BECKHOFF New Automation Technology

Manual | EN

TS6600

TwinCAT 2 | PLC RFID Reader Communication

Supplement | Communication

2022-11-09 | Version: 1.1

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 Introduction

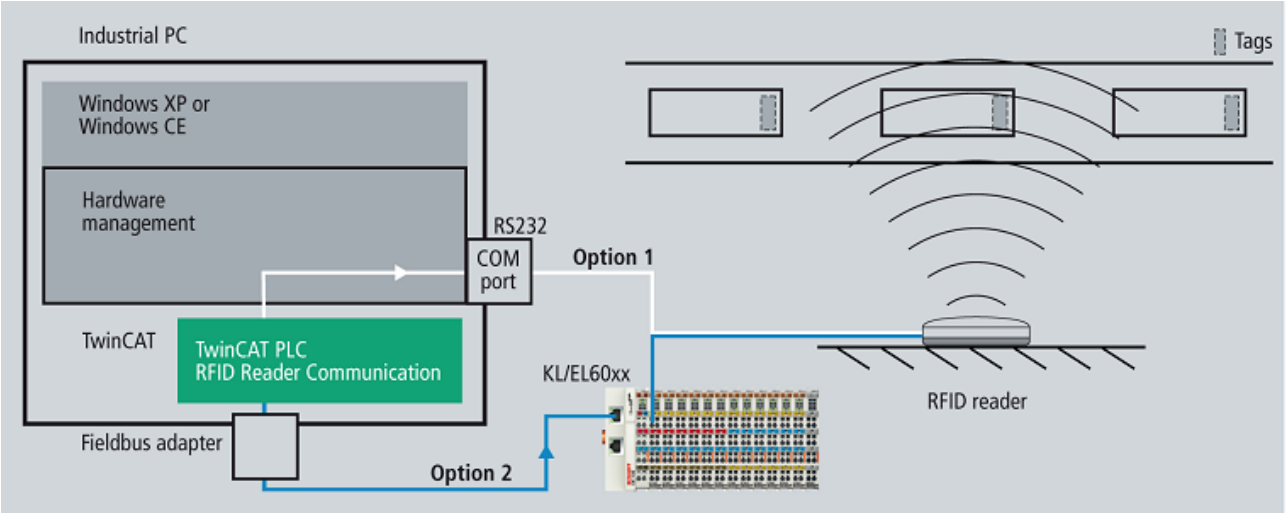
The TwinCAT RFID Reader Communication library makes communication with RFID readers possible from the PLC program.

RFID readers can be both pure readers and write/read devices. The term RFID reader is therefore used below without implying any restriction of functionality.

Even extensive applications that use different functions of the RFID reader can now be implemented easily. The implementation expenditure is very low, because no manufacturer-specific interface protocol needs to be investigated in detail and implemented. The frame structure, telegram composition, command designation, telegram detection and several other protocol peculiarities are implemented automatically by the library. The user can concentrate fully on his application and saves valuable development time.

The handling of the PLC library is the same for all supported RFID reader models. Even in the case of a change of manufacturer, only small changes need to be made to the application.

With the TwinCAT RFID library, the entire world of RFID technology can be used for the first time without expenditure.



Schematic diagram of an RFID reader connection [▶ 14]

Documentation last updated: 14.10.2013

3 System requirements

- Programming environment:
  - XP, XPe, WES, WES7, Win7, Win10
  - TwinCAT installation level: TwinCAT PLC or higher;
  - TwinCAT System Version 2.10.0 Build 1340 or higher;
  - **TcRFID.lib** This PLC library must be integrated in the PLC project. All other libraries are added automatically.
    (Standard.lib; TcBase.lib; TcSystem.lib; TcUtilities.lib; COMlibV2.lib are included automatically)
- Target platform:
  - PC or CX (x86): XP, XPe, WES, WES7, CE, Win7, Win10
  - TwinCAT PLC runtime system version 2.10.0 build 1340 or higher;

Depending on the RFID reader model, a manufacturer’s own tool is required once only for the basic configuration. (see chapter: 'RFID Reader Settings and Handling') In this case, its system requirements should be observed. This pre-setting can be similarly be performed from another PC. The use of proprietary test tools for the setup is also possible.

4 RFID Reader Hardware

Assembly instructions are to be taken from the manufacturer's own product manuals. Information on the handling between RFID transponder and readers as well as reading speeds etc. is likewise to be procured from the respective manufacturer.

RFID readers sometimes offer an external trigger or a switching output. This does not have to be used for the functions of the TwinCAT RFID library.

Note

The TcRFID library does not illustrate the complete scope of the manufacturer's own RFID communication protocol. More detailed information is also given in the command set for the library block [▶ 17]. In addition, recourse can be taken to the integrated possibility of sending and receiving raw data - see the command 'eRFC_Send_RawData' regarding this.

RFID reader models

The TwinCAT RFID library supports different RFID reader models.

The following table indicates which RFID reader models of which manufacturers are supported.

The images are symbol photos of RFID reader models. There may be variations to the supported models and likewise not every supported model is shown as a photo.

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) [from library v.1.2.5] BIS L-6020-007 (2 read heads) (RS232) [from library v.1.2.5]	  Picture credits: BALLUFF
Baltech [from library v.1.2.1]	ID-engine SD-M1415-ANT1F (RS232 or USB) ID-engine SD-LP-ANT1F (RS232 or USB) ID-engine PAD M1415 (USB) [USB from setup v1.3.0]	

RFID reader manufacturer	RFID reader model	Sample photo
Deister electronic	RDL90 (deBus) (RS232, RS485) UDL 500 (deBus) (RS485) PRM5M/2V (deBus) (RS232, RS485)	
Leuze electronic	RFM12 (SL200) (RS232) RFM32 (SL200) (RS232) RFM32ex (SL200) (RS232)	
Pepperl+Fuchs [from library v.1.1.0]	IDENTControl Compact (2 read heads) [IC-KP2-2HRX-2V1] (RS232) [from library v.1.1.0] IDENTControl (4 read heads) [IC-KP-R2-V1] (RS232) [from library v.1.2.8]	

Outdated firmware versions are sometimes not supported by the readers.

The following 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)

Further models by the above manufacturers that are not known to us are possibly supported by implication. According to Deister electronic, the same protocol (deBus) is implemented in other models. The use of these models may only be possible with limited functionality of the TcRFID library.

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

Note

Further RFID readers are supported with the TwinCAT PLC library Serial Communicationremoved link: TwinCAT PLC Library Serial Communication. This library makes it possible to exchange data bytes with any serial device. This alternative to the TwinCAT PLC 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.

Transponder types

A complete list of all supported transponder types can be taken from the manual of the respective RFID reader. The type of transponder that appears to be meaningful for the application is similarly to be clarified with the manufacturer of the RFID reader or transponder if necessary.

The TwinCAT library works with the data procured from the serial interface. The manufacturer's serial transmission protocol is therefore decisive for support by the PLC library. The radio frequency used, for example, is irrelevant.

The following table indicates by way of example which transponder types are supported for the respective RFID reader models according to the manufacturer. This list is not complete. The respective manufacturer of the RFID reader model has completed and more detailed information.
Note that some RFID readers accepts only 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 PLC library can be adapted by the user by using a special transponder. See the description of the structure ST_RFID_AccessData [▶ 49] regarding this.
Transponders up to a maximum size of 64 kilobytes are supported by the TcRFID library.

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)	EPCclass 1gen2, ...	ISO 18000-6, EPC-Gen2	< 10m	-	-	+	+	+	+
MW 2.45 GHz	...		< 12m	-	-	+ +	+ +	-	-

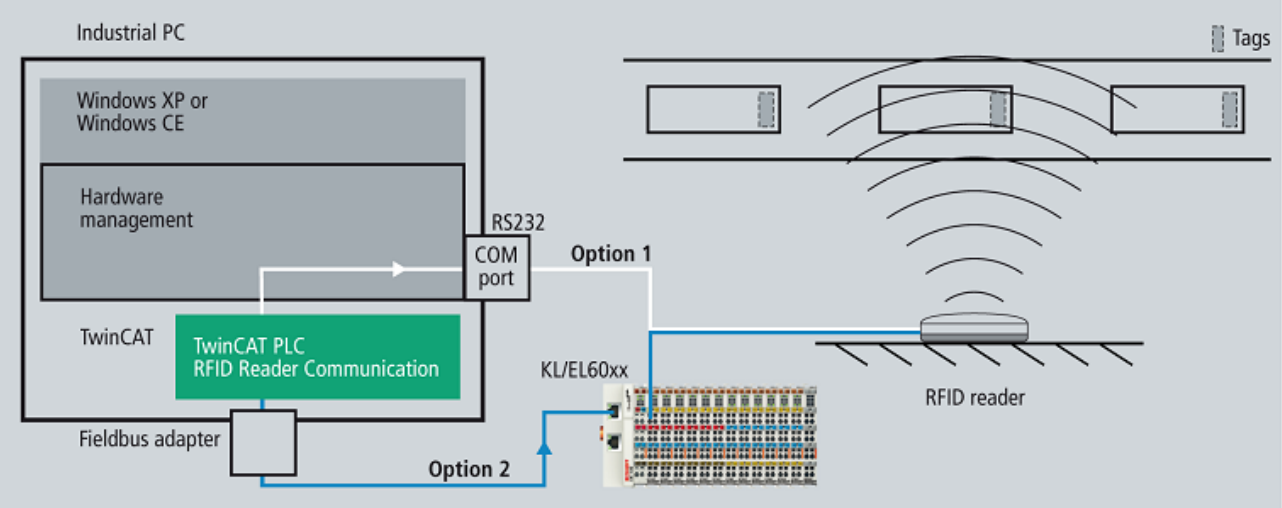
[++ very good; + good; - bad]

5 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.



Using multiple RFID readers

A separate connection must be made to a separate terminal for each RFID reader. The PLC RFID library does not support more than one RFID reader on one RS485 network for the time being.

Setting up serial communication under TwinCAT PLC Control

The serial data exchange is established with the blocks of the TwinCAT PLC library COMlibV2.

Create a send buffer and a receive buffer of type ComBuffer. This can take place globally but does not have to.

Additionally create two data structures, as used for serial communication in the TwinCAT system manager:

If the COM port is used, this looks as follows:

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

When using a serial terminal, EL6inData22B/EL6outData22B as well as KL6inData5B/KL6outData5B and other data types are possible in addition to PcComInData/PcComOutData. These structures are linked in the system manager with the channels of the serial interface.

In case of using the ComPort the SyncMode in the TcSystemManager must be activated.

In the TcSystemManager the Plc variables must be related to the right (fast) task and must be mapped. If several tasks are used and the PLC variables are in the wrong task, they can be moved to the right task by Drag & Drop.

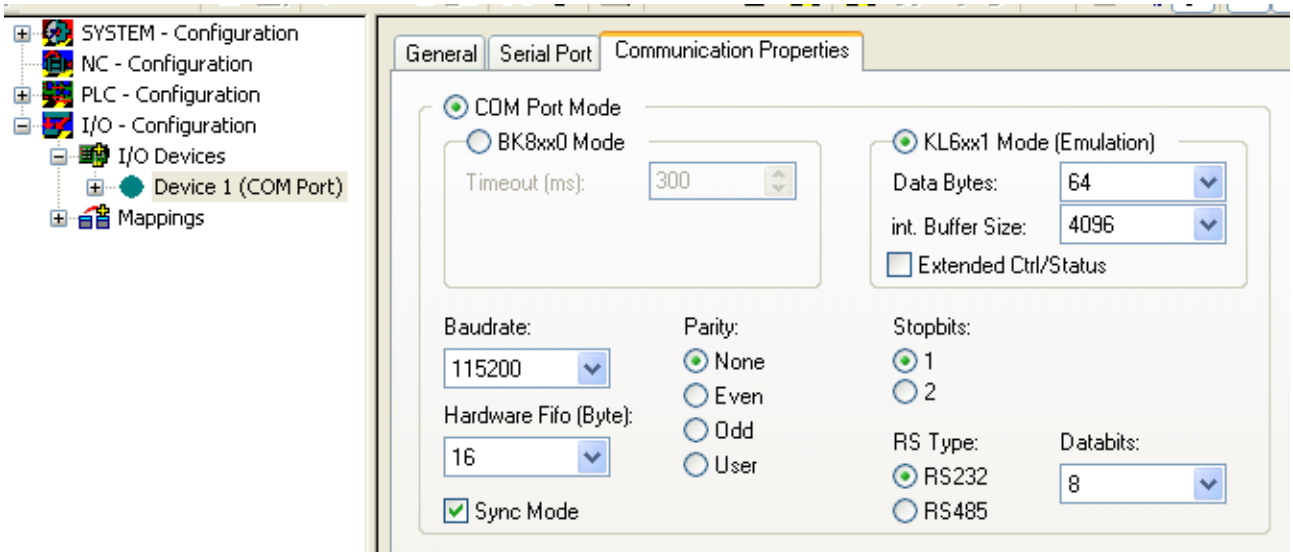
For serial communication, an instance of the 'SerialLineControl' must be created. This is called cyclically in a fast task (<= 1 ms). The required task cycle time depends on the application, the data amount, the baud rate, and the interface.

Depending on the application and the interface it is often recommended to process this in an additional task, which is faster than the task of the application.
Example 1: For communication with a COM port at a baud are of 115200 baud, a cycle time of 1 ms is required.

Example 2: For communication with a serial EL6001 terminal at a baud rate of 9600 baud, a cycle time of 6 ms is required.

Further information as well as explanation of virtual serial COM port usage are to be found in the Documentation for the PLC library serial communication.

The COM Port properties in the TwinCAT System Manager are represented by way of example below:



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(Kl6ComInData3),
    pComOut         := ADR(Kl6ComOutData3),
    SizeComIn       := SIZEOF(Kl6ComInData3),
    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
```

6 Command set

The following matrix lists the available command 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 means that it is inevitable that not every command or every detailed parameterisation option can also be given 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 on this can be found in the command description 'SendRawData' and in the chapter Low level communication [▶ 40].

Information on the characteristics and peculiarities of the proprietary protocols is to be procured for each model from the manufacturer and is usually provided 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 behaviour and characteristics. A detailed description of each command and the special behaviour 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 likewise be taken from the output structure ST_RFID_RawData [▶ 43] of the function block FB_RFIDReader.

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 [▶ 18]		x	x	x	x	x	x
GetConfig [▶ 18]			x	x		x	x
SetConfig [▶ 18]		x	x	x		x	
GetInventory [▶ 18]	x	x	x			x	x
Polling [▶ 19]			x	x	x		
TriggerOn [▶ 19]			x	x		x	
TriggerOff [▶ 19]			x	x		x	
AbortCommand [▶ 19]			x			x	x
ResetReader [▶ 19]	x	x	x	x	x	x	x
ReadBlock [▶ 20]	x	x	x	x		x	x
WriteBlock [▶ 20]	x	x	x	x		x	x
OutputOn [▶ 20]			x	x		x	
OutputOff [▶ 20]			x	x		x	
FieldOn [▶ 21]		x	x	x		x	

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
FieldOff [▶ 21]		x	x	x		x	
SendRawData [▶ 21]	x	x	x	x	x	x	x
ChangeDCType [▶ 21]							x

This list is analogous to the enumeration E_RFID_Command [▶ 58] in the RFID PLC library.
The successful processing of the requested command by the RFID reader can be recognised by the status outputs of the function block and the respective response. A list of possible responses can be viewed under E_RFID_Response [▶ 59].

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 described under ST_RFID_Config [▶ 49].

Further information is available in the chapter Configuration [▶ 38].

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

SetConfig [16#03]

Parameterized configuration settings can be transferred to the RFID reader. More detailed information on the possible configuration of the RFID reader can be found under ST_RFID_ConfigIn [▶ 48].

Following a configuration command it is recommended to query the current configuration of the reader once again using the GetConfig command.
Further information is available in the chapter Configuration [▶ 38].

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. An appropriate response follows if no transponder is found.

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]

Information is queried from the stack of the RFID reader. This may be, for example, the serial number of the last transponder. Please note that different RFID readers have differently sized stacks and that sometimes only one message is stored.

Presence detection

If this cannot be adjusted via a configuration parameter, then it is necessary to keep the reader in readiness for reading by means of cyclic polling command, so that a transponder in range is detected automatically.

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 response telegram that follows is received by the function block of the Tc RFID library. There is no assignment of read transponder data in this case. The received raw data can be taken from the block interface for further processing.

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

TriggerOff [16#07]

See TriggerOn.

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

AbortCommand [16#08]

If a command is currently being executed by the RFID reader, it can be 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 [▶ 49] is necessary for this command.
Before reading data from the transponder, it is usual to recognize and select the transponder (see command GetInventory).

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 [▶ 49] is necessary for this command.
Before writing data to a transponder, it is usual to recognize and select the transponder (see command GetInventory).

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 addressed automatically by means of 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 addressed automatically by means of 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 [16#0F]

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]

This command enables the RFID function block to 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 after that are available at the output of the function block in the raw data structure [▶ 43] as an addressed data field. Further information on the sequence is available in the chapter Low level communication [▶ 40].

When using the SendRawData command, 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 [▶ 49]. An evaluation/storage of the data should therefore also take place manually with the aid of the raw data structure [▶ 43], which is always specified.

ChangeDCType [16#11]

The 'ChangeDCType' 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'

7 RFID reader settings and handling

This chapter gives important information about each individual RFID reader model.

For all devices the necessary setting and the specific handling is described.

The information is separated by the respective device manufacturers:

7.1 Balluff RFID reader

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.

This default setting for data transmission has proven itself for all supported RFID reader models:

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

If required, other parameters can be set depending on the hardware or the factory settings of the RFID reader can be used. These must then also be adopted into the reader connection on the software side [▶ 14].

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	disabled (or enabled – however, no evaluation takes place)
Dynamic operation	disabled
Send power-on message	disabled (or enabled – however, no evaluation takes place)
CRC16 data check	disabled
Type and serial number with CT pres.	disabled (or enabled, as required)

If 'Type and serial number if CT pres.' is activated, then 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 recommended to query existing transponders manually using the 'GetInventory' 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.

Handling of the RFID reader

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 (see input structure stCtrl [▶ 44]) must be set.
- When the 'Get Inventory' command is called, information from both read heads is returned via the serial interface. However, only the information from the read head selected via *stCtrl.iHeadNumber* is evaluated and output at the *stTranspInfo* output.
- If ' Type and serial number at 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).

Not all peculiarities of each supported RFID reader model can be mentioned here. Therefore you are referred to the manufacturer’s own documentation for more detailed information.

7.2 Baltech RFID reader

If a supported Baltech RFID stand-alone device is used, the TwinCAT PLC library can be used as interface.

Alternatively certain Beckhoff Control Panels or Panel PCs can be used. In these devices a 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 be used to carry out a function test to check whether the RFID device is recognized and whether the transponder cards are detected.

The following default settings are recommended for the data transmission:

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 into the reader connection on the software side [▶ 14].
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 an RFID device to a COM port and a baud rate of 115200 baud, a cycle time of 1 ms is required. (see chapter Reader connection).

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]. Further information can be found in section "Low level communication". This is also possible under Windows CE, if a transfer with the currently set baud rate is possible.

Usage of virtual serial COM ports (USB)

If the device is connected via USB, the corresponding Usb-To-Virtual-Com-Port driver must be installed. This driver is already installed if it is about a Beckhoff Panel-PC. The SDK of the RFID device contains the driver too. The virtual COM port is displayed in the Windows device manager.

The Beckhoff TwinCAT 'Serial Communication' is used for the communication to driver. But neither a device is added to TwinCAT System Manager nor a mapping is done. Further information is to be found in the Documentation for the PLC library serial Communication.

Handling of the RFID reader

The library supports the default settings for the Baltech communication. 'Host Operation' mode is supported. Other modes are not supported. The BRP (Baltech Reader Protocol) is used for internal access in 'Communication Mode' 'Normal Mode'. If raw data are sent via the low level communication option it is important to ensure that the above settings are correctly specified within the frame.

Not all peculiarities of each supported RFID reader model can be mentioned 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 line with the encryption of a transponder card a VHL file is stored in the device configuration. Several VHL files can be stored in order to be able to access different cards access without reconfiguration.

There are three options for transferring VHL file into the configuration of a RFID device.

Configuration type	Description
Configuration card	A configuration can be transferred via a configuration card. This is the preferred option.

Configuration type	Description
'Baltech id-engine explorer' tool	This tool can be used to transfer a configuration into the memory of the Baltech RFID device. (see Baltech documentation: IdEngineExplorer.pdf) The specific configuration can be created directly in the tool in simple cases. Alternative Baltech offers technical support and can provide a file containing the configuration. **The tool 'Baltech id-engine explorer' only runs under 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 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 [▶ 50]. (More detailed information about Mifare cards can be found in the Baltech document entitled Mifare.pdf)

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

7.3 Deister RFID reader

Settings of the RFID reader

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.

This standard setting for data communication has proven itself for all supported RFID reader models:

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 adjusted if necessary or the default settings of the RFID reader can be used. These must then also be adopted into the reader connection on the software side [▶ 14].

The following special settings may need to be parameterised using the proprietary tools before the system startup.

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

Handling of the RFID reader

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 [▶ 17]).
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 for 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.
The Deister RDL devices support 4 bytes or 8 bytes block size.
Example: if a block size of 8 bytes is specified for the transponder, the reader must be configured with the parameter *iBlocksize:=8* and the read or write access via the structure ST_RFID_AccessData [▶ 49] must occur with 8 bytes 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.

Not all peculiarities of each supported RFID reader model can be mentioned here. Therefore you are referred to the manufacturer's own documentation for more detailed information.

Configuration

Is a new configuration written to the RFID device (command 'SetConfig') the following must be attended

Deister RDL devices:
Some combinations of configuration parameters (structure ST_RFID_CfgStruct_DeisterRDL [▶ 52]) are invalid. Disregard of these dependencies leads to an error (*eRFERR_InvalidCfg*).

configuration parameter	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

These dependencies also exist if the configuration is transferred as register. The RFID device will return an error code in case of conflicts.

7.4 Leuze RFID reader

Settings of the RFID reader

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 to transfer these settings to the RFID.

This standard setting for data communication has proven itself for all supported RFID reader models:

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 adjusted if necessary or the default settings of the RFID reader can be used. These must then also be adopted into the reader connection on the software side [▶ 14].

The following special settings may need to be parameterised using the proprietary tools before the system startup.

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

If the RFID reader should be activated by means of a trigger, then the ensuing response telegram is received by the function block of the Tc RFID library. There is no assignment of read transponder data in this case. The received raw data can be taken from the block interface for further processing.

Handling of the RFID reader

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 [▶ 49] 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.

Not all peculiarities of each supported RFID reader model can be mentioned here. Therefore you are referred to the manufacturer's own documentation for more detailed information.

Configuration

Is a new configuration written to the RFID device (command 'SetConfig') the following has to be attended:

Some combinations of configuration parameters (structure ST_RFID_CfgStruct_LeuzeRFM [▶ 55]) are invalid. Disregard of these dependencies leads to an error (*eRFERR_InvalidCfg*).

configuration parameter	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

These dependencies also exist if the configuration is transferred as register. The RFID device will return an error code in case of conflicts.

7.5 Pepperl+Fuchs RFID reader

Settings of the RFID reader

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 to transfer these settings to the RFID.

This standard setting for data communication has proven itself for all supported RFID reader models:

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 adjusted if necessary or the default settings of the RFID reader (38400 Baud) can be used. These must then also be adopted into the reader connection on the software side [▶ 14].

Handling of the RFID reader

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

The received device status is displayed via the *iErrCodeRcv* output of the *FB_RFIDReader* function block and is 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 [▶ 57] read via 'GetConfig'.

The set transponder types should be checked on restarting. If the configuration structure read via 'GetConfig' does not show the correct transponder types for each read head, they can be changed with the 'ChangeDCType' command. It is recommended to set the value specified for the transponder in place of the default value (99).

For read as well as write access to the data memory of a transponder, a block size of 4 bytes (see ST_RFID_AccessData [▶ 49]) must be used for all Pepperl+Fuchs RFID devices.

Not all peculiarities of each supported RFID reader model can be mentioned here. Therefore you are referred to the manufacturer's own documentation for more detailed information.

Buffered Command

The input variable *bBufferedCmd* in ST_RFID_Control [▶ 44] can be used to send commands that are buffered for later permanent execution.
This is possible with the commands *eRFC_GetInventory*, *eRFC_ReadBlock* and *eRFC_WriteBlock*.
A buffered command can be terminated with the command *eRFC_AbortCommand*.

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 therefore be disabled for all channels.
By factory setting of the RFID device the trigger is disabled on 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*).

If a trigger is required as a sensor channel on the RFID unit, the TwinCAT library provides the following option:
The trigger provides a message whether it is currently triggered or whether the trigger range is exited. These messages are received and displayed as eResponse = eRFR_CmdConfirmation or eRFR_NoTransponder. The application can react to this and trigger the desired command.
To configure a channel accordingly as a sensor channel/trigger, the associated ident channel must = 0. The required raw data command is explained in the next paragraph.

The trigger setting must not be made from the manufacturer's own tool. Otherwise incoming messages from the sensor channel cannot be read by the TwinCAT library block.

The correct setting of the trigger mode should be checked with the command 'GetConfig' and by analyzing 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 sectionLowLevelKommunikation [▶ 40]and the description of the structuresST_RFID_Control [▶ 44]andST_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_CmdConfirmationwhenthe sensor is triggered.
Answer:eResponse = eRFR_NoTransponderwhen the sensor is exited.
At the output stTranspInfo.iHeadNumberthe corresponding sensor channel is shown.

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.
At the output stTranspInfo.iHeadNumberthe corresponding sensor channel is shown.

Once such setting a setting has been made, the device configuration must be re-read with the command 'Get Config'. It is advisable to check the settings by analysing the read configuration structure.

8 Tutorial

This tutorial is a guide to putting an RFID reader into operation with the aid of the TwinCAT PLC.
The following steps are thereby performed:

1. Glossary

2. Installation/libraries

3. Serial connection

4. Function block declaration

5. Function block usage

6. Test

It is assumed that you have planned your RFID application in detail and that your application has already been modeled.
Based on this, you have determined that a reader supported by the TcRFID library basically meets your requirements and you can implement your application with the commands offered. Therefore, you have decided to use the TwinCAT library in order to facilitate communication with your RFID reader.

We will work through this tutorial with the RFID reader model Balluff Bis M 401. In principle, however, the same procedure can be adopted for other models.

next tutorial page [▶ 30]

8.1 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

nächste Tutorial-Seite [▶ 30]

8.2 Installation/libraries

- Start TwinCAT PLC Control

- Create a new PLC project with 'File > New'.

- Select your target platform PC and CX (x86) or CX (ARM)

- Open the 'Resources' tab and the 'Library Manager'

- Add the library **TcRFID.lib** with 'Add > further libraries'

Now all necessary PLC blocks are available to you for the RFID reader communication. All further implicitly required libraries have been automatically integrated with the TcRFID.lib.

nächste Tutorial-Seite [▶ 31]

8.3 Serial connection

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

Variables:

Create a send buffer and a receive buffer (gEL6ComTxBuffer, gEL6ComRxBuffer) of type ComBuffer.

Additionally create two data structures, as used for serial communication in the TwinCAT system manager:

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

Link these structures with the channels of the serial port in the system manager.

For serial communication, create an instance of the 'SerialLineControl'. Call this cyclically in a fast task.

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

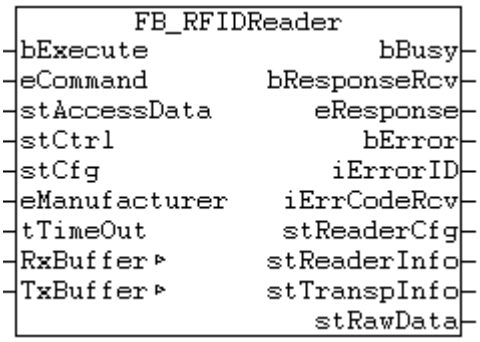
Mode: As a handle, specify the serial EtherCAT terminal with 22 bytes of user data in our example.

Further instructions can be found in the chapter Serial RFID reader connection [▶ 14].

nächste Tutorial-Seite [▶ 31]

8.4 Block declaration

The function block FB_RFIDReader is the core of the entire RFID reader communication. The declaration and initialisation of this block are described in the following chapter.



Create an instance of the FB_RFIDReader.

Transfer the manufacturer of your RFID model to it at the *eManufacturer* input.

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

The FB_RFIDReader has 7 inputs (6 in the case of the specific FBs due to the missing manufacturer data), 2 input/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. To do this, call the 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 failure analyzes it's necessary to process the outputs bError and iErrorID etc. too.

nächste Tutorial-Seite [▶ 32]

8.5 Block usage

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     := stTranspInfo.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*.
With the aid of a trigger, further data are only evaluated if a new message is received.
Your string variable *sTranspSerialNumber* now always shows 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 are available at 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.

nächste Tutorial-Seite [▶ 33]

8.6 Test

As soon as you have created the program in accordance with the preceding chapters, you can now activate the current configuration with the linked variables in the System Manager and place TwinCAT in RunMode. This is followed by the login on the part of TwinCAT PLC Control and the start of 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.

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 chapter RFID Reader Settings - Balluff [▶ 22]. Not every RFID reader supports this setting.

You can download the complete project as well: https://infosys.beckhoff.com/content/1033/tcplclibrfid/Resources/11421957643/.zip

Further examples can be found in the chapter Examples [▶ 66].

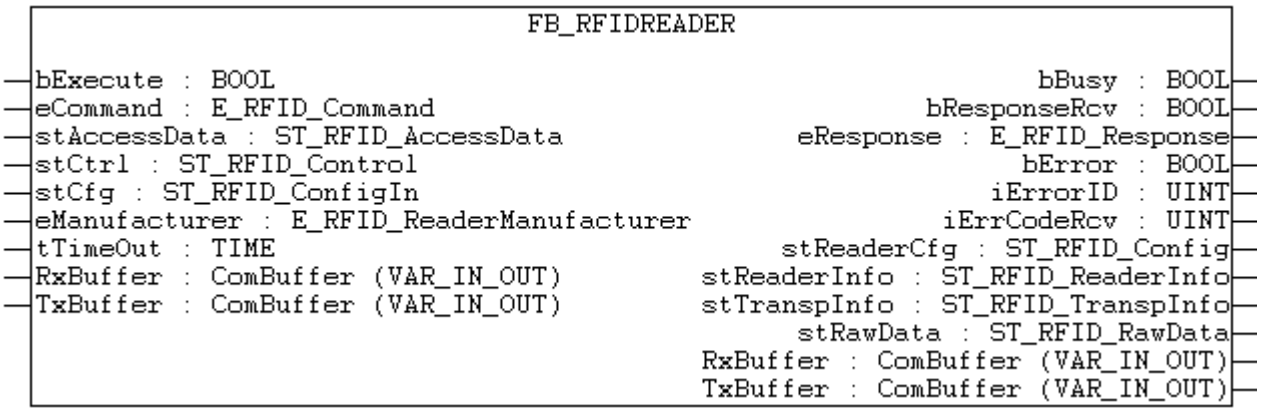
9 FB_RFIDReader

The TwinCAT RFID library consists of only one function block.

Its interface is to a large extent self-descriptive. For a fast introduction to the handling of the library, there now follows a description of the interface variables.
You are also referred to the Tutorial [▶ 30] and the examples.

The uniform handling of all RFID reader models and the associated prepared interface declarations are particularly user-friendly.

However, it should be pointed out that the function block of the RFID library possesses a slight overheat due to the differences between some RFID readers models. This indispensable characteristic is, however, strongly outweighed by the advantages that the available flexibility offers.



Input variables

```
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
```

bExecute

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

The *eCommand* input offers a choice of commands, such as reading from or writing to a transponder, in the form of an enumeration. See description of the command set [▶ 17]. A command is executed by setting the *bExecute* input.

stAccessData

If a write or read command is to be executed, then parameters must be transferred with this input structure. Details for this structure: ST_RFID_AccessData [▶ 49]

stCtrl

Various control parameters can be transferred at the input with *stCtrl*. This includes the possibility to specify delay times. Details for this structure: ST_RFID_Control [▶ 44]

stCfg

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 description of the configuration options [▶ 38].
Details for this structure: ST_RFID_ConfigIn [▶ 48]

eManufacturer

The manufacturer of the RFID reader model in use is specified at this input. Details for this enumeration: E_RFID_ReaderManufacturer [▶ 61]

tTimeOut

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].

Input/output variables

```
VAR_IN_OUT
    RxBuffer            : ComBuffer;
    TxBuffer            : ComBuffer;
END_VAR
```

RxBuffer

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 [▶ 14] regarding this.

TxBuffer

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 [▶ 14] regarding this.

Output variables

```
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
```

bBusy

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. Hence, if *bBusy* = FALSE is recognized, all further output variables bResponseRcv, eResponse, bError, iErrCodeRcv, etc. and stRawData can in turn 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 if the sum of these times. See ST_RFID_Control [▶ 44] regarding this configuration setting.

bResponseRcv

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 indicated by a rising edge to *bResponseRcv* = TRUE. Hence, if this flag is recognized, unexpected telegrams and its output variables eResponse, bError, iErrCodeRcv, etc. and stRawData can in turn 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. Depending on the executed command call, more than one response can be received before the action is terminated and *bBusy* is 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

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.
Details for this enumeration: E_RFID_Response [▶ 59]

bError

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 following output variable *iErrorID*. Details of error representation are given in the chapter Error codes [▶ 63].

iErrorID

If an error occurs, the type of this error is indicated at the *iErrorID* output.
Details of the possible ErrorIDs are given in the chapter Error codes [▶ 63].

iErrCodeRcv

The error code indicated at the iErrCodeRcv output corresponds to the error code sent by the RFID reader to the controller.
Details of error representation are given in the chapter Error codes [▶ 63].

stReaderCfg

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. Details for this structure: ST_RFID_Config [▶ 49]

stReaderInfo

Each RFID reader has its own characteristic data, such as designation, hardware version etc. These values, which can be queried among other things using the 'GetReaderVersion' command, are indicated in the output structure *stReaderInfo*. Details for this structure: ST_RFID_ReaderInfo [▶ 42]

stTranspInfo

The structure *stTranspInfo* contains information on the last read transponder. Among other things, the serial number of the transponder is output here. Details for this structure: ST_RFID_TranspInfo [▶ 42]

stRawData

The output structure *stRawData* outputs both the sent and the received raw data. Details for this structure: ST_RFID_RawData [▶ 43]

9.1 Handling instructions

Handling the RFID library

If you have integrated the library file TcRFID.lib, you can access all functions. The library provides a function block for communication with an RFID reader.
The general function block FB_RFIDReader can be used, 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 command set [▶ 17] provides a wide variety of commands.
In addition, the integrated low-level communication [▶ 40] enables the sending and receiving of raw data.

The TcRFID library expects the RFID reader to respond immediately to a command and that the dialogue 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 command set [▶ 17]. The following procedure is common to all commands. The function block is called by a positive 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). Further handling instructions are supplied by the description of the input and output variables [▶ 35] of the function block as well as the tutorial/example in this documentation. 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.

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 command set, the model information ('GetReaderVersion' command) and the current reader configuration ('GetConfig' 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.

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

Handling of the RFID reader

In the chapter 'RFID Reader settings and handling' characteristics of the supported RFID reader models are pointed out.
The instructions listed there are assigned to the special RFID reader manufacturers:

9.2 Configuration

All supported RFID readers can be configured with the same command. This must be available in the command set [▶ 17] for the special reader model.

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

Because the RFID readers from different manufacturers never have identical configuration options, the Plc RFID library offers a substructure for each manufacturer with the specific parameters, in addition to the input configuration structure. (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' command from the command set [▶ 17] is used 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' command from the command set [▶ 17] is used in order to write an RFID reader configuration.

Following a 'SetConfig' command, the current configuration must be read once with the 'GetConfig' 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.

Note

Certain combinations of configuration parameters are sometimes impermissible. Refer to the RFID reader manufacturer's proprietary protocol specifications for details of which parameter values are excluded in which combinations. 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 a register (byte array) or a structure. This does not concern the parameterization of the PLC RFID library, but rather the proprietary configuration of the RFID reader. Hence, there are various configuration structures in the PLC RFID library, which process the raw data from the configuration registers of different RFID readers. Both variants are made available at the output ST_RFID_Config [▶ 49] of the library function block. This takes place via pointers.

Baltech

The configuration data are used as structure for Baltech RFID readers.

ST_RFID_CfgStruct_BaltechMifVHLFile [▶ 50]

This structure is suitable for the writing with the command eRFC_SetConfig. (see command set [▶ 17])

Balluff

There is no possibility for configuration supported.

Deister

The configuration data can be handled as structure or as register for Deister RFID readers.
If a register (byte array) is used it always has to be defined with the size of the whole configuration data. For the supported Deister RDL devices this is 88 bytes and for the UDL devices 117 bytes.

ST_RFID_CfgStruct_DeisterRDL [▶ 52]
ST_RFID_CfgStruct_DeisterUDL [▶ 54]

The structures are suitable for writing with eRFC_SetConfig and reading with eRFC_GetConfig. (see command set [▶ 17])

Leuze

The configuration data can be handled as structure or as register for Leuze RFID readers.
If a register (byte array) is used it always has to be defined with the size of the whole configuration data. For the supported Leuze devices this is 88 bytes.

ST_RFID_CfgStruct_LeuzeRFM [▶ 55]

The structure is suitable for writing with eRFC_SetConfig and reading with eRFC_GetConfig. (see command set [▶ 17])

Pepperl+Fuchs

The configuration data are used as structure for Pepperl+Fuchs RFID readers.

ST_RFID_CfgStruct_PepperlFuchsIDENT [▶ 57]

The structure is suitable for reading with eRFC_GetConfig. (see command set [▶ 17])

9.3 Low level communication

In addition to the high level command set [▶ 17], the PLC RFID library also offers the possibility of low level communication. This is solved implicitly. The same function block is used.
Arbitrary telegrams can be received (up to a maximum size of 1024 bytes) and sent (up to a maximum 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

For this purpose the command eRFC_SendRawData from the command set is used and the raw data to be sent is specified in the input structure ST_RFID_Control [▶ 44].
To send a low level telegram *only* the raw data are specified. The other components of the telegram are supplemented automatically by the PLC RFID library. Checking data such as CRC are likewise generated and added internally.

If the protocol demands the re-encoding of certain bytes within the raw data, this is similarly performed automatically by the PLC RFID library and does not have to be considered.

If an RS485 interface is concerned, then 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 last raw data received can be viewed at any time at the output of the function block using the structure ST_RFID_RawData [▶ 43].
The associated addressing is output in the structure ST_RFID_ReaderInfo [▶ 42].

When using the SendRawData command, 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 [▶ 49]. The evaluation/storage of the data should therefore also take place manually with the aid of the raw data structure ST_RFID_RawData [▶ 43], which is always specified. The specified raw data is the received telegram, but without prefix, suffix, checksum, CRC or shift sequence coding.
If the received telegram was not evaluated regularly by the function block, this is also 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.iRecei
vedRspLen, 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.

10 Data types

10.1 Structures

10.1.1 ST_RFID_TranspInfo

```
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 *)
END_STRUCT
END_TYPE
```

The structure ST_RFID_TranspInfo indicates the type [▶ 61] and the serial number [▶ 61] of the last detected RFID transponder.

With the command 'GetInventory' this information will be requested and updated.

All manufacturers:

sSerialNumber	The serial number of the transponder (frequently 8 bytes) is indicated in the hexadecimal string *sSerialNumber*. **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.

Pepperl+Fuchs:

iHeadNumber	If a RFID reader with multiple read heads is connected, the head number where the RFID transponder was detected is output here.

10.1.2 ST_RFID_ReaderInfo

```
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
```

Following the GetReaderVersion command from the command set [▶ 17], 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. Hence, for example, one reader model returns the production date, whilst another reader model transmits the hardware version.
More detailed information on the entries can be found in the manufacturer's own protocol specification and manuals.

all manufacturers:

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:

dDate	The production date of the RFID reader is indicated at this output. **The date 1970-01-01 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 *sCode* output 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:

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 *sCode* output 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:

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

Pepperl+Fuchs:

dDate	The production date of the RFID reader is indicated at this output. **The date 1970-01-01 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 *sCode* output as a string.

10.1.3 ST_RFID_RawData

```
TYPE ST_RFID_RawData :
STRUCT
    pReceivedRsp        :DWORD;
    pSentCommand        :DWORD;

    iReceivedRspLen     :UINT;
```

```
    iSentCommandLen     :UINT;
END_STRUCT
END_TYPE
```

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 more detail in chapter Low level communication [▶ 40].

The byte sequences can be viewed via the specified pointers.

The MEMCPY function can be used for evaluation.

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.

10.1.4 ST_RFID_Control

```
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 addres
ses are handled automatically and set addresses are not used. *)

    bLogging            :BOOL;

    iHeadNumber         :USINT  := 1;       (* if multiple read heads are installed at the reader un
it, one can be selected *)
    iDCType             :USINT  := 1;

    pRawDataCommand     :DWORD;             (* data input for low level communication *)
    iRawDataCommandLen  :UINT;

    bBufferedCmd        :BOOL;

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

    iCardTypeMask       :UINT   := 16#FFFF;     (* select which card type should be detected via Get
Inventory; default 0xFFFF (all types) *)
    bReselect           :BOOL   := TRUE;        (* with Reselect every GetInventory gets the first i
tem of the reader's detected card list *)
    bAcceptConfCard     :BOOL   := TRUE;        (* a read command also accept configuration cards to
 configure the rfid reader *)
END_STRUCT
END_TYPE
```

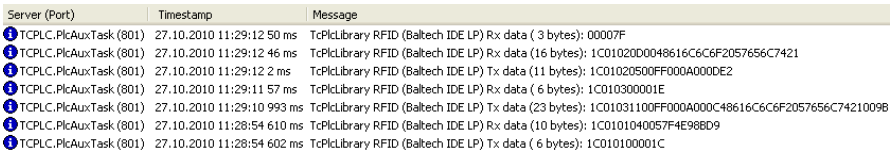
The input structure *stCtrl* contains variables, such as the serial number [▶ 61], with which the behaviour of the function block can be parameterised.

Optional the two variables *tPreSendDelay* and *tPostSendDelay* offer the possibility to parameterize delay times. Further information at the ende of this page.

RS485

The PLC RFID library does not support more than one RFID reader on one RS485 network for the time being. A separate connection must be made to a separate terminal for each RFID reader. Individual addressing with iSrcAddrSnd and iDstAddrSnd is not necessary in this stand-alone operation. Accordingly, the addressing can be performed automatically by the PLC RFID library, to which end the input variable bAddrAutoMode can be left at TRUE.

all manufacturers:

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 by then, 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. The RS485 addresses are assigned automatically if *bAddrAutoMode* is set (default = TRUE). The addresses given above are not used. If *bAddrAutoMode* is deactivated (FALSE), then the addresses given 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 test and analysis purposes. The output format is not defined to allow enhancements.  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 message 'Logging initialized.' is output. In this first cycle no communication data is monitored.
pRawDataCommand	The regular function block of the PLC RFID library can similarly 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 TcRFID library only accesses this address for reading. Further information on the low-level communication sequence is summarized in the chapter 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:

iHeadNumber	If a RFID reader with multiple read heads is connected, the choice for a specific read head is done in the input variable *iHeadNumber*.
iDCType	With *iDCType* the memory size (0->64bytes, 1->32bytes) of the chip can be set for Balluff readers.

Baltech:

iVHLFileID	Read und Write commands will be executed using the VHL-file specified in *iVHLFileID*. This file has to be saved in the reader's configuration. If *iVHLFileID* is 0xFF [default] the reader will not use a normal VHL-file stored in the reader's configuration but a simple ad hoc VHL-file which can be used to access blank non configured/encoded cards.		
iCardTypeMask [since library v.1.2.4]	The *iCardTypeMask* instructs the RFID Reader to select only the parametrized specific card families. Only the parametrized card families will be detected with the command GetInventory. All other card types will be filtered out. Keep the default value [default: 0xFFFF] if every card type should be available. For each card family the corresponding bit has to be set in *iCardTypeMask*. see "Extracted nested table 1"		
	Card Type	**Card Family**	**CardTypeMask**
	Mifare / ISO 14443-A	1	0x0001 (Bit 1)
	LEGIC	2	0x0002 (Bit 2)
	ISO 15693	3	0x0004 (Bit 3)
	ISO 14443-B	4	0x0008 (Bit 4)
	PICOPASS via ISO 14443-2-B	5	0x0010 (Bit 5)
	PICOPASS via ISO 15693	6	0x0020 (Bit 6)
bReselect [since library v.1.2.4]	If *bReselect* is set [default: TRUE] every call of GetInventory outputs a transponder type if a transponder is within the HF field. If it's sure that only one card is within the HF field this setting should be kept. If there are more than one card within the HF field a list of detected cards exists in the RFID device. To enable the detection of all cards by the command GetInventory the input variable *bReselect* has to be switched off [FALSE]. The command GetInventory works them off, one by one. After all cards have been selected the response is *eRFR_NoTransponder*. Only if a card is moved out of and again into the HF field the command GetInventory outputs its card type and serial number again.		
bAcceptConfCard [since library v.1.2.4]	If *bAcceptConfCard* is set [default: TRUE], also configuration cards are detected with the command GetInventory. The RFID Reader tries to adopt automatically the configuration. To avoid an influence by unknown configuration cards the variable can be deactivated.		

Leuze:

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*. 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:

iHeadNumber	If an RFID reader with several read heads is addressed, a choice of read head is specified in the input variable *iHeadNumber*. *iHeadNumber* is also referred to as IdentChannel.			
iDCType	With this variable and the command 'ChangeDCType' 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 E_RFID_TranspType enumeration) Extract of supported transponder types: see "Extracted nested table 2"			
	Data Carrier Type [dec]	**Description P+F**	**Chip Type**	**Frequency**
	02	IPC02	Unique, EM4102	125 KHz
	03	IPC03	EM4450	125 KHz

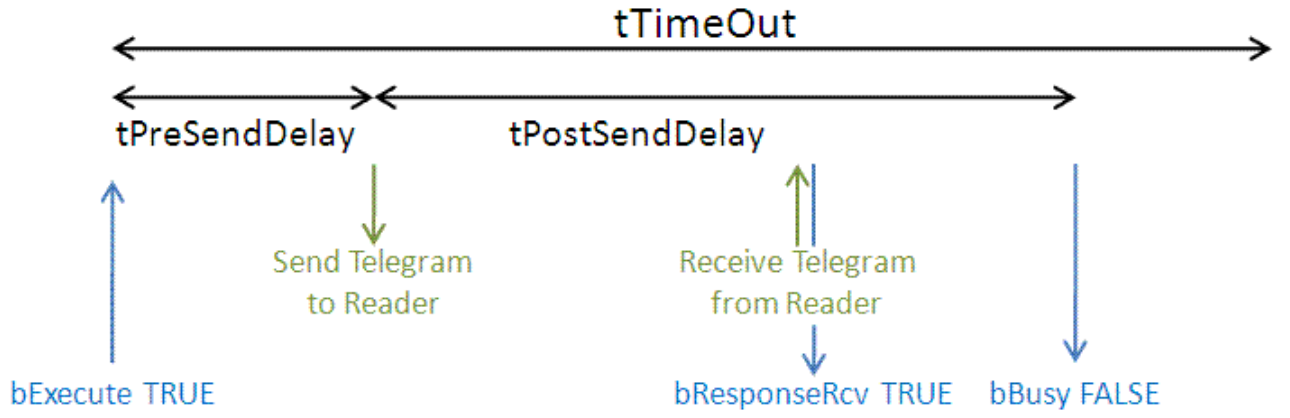
	14	IPC14	T5557 (Atmel)	125 KHz
	20		all ISO 15693 complaint data carriers	13.56 MHz
	21	IQC21	I-Code SLI (NXP)	13.56 MHz
	22	IQC22	Tag-it HF-I Plus (TI)	13.56 MHz
	23	IQC23	my-D (infineon) SRF55V02P	13.56 MHz
	24	IQC24	my-D (infineon) SRF55V10P	13.56 MHz
	33	IQC33	Fujitsu FRAM MB89R118	13.56 MHz
	35	IQC35	I-Code SLI-S (NXP)	13.56 MHz
	...	...	...	...
	99		default type of the connected read head	
bBufferedCmd	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!**			

Delay times

The two variables *tPreSendDelay* and *tPostSendDelay* offer the option of parameterizing delay times. Both variables ensure that a delay is waited for 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.



10.1.5 ST_RFID_ConfigIn

```
TYPE ST_RFID_ConfigIn :
(* defines the configuration input parameters.
Tha data can be set via Config structure or Config register.
Different RFID Reader in different ReaderGroups can differ in their configuration data. *)
STRUCT
    pCfg            : DWORD;            (* 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 CfgStructu
re *)

    (*      An additional option to demand/
set a specific config parameter without transmission of the whole config register. Not possible at a
ll 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
```

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.

Further information on RFID reader configuration is summarized in the Configuration chapter [▶ 38].

Baltech:

pCfg	This pointer must contain the memory address of the configuration to be written. This must be 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:

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:

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.

The following configuration variant is available only for Leuze electronic RFID readers.

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 a single register.
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*.

10.1.6 ST_RFID_Config

```
TYPE ST_RFID_Config :
(* defines the configuration as structure and register.
Different RFID Reader in different ReaderGroups can differ in their configuration data. *)
STRUCT
    pCfgStruct      : DWORD;        (* pointer to config structure *)
    pCfgReg         : DWORD;        (* 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
```

The structure indicates the RFID reader configuration that was last read. This does not concern the parameterisation of the PLC RFID library, but rather the proprietary configuration of the RFID reader. This can be queried with the command *eRFC_GetConfig* (see command set [▶ 17]).

Each configuration can be a register (byte array) or a structure. Hence, there are various configuration structures in the PLC 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.

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 are not available as register (byte array).

Further information on RFID reader configuration is summarised in the Configuration chapter [▶ 38].

10.1.7 ST_RFID_AccessData

```
TYPE ST_RFID_AccessData :
STRUCT
    (* access specific parameters *)
    pData               : DWORD;        (* 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 st
arts with block index 0 or higher *)
    (*      The upper parameter iStartBlock depends on the iUserDataStartBlock. The used StartBlock
is iStartBlock+iUserDataStartBlock.     *)
    (*      Different RFID Readers can differ in their interpretation of the first block.   *)
    iReserved           : UINT;
END_STRUCT
END_TYPE
```

If a read or write command is to be executed, it is necessary to specify the input structure stAccessData. This structure is used to specify how many and which data are to be read and where they are to be stored, or how many and which data are to be written.

pData	The pointer *pData* points to the data to be written or to the free storage space 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.
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.

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 one reader model stores the data blocks in a reversed byte order on the transponder 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 RFID library does not affect this. The user must select the above input parameters accordingly.

The possible block size depends on the transponder type. 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.
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.

iUserDataStartBlock	With the variable *iUserDataStartBlock* the start block (as index of blocks) of the user data can be specified on the transponder. **Note the block size.**

10.1.8 Configuration data

10.1.8.1 ST_RFID_CfgStruct_BaltechMifVHLFile

[available from library v.1.2.4]

```
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
```

The structure is suitable for the writing with the command eRFC_SetConfig. (see command set [▶ 17])
This does not concern the parameterization of the PLC RFID library, but rather the proprietary configuration of the RFID reader.

Structure of a Mifare card (up to 2 KB memory):
a Mifare card with 1 KB memory has 16 sectors of 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 with the asterisk symbol (*) contain usable memory area for the user. The maximum size of the user data is therefore 752 bytes (47 x 16 byte) for a 1024-byte Mifare card.

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, the respective keys are stored in arrKeyList[0] and arrLeyList[1]. TYPE T_RFID_MifareKey : ARRAY[0..5] OF BYTE; END_TYPE
arrSectorList	In the array *arrSectorList* all sectors are stored which are to be used for user data. Example: if only sectors 4-6 are to be 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 are to 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); must therefore be stored. **If a single key is to be used for all sectors, it is the key index 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 are to 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]). The array *arrWrKeyAssign* must therefore contain the indices stCfg:ST_RFID_CfgStruct_BaltechMifVHLFile:=(arrWrKeyAssign:=0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1);. **If one key is to be used for all sectors, it is the key index 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 regarding the RFID reader configuration can be found in chapter Configuration [▶ 38].

More detailed information on VHL files and configuration of Baltech RFID devices can be found in the documentation provided by the manufacturer (files Mifare.pdf and ConfigurationValues.pdf).

10.1.8.2 ST_RFID_CfgStruct_DeisterRDL

The structure is suitable for writing with eRFC_SetConfig and reading with eRFC_GetConfig. (see command set [▶ 17])
This does not concern the parameterization of the PLC RFID library, but rather 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 extentio
n *)
    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
```

eOpMode	This operation 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 answer telegram that follows is received by the function block of the Tc 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 (* This option is not possible at all RFID Reader types. *));END_TYPE

eTriggerMode	If eRFTR_ImmediateRead is set, the device is always ready to read. This setting means that the trigger condition always applies. [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 purpose. (see command set [▶ 17]) **The answer telegram that follows is received by the function block of the Tc RFID library. There is no assignment of read transponder data in this case. The received raw data can be taken from the module 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 also transferred with write and read commands. By default, this corresponds to the last transponder serial number detected with the GetInventory command. Otherwise, the transponder serial number is specified in ST_RFID_Control [▶ 44].
bMultiTranspMode	If *bMultiTranspMode* is TRUE, anti-collision is active if several transponders are in the field.
bOutputAutomatic	If *bOutputAutomatic* is TRUE, the switching output is switched automatically.
iBlockSize	The block size can be set as 4 bytes or 8 bytes. It must match the block size used for reading and writing in ST_RFID_AccessData [▶ 49].
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 [▶ 61]. No restriction is made with the value 16#FE. The following values are possible: 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 regarding the RFID reader configuration can be found in chapter Configuration [▶ 38].

10.1.8.3 ST_RFID_CfgStruct_DeisterUDL

```
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
```

The structure is suitable for writing with eRFC_SetConfig and reading with eRFC_GetConfig. (see command set [▶ 17])
This does not concern the parameterization of the PLC RFID library, but rather the proprietary configuration of the RFID reader.

As previously mentioned, attention must be paid to the difference between polling and triggering if necessary. In addition it must be considered in this context that the TriggerMode can nevertheless be present in addition to the PollingMode.

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. This setting means that the trigger condition always applies. [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 purpose. (see command set [▶ 17])

	The answer telegram that follows is received by the function block of the Tc RFID library. There is no assignment of read transponder data in this case. The received raw data can be taken from the module 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 provides the serial number. [default] **The answer telegram that follows is received by the function block of the Tc 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 (* This option is not possibe at all RFID Reader types. *));END_TYPE
eTranspType	If the RFID device is to detect only transponders of a certain type, this can be set with eTranspType [▶ 61]. No restriction is made with the value 16#FE. The following values are possible: 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 regarding the RFID reader configuration can be found in chapter Configuration [▶ 38].

10.1.8.4 ST_RFID_CfgStruct_LeuzeRFM

The structure is suitable for writing with eRFC_SetConfig and reading with eRFC_GetConfig. (see command set [▶ 17])
This does not concern the parameterization of the PLC RFID library, but rather 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 extentio
```

```
n *)
    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
```

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 answer telegram that follows is received by the function block of the Tc 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 (* This option is not possible at all RFID Reader types. *));END_TYPE
eTriggerMode	If eRFTR_ImmediateRead is set, the device is always ready to read. This setting means that the trigger condition always applies. [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 purpose. (see command set [▶ 17]) **The answer telegram that follows is received by the function block of the Tc 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 also transferred with write and read commands. By default, this corresponds to the last transponder serial number detected with the GetInventory command. Otherwise, the transponder serial number is specified in ST_RFID_Control [▶ 44].
bMultiTranspMode	If *bMultiTranspMode* is TRUE, anti-collision is active if several transponders are in the field.
bOutputAutomatic	If *bOutputAutomatic* is TRUE, the switching output is switched automatically.
iBlockSize	The block size can be set as 4 bytes or 8 bytes. It must match the block size used for reading and writing in ST_RFID_AccessData [▶ 49].
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 [▶ 61]. No restriction is made with the value 16#FE. The following values are possible: 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 regarding the RFID reader configuration can be found in chapter Configuration [▶ 38].

10.1.8.5 ST_RFID_CfgStruct_PepperlFuchsIDENT

```
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 structure is suitable for reading with eRFC_GetConfig. (see command set [▶ 17])
This does not concern the parameterization of the PLC RFID library, but rather the proprietary configuration of the RFID reader.

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. By default, the central processing unit is assigned channel 0 and the read/write heads are assigned channels 1-4.
The eRFC_GetConfig command and the outputs at the stReaderCfg output can be used to check the settings for all identification channels.

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. This information is stored in a structure of the type ST_RFID_HeadCfg for each head. The status of the central processing unit is output in *iErrCodeRcv* directly at the output of FB_RFIDReader [▶ 35]. 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 read/write 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, the trigger signal is inverted. **Further information about TriggerMode can be found in the chapter** RFID Reader Settings Pepper+Fuchs [▶ 27]**.** TYPE ST_RFID_TriggerCfg : STRUCT iIdentChannel :USINT; bTriggerMode :BOOL; bInverted :BOOL; bReserved :BOOL; END_STRUCT END_TYPE

Further information regarding the RFID reader configuration can be found in chapter Configuration [▶ 38].

10.2 Enumerations

10.2.1 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* of the TwinCAT PLC RFID library offers the above enumeration values at the *eCommand* input.
These are the available commands, such as reading or writing to a transponder. For details see description of the command set [▶ 17].

10.2.2 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* of the TwinCAT PLC RFID library offers the above enumeration values at the *eResponse* output.
These are described briefly below. Analogies to the telegram response types of the manufacturer's proprietary protocols are sometimes given.

The manufacturer's proprietary MessageID corresponding to the response listed here is indicated below in each case in *italics*. 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.

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 ready status, for example 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 conveyed an error code, then *eRFR_Error* is output at the *eResponse* output. The conveyed error code is specified in the output variable *iErrCodeRcv*; this is described in greater detail in the chapter RFID error codes [▶ 63]. If *eResponse* assumes the value *eRFR_Error*, then an error is also signalled by means of *bError* = TRUE at the output of the function block.

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

10.2.3 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
```

10.2.4 E_RFID_Manufacturer

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

10.2.5 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
```

10.3 T_RFID_TranspSRN

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

The data type contains a RFID transponder serial number.
The serial number of the transponder (frequently 8 bytes) is indicated as hexadecimal string.
(iRFID_MAXSRNLENGTH = 51)

11 Functions

11.1 F_GetVersionTcRFID

This function can be used to read PLC library version information.

FUNCTION F_GetVersionTcRFID: UINT

```
VAR_INPUT
    nVersionElement : INT;
END_VAR
```

nVersionElement : Version element to be read. Possible parameters:

- 1 : major number;
- 2 : minor number;
- 3 : revision number;

12 RFID Error Codes

Errors reports are made available at two outputs of the RFID PLC function block.
These two output variables iErrorID [▶ 63] and iErrCodeRcv [▶ 65] are described below.

Error ID - iErrorID

If an error is present, this output 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 reader, which is also indicated on iErrCodeRcv.

	ID (hex)	ID (dec)	Description
eRFERR_TimeOutElapsed	0x4001	16385	This error occurs if the time period specified as timeout at the input (default = 5 sec) 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.** More detailed information can be found in the chapter RFID reader connection [▶ 14] and in the 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 commands available are listed in the command set [▶ 17]. In order to ensure that the RFID reader model is

	ID (hex)	ID (dec)	Description
			known to the function block, the command 'GetReaderVersion' 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 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 specification of the input variable *tTimeOut* is invalid. The condition '*tTimeOut* > *tPreSendDelay* + *tPostSendDelay*' applies.
eRFERR_InvalidRawDataParam	0x4105	16645	If raw data are to be sent, then these must be specified at the input of the function block. The error *eRFERR_InvalidRawDataParam* is output if the specification of the input variable *pRawDataCommand* or *iRawDataCommandLen* is invalid. More details on this topic in the chapter 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 [▶ 49] 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 case of doubt, the exact valid value ranges of the data are given in the proprietary protocol specification by the manufacturer. If the last read configuration is used for the parameters not available in the configuration structure (bUseCfgDefault = FALSE in ST_RFID_ConfigIn [▶ 48]), it must be ensured that the configuration was read beforehand. 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_ConfigIn [▶ 48] is invalid.
eRFERR_InvalidCtrlHeadNumber	0x4109	16649	If an RFID reader with several read heads is addressed, a choice of 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.

	ID (hex)	ID (dec)	Description
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].
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 chapter).
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 has not yet been assigned to a group. For this reason, the first query that should be made at the program start is the GetReaderVersion command, depending on the RFID reader model.
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.

Error Code Received - iErrCodeRcv

If an error code is sent by the RFID reader, it will be output here.

Status messages, which do not lead to an error, are sometimes sent by the RFID reader and then output at iErrCodeRcv (bError remains FALSE and iErrorID shows no error).

A list of possible values can be found either in the data type declaration of the PLC RFID Library in the Library Manager or directly in the protocol specification.
Data types in this regard are the enumerations *E_RFID_ErrCodeRcv_Balluff, E_RFID_ErrCodeRcv_Deister, E_RFID_ErrCodeRcv_Leuze* etc.

13 Examples

The following examples have been developed with different RFID reader models.
Basically there are only few differences in handling between the RFID reader models. Hence each example can be used as orientation, also for another reader model.

It's recommended to look at two samples at least to get an understanding of the handling.

Tutorial

The Tutorial [▶ 30] 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.

Project download: https://infosys.beckhoff.com/content/1033/tcplclibrfid/Resources/11421957643/.zip

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 electronics 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 example program is already prepared for two RFID readers. Only the second linking in the TwinCAT system manager needs to be carried out.

Go to Sample [▶ 66]

Sample 2

This example is developed with a Baltech RFID Reader, which is also optionally integrated in Beckhoff Control-Panels and Panel-PCs. The device is connected via serial Com Port or USB port.

It can be used for a comfortable start-up and test of the device. The sample includes a simple visualization.

Go to Sample [▶ 67]

Sample 3

This example equates to a small application. It contains the detection, the reading and the writing of a transponder in an automatic process flow.

It's developed with a Pepperl+Fuchs RFID Reader. Either the 2-channel or the 4-channel model can be used.

Go to Sample [▶ 68]

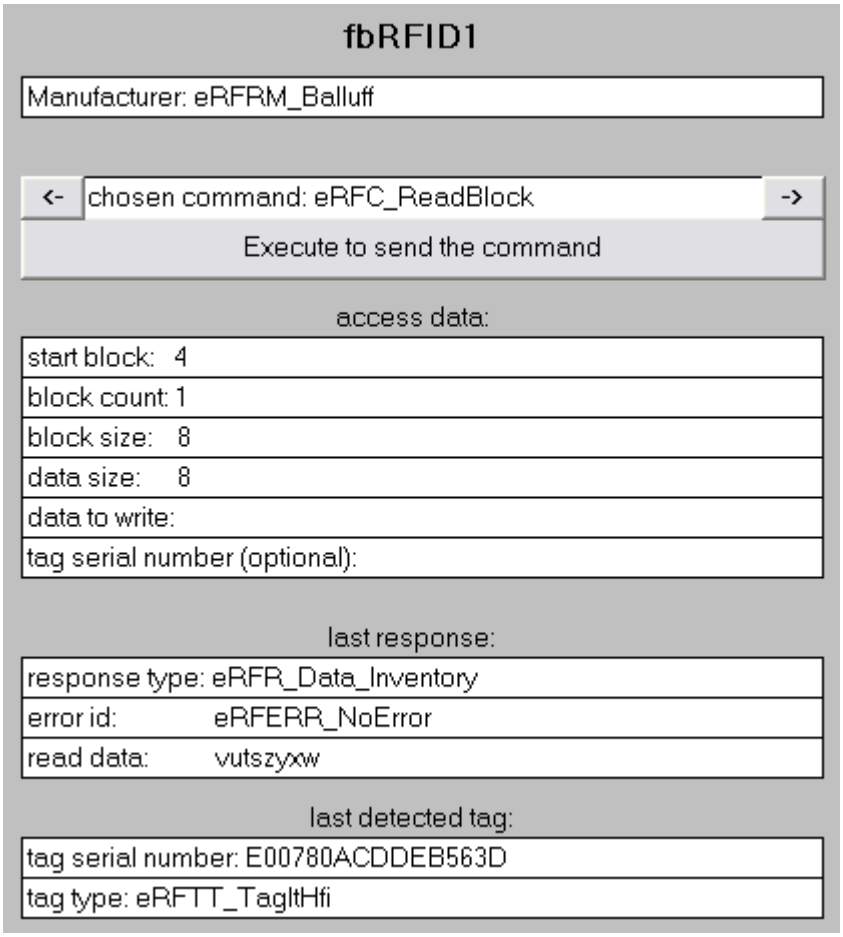
13.1 Sample 1

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

The example is tested with the Balluff BIS M 401 and Leuze electronic RFM32 models.
In the project, an RFID reader was connected to a 1-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 chapter RFID Reader Connection [▶ 14])
The project can also 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, among other things, reading and writing from the RFID transponder memory.
After program start, the appropriate RFID reader manufacturer must be selected via the local enumeration eManufacturer.

The command type can be selected by means of the local enumeration eCommand.
To start the call, the local variable bExecute must be set once to TRUE.
The results of the query are then indicated at the outputs of the RFID function block.
Alternatively, the operation can be done with the visualization integrated in the example:



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

For further information on the RFID reader communication sequence, please refer to the chapter Handling the RFID function block [▶ 38].

Project Download: https://infosys.beckhoff.com/content/1033/tcplclibrfid/Resources/11421959051/.zip

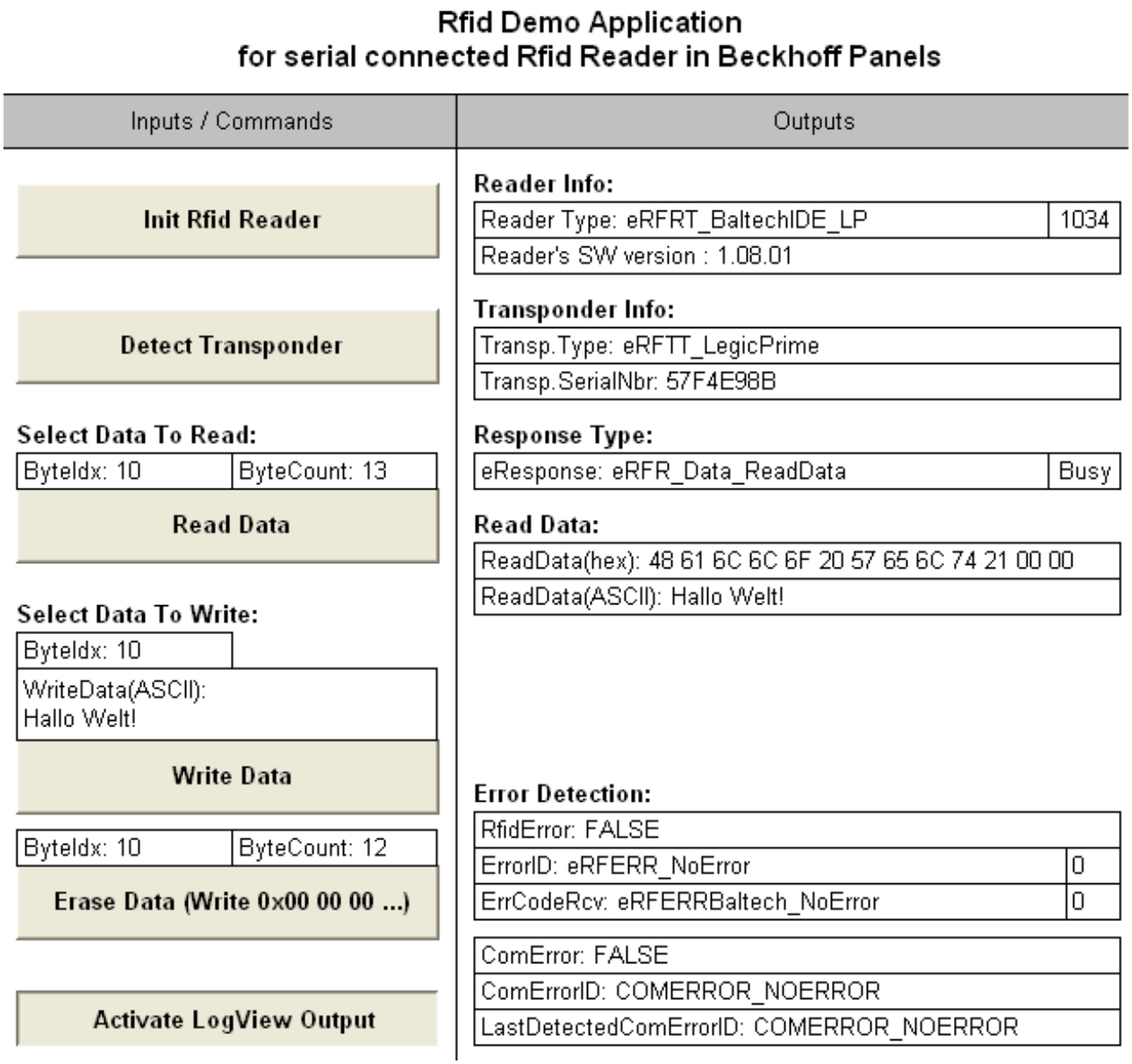
13.2 Sample 2

This example is developed with a Baltech RFID Reader, which is also optionally integrated in Beckhoff Control-Panels and Panel-PCs. The device is connected via serial Com Port or USB port.
It can be used for a comfortable start-up and test of the device.

Using a Baltech RFID reader connected to a serial Beckhoff terminal instead of a Com Port, you must change the serial background communication in the PLC code and reconfigure the Tc System Manager. (See chapter RFID reader connection [▶ 14])

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

With the integrated visualization you can execute them. The commands GetReaderVersion, GetInventory, ReadBlock and WriteBlock are implemented. Thus, a RFID transponder can be tested, and data (ASCII string) can be written to or read from it too.



First the RFID reader has to be initialized in order to make correct communication with the RFID reader possible. The button InitRfidReader executes the command GetReaderVersion for it.

For further information on the RFID reader communication sequence, please refer to the chapter Handling the RFID function block [▶ 38].

Activating the LogView Output shows the full serial communication in the LoggerView of the Tc System Manager.

Project download: https://infosys.beckhoff.com/content/1033/tcplclibrfid/Resources/11421960459/.zip or https://infosys.beckhoff.com/content/1033/tcplclibrfid/Resources/11421961867/.zip

13.3 Sample 3

This example corresponds to a small application. The application includes the recognition, 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 chapter RFID reader connection [▶ 14])

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 could also be solved by checking the tag serial number)

The program essentially contains a state machine with 6 states:
0: Initialization - execution of GetReaderVersion, GetConfig, etc.
1: Tag detection at read head 1- buffered GetInventory
2: Tag detection at 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 sequence, please refer to the chapter Handling the RFID function block [▶ 38].

Project Download: https://infosys.beckhoff.com/content/1033/tcplclibrfid/Resources/11421963275/.zip

The current version of the TwinCAT library must be used.

More Information:
www.beckhoff.com/ts6600

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