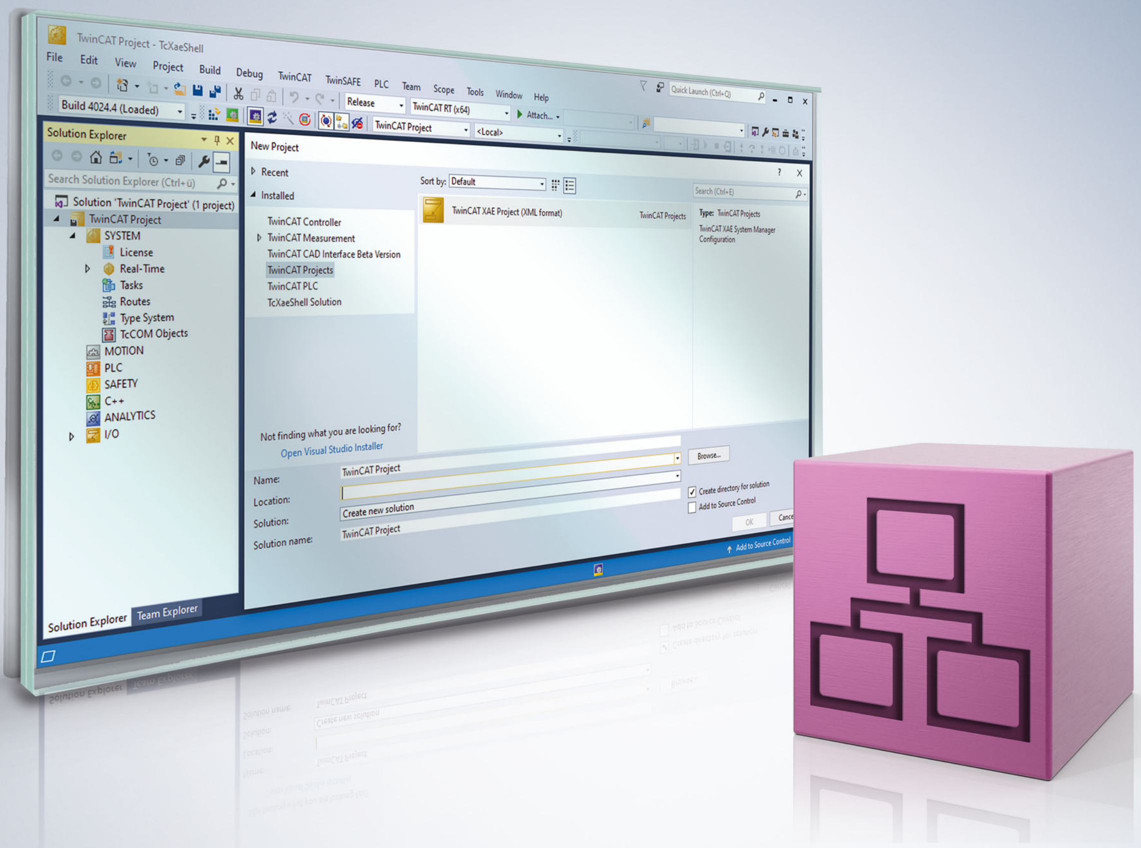
BECKHOFF New Automation Technology

Manual | EN

TF6600

TwinCAT 3 | RFID Reader Communication



2023-11-22 | Version: 1.3.1

Table of contents

1 Foreword

1.1 Notes on the documentation

This description is intended exclusively for trained specialists in control and automation technology who are familiar with the applicable national standards.
For installation and commissioning of the components, it is absolutely necessary to observe the documentation and the following notes and explanations.
The qualified personnel is obliged to always use the currently valid documentation.

The responsible staff must ensure that the application or use of the products described satisfies all requirements for safety, including all the relevant laws, regulations, guidelines, and standards.

Disclaimer

The documentation has been prepared with care. The products described are, however, constantly under development.
We reserve the right to revise and change the documentation at any time and without notice.
No claims to modify products that have already been supplied may be made on the basis of the data, diagrams, and descriptions in this documentation.

Trademarks

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® and XPlanar® are registered and licensed trademarks of Beckhoff Automation GmbH.
If third parties make use of designations or trademarks used in this publication for their own purposes, this could infringe upon the rights of the owners of the said designations.

Patents

The EtherCAT Technology is covered, including but not limited to the following patent applications and patents:
EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702
and similar applications and registrations in several other countries.

EtherCAT® is registered trademark and patented technology, licensed by Beckhoff Automation GmbH, Germany

Copyright

1.2 For your safety

Safety regulations

Read the following explanations for your safety.
Always observe and follow product-specific safety instructions, which you may find at the appropriate places in this document.

Exclusion of liability

All the components are supplied in particular hardware and software configurations which are appropriate for the application. Modifications to hardware or software configurations other than those described in the documentation are not permitted, and nullify the liability of Beckhoff Automation GmbH & Co. KG.

Personnel qualification

This description is only intended for trained specialists in control, automation, and drive technology who are familiar with the applicable national standards.

Signal words

The signal words used in the documentation are classified below. In order to prevent injury and damage to persons and property, read and follow the safety and warning notices.

Personal injury warnings

⚠ DANGER
Hazard with high risk of death or serious injury.

⚠ WARNING
Hazard with medium risk of death or serious injury.

⚠ CAUTION
There is a low-risk hazard that could result in medium or minor injury.

Warning of damage to property or environment

NOTICE
The environment, equipment, or data may be damaged.

Information on handling the product

This information includes, for example:
recommendations for action, assistance or further information on the product.

1.3 Notes on information security

The products of Beckhoff Automation GmbH & Co. KG (Beckhoff), insofar as they can be accessed online, are equipped with security functions that support the secure operation of plants, systems, machines and networks. Despite the security functions, the creation, implementation and constant updating of a holistic security concept for the operation are necessary to protect the respective plant, system, machine and networks against cyber threats. The products sold by Beckhoff are only part of the overall security concept. The customer is responsible for preventing unauthorized access by third parties to its equipment, systems, machines and networks. The latter should be connected to the corporate network or the Internet only if appropriate protective measures have been set up.

In addition, the recommendations from Beckhoff regarding appropriate protective measures should be observed. Further information regarding information security and industrial security can be found in our https://www.beckhoff.com/secguide.

Beckhoff products and solutions undergo continuous further development. This also applies to security functions. In light of this continuous further development, Beckhoff expressly recommends that the products are kept up to date at all times and that updates are installed for the products once they have been made available. Using outdated or unsupported product versions can increase the risk of cyber threats.

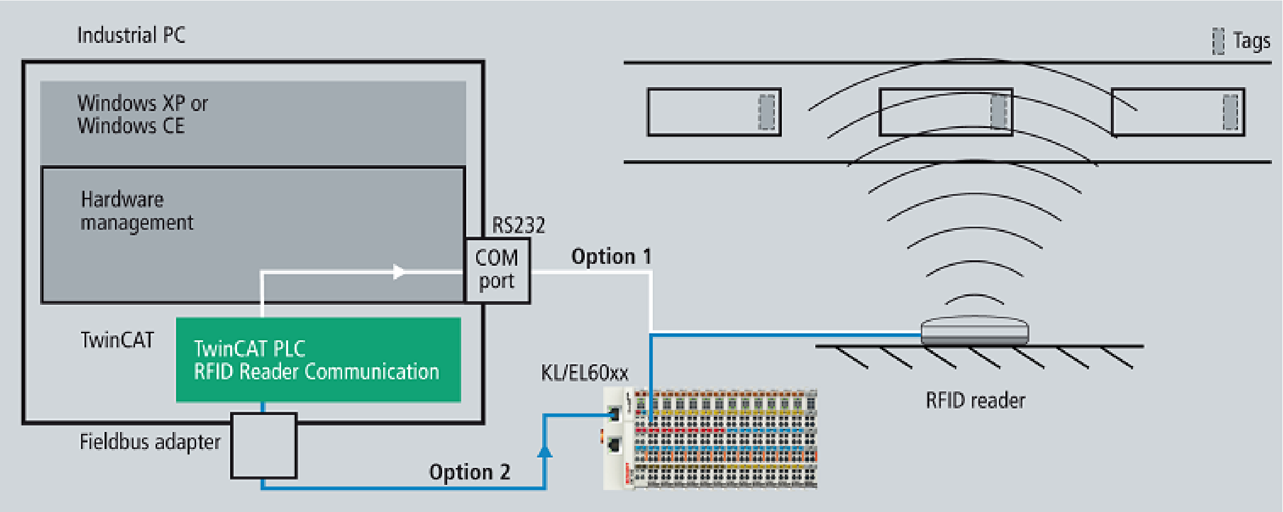
To stay informed about information security for Beckhoff products, subscribe to the RSS feed at https://www.beckhoff.com/secinfo.

2 Overview

The TC3 RFID Reader Communication library makes communication with RFID readers possible from the PLC program. RFID readers can be read-only or read/write devices.

The TwinCAT RFID library facilitates implementation of a wide range of applications using different RFID reader functions. The implementation expenditure is very low, because no manufacturer-specific interface protocol needs to be investigated in detail and implemented. The library automatically deals with frame configuration, telegram composition, command designation, telegram recognition and some other protocol characteristics.

The following figure illustrates an RFID reader application.



Handling of the library is the same for all supported RFID reader models. In the event of a change of manufacturer, only minor changes to the application are required.

An overview of the supported RFID reader models can be found in the section Technical introduction > RFID reader hardware [▶ 15]. Among the supported models are devices of the manufacturers Balluff, Baltech, Deister electronic, Leuze electronic and Pepperl+Fuchs.

For Beckhoff Multi-touch Control Panels the Compact RFID Reader is optionally available. In contrast to the other RFID reader models, the Compact RFID Reader is not addressed from TwinCAT using the TF6600 RFID Reader Communication, but solely using the TF6340 Serial Communication. A product description of the Compact RFID Reader and a PLC example showing the communication with this device can be found in the Online Information System in the section Industrial PC > Compact RFID Reader.

3 Installation

3.1 System requirements

Technical data	Description
Operating system	WinXP, WES, Win7, WES7, Win10
Target platform	PC or CX (x86, x64, ARM): WinXP, WES, Win7, WES7, WEC7, Win10
Minimum platform level	P 20 P 30 for connection via USB / VirtualComPort
TwinCAT version	TwinCAT 3.1.4013 or higher
Required TwinCAT setup level	TwinCAT XAE TC3 SPS
Required TwinCAT license	TF6600 TC3 RFID Reader Communication
TwinCAT PLC library to be integrated	Tc2_RFID Tc2_SerialCom

Depending on the RFID reader model, you may need a proprietary tool for the initial basic configuration (see RFID reader settings and handling [▸ 26]). Also note its system requirements. This presetting can also be performed from another PC. The use of proprietary test tools for the setup is also possible.

3.2 Installation

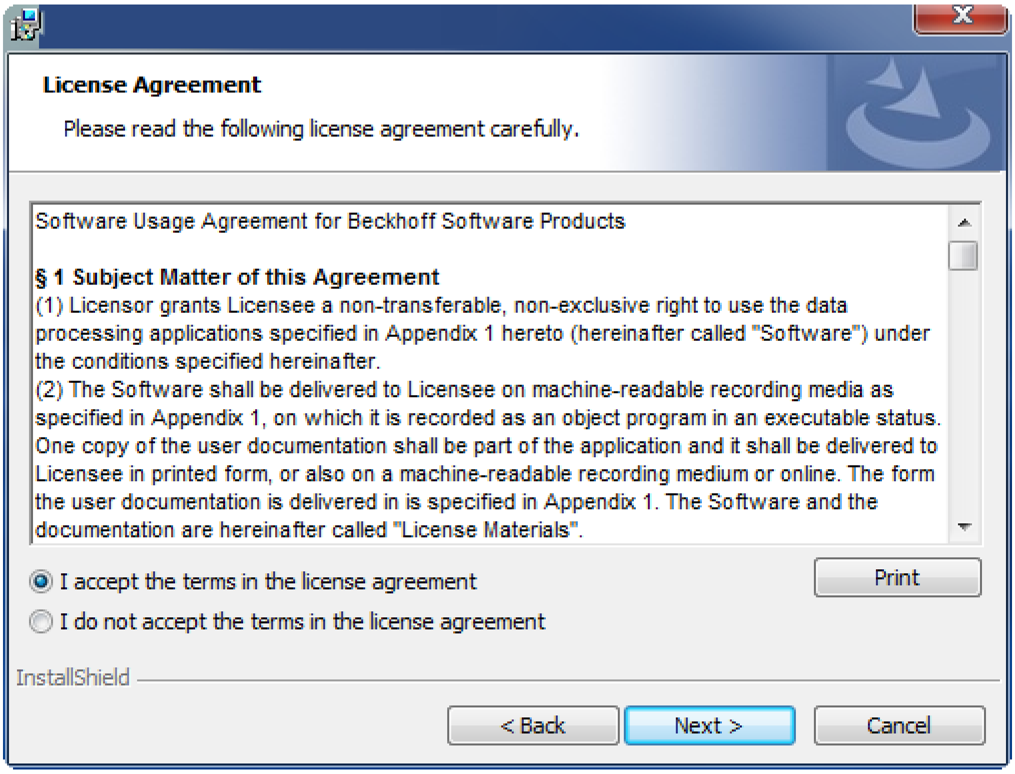
The following section describes how to install the TwinCAT 3 Function for Windows-based operating systems.

✓ The TwinCAT 3 Function setup file was downloaded from the Beckhoff website.

1. Run the setup file as administrator. To do this, select the command **Run as administrator** in the context menu of the file.

   ⇨ The installation dialog opens.

2. Accept the end user licensing agreement and click **Next**.



License Agreement

Please read the following license agreement carefully.

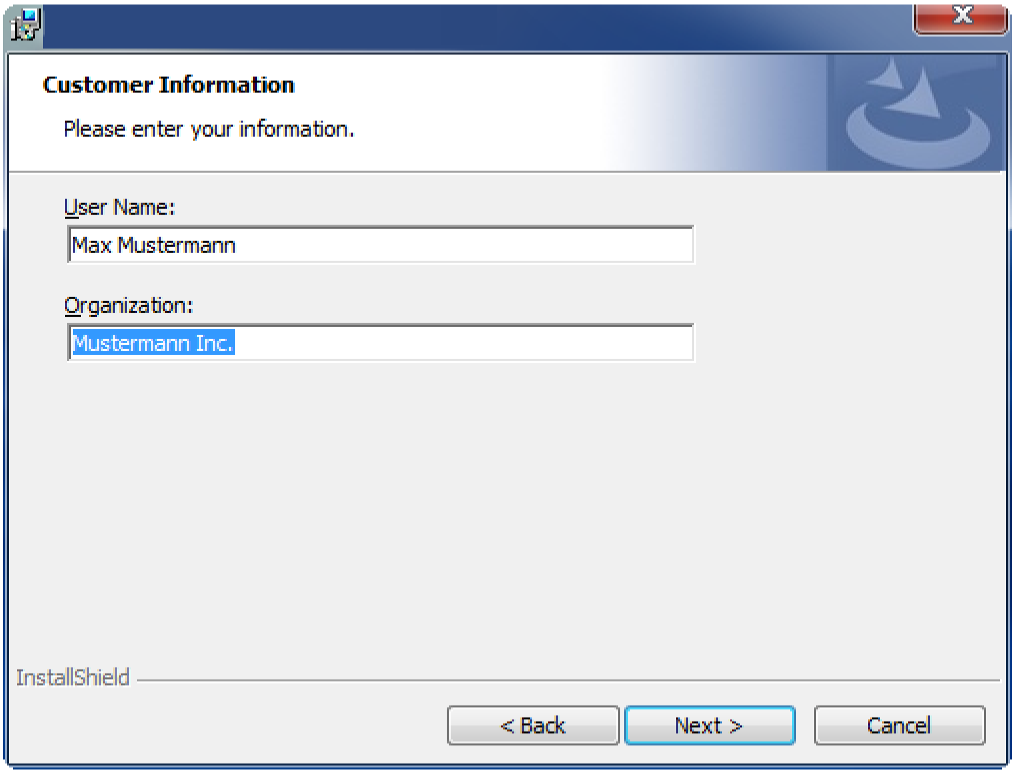
Software Usage Agreement for Beckhoff Software Products

§ 1 Subject Matter of this Agreement
(1) Licensor grants Licensee a non-transferable, non-exclusive right to use the data processing applications specified in Appendix 1 hereto (hereinafter called "Software") under the conditions specified hereinafter.
(2) The Software shall be delivered to Licensee on machine-readable recording media as specified in Appendix 1, on which it is recorded as an object program in an executable status. One copy of the user documentation shall be part of the application and it shall be delivered to Licensee in printed form, or also on a machine-readable recording medium or online. The form the user documentation is delivered in is specified in Appendix 1. The Software and the documentation are hereinafter called "License Materials".

● I accept the terms in the license agreement

○ I do not accept the terms in the license agreement

3. Enter your user data.



Customer Information

Please enter your information.

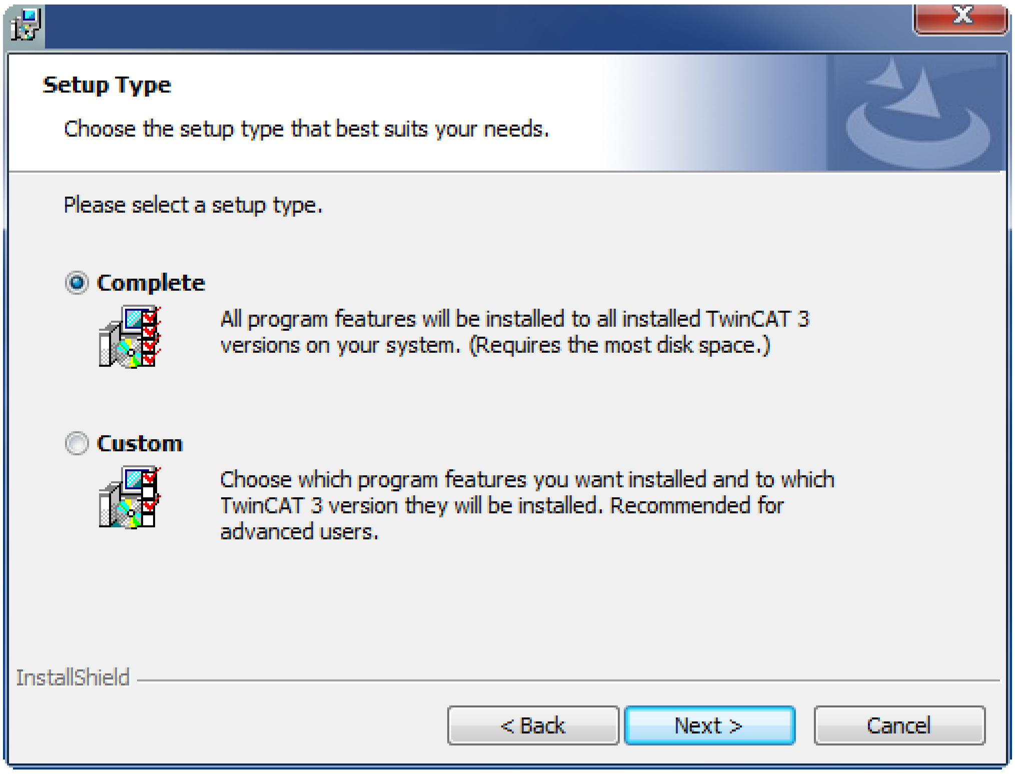
User Name:

Max Mustermann

Organization:

Mustermann Inc.

4. If you want to install the full version of the TwinCAT 3 Function, select **Complete** as installation type. If you want to install the TwinCAT 3 Function components separately, select **Custom**.



Setup Type

Choose the setup type that best suits your needs.

Please select a setup type.

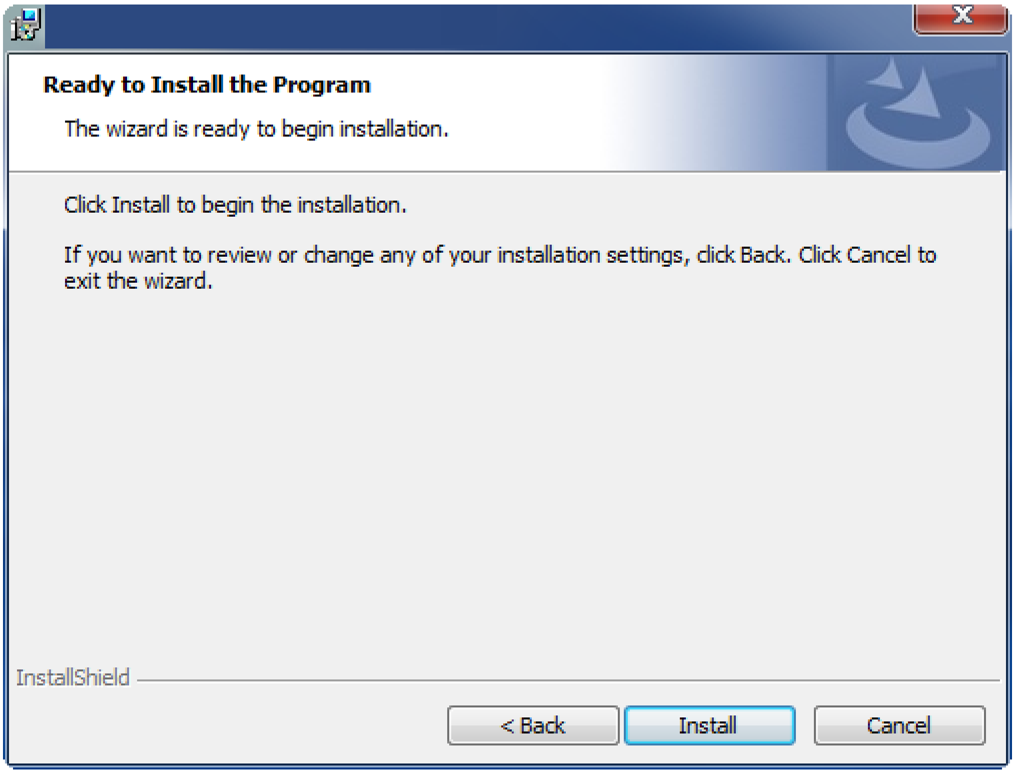
● **Complete**

All program features will be installed to all installed TwinCAT 3 versions on your system. (Requires the most disk space.)

○ **Custom**

Choose which program features you want installed and to which TwinCAT 3 version they will be installed. Recommended for advanced users.

5. Select **Next**, then **Install** to start the installation.



Ready to Install the Program

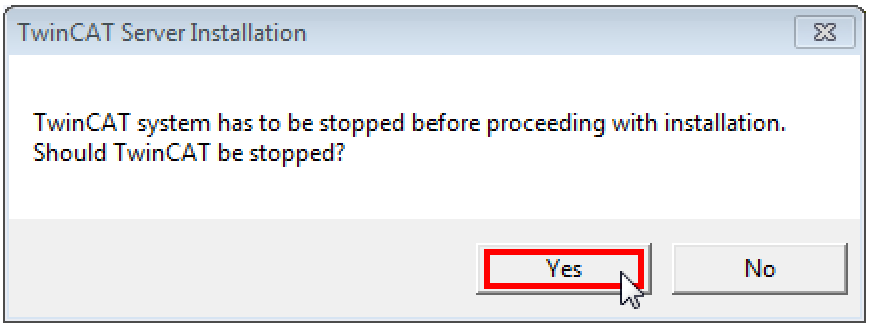
The wizard is ready to begin installation.

Click Install to begin the installation.

If you want to review or change any of your installation settings, click Back. Click Cancel to exit the wizard.

⇨ A dialog box informs you that the TwinCAT system must be stopped to proceed with the installation.

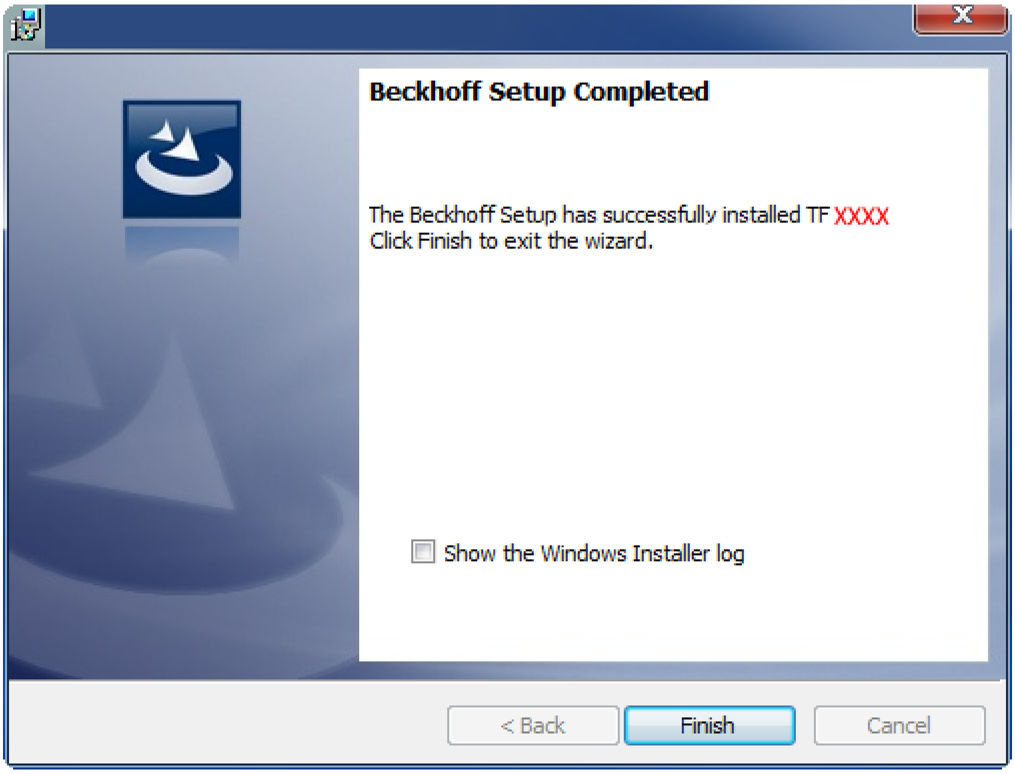
6. Confirm the dialog with **Yes**.



TwinCAT Server Installation

TwinCAT system has to be stopped before proceeding with installation. Should TwinCAT be stopped?

7. Select **Finish** to exit the setup.



Beckhoff Setup Completed

The Beckhoff Setup has successfully installed TF XXXX
Click Finish to exit the wizard.

⇨ The TwinCAT 3 Function has been successfully installed and can be licensed (see Licensing [▶ 12]).

3.3 Licensing

The TwinCAT 3 function can be activated as a full version or as a 7-day test version. Both license types can be activated via the TwinCAT 3 development environment (XAE).

Licensing the full version of a TwinCAT 3 Function

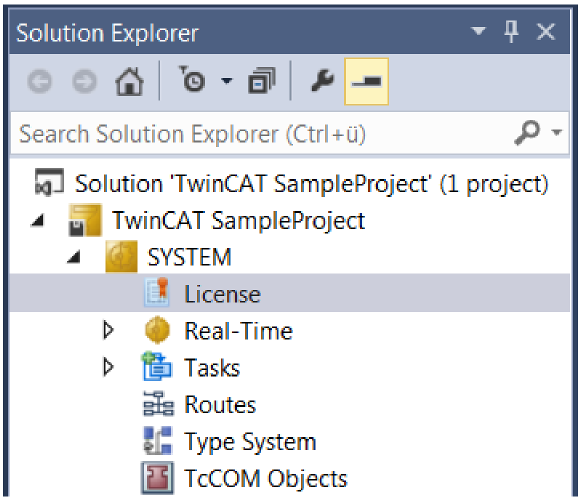
A description of the procedure to license a full version can be found in the Beckhoff Information System in the documentation "TwinCAT 3 Licensing".

Licensing the 7-day test version of a TwinCAT 3 Function

A 7-day test version cannot be enabled for a TwinCAT 3 license dongle.

1. Start the TwinCAT 3 development environment (XAE).
2. Open an existing TwinCAT 3 project or create a new project.
3. If you want to activate the license for a remote device, set the desired target system. To do this, select the target system from the **Choose Target System** drop-down list in the toolbar.
   ⇨ The licensing settings always refer to the selected target system. When the project is activated on the target system, the corresponding TwinCAT 3 licenses are automatically copied to this system.

4. In the **Solution Explorer**, double-click **License** in the **SYSTEM** subtree.



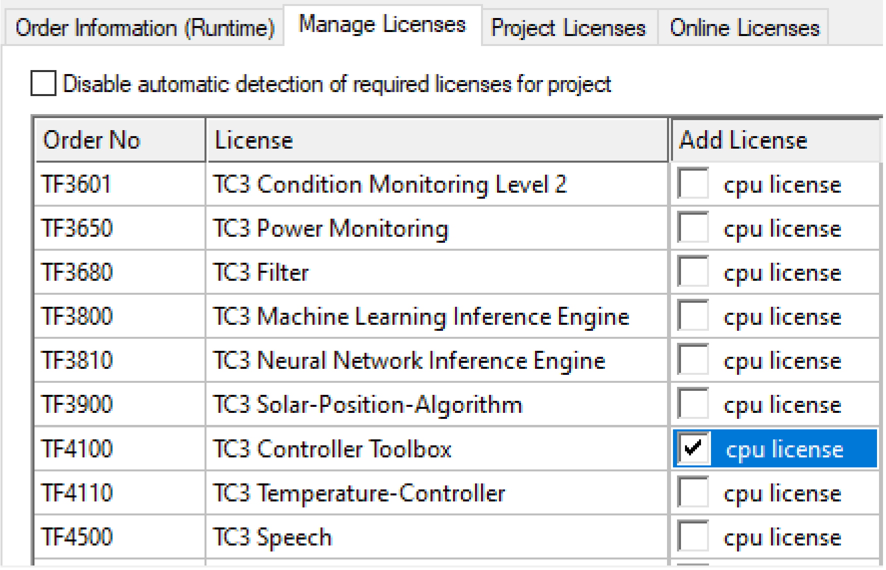
Solution Explorer

Search Solution Explorer (Ctrl+ü)

- Solution 'TwinCAT SampleProject' (1 project)
  - TwinCAT SampleProject
    - SYSTEM
      - License
      - Real-Time
      - Tasks
      - Routes
      - Type System
      - TcCOM Objects

⇨ The TwinCAT 3 license manager opens.

5. Open the **Manage Licenses** tab. In the **Add License** column, check the check box for the license you want to add to your project (e.g. "TF4100 TC3 Controller Toolbox").



Order Information (Runtime) | Manage Licenses | Project Licenses | Online Licenses

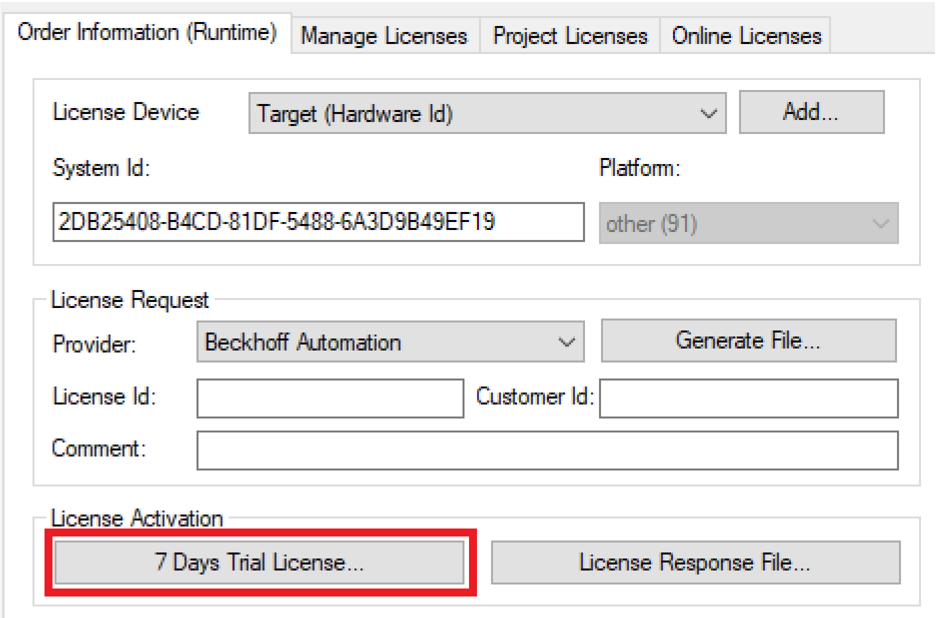
□ Disable automatic detection of required licenses for project

Order No	License	Add License
TF3601	TC3 Condition Monitoring Level 2	□ cpu license
TF3650	TC3 Power Monitoring	□ cpu license
TF3680	TC3 Filter	□ cpu license
TF3800	TC3 Machine Learning Inference Engine	□ cpu license
TF3810	TC3 Neural Network Inference Engine	□ cpu license
TF3900	TC3 Solar-Position-Algorithm	□ cpu license
TF4100	TC3 Controller Toolbox	☑ cpu license
TF4110	TC3 Temperature-Controller	□ cpu license
TF4500	TC3 Speech	□ cpu license

6. Open the **Order Information (Runtime)** tab.

⇨ In the tabular overview of licenses, the previously selected license is displayed with the status “missing”**.**

7. Click **7-Day Trial License...** to activate the 7-day trial license.



License Device: Target (Hardware Id)

System Id: 2DB25408-B4CD-81DF-5488-6A3D9B49EF19

Platform: other (91)

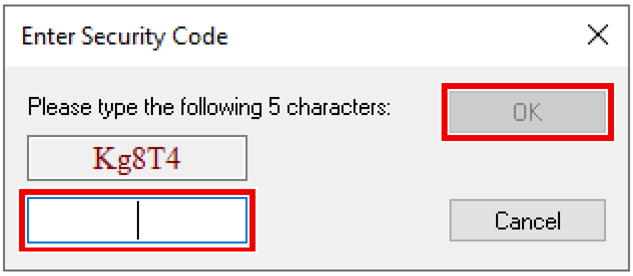
Provider: Beckhoff Automation

License Id:

Customer Id:

Comment:

⇨ A dialog box opens, prompting you to enter the security code displayed in the dialog.



Please type the following 5 characters:

Kg8T4

8. Enter the code exactly as it is displayed and confirm the entry.
9. Confirm the subsequent dialog, which indicates the successful activation.

   ⇨ In the tabular overview of licenses, the license status now indicates the expiry date of the license.
10. Restart the TwinCAT system.

⇨ The 7-day trial version is enabled.

4 Technical introduction

4.1 RFID reader hardware

General notes

- Assembly instructions as well as information on transponder reader handling and reading speeds etc. can be found in the manufacturer's own product manuals.
- RFID readers sometimes offer an external trigger or a switching output. This does not have to be used for the functionality of the TwinCAT RFID library.
- The TwinCAT RFID library does not cover the complete scope of the manufacturer's own RFID communication protocols. Further information can be found in the description of the instruction set of the library function block (see RFID command set [▸ 21]). In addition, the integrated option to send and receive raw data can be used (see command eRFC_Send_RawData).

RFID reader models

The TwinCAT RFID library supports different RFID reader models.

The following table shows which RFID reader models from which manufacturers are supported. The sample photos illustrate the various RFID reader models. They may differ from the actual models that are listed. Also, not every supported model is illustrated by a photo. In some cases, outdated reader firmware versions are not supported.

RFID reader manufacturer	RFID reader model	Sample photo
Balluff	BIS M-400-007 (RS232) BIS M-401-007 (RS232) BIS L-6000-007 (2 read heads) (RS232) BIS L-6020-007 (2 read heads) (RS232)	 Picture credits: BALLUFF
Baltech	ID-engine SD-M1415-ANT1F (RS232 or USB) ID-engine SD-LP-ANT1F (RS232 or USB) ID-engine PAD M1415 (USB) ID-engine ZM-L2M-U2-A1 (RS232)	
Deister electronic	RDL90 (deBus) (RS232, RS485) UDL 500 (deBus) (RS485) PRM5M/2V (deBus) (RS232, RS485)	

RFID reader manufacturer	RFID reader model	Sample photo
Leuze electronic	RFM12 (SL200) (RS232) RFM32 (SL200) (RS232) RFM32ex (SL200) (RS232)	
Pepperl+Fuchs	IDENTControl Compact (2 read heads) [IC-KP2-2HRX-2V1] (RS232) IDENTControl (4 read heads) [IC-KP-R2-V1] (RS232)	

The following table shows which RFID reader models are compatible according to the manufacturer's description and protocol. The compatibility of the listed models as well as other models is, however, not confirmed by Beckhoff. The devices are not officially supported. We recommend contacting Beckhoff Automation before using them.

RFID reader manufacturer	Reader models
Balluff	BIS M-6000
Baltech	ID-engine series (BRP)
Deister electronic	RDL30; RDL150; RDL160; UDL 50; UDL 100; UDL 120; UDL 150; UDL 160; PRM5
Leuze electronic	RFM62 (SL200)
Pepperl+Fuchs	IDENTControl Compact (1 read head)

Other models from the above-mentioned manufacturers, which Beckhoff Automation may not be aware of, may be supported implicitly. According to Deister electronic, the same protocol (deBus) is implemented in other models. It may be the case that some of these models can only be used with limited functionality.

Furthermore, some manufacturers offer their own software to make their devices accessible for Beckhoff TwinCAT systems.

TwinCAT PLC library "Serial Communication"

Further RFID readers are supported with the TwinCAT PLC library "Serial Communication". This library makes it possible to exchange data bytes with any serial device.
This alternative to the TwinCAT RFID library can be useful with read-only RFID readers. It may enable unsupported devices to be used with TwinCAT on a Beckhoff controller. If only the serial number of the transponder is required and this is sent autonomously from the RFID device, the effort involved in evaluating the bytes received is manageable.

Use of the Beckhoff Compact RFID Reader (iDTRONIC)

The Beckhoff Compact RFID Reader is a possibility to integrate an RFID device from the manufacturer iDTRONIC in a push button extension for Beckhoff Multitouch Control Panels. Contrary to the integration of Baltech devices in Beckhoff Panels, the PLC library Tc2_SerialCom (TF6340) is used instead of the PLC library Tc2_RFID (TF6600).

Transponder types

For a complete list of all supported transponder types, please refer to the manual for the respective RFID reader. If necessary, clarify with the manufacturer of the RFID reader or transponder which transponder type is appropriate for the application.

The TwinCAT RFID library uses the data obtained from the serial interface. The manufacturer's serial transmission protocol is therefore decisive for support by the library. The radio frequency used, for example, is irrelevant.

The following table indicates which transponder types are supported for the respective RFID reader models according to the manufacturer. This list is not complete. For complete and more detailed information, please contact the manufacturer of the RFID reader model. Please note that some RFID readers only accept transponders with certain manufacturer IDs. Unfortunately, this restriction cannot be influenced.

RFID reader model	RFID transponder types
Balluff M-401	[13.56 MHz] Fujitsu MB89R118; I-Code SLI; Infineon My-D SRF55(1024 Bytes); Mifare Classic (752 Bytes); TI TagIT HFI (256 Bytes), ...
Balluff L-6000	[125kHz]
Baltech ID-engine SD ANT1F (M1415, LP)	[13.56 MHz] Infineon My-D, Legic Prime, Mifare Classic, ...
Deister electronic RDL90	[13.56 MHz] I-Code SLI; Infineon My-D SRF55(1024 Bytes), ...
Deister electronic UDL 500	[868 MHz] EPCclass1gen2 (12 Bytes), ...
Deister electronic PRM5	[13.56 MHz] Mifare Classic (752 Bytes), ...
Leuze electronic RFM12, RFM32, RFM32ex	[13.56 MHz] I-Code SLI; Infineon My-D SRF55(1024 Bytes); TI TagIT HFI (256 Bytes), ...
Pepperl+Fuchs IDENTControl Compact	[125 kHz; 250 kHz; 13.56 MHz; 2.45 GHz (depends on read head)] I-Code SLI; Fujitsu MB89R118; TI TagIT HFI; Infineon My-D SRF55, ...

These transponder types are additionally available with other memory capacities. Compatibility is hardware-dependent and is not guaranteed. A test is recommended.

Special factory programming of the transponders is sometimes possible. This has no effect on the protocol and must therefore be decided on according to the application in consultation with the manufacturer.

Specific transponder parameters used in the TwinCAT RFID library can be adapted by the user using a special transponder (see ST_RFID_AccessData [▶ 50]).

The TwinCAT RFID library supports transponders up to a maximum size of 64 kilobytes.

Frequency	Transponder Types	HF standards	range	metallic influence	fluid influence	data rate	radio interaction	hardware positioning	temperature influence
LF 125-135 kHz	...	ISO 11784/5, ISO 14223, ISO 18000-2	< 2m	+ +	+	-	-	+ +	++
HF 13,56 MHz	Fujitsu MB89R118, I-Code SLI, Infineon My-D, Legic, Mifare, TI TagIT HFI, ...	ISO 14443, ISO 15693, ISO 18000-3	< 1m	+	+	+	+	+ +	+
UHF 865-868 MHz (EU), 902-928 MHz (USA)	EPCclass1gen2, ...	ISO 18000-6, EPC-Gen2	< 10m	-	-	+	+	+	+
MW 2.45 GHz	...		< 12m	-	-	+ +	+ +	-	-

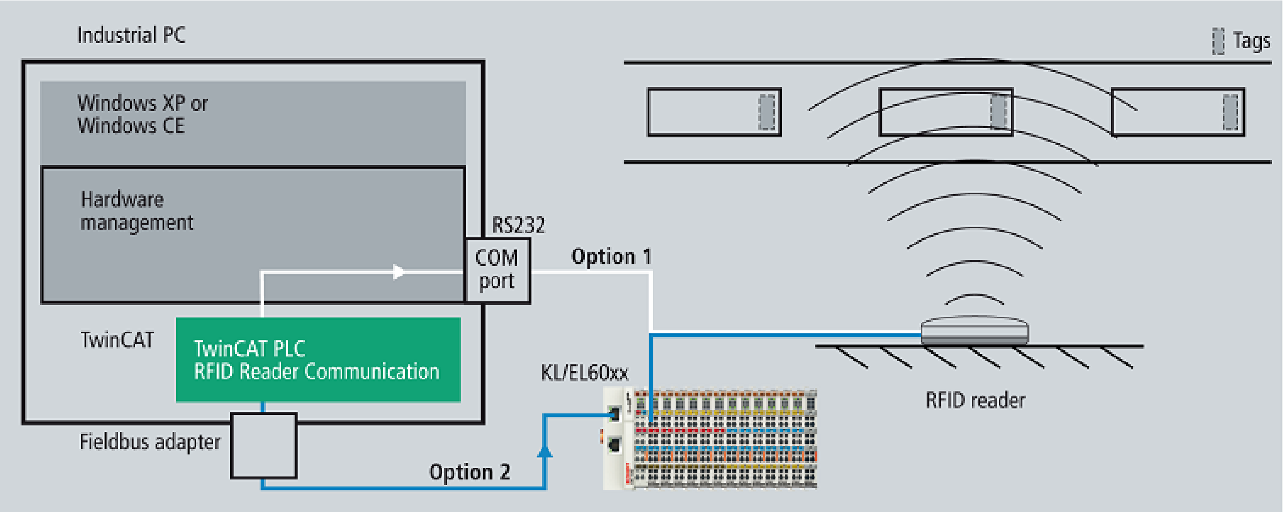
[++ very good; + good; - bad]

4.2 RFID reader connection

All RFID readers supported by this PLC library are connected to the controller via serial communication interfaces (RS 232, RS 422, RS 485 and virtual serial COM ports).

The following Beckhoff products can be used for this:

- Serial EtherCAT Terminals: EL6001, EL6002, EL6021, ...
- Serial K-bus terminals: KL6001/KL6031, KL6021, ...
- COM port of any IPC and Embedded PC with TwinCAT system



A separate connection must be made to a separate terminal for each RFID reader. The TwinCAT RFID library does not support multiple RFID readers on the same RS485 network for the time being.

Setting up serial communication in TwinCAT 3 XAE

Serial data exchange is set up with the function blocks of the TwinCAT PLC library Tc2_SerialCom.

Create a send buffer and a receive buffer of type "ComBuffer". This can take place globally, but does not have to. In addition, you should create two data structures as they are used for serial communication in the TwinCAT System Manager.

If the COM port is used, it looks like this:

```
gPcComRxBuffer          : ComBuffer;
gPcComTxBuffer          : ComBuffer;
PcComInData     AT %I* : PcComInData;
PcComOutData    AT %Q* : PcComOutData;
```

When using a serial terminal, EL6inData22B/EL6outData22B as well as KL6inData5B/KL6outData5B, other data types are possible in addition to PcComInData/PcComOutData.

Link the structures in the TwinCAT System Manager to the channels of the serial interface. When using the ComPort, you must additionally activate the **SyncMode** option on the IO device in the TwinCAT System Manager. The PLC variables must be assigned to the correct (fast) task in the TwinCAT System Manager and linked appropriately from there.

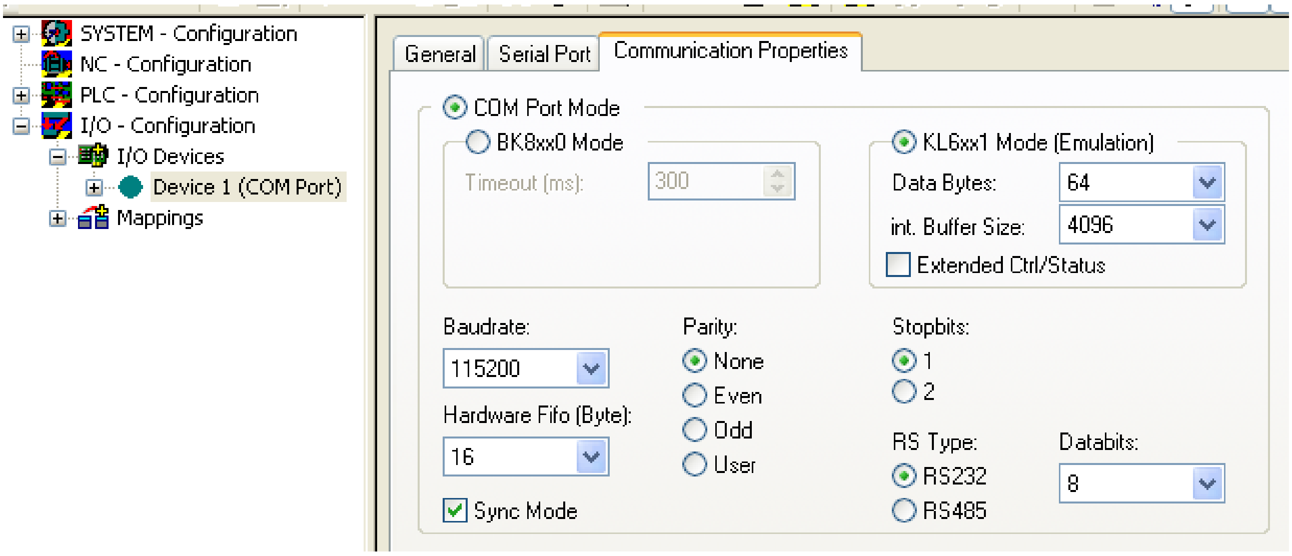
For serial communication, create an instance of the SerialLineControl. This must be called cyclically in a fast task (<= 1 ms). The required task cycle time depends on the application, the data volume, the baud rate and the interface. Depending on the application and interface, it often makes sense to execute this in an additional task that is faster than the application's task.

Example 1: When connecting an RFID device to a COM port and a baud rate of 115,200 baud, a cycle time of 1 ms is required.

Example 2: When connecting an RFID device to an EL6001 and a baud rate of 9,600 baud, a cycle time of 6 ms max. is required.

Further information and explanations on the use of virtual COM ports can be found in the documentation for the PLC library "Serial Communication".

Exemplary display of the COM port settings in the TwinCAT System Manager:



● COM Port Mode

○ BK8xx0 Mode

Timeout (ms): 300

● KL6xx1 Mode (Emulation)

Data Bytes: 64

int. Buffer Size: 4096

□ Extended Ctrl/Status

Baudrate: 115200

Hardware Fifo (Byte): 16

☑ Sync Mode

Parity: ● None ○ Even ○ Odd ○ User

Stopbits: ● 1 ○ 2

RS Type: ● RS232 ○ RS485

Databits: 8

The call of the SerialLineControl is represented by way of example below.

Call as StructuredText in the case of use of the COM port:

```
LineControl(
    Mode       := SERIALLINEMODE_PC_COM_PORT,
    pComIn     := ADR(PcComInData),
    pComOut    := ADR(PcComOutData),
    SizeComIn  := SIZEOF(PcComInData),
    TxBuffer   := gPcComTxBuffer,
    RxBuffer   := gPcComRxBuffer
);
```

Call as StructuredText in the case of use of an EtherCAT terminal:

```
LineControl(
    Mode       := SERIALLINEMODE_EL6_22B,
    pComIn     := ADR(EL6ComInData),
    pComOut    := ADR(EL6ComOutData),
    SizeComIn  := SIZEOF(EL6ComInData),
    TxBuffer   := gEL6ComTxBuffer,
    RxBuffer   := gEL6ComRxBuffer
);
```

Call as StructuredText in the case of use of a K-bus terminal:

```
KL6Config3(
    Execute        := bConfig3,
    Mode           := SERIALLINEMODE_KL6_5B_STANDARD,
    Baudrate       := 9600,
    NoDatabits     := 8,
    Parity         := 0,
    Stopbits       := 1,
    Handshake      := RS485_FULLDUPLEX,
    ContinousMode  := FALSE,
    pComIn         := ADR(KlComInData3),
    pComOut        := ADR(KlComOutData3),
    SizeComIn      := SIZEOF(KlComInData3),
    Busy  => bConfig3Act,
    Done  => bConfig3Done,
    Error => bConfig3Error
);
IF NOT KL6Config3.Busy THEN
    bConfig3 := FALSE;

    LineControl3(
        Mode       := SERIALLINEMODE_KL6_5B_STANDARD,
        pComIn     := ADR(KlComInData3),
        pComOut    := ADR(KlComOutData3),
        SizeComIn  := SIZEOF(KlComInData3),
        TxBuffer   := gKlComTxBuffer3,
        RxBuffer  := gKlComRxBuffer3
    );
END_IF
```

4.3 RFID command set

The following matrix lists the available RFID instruction set.

Because of the fundamental differences between the various RFID reader models, not all instructions can be made available with each model.

The complexity of some proprietary protocols makes it necessary that not every command or every detailed parameterization option can be provided via the TwinCAT PLC library. In individual cases, therefore, recourse can be taken to the less convenient communication option by means of the offered low level interface. Information can be found in the description of the instruction SendRawData [▸ 25] and in the section low-level communication [▸ 40].

Information on the features and characteristics of proprietary protocols can be obtained from the manufacturer for each model and is usually supplied with the device. The user should at least be in possession of these protocol specifications, in order to be able to investigate detailed questions and to read up on the peculiarities of the RFID reader. Reference is already made as far as possible to the peculiarities of the individual RFID readers at special places within this documentation. However, the manufacturer of the RFID devices remains responsible for describing its devices and for guaranteeing their behavior and characteristics. A detailed description of each command and the special behavior of the RFID reader is given in the proprietary protocol specifications. The manufacturer’s proprietary command corresponding to the command listed here is indicated below in each case in *italics*. Details can also be found in the output structure ST_RFID_RawData [▸ 43] of the function block FB_RFIDReader [▸ 34].

Command	Balluff BIS M-40x BIS L-60x0	Baltech IDE SD ANT1F	Deister electronic RDL90	Deister electronic UDL 500	Deister electronic PRM5M/2V	Leuze electronic RFM12; RFM32; RFM32ex	Pepperl+Fuchs IDENTControl Compact
GetReaderVersion [▸ 22]		x	x	x	x	x	x
GetConfig [▸ 22]			x	x		x	x
SetConfig [▸ 23]		x	x	x		x	
GetInventory [▸ 23]	x	x	x			x	x
Polling [▸ 23]			x	x	x		
TriggerOn [▸ 23]			x	x		x	
TriggerOff [▸ 23]			x	x		x	
AbortCommand [▸ 24]			x			x	x
ResetReader [▸ 24]	x	x	x	x	x	x	x
ReadBlock [▸ 24]	x	x	x	x		x	x
WriteBlock [▸ 24]	x	x	x	x		x	x
OutputOn [▸ 25]			x	x		x	
OutputOff [▸ 25]			x	x		x	
FieldOn [▸ 25]		x	x	x		x	
FieldOff [▸ 25]		x	x	x		x	
SendRawData [▸ 25]	x	x	x	x	x	x	x
ChangeDCType [▸ 25]							x

This list is analogous to the enumeration E_RFID_Command [▸ 62] in the TwinCAT RFID library. Successful processing of the requested command by the RFID reader is indicated by the status outputs of the function block and the respective response. A list of possible responses can be found in the description of the enumeration E_RFID_Response [▸ 63].

The commands are explained in detail below:

GetReaderVersion [16#01]

Information on the RFID reader can be queried with this command. Depending upon availability, the model designation, the hardware and software version of the reader etc. are output from the function block in the structure ST_RFID_ReaderInfo [▸ 42].

Equivalent in proprietary protocol:
Deister: 0x02
Leuze: 'V'
Pepperl+Fuchs: 'VE'
Baltech: System GetInfo

GetConfig [16#02]

The current configuration of the RFID reader is queried with this command. All relevant received parameters are explained in the description of structure ST_RFID_Config [▸ 49]. Further information is summarized in section Configuration [▸ 39].

Equivalent in proprietary protocol:
Deister: 0x09
Leuze: 'G'
Pepperl+Fuchs: 'GS'

SetConfig [16#03]

Parameterized configuration settings can be transferred to the RFID reader. For more information about the possible configuration of the RFID reader, see the description of structure ST_RFID_ConfigIn [▸ 48].

Following a configuration command it is recommended to query the current configuration of the reader once again using the GetConfig [▸ 22] command. Further information is summarized in section Configuration [▸ 39].

Equivalent in proprietary protocol:
Baltech: System CfgWriteTLVBlock
Deister: 0x09
Leuze: 'C'

GetInventory [16#04]

This command is used to query the type and serial number of a transponder currently present in the reading field. If no transponder is found, a corresponding response follows.

Pepperl+Fuchs: The parameter iHeadNumber in the structure ST_RFID_Control [▸ 44] determines for which read head the command is to be executed.

Equivalent in proprietary protocol:
Balluff: 'U'
Deister: 0x82
Leuze: 'I'
Pepperl+Fuchs: 'SF' & 'EF'
Baltech: VHLSelect + VHLGetSnr

Polling [16#05]

This command is used to retrieve information from the RFID reader stack. This may be, for example, the serial number of the last transponder. Note that different RFID readers have stacks of different sizes. In some cases only one message is stored.

Presence detection: If this cannot be set via a configuration parameter, it is necessary to keep the reader ready for reading by means of a cyclic polling command so that a transponder within range is automatically detected.

Equivalent in proprietary protocol:
Deister: 0x0B

TriggerOn [16#06]

If the trigger mode is active, then a software trigger instead of hardware trigger can be initiated with this command. The subsequent response telegram is received by the function block of the TwinCAT RFID library. There is no assignment of read transponder data in this case. The received raw data can be taken from the function block interface for further processing.

Equivalent in proprietary protocol:
Deister: 0x85
Leuze: '+'

TriggerOff [16#07]

See TriggerOn [▸ 23].

Equivalent in proprietary protocol:
Deister: 0x85
Leuze: '-'

AbortCommand [16#08]

If a command is being processed by the RFID reader, it is aborted with this command.

Pepperl+Fuchs: The parameter iHeadNumber in the structure ST_RFID_Control [▶ 44] determines for which read head the command is to be executed.

Equivalent in proprietary protocol:
Leuze: 'H'
Pepperl+Fuchs: 'QU'

ResetReader [16#09]

This command causes the RFID reader to perform a reset.

Equivalent in proprietary protocol:
Balluff: 'Q'
Deister: 0x01
Leuze: 'R'
Pepperl+Fuchs: 'RS'
Baltech: System Reset

ReadBlock [16#0A]

This command is used to read a certain number of data bytes from the transponder memory in the form of blocks of a defined size.

The transfer of the input structure ST_RFID_AccessData [▶ 50] is necessary for this command.

Before data is read from a transponder, it is usual to identify and select the transponder (see command GetInventory [▶ 23]).

Pepperl+Fuchs: The parameter iHeadNumber in the structure ST_RFID_Control [▶ 44] determines for which read head the command is to be executed.

Equivalent in proprietary protocol:
Balluff: 'L'
Deister: 0x83
Leuze: 'N'
Pepperl+Fuchs: 'SR' & 'ER'
Baltech: VHLRead

WriteBlock [16#0B]

This command is used to write a certain number of data bytes to the transponder memory in the form of blocks of a defined size.

The transfer of the input structure ST_RFID_AccessData [▶ 50] is necessary for this command.

Before data is written to a transponder, it is usual to identify and select the transponder (see command GetInventory [▶ 23]).

Pepperl+Fuchs: The parameter iHeadNumber in the structure ST_RFID_Control [▶ 44] determines for which read head the command is to be executed.

Equivalent in proprietary protocol:
Balluff: 'P'
Deister: 0x84
Leuze: 'W'
Pepperl+Fuchs: 'SW' & 'EW'
Baltech: VHLWrite

OutputOn [16#0C]

This command sets the switching output of the RFID reader permanently to TRUE. This is only possible if the switching output is not automatically addressed via the configuration.

Equivalent in proprietary protocol:
Deister: 0x0F
Leuze: 'A0FF'

OutputOff [16#0D]

This command sets the switching output of the RFID reader permanently to FALSE. This is only possible if the switching output is not automatically addressed via the configuration.

Equivalent in proprietary protocol:
Deister: 0x0F
Leuze: 'A000'

FieldOn [16#0E]

The RFID field can be turned on with this command.

Equivalent in proprietary protocol:
Deister: 0x81
Leuze: 'F1'
Baltech: System HFReset

FieldOff

The RFID field can be turned off with this command. Depending on the RFID reader model, the field is reactivated in the case of a trigger or another command.

Equivalent in proprietary protocol:
Deister: 0x81
Leuze: 'F2'
Baltech: System HFReset

SendRawData [16#10]

With this command, the RFID function block can be used as a low-level interface. The data to be transmitted are transferred in the control structure [▶ 44] as a pointer. A telegram is composed internally in the library and sent. Arbitrary data can be sent to the RFID reader in this way. The data received as a result is available at the output of the function block in the raw data structure [▶ 43] as an addressed data field. For more information on the process, see section Low-level communication [▶ 40].

When using the command SendRawData [▶ 25], an evaluation of the received response cannot be guaranteed.

Example: If a read command is sent manually as a byte sequence by means of the SendRawData command, then received transponder data are not written at an address specified in ST_RFID_AccessData [▶ 50]. Evaluation/storage of the data should therefore also be done manually with the help of the raw data structure [▶ 43], which is always specified.

ChangeDCType [16#11]

This command can be used to set the transponder type on the read head. Use iDCType in ST_RFID_Control [▶ 44] to specify the type.

Pepperl+Fuchs: The parameter iHeadNumber in the structure ST_RFID_Control [▶ 44] determines for which read head the command is to be executed.

Equivalent in proprietary protocol:
Pepperl+Fuchs: 'CT'

5 Configuration

5.1 RFID reader settings and handling

The following sections describe the individual RFID reader models based on information provided by the device manufacturers. The required settings and the handling are explained for each device.

5.1.1 Balluff

RFID reader settings

For smooth communication between controller and RFID readers, some settings need to be made before the system startup. These include, for example, the baud rate for the serial communication. A proprietary tool from the manufacturer of the RFID reader may be required in order to transfer these settings to the RFID.

For all supported RFID reader models, the following standard data transfer settings have been tried and tested:

Setting	Value
Baud rate (RS232 and RS485)	9600 baud
Parity Bit	none
Data bits	8
Stop bits	1

Depending on the hardware, other parameters can also be set, or the factory settings of the RFID reader can be used. These must then also be adopted in the software reader connection (see RFID reader connection [▶ 19]).

Using the proprietary tools, the following special settings must be parameterized before starting the system:

Setting	Value
Data transmission parameters (see above)	Setting analogous to the values selected in the PLC program
Type of protocol - telegram end identifier	LF CR
Data carrier type	All types (or setting according to needs)
Send CT data immediately	deactivated (or activated – however, no evaluation takes place)
Dynamic operation	deactivated
Send power-on message	deactivated (or activated – however, no evaluation takes place)
CRC16 data check	deactivated
Type and serial number with CT pres.	deactivated (or activated, as required)

If **Type and serial number with CT pres.** is enabled, the RFID reader automatically sends the transponder type and its serial number as soon as a transponder is detected. If a command is sent immediately after the detection of a transponder and receipt of this set message, then the correct attribution of the type of the following response and an associated evaluation cannot be guaranteed. It is advisable to manually query existing transponders using the GetInventory [▶ 23] command. Otherwise at least a short waiting period should be adhered to before sending the command and the structure should be subjected to a test cycle.

If the RFID reader is set up so that telegrams are sent automatically from the reader to the controller (for example, on detection of a transponder by **Type and serial number with CT pres.**), then the following must be observed:

The end identifier (LF CR) is used in this case as a suffix for the recognition of telegrams. Previous data are combined into a telegram as soon as these 2 bytes are detected in the data stream. This may lead to an error and failure to evaluate the telegram. If it is possible for the end identifier to be present within the data in automatically sent telegrams, then a data query must be selected by means of a command call instead of the automatic transmission. The telegrams are recognized reliably as a result of this measure.

RFID reader handling

The function blocks of the library support communication from Balluff readers to transponders with 4-8 bytes serial number.

If Balluff RFID readers are used, the serial number of 13.56 Mhz transponders is rotated byte-by-byte by the library function block. This takes place because the serial number read out from a transponder would otherwise not correspond to the serial number read out on another reader. This allows devices from different manufacturers to be operated together in the same network.

When using a Balluff BIS-L60x0:

- The variable iDCType = 0 must be set (see input structure stCtrl [▸ 44]).
- When the GetInventory [▸ 23] command is called, information from both read heads is returned via the serial interface. However, only the information from the read head selected by stCtrl.iHeadNumber is evaluated and output at the stTranspInfo output.
- If **Type and serial number with CT pres.** is enabled, the RFID reader automatically sends the transponder type and its serial number as soon as a transponder is detected. By default, this only affects the first read head. Switching to the second read head is not directly supported by the library. Furthermore, the number of the read head on which the tag was recognized cannot be assigned (iHeadNumber = 0).

It must be pointed out here that not all peculiarities of every supported RFID reader model can be named here. For detailed information, please refer to the manufacturer's own documentation.

5.1.2 Baltech

If a supported Baltech RFID stand-alone device is used, the TwinCAT RFID library can be used as an interface. Alternatively certain Beckhoff Control Panels or Panel PCs can be used. In these devices an RFID reader can be integrated as an option. In this case an SDK containing the proprietary documentation is provided. The functionality if the TwinCAT library is the same in both cases.

RFID reader settings

For smooth communication between controller and RFID readers, some settings need to be made before the system startup. These include, for example, the baud rate for the serial communication. The proprietary tool "Baltech id-engine explorer" provided by the RFID reader manufacturer can be used to transfer these settings to the RFID reader. The tool can also be used to perform a function test to determine whether the RFID device and the transponder cards are recognized.

The following standard data transmission settings have been tried and tested:

Setting	Value
Baud rate	115200 baud
Parity Bit	none
Data bits	8
Stop bit	1

This matches the factory setting of the supported Baltech RFID devices. Other parameters can be set, if required. These must then also be adopted in the software reader connection (see RFID reader connection [▸ 19]).

The baud rate of the readers can be changed with the tool "Baltech id-engine explorer" (see Baltech documentation: IdEngineExplorer.pdf).

The tool "Baltech id-engine explorer" only runs under Windows XP. It is not available for Windows CE. The baud rate is therefore not configurable under Windows CE.

In the PLC a fast task is required for processing the incoming data. When connecting the RFID device to a COM port and a baud rate of 115,200 baud, a cycle time of 1 ms is required (see RFID reader connection [▸ 19]).

In order to configure the baud rate from the PLC, the following byte sequence can be transferred as a raw data block [0x 1C 00 09 06 00 01 03 00 02 xx xx - with xx xx representing the baud rate, e.g. 9600 baud: 96 units at 100 baud -> 0x 00 60]. Details are explained in section Low-level communication [▸ 40]. This is also possible under Windows CE, if a transfer with the currently set baud rate is possible.

Using the virtual serial COM port (USB)

If the device is connected via USB, the appropriate USB-to-Virtual-Com-Port driver must be installed. If it is a Beckhoff Panel PC, the driver is already pre-installed. The SDK of the RFID device also contains the driver. The virtual COM port is displayed in the Windows Device Manager.

Communication to the driver takes place via Beckhoff TwinCAT serial communication. However, no corresponding device is created in the TwinCAT System Manager, and no link is established there. Further information can be found in the documentation for the PLC library "Serial Communication".

RFID reader handling

The library supports the standard settings for the Baltech communication. "Host Operation" mode is supported as "Operational Mode". Other modes are not supported. The BRP (Baltech Reader Protocol) is used for internal access in "Communication Mode" "Normal Mode". If raw data is sent via the low-level communication option, make sure that the above settings are correctly specified within the frame.

It must be pointed out here that not all peculiarities of every supported RFID reader model can be named here. Therefore you are referred to the manufacturer’s own documentation for more detailed information.

Configuration

If encrypted transponder cards are used, the same key must be available in the RFID device. The Baltech RFID reader is configured once, i.e. the key only has to be specified once. For security reasons the key cannot be read from the device configuration. In keeping with the encryption of a transponder card, a VHL file is stored in the device configuration. Several such VHL files can be stored in order to access different cards without the need for reconfiguration.

There are three ways to transfer such a VHL file into the configuration of the RFID device:

Configuration type	Description
Configuration card	A configuration can be transferred via a configuration card. This is the preferred option.
"Baltech id-engine explorer" tool	The tool can be used to transfer a configuration to the memory of the Baltech RFID device (see Baltech documentation: IdEngineExplorer.pdf). In simple cases the specific configuration can be created directly in the tool. Alternative Baltech offers technical support and can provide a file containing the configuration. The tool "Baltech id-engine explorer" runs on all Windows operating systems from Windows XP. It is not available for Windows CE.
from the PLC	For Mifare Classic cards the transfer of a VHL file configuration can be programmed in the PLC program code. The command SetConfig [▸ 23] transfers the configuration specified at the input in ST_RFID_ConfigIn [▸ 48]. The structure of a Mifare card and the possible settings for key allocation are explained in ST_RFID_CfgStruct_BaltechMifVHLFile [▸ 51]. (Detailed information about Mifare cards can be found in the Baltech document Mifare.pdf in the Baltech SDK)

Transponders

Suitable transponder cards for Baltech RFID devices are available from several manufacturers. If cards with encryption are to be used, Baltech offers preconfigured cards for this purpose.

Manufacturer contact

http://www.baltech.de

5.1.3 Deister electronic

RFID reader settings

For smooth communication between controller and RFID readers, some settings need to be made before the system startup. These include, for example, the baud rate for the serial communication. A proprietary tool from the manufacturer of the RFID reader may be required in order to transfer these settings to the RFID.

For all supported RFID reader models, the following standard data transfer settings have been tried and tested:

Setting	Value
Baud rate (RS232 and RS485)	9600 baud
Parity Bit	none
Data bits	8
Stop bit	1

Depending on the hardware, other parameters can also be set, or the factory settings of the RFID reader can be used. These must then also be adopted in the software reader connection (see RFID reader connection [▸ 19]).

Using the proprietary tools, the following special settings may have to be parameterized before the system is started:

Setting	Value
Data transmission parameters (see above)	Setting analogous to the values selected in the PLC program

RFID reader handling

Here, too, the "Polling" function should be noted, which can be used to call the command repeatedly in cases where the current transponder information is important (see command description [▸ 21]).

In addition there is the peculiarity that, in the case of the proxEntry models, a polling command must be present in order to establish a connection to the transponder. In the case of UDL models, on the other hand, the configuration provides the automatic establishment of a connection to detected transponders, so that no polling command is absolutely necessary.

The block size corresponding to the tag must be configured in the RFID reader configuration.

Deister RDL devices support 4 bytes or 8 bytes block size.

Example: If a block size of 8 bytes is specified for the transponder, then the reader must be configured with the parameter iBlocksize:=8 and the read or write access via the structure ST_RFID_AccessData [▸ 50] must take place with an 8-byte block size.

Deister RDL: A write command can write up to 36 bytes of data at a time. If more data is to be written to the transponder, it must be divided into several commands.

It must be pointed out here that not all peculiarities of every supported RFID reader model can be named here. Therefore you are referred to the manufacturer's own documentation for more detailed information.

Configuration

Note the following when a new configuration is written to the RFID device (command SetConfig [▸ 23]):

Deister RDL devices: Not every combination of configuration parameters is allowed (see ST_RFID_CfgStruct_DeisterRDL [▸ 54]). Disregard of the required dependencies leads to an error (eRFERR_InvalidCfg):

Configuration parameters	Required dependencies

eReadMode = eRFRD_ContinuousRead	eTriggerMode = eRFTR_ImmediateRead
	eWriteMode = eRFWR_ImmediateWrite

eWriteMode = eRFWR_WriteToNextTag	eTriggerMode = eRFTR_ReadWithTrigger

bMultiTranspMode = TRUE	bSerialNumberMode = TRUE
	eWriteMode = eRFWR_ImmediateWrite
	eReadMode = eRFRD_SingleShot

If the configuration is transmitted as a register, these dependencies also exist and the RFID device will return an error code if inadmissible.

5.1.4 Leuze electronic

RFID reader settings

For smooth communication between controller and RFID readers, some settings need to be made before the system startup. These include, for example, the baud rate for the serial communication. A proprietary tool from the manufacturer of the RFID reader may be required in order to transfer these settings to the RFID.

For all supported RFID reader models, these standard data transmission settings have been tried and tested:

Setting	Value
Baud rate (RS232 and RS485)	9600 baud
Parity Bit	none
Data bits	8
Stop bit	1

Depending on the hardware, other parameters can also be set, or the factory settings of the RFID reader can be used. These must then also be adopted in the software reader connection (see RFID reader connection [▶ 19]).

Using the proprietary tools, the following special settings may have to be parameterized before the system is started:

Setting	Value
Data transmission parameters (see above)	Setting analogous to the values selected in the PLC program

If the RFID reader is triggered, the following response telegram is received by the function block of the TwinCAT RFID library. There is no assignment of read transponder data in this case. The received raw data can be taken from the function block interface for further processing.

RFID reader handling

The block size corresponding to the tag must be configured in the RFID reader configuration.

The Leuze devices support 4 bytes or 8 bytes block size.

Example: If a block size of 8 bytes is specified for the transponder, then the reader must be configured with the parameter iBlocksize := 8 and the read or write access via the structure ST_RFID_AccessData [▶ 50] must take place with an 8-byte block size.

A write command can write up to 36 bytes of data at a time. If more data is to be written to the transponder, it must be divided into several commands.

It must be pointed out here that not all peculiarities of every supported RFID reader model can be named here. Therefore you are referred to the manufacturer’s own documentation for more detailed information.

Configuration

Note the following when a new configuration is written to the RFID device (command SetConfig [▶ 23]):

Not every combination of configuration parameters (structure ST_RFID_CfgStruct_LeuzeRFM [▶ 59]) is allowed. Disregard of the required dependencies leads to an error (eRFERR_InvalidCfg):

Configuration parameters	Required dependencies

eReadMode = eRFRD_ContinuousRead	eTriggerMode = eRFTR_ImmediateRead
	eWriteMode = eRFWR_ImmediateWrite

eWriteMode = eRFWR_WriteToNextTag	eTriggerMode = eRFTR_ReadWithTrigger

bMultiTranspMode = TRUE	bSerialNumberMode = TRUE
	eWriteMode = eRFWR_ImmediateWrite
	eReadMode = eRFRD_SingleShot

If the configuration is transmitted as a register, these dependencies also exist and the RFID device will return an error code if inadmissible.

5.1.5 Pepperl-Fuchs

RFID reader settings

For smooth communication between controller and RFID readers, some settings need to be made before the system startup. These include, for example, the baud rate for the serial communication. A proprietary tool from the manufacturer of the RFID reader may be required in order to transfer these settings to the RFID.

For all supported RFID reader models, the following standard data transfer settings have been tried and tested:

Setting	Value
Baud rate (RS232 and RS485)	38400 baud
Parity Bit	none
Data bits	8
Stop bit	1

If required, other parameters can be set, depending on the hardware. These must then also be adopted in the software reader connection (see RFID reader connection [▸ 19]).

RFID reader handling

The model information (GetReaderVersion [▸ 22] command) and the current reader configuration (GetConfig [▸ 22] command) must be evaluated at the system start.

The received status of the device is displayed via output iErrCodeRcv of function block FB_RFIDReader and signaled in case of error by bError = TRUE and iErrorId = eRFERR_ErrorRcv. The read heads also have their own status. These can be checked with the configuration structure [▸ 61] read via GetConfig.

The set transponder types should be checked on restarting. If the configuration structure read via GetConfig does not display the correct transponder types for each read head, they can be corrected with the ChangeDCType [▸ 25] command. It is recommended to set the value specified for the transponder in place of the default value (99). The read/write head additionally recognizes the data storage device more quickly as a result.

When accessing the data memory of a transponder for writing and/or reading, a block size (see ST_RFID_AccessData [▸ 50]) suitable for the transponder must be used for all Pepperl+Fuchs RFID devices!

Block sizes of possible transponders:

4 bytes (IQC21, IPC03, IQC22, IQC24)
8 bytes (IQC33)
16 bytes (IQC40, IQC41, IQC42 and IQC43)
32 bytes (IQC37)

The use of a 4-byte block size only is supported up to version 3.3.3.0 of the library.

It must be pointed out here that not all peculiarities of every supported RFID reader model can be named here. Therefore you are referred to the manufacturer’s own documentation for more detailed information.

Buffered Command

Using the bBufferedCmd input variable in ST_RFID_Control [▸ 44], it is possible to transmit commands that can be buffered for continuous execution at a later time. This is possible with the eRFC_GetInventory, eRFC_ReadBlock and eRFC_WriteBlock commands. A buffered command can be ended with the eRFC_AbortCommand command.

Buffered command

If such a buffered command is active on a read head, the trigger mode may neither be active nor activated for this channel! Similarly, no raw data command may be transmitted that concerns this channel!

Trigger mode

It is recommended not to use any trigger or sensor channel. The trigger mode should also be deactivated for all channels. In the factory setting for the RFID device the trigger is deactivated for all channels.

Alternatively, for example, the GetInventory command can be called cyclically or GetInventory can be called as a buffered command (bBufferedCmd in ST_RFID_Control [▸ 44]).

The TwinCAT library offers the following option for using a sensor channel as trigger for the RFID unit:

The trigger sends a message to indicate whether it is triggered or the value is outside the trigger range. These messages are received and displayed as eResponse = eRFR_CmdConfirmation or eRFR_NoTransponder. The application can respond accordingly and trigger the required command.

To configure a channel as sensor channel/trigger, the corresponding Ident channel must be 0. The required raw data command is explained in the next paragraph.

Trigger setting

The trigger setting must not be accomplished from the manufacturer's proprietary tool. Otherwise the messages of the sensor channel received as a result cannot be read by the function block of the TwinCAT RFID library.

The correct setting of the trigger mode should be checked with the command GetConfig [▸ 22] and by evaluating the read configuration structure. If necessary the setting can be made up for on program startup.

Sending the settings via raw data commands

For details see Low-level communication [▸ 40] and the description of structures ST_RFID_Control [▸ 44] and ST_RFID_RawData [▸ 43].

Baud rate:

To set the baud rate of the RFID device to 9600 baud send the following raw data:

ASCII	hex
CI0,9600	43 49 30 2C 39 36 30 30

After changing the baud rate, a reset of the RFID device is necessary.

Trigger mode:

To deactivate a trigger sensor on channel 3 so that the channel can be used as read head, the following raw data should be sent:

ASCII	hex
TM300	54 4D 33 30 30

To configure a sensor as trigger on channel 2, the following raw data should be sent:

ASCII	hex
TM201	54 4D 32 30 31

Answer: eResponse = eRFR_CmdConfirmation when the sensor is triggered.

Answer: eResponse = eRFR_NoTransponder when the sensor is exited.

The output stTranspInfo.iHeadNumber indicates the sensor channel from which the response was sent.

To configure a sensor as inverted trigger on channel 4, the following raw data should be sent:

ASCII	hex
TM402	54 4D 34 30 32

Answer: eResponse = eRFR_NoTransponder when the sensor is triggered.

Answer: eResponse = eRFR_CmdConfirmation when the sensor is exited.

The output stTranspInfo.iHeadNumber indicates the sensor channel from which the response was sent.

Once such setting has been made, the device configuration must be re-read with the command Get Config [▸ 22]. It is advisable to check the settings by analyzing the read configuration structure.

6 PLC API

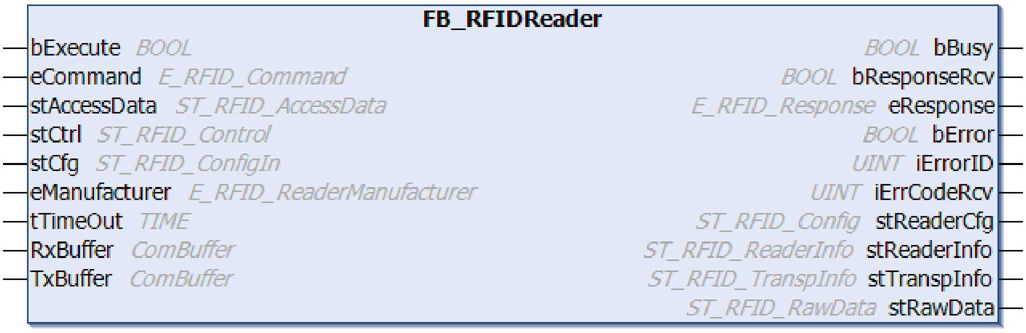
6.1 Function block

6.1.1 FB_RFIDReader

The TwinCAT RFID library only consists of one function block.

This section explains the interface variables of the function block for a quick introduction to the handling of the library. See also the Tutorial [▶ 67] and the Examples [▶ 67].

The uniform handling of all RFID reader models and the associated prepared interface declarations are particularly user-friendly. However, it should be noted that the function block of the TwinCAT RFID library has a slight overheat due to the differences of some RFID reader models. This indispensable characteristic is, however, strongly outweighed by the advantages that the available flexibility offers.



Syntax

```
FUNCTION_BLOCK FB_RFIDREADER
VAR_INPUT
    bExecute      : BOOL;
    eCommand      : E_RFID_Command;
    stAccessData  : ST_RFID_AccessData;
    stCtrl        : ST_RFID_Control;
    stCfg         : ST_RFID_ConfigIn;
    eManufacturer : E_RFID_ReaderManufacturer;
    tTimeOut      : TIME  := T#5s;
END_VAR
VAR_IN_OUT
    RxBuffer      : ComBuffer;
    TxBuffer      : ComBuffer;
END_VAR
VAR_OUTPUT
    bBusy         : BOOL;
    bResponseRcv  : BOOL;
    eResponse     : E_RFID_Response;

    bError        : BOOL;
    iErrorID      : UINT;     (* general RFID error *)
    iErrCodeRcv   : UINT;     (* error received by reader *)

    stReaderCfg   : ST_RFID_Config;
    stReaderInfo  : ST_RFID_ReaderInfo;
    stTranspInfo  : ST_RFID_TranspInfo;
    stRawData     : ST_RFID_RawData;
END_VAR
```

Inputs

Name	Type	Description
bExecute	BOOL	In order to receive messages from the RFID reader, the function block is called with a FALSE on this input. The function block reacts to a rising edge on bExecute by executing the selected command eCommand or querying it at the RFID reader.
eCommand	E_RFID_Command [▸ 62]	The eCommand input offers a choice of commands, such as reading from or writing to a transponder, in the form of an enumeration. A command is executed by setting the bExecute input.
stAccessData	ST_RFID_AccessData [▸ 50]	If a write or read command is to be executed, then parameters must be transferred with this input structure.
stCtrl	ST_RFID_Control [▸ 44]	Various control parameters can be transferred at the input with stCtrl. This includes the possibility to specify delay times.
stCfg	ST_RFID_ConfigIn [▸ 48]	An RFID reader has an internal configuration. This can be read out and changed on some devices. Configuration parameters to be transmitted to the RFID reader are transferred at the stCfg input. See also: Configuration [▸ 39]
eManufacturer	E_RFID_ReaderManufacturer [▸ 65]	The manufacturer of the RFID reader model in use is specified at this input.
tTimeOut	TIME	Specifies a maximum length of time for the execution of the function block. The default value is 5 seconds.

The condition tTimeOut > tPreSendDelay + tPostSendDelay applies. Otherwise an error is generated at the output. See details of the delay times in ST_RFID_Control [▸ 44].

/ Inputs/outputs

Name	Type	Description
RxBuffer	ComBuffer	Specifies the receive buffer that was declared as an input variable and linked to the serial terminal in the TwinCAT System Manager. See the description of the serial connection of an RFID reader in section RFID reader connection [▸ 19].
TxBuffer	ComBuffer	Specifies the transmit buffer that was declared as an output variable and linked to the serial terminal in the TwinCAT System Manager. See the description of the serial connection of an RFID reader in section RFID reader connection [▸ 19].

Outputs

Name	Type	Description
bBusy	BOOL	If the command call is valid, the bBusy output goes to TRUE for at least one cycle. The function block may only be called again for a new command with bExecute = TRUE if bBusy has changed to FALSE and the function block is thus no longer in the active transmission state. Thus, if bBusy = FALSE is detected, all other output variables bResponseRcv, eResponse, bError, iErrCodeRcv,... and stRawData can be evaluated. If a response to an executed command call is expected, then the function block remains at bBusy = TRUE until a telegram is received or until the timeout tTimeOut is reached. If the delay times tPreSendDelay and/or tPostSendDelay have been given to the function block, bBusy remains TRUE for at least as long as the sum of these times. See also: ST_RFID_Control [▸ 44].
bResponseRcv	BOOL	As soon as a response from the RFID reader has arrived at the controller, this flag is set for at least one cycle. The arrival of a telegram is generally signaled with a rising edge to bResponseRcv = TRUE. Thus, if this flag is detected, unexpected telegrams and the corresponding output variables eResponse, bErr, iErrCodeRcv,... and stRawData can be evaluated. If a response is expected for an executed command call, the function block remains bBusy = TRUE until a telegram is received. Depending on the command call, more than one response can arrive before the action is completed and bBusy = FALSE. Depending on the configuration setting of the delay times in ST_RFID_Control [▸ 44], bResponseRcv can go to TRUE even before bBusy changes to FALSE.
eResponse	E_RFID_Response [▸ 63]	As soon as bResponseRcv shows TRUE, this enumeration indicates the type of message received. The appropriate evaluation can follow, for example, depending on the type.
bError	BOOL	The bError output becomes TRUE as soon as an error occurs. This can be due to incorrect input parameters, transmission errors, errors on the part of the RFID reader or a timeout. The type of error that has occurred is indicated by the output variable iErrorID. Details about the error display can be found in the Error codes [▸ 75] section.
iErrorID	UNIT	If an error occurs, the type of error at output iErrorID is displayed. Details of the possible error IDs are given in the Error codes [▸ 75] section.
iErrCodeRcv	UINT	The error code indicated at the iErrCodeRcv output corresponds to the error code sent by the RFID reader to the controller. Details about the error display can be found in the Error codes [▸ 75] section.
stReaderCfg	ST_RFID_Config [▸ 49]	An RFID reader has an internal configuration. This can be read out and changed on some devices. These read-out configuration parameters are made available at the stReaderCfg output.
stReaderInfo	ST_RFID_ReaderInfo [▸ 42]	Each RFID reader has its own identification data such as designation, hardware version, etc. These values, which can be queried with the command GetReaderVersion [▸ 22] among others, are specified in the output structure stReaderInfo.

Name	Type	Description
stTranspInfo	ST_RFID_TranspInfo [▶ 41]	The structure stTranspInfo contains information on the last read transponder. Among other things, the serial number of the transponder is output here.
stRawData	ST_RFID_RawData [▶ 43]	The output structure stRawData outputs both the sent and the received raw data.

Requirements

Development Environment	Target Platform	PLC Libraries to include
TC3.1.4013	PC or CX (x86, x64)	Tc2_RFID

6.1.2 Handling instructions

RFID library handling

If you have included the library file Tc2_RFID, you have access to all functions. The library places a function block at your disposal for communication with an RFID reader.

You can use the general function block FB_RFIDReader, which is usable for all RFID reader models, or one of the manufacturer-specific function blocks. These offer the same range of functions and almost the same interface and the same handling, and they are also optimized in terms of the code and the performance.

The function block made available for RFID reader communication offers high level communication with a high level interface. A instruction set provides various commands (see RFID command set [▶ 21]). In addition, the integrated low-level communication allows sending and receiving of raw data (see Low level communication [▶ 40]).

The Tc2_RFID library expects the RFID reader to respond immediately to a command and that the dialog is not interrupted by another telegram. Otherwise an evaluation may not be possible.

General handling of the function block

Depending on the RFID reader model, the device can send a telegram to the controller without a prior request. A cyclic call of the RFID function block with bExecute = FALSE is sufficient for reception.

All possible active accesses to the RFID device are listed in the instruction set (see RFID command set [▶ 21]). The following procedure is common to all commands:

The function block is called by a rising edge on the bExecute input. Afterwards, cyclic calling of the function block (bExecute = FALSE) returns the result of the query at the output as soon as the processing of the query has been completed (bBusy = FALSE). The function block must be called (bExecute = FALSE) for as long as it takes for the internal processing (bBusy = FALSE) to be completed. During that time, all inputs of the function block must remain unchanged.

All received messages are additionally made available completely as raw data in unprocessed form at the output.

Further handling instructions can be found in the description of the input and output variables of the function block and in the tutorial and examples in this documentation.

Initialization of an RFID reader integrated via the TwinCAT library

When the system is started, the following actions are required for initializing an RFID reader integrated via the TwinCAT library:

Insofar as they are available according to the instruction set, the model information (GetReaderVersion [▶ 22] command) and the current reader configuration (GetConfig [▶ 22] command) must be evaluated. Because successful communication with the RFID reader is dependent on these data, it must be ensured that the current values are always available and queried if necessary.

RFID reader handling

The characteristics of the supported RFID reader models are listed in section RFID reader settings and handling [▶ 26]. The notes listed there are allocated to the special RFID reader manufacturers.

6.1.3 Configuration

All supported RFID readers can be configured with the same command. This must be available according to the instruction set for the specific model (see RFID command set [▶ 21]).

In addition to the reader version, the current configuration of the reader should also be requested at each program start.

Since RFID readers from different manufacturers never have identical configuration options, in addition to the input configuration structure the TwinCAT RFID library offers a substructure for each manufacturer with the specific parameters (ST_RFID_CfgStruct_DeisterUDL, ST_RFID_CfgStruct_LeuzeRFM,...). The parameters listed there can be parameterized by the user as desired within the limits of the valid ranges of values. The meaning of the parameters is to be taken either from the structure declaration or the proprietary specifications.

Reading the configuration

The GetConfig [▶ 22] command from the instruction set is used in order to read the current RFID reader configuration. After that, the configuration data can be taken from the output of the function block if the query was successful. They are available there in the structure ST_RFID_Config [▶ 49] both as a configuration structure and as a configuration register.

Changing the configuration

The SetConfig [▶ 23] command from the instruction set is used in order to write an RFID reader configuration. After a SetConfig command, the current configuration must be read once with the GetConfig [▶ 22] command.

If the user sets further reaching special configuration parameters via an external tool and wants to retain these, then the flag for "Default Values" bUseCfgDefault in the structure ST_RFID_ConfigIn [▶ 48] should be deactivated.

Invalid combination of configuration parameters

Certain combinations of configuration parameters are sometimes impermissible. You can find out which parameter values are excluded in which combination from the proprietary protocol specifications of the RFID reader manufacturers.

If the parameters are entered incorrectly, then either an error will be generated even before the configuration query, or the RFID reader signals by its response that the configuration data could not be adopted.

Configuration data

Each configuration can be seen as a register (byte array) or as a structure. This is not the parameterization of the TwinCAT RFID library but the proprietary configuration of the RFID reader. The TwinCAT RFID library contains various configuration structures, which process the raw data of the configuration registers of different RFID readers. If available, both variants are provided in ST_RFID_Config [▶ 49] at the output of the library function block. This takes place via pointers.

Baltech

The configuration data is used as a structure for Baltech RFID readers.

- ST_RFID_CfgStruct_BaltechMifVHLFile [▶ 51]

The structure is suitable for writing with the eRFC_SetConfig command (see RFID command set [▶ 21]).

Balluff

No configuration options are offered.

Deister

The configuration data can be used for Deister RFID readers both as a structure and as a register.

If a register array (byte array) is used, it must always have the size of the complete configuration data. For the supported Deister RDL devices this is 88 bytes, for the UDL devices 117 bytes.

- ST_RFID_CfgStruct_DeisterRDL [▸ 54]
- ST_RFID_CfgStruct_DeisterUDL [▸ 56]

The structures are suitable for writing with eRFC_SetConfig and reading with eRFC_GetConfig (see RFID command set [▸ 21]).

Leuze

For Leuze RFID readers the configuration data can be used as a structure and as a register.

If a register array (byte array) is used, it must always have the size of the complete configuration data. For supported Leuze devices this is 88 bytes.

- ST_RFID_CfgStruct_LeuzeRFM [▸ 59]

The structure is suitable for writing with eRFC_SetConfig and reading with eRFC_GetConfig (see RFID command set [▸ 21]).

Pepperl+Fuchs

For Pepperl+Fuchs RFID readers the configuration data are used as a structure.

- ST_RFID_CfgStruct_PepperlFuchsIDENT [▸ 61]

The structure is suitable for reading with eRFC_GetConfig (see RFID command set [▸ 21]).

6.1.4 Low level communication

In addition to the high-level instruction set, the TwinCAT RFID library also offers the possibility of low-level communication. This is solved implicitly. The same function block is used.
Any telegrams up to a maximum size of 1024 bytes can be received and sent up to a size of 300 bytes.

A complete telegram is composed as follows:

| Prefix | Addressing | **Raw data** | CRC | Suffix |

Depending upon the proprietary protocol specification, individual components may be missing. In general, however, the composition is the same.

Sending

The eRFC_SendRawData command from the instruction set is used for sending (see RFID command set [▸ 21]). The raw data to be sent is specified in the input structure ST_RFID_Control [▸ 44].

In order to send a low level telegram, only the raw data is specified. The other telegram components are automatically supplemented by the TwinCAT RFID library. Checking data such as CRC are likewise generated and added internally.

If the protocol requires recoding of certain bytes within the raw data, this is also done automatically by the TwinCAT RFID library.

If an RS485 interface is involved, the addressing must be specified separately. It may not be contained in the specified raw data. By default, the addressing is performed automatically by the library. However, it can be parameterized via the input variables in ST_RFID_Control [▸ 44].

The raw data last sent can be viewed at any time at the output of the function block by means of the structure ST_RFID_RawData [▸ 43]. This is independent of the command used.

Receiving

The raw data last received can be viewed at any time at the output of the function block by means of the structure ST_RFID_RawData [▶ 43]. The associated addressing is output in the structure ST_RFID_ReaderInfo [▶ 42].

When using the command SendRawData [▶ 25], a direct evaluation of the received response cannot be guaranteed.

Example: If a read command is sent manually as a byte sequence by means of the SendRawData command, then received transponder data are not written at an address specified in ST_RFID_AccessData [▶ 50]. Evaluation/storage of the data should therefore also be done manually with the help of the raw data structure ST_RFID_RawData [▶ 43], which is always specified. The specified raw data consists of the received telegram, but without prefix, suffix, checksum, CRC or shift sequence coding. If the received telegram has not been regularly evaluated by the function block, this is additionally indicated by an error.

In order to be able to use the received data, these must be copied, for example, to a byte array.

Example of the assignment of received data with the aid of the MEMCPY() function:

```
fbRFIDReader     : FB_RFIDReader;
arrReceivedData : ARRAY [0..511] OF BYTE;
```

```
MEMSET( ADR(arrReceivedData), 0, SIZEOF(arrReceivedData) );
MEMCPY( ADR(arrReceivedData),
        fbRFIDReader.stRawData.pReceivedRsp,
        MIN(fbRFIDReader.stRawData.iReceivedRspLen, SIZEOF(arrReceivedData)) );
```

Balluff RFID Reader: The end identifier (LF CR) is used as a suffix for the recognition of telegrams. Previous data are combined into a telegram as soon as these 2 bytes are detected in the data stream.

6.2 Data types

6.2.1 Structures

6.2.1.1 ST_RFID_TranspInfo

The structure ST_RFID_TranspInfo indicates the type and the serial number of the last detected RFID transponder. The command GetInventory [▶ 23] is used to query and update this information.

```
TYPE ST_RFID_TranspInfo :
STRUCT
    sSerialNumber : T_RFID_TranspSRN; (* serial number shown as hex coded string(ascii) *)
    eType         : E_RFID_TranspType;
    iHeadNumber   : USINT;   (* read head where the last transponder was detected *)
    iDCType       : USINT;
(* data carrier type: the received transponder type code (see device specific type list) *)
END_STRUCT
END_TYPE
```

All manufacturers

Name	Description
sSerialNumber	The serial number of the transponder (frequently 8 bytes) is indicated in the hexadecimal string sSerialNumber. The data type is T_RFID_TranspSRN [▶ 66]. If Balluff RFID readers are used, the serial number of 13.56 Mhz transponders is rotated byte-by-byte by the library function block. This takes place because the serial number read out from a transponder would otherwise not correspond to the serial number read out on another reader. This allows devices from different manufacturers to be operated together in the same network.
eType	The type of transponder is indicated as an enumeration value of the enumeration E_RFID_TranspType [▶ 65].
iDCType	The type of transponder is specified as a numeric code. See the proprietary device-specific transponder type list.

Pepperl+Fuchs

Name	Description
iHeadNumber	If an RFID reader with several read heads is connected, iHeadNumber indicates by which head the RFID transponder was detected.

Requirements

Development Environment	Target Platform	PLC Libraries to include
TC3.1.4013	PC or CX (x86, x64)	Tc2_RFID

6.2.1.2 ST_RFID_ReaderInfo

Following the GetReaderVersion [▶ 22] command from the instruction set, the received data are processed in this output structure.

Not every variable is thereby served by every RFID reader model in the form of the version information. For example, one reader model indicates the production date while another RFID reader model transfers the hardware version. More detailed information on the entries can be found in the manufacturer's own protocol specification and manuals.

```
TYPE ST_RFID_ReaderInfo :
STRUCT
    dDate         : DATE;
    eType         : E_RFID_ReaderType;
    eGroup        : E_RFID_ReaderGroup;
    eManufacturer : E_RFID_ReaderManufacturer;
    iReserved     : UINT;
    sSWVersion    : STRING(31);
    sHWVersion    : STRING(31);
    sCode         : STRING(39);
    sSerialNumber : STRING(39);

    iSrcAddrRcv   : UDINT;            (* RS485 address *)
    iDstAddrRcv   : UDINT;            (* RS485 address *)
END_STRUCT
END_TYPE
```

All manufacturers

Name	Description
eType	The RFID reader type is indicated as an enumeration at this output.
eGroup	The RFID reader group/series is indicated at this output. The internal processing of all telegrams in the library is specified by this group allocation.

eManufacturer	The manufacturer of the connected RFID reader is indicated at this output.
	The enumeration provides the following choices:
	`TYPE E_RFID_ReaderManufacturer : (`
	`     eRFRM_Unknown,`
	`     eRFRM_Balluff,`
	`     eRFRM_Deister,`
	`     eRFRM_Leuze,`
	`     eRFRM_PepperlFuchs,`
	`     eRFRM_Baltech );`
	`END_TYPE`

iSrcAddrRcv	In case of the RS485 interface, the received source address is indicated here.
iDstAddrRcv	In case of the RS485 interface, the received destination address is indicated here.

Baltech

Name	Description
dDate	The production date of the RFID reader is indicated at this output. The date 01/01/1970 means that no production date was transmitted.
sSWVersion	Indicates the software version as text.
sCode	The special type of the RFID reader is transmitted as numeric code. This is output at output sCode as a string.
sSerialNumber	The serial number of the RFID reader is output as a hexadecimal string at the sSerialNumber output. This is not to be confused with the transponder serial number.

Deister

Name	Description
sSWVersion	Indicates the software version as text.
sHWVersion	Indicates the hardware version as text.
sCode	The special type of the RFID reader is transmitted as numeric code. This is output at output sCode as a string.
sSerialNumber	The serial number of the RFID reader is output as a hexadecimal string at the sSerialNumber output. This is not to be confused with the transponder serial number.

Leuze

Name	Description
dDate	The production date of the RFID reader is indicated at this output. The date 01/01/1970 means that no production date was transmitted.
sCode	The special type of the RFID reader is transmitted as numeric code. This is output at output sCode as a string.

Pepperl+Fuchs:

Name	Description
dDate	The production date of the RFID reader is indicated at this output. The date 01/01/1970 means that no production date was transmitted.
sSWVersion	Indicates the software version as text.
sCode	The special type of the RFID reader is transmitted as numeric code. This is output at output sCode as a string.

Requirements

Development Environment	Target Platform	PLC Libraries to include
TC3.1.4013	PC or CX (x86, x64)	Tc2_RFID

6.2.1.3 ST_RFID_RawData

This structure outputs the sent and received data as raw data. This is always the complete telegram, but without prefix, suffix, checksum, CRC or shift sequence coding. Low-level communication is described in detail in section Low-level communication [▶ 40].

The byte sequences can be viewed via the specified pointers. The MEMCPY() function can be used for evaluation.

```
TYPE ST_RFID_RawData :
STRUCT
    pReceivedRsp :POINTER TO BYTE;
    pSentCommand :POINTER TO BYTE;
    iReceivedRspLen :UINT;
    iSentCommandLen :UINT;
END_STRUCT
END_TYPE
```

Name	Description
pReceivedRsp	A received telegram is stored as a byte sequence and the pReceivedRsp pointer points to this byte sequence.
pSentCommand	A sent telegram is stored as a byte sequence and the pSentCommand pointer points to this byte sequence.
iReceivedRspLen	Specifies the length of the stored byte sequence in bytes.
iSentCommandLen	Specifies the length of the stored byte sequence in bytes.

Requirements

Development Environment	Target Platform	PLC Libraries to include
TC3.1.4013	PC or CX (x86, x64)	Tc2_RFID

6.2.1.4 ST_RFID_Control

The input structure stCtrl contains variables with which the behavior of the function block can be parameterized.

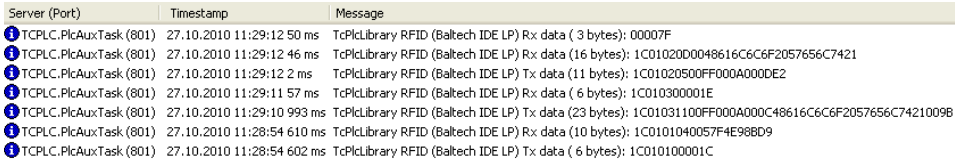
The two variables tPreSendDelay and tPostSendDelay offer the option to parameterize delay times.

```
TYPE ST_RFID_Control :
STRUCT
    sSelTranspSRN      :T_RFID_TranspSRN; (* serial number shown as hex coded string(ascii) *)

    tPreSendDelay      :TIME;          (* condition: tTimeOut > tPreSendDelay + tPostSendDelay *)
    tPostSendDelay     :TIME;          (* condition: tTimeOut > tPreSendDelay + tPostSendDelay *)

    iSrcAddrSnd        :UDINT;         (* RS485 address *)
    iDstAddrSnd        :UDINT;         (* RS485 address *)
    bAddrAutoMode      :BOOL := TRUE;  (* if AutoMode is activated the communication addresses are ha
ndled automatically and set addresses are not used. *)

    bLogging           :BOOL;

    iHeadNumber        :USINT := 1;    (* if multiple read heads are installed at the reader unit, on
e can be selected *)
    iDCType            :USINT := 1;

    pRawDataCommand    :POINTER TO BYTE; (* data input for low level communication *)
    iRawDataCommandLen :UINT;

    bBufferedCmd       :BOOL;

    iVHLFileID         :USINT := 16#FF;  (* selection of VHL file; default 0xFF (ad hoc VHL file of
vhlSetup) *)

    iCardTypeMask      :UINT := 16#FFFF; (* select which card type should be detected via GetInvento
ry; default 0xFFFF (all types) *)
    bReselect          :BOOL := TRUE;    (* with Reselect every GetInventory gets the first item of
the reader's detected card list *)
    bAcceptConfCard    :BOOL := TRUE;    (* a read command also accept configuration cards to config
ure the RFID Reader *)
END_STRUCT
END_TYPE
```

Multiple RFID readers on the same RS485 network

The TwinCAT RFID library does not support multiple RFID readers on the same RS485 network for the time being. A separate connection must be made to a separate terminal for each RFID reader. In this stand-alone mode, individual addressing with iSrcAddrSnd and iDstAddrSnd is not necessary. Addressing can therefore be handled automatically by the TwinCAT RFID library, for which the input variable bAddrAutoMode can be left as TRUE.

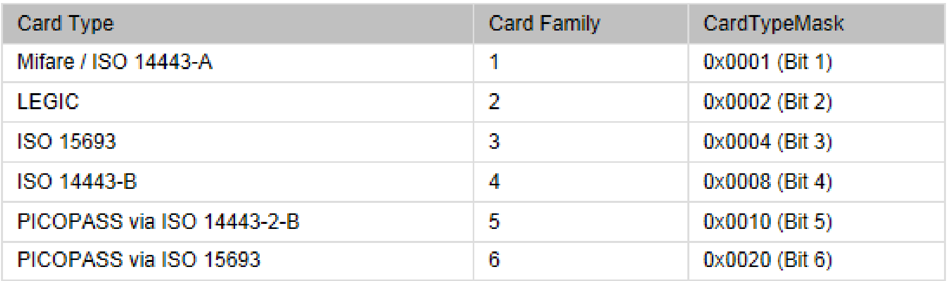
All manufacturers

Name	Description
tPreSendDelay	Before sending a command to the RFID reader, the function block waits until the time specified by tPreSendDelay has elapsed.
tPostSendDelay	After sending a command to the RFID reader, the function block waits until at least the time specified by tPostSendDelay has elapsed. If the expected response has not arrived yet, the function block continues to wait for the response until the specified timeout time tTimeOut has elapsed at the latest.
iSrcAddrSnd	Source address in the case of RS485 communication. This address is used if bAddrAutoMode is not set.
iDstAddrSnd	Destination address in the case of the RS485 communication. This address is used if bAddrAutoMode is not set.
bAddrAutoMode	Option in the case of RS485 communication. If bAddrAutoMode is enabled [default: TRUE], the RS485 addresses are assigned automatically. The addresses given above are not used. If bAddrAutoMode is disabled (FALSE), the addresses specified above are used.
bLogging	An additional output can be activated with bLogging. The serial communication can thus be reconstructed with the aid of Log View Messages. These messages can be viewed in the Windows Event Viewer as well as in the TwinCAT System Manager. This is useful for testing and analysis purposes. The format of the output is not specified, in order to facilitate extensions.  Server (Port) Timestamp Message TCPLC.PlcAuxTask (801) 27.10.2010 11:29:12 50 ms TcPlcLibrary RFID (Baltech IDE LP) Rx data (3 bytes): 00007F TCPLC.PlcAuxTask (801) 27.10.2010 11:29:12 46 ms TcPlcLibrary RFID (Baltech IDE LP) Rx data (16 bytes): 1C01020D0048616C6C6F2057656C7421 TCPLC.PlcAuxTask (801) 27.10.2010 11:29:12 2 ms TcPlcLibrary RFID (Baltech IDE LP) Tx data (11 bytes): 1C01020500FF000A000DE2 TCPLC.PlcAuxTask (801) 27.10.2010 11:29:11 57 ms TcPlcLibrary RFID (Baltech IDE LP) Rx data (6 bytes): 1C010300001E TCPLC.PlcAuxTask (801) 27.10.2010 11:29:10 993 ms TcPlcLibrary RFID (Baltech IDE LP) Tx data (23 bytes): 1C01031100FF000A000C48616C6C6F2057656C7421009B TCPLC.PlcAuxTask (801) 27.10.2010 11:28:54 610 ms TcPlcLibrary RFID (Baltech IDE LP) Rx data (10 bytes): 1C0101040057F4E98BD9 TCPLC.PlcAuxTask (801) 27.10.2010 11:28:54 602 ms TcPlcLibrary RFID (Baltech IDE LP) Tx data (6 bytes): 1C010100001C The first time the function block is called with bLogging = TRUE, the output "Logging initialized" is generated. No communication data is monitored in this first cycle.
pRawDataCommand	The regular function block of the TwinCAT RFID library can also be used for low-level communication. In this case, raw data can be sent directly to the RFID reader. The raw data to be sent (e.g. as a byte array) must be specified in this input variable. This is a pointer to the raw data to be sent. The library function block accesses this address in read-only mode. Further information on the process of low-level communication is summarized in section Low-level communication [▶ 40]. Both the transmitted and the received raw data can be viewed at any time at the output by means of the structure ST_RFID_RawData [▶ 43].
iRawDataCommandLen	The input variable iRawDataCommandLen specifies the length in bytes of the raw data specified by the pointer pRawDataCommand.

Balluff

Name	Description
iHeadNumber	If an RFID reader with several read heads is addressed, the read head is specified in the input variable iHeadNumber.
iDCType	If Balluff readers are used via iDCType, the block size of the chip memory (0→64bytes, 1→32bytes) can be specified.

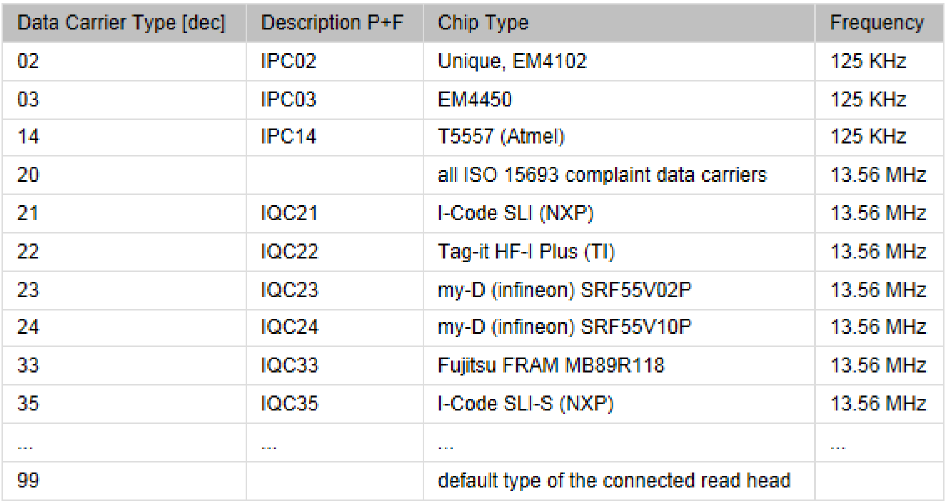
Baltech

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>iVHLFileID</td><td>Write and read commands are executed with the VHL file selected here. This must be stored in the configuration of the RFID device. If iVHLFileID has the value 0xFF [default], the RFID reader will not take a normal VHL file from its configuration, but a simple ad-hoc file, which can be selected so that non-configured/unencrypted blank cards can be used.</td></tr>
<tr><td>iCardTypeMask</td><td>The iCardTypeMask can be used to set specific transponder card families. Only the set card types are recognized with the command GetInventory. All other card types are filtered out. If all card types are to be recognized, this value does not have to be changed [default: 0xFFFF]. For each accepted card family, the respective bit is set in the iCardTypeMask.

<table>
<tr><th>Card Type</th><th>Card Family</th><th>CardTypeMask</th></tr>
<tr><td>Mifare / ISO 14443-A</td><td>1</td><td>0x0001 (Bit 1)</td></tr>
<tr><td>LEGIC</td><td>2</td><td>0x0002 (Bit 2)</td></tr>
<tr><td>ISO 15693</td><td>3</td><td>0x0004 (Bit 3)</td></tr>
<tr><td>ISO 14443-B</td><td>4</td><td>0x0008 (Bit 4)</td></tr>
<tr><td>PICOPASS via ISO 14443-2-B</td><td>5</td><td>0x0010 (Bit 5)</td></tr>
<tr><td>PICOPASS via ISO 15693</td><td>6</td><td>0x0020 (Bit 6)</td></tr>
</table></td></tr>
<tr><td>bReselect</td><td>If bReselect is set [default: TRUE], a transponder is specified at the output for each query with the GetInventory [▸ 23] command, provided a transponder is in the HF field. If it is ensured that there is only one card before the RFID reader at a time, this setting should be retained. If there is more than one transponder card in the HF field, the RFID device contains a list of detected cards. To be able to select all transponders with the command GetInventory, bReselect should be switched off [FALSE]. The command GetInventory then goes through this list through multiple sequential calls. After the last card has been selected, the feedback eRFR_NoTransponder follows. The GetInventory command does not return the transponder and its serial number to the output again until a card is removed from the HF field and inserted again.</td></tr>
<tr><td>bAcceptConfCard</td><td>If bAcceptConfCard is set [default: TRUE], configuration cards are also recognized with the command GetInventory. An attempt is made automatically to transfer the configuration on these cards to the RFID reader. This parameter can be disabled to prevent influencing by other configuration cards.</td></tr>
</table>

Leuze

Name	Description
sSelTranspSRN	The serial number of the transponder to which the command (e.g. read/write) is to be applied can be specified as a string on the input variable sSelTranspSRN. The data type is T_RFID_TranspSRN [▸ 66]. This is not necessary in most cases. If a certain reader configuration makes this necessary, details for this are to be found in the relevant description in the proprietary specification.

Pepperl+Fuchs

<table>
<tr><th>Name</th><th>Description</th></tr>
<tr><td>iHeadNumber</td><td>If an RFID reader with several read heads is addressed, the read head is specified in the input variable iHeadNumber. iHeadNumber is also referred to as IdentChannel.</td></tr>
<tr><td>iDCType</td><td>With this variable and the command ChangeDCType [▸ 25] the transponder type can be set on the read head of Pepperl+Fuchs readers. Possible values for this are listed in the proprietary protocol under the ChangeTag command. (They do not match the values of the enumeration E_RFID_TranspType.)
Extract of supported transponder types:

<table>
<tr><th>Data Carrier Type [dec]</th><th>Description P+F</th><th>Chip Type</th><th>Frequency</th></tr>
<tr><td>02</td><td>IPC02</td><td>Unique, EM4102</td><td>125 KHz</td></tr>
<tr><td>03</td><td>IPC03</td><td>EM4450</td><td>125 KHz</td></tr>
<tr><td>14</td><td>IPC14</td><td>T5557 (Atmel)</td><td>125 KHz</td></tr>
<tr><td>20</td><td></td><td>all ISO 15693 complaint data carriers</td><td>13.56 MHz</td></tr>
<tr><td>21</td><td>IQC21</td><td>I-Code SLI (NXP)</td><td>13.56 MHz</td></tr>
<tr><td>22</td><td>IQC22</td><td>Tag-it HF-I Plus (TI)</td><td>13.56 MHz</td></tr>
<tr><td>23</td><td>IQC23</td><td>my-D (infineon) SRF55V02P</td><td>13.56 MHz</td></tr>
<tr><td>24</td><td>IQC24</td><td>my-D (infineon) SRF55V10P</td><td>13.56 MHz</td></tr>
<tr><td>33</td><td>IQC33</td><td>Fujitsu FRAM MB89R118</td><td>13.56 MHz</td></tr>
<tr><td>35</td><td>IQC35</td><td>I-Code SLI-S (NXP)</td><td>13.56 MHz</td></tr>
<tr><td>...</td><td>...</td><td>...</td><td>...</td></tr>
<tr><td>99</td><td></td><td>default type of the connected read head</td><td></td></tr>
</table></td></tr>
<tr><td>bBufferedCmd</td><td>Most commands are processed once immediately after the call. Depending on the RFID reader, commands can be present continuously. This enables, among other things, a read process as soon as the RFID transponder enters the reading field, without sending an extra command. This can either be configured in the RFID reader configuration (Balluff, Deister, Leuze) or selected with this input variable (Pepperl+Fuchs).
If such a buffered command is active on a read head, the trigger mode may neither be active nor activated for this channel! Similarly, no raw data command may be transmitted that concerns this channel!</td></tr>
</table>

Delay times

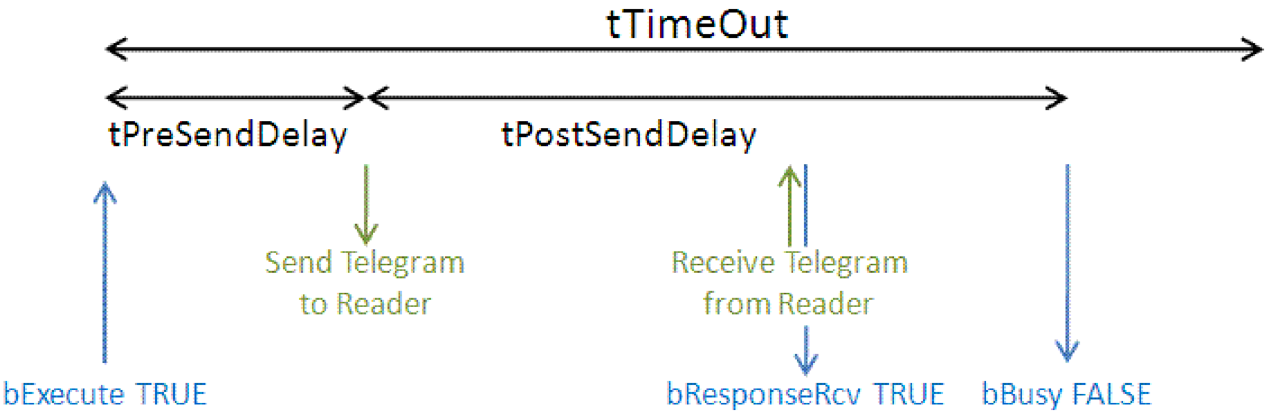
The two variables tPreSendDelay and tPostSendDelay offer the option to parameterize delay times. Both variables ensure that a delay is awaited between two requests to the RFID reader.

If the delay time is specified as tPreSendDelay, a delay between the last response telegram and the next request telegram is ensured. If the request telegram is to be sent as directly as possible, tPostSendDelay can be used.

The condition tTimeOut > tPreSendDelay + tPostSendDelay applies. Otherwise an error is generated at the output.

A minimum delay time of 300 ms between two commands is specified in the proprietary protocol of the Balluff RFID reader.

A minimum delay time of 150 ms between reception and command is specified in the proprietary protocol of the Leuze electronic RFID reader.



Requirements

Development Environment	Target Platform	PLC Libraries to include
TC3.1.4013	PC or CX (x86, x64)	Tc2_RFID

6.2.1.5 ST_RFID_ConfigIn

At the input of the RFID function block, this structure provides the possibility to transfer an arbitrary configuration to the RFID reader.

The RFID reader configuration last read is specified at the output with the structure ST_RFID_Config [▶ 49]. Supplementary information about configurations can be found in its description.

Configuration data can be available in the form of a specific configuration structure (ST_RFID_CfgStruct_DeisterUDL, ST_RFID_CfgStruct_LeuzeRFM,…) or also in the form of a configuration register (byte array). This selection can be made with the variable bUseCfgReg.

```
(* defines the configuration input parameters.
The data can be set via Config structure or Config register.
Different RFID Reader in different ReaderGroups can differ in their configuration data. *)
TYPE ST_RFID_ConfigIn :
STRUCT
    pCfg            : POINTER TO BYTE;  (* pointer to config structure or register *)
    iCfgSize        : UINT := 0;        (* size in bytes of the structure or register *)

    bUseCfgReg      : BOOL := FALSE;    (* Set Config via Register instead of CfgStructure *)
    bUseCfgDefault  : BOOL := TRUE;     (* Set Config using default parameters beside CfgStructure *)

    (* An additional option to demand/
set a specific config parameter without transmission of the whole config register.
    Not possible at all reader models.
    Set a desired value before calling GetConfig/
SetConfig or keep the default for full register request. *)
    iRegIdx    : UINT := 0;
    iRegGroup : USINT := 0;      (* 0:full register; 1:reg.00-0F; 2:single register *)

    bReserved : BOOL;
END_STRUCT
END_TYPE
```

Baltech

Name	Description
pCfg	This pointer must contain the memory address of the configuration to be written. This is the configuration structure ST_RFID_CfgStruct_BaltechMifVHLFile.
iCfgSize	This input variable indicates the length in bytes of the configuration data specified via the pointer.

Deister

Name	Description
pCfg	This pointer must contain the memory address of the configuration to be written. This can be both a configuration structure and a configuration register.
iCfgSize	This input variable indicates the length in bytes of the configuration data specified via the pointer.
bUseCfgReg	If the input variable bUseCfgReg is set (TRUE), then a configuration register (byte array) can be addressed via the pointer pCfg instead of a configuration structure. By default a specific configuration structure is specified.
bUseCfgDefault	This parameter is relevant only if the configuration data is present in the form of a specific configuration structure. A configuration structure is not specifically an overall representation of the configuration register. The structure contains only the most important configuration parameters. If the input variable bUseCfgDefault is set (TRUE), then default values are used for the unspecified configuration parameters. Otherwise the value of this configuration parameter is not changed, because the last read values are reused.

Leuze

Name	Description
pCfg	This pointer must contain the memory address of the configuration to be written. This can be both a configuration structure and a configuration register.
iCfgSize	This input variable indicates the length in bytes of the configuration data specified via the pointer.
bUseCfgReg	If the input variable bUseCfgReg is set (TRUE), then a configuration register (byte array) can be addressed via the pointer pCfg instead of a configuration structure. By default a specific configuration structure is specified.
bUseCfgDefault	This parameter is relevant only if the configuration data is present in the form of a specific configuration structure. A configuration structure is not specifically an overall representation of the configuration register. The structure contains only the most important configuration parameters. If the input variable bUseCfgDefault is set (TRUE), then default values are used for the unspecified configuration parameters. Otherwise the value of this configuration parameter is not changed, because the last read values are reused.
iRegIdx	If a special index is specified on iRegIdx, then exclusively this index of the configuration register is changed/read. To this end, iRegGroup must additionally be specified as "SingleRegister". This configuration variant is only available for Leuze electronic RFID readers.
iRegGroup	Three values are available: 0 to change/read the entire configuration register; 1 to change/read the indices 16#00-16#0F of the register; 2 to change/read an individual index of the register, whereby this must be specified with iRegIdx. This configuration variant is only available for Leuze electronic RFID readers.

Further information on RFID reader configuration is summarized in section Configuration [▶ 39].

Requirements

Development Environment	Target Platform	PLC Libraries to include
TC3.1.4013	PC or CX (x86, x64)	Tc2_RFID

6.2.1.6 ST_RFID_Config

The structure indicates the RFID reader configuration that was last read. This is not the parameterization of the TwinCAT RFID library but the proprietary configuration of the RFID reader. This can be queried with the eRFC_GetConfig command (see RFID command set [▶ 21]).

Each configuration can be seen as a register (byte array) or as a structure. Hence, there are various configuration structures in the TwinCAT RFID library (ST_RFID_CfgStruct_DeisterUDL, ST_RFID_CfgStruct_LeuzeRFM,…), which process the raw data from the configuration registers of different RFID readers. Both variants are made available at the output of the library function block. This takes place via pointers. For further evaluation, the MEMCPY () function can be used with the specified data length in bytes.

```
(* defines the configuration as structure and register.
Different RFID Reader in different ReaderGroups can differ in their configuration data. *)
TYPE ST_RFID_Config :
STRUCT
    pCfgStruct     : POINTER TO BYTE;     (* pointer to config structure *)
    pCfgReg        : POINTER TO BYTE;     (* pointer to config register *)
    iCfgStructSize : UINT := 0;  (* size in bytes of the structure *)
    iCfgRegSize    : UINT := 0;  (* size in bytes of the register *)
END_STRUCT
END_TYPE
```

Name	Description
pCfgStruct	This pointer indicates the memory address of the specific configuration structure.
pCfgReg	This pointer indicates the memory address of the specific configuration register.
iCfgStructSize	This output variable indicates the length in bytes of the specific configuration structure.
iCfgRegSize	This output variable indicates the length in bytes of the specific configuration register. If iCfgRegSize = 0, the configuration data is not available as a register (byte array).

Further information on RFID reader configuration is summarized in section Configuration [▶ 39].

Requirements

Development Environment	Target Platform	PLC Libraries to include
TC3.1.4013	PC or CX (x86, x64)	Tc2_RFID

6.2.1.7 ST_RFID_AccessData

If a read or write command is to be executed, it is necessary to specify the input structure stAccessData.

This structure specifies how many and which data should be read and where these should be stored, or how many and which data are to be written.

```
TYPE ST_RFID_AccessData :
STRUCT
    (* access specific parameters *)
    pData       : POINTER TO BYTE;  (* pointer to write data or free space for read data *)
    iDataSize   : UINT;      (* length of data buffer in Bytes *)
    iStartBlock : UINT;      (* attend that the UserDataStartBlock which is not obligatory 0 is added
 automatically. *)
    iBlockCount : UINT;      (* condition: Blockcount*Blocksize=Datasize *)
    iBlockSize  : UINT := 1;  (* in Bytes *)

    iUserDataStartBlock : UINT := 0;  (* depending on the transponder type its user data memory start
s with block index 0 or higher *)
    (* The upper parameter iStartBlock depends on the iUserDataStartBlock. The used StartBlock is iS
tartBlock+iUserDataStartBlock. *)
    (* Different RFID Readers can differ in their interpretation of the first block. *)
    iReserved : UINT;
END_STRUCT
END_TYPE
```

Name	Description
pData	The pointer pData points to the data to be written or to the free memory location for the data to be read.
iDataSize	Specifies the size of the data in bytes to be written/read.
iStartBlock	Specifies the first block index from which data is to be read from or written to the transponder memory. However, different RFID reader models sometimes interpret this index differently. Example: Upon a read command with start index 0, reader A returns the first block of user data and reader B the serial number.
iBlockCount	Specifies the number of blocks that are to be read or written.
iBlockSize	The block size of the user data (in bytes) can be specified by the variable iBlockSize. Depending on the transponder and the RFID reader model, only certain settings are possible here (e.g. 8, 4 or 1 byte is common). This should be ascertained in advance from the transponder information and tested. The variable iBlockSize is similarly to be selected in correspondence with the setting in the RFID reader configuration. Otherwise it is sometimes possible that access to the transponder or the evaluation of the received data cannot take place.
iUserDataStartBlock	With the variable iUserDataStartBlock the start block (as index of blocks) of the user data can optionally be specified on the transponder. Note the block size (iBlockSize). Depending on the transponder, its first blocks can be reserved for system data, such as the serial number. The user data area can accordingly begin at index 0 or also at a higher value. If this is the case, the variable iUserDataStartBlock can be used to specify this additional parameter and to leave the actual index iStartBlock as it is. Both values are added together internally.

Access by different RFID readers to the same transponder

If different RFID readers are to access the same transponder, then access to the transponder memory must be tested in advance. It is possible that a reader model stores the data blocks on the transponder in a rotated byte order compared to another reader model. Or a reader model sees the entire memory area in reverse order compared to another reader model. The readable memory size of the transponder can also vary slightly between different reader models. This depends additionally on the transponder type. The TwinCAT RFID library has no influence on this. The user must select the above input parameters accordingly.

Requirements

Development Environment	Target Platform	PLC Libraries to include
TC3.1.4013	PC or CX (x86, x64)	Tc2_RFID

6.2.1.8 Configuration data

6.2.1.8.1 ST_RFID_CfgStruct_BaltechMifVHLFile

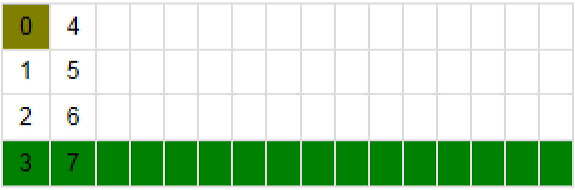
The structure is suitable for writing with the eRFC_SetConfig command (see RFID command set [▶ 21]).

This is not the parameterization of the TwinCAT RFID library but the proprietary configuration of the RFID reader.

```
TYPE ST_RFID_CfgStruct_BaltechMifVHLFile :
STRUCT
    iVHLFile        : USINT := 1;         (* nr. of VHL file to configure *)
    iNrOfKeys       : USINT(1..8) := 1;
    iNrOfSectors    : USINT(1..56) := 16;     (* default: 16 sectors -
> 1024 bytes mifare card with 752 bytes user data *)
    iRC500EEPOffset : USINT := 16#FF;
    arrKeyList      : ARRAY [0..7] OF T_RFID_MifareKey; (* up to 8 keys, 6 byte each *)
    arrSectorList   : ARRAY [0..55] OF BYTE        (* up to 56 sectors accessible *)
                := 0,1,2,3,4,5,6,7,8,9,10, (* default: 16 sectors -
> 1024 bytes mifare card with 752 bytes user data *)
                11,12,13,14,15,16,17,18,19,20,
                21,22,23,24,25,26,27,28,29,30,
                31,32,33,34,35,36,37,38,39,40,
                41,42,43,44,45,46,47,48,49,50,
                51,52,53,54,55;
    arrRdKeyAssign  : ARRAY [0..55] OF BYTE;         (* Key index for each sector *)
    arrWrKeyAssign  : ARRAY [0..55] OF BYTE;         (* Key index for each sector *)
    bMAD_Mode : BOOL := FALSE;          (* use MAD AID [default = FALSE] *)
    iMAD_AID  : USINT;
    iReserved : INT;
END_STRUCT
END_TYPE
```

Structure of a Mifare card (up to 2 kB memory):

A Mifare card with 1 kB memory has 16 sectors with 64 bytes each. Each sector has 4 blocks. Sector 0 consists of blocks 0-3, sector 1 consists of block 4-7, and the following sectors are formed accordingly. In the diagram each column represents a sector, while a box represents a 16-byte block.



0	4														
1	5														
2	6														
3	7														

Only the blocks shown in white contain memory areas that can be used by the user. The maximum size of the user data is therefore 752 bytes (47 x 16 byte) for a 1024-byte Mifare card.

Name	Description
iVHLFile	iVHLFile is used to specify the number of the VHL file to be configured. The configuration of the RFID device can contain several VHL files side by side.
iNrOfKeys	iNrOfKeys is used to specify the required number of keys. 1 to 8 keys can be defined.
iNrOfSectors	iNrOfSectors is used to specify the number of sectors to be used for user data. A 1 KB Mifare card has 16 sectors [default: 16]. Example: If only sectors 4-6 are to be used, iNrOfSectors = 3 is specified.
iRC500EEPOffset	This parameter refers to the internal transfer of the keys within the hardware of the RFID device. The default setting [16#FF] offers enhanced security. It is recommended to leave this setting unchanged.

arrKeyList	All keys are stored in the array arrKeyList. A key is of type T_RFID_MifareKey and consists of 6 bytes.
	Example: If two keys are to be used, they are stored in arrKeyList[0] and arrLeyList[1].
	`TYPE T_RFID_MifareKey :`
	`    ARRAY[0..5] OF BYTE;`
	`END_TYPE`

arrSectorList	In array arrSectorList all sectors are stored that are to be used for user data. Example: If only sectors 4-6 used, the following specification is used: arrSectorList[0]=4, arrSectorList[1]=5, arrSectorList[2]=6. The array is already initialized with consecutive numbering. In most cases no change is required.
arrRdKeyAssign	In the array arrRdKeyAssign the key index for each used sector is stored. Example: For a 1 kB Mifare card all sectors should be used (iNrOfSectors = 16). Two keys are used (iNrOfKeys = 2) The first half of the sectors on this card should be read with the first key (arrKeyList[0]), the second half with the second key (arrKeyList[1]). In the array arrRdKeyAssign the indices stCfg : ST_RFID_CfgStruct_BaltechMifVHLFile := (arrRdKeyAssign:=0,0,0,0,0,0,0,0,1, 1,1,1,1,1,1,1); should therefore be stored. If a single key is to be used for all sectors, the key index is 0 for all sectors. The array is already initialized with 0. In this case no change is required.
arrWrKeyAssign	In the array arrWrKeyAssign the key index for each used sector is stored. Example: For a 1 kB Mifare card all sectors should be used (iNrOfSectors = 16). Two keys are used (iNrOfKeys = 2) The first half of the sectors on this card should be written with the first key (arrKeyList[0]), the second half with the second key (arrKeyList[1]). In the array arrWrKeyAssign the indices stCfg:ST_RFID_CfgStruct_BaltechMifVHLFile:=(arrWrKeyAssign:=0,0,0,0,0,0,0,0,1,1, 1,1,1,1,1,1); should therefore be stored. If a single key is to be used for all sectors, the key index is 0 for all sectors. The array is already initialized with 0. In this case no change is required.
bMAD_Mode	If MAD (Mifare Application Directory) with AIDs (Application Identifiers) is to be used instead of the sector allocation, bMAD_Mode must be set (TRUE). The sector allocation is used as standard [default = FALSE].
iMAD_AID	The input is only required if MAD (Mifare Application Directory) is used (bMAD_Mode = TRUE). At the configuration input iMAD_AID the MAD AID (Application Identifier) for the VHL file is specified.

Further information on the RFID reader configuration process is summarized in section Configuration [⏵ 39].

Detailed information on the VHL file and the configuration of Baltech RFID devices can also be found in the manufacturer's documentation Mifare.pdf and ConfigurationValues.pdf in the Baltech SDK.

Requirements

Development Environment	Target Platform	PLC Libraries to include
TC3.1.4013	PC or CX (x86, x64)	Tc2_RFID

6.2.1.8.2 ST_RFID_CfgStruct_DeisterRDL

The structure is suitable for writing with eRFC_SetConfig and reading with eRFC_GetConfig (see RFID command set [▶ 21]).

This is not the parameterization of the TwinCAT RFID library but the proprietary configuration of the RFID reader.

```
TYPE ST_RFID_CfgStruct_DeisterRDL :
STRUCT
    eOpMode      : E_RFID_OpMode := eRFOP_ReadData;
    eTriggerMode : E_RFID_TriggerMode := eRFTR_ImmediateRead;
    eReadMode    : E_RFID_ReadMode := eRFRD_SingleShot;
    eWriteMode   : E_RFID_WriteMode := eRFWR_ImmediateWrite;

    eNetworkMode : E_RFID_NetworkMode := eRFNM_StandAlone;
    bAFI : BOOL := FALSE;     (* not implemented; ready for future extention *)
    iAFI : BYTE;             (* not implemented; ready for future extention *)

    bSerialNumberMode : BOOL := FALSE;
    bMultiTranspMode  : BOOL := FALSE;
    bOutputAutomatic  : BOOL := TRUE;
    iBlockSize        : USINT := 8;

    tOutputPulseTime : TIME := T#300ms;

    eTranspType : E_RFID_TranspType := eRFTT_TagItHfi;

    iCountBlocksRead  : USINT := 1;
    iCountBlocksWrite : USINT := 1;

    iStartBlockRead  : UINT := 16#4000;
    iStartBlockWrite : UINT := 5;
    arrWriteData     : ARRAY [0..71] OF BYTE;
END_STRUCT
END_TYPE
```

Name	Description

eOpMode	This operating mode defines which function is triggered by a trigger pulse. The command eRFC_TriggerOn or a pulse at the optional trigger input triggers the action set here.
	If eRFOP_WriteData is set, a write access is executed.
	If eRFOP_ReadData is set, a read access is executed [default].
	The response telegram that follows is received by the function block of the RFID library. There is no assignment of read transponder data in this case. The received raw data can be taken from the function block interface for further processing.
	`TYPE E_RFID_OpMode : (`
	`    eRFOP_WriteData,`
	`    eRFOP_ReadData,`
	`    eRFOP_ReadSerialNumber`
	`);END_TYPE`

eTriggerMode	If eRFTR_ImmediateRead is set, the device is always ready to read. With this setting, the trigger condition is always considered met [default].
	if eRFTR_ReadWithTrigger is set, the device only reads if the trigger condition is met. The eRFC_TriggerOn command can be used for this (see RFID command set [▶ 21])
	The subsequent response telegram is received by the function block of the TwinCAT RFID library. There is no assignment of read transponder data in this case. The received raw data can be taken from the function block interface for further processing.
	`TYPE E_RFID_TriggerMode : (`
	`    eRFTR_ImmediateRead,`
	`    eRFTR_ReadWithTrigger`
	`);END_TYPE`

eReadMode	If eRFRD_ContinuousRead is set, the device reads continuously and continuously outputs read data.
	If eRFRD_SingleShot is set, the device reads precisely once [default].
	`TYPE E_RFID_ReadMode : (`
	`    eRFRD_ContinuousRead,`
	`    eRFRD_SingleShot`
	`);END_TYPE`

eWriteMode	If eRFWR_ImmediateWrite is set, the transponder must be in the field in order to execute a write or read command correctly [default].
	If eRFWR_WriteToNextTag is set, the data from a write command is written to the next transponder. ("Preload")
	`TYPE E_RFID_WriteMode : (`
	`    eRFWR_ImmediateWrite,`
	`    eRFWR_WriteToNextTag`
	`);END_TYPE`

eNetworkMode	If eRFNM_Network is set, several devices can be integrated in an RS485 network.
	If eRFNM_StandAlone is set, the device is in stand-alone mode [default].
	The library does not support the operation of several devices within an RS485 network.
	`TYPE E_RFID_NetworkMode :(`
	`    eRFNM_Network,`
	`    eRFNM_StandAlone`
	`);END_TYPE`

bSerialNumberMode	If bSerialNumberMode is TRUE, the serial number is transferred with write and read commands. By default, this corresponds to the last transponder serial number detected with the GetInventory [▶ 23] command. Otherwise, the transponder serial number is specified in ST_RFID_Control [▶ 44].
bMultiTranspMode	If bMultiTranspMode is TRUE, anti-collision is active when several transponders are in the field.
bOutputAutomatic	If bOutputAutomatic is TRUE, the switching output is switched automatically.

Name	Description
iBlockSize	The block size can be set to 4 or 8 bytes. It must match the block size used for reading and writing in ST_RFID_AccessData [▶ 50].
tOutputPulseTime	tOutputPulseTime is used to configure the action time of the output. The pulse duration of the optional output signal can be set between 30 ms and 9000 ms.
eTranspType	If the RFID device is to detect only transponders of a certain type, this can be set with eTranspType. The value 16#FE is used for unlimited options. The following values are possible (E_RFID_TranspType [▶ 65]): eRFTT_ICode eRFTT_STmLRI512 eRFTT_TagIt eRFTT_ICodeSli eRFTT_InfineonSRF55 eRFTT_Inside eRFTT_TagItHfi
iCountBlocksRead	iCountBlocksRead is used to configure the number of blocks that are to be read automatically. The product with iBlockSize provides the number of bytes. The maximum number of blocks is between 4 and 9, depending on the block size and other settings.
iCountBlocksWrite	iCountBlocksWrite is used to configure the number of blocks that are to be written automatically. The product with iBlockSize provides the number of bytes. The maximum number of blocks is between 4 and 9, depending on the block size and other settings.
iStartBlockRead	iStartBlockRead is used to configure the start address for automatic reading.
iStartBlockWrite	iStartBlockWrite is used to configure the start address for automatic writing.
arrWriteData	Write data are limited to a maximum of 72 bytes.

Certain combinations of values are not permitted. The existing dependencies are defined in the proprietary manufacturer's specification. An attempt to write an invalid configuration will result in error eRFERR_InvalidCfg, or the RFID device receives an error code.

Further information on the RFID reader configuration process is summarized in section Configuration [▶ 39].

Requirements

Development Environment	Target Platform	PLC Libraries to include
TC3.1.4013	PC or CX (x86, x64)	Tc2_RFID

6.2.1.8.3 ST_RFID_CfgStruct_DeisterUDL

The structure is suitable for writing with eRFC_SetConfig and reading with eRFC_GetConfig (see RFID command set [▶ 21]).

This is not the parameterization of the TwinCAT RFID library but the proprietary configuration of the RFID reader.

```
TYPE ST_RFID_CfgStruct_DeisterUDL :
STRUCT
    ePollingMode : E_RFID_PollingMode := eRFPO_PollingMode; (* CMD: 0x0A OR    Byte 32, Bit 5 *)
    eTriggerMode : E_RFID_TriggerMode := eRFTR_ImmediateRead; (* Byte 15, Bit 1 *)
    eOpMode      : E_RFID_OpMode := eRFOP_ReadSerialNumber; (* Byte 15, Bit 6,7 *)
    eTranspType  : E_RFID_TranspType := eRFTT_EPC1Gen2; (* Byte 33 *)

    tOutputPulseTime  : TIME := T#300ms; (* Byte 38 and 39 *)
    bOutputLevel      : BOOL;         (* TRUE = high; FALSE = low *)

    iReserved : USINT;

    iCountBlocksRead  : USINT := 1;     (* Byte 41 *)
    iCountBlocksWrite : USINT := 1;     (* Byte 43 *)

    iStartBlockRead   : UINT := 0;     (* Byte 40 *)
    iStartBlockWrite  : UINT := 0;     (* Byte 42 *)
    arrWriteData       : ARRAY [0..31] OF BYTE; (* Byte 44 - 75 *)
END_STRUCT
END_TYPE
```

If applicable, the difference between polling and trigger must be taken into account. In addition it must be considered in this context that the TriggerMode can nevertheless be present in addition to the PollingMode.

Name	Description

ePollingMode	If eRFPO_PollingMode is set, the RFID device sends data only on request [default].
	If eRFPO_ReportMode is set, the RFID device can transfer data automatically at any time.
	`TYPE E_RFID_PollingMode :(`
	`    eRFPO_ReportMode,`
	`    eRFPO_PollingMode`
	`);END_TYPE`

eTriggerMode	If eRFTR_ImmediateRead is set, the device is always ready to read. With this setting, the trigger condition is always considered met [default].
	if eRFTR_ReadWithTrigger is set, the device only reads if the trigger condition is met. The eRFC_TriggerOn command can be used for this (see RFID command set [▶ 21]).
	The subsequent response telegram is received by the function block of the TwinCAT RFID library. There is no assignment of read transponder data in this case. The received raw data can be taken from the function block interface for further processing.
	`TYPE E_RFID_TriggerMode : (`
	`    eRFTR_ImmediateRead,`
	`    eRFTR_ReadWithTrigger`
	`);END_TYPE`

eOpMode	These operating modes are only available with certain transponder types.
	If eRFOP_WriteData is set, a write access is executed as soon as a transponder is detected.
	If eRFOP_ReadData is set, a read access is executed as soon as a transponder is detected.
	If eRFOP_ReadSerialNumber is set, no action is executed. The Polling command returns the serial number [default].
	The subsequent response telegram is received by the function block of the TwinCAT RFID library. There is no assignment of read transponder data in this case. The received raw data can be taken from the function block interface for further processing.
	`TYPE E_RFID_OpMode : (`
	`    eRFOP_WriteData,`
	`    eRFOP_ReadData,`
	`    eRFOP_ReadSerialNumber`
	`);END_TYPE`

eTranspType	If the RFID device is to detect only transponders of a certain type, this can be set with eTranspType. The value 16#FE is used for unlimited options. The following values are possible (E_RFID_TranspType [▶ 65]): eRFTT_EPC1Gen1 eRFTT_EPC1Gen2
tOutputPulseTime	tOutputPulseTime is used to configure the action time of the output. The pulse duration of the optional output signal can be set between 30 ms and 9000 ms.
bOutputLevel	bOutputLevel is used to influence the control of the optional digital output. After a successful read operation the output can be set to HighLevel (bOutputLevel = TRUE) or LowLevel (bOutputLevel = FALSE).
iCountBlocksRead	iCountBlocksRead is used to configure the number of blocks that are to be read automatically. The product with iBlockSize provides the number of bytes. The maximum number of blocks is between 4 and 9, depending on the block size and other settings. The block size depends on the transponder type.
iCountBlocksWrite	iCountBlocksWrite is used to configure the number of blocks that are to be written automatically. The product with iBlockSize provides the number of bytes. The maximum number of blocks is between 4 and 9, depending on the block size and other settings. The block size depends on the transponder type.
iStartBlockRead	iStartBlockRead is used to configure the start address for automatic reading.
iStartBlockWrite	iStartBlockWrite is used to configure the start address for automatic writing.
arrWriteData	Write data are limited to a maximum of 32 bytes.

Further information on the RFID reader configuration process is summarized in section Configuration [▶ 39].

Requirements

Development Environment	Target Platform	PLC Libraries to include
TC3.1.4013	PC or CX (x86, x64)	Tc2_RFID

6.2.1.8.4 ST_RFID_CfgStruct_LeuzeRFM

The structure is suitable for writing with eRFC_SetConfig and reading with eRFC_GetConfig (see RFID command set [▶ 21]).

This is not the parameterization of the TwinCAT RFID library but the proprietary configuration of the RFID reader.

```
TYPE ST_RFID_CfgStruct_LeuzeRFM :
STRUCT
    eOpMode      : E_RFID_OpMode := eRFOP_ReadData;
    eTriggerMode : E_RFID_TriggerMode := eRFTR_ImmediateRead;
    eReadMode    : E_RFID_ReadMode := eRFRD_SingleShot;
    eWriteMode   : E_RFID_WriteMode := eRFWR_ImmediateWrite;

    eNetworkMode : E_RFID_NetworkMode := eRFNM_Network;
    bAFI : BOOL := FALSE; (* not implemented; ready for future extention *)
    iAFI : BYTE;         (* not implemented; ready for future extention *)

    bSerialNumberMode : BOOL         := FALSE;
    bMultiTranspMode  : BOOL         := FALSE;
    bOutputAutomatic  : BOOL         := TRUE;
    iBlockSize        : USINT         := 8;

    tOutputPulseTime  : TIME         := T#300ms;

    eTranspType       : E_RFID_TranspType := eRFTT_TagItHfi;

    iCountBlocksRead  : USINT         := 1;
    iCountBlocksWrite : USINT         := 1;

    iStartBlockRead   : UINT         := 16#4000;
    iStartBlockWrite  : UINT         := 5;
    arrWriteData      : ARRAY [0..71] OF BYTE;
END_STRUCT
END_TYPE
```

Name	Description

eOpMode	This operating mode defines which function is triggered by a trigger pulse.
	The command eRFC_TriggerOn or a pulse at the optional trigger input triggers the action set here.
	If eRFOP_WriteData is set, a write access is executed.
	If eRFOP_ReadData is set, a read access is executed [default].
	The subsequent response telegram is received by the function block of the TwinCAT RFID library. There is no assignment of read transponder data in this case. The received raw data can be taken from the function block interface for further processing.
	`TYPE E_RFID_OpMode : (`
	`    eRFOP_WriteData,`
	`    eRFOP_ReadData,`
	`    eRFOP_ReadSerialNumber`
	`);END_TYPE`

eTriggerMode	If eRFTR_ImmediateRead is set, the device is always ready to read. With this setting, the trigger condition is always considered met [default].
	if eRFTR_ReadWithTrigger is set, the device only reads if the trigger condition is met. The eRFC_TriggerOn command can be used for this (see RFID command set [▶ 21]).
	The subsequent response telegram is received by the function block of the TwinCAT RFID library. There is no assignment of read transponder data in this case. The received raw data can be taken from the function block interface for further processing.
	`TYPE E_RFID_TriggerMode : (`
	`    eRFTR_ImmediateRead,`
	`    eRFTR_ReadWithTrigger`
	`);END_TYPE`

eReadMode	If eRFRD_ContinuousRead is set, the device reads continuously and continuously outputs read data.
	If eRFRD_SingleShot is set, the device reads precisely once [default].
	`TYPE E_RFID_ReadMode : (`
	`    eRFRD_ContinuousRead,`
	`    eRFRD_SingleShot`
	`);END_TYPE`

eWriteMode	If eRFWR_ImmediateWrite is set, the transponder must be in the field in order to execute a write or read command correctly [default].
	If eRFWR_WriteToNextTag is set, the data from a write command is written to the next transponder. ("Preload")
	`TYPE E_RFID_WriteMode : (`
	`    eRFWR_ImmediateWrite,`
	`    eRFWR_WriteToNextTag`
	`);END_TYPE`

eNetworkMode	If eRFNM_Network is set, several devices can be integrated in an RS485 network [default].
	If eRFNM_StandAlone is set, the device is in stand-alone mode.
	The library does not support the operation of several devices within an RS485 network.
	`TYPE E_RFID_NetworkMode :(`
	`    eRFNM_Network,`
	`    eRFNM_StandAlone`
	`);END_TYPE`

bSerialNumberMode	If bSerialNumberMode is TRUE, the serial number is transferred with write and read commands. By default this is the last transponder serial number detected with the command GetInventory. Otherwise the transponder serial number is specified in ST_RFID_Control [▶ 44].
bMultiTranspMode	If bMultiTranspMode is TRUE, anti-collision is active when several transponders are in the field.

Name	Description
bOutputAutomatic	If bOutputAutomatic is TRUE, the switching output is switched automatically.
iBlockSize	The block size can be set to 4 or 8 bytes. It must match the block size set for reading and writing in ST_RFID_AccessData [▶ 50].
tOutputPulseTime	tOutputPulseTime is used to configure the action time of the output. The pulse duration of the optional output signal can be set between 30 ms and 9000 ms.
eTranspType	If the RFID device is to detect only transponders of a certain type, this can be set with eTranspType. The value 16#FE is used for unlimited options. The following values are possible (E_RFID_TranspType [▶ 65]): eRFTT_ICode eRFTT_STmLRI512 eRFTT_TagIt eRFTT_ICodeSli eRFTT_InfineonSRF55 eRFTT_Inside eRFTT_TagItHfi
iCountBlocksRead	iCountBlocksRead is used to configure the number of blocks that are to be read automatically. The product with iBlockSize provides the number of bytes. The maximum number of blocks is between 4 and 9, depending on the block size and other settings.
iCountBlocksWrite	iCountBlocksWrite is used to configure the number of blocks that are to be written automatically. The product with iBlockSize provides the number of bytes. The maximum number of blocks is between 4 and 9, depending on the block size and other settings.
iStartBlockRead	iStartBlockRead is used to configure the start address for automatic reading.
iStartBlockWrite	iStartBlockWrite is used to configure the start address for automatic writing.
arrWriteData	Write data are limited to a maximum of 72 bytes.

Certain combinations of values are not permitted. The existing dependencies are defined in the proprietary manufacturer's specification. An attempt to write an invalid configuration will result in error eRFERR_InvalidCfg, or the RFID device receives an error code.

Further information on the RFID reader configuration process is summarized in section Configuration [▶ 39].

Requirements

Development Environment	Target Platform	PLC Libraries to include
TC3.1.4013	PC or CX (x86, x64)	Tc2_RFID

6.2.1.8.5 ST_RFID_CfgStruct_PepperlFuchsIDENT

The structure is suitable for reading with eRFC_GetConfig (see RFID command set [▶ 21]). This is not the parameterization of the TwinCAT RFID library but the proprietary configuration of the RFID reader.

```
TYPE ST_RFID_CfgStruct_PepperlFuchsIDENT :
STRUCT
    tTimeout       :TIME;
    iBaudrate      :UINT;
    iIdentChannel  :USINT;
    bMultiplexMode :BOOL;
    arrHeadCfg     :ARRAY [0..3] OF ST_RFID_HeadCfg;
    arrTriggerCfg  :ARRAY [0..1] OF ST_RFID_TriggerCfg;
END_STRUCT
END_TYPE
```

The Ident Control Compact device from Pepper+Fuchs consists of a central unit and 1-4 write/read heads. Each of these five elements has an ID channel (Ident Channel) that can be used for assigning commands to individual elements. In the standard case the central unit is assigned channel 0 and the write/read heads channels 1-4.

The command eRFC_GetConfig together with the data at output stReaderCfg can be used to check the settings for all ID channels.

Name	Description
tTimeout	tTimeOut specifies the duration for which the RFID device waits for further telegram data. An error message is issued if the device has not detected a comprehensible command after this time. (The default is 0 ms)
iBaudrate	iBaudrate is used to display the current baud rate of the RFID device. The supported RFID devices have a maximum transfer rate of 38400 baud.
iIdentChannel	ID channel of the central unit
bMultiplexMode	In multiplex mode interference between the write/read heads is minimized, since only one head is active at any time.

arrHeadCfg	Devices with up to four write/read heads are available. Each head has a status and a DataCarrierType. For each head this information is stored in a structure of type ST_RFID_HeadCfg.
	The status of the central unit is output in iErrCodeRcv directly at the output of FB_RFIDReader [▶ 34].
	Possible values for iDCType are explained in ST_RFID_Control [▶ 44].
	`TYPE ST_RFID_HeadCfg :`
	`STRUCT`
	`    eStatus :E_RFID_ErrCodeRcv_PepperlFuchs;`
	`    iDCType :USINT; (* not equal to E_RFID_TranspType enumeration *)`
	`     iReserved :USINT;`
	`END_STRUCT`
	`END_TYPE`

arrTriggerCfg	Devices with up to four write/read heads are available. Each head has an iIdentChannel and a bTriggerMode. For each trigger sensor this information is stored in a structure of type ST_RFID_TriggerCfg. iIdentChannel indicates the write/read head for which the trigger sensor is configured.
	If iTriggerMode is TRUE, the trigger mode is active for a trigger sensor.
	If bInverted is TRUE, it is an inverted trigger signal.
	For more information on the trigger mode, see section Pepperl-Fuchs [▶ 31]
	`TYPE ST_RFID_TriggerCfg :`
	`STRUCT`
	`    iIdentChannel :USINT;`
	`    bTriggerMode :BOOL;`
	`    bInverted :BOOL;`
	`    bReserved :BOOL;`
	`END_STRUCT`
	`END_TYPE`

Further information on the RFID reader configuration process is summarized in section Configuration [▶ 39].

Requirements

Development Environment	Target Platform	PLC Libraries to include
TC3.1.4013	PC or CX (x86, x64)	Tc2_RFID

6.2.2 Enumerations

6.2.2.1 E_RFID_Command

```
TYPE E_RFID_Command : (
    eRFC_Unknown     := 0,
    eRFC_GetReaderVersion,
    eRFC_GetConfig,
    eRFC_SetConfig,
    eRFC_GetInventory,
    eRFC_Polling,
    eRFC_TriggerOn,
    eRFC_TriggerOff,
    eRFC_AbortCommand,
    eRFC_ResetReader,
    eRFC_ReadBlock,
    eRFC_WriteBlock,
    eRFC_OutputOn,
    eRFC_OutputOff,
    eRFC_FieldOn,
    eRFC_FieldOff,
    eRFC_SendRawData,
    eRFC_ChangeDCType,
END_TYPE
```

The function block FB_RFIDReader [▶ 34] of the TwinCAT PLC RFID library offers the enumeration values displayed in the code area at the eCommand input. This is a selection of commands, such as writing to or reading a transponder. See section RFID command set [▶ 21] for a detailed explanation of the commands.

Requirements

Development Environment	Target Platform	PLC Libraries to include
TC3.1.4013	PC or CX (x86, x64)	Tc2_RFID

6.2.2.2 E_RFID_Response

```
TYPE E_RFID_Response : (
    eRFR_NoRsp,
    eRFR_Unknown,
    eRFR_Ready,
    eRFR_CmdConfirmation,
    eRFR_CfgChangeExecuted,
    eRFR_WriteCmdSucceded,
    eRFR_NoTransponder,
    eRFR_Error,
    eRFR_Data_ReaderVersion,
    eRFR_Data_Config,
    eRFR_Data_Inventory,
    eRFR_Data_ReadData,
);
END_TYPE
```

The function block FB_RFIDReader [▶ 34] of the TwinCAT PLC RFID library offers the enumeration values displayed in the code area at the eResponse output. These are partly analogous to the telegram response types of the manufacturer-specific protocols. Which manufacturer-specific MessageID corresponds to the response listed here is indicated in *italics* in the following description. Due to the complexity of the evaluation, not all equivalents are listed. Detailed information is given if necessary by the raw data representation ST_RFID_RawData [▶ 43] at the output of the function block.

Value	Description
eRFR_NoRsp	This value indicates that no response has arrived recently.
eRFR_Unknown	This value indicates that the telegram that arrived could not be assigned to a certain type and was therefore also not evaluated. This is usually accompanied by an error message (bError = TRUE).
eRFR_Ready	Some RFID reader models indicate their operational readiness, e.g. after a reset, with an extra telegram. In this case eResponse assumes the value eRFR_Ready. *Equivalent in proprietary protocol:* *Leuze: 'S'* *Pepperl+Fuchs: Status='2'*
eRFR_CmdConfirmation	The sent command is confirmed with this response. This can occur with many commands. Exceptions are the commands "Changing the configuration" and "Writing data", because these two commands are followed by special confirmations, which are represented by the following two enumeration values. *Equivalent in proprietary protocol:* *Leuze: 'Q2', 'Q4'*
eRFR_CfgChangeExecuted	If the RFID reader configuration has been changed, the RFID reader sends this telegram in order to confirm the action. *Equivalent in proprietary protocol:* *Leuze: 'Q1'*
eRFR_WriteCmdSucceded	The RFID reader sends this confirmation as soon as data has been written to the transponder. *Equivalent in proprietary protocol:* *Leuze: 'Q5'*
eRFR_NoTransponder	This response is given if no transponder is in the reading field. This is not necessarily evaluated as an error, so that the output bError is also not set. *Equivalent in proprietary protocol:* *Leuze: '$18'* *Pepperl+Fuchs: Status='5'*
eRFR_Error	If a telegram has been received that transferred an error code, then eRFR_Error is output at the eResponse output. The transmitted error code is specified in the output variable iErrCodeRcv (see RFID error codes [▸ 75]). If eResponse takes the value eRFR_Error, an error is also signaled at the output of the function block with bError = TRUE. *Equivalent in proprietary protocol:* *Balluff: <NAK>+Failurenumber* *Deister: MessageErrorCode>=16#20* *Leuze: 'Exx', 'Q0'*
eRFR_Data_ReaderVersion	An RFID reader is requested by a version query to send model information. This kind of received data is designated with the value eRFR_Data_ReaderVersion at the eResponse output.
eRFR_Data_Config	A read-out RFID reader configuration is indicated by means of the enumeration value eRFR_Data_Config. *Equivalent in proprietary protocol:* *Leuze: 'G'*
eRFR_Data_Inventory	This type of telegram is indicated if a transponder has been detected or if the serial number of a transponder has been read out.
eRFR_Data_ReadData	The receipt of read-out data from the transponder memory is indicated by the value eRFR_Data_ReadData. *Equivalent in proprietary protocol:* *Deister: MessageID=16#40 or 16#41*

Requirements

Development Environment	Target Platform	PLC Libraries to include
TC3.1.4013	PC or CX (x86, x64)	Tc2_RFID

6.2.2.3 E_RFID_ReaderGroup

```
TYPE E_RFID_ReaderGroup : (
    eRFRG_Unknown,
    eRFRG_BalluffBIS,
    eRFRG_DeisterBasic,
    eRFRG_DeisterRDL,
    eRFRG_DeisterUDL,
    eRFRG_DeisterVReader,
    eRFRG_LeuzeRFM,
    eRFRG_PepperlFuchsIDENT,
    eRFRG_BaltechIDE
);
END_TYPE
```

Requirements

Development Environment	Target Platform	PLC Libraries to include
TC3.1.4013	PC or CX (x86, x64)	Tc2_RFID

6.2.2.4 E_RFID_ReaderManufacturer

```
TYPE E_RFID_ReaderManufacturer : (
    eRFRM_Unknown,
    eRFRM_Balluff,
    eRFRM_Deister,
    eRFRM_Leuze,
    eRFRM_PepperlFuchs,
    eRFRM_Baltech
);
END_TYPE
```

Requirements

Development Environment	Target Platform	PLC Libraries to include
TC3.1.4013	PC or CX (x86, x64)	Tc2_RFID

6.2.2.5 E_RFID_TranspType

```
TYPE E_RFID_TranspType : (
    eRFTT_NoTag,
    eRFTT_TypeUnknown,
    eRFTT_ATA5590,
    eRFTT_ATA5590UID,
    eRFTT_EM4022_4222,
    eRFTT_EM4135,
    eRFTT_EPC1Gen1,
    eRFTT_EPC1Gen2,
    eRFTT_FujitsuMB89R118,
    eRFTT_ICode,
    eRFTT_ICodeSli,
    eRFTT_InfineonSLE55,
    eRFTT_InfineonSRF55,      (* also known as Infineon my-d vicinity *)
    eRFTT_Inside,
    eRFTT_ISO180006TypB,
    eRFTT_LegicPrime,
    eRFTT_LegicAdvant,
    eRFTT_MifareClassic,      (* Philips *)
    eRFTT_MifareUltraLight,
    eRFTT_MifareDESFire,
    eRFTT_STmLRI512,
    eRFTT_TagIt,
    eRFTT_TagItHfi,          (* TI *)
    eRFTT_UCodeEPC119,          (* Philips *)
    eRFTT_PICOPASS,          (* INSIDE Contactless *)
);
END_TYPE
```

Requirements

Development Environment	Target Platform	PLC Libraries to include
TC3.1.4013	PC or CX (x86, x64)	Tc2_RFID

6.2.3 T_RFID_TranspSRN

```
(* serial number shown as hex coded string(ascii) *)
TYPE T_RFID_TranspSRN : STRING(iRFID_MAXSRNLENGTH);
END_TYPE
```

The data type contains an RFID transponder serial number. The serial number of the transponder (often 8 bytes) is specified as a string in hexadecimal format. (iRFID_MAXSRNLENGTH = 51)

Requirements

Development Environment	Target Platform	PLC Libraries to include
TC3.1.4013	PC or CX (x86, x64)	Tc2_RFID

6.3 Global Constants

6.3.1 Global_version

All libraries have a specific version. This version is shown in the PLC library repository too. A global constant contains the library version information:

```
VAR_GLOBAL CONSTANT
    stLibVersion_Tc2_RFID : ST_LibVersion;
END_VAR
```

To compare the existing version to a required version the function F_CmpLibVersion (defined in Tc2_System library) is offered.

All other possibilities known from TwinCAT 2 libraries to query a library version are obsolete.

7 Examples

The following examples have been developed using different RFID reader models.

Since there are basically hardly any differences in the handling of the RFID reader with the TwinCAT RFID library, an example developed with another model can also be used for familiarization.

Tutorial

The Tutorial describes how an RFID reader is put into operation. It is a step-by-step procedure, from the integration of the TwinCAT library through to the detection of the presence of RFID transponders. (See Tutorial [▸ 67])

Example 1

This example can be used for different RFID readers (Balluff, Baltech, Deister, Leuze, Pepper+Fuchs).

The example has been tested with the Balluff BIS M 401 and Leuze electronic RFM32 models. In the project, an RFID reader was connected to a single-channel serial EL6001 (on an EK1100). Other serial terminals can also be used.

The project can similarly be used for two RFID readers. The sample program is already prepared for two RFID readers. Only the second linking in the TwinCAT System Manager needs to be carried out. (See Sample 1 [▸ 71])

Example 2

This example was developed with a Baltech RFID reader, which is optionally installed in the Beckhoff Control Panels and Panel PCs. The device is connected to a serial Com port or a USB port.

It can be used for convenient commissioning and testing of the device. The example features a simple visualization. (See Sample 2 [▸ 72])

Example 3

This example corresponds to a small application. The application includes detection, reading and writing of a transponder in an automatic sequence.

The example was created with a Pepperl+Fuchs RFID reader. Both the 2-channel and the 4-channel model can be used. (See Sample 3 [▸ 73])

7.1 Tutorial

This tutorial shows how you can commission an RFID reader using the TwinCAT PLC. The tutorial is conducted with the RFID reader model Balluff BIS M 401. In principle, however, the same procedure can be adopted for other models.

The following steps are thereby performed:

1. Glossary
2. Installation/libraries
3. Serial connection
4. Function block declaration
5. Function block usage
6. Test

Assumptions:

- You have planned your RFID application in detail and already modeled your application.
- You have determined that a reader supported by the Tc2_RFID library basically meets your requirements and you can implement your application with the commands offered.

- You decide to use the TwinCAT RFID library to facilitate communication with your RFID reader.

Download: https://infosys.beckhoff.com/content/1033/tf6600_tc3_rfid/Resources/234213899/.zip

Requirements

Development Environment	Target Platform	PLC Libraries to include
TC3.1.4013	PC or CX (x86, x64)	Tc2_RFID

7.1.1 Glossary

Term	Explanation
CT	Carrier type, data carrier type, transponder type
DC	Data carrier - RFID transponder (data memory)
HF	High Frequency
Label	RFID Transponder
LF	Low Frequency
RF	Radio Frequency
RFID	Radio Frequency Identifícation
RFID Reader	A RFID reader can be both a purely readable device and a read/ write device.
Smart Label	RFID Transponder
SRN	Serial Number
Tag	RFID Transponder
UHF	Ultra High Frequency

See: Installation/libraries [▸ 68]

7.1.2 Installation/libraries

1. Start the TwinCAT XAE.
2. Create a new TwinCAT project (menu **File** > command **New**).
3. Create a new PLC project (context menu of the PLC object in the TwinCAT project tree > command **Add New Item**).
4. Select your target platform PC and CX (x86, x64).
5. Add the libraries Tc2_RFID and Tc2_SerialCom to the PLC project (context menu of the Reference object > command **Add library**)

⇨ All required PLC modules for RFID reader communication are available.

See: Serial connection [▸ 68]

7.1.3 Serial connection

In this example the RFID reader is connected via an EL 6001 serial EtherCAT Terminal.

1. Create a send buffer and a receive buffer (gEL6ComTxBuffer, gEL6ComRxBuffer) of type "ComBuffer". In addition, you should create two data structures as they are used for serial communication in the TwinCAT System Manager.

```
gEL6ComRxBuffer          :ComBuffer;
gEL6ComTxBuffer          :ComBuffer;
EL6ComInData      AT %I* : EL6ComInData;
EL6ComOutData     AT %Q* : EL6ComOutData;
```

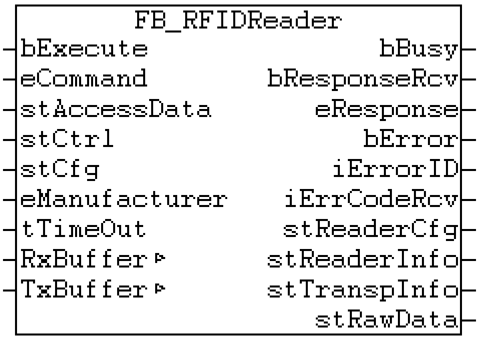
2. Link the structures in the TwinCAT System Manager to the channels of the serial port.
3. For serial communication, create an instance of the SerialLineControl. Call this cyclically in a fast task. (Mode: Enter the EtherCAT Terminal with 22 bytes of user data as handle.)

```
LineControl(
    Mode       := SERIALLINEMODE_EL6_22B,
    pComIn     := ADR(EL6ComInData),
    pComOut    := ADR(EL6ComOutData),
    SizeComIn  := SIZEOF(EL6ComInData),
    TxBuffer   := gEL6ComTxBuffer,
    RxBuffer   := gEL6ComRxBuffer
);
```

See: RFID reader connection [▶ 19] and Function block declaration [▶ 69]

7.1.4 Function block declaration

The function block FB_RFIDReader is the core of the entire RFID reader communication. The declaration and initialization of the function block is described below.



1. Create an instance of the function block FB_RFIDReader.
2. Hand over the manufacturer of your RFID model to the instance at the eManufacturer input.

```
fbRFIDReader : FB_RFIDReader := (eManufacturer := eRFRM_Balluff);
sTranspSerialNumber : STRING;
```

FB_RFIDReader has 7 inputs (6 in the case of the specific FBs due to the missing manufacturer data), 2 inputs/outputs and 10 outputs. In order to receive messages sent by the RFID reader to the controller, it is sufficient to call the function block cyclically. The bExecute input must thereby remain FALSE. This is used in this example in order to implement simple presence detection for the time being.

3. Call the function block as follows:

```
fbRFIDReader(
    bExecute     := FALSE,

    RxBuffer     := RxBuffer,
    TxBuffer     := TxBuffer,

    bBusy        => ,
    bError       => ,
    iErrorID     => ,
    iErrCodeRcv =>
);
sTranspSerialNumber := fbRFIDReader.stTranspinfo.sSerialNumber;
```

Now the last read serial number of an RFID transponder is shown in your string variable. For error analysis the outputs bError and iErrorID etc. should also be evaluated.

See: Using the function block [▶ 69]

7.1.5 Using the function block

You can achieve a more effective evaluation of the received data with the following instructions:

Declarations:

```
fbRFIDReader        : FB_RFIDReader := (eManufacturer := eRFRM_Balluff);
sTranspSerialNumber : STRING;

bBusy       : BOOL;
bError      : BOOL;
```

```
iErrorID    : UINT;
iErrCodeRcv : UINT;

stTranspInfo : ST_RFID_TranspInfo;

eErrorID    : E_RFID_ErrID;
eErrCodeRcv : E_RFID_ErrCodeRcv_Balluff;

fbTriggerResponse : R_TRIG;
arrRspRcv         : ARRAY[0..99] OF BYTE;
```

Program sequence:

```
fbRFIDReader(
    bExecute    := FALSE,

    RxBuffer    := RxBuffer,
    TxBuffer    := TxBuffer,

    bBusy       => bBusy,
    bError      => bError,
    iErrorID    => iErrorID,
    iErrCodeRcv => iErrCodeRcv
);
(* convert Error Codes *)
eErrorID := UINT_TO_INT(iErrorID);
eErrCodeRcv := UINT_TO_INT(iErrCodeRcv);

fbTriggerResponse(CLK := fbRFIDReader.bResponseRcv);
IF (fbTriggerResponse.Q) THEN
    stTranspInfo := fbRFIDReader.stTranspInfo;
    sTranspSerialNumber := sTranspInfo.sSerialNumber;      (* detected RFID Tag Serial Number *)

    MEMSET(ADR(arrRspRcv), 0 , SIZEOF(arrRspRcv) );
    MEMCPY(ADR(arrRspRcv), fbRFIDReader.stRawData.pReceivedRsp, MIN(fbRFIDReader.stRawData.iReceived
RspLen, SIZEOF(arrRspRcv)) );
END_IF
```

Received error codes can be represented online as enumeration values by directly assigning the integer variables iErrorID and iErrCodeRcv.

Using a trigger, additional data is only evaluated when a new message is received.

The string variable sTranspSerialNumber now always returns the serial number of the last detected transponder. In this case this can also be seen at the function block output fbRFIDReader.stTranspInfo.sSerialNumber.

Depending on the application, further information can be taken from the outputs of the function block.

In order to display a received message as a complete byte sequence, use the MEMCPY function, for example, and copy the raw data into your declared byte array.

Each message from your RFID reader is now received and evaluated in the above manner.

See: Test [▶ 70]

7.1.6 Test

As soon as you have created the program according to the procedure described, activate the current configuration with the linked variables in the TwinCAT System Manager and set TwinCAT to Run mode. Log into the controller and start the application.

If you move a transponder in front of your RFID reader, this is detected and the received message as well as the serial number are adopted in the program code. The values are represented online in the program code or in an additional visualization.

Device configuration

The Balluff RFID reader must be configured in a way that is suitable for this functionality. The option **Type and serial number with CT pres.** must be activated. These and other configuration parameters are described in more detail in the section Balluff > RFID reader settings [▶ 26]. Not every RFID reader supports this setting.

Download: https://infosys.beckhoff.com/content/1033/tf6600_tc3_rfid/Resources/234213899/.zip

7.2 Sample 1

This example can be used for different RFID readers (Balluff, Baltech, Deister, Leuze, Pepper+Fuchs).

The example has been tested with the Balluff BIS M 401 and Leuze electronic RFM32 models.

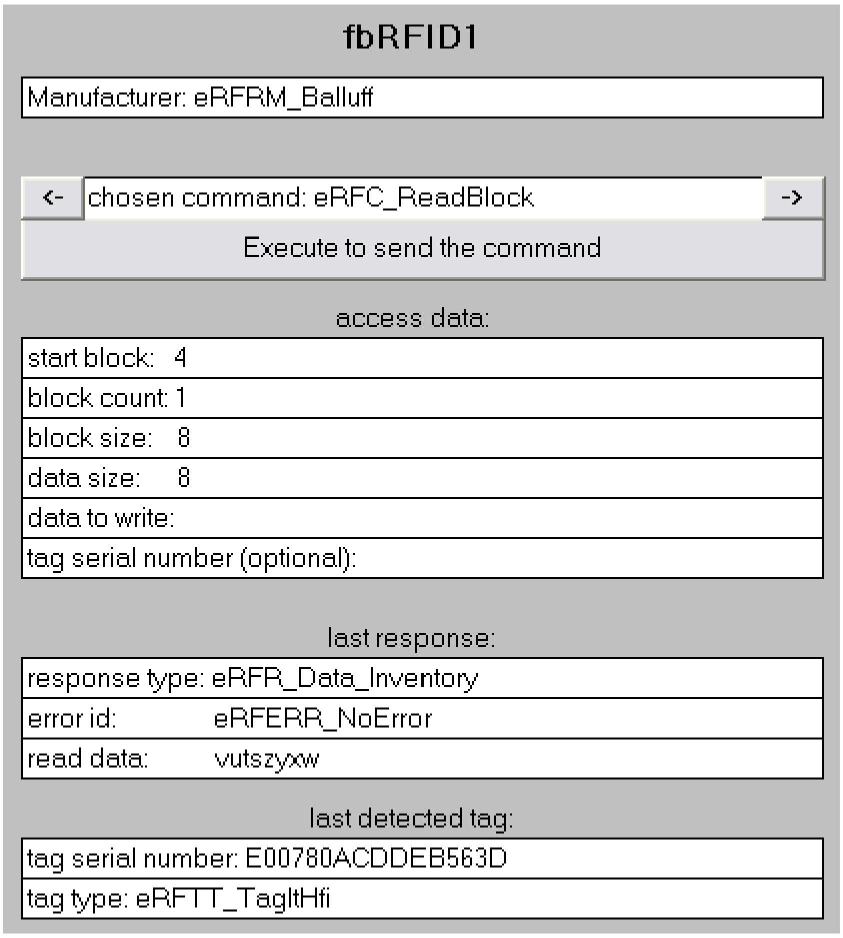
In the project, an RFID reader was connected to a single-channel serial EL6001 (on an EK1100). Other serial terminals can also be used. When using KL terminals, the call of the Serial Line Control in the program code must be adapted. (See RFID reader connection [▶ 19])

The project can similarly be used for two RFID readers. The sample program is already prepared for two RFID readers. Only the second linking in the TwinCAT System Manager needs to be carried out.

The example project contains the call of the RFID function block with different commands. The most important commands were implemented in this example. This includes reading from and writing to the RFID transponder memory.

After starting the program, a suitable RFID reader manufacturer must be selected via the local enumeration eManufacturer.

The command type can be selected using the local eCommand enumeration. In order to start the call, the local variable bExecute must be set once to TRUE. After that, the results of the query are shown at the outputs of the RFID function block. Alternatively, the operation can be take place with the visualization integrated in the example:



fbRFID1

Manufacturer: eRFRM_Balluff

chosen command: eRFC_ReadBlock

access data:

start block: 4
block count: 1
block size: 8
data size: 8
data to write:
tag serial number (optional):

last response:

response type: eRFR_Data_Inventory
error id: eRFERR_NoError
read data: vutszyxw

last detected tag:

tag serial number: E00780ACDDEB563D
tag type: eRFTT_TagItHfi

Depending on the RFID reader model, the GetReaderVersion [▶ 22] and GetConfig [▶ 22] (if necessary also SetConfig [▶ 23]) commands must be executed first, in order to make correct communication with the RFID reader possible.

For further information on the RFID reader communication process, refer to the section Function block FB_RFIDReader > Handling instructions [▶ 38].

Download: https://infosys.beckhoff.com/content/1033/tf6600_tc3_rfid/Resources/5252190859/.zip

Requirements

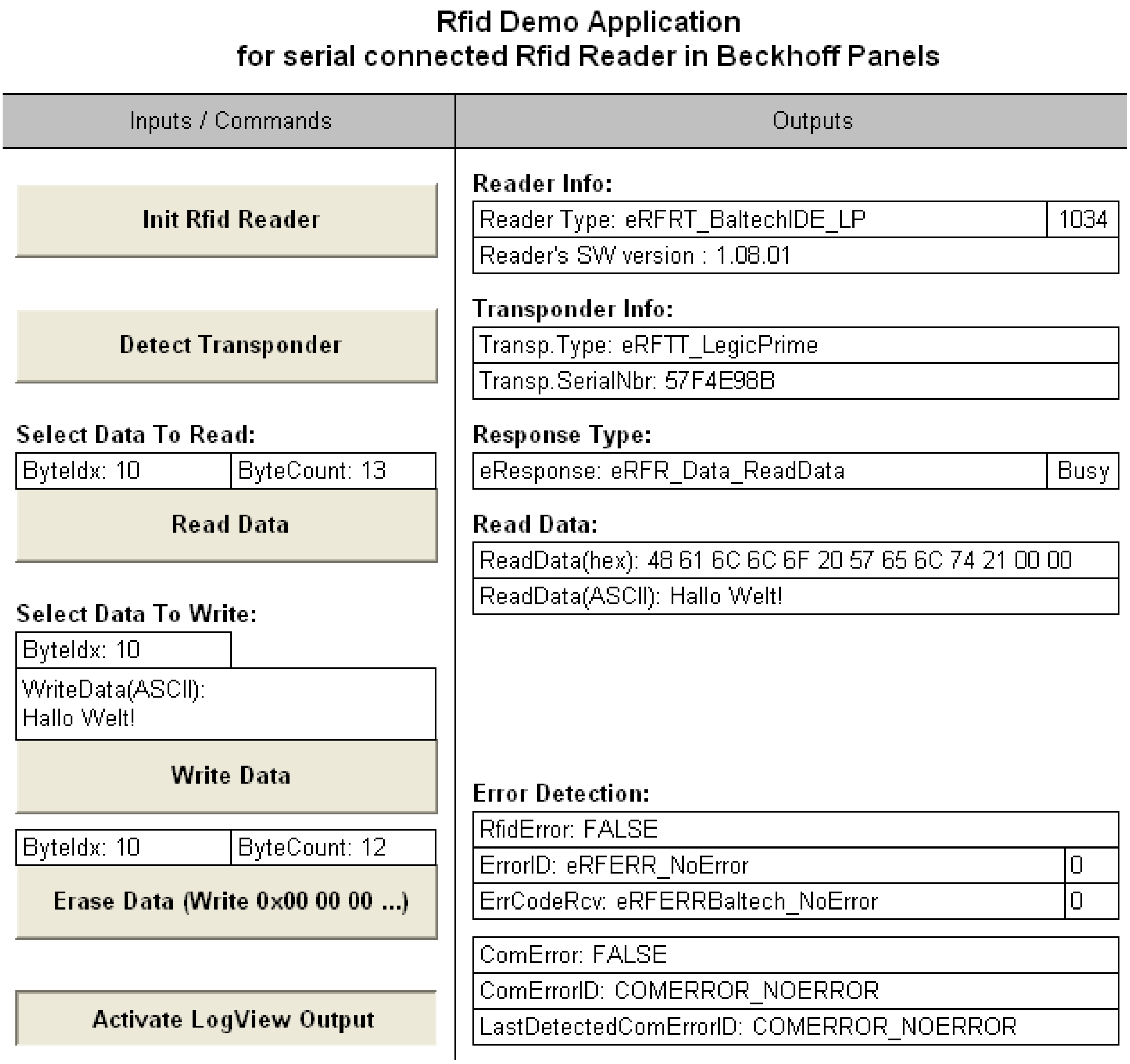
Development Environment	Target Platform	PLC Libraries to include
TC3.1.4013	PC or CX (x86, x64)	Tc2_RFID

7.3 Sample 2

This example was developed with a Baltech RFID reader, which is optionally installed in the Beckhoff Control Panels and Panel PCs. The device is connected to a serial Com port or a USB port. It can be used for convenient commissioning and testing of the device.

If a Baltech RFID reader is used that is connected to a serial Beckhoff terminal instead of to the Com port, the serial background communication in the PLC code must be changed and reconfigured in the TwinCAT System Manager. (See RFID reader interfacing [▶ 19])

The example project contains the call of the RFID function block with different commands. The commands can be executed using the built-in visualization. The commands GetReaderVersion [▶ 22], GetInventory [▶ 23], ReadBlock [▶ 24] and WriteBlock [▶ 24] are implemented. Thus, an RFID transponder can also be tested and data can be written to and read from it in the form of an ASCII string.



Rfid Demo Application
for serial connected Rfid Reader in Beckhoff Panels

Inputs / Commands

Select Data To Read:

ByteIdx: 10 | ByteCount: 13

Select Data To Write:

ByteIdx: 10

WriteData(ASCII):
Hallo Welt!

ByteIdx: 10 | ByteCount: 12

Outputs

Reader Info:

Reader Type: eRFRT_BaltechIDE_LP | 1034

Reader's SW version : 1.08.01

Transponder Info:

Transp.Type: eRFTT_LegicPrime

Transp.SerialNbr: 57F4E98B

Response Type:

eResponse: eRFR_Data_ReadData | Busy

Read Data:

ReadData(hex): 48 61 6C 6C 6F 20 57 65 6C 74 21 00 00

ReadData(ASCII): Hallo Welt!

Error Detection:

RfidError: FALSE

ErrorID: eRFERR_NoError | 0

ErrCodeRcv: eRFERRBaltech_NoError | 0

ComError: FALSE

ComErrorID: COMERROR_NOERROR

LastDetectedComErrorID: COMERROR_NOERROR

The RFID reader must first be initialized to enable correct communication with the RFID reader. The button **Init Rfid Reader** executes the command GetReaderVersion.

For further information on the RFID reader communication process, refer to the section Function block FB_RFIDReader > Handling instructions [▶ 38].

Activating LogView output displays the complete serial transmission in the TwinCAT System Manager LoggerView.

Download: https://infosys.beckhoff.com/content/1033/tf6600_tc3_rfid/Resources/227370251/.zip or https://infosys.beckhoff.com/content/1033/tf6600_tc3_rfid/Resources/949454859/.zip

Requirements

Development environment	Target platform	PLC libraries to include
TC3.1.4013	PC or CX (x86, x64)	Tc2_RFID v3.3.6.0

7.4 Sample 3

This example corresponds to a small application. The application includes detection, reading and writing of a transponder in an automatic sequence.

The example was created with a Pepperl+Fuchs RFID reader. Both the 2-channel and the 4-channel model can be used.

In the example, the device is connected directly to the Com port. If a Pepperl+Fuchs RFID reader is used that is connected to a serial Beckhoff terminal instead of to the Com port, the serial background communication in the PLC code must be changed and reconfigured in the TwinCAT System Manager. (See RFID reader interfacing [▶ 19])

Sequence of the implemented application:

Two read/write heads are connected to the RFID device. Both detect transponders arriving in the field fully automatically. After detection, a memory block is read out from the transponder's data memory. The 4-byte value in it is incremented by one by the first read head or decremented by one by the second read head. The new value is immediately written back to the transponder. This process of detection, reading and writing takes about half a second in total. There must be an interval of at least three seconds between two such processes on the same read head in order to avoid unwanted multiple execution. (This can also be solved by checking the tag serial number.)

The program essentially contains a state machine with 6 states:

0: Initialization - running GetReaderVersion, GetConfig etc.
1: Tag detection on read head 1 - buffered GetInventory
2: Tag detection on read head 2 - buffered GetInventory
3: Waiting for tag detection
4: Action at read head 1 - ReadBlock and WriteBlock
5: Action at read head 2 - ReadBlock and WriteBlock

The data carrier type (iUSEDDCTYPE) should be adapted to the transponder types used.

The example project contains the call of the RFID function block with different commands.

For further information on the RFID reader communication process, refer to the section Function block FB_RFIDReader > Handling instructions [▶ 38].

Download: https://infosys.beckhoff.com/content/1033/tf6600_tc3_rfid/Resources/5252195339/.zip

Requirements

Development Environment	Target Platform	PLC Libraries to include
TC3.1.4013	PC or CX (x86, x64)	Tc2_RFID

8 Appendix

8.1 RFID error codes

Error outputs are provided at two outputs of the RFID PLC function block. The two output variables iErrorID [▶ 75] and iErrCodeRcv [▶ 77] are described below.

iErrorID

If there is an error, the output variable iErrorID shows the type of error. The following list shows the possible values.

```
TYPE E_RFID_ErrID :(
    eRFERR_NoError             := 0,

    (* general errors *)
    eRFERR_TimeOutElapsed     := 16#4001,

    (* invalid input parameters *)
    eRFERR_InvalidCommand        := 16#4101,
    eRFERR_IncompatibleCfg       := 16#4102,
    eRFERR_InvalidManufacturer   := 16#4103,
    eRFERR_InvalidTimeOutParam   := 16#4104,
    eRFERR_InvalidRawDataParam   := 16#4105,
    eRFERR_InvalidAccessData     := 16#4106,
    eRFERR_InvalidCfg            := 16#4107,
    eRFERR_InvalidCfgParam       := 16#4108,
    eRFERR_InvalidCtrlHeadNumber := 16#4109,

    (* error at receive of response *)
    eRFERR_InvalidResponse  := 16#4201,
    eRFERR_InvalidRspLen    := 16#4202,
    eRFERR_InvalidBlocksize := 16#4203,
    eRFERR_ErrorRcv         := 16#4204,
    eRFERR_ChecksumError    := 16#4205,

    (* internal errors *)
    eRFERR_UnknownReaderGroup    := 16#4401,
    eRFERR_CreatedTelegramTooBig := 16#4402,
);
END_TYPE
```

If iErrorID has the value eRFERR_ErrorRcv, an error code was received from the RFID reader. This received error code is indicated in the output variable iErrCodeRcv.

If iErrorID has a different value it may nevertheless be the case that an error has additionally been received from the RFID reader, which is also indicated on iErrCodeRcv.

Value	ID (hex)	ID (dec)	Description
eRFERR_TimeOutElapsed	0x4001	16385	This error occurs if the time period specified as timeout at the input (default = 5 s) has elapsed. Any action still being processed will hence be aborted. A timeout error also occurs if serial background communication does not work. This may be the case, for example, if an incorrect baud rate is set or if the task cycle time is not sufficient to process the data. See also: RFID reader connection [▶ 19] and Documentation of the PLC library serial communication.
eRFERR_InvalidCommand	0x4101	16641	The command has not been executed. The selected command cannot be executed with this RFID reader model. The available commands are listed in the instruction set [▶ 21]. To ensure that the function block knows the existing RFID reader model, the GetReaderVersion command must be executed first, if possible.
eRFERR_IncompatibleCfg	0x4102	16642	The command has not been executed. The current RFID reader configuration (see output structure ST_RFID_Config [▶ 49]) is not compatible with the selected command (and the associated input parameters). Ensure that the configuration has been read once before. Otherwise this error code may also occur.
eRFERR_InvalidManufacturer	0x4103	16643	The RFID manufacturer of the RFID reader model must be specified with the variable eManufacturer at the input of the generic function block FB_RFIDReader. The error eRFERR_InvalidManufacturer indicates invalid data.
eRFERR_InvalidTimeOutParam	0x4104	16644	This error is output if the input variable "tTimeOut" is invalid. The condition tTimeOut > tPreSendDelay + tPostSendDelay applies.
eRFERR_InvalidRawDataParam	0x4105	16645	If raw data is to be sent, it must be specified at the inbound channel of the function block. The error eRFERR_InvalidRawDataParam is output if the specification of the input variable pRawDataCommand or iRawDataCommandLen is invalid. See also: Low-level communication [▶ 40].
eRFERR_InvalidAccessData	0x4106	16646	The command was not executed because the parameters specified at the input in the structure ST_RFID_AccessData [▶ 50] are invalid.
eRFERR_InvalidCfg	0x4107	16647	The command "Changing the configuration" was not executed because the specified configuration data are invalid. The configuration data are present as a configuration structure (bUseCfgReg = FALSE). In the case of doubt, the exact valid ranges of values for the data are given in the manufacturer's proprietary protocol specification. If the last read configuration is used for parameters that are not available in the configuration structure (bUseCfgDefault = FALSE in ST_RFID_CfgIn [▶ 48]), make sure that the configuration was read first. Otherwise this error code may also occur.
eRFERR_InvalidCfgParam	0x4108	16648	The command "Changing the configuration" was not executed because the specified input parameters of the configuration data are invalid. The configuration data should be present as a configuration register or a configuration structure. Its length or another parameter in ST_RFID_CfgIn [▶ 48] is invalid.
eRFERR_InvalidCtrlHeadNumber	0x4109	16649	If an RFID reader with several read heads is addressed, the read head is specified in the input structure ST_RFID_Control [▶ 44]. If no read head is available for the specified value, this error value is output.
eRFERR_InvalidResponse	0x4201	16897	This error is output if the byte sequence of the received response does not correspond to any known message type or if the individual bytes do not exhibit the necessary values. The received data can be analyzed as a raw data block at the output ST_RFID_RawData [▶ 43].
eRFERR_InvalidRspLen	0x4202	16898	This error message is generated if the byte sequence theoretically corresponds to a known message type, but the length is not as expected. The received data can be analyzed as a raw data block at the output ST_RFID_RawData [▶ 43].

Value	ID (hex)	ID (dec)	Description
eRFERR_InvalidBlocksize	0x4203	16899	If this error value occurs, then the received data read out from the transponder could not be evaluated. The number of received bytes indicates that the configured block size does not correspond to the input at the command call.
eRFERR_ErrorRcv	0x4204	16900	An error code was sent with the received message. The error displayed by the RFID reader is likewise output and represented by the output variable iErrCodeRcv (description at the end of this section).
eRFERR_ChecksumError	0x4205	16901	Depending on the protocol specification a checksum, for example a CRC checksum, is sent with the telegram. If a telegram with an incorrect checksum is received by the controller, then this error is output.
eRFERR_UnknownReaderGroup	0x4401	17409	The RFID library assigns the RFID reader models internally to groups. The error eRFERR_UnknownReaderGroup can occur if the RFID reader is not yet assigned to a group. Therefore, depending on the RFID reader model, the "GetReaderVersion" command must be executed as the first query when the program starts.
eRFERR_CreatedTelegramTooBig	0x4402	17410	An attempt was made to send a telegram that exceeded the maximum size of 300 bytes. Only telegrams with up to 300 bytes can be sent.

In a few cases, the RFID device sends several telegrams directly one after the other. It is therefore important to always detect and correct the first error that occurred.

iErrCodeRcv

If an error code is sent along by the RFID reader, it is output at the output variable iErrCodeRcv. Sometimes status messages are sent by the RFID reader and then sent to iErrCodeRcv, which do not lead to an error (bError remains FALSE and iErrorID shows no error).

A list of possible values can either be found in the data type declarations (E_RFID_ErrCodeRcv_Balluff, E_RFID_ErrCodeRcv_Deister, E_RFID_ErrCodeRcv_Leuze etc.) of the PLC RFID library via the TwinCAT XAE library details or directly in the protocol specification.

For further error analysis, reference is made to the logging option. To this end, the input parameter bLogging is set. See the Parameter description in API [▸ 44] for details.

8.2 Support and Service

Beckhoff and their partners around the world offer comprehensive support and service, making available fast and competent assistance with all questions related to Beckhoff products and system solutions.

Download finder

Our download finder contains all the files that we offer you for downloading. You will find application reports, technical documentation, technical drawings, configuration files and much more.

The downloads are available in various formats.

Beckhoff's branch offices and representatives

Please contact your Beckhoff branch office or representative for local support and service on Beckhoff products!

The addresses of Beckhoff's branch offices and representatives round the world can be found on our internet page: www.beckhoff.com

You will also find further documentation for Beckhoff components there.

Beckhoff Support

Support offers you comprehensive technical assistance, helping you not only with the application of individual Beckhoff products, but also with other, wide-ranging services:

- support
- design, programming and commissioning of complex automation systems
- and extensive training program for Beckhoff system components

Hotline:	+49 5246 963-157
e-mail:	support@beckhoff.com

Beckhoff Service

The Beckhoff Service Center supports you in all matters of after-sales service:

- on-site service
- repair service
- spare parts service
- hotline service

Hotline:	+49 5246 963-460
e-mail:	service@beckhoff.com

Beckhoff Headquarters

Beckhoff Automation GmbH & Co. KG

Huelshorstweg 20
33415 Verl
Germany

Phone:	+49 5246 963-0
e-mail:	info@beckhoff.com
web:	www.beckhoff.com

More Information:
www.beckhoff.com/tf6600

Beckhoff Automation GmbH & Co. KG
Hülshorstweg 20
33415 Verl
Germany
Phone: +49 5246 9630
info@beckhoff.com
www.beckhoff.com