in the tag. When sending the command, only one tag can be located in the detection range of the reader. After the password is sent, other commands (e.g. **Set tag protection**) can be sent to the tag. The **Set tag password** command prevents a kill password from being set in the tag.

Overview of output data

Request	
Loop counter	See description of the output data
Command code	0x0102 (hex.), 258 (dec.)
Read/write head address	See description of the output data
EPC length	The EPC size should be entered in bytes if a particular tag is to be protected. The EPC must be defined in the write data (start byte: 0). The function of the EPC length depends on the command used. 0: No entry of a EPC for executing the command. Only one tag may be in the detection range of the read/write head. > 0: EPC length of the tag to be protected if a EPC is present in the write data.
Start address	Not required
Length	4 bytes
Command timeout	See description of the output data
Write fragment no.	0
Read fragment no.	See description of the output data
Write data, byte 0…3	Password: ARRAY [0…3] OF BYTE
Write data, byte 4…127	Not required

Overview of input data

Response	
Loop counter	See description of the input data
Response code	0x0102 (hex.), 258 (dec.)
Length	Not required
Error code	See description of the input data
Tag in detection range	See description of the input data
Data (bytes) available	See description of the input data
Tag counter	See description of the input data
Write fragment no.	0
Read fragment no.	See description of the input data
Read data	Not required

7.12.13 Command: Set read/write head password

The **Set read/write head password** command directly sets a password for write access, read access or a kill command in the tag. The password is stored temporarily in the memory of the reader. After a voltage reset of the reader, the password must be set again in the reader. With UHF applications, the password is stored in the memory of the interface.

Overview of output data

Request	
Loop counter	See description of the output data
Command code	0x0100 (hex.), 256 (dec.)
Read/write head address	See description of the output data
EPC length	Not required
Start address	Not required
Length	Not required
Command timeout	See description of the output data
Write fragment no.	0
Read fragment no.	See description of the output data
Write data, byte 0…3	Password: ARRAY [0…3] OF BYTE
Write data, byte 4…127	Not required

Overview of input data

Response	
Loop counter	See description of the input data
Response code	0x0100 (hex.), 256 (dec.)
Length	Not required
Error code	See description of the input data
Tag in detection range	See description of the input data
Data (bytes) available	See description of the input data
Tag counter	See description of the input data
Write fragment no.	0
Read fragment no.	See description of the input data
Read data	Not required

7.12.14 Command: Reset read/write head password

The **Reset read/write head password** command directly resets a password for write access, read access or a kill command in the reader. The password function is switched off and passwords are no longer exchanged between the reader and the password function.

Overview of output data

Request	
Loop counter	See description of the output data
Command code	0x0101 (hex.), 257 (dec.)
Read/write head address	See description of the output data
EPC length	Not required
Start address	Not required
Length	Not required
Command timeout	See description of the output data
Write fragment no.	0
Read fragment no.	See description of the output data
Write data	Not required

Overview of input data

Response	
Loop counter	See description of the input data
Response code	0x0101 (hex.), 257 (dec.)
Length	Not required
Error code	See description of the input data
Tag in detection range	See description of the input data
Data (bytes) available	See description of the input data
Tag counter	See description of the input data
Write fragment no.	0
Read fragment no.	See description of the input data
Read data	Not required

7.12.15 Command: Set tag protection

NOTE
The command code for rapid processing with the loop counter is 0x2103 (hex.) or 8451 (dec.).

The **Set tag protection** command is a direct command used to define the password protection for the tag. To do this, it must be specified whether read protection and/or write protection is to be set, and to which area of the tag the password applies. Protection for all areas is defined with one command. When sending the command, only one tag can be located in the detection range of the reader.

Read protection also always includes write protection.

NOTE
Write protection for UHF tags cannot be reversed.

Overview of output data

Request	
Loop counter	See description of the output data
Command code	0x0103 (hex.), 259 (dec.)
Read/write head address	See description of the output data
EPC length	The EPC size should be entered in bytes if a particular tag is to be protected. The EPC must be defined in the write data (start byte: 0). The function of the length of the EPC depends on the command used. 0: The command is executed for the tag that is in the detection range of the read/write head. > 0: EPC length of the tag to be protected if a EPC is present in the write data.
Start address	Not required
Memory area	Possible values: ■ PC and EPC (memory area 1) ■ USER memory (memory area 3) The entire memory selected is protected with a password.
Length	0 byte
Command timeout	See description of the output data
Write fragment no.	0
Read fragment no.	See description of the output data
Write data, byte 0	Not required
Write data, byte 1	0
Write data, byte 2	0
Write data, byte 3	0
Write data, byte 4	Not required
Write data, byte 5	0
Write data, byte 6	0
Write data, byte 7	0
Write data, byte 8…127	Not required

Overview of input data

Response	
Loop counter	See description of the input data
Response code	0x0103 (hex.), 259 (dec.)
Length	Not required
Error code	See description of the input data
Tag in detection range	See description of the input data
Data (bytes) available	See description of the input data
Tag counter	See description of the input data
Write fragment no.	0
Read fragment no.	See description of the input data
Read data	Not required

7.12.16 Command: Tag info

NOTE
The command code for rapid processing with the loop counter is 0x2050 (hex.) or 8272 (dec.).

The **Tag info** command enables the following chip information of a tag to be scanned:

- Allocation class identifier
- Tag mask designer identifier
- Tag Model Number

The data is queried from the GSI record of the tag.

Overview of output data

Request	
Loop counter	See description of the output data
Command code	0x0050 (hex.), 80 (dec.)
Read/write head address	See description of the output data
EPC length	Not required
Start address	Start address in the GSI record
Length	Length of the system data read (bytes) 0: All system data is read
Command timeout	Not required
Write fragment no.	Not required
Read fragment no.	See description of the output data
Write data	Not required

Overview of input data

Response	
Loop counter	See description of the input data
Response code	0x0050 (hex.), 80 (dec.)
Length	See description of the input data
Error code	See description of the input data
Tag in detection range	See description of the input data
Data (bytes) available	See description of the input data
Tag counter	See description of the input data
Write fragment no.	0
Read fragment no.	See description of the input data
Read data, bytes 0…3	First 32 bytes of the TID (tag class, manufacturer, and chip type)
Read data, bytes 4…n	EPC (variable length)

Chip Information on the UHF Tags

Name	TID memory			Size (bits)		
	Allocation class identifier	Tag mask designer	Tag Model Number	EPC	TID	USER
Alien Higgs-3	0xE2	0x003	0x412	96…480	96	512
Alien Higgs-4	0xE2	0x003	0x414	16…128	96	128
NXP U-Code G2XM	0xE2	0x006	0x003	240	64	512
NXP U-Code G2XL	0xE2	0x006	0x004	240	64	–
NXP U-Code G2iM	0xE2	0x006	0x80A	256	96	512
NXP U-Code G2iM+	0xE2	0x006	0x80B	128…448	96	640…320
NXP U-Code G2iL	0xE2	0x006	0x806, 0x906, 0xB06	128	64	–
NXP U-Code G2iL+	0xE2	0x006	0x807, 0x907, 0xB07	128	64	–
NXP U-Code 7	0xE2	0x806	0x890	128	96	–
NXP U-Code 7xm (2k)	0xE2	0x806	0xF12	448	96	2048
Impinj Monza 4E	0xE2	0x001	0x10C	496	96	128
Impinj Monza 4D	0xE2	0x001	0x100	128	96	32
Impinj Monza 4QT	0xE2	0x001	0x105	128	96	512
Impinj Monza 5	0xE2	0x001	0x130	128	96	–
Impinj Monza R6	0xE2	0x001	0x160	96	96	–
Impinj Monza R6-P	0xE2	0x001	0x170	128	96	64

7.12.17 Command: Permanently deactivate UHF tags (Kill)

NOTE
The command code for rapid processing with the loop counter is 0x2200 (hex.) or 8704 (dec.).

The **Kill UHF tag** command makes the tag memory unusable. After a kill command, the tag can neither be read nor written. A kill command cannot be reversed.

Overview of output data

Request	
Loop counter	See description of the output data
Command code	0x0200 (hex.), 512 (dec.)
Read/write head address	See description of the output data
EPC length	Enter the EPC size in bytes if a particular tag is to be deleted. The EPC must be defined in the write data (start byte: 0). The function of the length of the EPC depends on the command used. 0: No entry of an EPC for executing the command. Only one tag can be located in the detection range of the reader. > 0: EPC length of the tag that is to be deleted if an EPC is present in the write data.
Start address	Not required
Length	1 byte
Command timeout	See description of the output data
Write fragment no.	0
Read fragment no.	See description of the output data
Write data, byte 0…3	Password: ARRAY [0…3] OF BYTE
Write data, byte 4…127	Not required

Overview of input data

Response	
Loop counter	See description of the input data
Response code	0x0200 (hex.), 512 (dec.)
Length	Not required
Error code	See description of the input data
Tag in detection range	See description of the input data
Data (bytes) available	See description of the input data
Tag counter	See description of the input data
Write fragment no.	0
Read fragment no.	See description of the input data
Read data	Not required

7.12.18 Command: Restore UHF read/write head settings

The **Restore UHF read/write head settings** command restores the parameters of the UHF reader from a backup. To execute the command, a backup must be created beforehand via the **Backup of the settings of the UHF read/write head** command.

Overview of output data

Request	
Loop counter	See description of the output data
Command code	0x1000 (hex.), 4096 (dec.)
Read/write head address	See description of the output data
EPC length	Not required
Start address	Not required
Length	Not required
Command timeout	See description of the output data
Write fragment no.	0
Read fragment no.	See description of the output data
Write data	Not required

Overview of input data

Response	
Loop counter	See description of the input data
Response code	0x1000 (hex.), 4096 (dec.)
Length	Not required
Error code	See description of the input data
Tag in detection range	See description of the input data
Data (bytes) available	See description of the input data
Tag counter	See description of the input data
Write fragment no.	0
Read fragment no.	See description of the input data
Read data	Not required

7.12.19 Command: Backup settings of the UHF read/write head

The **Backup of the settings of the UHF read/write head** command stores the current settings of the reader in the memory of the interface. The backup is retained even after a voltage reset. The backup data can be restored using the **Restore UHF read/write head settings** command.

Overview of output data

Request	
Loop counter	See description of the output data
Command code	0x1001(hex.), 4097 (dec.)
Read/write head address	See description of the output data
EPC length	Not required
Start address	Not required
Length	Not required
Command timeout	See description of the output data
Write fragment no.	0
Read fragment no.	See description of the output data
Write data	Not required

Overview of input data

Response	
Loop counter	See description of the input data
Response code	0x1001(hex.), 4097 (dec.)
Length	Not required
Error code	See description of the input data
Tag in detection range	See description of the input data
Data (bytes) available	See description of the input data
Tag counter	See description of the input data
Write fragment no.	0
Read fragment no.	See description of the input data
Read data	Not required

7.12.20 Command: Query error/status of UHF read/write head

Fault and status signals of the UHF reader can be read out using the **Read error/status of UHF read/write head** command.

Overview of output data

Request	
Loop counter	See description of the output data
Command code	0x0042 (hex.), 66 (dec.)
Read/write head address	Not required
EPC length	Not required
Start address	Address in the **Get Status response** record
Length	Length of the data to be read from the **Get Status response** record 0: Read entire **Get Status response** record
Command timeout	See description of the output data
Write fragment no.	0
Read fragment no.	See description of the output data
Write data	Not required

Overview of input data

Response	
Loop counter	See description of the input data
Response code	0x0042 (hex.), 66 (dec.)
Length	See description of the input data
Error code	See description of the input data
Tag in detection range	See description of the input data
Data (bytes) available	See description of the input data
Tag counter	See description of the input data
Write fragment no.	0
Read fragment no.	See description of the input data
Read data, Byte 0…(Length-1)	■ Status general: 1 byte general status ■ RF status: 1 byte status of the RF module ■ Device status: 1 byte device-specific status information ■ RF mode: 1 byte, defines the reason for starting the read operation ■ Trigger status: 1 byte, trigger number of the RF mode ■ I/O status: 1 byte, status of the inputs and outputs (0 = low, 1 = high) ■ Ambient temperature: 1 byte, ambient temperature in °C (data format: 8 bit, two's complement) ■ PA temperature: 1 byte, PA temperature in °C (data format: 8 bit, two's complement) ■ RF antenna temperature: 1 byte, antenna temperature in °C (data format: 8 bit, two's complement) ■ Transmit power: 2 bytes, output power of the reader in 1/10-dBm steps, LSB…MSB (data format: 16-bit, two's complement) ■ Reverse power: 2 bytes, reverse power in 1/10-dBm steps, LSB…MSB (data format: 16-bit, two's complement) ■ Antenna DC resistance: 4 bytes, resistance at the antenna port in Ω, LSB…MSB ■ Jammer power: 2 bytes, input power at the RX port in 1/10-dBm steps, LSB…MSB (data format: 16-bit, two's complement) ■ Channel: Number of the currently used channel (offset from the next available channel)
Read data, Byte (Length)…127	Not required

Evaluating read data – General status

Bit	Meaning
7	Reader has been reset (after reset)
6	Reader configuration damaged; default settings are used
5	Test mode active
1	Tag present

Evaluating read data – RF status

Bit	Meaning
4	Limit value for radiated power exceeded
3	No free channel present
2	Antenna resistance too high or too low
1	Reverse power too high
0	PLL not locked

Evaluating read data – Device Status

Bit	Meaning
4	Error in message generation (in Polling mode outside of memory area)
3	Temperature warning
2	Temperature too high
1	Communication error
0	Configuration invalid. Command execution not possible.

Evaluating read data – RF mode

Value	Meaning
0x00	None (tag off)
0x01	Mode 1: Trigger is digital signal (edge), Timeout
0x02	Mode 2: Trigger is digital signal (edge), Timeout
0x03	Mode 3: Trigger is digital signal (level), no timeout
0x04	Trigger is a command
0x08	Reserved
0x10	DCU-controlled read operation
0x20	Continuous Mode
0x80	Automatic trigger (presence sensing mode)

Evaluating read data – I/O status

Value	Meaning
7	Output 4
6	Output 3
5	Output 2
4	Output 1
3	Input 4
2	Input 3
1	Input 2
0	Input 1

7.12.21 Command: Reset

The **Reset** command is used to reset the reader and interface.

Overview of output data

Request	
Loop counter	See description of the output data
Command code	0x8000 (hex.), 32768 (dec.)
Read/write head address	See description of the output data
EPC length	Not required
Start address	0: Software reset 1: Voltage reset
Length	Not required
Command timeout	See description of the output data
Write fragment no.	0
Read fragment no.	See description of the output data
Write data	Not required

Overview of input data

Response	
Loop counter	See description of the input data
Response code	0x8000 (hex.), 32768 (dec.)
Length	Not required
Error code	See description of the input data
Tag in detection range	See description of the input data
Data (bytes) available	See description of the input data
Tag counter	See description of the input data
Write fragment no.	0
Read fragment no.	See description of the input data
Read data	Not required

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

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.

10 Maintenance

Ensure regularly that the plug connections and cables are in good condition.

The devices are maintenance-free, clean dry if required.

10.1 Updating the firmware via the Web server

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.
- ▶ Do not interrupt the Ethernet connection during the firmware update.

- ▶ Open the web server and log in on the device.
- ▶ Click **Firmware** → **SELECT FIRMWARE FILE**.

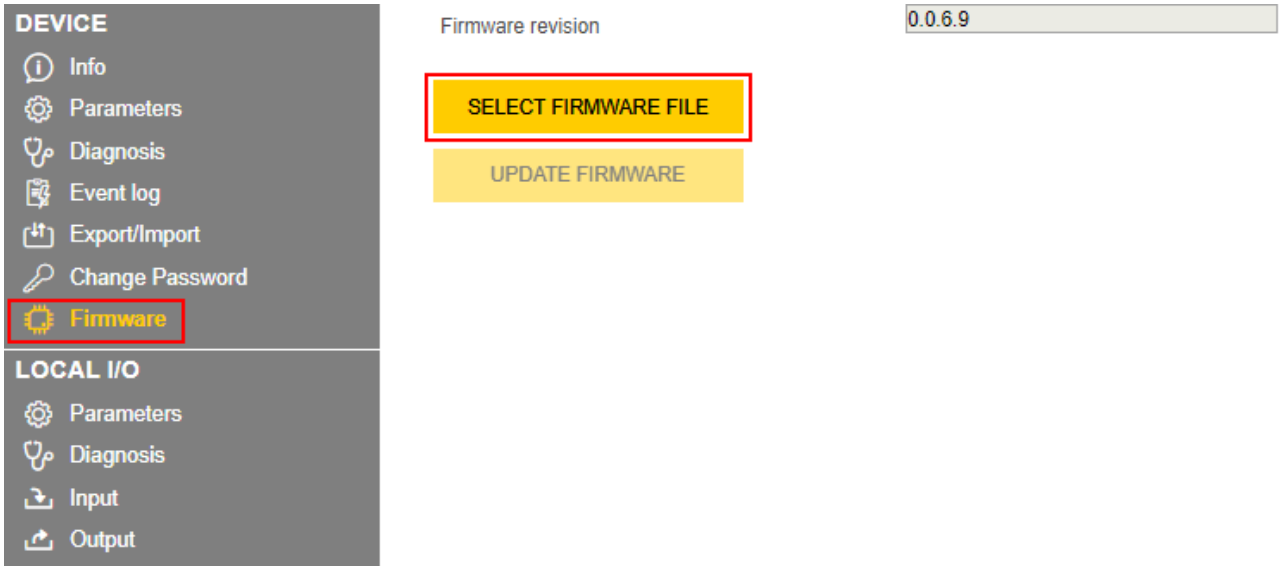


Fig. 39: Selecting the new firmware file

- ▶ Select the storage location of the file and select the file.
- ▶ Start the firmware update via the **UPDATE FIRMWARE** button.
  ⇨ The progress of the firmware update is displayed.
- ▶ After a firmware update has been successfully completed, start the device by clicking **OK**.

11 Repair

The device is not intended for repair 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

If a device has to be returned, bear in mind that only devices with a decontamination declaration will be accepted. This is available for download at
https://www.turck.de/en/return-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 properly and do not belong in the domestic waste.

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 EN 302208 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

Hereby, TURCK GmbH declares that the radio equipment type TN-UHF-Q…L…-EU… is in compliance with Directive 2014/53/EU. The full text of the EU declaration of conformity is available at the following internet address: **www.turck.com**

15 TURCK branches — contact data

Germany	TURCK GmbH Witzlebenstraße 7, 45472 Mülheim an der Ruhr www.turck.de
Australia	Turck Australia Pty Ltd Building 4, 19-25 Duerdin Street, Notting Hill, 3168 Victoria www.turck.com.au
Austria	Turck GmbH Graumanngasse 7/A5-1, A-1150 Vienna www.turck.at
Belgium	Turck Multiprox N. V. Lion d'Orweg 12, B-9300 Aalst www.multiprox.be
Brazil	Turck do Brasil Automação Ltda. Rua Anjo Custódio Nr. 42, Jardim Anália Franco, CEP 03358-040 São Paulo www.turck.com.br
Canada	Turck Canada Inc. 140 Duffield Drive, CDN-Markham, Ontario L6G 1B5 www.turck.ca
China	Turck (Tianjin) Sensor Co. Ltd. 18,4th Xinghuazhi Road, Xiqing Economic Development Area, 300381 Tianjin www.turck.com.cn
Czech Republic	TURCK s.r.o. Na Brne 2065, CZ-500 06 Hradec Králové www.turck.cz
France	TURCK BANNER S.A.S. 11 rue de Courtalin Bat C, Magny Le Hongre, F-77703 MARNE LA VALLEE Cedex 4 www.turckbanner.fr
Hungary	TURCK Hungary kft. Árpád fejedelem útja 26-28., Óbuda Gate, 2. em., H-1023 Budapest www.turck.hu
India	TURCK India Automation Pvt. Ltd. 401-403 Aurum Avenue, Survey. No 109 /4, Near Cummins Complex, Baner-Balewadi Link Rd., 411045 Pune - Maharashtra www.turck.co.in
Italy	TURCK BANNER S.R.L. Via San Domenico 5, IT-20008 Bareggio (MI) www.turckbanner.it
Japan	TURCK Japan Corporation ISM Akihabara 1F, 1-24-2, Taito, Taito-ku, 110-0016 Tokyo www.turck.jp

Korea	Turck Korea Co, Ltd. A605, 43, Iljik-ro, Gwangmyeong-si 14353 Gyeonggi-do www.turck.kr
Malaysia	Turck Banner Malaysia Sdn Bhd Unit A-23A-08, Tower A, Pinnacle Petaling Jaya, Jalan Utara C, 46200 Petaling Jaya Selangor www.turckbanner.my
Mexico	Turck Comercial, S. de RL de CV Blvd. Campestre No. 100, Parque Industrial SERVER, C.P. 25350 Arteaga, Coahuila www.turck.com.mx
Netherlands	Turck B. V. Ruiterlaan 7, NL-8019 BN Zwolle www.turck.nl
Poland	TURCK sp.z.o.o. Wroclawska 115, PL-45-836 Opole www.turck.pl
Romania	Turck Automation Romania SRL Str. Siriului nr. 6-8, Sector 1, RO-014354 Bucuresti www.turck.ro
Sweden	Turck AB Fabriksstråket 9, 433 76 Jonsered www.turck.se
Singapore	TURCK BANNER Singapore Pte. Ltd. 25 International Business Park, #04-75/77 (West Wing) German Centre, 609916 Singapore www.turckbanner.sg
South Africa	Turck Banner (Pty) Ltd Boeing Road East, Bedfordview, ZA-2007 Johannesburg www.turckbanner.co.za
Turkey	Turck Otomasyon Ticaret Limited Sirketi Inönü mah. Kayisdagi c., Yesil Konak Evleri No: 178, A Blok D:4, 34755 Kadiköy/ Istanbul www.turck.com.tr
United Kingdom	TURCK BANNER LIMITED Blenheim House, Hurricane Way, GB-SS11 8YT Wickford, Essex www.turckbanner.co.uk
USA	Turck Inc. 3000 Campus Drive, USA-MN 55441 Minneapolis www.turck.us

TURCK

Your Global Automation Partner

Over 30 subsidiaries and
60 representations worldwide!

100003067 | 2025/09

www.turck.com