

Your Global Automation Partner

**TURCK**

# TBEN-L5-4RFID-8DXP-WIN

## Compact RFID and I/O Module

Operating Instructions



# Table of Contents

<b>1</b>	<b>About these Instructions</b>	<b>7</b>
1.1	Target Groups	7
1.2	Explanation of Symbols	7
1.3	Additional Documents	7
1.4	Feedback on these Instructions	8
<b>2</b>	<b>Notes on the Product</b>	<b>9</b>
2.1	Product Identification	9
2.2	Scope of Delivery	9
2.3	Legal Requirements	9
2.4	Manufacturer and Service	9
<b>3</b>	<b>For Your Safety</b>	<b>10</b>
3.1	Intended Use	10
3.2	General Safety Instructions	10
<b>4</b>	<b>Product Description</b>	<b>11</b>
4.1	Device Overview	11
4.1.1	Display Elements	11
4.2	Properties and Characteristics	11
4.3	Functional Principle	12
4.4	Functions and Operating Modes	12
<b>5</b>	<b>Mounting</b>	<b>13</b>
5.1	Grounding the Device	14
5.1.1	Grounding and Shielding Concept	14
5.1.2	Grounding the Module (FE)	15
<b>6</b>	<b>Connecting</b>	<b>16</b>
6.1	Connecting the Modules to the Ethernet	16
6.2	Connecting the Power Supply	17
6.3	Connecting the RFID Read/Write Heads	18
6.4	Connecting Digital Sensors and Actuators	19
<b>7</b>	<b>Commissioning</b>	<b>20</b>
7.1	Setting the IP Address	20
7.1.1	Setting the IP Address via Switches on the Device	20
7.1.2	Setting the IP Address via the Turck Service Tool	22
7.2	Getting Started	25
7.2.1	Prerequisites	25
7.2.2	Creating Applications	25
7.3	Drivers	26
7.3.1	Ethernet	26
7.3.2	NAND Flash	26
7.3.3	USB Host	26
7.3.4	USB OTG	26
7.3.5	UART	27
7.3.6	GPIO	27
7.3.7	SPI	33

7.3.8	I2C	36
7.3.9	RTC	37
7.3.10	Using the Application	37
7.3.11	Debugging the Application	38
7.3.12	Using a Network Socket in C#	38
7.3.13	Using the TBOX API Library	40
7.3.14	Procedure with a C# Application	42
7.4	Specific Settings / Implementations	43
7.4.1	Autostart Application	43
7.4.2	Image Version Readout	43
7.4.3	Addressing Mode Readout	43
<b>8</b>	<b>Operation</b>	<b>44</b>
8.1	LED Indicators	44
<b>9</b>	<b>Troubleshooting</b>	<b>45</b>
<b>10</b>	<b>Maintenance</b>	<b>45</b>
10.1	Carrying out a Firmware Update	45
<b>11</b>	<b>Repairs</b>	<b>55</b>
11.1	Returning Devices	55
<b>12</b>	<b>Disposal</b>	<b>55</b>
<b>13</b>	<b>Technical Data</b>	<b>56</b>





# 1 About these Instructions

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

## 1.1 Target Groups

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

The system integrator must be familiar with implementing applications under Windows and must be capable of commissioning the device for implementation on the Windows operating system without any additional support. Knowledge of integrating RFID read/write head protocols is also required.

## 1.2 Explanation of Symbols

The following symbols are used in these instructions:



**DANGER**

DANGER indicates an imminently hazardous situation with a high risk of death or serious injury if it is not prevented.



**WARNING**

WARNING indicates a potentially hazardous situation with a medium risk of death or serious injury if it is not prevented.



**CAUTION**

CAUTION indicates a situation that may result in damage to property if it is not prevented.



**NOTE**

NOTE indicates tips, recommendations and important information. The notes will make work easier, contain information on specific action steps and help prevent more work due to incorrect processes.



**CALL TO ACTION**

This symbol identifies action steps that the user has to perform.



**ACTION RESULT**

This symbol identifies relevant results of actions and action sequences.

## 1.3 Additional Documents

The following additional documents are available online at [www.turck.com](http://www.turck.com):

- Configuration manual
- Operating instructions for the read/write heads

### 1.4 Feedback on these Instructions

We are committed to always keeping these instructions as informative and as clear as possible. Should you have any suggestions for a better design or if any information is missing from the instructions, please send your suggestions to [techdoc@turck.com](mailto:techdoc@turck.com).

## 2 Notes on the Product

### 2.1 Product Identification

These instructions apply for the following compact RFID interfaces:

- TBEN-L5-4RFID-8DXP-WIN

### 2.2 Scope of Delivery

Included in the scope of delivery:

- Compact RFID interface
- Closure caps for M12 connectors
- Quick start guide

### 2.3 Legal Requirements

The device falls under the following EU directives:

- 2014/30/EU (electromagnetic compatibility)

### 2.4 Manufacturer and Service

Turck supports you with your projects, from initial analysis to the commissioning of your application. The Turck product database contains software tools for programming, configuration or commissioning, data sheets and CAD files in numerous export formats. You can access the product database at the following address: [www.turck.de/products](http://www.turck.de/products)

Should you have any further questions, please contact the sales and service team in Germany under the following telephone numbers:

Sales: +49 208 4952-380

Technology: +49 208 4952-390

Internet: [www.turck.com/support](http://www.turck.com/support)

Outside Germany, please contact your local Turck representative.

Hans Turck GmbH & Co. KG  
Witzlebenstraße 7  
45472 Mülheim an der Ruhr  
Germany

## 3 For Your Safety

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

### 3.1 Intended Use

The devices are only intended for use in industrial applications.

The block module TBEN-L5-4RFID-8DXP-WIN is an RFID interface for use in the Turck RFID system. The Turck RFID system is used for contactless exchange of data between a data medium and a read/write head in order to identify objects. The interfaces communicate with third-party systems such as ERP systems via TCP/IP. The device functions can be programmed via the Windows Embedded Compact 2013 operating system using .Net, C++ or C#. In addition, middle-ware functions can also be integrated on the device.

The devices must only be used as described in these instructions. Any other usage shall be considered improper and Turck shall not be held liable for any resulting damage.

### 3.2 General Safety Instructions

- The device may only be assembled, installed, operated and maintained by professionally-trained personnel.
- The device may only be used in accordance with applicable national and international regulations, standards and laws.
- The device only meets the EMC requirements for industrial areas and is not suitable for use in residential areas.

## 4 Product Description

The devices are executed in a fully encapsulated plastic housing with protection class IP67/IP69K. Four RFID channels are available for connecting read/write heads. You can also connect sensors and actuators via eight digital I/O channels which you can freely configure as inputs and outputs. The connections for read/write heads and for digital I/Os are executed as M12 connectors. One M12 connector is available for the connection to the Ethernet.

### 4.1 Device Overview

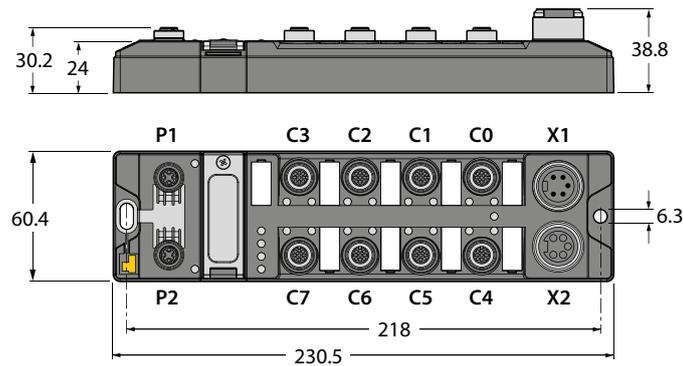


Fig. 1: Dimensions

#### 4.1.1 Display Elements

The devices have configurable multi-color LEDs.

### 4.2 Properties and Characteristics

- Communication via TCP/IP
- Freely programmable compact module based on Windows Embedded Compact 2013
- Programming language: .Net, C++, C#
- API available on request
- 4 channels with M12 connection for RFID
- 8 digital channels, configurable as PNP inputs and/or 2 A outputs
- Integrated Ethernet switch allows line topology
- Transmission rate: 10 Mbps/100 Mbps
- Fully encapsulated module electronics
- Protection classes IP65/IP67/IP69K
- LEDs for status display

## 4.3 Functional Principle

The RFID interfaces connect the RFID system to other systems which communicate via TCP/IP (e.g. ERP systems). The interfaces have one Ethernet interface and multiple RFID interfaces. The RFID system is coupled with a third-party system, such as an ERP system, via the TCP/IP interface. The read/write heads are connected to the interfaces via the RFID interfaces. In addition, the interfaces can process signals from sensors and actuators via eight configurable digital channels.

## 4.4 Functions and Operating Modes

HF and UHF read/write heads can be connected to the RFID channels. It is also possible for HF and UHF read/write heads on one device to operate in parallel. The interface can perform control functions autonomously.

The device functions can be programmed via the Windows Embedded Compact 2013 operating system using .Net, C++ or C#. In addition, middleware functions can also be integrated on the device.

### 4.4.1 Functions in the Turck Service Tool

The device supports the following functions of the Turck Service Tool:

- Changing the IP address
- Reset to factory settings
- Carrying out a voltage reset
- Wink command
- Checking the firmware status

## 5 Mounting

The devices must be attached to a level, pre-drilled and grounded mounting surface.

- ▶ Attach the module to the mounting surface with two M6 screws. The maximum tightening torque for the screws is 1.5 Nm.

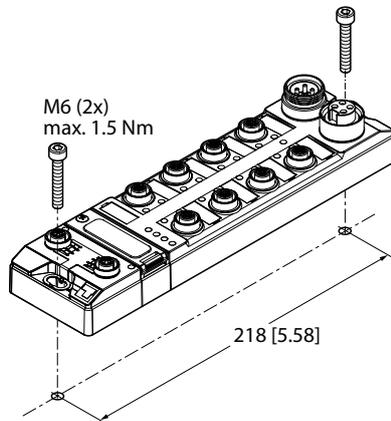


Fig. 2: Attaching the device to the mounting plate

## 5.1 Grounding the Device

### 5.1.1 Grounding and Shielding Concept

The grounding and shielding concept of the TBEN-S modules allows the fieldbus and I/O parts to be grounded separately.

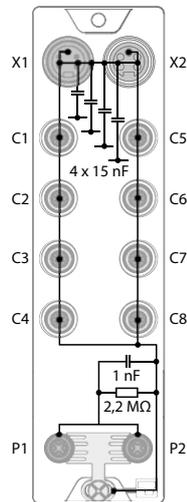


Fig. 3: Replacement wiring diagram, shielding concept

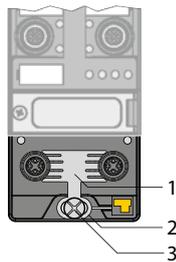


Fig. 4: Grounding components

The grounding clip (1) on the M12 connectors for the fieldbus connection (P1, P2) connects the shield of the fieldbus lines.

The grounding ring (2) is attached below the grounding clip and connects the functional ground of the 7/8" connector (pin 3) for the power supply with the functional ground of the M12 connector (pin 5) for connecting the read/write heads, sensors, and actuators.

The grounding screw (3) connects the device with the system's reference potential.

## 5.1.2 Grounding the Module (FE)

The grounding clip and the metal ring are connected to each other. A mounting screw through the bottom mounting hole in the module connects the shielding of the fieldbus lines to the functional ground of the power supply and the connected devices and to the reference potential of the system.

If a common reference potential is not required, remove the grounding clip to disconnect the fieldbus shield or attach the module with a plastic screw.

### Removing the Grounding Clip

- Use a flat standard screwdriver to lever the grounding clip upwards and remove it.

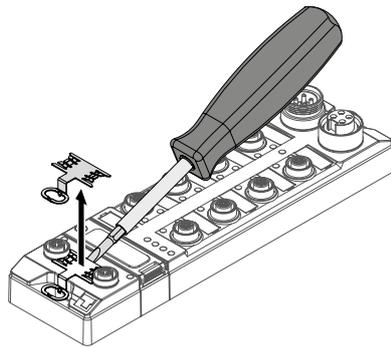


Fig. 5: Removing the grounding clip

### Mounting the Grounding Clip

- Insert the grounding clip between the fieldbus connectors (using a screwdriver if necessary) so that it makes contact with the metal housing of the connector.
- The shield of the fieldbus lines lies flush to the grounding clip.

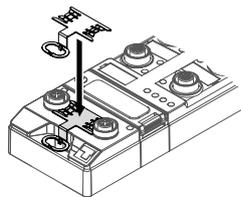


Fig. 6: Mounting the grounding clip

## 6 Connecting

### 6.1 Connecting the Modules to the Ethernet

For the connection to a TCP/IP system, the device has an integrated autocrossing switch with two 4-pin M12 Ethernet connectors. The maximum tightening torque is 0.6 Nm.

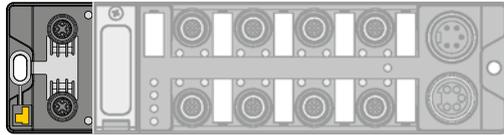


Fig. 7: M12 Ethernet connector for connection to a TCP/IP system

► Connect the device to a TCP/IP system according to the pin assignment shown below.

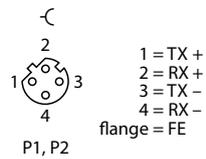


Fig. 8: Pin assignment for Ethernet connections

## 6.2 Connecting the Power Supply

For the connection to the power supply, the device has two 5-pin 7/8" connectors. V1 and V2 are galvanically isolated from one another. The maximum tightening torque is 0.8 Nm.

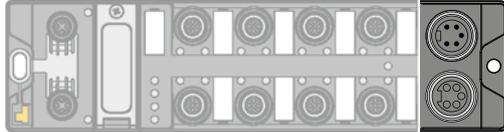


Fig. 9: 7/8" connector for the connection to the power supply

► Connect the device to the power supply according to the pin assignment shown below.

Pin assignment	
<p>X1</p>	<p>1 BK = GND V2                  2 BU = GND V1                  3 GNYE = FE                  4 BN = 24 VDC V1                  5 WH = 24 VDC V2</p>
<p>X2</p>	<p>X1 Power feed                  X2 Continuation of the power to the next participant                  V1 Power supply 1 (incl. supply to the electronics)                  V2 Power supply 2</p>

Fig. 10: Pin assignment for the power supply connections



**NOTE**

The system voltage (V1) and the load voltage (V2) are fed in and monitored separately. If the permitted voltage is not reached, the slots are switched off according to the supply concept for the module type. If V2 is not reached, the PWR LED changes from green to red. If V1 is not reached, the PWR LED goes out.

### 6.3 Connecting the RFID Read/Write Heads

The device has four 5-pin M12 connectors for connecting RFID read/write heads. The maximum tightening torque is 0.8 Nm.

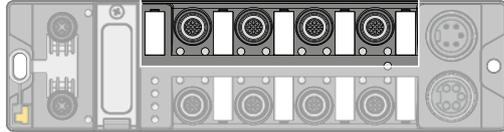


Fig. 11: M12 connector for connecting read/write heads

► Connect the read/write heads to the device according to the pin assignment shown below.

Pin assignment	Pin
<p>1 = V<sub>AUX1</sub> 2 = Data A 3 = GND 4 = Data B 5 = FE/Shield</p>	1 V <sub>AUX1</sub>
	2 TX/RX-
	3 GND (V1)
	4 TX/RX+
	5 FE

Fig. 12: RS485 — pin assignment for read/write head connections

Pin assignment	Pin
<p>1 = BN (+) 2 = BK (Data) 3 = BU (GND) 4 = WH (Data) 5 = shield</p>	1 V <sub>AUX1</sub>
	2 Data
	3 GND (V1)
	4 Data
	5 FE

Fig. 13: Connection lines .../S2500 — pin assignment for read/write head connections

Pin assignment	Pin
<p>1 = BN (+) 2 = WH (Data) 3 = BU (GND) 4 = BK (Data) 5 = shield</p>	1 V <sub>AUX1</sub>
	2 Data
	3 GND (V1)
	4 Data
	5 FE

Fig. 14: Connection lines .../S2501 — pin assignment for read/write head connections

Pin assignment	Pin
<p>1 = RD (+) 2 = BU (Data) 3 = BK (GND) 4 = WH (Data) 5 = shield</p>	1 V <sub>AUX1</sub>
	2 Data
	3 GND (V1)
	4 Data
	5 FE

Fig. 15: Connection lines .../S2503 — pin assignment for read/write head connections

## 6.4 Connecting Digital Sensors and Actuators

The device has four 5-pin M12 connectors for connecting digital sensors and actuators. The maximum tightening torque is 0.8 Nm.

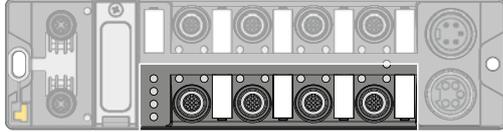


Fig. 16: M12 connector for connecting digital sensors and actuators

► Connect the sensors and actuators to the device according to the pin assignment shown below.

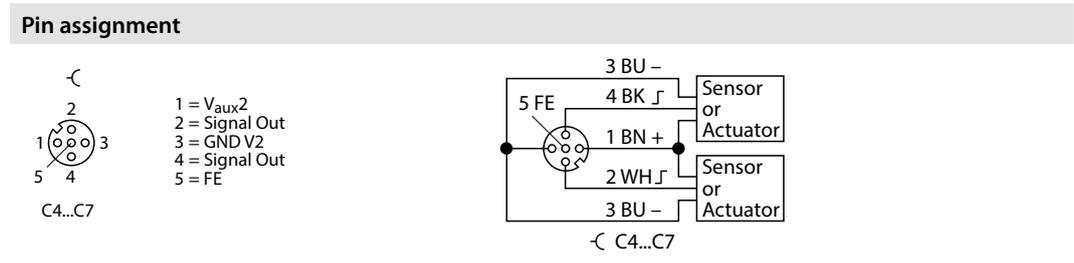


Fig. 17: Pin assignment for connecting digital sensors and actuators

## 7 Commissioning

### 7.1 Setting the IP Address

The IP address can be set via 2 decimal rotary coding switches and DIP switches on the device or via the Turck Service Tool.

#### 7.1.1 Setting the IP Address via Switches on the Device

The IP address can be set via 2 decimal rotary coding switches and the "Mode" DIP switch on the device. The switches are located under a cover, along with the USB ports and the SET button.

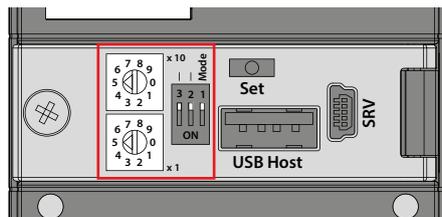


Fig. 18: Switches for setting the IP address

- Open the cover over the switches.
- Set the rotary coding switches to the desired position in accordance with the table below.
- Set the "Mode" DIP switch to the desired position in accordance with the table below.
- Reset the voltage.
- **ATTENTION!** If the cover over the rotary coding switches is open, protection class IP67 or IP69K is not guaranteed. The device may be damaged by the ingress of foreign objects or liquids. Tightly close the cover over the switches.

## Addressing Options

The IP address for the interfaces can be set in various ways. The following addressing options can be selected via the switches on the device. Changes to the settings are activated when the voltage is reset.

Setting option	"MODE" DIP switch	Rotary coding switch	Description
Default address	0	00	IP address: 192.168.1.100 Subnet mask: 255.255.255.0 Gateway: 192.168.1.1
Rotary mode	0	01...99	In rotary mode, the last byte of the IP address can be set manually on the gateway. The other network settings are permanently stored in the gateway memory and cannot be changed in rotary mode. Addresses from 1...99 can be set.
DHCP mode	1	40	In DHCP mode, the complete IP address is automatically assigned by a DHCP server in the network. The subnet mask allocated by the DHCP server and the default gateway address are permanently stored in the gateway memory. DHCP supports 3 types of IP address allocation: <ul style="list-style-type: none"> <li>– Automatic address assignment: The DHCP server assigns a permanent IP address to the client.</li> <li>– Dynamic address assignment: The IP address assigned by the server is always only reserved for a specific period. Once this period has elapsed, or if it is explicitly released by a client, the IP address is reassigned.</li> <li>– Manual address assignment: A network administrator allocates an IP address to the client. In this case, DHCP is only used to transfer the allocated IP address to the client.</li> </ul>
PGM mode	1	50	In PGM mode, the complete IP address is assigned manually via the Turck Service Tool. In PGM mode, the set IP address and the subnet mask are stored in the gateway memory. All network settings (IP address, subnet mask, default gateway) are assumed by the module's internal EEPROM.
F_Reset	1	90	This mode resets all device settings to the default values and deletes all data in the device's internal flash memory. The following values are reset or deleted: <ul style="list-style-type: none"> <li>– IP address and subnet mask</li> <li>– Parameter</li> </ul>
Restore	1	00	IP address: 192.168.1.100 Network mask: 255.255.255.0 Gateway: 192.168.1.1

## 7.1.2 Setting the IP Address via the Turck Service Tool

The device is factory set to IP address 192.168.1.100. The IP address must be set via the Turck Service Tool. The Turck Service Tool is available to download free of charge from [www.turck.com](http://www.turck.com).

- ▶ Launch the Turck Service Tool.
- ▶ Click "Search" or press F5.

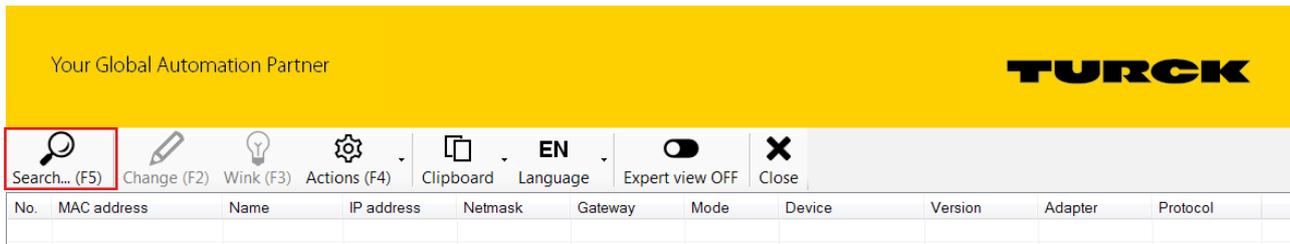


Fig. 19: Turck Service Tool – homescreen

➔ The Turck Service Tool displays the connected devices.

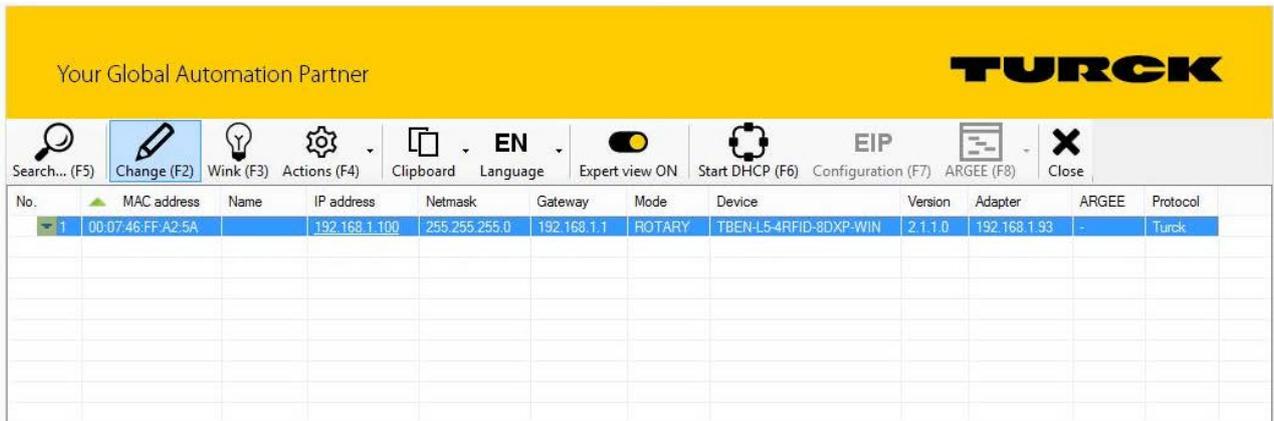


Fig. 20: Turck Service Tool – found devices

- Select the device.
- Click “Change” or press F2.

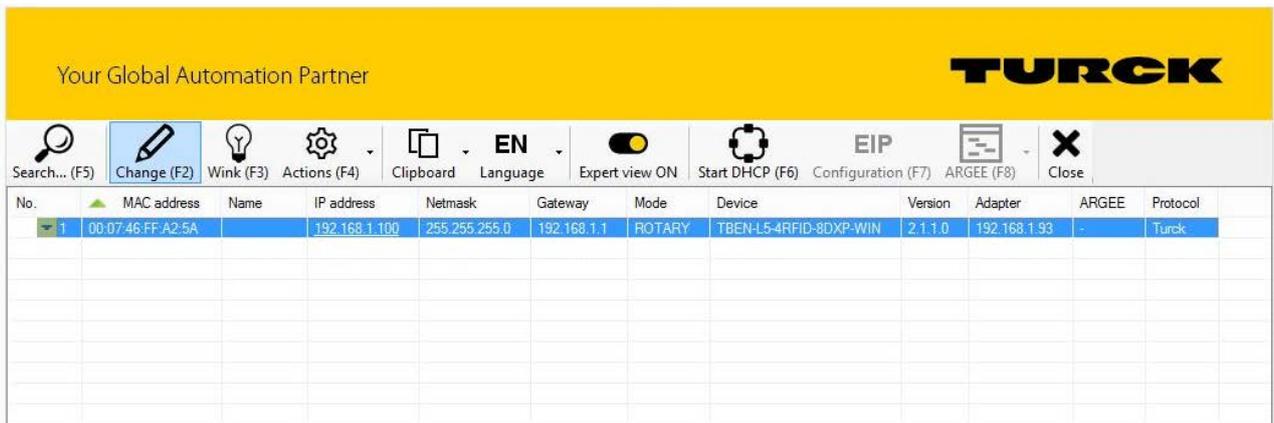


Fig. 21: Turck Service Tool – select the device to be addressed.

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

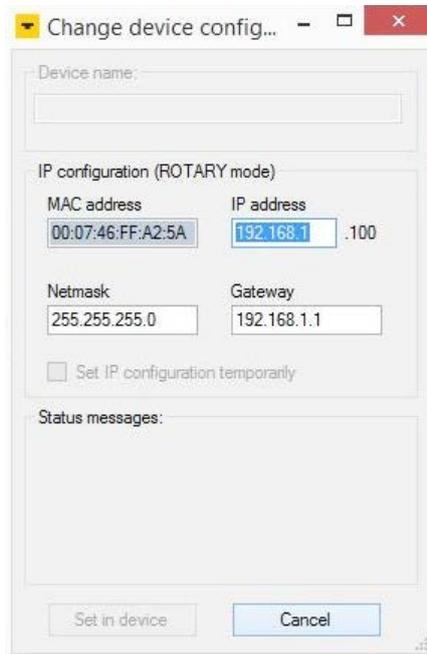


Fig. 22: Turck Service Tool – changing the device configuration

## 7.2 Getting Started

### 7.2.1 Prerequisites

- Windows operating system (min. Windows 7)
- Visual Studio 2012/2013/2015  
(<https://www.visualstudio.com/de-de/products/visual-studio-express-vs.aspx>)
- Application Builder (<https://www.microsoft.com/en-us/download/details.aspx?id=38819>)
- SDK (e.g. Turck\_AM335x\_RFID\_SDKx.msi)

### 7.2.2 Creating Applications

#### Creating Applications in C/C++

- ▶ Make sure that the SDK is installed in the system (e.g. Turck\_AM335x\_RFID\_SDKx.msi).
- ▶ Start Visual Studio.
- ▶ Select "New Project...".
- ▶ In the left-hand window area, under "Templates → Visual C++ -> Windows Embedded Compact", select "AM335x\_Turck\_RFID\_SDKx".
- ▶ In the central window area, select "Win32 Console Application".
- ▶ Enter a name in the lower area.
- ▶ Click "OK". Visual Studio starts in the Application Builder view with an automatically generated, simple console application.
- ▶ Add "#include "windows.h"".
- ▶ Add the following lines to the "wmain" function:

```
printf("Hello World\n");
Sleep(5000);
```

- ▶ Press F7 to compile the application.

#### Creating Applications in C#

- ▶ Make sure that the SDK is installed in the system (e.g. Turck\_AM335x\_RFID\_SDKx.msi).
- ▶ Start Visual Studio.
- ▶ Select "New Project...".
- ▶ In the left-hand window area, under "Templates → Other Languages -> Visual C++ -> Windows Embedded Compact", select "AM335x\_Turck\_RFID\_SDKx".
- ▶ In the central window area, select "Console Application".
- ▶ Enter a name in the lower area.
- ▶ Click "OK". Visual Studio starts in the Application Builder view with an automatically generated, simple console application.
- ▶ At the start of the "main.cs" file, add "using System.Threading;" and "using System.Diagnostics;"
- ▶ Add the following lines in "Main Method":

```
Debug.WriteLine("Hello World");
Thread.Sleep(5000);
```

- ▶ Press F7 to compile the application.

## 7.3 Drivers

### 7.3.1 Ethernet

The Ethernet driver supports the Ethernet controller AM335x CPSW3G in switching mode.

The external ports of the Ethernet controller are mapped to the X\_P1 and X\_P2 plugs. The internal port of the Ethernet switch is available to the system as Ethernet device CPSW3G1.

The FTP server and the Telnet server are activated as default (with no authentication).

For details about WinSock, see

<https://msdn.microsoft.com/EN-US/library/ee494651%28v=VS.80,d=HV.2%29.aspx>.

### 7.3.2 NAND Flash

The NAND flash driver supports the AM335x GPMC controller and the inserted NAND flash card. The NAND flash driver is loaded automatically at the start and maps the FAT partition to the directory "/Mounted\_Volume" transparently as standard.

The device directory is stored in the NAND flash file system permanently as standard.

### 7.3.3 USB Host

The USB host supports the Ethernet controller AM335x USB1 in host mode. The host is available on plug X25.

If the device is inserted and supported by the system, USB device drivers are loaded automatically. Human interface devices (MMIs) and USB mass storage are supported as standard.

### 7.3.4 USB OTG

In device mode, the USB OTG driver supports the Ethernet controller AM335x USB0. The USB OTG is available on plug X18.

Per default, the USB OTG device driver is configured to USB serial mode.

### 7.3.5 UART

The UART driver supports the UART1, UART2, UART3, and UART4 devices of the AM335x. The driver only supports RX/TX. Signals for a flow control are not available. The UARTs are achieved as COM1: - COM4: devices that use Win32 Serial Port API. For details, see <https://msdn.microsoft.com/EN-US/library/ee488234%28v=VS.80,d=HV.2%29.aspx>.

In order to correspond to the TBOX channels, the processor UARTs are mapped to the COM ports as follows:

- UART1 → COM3:
- UART2 → COM4:
- UART3 → COM1:
- UART4 → COM2:

By changing the value of the key "HKEY\_LOCAL\_MACHINE\Drivers\BuiltIn\UARTX\RxFifoTriggerLevel" (possible values are 1-63), you can influence the receive FIFO of the UART.

### 7.3.6 GPIO

The GPIO driver supports the AM335x GPIOs as well as the PCA9506 GPIO expander and the XMC LED GPIOs.

The GPIO driver is available as a GIO1: device. It is available via the driver functions of the stream interface. For details, see <https://msdn.microsoft.com/EN-US/library/ee488234%28v=VS.80,d=HV.2%29.aspx>.

The GPIOs must be identified via their GPIO ID. These are defined in the gpio\_defines.h file which can be found in the SDK. The following GPIOs are supported by the BSP:

AM335x GPIO	GPIO ID	GPIO define
GPIO0_7	7	GPIO_7
GPIO0_12	12	GPIO_12
GPIO0_13	13	GPIO_13
GPIO0_19	19	GPIO_19
GPIO0_23	23	GPIO_23
GPIO1_15	47	GPIO_47
GPIO1_31	63	GPIO_63
GPIO2_22	86	GPIO_86
GPIO2_23	87	GPIO_87
GPIO2_24	88	GPIO_88
GPIO2_25	89	GPIO_89
GPIO3_14	110	GPIO_110
GPIO3_15	111	GPIO_111
GPIO3_16	112	GPIO_112
GPIO3_17	113	GPIO_113
GPIO3_18	114	GPIO_114
GPIO3_20	116	GPIO_116
GPIO3_21	117	GPIO_117

XMC GPIO	GPIO ID	GPIO define
P0.13	208	XGPIO_0
P0.12	209	XGPIO_1
P1.1	210	XGPIO_2
P1.0	211	XGPIO_3
P0.5	212	XGPIO_4
P0.4	213	XGPIO_5
P0.11	214	XGPIO_6
P0.10	215	XGPIO_7
P1.6	216	XGPIO_8

PCA GPIO	GPIO ID	GPIO define
PCA0_0[0]	128	EGPIO_0
...	...	...
PCA0_4[7]	167	EGPIO_39
PCA1_0[0]	168	EGPIO_40
...	...	...
PCA1_4[7]	207	EGPIO_79

The driver supports the following IOControl codes (defined in gpio\_ioctls.h):

## IOCTL\_GPIO\_SETBIT

Sets the corresponding GPIO to Level 1

Parameter	
lpInBuffer	Pointer to DWORD which contains the GPIO ID to be set

## IOCTL\_GPIO\_CLRBIT

Sets the corresponding GPIO to Level 0

Parameter	
lpInBuffer	Pointer to DWORD which contains the GPIO ID to be set

## IOCTL\_GPIO\_GETBIT

Reads from the level of the corresponding GPIO

Parameter	
pInBuffer	Pointer to DWORD which contains the GPIO ID to be read
lpOutBuffer	Pointer to DWORD which receives the current level

## IOCTL\_GPIO\_SETMODE

Configures the mode of the corresponding GPIO

Parameter	
pInBuffer	Pointer to an array of two DWORDs which contains the GPIO ID (array element 0) and the mode to be set (array element 1)

The following modes are supported (defined in gpio\_defines.h):

- GPIO\_DIR\_OUTPUT : configures the GPIO as an output
- GPIO\_DIR\_INPUT : configures the GPIO as an input
- GPIO\_INT\_LOW\_HIGH : activation of the interrupt with a rising edge
- GPIO\_INT\_HIGH\_LOW : activation of the interrupt with a falling edge
- GPIO\_INT\_LOW : activation of the low-level interrupt
- GPIO\_INT\_HIGH : activation of the high-level interrupt
- GPIO\_DEBOUNCE\_ENABLE : activation of the debounce

The modes GPIO\_INT\_LOW\_HIGH, GPIO\_INT\_HIGH\_LOW, GPIO\_INT\_LOW, GPIO\_INT\_HIGH, and GPIO\_DEBOUNCE\_ENABLE are not supported by the PCA (EGPIO\_0 – EGPIO\_79) and XMC GPIOs (XGPIO\_0 – XGPIO\_8).

## IOCTL\_GPIO\_GETMODE

Returns the current mode of the corresponding GPIO

Parameter	
lpInBuffer	Pointer to DWORD which contains the GPIO ID
lpOutBuffer	Pointer to DWORD which receives the current mode

## IOCTL\_GPIO\_CONFIGURE\_PERIODIC\_UPDATE

Sets the corresponding GPIO in toggle mode with 1 Hz or 2 Hz.

Parameter	
lpInBuffer	Pointer to a structure IOCTL_GPIO_PERIODIC_INFO (defined in gpio_ioctls.h) which contains the GPIO ID and the periodic mode to be set:
	<pre>typedef struct {     UINT uGpioID;     UINT dwPeriMode; } IOCTL_GPIO_PERIODIC_INFO;</pre>

The following modes are supported (defined in gpio\_defines.h):

- GPIO\_PERIODIC\_DISABLE : deactivates the periodic switching (toggling) of the GPIO
- GPIO\_PERIODIC\_1HZ : activation of the periodic switching of the GPIO with 1 Hz
- GPIO\_PERIODIC\_2HZ : activation of the periodic switching of the GPIO with 2Hz

This IOCONTROL function is not supported by XGPIO\_8 GPIO.

## IOCTL\_GPIO\_CONFIGURE\_RXTX\_UPDATE

Configures the corresponding GPIO for RX/TX signals

Parameter	
lpInBuffer	Pointer to DWORD which contains the GPIO ID to be configured

The RX/TX signaling can only be deactivated by activating another mode via IOCTL\_GPIO\_CONFIGURE\_PERIODIC\_UPDATE, IOCTL\_GPIO\_SETBIT, or IOCTL\_GPIO\_CLRBIT.

This IOCONTROL is supported by GPIOs XGPIO\_0 to XGPIO\_8.

## IOCTL\_GPIO\_CONFIGURE\_AUTO\_RECOVERY

Configures the automatic recovery of the GPIO

Parameter	
lpInBuffer	<p>Pointer to a structure IOCTL_OC_INFO (defined in gpio_ioctls.h) which contains the GPIO ID for activating the automatic recovery, the excess current detection of the GPIO, and the mode (automatic or manual recovery):</p> <pre>typedef struct {     UINT uGpioID;     UINT uOCGpioID;     BYTE bOCLevel;     BYTE bMode; }</pre>

For the parameter bOCLevel (defined in gpio\_defines.h), the following conditions of the trigger level are supported:

- GPIO\_AUTOREC\_OC\_LOWLEV: the excess current condition of uGpioID is triggered by a low-level interrupt at uOCGpioID
- GPIO\_AUTOREC\_OC\_HIGHLEV: the excess current condition of uGpioID is triggered by a high-level interrupt at uOCGpioID
- GPIO\_AUTOREC\_OC\_SAMELEV: the excess current condition of uGpioID is triggered when uOCGpioID is on the same level
- GPIO\_AUTOREC\_OC\_DIFFLEV: the excess current condition of uGpioID is triggered when uOCGpioID is on the additional level

For the parameter bMode, the following modes (defined in gpio\_defines.h) are supported:

- GPIO\_AUTOREC\_OC\_DISABLED: excess current detection deactivated
- GPIO\_AUTOREC\_OC\_SHUTDOWN: the excess current condition is handled with a shutdown (manual recovery)
- GPIO\_AUTOREC\_OC\_AUTOREC: the excess current condition is handled with a shutdown and after a safe period, with an automatic recovery

### IOCTL\_GPIO\_GETIRQ

Returns the IRQ of the corresponding GPIO

Parameter	
lpInBuffer	Pointer to DWORD which contains the GPIO ID
lpOutBuffer	Pointer to DWORD which receives the IRQ number

This IOCTL function is not supported by the PCA (EGPIO\_0 – EGPIO\_79) and XMC GPIOs (XGPIO\_0 – XGPIO\_8).

### IOCTL\_GPIO\_SET\_DEBOUNCE\_TIME

Sets the debounce time of the GPIO (bank)

Parameter	
lpInBuffer	Pointer to a structure IOCTL_GPIO_SET_DEBOUNCE_TIME_IN (defined in gpio_ioctls.h) which contains the GPIO ID and the debounce time to be set:
	<pre>typedef struct {     UINT gpioId;     UINT debounceTime; } IOCTL_GPIO_SET_DEBOUNCE_TIME_IN;</pre>

The debounce time is calculated as follows:

Debounce time = (DEBOUNCETIME + 1) × 31 μs. The debounce time is valid globally for all GPIOs of the same bank.

This IOCTL function is not supported by the PCA (EGPIO\_0 – EGPIO\_79) and XMC GPIOs (XGPIO\_0 – XGPIO\_8).

### IOCTL\_GPIO\_GET\_DEBOUNCE\_TIME

Returns the debounce time of the GPIO (bank)

Parameter	
lpInBuffer	Pointer to DWORD which contains the GPIO ID
lpOutBuffer	Pointer to DWORD which receives the debounce time

This IOCTL function is not supported by the PCA (EGPIO\_0 – EGPIO\_79) and XMC GPIOs (XGPIO\_0 – XGPIO\_8).

## IOCTL\_GPIO\_INIT\_INTERRUPT

Initializes the interrupt for the GPIO

Parameter	
lpInBuffer	Pointer to a structure IOCTL_GPIO_INIT_INTERRUPT_INFO (defined in gpio_ioctls.h):
	<pre>typedef struct {     UINT    uGpioID;     DWORD   dwSysIntrID;     HANDLE  hEvent; } IOCTL_GPIO_INIT_INTERRUPT_INFO;</pre>

uGpioID must be set to the GPIO ID and hEvent to an event handle. The SysIntr used is returned in the element dwSysIntrID.

This IOCTL function is not supported by the PCA (EGPIO\_0 – EGPIO\_79) and XMC GPIOs (XGPIO\_0 – XGPIO\_8).

## IOCTL\_GPIO\_ACK\_INTERRUPT

Acknowledges a GPIO interrupt

Parameter	
lpInBuffer	Pointer to a structure IOCTL_GPIO_INTERRUPT_INFO (defined in gpio_ioctls.h):
	<pre>typedef struct {     UINT    uGpioID;     DWORD   dwSysIntrID; } IOCTL_GPIO_INTERRUPT_INFO;</pre>

This IOCTL function is not supported by the PCA (EGPIO\_0 – EGPIO\_79) and XMC GPIOs (XGPIO\_0 – XGPIO\_8).

## IOCTL\_GPIO\_DISABLE\_INTERRUPT

Deactivates the interrupt of a GPIO

Parameter	
lpInBuffer	Pointer to a structure IOCTL_GPIO_INTERRUPT_INFO

This IOCTL function is not supported by the PCA (EGPIO\_0 – EGPIO\_79) and XMC GPIOs (XGPIO\_0 – XGPIO\_8).

7.3.7 SPI

The SPI driver supports the AM335x MCSPI0 interface. The SPI0 interface is available as an SPI1: device.

The driver supports the following functions (defined in sdk\_spi.h):

HANDLE SPIOpen(LPCTSTR pSpiName)

Opens the driver for later use

Parameter	
pSpiName	String with the device name ("SPI1:")
Return value	
Handle on driver	

VOID SPIClose(HANDLE hContext)

Ends the driver after use

Parameter	
hContext	Handle returned via SPIOpen()

BOOL SPILockController(HANDLE hContext, DWORD dwTimeout)

Locks the access to the driver for the current thread

Parameter	
hContext	Handle returned by SPIOpen()
dwTimeout	Timeout for activation of the lock
Return value	
TRUE	Successful
FALSE	Not successful

BOOL SPIUnlockController(HANDLE hContext)

Unlocks the access to the driver.

Locks the access to the driver for the current thread

Parameter	
hContext	Handle returned by SPIOpen()
Return value	
TRUE	Successful
FALSE	Not successful

BOOL SPIConfigure(HANDLE hContext, DWORD address, DWORD config)

Configures the SPI device for further actions

Parameter	
hContext	Handle returned by SPIOpen()
address	Chipselect number (only CS0 is supported)
config	DWORD which contains the desired configuration. The configuration must take place in accordance with the description of the register MCSPI_CHOCONF in the manual of the AM335x (Technical Reference Manual).

Return value	
TRUE	Successful
FALSE	Not successful

BOOL SPIEnableChannel(HANDLE hContext)

Activates the channel configured by the address parameter SPIConfigure() and therefore also the corresponding chipselect

Parameter	
hContext	Handle returned by SPIOpen()
Return value	
TRUE	Successful
FALSE	Not successful

BOOL SPIDisableChannel(HANDLE hContext)

Deactivates the channel previous activated by SPIEnableChannel()

Parameter	
hContext	Handle returned by SPIOpen()
Return value	
TRUE	Successful
FALSE	Not successful

BOOL SPISetSlaveMode(HANDLE hContext)

Configures the SPI controller for slave mode

Parameter	
hContext	Handle returned by SPIOpen()
Return value	
TRUE	Successful
FALSE	Not successful

DWORD SPIRead(HANDLE hContext, DWORD size, VOID \*pBuffer)

Reads from the SPI bus

Parameter	
hContext	Handle returned by SPIOpen()
size	Number of bytes to be read
pBuffer	Pointer to receivebuffer
Return value	
Number of bytes actually read	

DWORD SPIWrite(HANDLE hContext, DWORD size, VOID \*pBuffer)

Writes to the SPI bus

Parameter	
hContext	Handle returned by SPIOpen()
size	Number of bytes to be written
pBuffer	Pointer to sendbuffer
Return value	
Number of bytes actually written	

DWORD SPIWriteRead(HANDLE hContext, DWORD size, VOID \*pOutBuffer, VOID \*pInBuffer)

Reads/writes simultaneously from/to the SPI bus

Parameter	
hContext	Handle returned by SPIOpen()
size	Number of bytes to be read/written
pOutBuffer	Pointer to sendbuffer
pInBuffer	Pointer to receivebuffer
Return value	
Number of bytes actually read/written	

DWORD SPIAsyncWriteRead(HANDLE hContext, DWORD size, VOID \*pOutBuffer, VOID \*pInBuffer)

Reads/writes simultaneously from/to the SPI bus via DMA

Parameter	
hContext	Handle returned by SPIOpen()
size	Number of bytes to be read/written
pOutBuffer	Pointer to sendbuffer
pInBuffer	Not used, set to NULL
Return value	
Value of the size parameter	

DWORD SPIWaitForAsyncWriteReadComple(HANDLE hContext, DWORD size, VOID \*pOutBuffer)

Waiting for completion of the DMA transfer

Parameter	
hContext	Handle returned by SPIOpen()
size	Number of bytes to be written
pOutBuffer	Pointer to receivebuffer
Return value	
Value of the size parameter	

## 7.3.8 I2C

The I2C proxy driver (available in user mode) supports the AM335x-I2C0 interface. The I2C0 interface is available as an I2C1: device.

The I2C proxy driver is available via the file API (CreateFile(), ReadFile(), WriteFile(), SetFilePointer()).

To select the base subaddress used in the subsequent ReadFile()- WriteFile() calls, SetFilePointer() is used. For the selection of the address and the baud rate of the I2C device, the following IOControl codes are available (defined in i2cproxy.h):

### IOCTL\_I2C\_SET\_SLAVE\_ADDRESS

Sets the slave address of the I2C device to be addressed

Parameter	
lpInBuffer	Pointer to DWORD which contains the slave address

### IOCTL\_I2C\_SET\_SUBADDRESS\_MODE

Sets the subaddress mode

Parameter	
lpInBuffer	Pointer to DWORD which contains the desired subaddress mode. The following modes are available (defined in sdk_i2c.h): - I2C_SUBADDRESS_MODE_0: no device subaddresses - I2C_SUBADDRESS_MODE_8: 1-byte subaddresses - I2C_SUBADDRESS_MODE_16: 2-byte subaddresses - I2C_SUBADDRESS_MODE_24: 3-byte subaddresses - I2C_SUBADDRESS_MODE_32: 4-byte subaddresses

## IOCTL\_I2C\_SET\_BAUD\_INDEX

Sets the baud rate of the I2C

Parameter	
lpInBuffer	– Pointer to DWORD which contains the desired baud rate. The following baud rates are available (defined in sdk_i2c.h): – SLOWSPEED_MODE: 100 KHz – FULLSPEED_MODE: 400 KHz – HIGHSPEED_MODE_1P16: 1.6 MHz – HIGHSPEED_MODE_2P4: 2.4 MHz – HIGHSPEED_MODE_3P2: 3.2 MHz

### 7.3.9 RTC

The OAL supports the onboard RTC. The RTC is used automatically by the system (OAL). Therefore, the time is read out and set via, for example, the functions GetSystemTime() and SetSystemTime() or via console functions for the date and time. The RTC is synchronized at system start (read) and when the value is changed (write).

### 7.3.10 Using the Application

To ensure that the device monitors the Application Builder deploy/debug requirements, components on the device have to be activated manually once the device has been started. This can be done via Telnet, for example:

- Connect the device via Telnet (default address IP 192.168.1.1).
- Run "start conmanclient3 & cmancept3" in the command line of the device.

Once the device has been prepared for the connection with Application Builder, the device can be used and run on the device by pressing F5 (or via "Debug → Start Debugging").

Application Builder should switch to the debug view, transfer the application to the device and then run it. When the example application referred to above is run, the string "Hello World" should be visible in the output window (debug) of Application Builder.

## 7.3.11 Debugging the Application

To insert a breakpoint in a specific line of the source code, click the vertical gray bar in front of the line. The breakpoint is identified with a red dot.

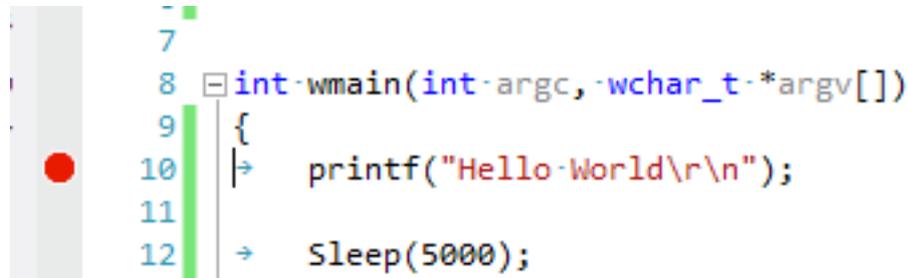


Fig. 23: Debugging the application

The next time the application is used and when it is run on the target device, it stops at the preselected breakpoint(s). This is indicated by a yellow arrow on top of the breakpoint.

A remote debugging via Visual Studio is very similar to the local debugging of an application, including the individual steps, callstack display (call list), memory display, etc.

## 7.3.12 Using a Network Socket in C#

You can implement a network communication in C# via the socket class. The following example code can be used to create and open a socket for a server at 192.168.1.1 on Port 80. For more information about the communication via the socket, see [https://msdn.microsoft.com/en-us/library/system.net.sockets.socket\\_members\(v=vs.90\)](https://msdn.microsoft.com/en-us/library/system.net.sockets.socket_members(v=vs.90)).

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Diagnostics;

using System.Net;
using System.Net.Sockets;

namespace TurckWinSock
{
    class Program
    {
        static void Main(string[] args)
        {
            Debug.WriteLine("Hello World");

            //Create an endpoint with the server IP and Port 80
            IPAddress ip = new IPAddress(new byte[] {192, 168, 4,
80});
            IPEndPoint IpEnd = new IPEndPoint(ip, 80);

            //create a socket object
            Socket TestSocket = new Socket(IpEnd.AddressFamily,
SocketType.Stream, ProtocolType.Tcp);

```

```

try
{
    //Connect the socket to the server
    TestSocket.Connect (IpEnd);

    //Check the connection
    if (TestSocket.Connected)
    {
        Debug.WriteLine("socket connected");
    }
    else
    {
        Debug.WriteLine("socket connection failed");
    }

    //Do something with the socket
    Thread.Sleep(5000);

    //Close the socket
    TestSocket.Shutdown(SocketShutdown.Both);
    TestSocket.Close();
}
catch (Exception e)
{
    Debug.WriteLine("exception while connecting
socket");
}
}
}
}

```

## 7.3.13 Using the TBOX API Library

An API library (TBOX API) is available for extracting functions of the DXPs and the COM ports. An overview and description of the functions of the API library are available in the header file TBOX\_API.h.

### Procedure with a C/C++ Application

To use the API library in a C/C++ application, integrate the header file TBOX\_API.h and create a static link between TBOX\_API\_LIB.lib and the application.

Copy TBOX\_API.h and TBOX\_API\_LIB.lib to your application directory and insert the library to the linker as an additional dependency:

- ▶ Select "PROJECT → Project Properties" (Alt + F7).
- ▶ In the left-hand window area, select "Configuration Properties → Linker → Input".
- ▶ In the right-hand window area, click the "Additional Dependencies" dropdown list and select "<Edit...>".

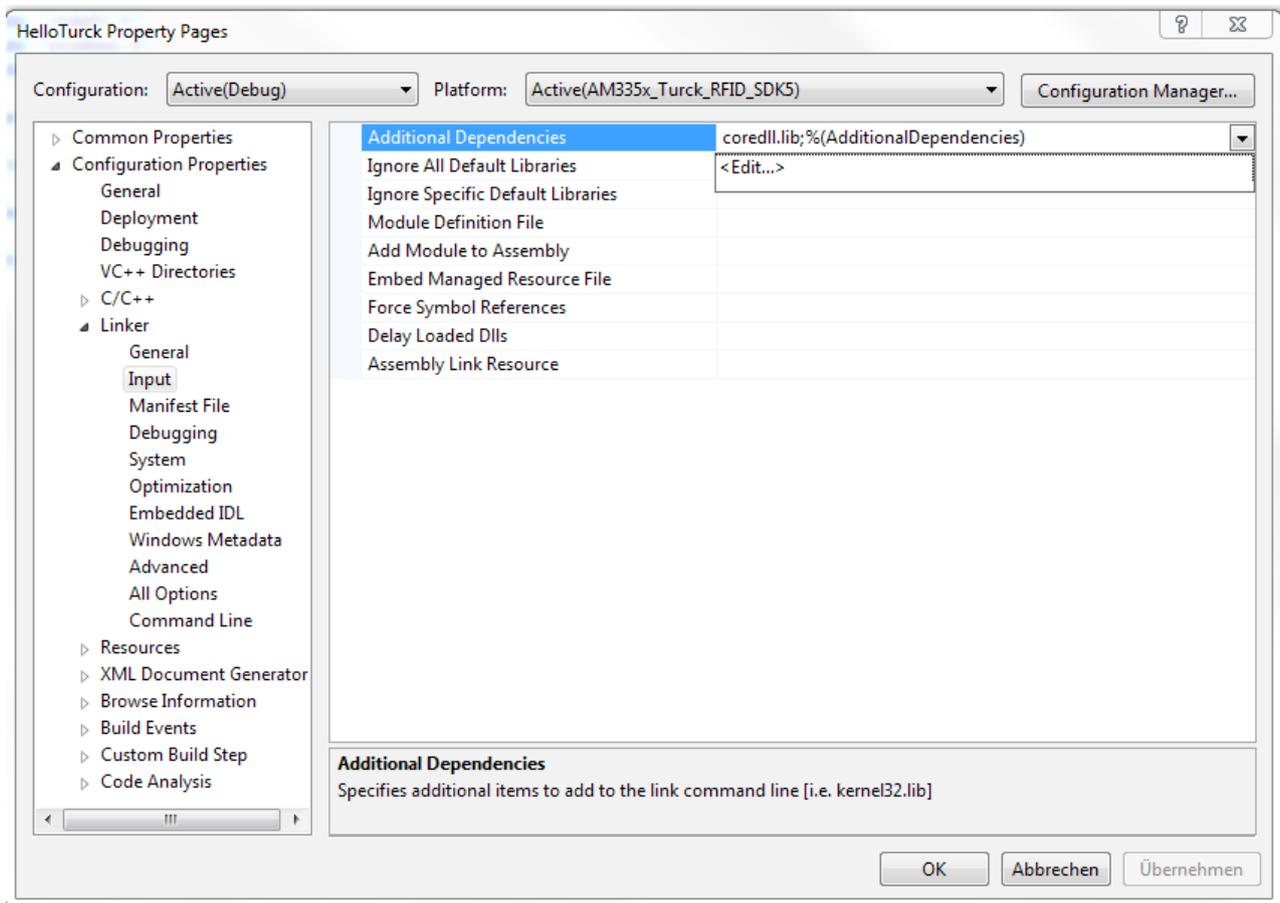


Fig. 24: Property pages

- Enter "TBOX\_API\_LIB.lib" in the editing field and confirm.

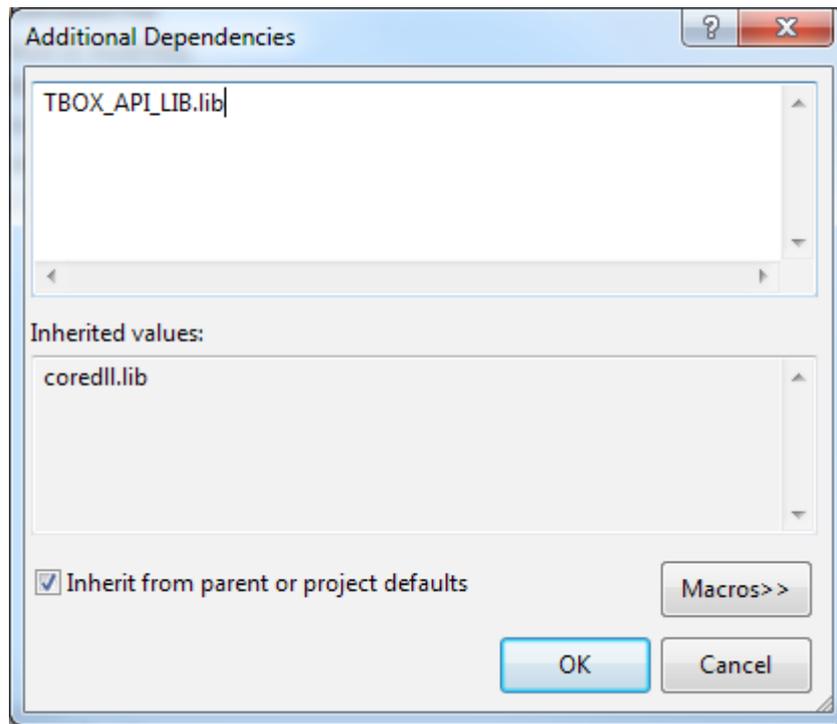


Fig. 25: Additional dependencies

- At the start of the main source file, integrate the header file TBOX\_API.h via "#include "TBOX\_API.h";".
- ➔ The application can use the TBOX API. Before using further API functions, "TBOX\_Init();" must first be called. This resets all settings to the standard values and puts the hardware in a defined state.

**Example 1: Have the green LED app flash:**

```
TBOX_SYS_LED_GreenSet(LED_APP, LED_2HZ);
```

The setting LED\_TxD\_RxD\_SIGNALING is supported only for LEDs assigned to a COM port.

**Example 2: Import the value of the rotary switch:**

```
DWORD dwVal = TBOX_SYS_SWITCH_Get(0);
printf("Rotary Switches Value: %d\n", dwVal);
```

**Example 3: Configure the DXP as an input and read its current value:**

```
TBOX_DXP_Init(8, FALSE, FALSE);
DWORD dwVal = TBOX_DXP_Get(8);
printf("DXP 8 Level: %d\n", dwVal2);
```

DXP0 to DXP7 are not used on this platform because they are occupied by the COM ports.

**Example 4: Read all DXP inputs immediately:**

```
dwVal = TBOX_DXP_Get(-1);
for(unsigned int i=8; i < 16; i++)
    printf("DXP %d Level: %d\n", i, (dwVal & (1 << i)) >> i);
```

The values of all DXPs are read out if "-1" is used as the DxpNo parameter. The return value is available via a bit field whose bits correspond to the number of the DXP level (e.g. bit0 for DXP 0, bit1 for DXP1, etc.).

**Example 5: Configure a COM with the settings RS485 mode, Swap Lines, No Bias and No Termination. Also activate RX and TX signals for the corresponding LEDs and switch on the power supply VAUX for the port:**

```
TBOX_COM_HardwareInit(TBOX_COM1,
                      0,
                      TBOX_COM_RS485_MODE,
                      TBOX_COM_SWAP_AB_MODE,
                      TBOX_COM_BIAS_OFF_MODE,
                      TBOX_COM_TERM_OFF_MODE);

TBOX_SYS_LED_GreenSet(LED_COM1_TX, LED_TxD_RxD_SIGNALING);
TBOX_SYS_LED_GreenSet(LED_COM1_RX, LED_TxD_RxD_SIGNALING);

TBOX_SYS_VAUX_Set(TBOX_COM1, TBOX_COM_POWER_24V_MODE);
```

The baud setting is ignored after the COM port has been set (standard value is 115200 baud). Only the RS485 mode is supported.

### 7.3.14 Procedure with a C# Application

To use the API in C# applications, the API is also available as a DLL (TBOX\_API\_DLL.dll). The DLL is not part of the OS standard image and must therefore be copied to the device.

Before a function from the DLL can be used, the DLL must be imported to your C# application.

Example:

```
[DllImport("TBOX_API_DLL.dll")]
public extern static int TBOX_Init();

[DllImport("TBOX_API_DLL.dll")]
public extern static void TBOX_SYS_LED_GreenSet(int LedNo, int State);
```

After the import, the DLL can be called up within the C# application. Example:

```
TBOX_Init();

TBOX_SYS_LED_GreenSet(4, 0xCCCC);
```

## 7.4 Specific Settings / Implementations

### 7.4.1 Autostart Application

The default registry contains a reference to a specific application in the flash storage to auto-start. Therefore it is possible to let your own application autostart on boot by placing it into “\Mounted\_Volume\” and naming it “autostart.exe”.

### 7.4.2 Image Version Readout

To read out the current WEC image version, include “bsp\_ioctls.h” from the SDK and use the following IOControl Code with KernelloControl: IOCTL\_HAL\_GET\_BSP\_VERSION  
 Parameter lpOutBuffer: pointer to a IOCTL\_HAL\_GET\_BSP\_VERSION\_OUT struct.

IOCTL\_HAL\_GET\_BSP\_VERSION\_OUT is defined as follows:

```
typedef struct {
    DWORD      dwVersionMajor;
    DWORD      dwVersionMinor;
    DWORD      dwVersionQFES;
    DWORD      dwVersionIncremental;
} IOCTL_HAL_GET_BSP_VERSION_OUT;
```

### 7.4.3 Addressing Mode Readout

In order to read out the addressing mode determined at boottime according to the pushbutton, dip- and rotary-switches, the following IOControl Code (defined in “bsp\_ioctls.h”) can be used with KernelloControl: IOCTL\_HAL\_GET\_ADDRMODE  
 Parameter lpOutBuffer: pointer to DWORD receiving the addressing mode.  
 The Ethernet IP Settings are automatically modified at WEC startup according to the detected mode.  
 The addressing modes are defined on p. 21.

## 8 Operation

### 8.1 LED Indicators

The devices have freely programmable multi-color LEDs for displaying information about:

- Power supply
- Collective and bus faults
- Status
- Diagnostics

LED PWR	Meaning
Off	No power or undervoltage at V1
Lights up in green	Power at V1 ok
Lights up in red	No power or undervoltage at V2

LED BUS	Meaning
Flashing red	Wink command active

COM LEDs (RFID channels)		
TXD LED	RXD LED	Meaning
Flashes green	Off	Data being sent
Off	Flashes green	Data being received
Flashing red	Flashing red	Short circuit in the power supply
	Illuminated in red	Memory overflow
Illuminated red/green	Illuminated red/green	Incorrect configuration

DXP LEDs (digital channels, LEDs 8...15)		
Green LED	Red LED	Meaning
Off	Off	No I/O signal present
Illuminated	Off	I/O signal present
Off	Illuminated	Overload at output
Flashing	Flashing	Overload of the auxiliary supply

## 9 Troubleshooting

If the device does not work as expected, first check whether ambient interference is present. If there is no ambient interference, check the device connections for faults. If there are no faults, there is a device malfunction. In this case, decommission the device and replace it with a new device of the same type.

## 10 Maintenance

Perform regular checks to ensure that the connections and cables are always in good condition. The devices are maintenance-free. If necessary, clean in a dry state.

### 10.1 Carrying out a Firmware Update

The firmware for the device can be updated using the PuTTY and WinSCP tools.



**CAUTION**

Interruption of the power supply during the firmware update

**Device damage due to faulty firmware update**

- Do not interrupt the power supply to the device during the firmware update.
- Do not reset the power supply during the firmware update.

#### Example: Updating Firmware

The example uses the following settings:

- The device is connected to the power supply via the X1 connection.
- The device is connected to the Ethernet via the ETH2 connection.
- The rotary coding switches on the device are in position 00 (default). The device's IP address depends on the firmware status:

Firmware status	IP address
Version 2.1.0.0 or earlier	192.168.1.1
Version 2.1.1.0 or later	192.168.1.100

- The device is connected to the host PC.
- The PuTTY and WinSCP software tools have been installed.

- Open PuTTY.
- Enter the following settings in PuTTY:
  - Host name: IP address of the device
  - Port: 23
- Optional: Assign a name for the current session (here: TBEN-Lx-WINCE). The session can be loaded via “Load” for later repetitions.
- With saved sessions: Select TBEN-Lx-WINCE and confirm with “Load”.
- Click “Open”.



**NOTE**

If a connection cannot be established, check the IP address of the host PC.

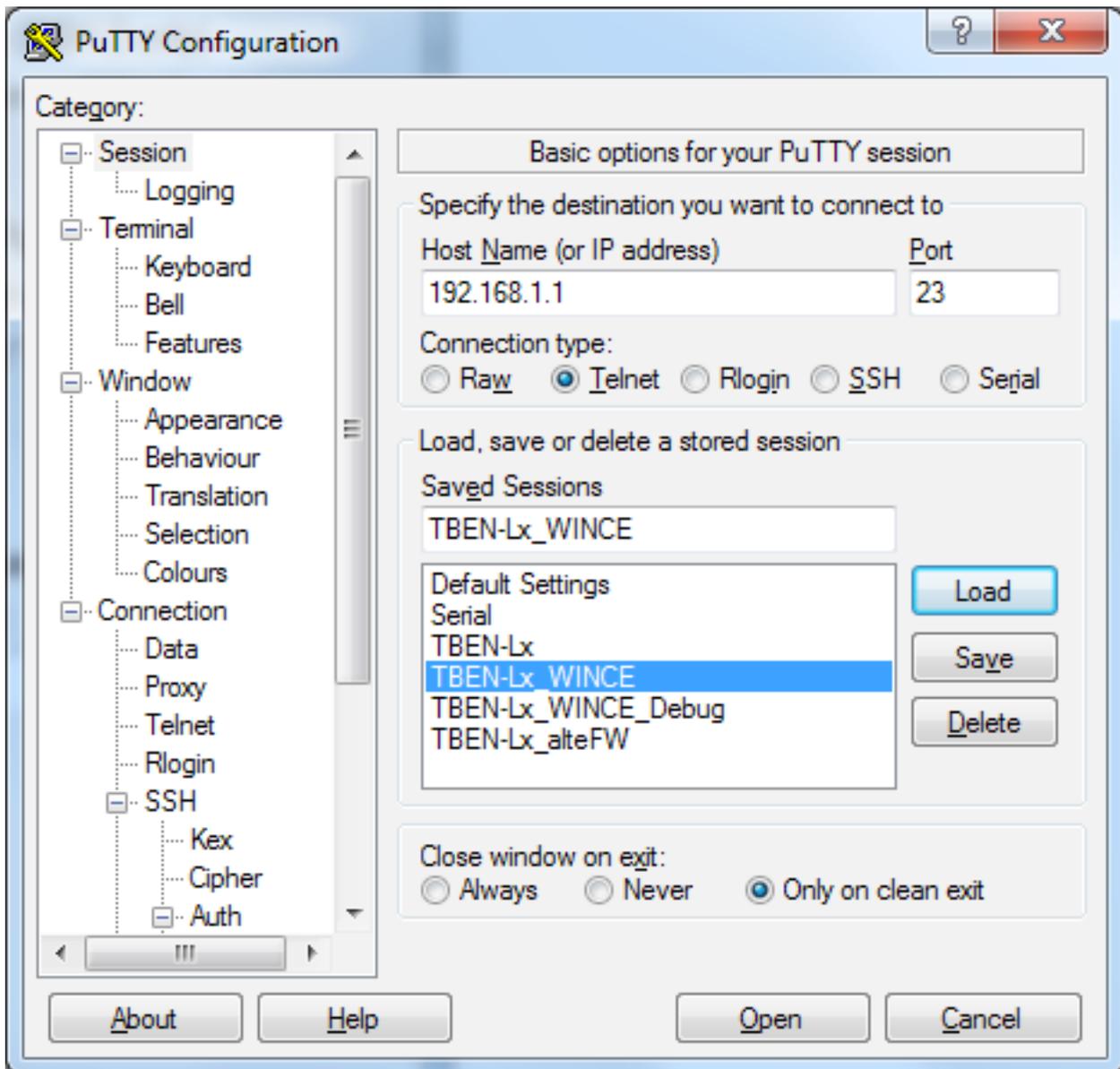


Fig. 26: PuTTY configuration

► Try to open the directory of the device with `cd \windows`.

```

192.168.1.1 - PuTTY
>
>
>
>
> dir

Directory of \

12/12/12  04:00a  <DIR>          Mounted Volume
01/01/06  04:00a  <DIR>          My Documents
01/01/06  04:00a  <DIR>          Temp
01/01/06  04:00a  <DIR>          Windows

Found 4 file(s). Total size 0 bytes.
1 Dir(s) 124346368 bytes free

> cd windows
\windows> dir

Directory of \windows

01/01/06  12:00p          89 FTPD.tmp
01/01/06  04:00a          0 initobj.dat
01/01/06  04:00a  <DIR>          Keys
10/21/15  06:06a          77824 cmd.exe
10/28/16  04:29a          40960 timezones.dll
10/21/15  06:08a          28672 ping.exe
10/21/15  06:08a          40960 ipconfig.exe
10/21/15  06:08a          28672 ndisconfig.exe
10/21/15  06:06a          45056 route.exe
01/20/17  02:37a          36864 netstat.exe
10/21/15  06:08a          28672 tracert.exe
10/21/15  06:06a          36864 ipv6.exe
01/20/17  02:36a          11614 ceconfig.h
02/17/16  04:51a          9932 firmware.bin
02/17/16  04:51a          65136 clientshutdown3.exe
02/17/16  04:51a          67160 CMAccept3.exe
02/17/16  04:51a          113256 ConmanClient3.exe
02/17/16  04:51a          80520 DeviceAgentTransport3.dll
02/17/16  04:51a          112208 eDbgTL3.dll
02/17/16  04:51a          92272 TcpConnectionA3.dll
02/17/16  04:51a          214960 vsdebugeng.impl.resources.dll
02/17/16  04:51a          154176 edm3.exe
02/17/16  04:51a          660368 MSDIA110.dll
02/17/16  04:51a          96144 msvsmon.exe
02/17/16  04:51a          364 msvsmon.exe.config
02/17/16  04:51a          1144216 VSDebugEng.dll
02/17/16  04:51a          594848 vsdebugeng.impl.dll
02/14/17  05:27a          24576 update.exe
02/14/17  05:27a          32768 XMC_Load.exe
02/14/17  05:27a          20480 XMC_CheckVersion.exe
04/19/16  12:23a          9366 XMC1x_ASCLoader.hex
09/06/16  10:15p          25722 83000xxx_V0.1.7.0.hex

Found 32 file(s). Total size 3894719 bytes.
1 Dir(s) 124346368 bytes free

\windows>

```

Fig. 27: Directory \windows in PuTTY

Files can be exchanged between the host PC and the device via WinSCP.

- Start WinSCP.
- Log onto the device in WinSCP.
- Activate the “Anonymous” checkbox.

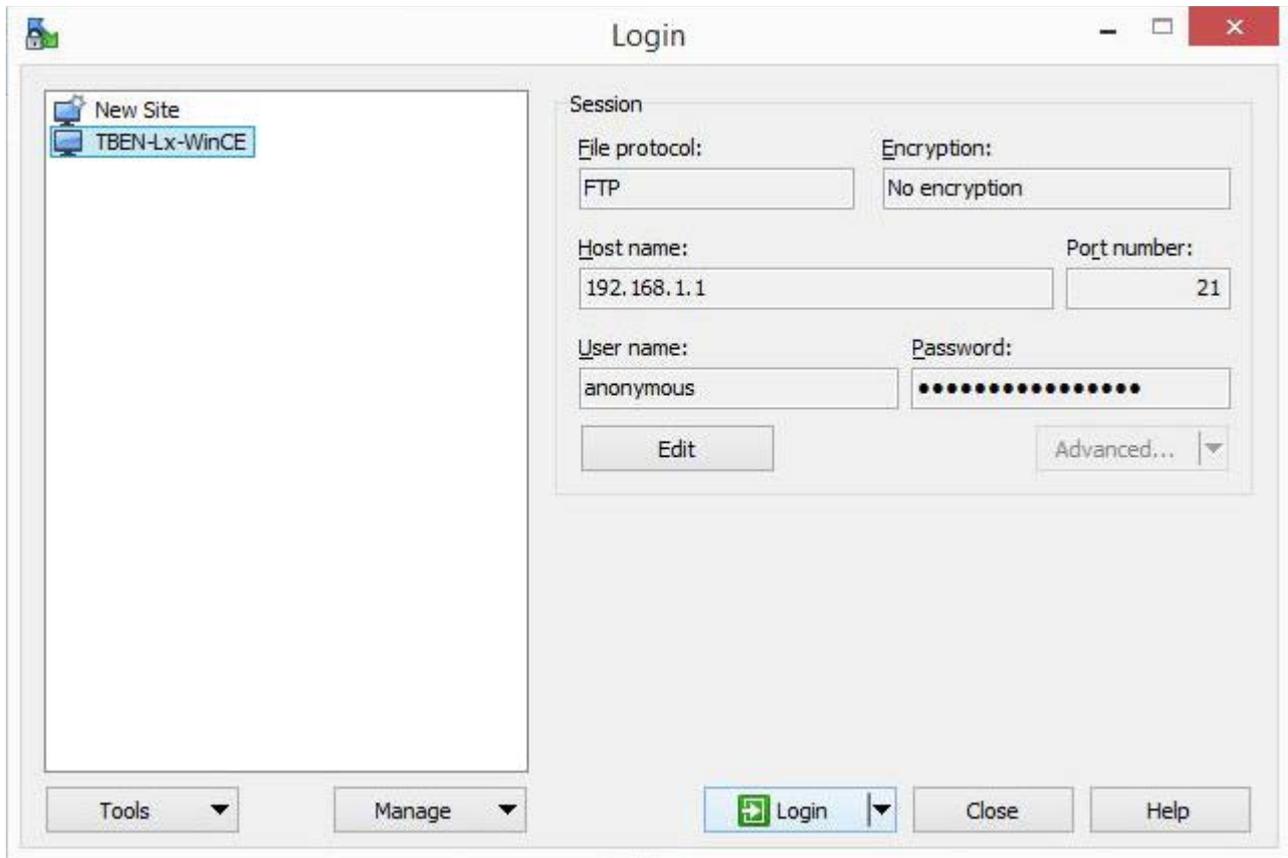


Fig. 28: Logging on in WinSCP

- In WinSCP, select the folders on the host PC and on the device between which you want to exchange files.

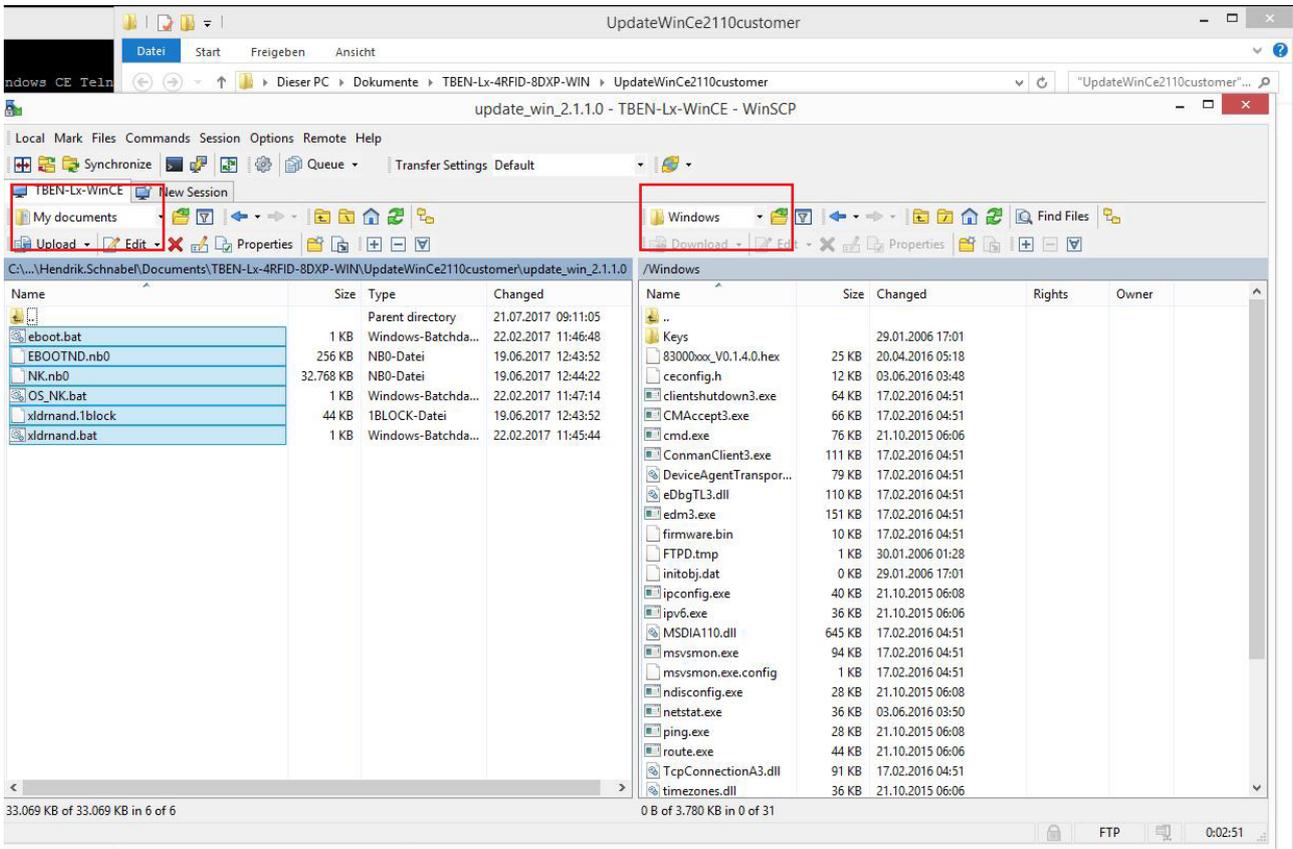


Fig. 29: WinSCP – selecting folders

► Click "Upload" to transfer the files to the device.

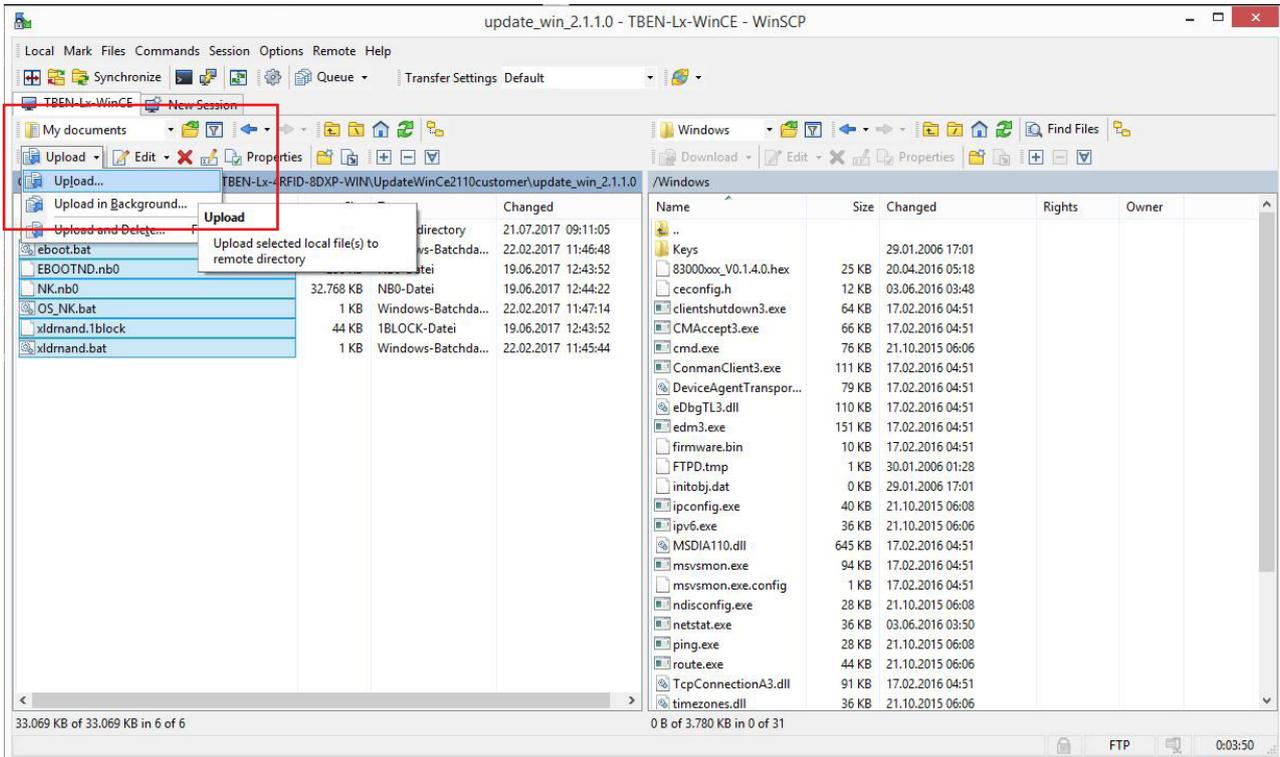


Fig. 30: Transferring files to the device

► Update the view in PuTTY.

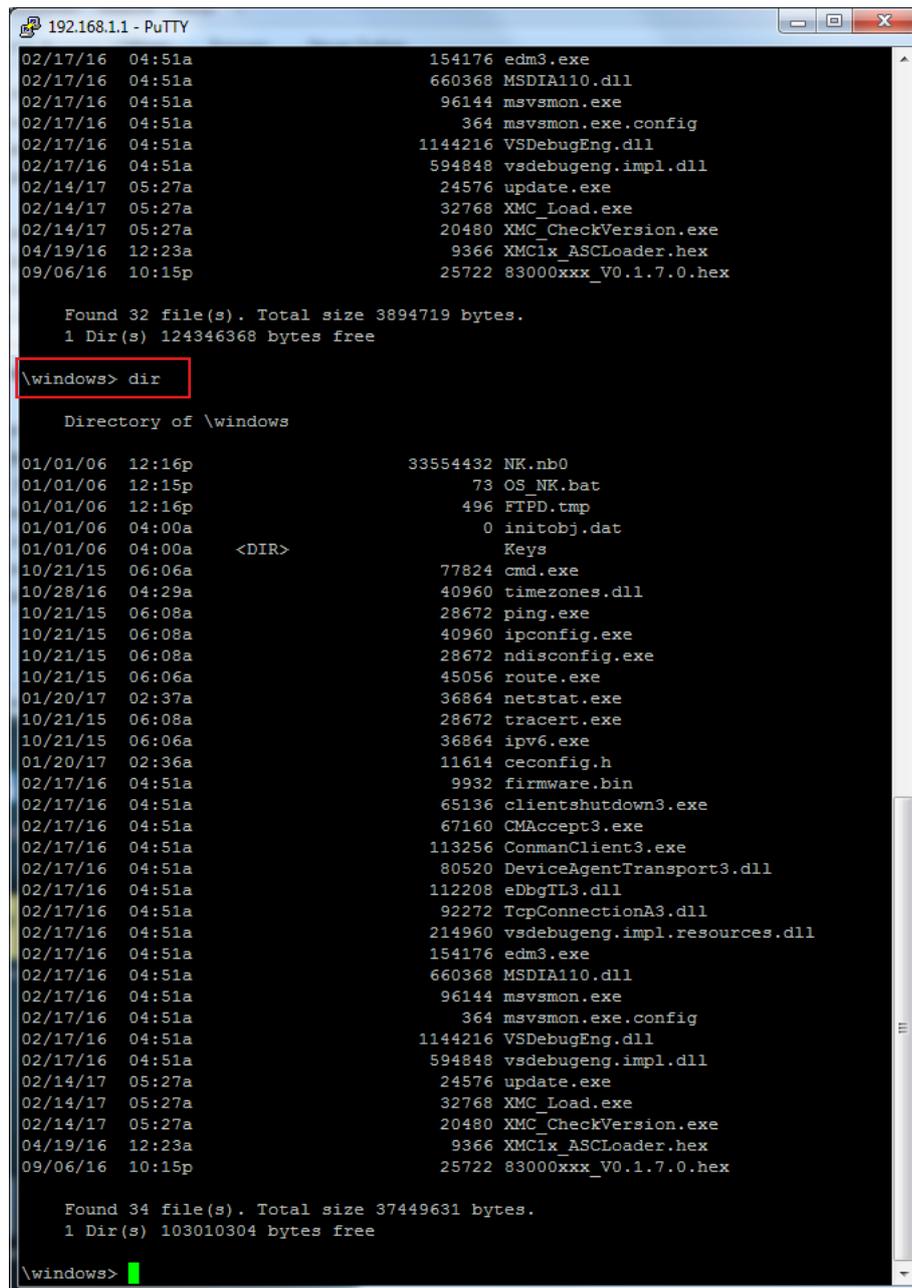


Fig. 31: PuTTY – updated view

- Start the firmware update: Enter OS\_NK.bat and confirm with ENTER.

**NOTE**

Instead of OS\_NK.bat, the firmware update can also be started using the commands xldrnan.bat and eboot.bat.

```

192.168.1.1 - PuTTY
02/17/16 04:51a          154176 edm3.exe
02/17/16 04:51a          660368 MSDIA110.dll
02/17/16 04:51a          96144  msvsmon.exe
02/17/16 04:51a           364  msvsmon.exe.config
02/17/16 04:51a          1144216 VSDebugEng.dll
02/17/16 04:51a          594848 vsdebugeng.impl.dll
02/14/17 05:27a          24576  update.exe
02/14/17 05:27a          32768  XMC_Load.exe
02/14/17 05:27a          20480  XMC_CheckVersion.exe
04/19/16 12:23a           9366  XMC1x_ASCLoader.hex
09/06/16 10:15p          25722  83000xxx_V0.1.7.0.hex

Found 32 file(s). Total size 3894719 bytes.
1 Dir(s) 124346368 bytes free

\windows> dir

Directory of \windows

01/01/06 12:16p          33554432 NK.nb0
01/01/06 12:15p           73  OS_NK.bat
01/01/06 12:16p          496  FTPD.tmp
01/01/06 04:00a           0  initobj.dat
01/01/06 04:00a      <DIR>
                                Keys
10/21/15 06:06a          77824  cmd.exe
10/28/16 04:29a          40960  timezones.dll
10/21/15 06:08a          28672  ping.exe
10/21/15 06:08a          40960  ipconfig.exe
10/21/15 06:08a          28672  ndisconfig.exe
10/21/15 06:06a          45056  route.exe
01/20/17 02:37a          36864  netstat.exe
10/21/15 06:08a          28672  tracert.exe
10/21/15 06:06a          36864  ipv6.exe
01/20/17 02:36a          11614  ceconfig.h
02/17/16 04:51a           9932  firmware.bin
02/17/16 04:51a          65136  clientshutdown3.exe
02/17/16 04:51a          67160  CMAccept3.exe
02/17/16 04:51a          113256  ConmanClient3.exe
02/17/16 04:51a          80520  DeviceAgentTransport3.dll
02/17/16 04:51a          112208  eDbgTL3.dll
02/17/16 04:51a          92272  TcpConnectionA3.dll
02/17/16 04:51a          214960  vsdebugeng.impl.resources.dll
02/17/16 04:51a          154176  edm3.exe
02/17/16 04:51a          660368  MSDIA110.dll
02/17/16 04:51a          96144   msvsmon.exe
02/17/16 04:51a           364   msvsmon.exe.config
02/17/16 04:51a          1144216 VSDebugEng.dll
02/17/16 04:51a          594848  vsdebugeng.impl.dll
02/14/17 05:27a          24576   update.exe
02/14/17 05:27a          32768   XMC_Load.exe
02/14/17 05:27a          20480   XMC_CheckVersion.exe
04/19/16 12:23a           9366   XMC1x_ASCLoader.hex
09/06/16 10:15p          25722   83000xxx_V0.1.7.0.hex

Found 34 file(s). Total size 37449631 bytes.
1 Dir(s) 103010304 bytes free

\windows> OS_NK.bat

```

Fig. 32: Starting the firmware update

The update in progress is shown with "update NK...".

```

192.168.1.1 - PuTTY
02/17/16 04:51a      1144216 VSDebugEng.dll
02/17/16 04:51a      594848 vsdebugeng.impl.dll
02/14/17 05:27a      24576 update.exe
02/14/17 05:27a      32768 XMC_Load.exe
02/14/17 05:27a      20480 XMC_CheckVersion.exe
04/19/16 12:23a      9366 XMC1x_ASCLoader.hex
09/06/16 10:15p      25722 83000xxx_V0.1.7.0.hex

Found 32 file(s). Total size 3894719 bytes.
1 Dir(s) 124346368 bytes free

\windows> dir

Directory of \windows

01/01/06 12:16p      33554432 NK.nb0
01/01/06 12:15p      73 OS_NK.bat
01/01/06 12:16p      496 FTPD.tmp
01/01/06 04:00a      0 initobj.dat
01/01/06 04:00a      <DIR> Keys
10/21/15 06:06a      77824 cmd.exe
10/28/16 04:29a      40960 timezones.dll
10/21/15 06:08a      28672 ping.exe
10/21/15 06:08a      40960 ipconfig.exe
10/21/15 06:08a      28672 ndisconfig.exe
10/21/15 06:06a      45056 route.exe
01/20/17 02:37a      36864 netstat.exe
10/21/15 06:08a      28672 tracert.exe
10/21/15 06:06a      36864 ipv6.exe
01/20/17 02:36a      11614 ceconfig.h
02/17/16 04:51a      9932 firmware.bin
02/17/16 04:51a      65136 clientshutdown3.exe
02/17/16 04:51a      67160 CMAccept3.exe
02/17/16 04:51a      113256 ConmanClient3.exe
02/17/16 04:51a      80520 DeviceAgentTransport3.dll
02/17/16 04:51a      112208 eDbgTL3.dll
02/17/16 04:51a      92272 TcpConnectionA3.dll
02/17/16 04:51a      214960 vsdebugeng.impl.resources.dll
02/17/16 04:51a      154176 edm3.exe
02/17/16 04:51a      660368 MSDIA110.dll
02/17/16 04:51a      96144 msvsmon.exe
02/17/16 04:51a      364 msvsmon.exe.config
02/17/16 04:51a      1144216 VSDebugEng.dll
02/17/16 04:51a      594848 vsdebugeng.impl.dll
02/14/17 05:27a      24576 update.exe
02/14/17 05:27a      32768 XMC_Load.exe
02/14/17 05:27a      20480 XMC_CheckVersion.exe
04/19/16 12:23a      9366 XMC1x_ASCLoader.hex
09/06/16 10:15p      25722 83000xxx_V0.1.7.0.hex

Found 34 file(s). Total size 37449631 bytes.
1 Dir(s) 103010304 bytes free

\windows> OS_NK.bat
\windows> rem Update OS WIN-CE im Verzeichnis windows
\windows> update.exe i \windows\NK.nb0
update NK...

```

Fig. 33: Firmware update in progress

The firmware update is complete when the “done” message appears.

```

192.168.1.1 - PuTTY
02/14/17 05:27a      24576 update.exe
02/14/17 05:27a      32768 XMC_Load.exe
02/14/17 05:27a      20480 XMC_CheckVersion.exe
04/19/16 12:23a       9366 XMClx_ASCLoader.hex
09/06/16 10:15p      25722 83000xxx_V0.1.7.0.hex

Found 32 file(s). Total size 3894719 bytes.
1 Dir(s) 124346368 bytes free

\windows> dir

Directory of \windows

01/01/06 12:16p      33554432 NK.nb0
01/01/06 12:15p         73 OS_NK.bat
01/01/06 12:16p         496 FTD.tmp
01/01/06 04:00a         0 initobj.dat
01/01/06 04:00a      <DIR>      Keys
10/21/15 06:06a      77824 cmd.exe
10/28/16 04:29a      40960 timezones.dll
10/21/15 06:08a      28672 ping.exe
10/21/15 06:08a      40960 ipconfig.exe
10/21/15 06:08a      28672 ndisconfig.exe
10/21/15 06:06a      45056 route.exe
01/20/17 02:37a      36864 netstat.exe
10/21/15 06:08a      28672 tracert.exe
10/21/15 06:06a      36864 ipv6.exe
01/20/17 02:36a      11614 ceconfig.h
02/17/16 04:51a         9932 firmware.bin
02/17/16 04:51a      65136 clientshutdown3.exe
02/17/16 04:51a      67160 CMAccept3.exe
02/17/16 04:51a      113256 ConmanClient3.exe
02/17/16 04:51a      80520 DeviceAgentTransport3.dll
02/17/16 04:51a      112208 eDbgTL3.dll
02/17/16 04:51a      92272 TcpConnectionA3.dll
02/17/16 04:51a      214960 vsdebugeng.impl.resources.dll
02/17/16 04:51a      154176 edm3.exe
02/17/16 04:51a      660368 MSDIA110.dll
02/17/16 04:51a      96144 msvsmon.exe
02/17/16 04:51a         364 msvsmon.exe.config
02/17/16 04:51a      1144216 VSDebugEng.dll
02/17/16 04:51a      594848 vsdebugeng.impl.dll
02/14/17 05:27a      24576 update.exe
02/14/17 05:27a      32768 XMC_Load.exe
02/14/17 05:27a      20480 XMC_CheckVersion.exe
04/19/16 12:23a       9366 XMClx_ASCLoader.hex
09/06/16 10:15p      25722 83000xxx_V0.1.7.0.hex

Found 34 file(s). Total size 37449631 bytes.
1 Dir(s) 103010304 bytes free

\windows> OS_NK.bat
\windows> rem Update OS WIN-CE im Verzeichnis windows
\windows> update.exe i \windows\NK.nb0
update NK...
done
\windows>
\windows>

```

Fig. 34: Firmware update complete

► Complete the firmware update: Reset the device's voltage.

## Checking the Firmware Status

From version 2.1.0.0, the firmware status can be displayed via the Turck Service Tool.

- Launch the Turck Service Tool.
- Click "Search".
- The device is displayed. The firmware status is listed under "Version".

Turck Service Tool, Vers. 3.0.0

Your Global Automation Partner

Suchen... (F5) | Ändern (F2) | Wink (F3) | Aktionen (F4) | Zwischenablage | Sprache: DE | Expertenmodus AN | Starte DHCP (F6) | Konfiguration (F7) | ARGEE (F8) | Schließen

Nr.	MAC-Adresse	Name	IP-Adresse	Netzmaske	Gateway	Modus	Gerätetyp	Version	Adapter	ARGEE	Protokoll
1	00:07:46:FF:A2:5B		<a href="http://192.168.1.100">192.168.1.100</a>	255.255.255.0	192.168.1.1	ROTARY	TBEN-L5-4RFID-8DXP-WIN	2.1.1.0	192.168.1.60	-	Turck

Fig. 35: TBEN-L5-4RFID-8DXP in the Turck Service Tool

## 11 Repairs

The device is not intended for repair by the user. If the device is faulty, please take it out of operation. If you are returning the device to Turck, please note our return terms and conditions.

### 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 at <http://www.turck.de/en/retoure-service-6079.php> and must be filled in completely and affixed to the outside of the packaging such that it is secure and cannot be impaired by adverse weather.

## 12 Disposal



The device must be properly disposed of, not in general household waste.

## 13 Technical Data

<b>Technical data</b>	
<b>Supply</b>	
Power supply	24 VDC
Admissible range	18...30 VDC
Total current	V1 max. 8 A, V2 max. 9 A at 70 °C per module
RFID supply	2 A per channel at 70 °C
Sensor/actuator supply	2 A per slot at 70 °C
Potential separation	V1 and V2 voltage groups galvanically isolated
Dielectric strength	Up to 500 VDC V1 and V2 over Ethernet
Power loss	Typically ≤ 5 W
<b>System description</b>	
Processor	Cortex A8 800 MHz
Memory	256 MB Flash ROM; 512 MB DDR3 RAM
Real-time clock	Yes
Operating system	Windows Embedded Compact 2013
<b>System data</b>	
Transmission rate	Ethernet 10 Mbps/100 Mbps
Connectivity	2 × M12, 4-pin, D-coded
<b>RFID</b>	
Number of channels	4
Connectivity	M12, 5-pin
Supply	2 A per channel at 70 °C, short-circuit proof
<b>Digital inputs</b>	
Number of channels	8
Connectivity	M12, 5-pin
Input type	PNP
Type of input diagnostics	Channel diagnostics
Switching threshold	EN 61131-2 Type 3, PNP
Low-level signal voltage	< 5 V
High-level signal voltage	> 11 V
Low-level signal current	< 1.5 mA
High-level signal current	> 2 mA
Potential separation	Galvanic isolation to P1/P2
Dielectric strength	Up to 500 VDC (V1 and V1 over Ethernet)
Cable length	Max. 50 m

<b>Technical data</b>	
<b>Digital outputs</b>	
Number of channels	8
Connectivity — outputs	M12, 5-pin
Output type	PNP
Type of output diagnostics	Channel diagnostics
Output voltage	24 VDC from potential group
Output current per channel	2.0 A, short-circuit proof, max. 4.0 A per port
Simultaneity factor	0.56
Load type	Resistive, inductive, lamp load
Short-circuit protection	Yes
Potential separation	Galvanic isolation to P1/P2
Dielectric strength	Up to 500 VDC (V1 and V1 over Ethernet)
<b>Standard/directive conformity</b>	
Vibration test	Acc. to EN 60068-2-6
Acceleration	Up to 20 g
Shock test	Acc. to EN 60068-2-27
Drop and topple	Acc. to IEC 60068-2-31/IEC 60068-2-32
Electromagnetic compatibility	Acc. to EN 61131-2
Approvals and certificates	CE
UL conditions	cULus LISTED 21 W2, Encl.Type 1 IND.CONT.EQ.
<b>General information</b>	
Dimensions (W x L x H)	60.4 × 230.4 × 39 mm
Operating temperature	-40 °C to +70 °C
Storage temperature	-40 °C to +70 °C
Operating altitude	Max. 5000 m
Protection class	IP65/IP67/IP69K
MTTF	
Housing material	PA6-GF30
Housing color	Black
Window material	Lexan
Screw material	303 stainless steel
Halogen-free	Yes
Mounting	2 mounting holes, Ø 6.3 mm

# TURCK

Over 30 subsidiaries and over  
60 representations worldwide!

D500062 | 2018/01



[www.turck.com](http://www.turck.com)