

Handbuch | DE

TS6600

TwinCAT 2 | PLC RFID Reader Communication



Inhaltsverzeichnis

1	Vorwort	5
1.1	Hinweise zur Dokumentation	5
1.2	Zu Ihrer Sicherheit.....	6
1.3	Hinweise zur Informationssicherheit	7
2	Einleitung	8
3	Systemvoraussetzungen	9
4	RFID Reader Hardware	10
5	RFID Reader Anbindung	14
6	Befehlssatz	17
7	RFID Reader Einstellungen und Handhabung	22
7.1	Balluff RFID Reader	22
7.2	Baltech RFID Reader	23
7.3	Deister RFID Reader.....	26
7.4	Leuze RFID Reader	27
7.5	Pepperl+Fuchs RFID Reader.....	28
8	Tutorial	31
8.1	Glossar	31
8.2	Installation/Bibliotheken	31
8.3	Serielle Anbindung	32
8.4	Baustein Deklaration	32
8.5	Baustein Verwendung	33
8.6	Test	34
9	FB_RFIDReader	36
9.1	Handhabungshinweise	39
9.2	Konfiguration	39
9.3	Low Level Kommunikation	41
10	Datentypen	43
10.1	Strukturen.....	43
10.1.1	ST_RFID_TransplInfo	43
10.1.2	ST_RFID_ReaderInfo	43
10.1.3	ST_RFID_RawData	45
10.1.4	ST_RFID_Control.....	45
10.1.5	ST_RFID_ConfigIn.....	49
10.1.6	ST_RFID_Config.....	50
10.1.7	ST_RFID_AccessData	51
10.1.8	Konfigurationsdaten	52
10.2	Enumerationen	61
10.2.1	E_RFID_Command.....	61
10.2.2	E_RFID_Response	61
10.2.3	E_RFID_ReaderGroup.....	63
10.2.4	E_RFID_ReaderManufacturer	63
10.2.5	E_RFID_TranspType	63

10.3 T_RFID_TranspSRN	64
11 Funktionen	65
11.1 F_GetVersionTcRFID	65
12 Fehlercodes	66
13 Beispiele	70
13.1 Beispiel 1	70
13.2 Beispiel 2	71
13.3 Beispiel 3	72

1 Vorwort

1.1 Hinweise zur Dokumentation

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs- und Automatisierungstechnik, das mit den geltenden nationalen Normen vertraut ist.

Zur Installation und Inbetriebnahme der Komponenten ist die Beachtung der Dokumentation und der nachfolgenden Hinweise und Erklärungen unbedingt notwendig.

Das Fachpersonal ist verpflichtet, stets die aktuell gültige Dokumentation zu verwenden.

Das Fachpersonal hat sicherzustellen, dass die Anwendung bzw. der Einsatz der beschriebenen Produkte alle Sicherheitsanforderungen, einschließlich sämtlicher anwendbaren Gesetze, Vorschriften, Bestimmungen und Normen erfüllt.

Disclaimer

Diese Dokumentation wurde sorgfältig erstellt. Die beschriebenen Produkte werden jedoch ständig weiterentwickelt.

Wir behalten uns das Recht vor, die Dokumentation jederzeit und ohne Ankündigung zu überarbeiten und zu ändern.

Aus den Angaben, Abbildungen und Beschreibungen in dieser Dokumentation können keine Ansprüche auf Änderung bereits gelieferter Produkte geltend gemacht werden.

Marken

Beckhoff®, TwinCAT®, TwinCAT/BSD®, TC/BSD®, EtherCAT®, EtherCAT G®, EtherCAT G10®, EtherCAT P®, Safety over EtherCAT®, TwinSAFE®, XFC®, XTS® und XPlanar® sind eingetragene und lizenzierte Marken der Beckhoff Automation GmbH.

Die Verwendung anderer in dieser Dokumentation enthaltenen Marken oder Kennzeichen durch Dritte kann zu einer Verletzung von Rechten der Inhaber der entsprechenden Bezeichnungen führen.

Patente

Die EtherCAT-Technologie ist patentrechtlich geschützt, insbesondere durch folgende Anmeldungen und Patente:

EP1590927, EP1789857, EP1456722, EP2137893, DE102015105702

mit den entsprechenden Anmeldungen und Eintragungen in verschiedenen anderen Ländern.

EtherCAT®

EtherCAT® ist eine eingetragene Marke und patentierte Technologie lizenziert durch die Beckhoff Automation GmbH, Deutschland

Copyright

© Beckhoff Automation GmbH & Co. KG, Deutschland.

Weitergabe sowie Vervielfältigung dieses Dokuments, Verwertung und Mitteilung seines Inhalts sind verboten, soweit nicht ausdrücklich gestattet.

Zuwendungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patent-, Gebrauchsmuster- oder Geschmacksmustereintragung vorbehalten.

1.2 Zu Ihrer Sicherheit

Sicherheitsbestimmungen

Lesen Sie die folgenden Erklärungen zu Ihrer Sicherheit.
Beachten und befolgen Sie stets produktspezifische Sicherheitshinweise, die Sie gegebenenfalls an den entsprechenden Stellen in diesem Dokument vorfinden.

Haftungsausschluss

Die gesamten Komponenten werden je nach Anwendungsbestimmungen in bestimmten Hard- und Software-Konfigurationen ausgeliefert. Änderungen der Hard- oder Software-Konfiguration, die über die dokumentierten Möglichkeiten hinausgehen, sind unzulässig und bewirken den Haftungsausschluss der Beckhoff Automation GmbH & Co. KG.

Qualifikation des Personals

Diese Beschreibung wendet sich ausschließlich an ausgebildetes Fachpersonal der Steuerungs-, Automatisierungs- und Antriebstechnik, das mit den geltenden Normen vertraut ist.

Signalwörter

Im Folgenden werden die Signalwörter eingeordnet, die in der Dokumentation verwendet werden. Um Personen- und Sachschäden zu vermeiden, lesen und befolgen Sie die Sicherheits- und Warnhinweise.

Warnungen vor Personenschäden

GEFAHR

Es besteht eine Gefährdung mit hohem Risikograd, die den Tod oder eine schwere Verletzung zur Folge hat.

WARNUNG

Es besteht eine Gefährdung mit mittlerem Risikograd, die den Tod oder eine schwere Verletzung zur Folge haben kann.

VORSICHT

Es besteht eine Gefährdung mit geringem Risikograd, die eine mittelschwere oder leichte Verletzung zur Folge haben kann.

Warnung vor Umwelt- oder Sachschäden

HINWEIS

Es besteht eine mögliche Schädigung für Umwelt, Geräte oder Daten.

Information zum Umgang mit dem Produkt



Diese Information beinhaltet z. B.:
Handlungsempfehlungen, Hilfestellungen oder weiterführende Informationen zum Produkt.

1.3 Hinweise zur Informationssicherheit

Die Produkte der Beckhoff Automation GmbH & Co. KG (Beckhoff) sind, sofern sie online zu erreichen sind, mit Security-Funktionen ausgestattet, die den sicheren Betrieb von Anlagen, Systemen, Maschinen und Netzwerken unterstützen. Trotz der Security-Funktionen sind die Erstellung, Implementierung und ständige Aktualisierung eines ganzheitlichen Security-Konzepts für den Betrieb notwendig, um die jeweilige Anlage, das System, die Maschine und die Netzwerke gegen Cyber-Bedrohungen zu schützen. Die von Beckhoff verkauften Produkte bilden dabei nur einen Teil des gesamtheitlichen Security-Konzepts. Der Kunde ist dafür verantwortlich, dass unbefugte Zugriffe durch Dritte auf seine Anlagen, Systeme, Maschinen und Netzwerke verhindert werden. Letztere sollten nur mit dem Unternehmensnetzwerk oder dem Internet verbunden werden, wenn entsprechende Schutzmaßnahmen eingerichtet wurden.

Zusätzlich sollten die Empfehlungen von Beckhoff zu entsprechenden Schutzmaßnahmen beachtet werden. Weiterführende Informationen über Informationssicherheit und Industrial Security finden Sie in unserem <https://www.beckhoff.de/secguide>.

Die Produkte und Lösungen von Beckhoff werden ständig weiterentwickelt. Dies betrifft auch die Security-Funktionen. Aufgrund der stetigen Weiterentwicklung empfiehlt Beckhoff ausdrücklich, die Produkte ständig auf dem aktuellen Stand zu halten und nach Bereitstellung von Updates diese auf die Produkte aufzuspielen. Die Verwendung veralteter oder nicht mehr unterstützter Produktversionen kann das Risiko von Cyber-Bedrohungen erhöhen.

Um stets über Hinweise zur Informationssicherheit zu Produkten von Beckhoff informiert zu sein, abonnieren Sie den RSS Feed unter <https://www.beckhoff.de/secinfo>.

2 Einleitung

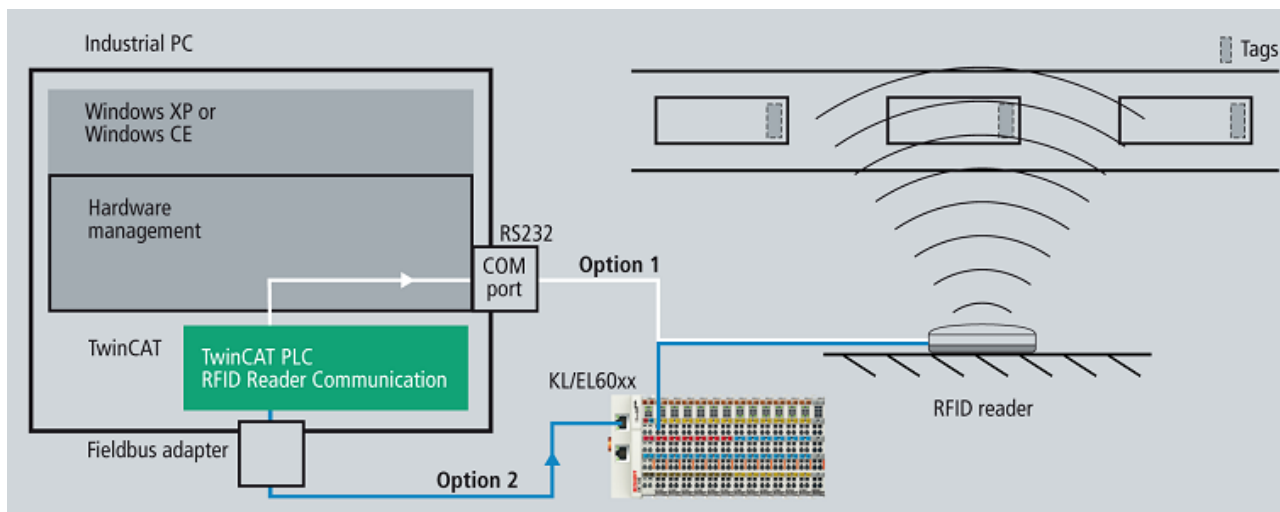
Die TwinCAT RFID Reader Communication Bibliothek ermöglicht die Kommunikation zu RFID Readern aus dem SPS Programm heraus.

Als RFID Reader werden sowohl reine Lesegeräte als auch Schreib-/Lesegeräte verstanden. Im Folgenden wird somit der Begriff RFID Reader verwendet, ohne dabei eine Beschränkung der Funktionalität zu implizieren.

Auch umfangreiche Applikationen, welche unterschiedliche Funktionen der RFID Reader nutzen, können nun leicht verwirklicht werden. Der Implementierungsaufwand ist sehr gering, weil kein herstellerepezifisches Schnittstellenprotokoll detailliert recherchiert und umgesetzt werden muss. Der Frameaufbau, die Telegrammzusammensetzung, die Befehlsbezeichnung, die Telegrammerkennung und einige weitere Protokolleigenarten werden automatisch durch die Bibliothek ausgeführt. Der Anwender kann sich vollständig auf seine Applikation konzentrieren und spart wertvolle Entwicklungszeit.

Die Handhabung der SPS Bibliothek ist für alle unterstützten RFID Reader Modelle gleich. Selbst bei einem Herstellerwechsel müssen nur kleine Änderungen der Applikation vorgenommen werden.

Mit der TwinCAT RFID Bibliothek kann erstmalig die ganze Welt der RFID Technik ohne Aufwand genutzt werden.



Schematische Darstellung einer RFID Reader Anbindung [► 14]

Stand der Dokumentation : 14.10.2013

3 Systemvoraussetzungen

- Programmierumgebung:
 - XP, XPe, WES, WES7, Win7, Win10
 - TwinCAT Installation Level: TwinCAT PLC oder höher;
 - TwinCAT System Version 2.10.0 Build 1340 oder höher;
 - **TcRFID.lib** Diese PLC Bibliothek muss in dem SPS-Projekt eingebunden werden. Alle anderen Bibliotheken werden automatisch hinzugefügt.
(Standard.lib; TcBase.lib; TcSystem.lib; TcUtilities.lib; COMlibV2.lib werden automatisch eingebunden)
- Zielplattform:
 - PC oder CX (x86): XP, XPe, WES, WES7, CE, Win7, Win10
 - TwinCAT SPS-Laufzeitsystem Version 2.10.0 Build 1340 oder höher;



Je nach RFID Reader Modell wird zur grundlegenden Konfiguration einmalig ein herstellereigenes Tool benötigt. (siehe dazu Kapitel: 'RFID Reader Einstellungen und Handhabung') In dem Fall sollten dessen Systemvoraussetzungen beachtet werden. Diese Voreinstellung kann ebenso von einem anderen PC aus vorgenommen werden. Es bietet sich auch die Nutzung von proprietären Testtools für den Aufbau an.

4 RFID Reader Hardware

Montagehinweise sind den herstellereigenen Produkthandbüchern zu entnehmen. Informationen zur Handhabung zwischen RFID Transponder und Readern sowie Lesegeschwindigkeiten etc. sind ebenfalls vom jeweiligen Hersteller zu beziehen.

Teilweise wird von den RFID Readern ein externer Trigger oder ein Schaltausgang angeboten. Dieser muss für die Funktionalität der TwinCAT RFID Bibliothek nicht verwendet werden.

i Hinweis


Die TcRFID Bibliothek bildet nicht den kompletten Leistungsumfang der herstellereigenen RFID Kommunikationsprotokolle ab. Nähere Informationen sind auch im [Befehlssatz des Bibliotheksbausteines \[► 17\]](#) beschrieben. Ergänzend kann auf die integrierte Möglichkeit zurückgegriffen werden, Rohdaten zu senden und zu empfangen - siehe dazu den Befehl 'eRFC_Send_RawData'.

RFID Readermodelle

Die TwinCAT RFID Bibliothek unterstützt unterschiedliche RFID Reader Modelle.

Die folgende Tabelle gibt an, welche RFID Reader Modelle welcher Hersteller unterstützt werden.

i Bei den Abbildungen handelt es sich um Symbolfotos von RFID Leser Modellen. Es kann Abweichungen zu den unterstützten Modellen geben und ebenso wird nicht jedes unterstützte Modell als Foto abgebildet.

RFID Reader Hersteller	RFID Reader Modell	Symbolfoto
Balluff	BIS M-400-007 (RS232) BIS M-401-007 (RS232) BIS L-6000-007 (2 read heads) (RS232) [ab Bibliothek v.1.2.5] BIS L-6020-007 (2 read heads) (RS232) [ab Bibliothek v.1.2.5]	  Bildnachweis: BALLUFF
Baltech [ab Bibliothek 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 ab Setup v1.3.0]	

RFID Reader Hersteller	RFID Reader Modell	Symbolfoto
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 [ab Bibliothek v. 1.1.1.0]	IDENTControl Compact (2 read heads) [IC-KP2-2HRX-2V1] (RS232) [ab Bibliothek v. 1.1.1.0] IDENTControl (4 read heads) [IC-KP-R2-V1] (RS232) [ab Bibliothek v. 1.2.8]	

Teilweise werden veraltete Firmwareversionen seitens der Reader nicht unterstützt.

Folgende RFID Reader Modelle sind laut Herstellerbeschreibung und -protokoll kompatibel. Die Kompatibilität der gelisteten Modelle sowie sonstiger Modelle ist jedoch nicht von Beckhoff bestätigt. Die Geräte werden nicht offiziell unterstützt. Vor Verwendung wird eine Kontaktaufnahme zu Beckhoff Automation empfohlen.

RFID Reader Hersteller	Reader Modelle
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)

Eine Unterstützung weiterer uns nicht bekannter Modelle der obigen Hersteller ist möglicherweise implizit gegeben. Laut Deister electronic ist in weiteren Modellen dasselbe Protokoll (deBus) implementiert. Eine Verwendung dieser Modelle ist ggf. nur mit beschränktem Funktionsumfang der TcRFID Bibliothek möglich.

Des Weiteren bieten manche Hersteller eigene Software an, um ihre Geräte für Beckhoff TwinCAT Systeme zugänglich zu machen.

i Hinweis

Weitere RFID Reader werden mit der TwinCAT SPS Bibliothek Serielle Kommunikation TwinCAT SPS Bibliothek Serielle Kommunikation unterstützt. So ist es mit dieser Bibliothek möglich mit einem beliebigen seriellen Gerät Datenbytes auszutauschen. Diese Alternative zur TwinCAT SPS RFID Bibliothek kann sinnvoll sein bei Read-Only RFID Readern. So können nicht unterstützte Geräte dennoch mit TwinCAT an einer Beckhoff Steuerung verwendet werden. Falls allein die Seriennummer des Transponders erforderlich ist und diese autark vom RFID Gerät gesendet wird, ist der Aufwand einer Auswertung der empfangenen Bytes überschaubar.

Transpondertypen

Eine vollständige Liste aller unterstützter Transpondertypen ist dem Handbuch des jeweiligem RFID Readers zu entnehmen. Welcher Transpondertyp für die Applikation sinnvoll scheint ist ebenfalls bei Bedarf mit dem Hersteller der RFID Reader oder Transponder zu klären.

Die TwinCAT Bibliothek arbeitet mit den Daten, welche aus der seriellen Schnittstelle bezogen werden. Das serielle Übertragungsprotokoll des Herstellers ist demnach entscheidend für eine Unterstützung durch die PLC Bibliothek. Die verwendete Funkfrequenz ist beispielsweise irrelevant.

Folgende Tabelle gibt exemplarisch an, welche Transpondertypen bei den jeweiligen RFID Reader Modellen laut Hersteller unterstützt werden. Diese Liste ist nicht vollständig. Vollständige und weitergehende Informationen hält der jeweilige Hersteller des RFID Reader Modells bereit.

Zu beachten ist, dass manche RFID Reader nur Transponder mit bestimmten Herstellerkennungen akzeptieren. Auf diese Beschränkung kann leider kein Einfluss genommen werden.

RFID Reader Modell	RFID Transpondertypen
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, ...

Diese Transpondertypen sind zudem in weiteren Speichergrößen erhältlich. Eine Kompatibilität ist hardwareabhängig und wird nicht garantiert. Ein Test wird empfohlen.

Teilweise sind spezielle werksseitige Programmierungen der Transponder möglich. Diese haben keinen Einfluss auf das Protokoll und müssen demnach applikationsabhängig mit Rücksprache zum Hersteller entschieden werden.

Spezifische Transponderparameter, welche in der PLC Bibliothek verwendet werden, können vom Nutzer bei Verwendung eines speziellen Transponders angepasst werden. Siehe dazu die Beschreibung der Struktur `ST_RFID_AccessData` [► 51].

Transponder werden bis zu einer maximalen Größe von 64 Kilobytes seitens der TcRFID Bibliothek unterstützt.

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 MB89R11 8, 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	-	-	++	++	-	-

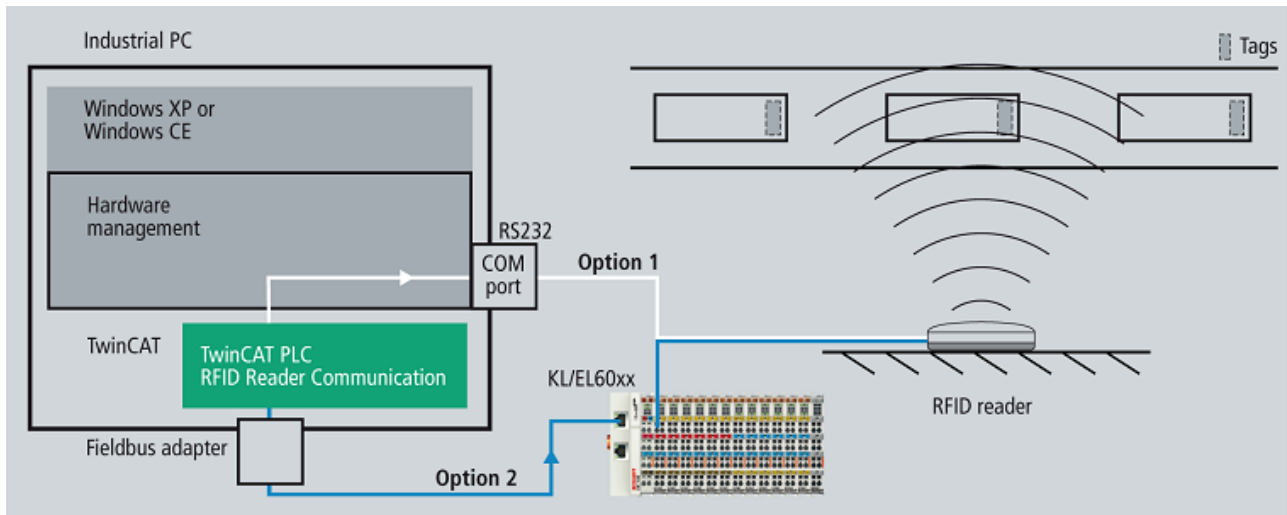
[++ very good; + good; - bad]

5 RFID Reader Anbindung

Alle mittels dieser SPS Bibliothek unterstützten RFID Reader werden über serielle Kommunikationsschnittstellen mit der Steuerung verbunden (RS 232, RS 422, RS 485 und virtuelle serielle COM Ports).

Dazu können folgende Beckhoff Produkte genutzt werden:

- Serielle EtherCAT Klemmen: EL6001, EL6002, EL6021, ...
- Serielle K-Bus Klemmen: KL6001/KL6031, KL6021, ...
- COM-Port eines beliebigen IPC und Embedded PC mit TwinCAT System.



● Verwendung mehrerer RFID-Reader

i Es muss je RFID-Reader eine separate Verbindung zu einer separaten Klemme erfolgen. Eine Unterstützung mehrerer RFID-Reader an einem RS485-Netz ist mit der SPS-RFID Bibliothek vorerst nicht gegeben.

Einrichten der seriellen Kommunikation in TwinCAT PLC Control

Der serielle Datenaustausch wird mit den Bausteinen der TwinCAT SPS Bibliothek COMlibV2 eingerichtet.

Legen Sie einen Sendepuffer sowie einen Empfangspuffer vom Type ComBuffer an. Dies kann global geschehen, muss aber nicht zwangsläufig.

Legen Sie außerdem noch zwei Datenstrukturen an, wie Sie im TwinCAT System Manager zur seriellen Kommunikation verwendet werden:

Falls der COM-Port verwendet wird, sieht dies wie folgt aus:

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

Neben PcComInData/PcComOutData sind bei Verwendung einer seriellen Klemme EL6inData22B/EL6outData22B sowie KL6inData5B/KL6outData5B und andere Datentypen möglich. Diese Strukturen werden im System Manager mit den Kanälen der seriellen Schnittstelle verlinkt.

i Bei Verwendung des Com-Ports muss hierzu im SystemManager zusätzlich der SyncMode aktiviert werden.

i Im TcSystemManager müssen die SPS-Variablen der richtigen (schnellen) Task zugeordnet und von dort passend verlinkt sein. Falls mehrere Tasks verwendet werden und die SPS-Variablen nicht innerhalb der richtigen Task liegen, können diese per Drag & Drop zur anderen Task verschoben werden.

Zur seriellen Kommunikation muss eine Instanz des 'SerialLineControl' angelegt werden. Diese wird in einer schnellen Task (<= 1 ms) zyklisch aufgerufen. Die nötige Taskzykluszeit ist abhängig von der Anwendung, der Datenmenge, der Baudrate und der Schnittstelle.

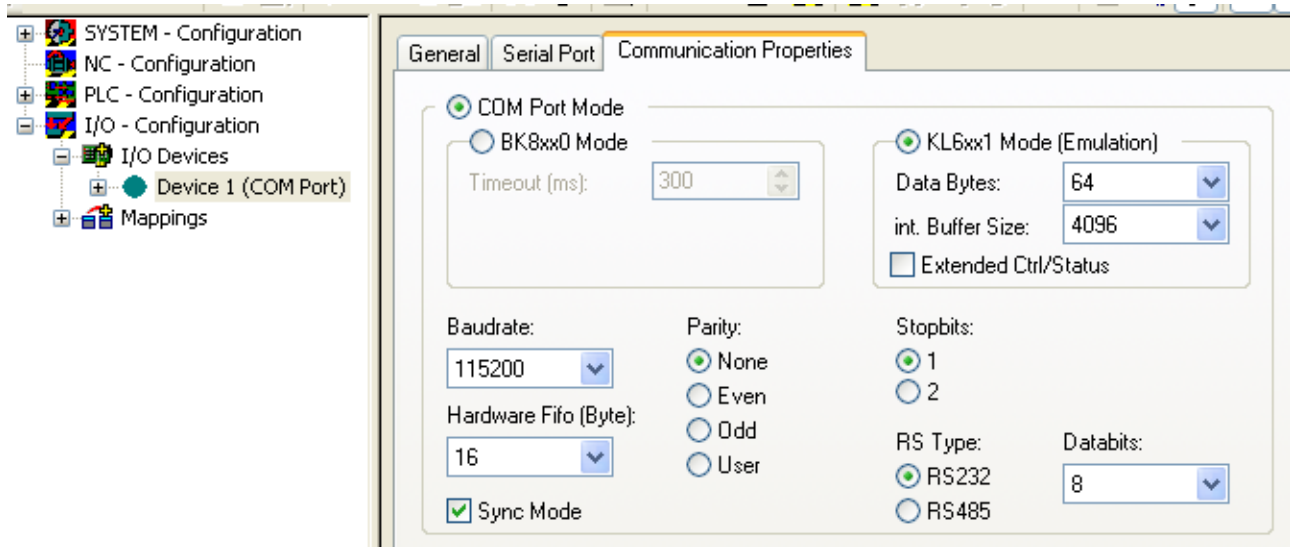
Je nach Anwendung und Schnittstelle ist es oft sinnvoll dies in einer zusätzlichen Task auszuführen, welche schneller ist als die Task der Applikation.

Beispiel 1: Beim Anschluss eines RFID Gerätes an einen COM Port und einer Baudrate von 115200 Baud ist eine Zykluszeit von 1ms notwendig.

Beispiel 2: Beim Anschluss eines RFID Gerätes an eine EL6001 und einer Baudrate von 9600 Baud ist eine Zykluszeit von maximal 6ms notwendig.

Weitere Informationen sowie Erläuterungen zur Verwendung virtueller COM-Ports sind in der [Dokumentation der SPS Bibliothek Serielle Kommunikation](#) zu finden.

Exemplarische Darstellung der COM Port Einstellungen im TwinCAT System Manager:



Der Aufruf des SerialLineControl ist im Folgenden exemplarisch dargestellt.

Aufruf als StructuredText im Falle der COM Port Verwendung:

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

Aufruf als StructuredText im Falle der Verwendung einer EtherCAT Klemme:

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

Aufruf als StructuredText im Falle der Verwendung einer K-Bus Klemme:

```
KL6Config3 (
  Execute      := bConfig3,
  Mode         := SERIALLINEMODE_KL6_5B_STANDARD,
  Baudrate     := 9600,
  NoDatabits   := 8,
  Parity       := 0,
  Stopbits     := 1,
  Handshake    := RS485_FULLDUPLEX,
  ContinousMode := FALSE,
  pComIn       := ADR (K1ComInData3) ,
  pComOut      := ADR (K1ComOutData3) ,
  SizeComIn    := SIZEOF (K1ComInData3) ,
```

```
    Busy      => bConfig3Act,  
    Done      => bConfig3Done,  
    Error     => bConfig3Error  
);  
IF NOT KL6Config3.Busy THEN  
    bConfig3 := FALSE;  
  
    LineControl3(  
        Mode      := SERIALLINE_MODE_KL6_5B_STANDARD,  
        pComIn    := ADR(KlComInData3),  
        pComOut   := ADR(KlComOutData3),  
        SizeComIn := SIZEOF(KlComInData3),  
        TxBuffer  := gKlComTxBuffer3,  
        RxBuffer  := gKlComRxBuffer3  
    );  
END_IF
```


6 Befehlssatz

Folgende Matrix listet den zur Verfügung stehenden Befehlssatz auf.

Wegen der grundsätzlichen Unterschiede der verschiedenen RFID Reader Modelle können nicht alle Befehle bei jedem Modell zur Verfügung gestellt werden.

Die Komplexität mancher proprietären Protokolle macht es zwangsläufig erforderlich, dass nicht jeder Befehl oder jede detaillierte Parametriermöglichkeit auch über die TwinCAT PLC Bibliothek gegeben sein kann. In Einzelfällen kann deshalb auf die weniger komfortable Kommunikationsmöglichkeit mittels dem angebotenen LowLevel Interface zurückgegriffen werden. Informationen dazu sind unter der Befehlsbeschreibung 'SendRawData' und im Kapitel [Low Level Kommunikation \[► 41\]](#) zu finden.

Informationen zu den Eigenschaften und Eigenarten der proprietären Protokolle sind zu jedem Modell vom Hersteller zu beziehen und werden meist mit dem Gerät mitgeliefert. Diese Protokoll-Spezifikationen sollten zumindest beim Anwender vorhanden sein, um Detailfragen recherchieren zu können und Eigenarten des RFID Readers nachzulesen. Auf die Eigenarten der einzelnen RFID Reader wird bereits soweit möglich an den speziellen Stellen innerhalb dieser Dokumentation hingewiesen. Allerdings bleibt der Hersteller der RFID Geräte weiterhin selber in der Verantwortung seine Geräte zu beschreiben und deren Verhalten und Eigenschaften zu gewährleisten. Eine detaillierte Beschreibung jedes Befehls und des speziellen Verhaltens des RFID Readers ist in den proprietären Protokoll-Spezifikationen gegeben. Welcher herstellerproprietäre Befehl dem hier gelisteten Befehl entspricht wird im Folgenden jeweils *kursiv* angegeben. Details können ebenso der Ausgangsstruktur [ST_RFID_RawData \[► 45\]](#) des Funktionsbausteines FB_RFIDReader entnommen werden.

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 IDENTCon- trol Com- pact
FieldOff [► 21]		x	x	x		x	
SendRawData [► 21]	x	x	x	x	x	x	x
ChangeDCType [► 21]							x

Diese Liste ist analog zur Enumeration [E_RFID_Command \[► 61\]](#) in der RFID SPS Bibliothek. Eine erfolgreiche Bearbeitung des angefragten Befehls durch den RFID Reader ist an den Statusausgängen des Funktionsbausteines sowie an der jeweiligen Response zu erkennen. Eine Liste möglicher Responses ist unter [E_RFID_Response \[► 61\]](#) einzusehen.

Im Folgenden werden die Befehle im Einzelnen erläutert:

GetReaderVersion [16#01]

Mit diesem Befehl können Informationen zum RFID Reader abgefragt werden. Je nach Verfügbarkeit wird die Modellbezeichnung, die Hard- und Softwareversion des Readers etc. am Bausteinanfang in der Struktur [ST_RFID_ReaderInfo \[► 43\]](#) ausgegeben.

Entsprechung im proprietären Protokoll:
 Deister: 0x02
 Leuze: 'V'
 Pepperl+Fuchs: 'VE'
 Baltech: System GetInfo

GetConfig [16#02]

Mit diesem Befehl wird die aktuelle Konfiguration des RFID Readers abgefragt. Alle relevanten empfangenen Parameter werden unter [ST_RFID_Config \[► 50\]](#) erläutert.

Weitere Informationen sind im Kapitel [Konfiguration \[► 39\]](#) zusammengefasst.

Entsprechung im proprietären Protokoll:
 Deister: 0x09
 Leuze: 'G'
 Pepperl+Fuchs: 'GS'

SetConfig [16#03]

Parametrierte Konfigurationseinstellungen können auf den RFID Reader übertragen werden. Nähere Informationen zur möglichen Konfiguration des RFID Readers befinden sich unter [ST_RFID_ConfigIn \[► 49\]](#).

Es wird empfohlen nach einem Konfigurationsbefehl erneut die aktuelle Konfiguration des Readers mittels dem Befehl GetConfig abzufragen.

Weitere Informationen sind im Kapitel [Konfiguration \[► 39\]](#) zusammengefasst.

Entsprechung im proprietären Protokoll:
 Baltech: System CfgWriteTLVBlock
 Deister: 0x09
 Leuze: 'C'

GetInventory [16#04]

Mit diesem Befehl wird der Typ und die Seriennummer eines aktuell im Lesefeld befindlichen Transponders abgefragt. Falls kein Transponder gefunden wird, folgt eine entsprechende Response.



Pepperl+Fuchs: Mit dem Parameter iHeadNumber in der Struktur ST_RFID_Control [► 45] wird festgelegt für welchen Lesekopf der Befehl auszuführen ist.

Entsprechung im proprietären Protokoll:

Balluff: 'U'

Deister: 0x82

Leuze: 'I'

Pepperl+Fuchs: 'SF' & 'EF'

Baltech: VHLSelect + VHLGetSnr

Polling [16#05]

Informationen aus dem Stack des RFID Readers werden abgefragt. Dabei kann es sich beispielsweise um die Seriennummer des letzten Transponders handeln. Es ist zu beachten, dass unterschiedliche RFID Reader verschieden große Stacks besitzen und teils nur eine Nachricht gespeichert wird.



Präsenzerkennung

Falls dies nicht über einen Konfigurationsparameter eingestellt werden kann, ist es nötig den Reader mittels zyklischem Polling Kommando lesebereit zu halten, so dass ein Transponder in Reichweite automatisch detektiert wird.

Entsprechung im proprietären Protokoll:

Deister: 0x0B

TriggerOn [16#06]

Falls der Trigger Mode aktiv ist, kann mit diesem Befehl ein Software Trigger anstatt eines Hardware Triggers ausgelöst werden.

Das darauffolgende Antworttelegramm wird vom Funktionsbaustein der Tc RFID Bibliothek empfangen. Eine Zuweisung von gelesenen Transponderdaten ist in dem Fall nicht gegeben. Die empfangenen Rohdaten können zur weiteren Verarbeitung dem Bausteininterface entnommen werden.

Entsprechung im proprietären Protokoll:

Deister: 0x85

Leuze: '+'

TriggerOff [16#07]

Siehe TriggerOn.

Entsprechung im proprietären Protokoll:

Deister: 0x85

Leuze: '-'

AbortCommand [16#08]

Falls ein Befehl seitens des RFID Readers in Bearbeitung ist, wird er mit diesem Kommando abgebrochen.



Pepperl+Fuchs: Mit dem Parameter iHeadNumber in der Struktur ST_RFID_Control [► 45] wird festgelegt für welchen Lesekopf der Befehl auszuführen ist.

Entsprechung im proprietären Protokoll:

Leuze: 'H'

Pepperl+Fuchs: 'QU'

ResetReader [16#09]

Dieses Kommando veranlasst den RFID Reader, ein Reset durchzuführen.

Entsprechung im proprietären Protokoll:

Balluff: 'Q'

Deister: 0x01

Leuze: 'R'

Pepperl+Fuchs: 'RS'

Baltech: System Reset

ReadBlock [16#0A]

Mit diesem Befehl wird eine bestimmte Anzahl von Datenbytes in Form von Blöcken definierter Größe aus dem Speicher des Transponders gelesen.

Für diesen Befehl ist die Übergabe der Eingangsstruktur [ST_RFID_AccessData \[► 51\]](#) nötig.

Bevor Daten vom Transponder gelesen werden, ist es üblich den Transponder zu erkennen und auszuwählen (siehe Befehl GetInventory).



Pepperl+Fuchs: Mit dem Parameter `iHeadNumber` in der Struktur [ST_RFID_Control \[► 45\]](#) wird festgelegt für welchen Lesekopf der Befehl auszuführen ist.

Entsprechung im proprietären Protokoll:

Balluff: 'L'

Deister: 0x83

Leuze: 'N'

Pepperl+Fuchs: 'SR' & 'ER'

Baltech: VHLRead

WriteBlock [16#0B]

Mit diesem Befehl wird eine bestimmte Anzahl von Datenbytes in Form von Blöcken definierter Größe in den Speicher des Transponders geschrieben.

Für diesen Befehl ist die Übergabe der Eingangsstruktur [ST_RFID_AccessData \[► 51\]](#) nötig.

Bevor Daten auf einen Transponder geschrieben werden, ist es üblich den Transponder zu erkennen und auszuwählen (siehe Befehl GetInventory).



Pepperl+Fuchs: Mit dem Parameter `iHeadNumber` in der Struktur [ST_RFID_Control \[► 45\]](#) wird festgelegt für welchen Lesekopf der Befehl auszuführen ist.

Entsprechung im proprietären Protokoll:

Balluff: 'P'

Deister: 0x84

Leuze: 'W'

Pepperl+Fuchs: 'SW' & 'EW'

Baltech: VHLWrite

OutputOn [16#0C]

Der Befehl setzt den Schaltausgang des RFID Readers permanent auf TRUE.



Dies ist nur möglich, falls der Schaltausgang nicht per Konfiguration automatisch angesprochen wird.

Entsprechung im proprietären Protokoll:

Deister: 0x0F

Leuze: 'A0FF'

OutputOff [16#0D]

Der Befehl setzt den Schaltausgang des RFID Readers permanent auf FALSE.



Dies ist nur möglich, falls der Schaltausgang nicht per Konfiguration automatisch angesprochen wird.

Entsprechung im proprietären Protokoll:

Deister: 0x0F

Leuze: 'A000'

FieldOn [16#0E]

Mit diesem Befehl kann das RFID Feld angeschaltet werden.

Entsprechung im proprietären Protokoll:

Deister: 0x81

Leuze: 'F1'

Baltech: System HFRreset

FieldOff [16#0F]

Mit diesem Befehl kann das RFID Feld ausgeschaltet werden.

Je nach RFID Reader Modell wird das Feld bei Trigger o.a. automatisch wieder aktiv.

Entsprechung im proprietären Protokoll:

Deister: 0x81

Leuze: 'F2'

Baltech: System HFRreset

SendRawData [16#10]

Mit diesem Kommando kann der RFID Funktionsbaustein als Low Level Schnittstelle genutzt werden. Die zu übermittelnden Daten werden in der [Control Struktur \[► 45\]](#) als Pointer übergeben. Bibliotheksintern wird ein Telegramm zusammengesetzt und versendet. Es können auf diese Art und Weise beliebige Daten an den RFID Reader gesendet werden.

Die daraufhin empfangenen Daten stehen am Ausgang des Funktionsbausteines in der [Rohdaten Struktur \[► 45\]](#) als adressiertes Datenfeld zur Verfügung. Weitere Informationen zum Ablauf im Kapitel [Low Level Kommunikation \[► 41\]](#).

Bei Nutzung des Kommandos SendRawData kann eine Auswertung der empfangenen Antwort nicht garantiert werden.

Beispiel: Wird ein Lesebefehl manuell als Bytefolge mittels des Kommandos SendRawData versandt, so werden empfangene Transponderdaten nicht auf eine in [ST_RFID_AccessData \[► 51\]](#) angegebene Adresse geschrieben. Eine Auswertung/Speicherung der Daten sollte demnach auch manuell mit Hilfe der immer angegebenen [Rohdaten Struktur \[► 45\]](#) geschehen.

ChangeDCType [16#11]

Mit dem Befehl 'ChangeDCType' kann der Transpondertyp am Lesekopf eingestellt werden. Dazu wird mittels *iDCType* in [ST_RFID_Control \[► 45\]](#) der Typ angegeben.



Pepperl+Fuchs: Mit dem Parameter *iHeadNumber* in der Struktur [ST_RFID_Control \[► 45\]](#) wird festgelegt für welchen Lesekopf der Befehl auszuführen ist.

Entsprechung im proprietären Protokoll:

Pepperl+Fuchs: 'CT'

7 RFID Reader Einstellungen und Handhabung

Dieses Kapitel gibt wichtige Hinweise zu den einzelnen RFID Reader Modellen.

Für jedes Gerät werden die nötigen Einstellungen beschrieben, sowie die Art der Handhabung.

Hier finden Sie Informationen entsprechend den Geräteherstellern:

- [Balluff RFID Reader \[▶ 22\]](#)
- [Baltech RFID Reader \[▶ 23\]](#)
- [Deister RFID Reader \[▶ 26\]](#)
- [Leuze RFID Reader \[▶ 27\]](#)
- [Pepperl+Fuchs RFID Reader \[▶ 28\]](#)

7.1 Balluff RFID Reader

RFID Reader Einstellungen

Für eine reibungslose Kommunikation zwischen Steuerung und RFID Reader müssen manche Einstellungen vor Systemstart vorgenommen werden. Hierzu zählt beispielsweise die Baudrate der seriellen Kommunikation. Um diese Einstellungen auf den RFID Reader zu übertragen kann ein proprietäres Tool des RFID Reader Herstellers nötig sein.

Für alle unterstützten RFID Reader Modelle hat sich diese Standardeinstellung der Datenübertragung bewährt:

Einstellung	Wert
Baudrate (RS232 und RS485)	9600 Baud
Parity Bit	none
Datenbits	8
Stopbits	1

Bei Bedarf lassen sich je nach Hardware auch andere Parameter einstellen oder es können die Werkseinstellungen des RFID Readers verwendet werden. Diese müssen dann auch in der [softwareseitigen Reader Anbindung \[▶ 14\]](#) übernommen werden.

Mittels der proprietären Tools müssen vor Systemstart folgende spezielle Einstellungen parametrieren werden.

Einstellung	Wert
Parameter der Datenübertragung (s.o.)	Einstellung in Analogie zu den im PLC Programm gewählten Werten
Protokolltyp - Telegramm Endekennung	LF CR
Datenträgertyp	All Types (oder Einstellung je nach Bedarf)
CT-Daten sofort senden	deaktiviert (oder aktiviert - es erfolgt jedoch keine Auswertung)
Dynamik-Betrieb	deaktiviert
Einschaltmeldung senden	deaktiviert (oder aktiviert - es erfolgt jedoch keine Auswertung)
CRC16 Datenprüfung	deaktiviert
Typ und serial number bei CT pres.	deaktiviert (oder aktiviert, je nach Bedarf)

i Falls 'Typ und serial number bei CT pres.' aktiviert ist, so sendet der RFID Reader automatisch den Transponder Typ und dessen Seriennummer sobald ein Transponder erkannt wurde. Wenn ein Befehl unverzüglich nach der Detektion eines Transponders und Erhalt dieser eingestellten Meldung abgesendet wird, so kann eine korrekte Zuordnung der Art der folgenden Response und eine zugehörige Auswertung nicht garantiert werden. Es wird empfohlen vorhandene Transponder manuell per Befehl 'Get Inventory' abzufragen. Es sollte andernfalls zumindest eine kurze Wartezeit bis zum Absenden des Befehles eingehalten werden und der Aufbau einem Testzyklus unterzogen werden.

i Ist der RFID Reader so eingestellt, dass automatisch Telegramme vom Reader zur Steuerung gesendet werden (beispielsweise bei Detektion eines Transponders durch 'Typ und serial number bei CT pres.>') muss folgendes beachtet werden: Die Endekennung (LF CR) wird in dem Fall als Suffix zur Erkennung von Telegrammen genutzt. Sobald diese 2 Bytes im Datenstrom erkannt werden, werden vorherige Daten zu einem Telegramm zusammengefasst. Ggf. führt dies zu einem Fehler und fehlender Auswertung des Telegrammes. Sollte der Fall auftreten können, dass die Endekennung in automatisch gesendeten Telegrammen innerhalb der Daten vorhanden ist, so muss anstatt der automatischen Übertragung eine Datenabfrage mittels Befehlsaufruf gewählt werden. Durch diese Maßnahme werden die Telegramme sicher erkannt.

RFID Reader Handhabung

Die Funktionsbausteine der Bibliothek unterstützen die Kommunikation von Balluff Readern zu Transponder mit 4-8 Bytes Seriennummer.

Bei Verwendung von Balluff RFID Readern wird die Seriennummer bei 13,56Mhz Transpondern im Gesamten byteweise vom Bibliotheksbaustein gedreht. Dies geschieht, weil die ausgelesene Seriennummer eines Transponders andernfalls nicht mit der an einem anderen Reader ausgelesenen Seriennummer übereinstimmen würde. So lassen sich Geräte verschiedener Hersteller gemeinsam in einem Verbund betreiben.

Bei Verwendung eines Balluff BIS-L60x0:

- Es muss die Variable *iDCType* = 0 (siehe Eingangsstruktur [stCtrl \[▶ 45\]](#)) gesetzt werden.
- Beim Aufruf des Befehls 'Get Inventory' werden Informationen von beiden Leseköpfen über die serielle Schnittstelle zurückgeliefert. Ausgewertet und am Ausgang *stTranspInfo* ausgegeben wird jedoch nur die Information von dem per *stCtrl.iHeadNumber* ausgewählten Lesekopfes.
- Falls 'Typ und serial number bei CT pres.' aktiviert ist, so sendet der RFID Reader automatisch den Transponder Typ und dessen Seriennummer, sobald ein Transponder erkannt wurde. Dies betrifft per default nur den ersten Lesekopf. Ein Umschalten auf den zweiten Lesekopf wird hierbei von der Bibliothek nicht direkt unterstützt. Des Weiteren kann die Nummer des Lesekopfes, an dem der Tag erkannt wurde, nicht zugewiesen werden (*iHeadNumber* = 0).

i Nicht alle Eigenarten jedes unterstützten RFID Reader Modells können hier genannt werden. Deshalb wird für detaillierte Informationen auf die herstellereigenen Dokumentationen hingewiesen.

7.2 Baltech RFID Reader

Falls ein unterstütztes Baltech RFID Stand-Alone Gerät verwendet wird, kann die TwinCAT SPS Bibliothek als Schnittstelle genutzt werden.

Eine Alternative ist die Verwendung mit bestimmten Beckhoff Control-Panels oder Panel-PCs. In diesen Geräten kann als Option ein RFID Reader integriert werden.

In diesem Fall wird ein SDK mitgeliefert, in dem sich die proprietären Dokumentationen befinden.



In beiden Fällen ist der Funktionsumfang der TwinCAT Bibliothek derselbe.

RFID Reader Einstellungen

Für eine reibungslose Kommunikation zwischen Steuerung und RFID Reader müssen manche Einstellungen vor Systemstart vorgenommen werden. Hierzu zählt beispielsweise die Baudrate der seriellen Kommunikation. Um diese Einstellungen auf den RFID Reader zu übertragen kann das proprietäre Tool 'Baltech id-engine explorer' des RFID Reader Herstellers verwendet werden. Ebenso kann mit dem Tool ein Funktionstest gemacht werden, um festzustellen, ob das RFID Gerät erkannt wird und ob die Transponder Karten erkannt werden.

Es hat sich diese Standardeinstellung der Datenübertragung bewährt:

Einstellung	Wert
Baudrate	115200 Baud
Parity Bit	none
Datenbits	8
Stopbit	1

Dies entspricht der Werkseinstellung der unterstützten Baltech RFID Geräte. Bei Bedarf lassen sich auch andere Parameter einstellen. Diese müssen dann auch in der softwareseitigen Reader Anbindung [► 14] übernommen werden.

Die Baudrate der Lesegeräte kann mit dem Tool 'Baltech id-engine explorer' geändert werden. (siehe Baltech Dokumentation: IdEngineExplorer.pdf)



Das Tool 'Baltech id-engine explorer' ist nur unter Windows XP lauffähig. Es ist nicht für Windows CE verfügbar. Somit ist die Baudrate unter Windows CE nicht konfigurierbar.



In der SPS wird eine schnelle Task benötigt, um die ankommenden Daten zu verarbeiten. Beim Anschluss des RFID Gerätes an einen COM-Port und einer Baudrate von 115200 Baud ist eine Zykluszeit von 1ms notwendig. (siehe Kapitel Reader Anbindung).



Um die Baudrate aus der SPS zu konfigurieren, kann folgende Bytefolge [0x 1C 00 09 06 00 01 03 00 02 xx xx - wobei xx xx für die Baudrate steht, so beispielsweise 9600 Baud: 96 Einheiten a 100Baud -> 0x 00 60] als Rohdatenblock übertragen werden. Näheres ist im Kapitel Low Level Kommunikation erläutert. Dies ist auch unter Windows CE möglich, sofern eine Übertragung mit der derzeitig eingestellten Baudrate möglich ist.

Verwendung vom virtuellen seriellen COM Port (USB)

Ist das Gerät mittels USB verbunden, so muss der passende Usb-To-Virtual-Com-Port Treiber installiert sein. Handelt es sich um einen Beckhoff Panel-PC, so ist der Treiber bereits vorinstalliert. Ebenso beinhaltet das SDK des RFID-Gerätes den Treiber. Der virtuelle COM-Port wird im Windows Gerätemanager angezeigt.

Die Kommunikation zum Treiber erfolgt über die Beckhoff TwinCAT 'Serielle Kommunikation'. Jedoch wird kein entsprechendes Gerät im TwinCAT System Manager angelegt und auf eine dortige Verknüpfung verzichtet. Weitere Informationen sind in der Dokumentation der SPS Bibliothek Serielle Kommunikation zu finden.

RFID Reader Handhabung

Die Bibliothek unterstützt die üblichen Standardeinstellungen der Baltech Kommunikation. Als 'Operational Mode' wird der Mode 'Host Operation' unterstützt. Andere Modi werden nicht unterstützt. Der Zugriff erfolgt intern über das BRP (Baltech Reader Protocol) im 'Communication Mode' 'Normal Mode'. Falls Rohdaten über die Low Level Kommunikationsmöglichkeit gesendet werden, ist darauf zu achten, dass obige Einstellungen korrekt innerhalb des Frames angegeben werden.



Nicht alle Eigenarten jedes unterstützten RFID Reader Modells können hier genannt werden. Deshalb wird für detaillierte Informationen auf die herstellereigenen Dokumentationen hingewiesen.

Konfiguration

Falls mit verschlüsselten Transponderkarten gearbeitet wird, so ist es nötig, dass derselbe Schlüssel auch im RFID Gerät bekannt ist. Der Baltech RFID Reader wird einmal konfiguriert. Daraufhin muss der Schlüssel nicht mehr angegeben werden. Ebenso kann der Schlüssel aus Sicherheitsgründen nicht aus der Gerätekonfiguration herausgelesen werden. Passend zur Verschlüsselung einer Transponderkarte wird in der Gerätekonfiguration ein VHL-file abgespeichert. Es können mehrere solcher VHL-files abgespeichert werden, um auf verschiedene Karten Zugriff ohne Umkonfiguration zu erhalten.

Es gibt drei Möglichkeiten, ein solches VHL-file in die Konfiguration des RFID Gerätes zu übertragen.

Konfigurationsart	Beschreibung
Konfigurationskarte	Eine Konfiguration kann über eine Konfigurationskarte übertragen werden. Dies ist die bevorzugte Variante.
Tool 'Baltech id-engine explorer'	Mit dem Tool ist die Übertragung einer Konfiguration in den Speicher des Baltech RFID Gerätes möglich. (siehe Baltech Dokumentation: IdEngineExplorer.pdf) Die spezifische Konfiguration kann in einfachen Fällen direkt in dem Tool erstellt werden. Alternativ bietet die Firma Baltech Unterstützung bei der Erstellung an und kann eine Datei mit der Konfiguration zusenden. Das Tool 'Baltech id-engine explorer' ist nur unter Windows XP lauffähig. Es ist nicht für Windows CE verfügbar.
aus der SPS	Für Mifare Classic Karten kann die Übertragung einer VHL-file Konfiguration im SPS Programmcode programmiert werden. Der Befehl SetConfig überträgt die Konfiguration welche am Eingang in <u>ST_RFID_ConfigIn</u> [▶ 49] angegeben ist. Die Struktur einer Mifare Karte und die möglichen Einstellungen zur Schlüsselvergabe sind auf der Detailseite von <u>ST_RFID_CfgStruct_BaltechMifVHLFile</u> [▶ 52] erläutert. (Ausführliche Informationen zu Mifare Karten finden sich zudem in dem Baltech Dokument: Mifare.pdf)

Transponder

Passende Transponderkarten für Baltech RFID Geräte können von verschiedenen Herstellern bezogen werden. Sollen die Karten mit Verschlüsselung genutzt werden, so bietet die Firma Baltech den Vertrieb von bereits vorkonfigurierten Karten an.

Herstellerkontakt

<http://www.baltech.de>

7.3 Deister RFID Reader

RFID Reader Einstellungen

Für eine reibungslose Kommunikation zwischen Steuerung und RFID Reader müssen manche Einstellungen vor Systemstart vorgenommen werden. Hierzu zählt beispielsweise die Baudrate der seriellen Kommunikation. Um diese Einstellungen auf den RFID Reader zu übertragen kann ein proprietäres Tool des RFID Reader Herstellers nötig sein.

Für alle unterstützten RFID Reader Modelle hat sich diese Standardeinstellung der Datenübertragung bewährt:

Einstellung	Wert
Baudrate (RS232 und RS485)	9600 Baud
Parity Bit	none
Datenbits	8
Stopbit	1

Bei Bedarf lassen sich je nach Hardware auch andere Parameter einstellen oder es können die Werkseinstellungen des RFID Readers verwendet werden. Diese müssen dann auch in der [softwareseitigen Reader Anbindung](#) [► 14] übernommen werden.

Mittels der proprietären Tools müssen gegebenenfalls vor Systemstart folgende spezielle Einstellungen parametrisiert werden.

Einstellung	Wert
Parameter der Datenübertragung (s.o.)	Einstellung in Analogie zu den im PLC Programm gewählten Werten

RFID Reader Handhabung

Hier sei erneut auf die Funktionsweise Polling hingewiesen (siehe [Befehlsbeschreibung](#) [► 17]), welche einen mehrfachen Aufruf des Befehls sinnvoll macht, falls auf aktuelle Transponderinformationen Wert gelegt wird. Hinzu kommt die Eigenart, dass bei den proxEntry Modellen ein Polling Befehl anliegen muss, um die Verbindung zum Transponder aufzubauen. Bei den UDL Modellen sieht die Konfiguration wiederum einen automatischen Verbindungsaufbau zu detektierten Transpondern vor, so dass kein Polling Befehl zwangsweise nötig ist.

In der RFID Reader Konfiguration muss die dem Tag entsprechende Blockgröße konfiguriert sein. Die Deister RDL Geräte unterstützen 4 Bytes oder 8 Bytes Blockgröße.

Beispiel: Falls für den Transponder eine Blockgröße von 8 Byte angegeben ist, muss der Reader mit dem Parameter *iBlocksize:=8* konfiguriert sein und der Lese- bzw. Schreibzugriff über die Struktur [ST_RFID_AccessData](#) [► 51] muss mit 8 Byte Blockgröße geschehen.

Deister RDL: Mit einem Schreibbefehl können maximal 36 Bytes Daten am Stück geschrieben werden. Sollen mehr Daten auf den Transponder geschrieben werden, müssen diese auf mehrere Befehle aufgeteilt werden.



Nicht alle Eigenarten jedes unterstützten RFID Reader Modells können hier genannt werden. Deshalb wird für detaillierte Informationen auf die herstellereigenen Dokumentationen hingewiesen.

Konfiguration

Wird eine neue Konfiguration auf das RFID Gerät geschrieben (Befehl 'SetConfig') muss Folgendes beachtet werden:

Deister RDL Geräte:

Nicht jede Kombination von Konfigurationparametern (Struktur [ST_RFID_CfgStruct_DeisterRDL](#) [► 54]) ist zulässig. Eine Missachtung der erforderlichen Abhängigkeiten führt zu einem Fehler (*eRFERR_InvalidCfg*):

Konfigurationsparameter	erforderliche Abhängigkeiten
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

Wird die Konfiguration als Register übertragen bestehen diese Abhängigkeiten ebenso und das RFID Gerät wird bei Unzulässigkeit einen Fehlercode zurückliefern.

7.4 Leuze RFID Reader

RFID Reader Einstellungen

Für eine reibungslose Kommunikation zwischen Steuerung und RFID Reader müssen manche Einstellungen vor Systemstart vorgenommen werden. Hierzu zählt beispielsweise die Baudrate der seriellen Kommunikation. Um diese Einstellungen auf den RFID Reader zu übertragen kann ein proprietäres Tool des RFID Reader Herstellers nötig sein.

Für alle unterstützten RFID Reader Modelle hat sich diese Standardeinstellung der Datenübertragung bewährt:

Einstellung	Wert
Baudrate (RS232 und RS485)	9600 Baud
Parity Bit	none
Datenbits	8
Stopbit	1

Bei Bedarf lassen sich je nach Hardware auch andere Parameter einstellen oder es können die Werkseinstellungen des RFID Readers verwendet werden. Diese müssen dann auch in der softwareseitigen Reader Anbindung [▶ 14] übernommen werden.

Mittels der proprietären Tools müssen gegebenenfalls vor Systemstart folgende spezielle Einstellungen parametrisiert werden.

Einstellung	Wert
Parameter der Datenübertragung (s.o.)	Einstellung in Analogie zu den im PLC Programm gewählten Werten

Sollte der RFID Reader mittels eines Triggers angesteuert werden, so wird das darauffolgende Antworttelegramm vom Funktionsbaustein der Tc RFID Bibliothek empfangen. Eine Zuweisung von gelesenen Transponderdaten ist in dem Fall nicht gegeben. Die empfangenen Rohdaten können zur weiteren Verarbeitung dem Bausteininterface entnommen werden.

RFID Reader Handhabung

In der RFID Reader Konfiguration muss die dem Tag entsprechende Blockgröße konfiguriert sein.

Die Leuze Geräte unterstützen 4 Bytes oder 8 Bytes Blockgröße.

Beispiel: Falls für den Transponder eine Blockgröße von 8 Byte angegeben ist, muss der Reader mit dem Parameter *iBlocksize:=8* konfiguriert sein und der Lese- bzw. Schreibzugriff über die Struktur ST_RFID_AccessData [▶ 51] muss mit 8 Byte Blockgröße geschehen.

Mit einem Schreibbefehl können maximal 36 Bytes Daten am Stück geschrieben werden. Sollen mehr Daten auf den Transponder geschrieben werden, müssen diese auf mehrere Befehle aufgeteilt werden.



Nicht alle Eigenarten jedes unterstützten RFID Reader Modells können hier genannt werden. Deshalb wird für detaillierte Informationen auf die herstellereigenen Dokumentationen hingewiesen.

Konfiguration

Wird eine neue Konfiguration auf das RFID Gerät geschrieben (Befehl 'SetConfig') muss Folgendes beachtet werden:

Nicht jede Kombination von Konfigurationparametern (Struktur [ST_RFID_CfgStruct_LeuzeRFM \[► 57\]](#)) ist zulässig. Eine Missachtung der erforderlichen Abhängigkeiten führt zu einem Fehler (*eRFERR_InvalidCfg*):

Konfigurationsparameter	erforderliche Abhängigkeiten
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

Wird die Konfiguration als Register übertragen bestehen diese Abhängigkeiten ebenso und das RFID Gerät wird bei Unzulässigkeit einen Fehlercode zurückliefern.

7.5 Pepperl+Fuchs RFID Reader

RFID Reader Einstellungen

Für eine reibungslose Kommunikation zwischen Steuerung und RFID Reader müssen manche Einstellungen vor Systemstart vorgenommen werden. Hierzu zählt beispielsweise die Baudrate der seriellen Kommunikation. Um diese Einstellungen auf den RFID Reader zu übertragen kann ein proprietäres Tool des RFID Reader Herstellers nötig sein.

Für alle unterstützten RFID Reader Modelle hat sich diese Standardeinstellung der Datenübertragung bewährt:

Einstellung	Wert
Baudrate (RS232 und RS485)	9600 Baud
Parity Bit	none
Datenbits	8
Stopbit	1

Bei Bedarf lassen sich je nach Hardware auch andere Parameter einstellen oder es können die Werkseinstellungen des RFID Readers (38400 Baud) verwendet werden. Diese müssen dann auch in der [softwareseitigen Reader Anbindung \[► 14\]](#) übernommen werden.

RFID Reader Handhabung

Bei Systemstart müssen die Modellinformationen (Befehl 'GetReaderVersion') und die aktuelle Reader Konfiguration (Befehl 'GetConfig') ausgewertet werden.

Der empfangene Status des Gerätes wird über den Ausgang *iErrCodeRcv* des Funktionsbausteines *FB_RFIDReader* angezeigt und im Fehlerfall durch *bError=TRUE* und *iErrorId=eRFERR_ErrorRcv* signalisiert. Die Leseköpfe besitzen ebenfalls eigene Status. Diese können mit der über 'GetConfig' gelesenen [Konfigurationsstruktur \[► 59\]](#) geprüft werden.

Bei einem Neustart sollten die eingestellten Transpondertypen überprüft werden. Falls die über 'GetConfig' gelesene Konfigurationsstruktur nicht die richtigen Transpondertypen für jeden Lesekopf anzeigt, so können diese mit dem Befehl 'ChangeDCType' korrigiert werden. Es wird empfohlen anstatt dem Default Wert (99) den für den Transponder spezifizierten Wert einzustellen.

Beim Lese- sowie Schreibzugriff auf den Datenspeicher eines Transponders muss für alle Pepperl+Fuchs Rfid Geräte eine Blockgröße von 4 Bytes (siehe [ST_RFID_AccessData \[▶ 51\]](#)) verwendet werden.



Nicht alle Eigenarten jedes unterstützten RFID Reader Modells können hier genannt werden. Deshalb wird für detaillierte Informationen auf die herstellereigenen Dokumentationen hingewiesen.

Buffered Command - Gepufferter Befehl

Mit der Eingangsvariablen *bBufferedCmd* in [ST_RFID_Control \[▶ 45\]](#) können Befehle abgesetzt werden, die für eine spätere dauerhafte Ausführung gepuffert werden. Dies ist mit den Befehlen *eRFC_GetInventory*, *eRFC_ReadBlock* und *eRFC_WriteBlock* möglich. Ein gepufferter Befehl kann mit dem Befehl *eRFC_AbortCommand* beendet werden.



Ist an einem Lesekopf ein solch gepufferter Befehl aktiv, darf der Trigger Mode nicht für diesen Kanal aktiviert werden bzw. aktiv sein! Ebenso darf kein Rohdatenbefehl abgesetzt werden, welcher diesen Kanal betrifft!

Trigger Mode

Es wird empfohlen keinen Trigger bzw. Sensorkanal zu verwenden. Der Trigger Mode sollte also für alle Kanäle deaktiviert sein.

Per Werkseinstellung des RFID Gerätes ist der Trigger auf allen Kanälen deaktiviert.

Alternativ kann beispielsweise der GetInventory Befehl zyklisch oder GetInventory als gepufferter Befehl (*bBufferedCmd* in [ST_RFID_Control](#)) aufgerufen werden.

Ist ein Trigger als Sensorkanal an der RFID Einheit nötig, so gibt es mit der TwinCAT Bibliothek folgende Möglichkeit:

Der Trigger liefert eine Meldung, ob er gerade ausgelöst oder ob der Triggerbereich verlassen wird. Diese Messages werden empfangen und als eResponse = eRFR_CmdConfirmation oder eRFR_NoTransponder angezeigt. In der Applikation kann darauf reagiert und der gewünschte Befehl ausgelöst werden.

Um einen Kanal dementsprechend als Sensorkanal/Trigger zu konfigurieren, muss der zugehörige Identkanal = 0 sein. Der notwendige Rohdatenbefehl ist im nächsten Absatz erläutert.



Die Trigger Einstellung darf nicht vom herstellereigenen Tool aus vorgenommen werden. Andernfalls sind die daraufhin eintreffenden Meldungen des Sensorkanals nicht von dem Baustein der TwinCAT Bibliothek lesbar.



Die korrekte Einstellung des Trigger Mode sollte mit dem Befehl 'GetConfig' und der Auswertung der gelesenen Konfigurationsstruktur überprüft werden. So kann die Einstellung falls nötig bei Programmstart nachgeholt werden.

Einstellungen per Rohdatenbefehle absetzen

Details sind dem Kapitel [LowLevelKommunikation \[▶ 41\]](#) und der Beschreibung der Strukturen [ST_RFID_Control \[▶ 45\]](#) sowie [ST_RFID_RawData \[▶ 45\]](#) zu entnehmen.

Baudrate:

Um die Baudrate des RFID Gerätes auf 9600 Baud einzustellen, müssen folgende Rohdaten gesendet werden:

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



Nach dem Ändern der Baudrate ist ein Reset des RFID Gerätes nötig.

Triggermode:

Um an Kanal 3 einen Triggersensor zu deaktivieren, so dass der Kanal als Lesekopf genutzt werden kann, müssen folgende Rohdaten gesendet werden:

ASCII	hex
TM300	54 4D 33 30 30

Um an Kanal 2 einen Sensor als Trigger zu konfigurieren, müssen folgende Rohdaten gesendet werden:

ASCII	hex
TM201	54 4D 32 30 31

Antwort: *eResponse = eRFR_CmdConfirmation*, sobald der Sensor ausgelöst wird.

Antwort: *eResponse = eRFR_NoTransponder*, sobald der Sensor verlassen wird.

Am Ausgang *stTransplInfo.iHeadNumber* ist der Sensorkanal angegeben, von dem die Antwort gesendet wurde.

Um an Kanal 4 einen Sensor als invertierten Trigger zu konfigurieren, müssen folgende Rohdaten gesendet werden:

ASCII	hex
TM402	54 4D 34 30 32

Antwort: *eResponse = eRFR_NoTransponder*, sobald der Sensor ausgelöst wird.

Antwort: *eResponse = eRFR_CmdConfirmation*, sobald der Sensor verlassen wird.

Am Ausgang *stTransplInfo.iHeadNumber* ist der Sensorkanal angegeben von dem die Antwort gesendet wurde.

Nachdem eine solche Einstellung vorgenommen wurde, muss die Gerätekonfiguration mit dem Befehl 'Get Config' erneut gelesen werden. Es empfiehlt sich die vorgenommenen Einstellungen mittels Auswertung der gelesenen Konfigurationsstruktur zu überprüfen.

8 Tutorial

Dieses Tutorial ist eine Anleitung, wie Sie mit Hilfe der TwinCAT SPS einen RFID Reader in Betrieb nehmen können.

Folgende Schritte werden dabei durchgeführt:

1. Glossar
2. Installation/Bibliotheken
3. Serielle Anbindung
4. Baustein Deklaration
5. Baustein Verwendung
6. Test

Es wird davon ausgegangen, dass Sie Ihre RFID Anwendung detailliert geplant haben und Ihre Applikation bereits modelliert ist.

Darauf aufsetzend haben Sie festgestellt, dass ein von der TcRFID Bibliothek unterstützter Reader grundsätzlich Ihren Anforderungen entspricht und Sie mit den angebotenen Kommandos Ihre Applikation implementieren können. Deshalb entscheiden Sie sich die TwinCAT Bibliothek zu nutzen, um sich die Kommunikation zu Ihrem RFID Reader zu erleichtern.



Wir führen dieses Tutorial mit dem RFID Reader Modell Balluff Bis M 401 durch. Für andere Modelle kann die Vorgehensweise allerdings grundsätzlich übernommen werden.

[nächste Tutorial-Seite \[▶ 31\]](#)

8.1 Glossar

Begriff	Erläuterung
CT	Carrier Type, Datenträgertyp, Transpondertyp
DC	Data Carrier - RFID Transponder (Datenspeicher)
HF	High Frequency
Label	RFID Transponder
LF	Low Frequency
RF	Radio Frequency
RFID	Radio Frequency Identification
RFID Reader	Ein RFID Lesegerät ('Reader') kann sowohl ein rein lesefähiges Gerät sein als auch ein Schreib-und Lesegerät.
Smart Label	RFID Transponder
SRN	Serial Number
Tag	RFID Transponder
UHF	Ultra High Frequency

[nächste Tutorial-Seite \[▶ 31\]](#)

8.2 Installation/Bibliotheken

- Starten Sie TwinCAT PLC Control
- Mit 'Datei > Neu' legen Sie ein neues PLC/SPS Projekt an

- Wählen Sie Ihre Zielplattform PC und CX (x86) oder CX (ARM)
- Öffnen Sie den Karteireiter 'Ressourcen' und den 'Library Manager'
- Fügen Sie mit 'Einfügen > Weitere Bibliotheken' die Bibliothek **TcRFID.lib** ein

Jetzt stehen Ihnen alle benötigten SPS Bausteine für die RFID Reader Kommunikation zur Verfügung. Alle weiteren implizit benötigten Bibliotheken wurden automatisch mit der TcRFID.lib eingebunden.

[nächste Tutorial-Seite \[▶ 32\]](#)

8.3 Serielle Anbindung

In diesem Beispiel wird der RFID Reader über die serielle EtherCAT Klemme EL 6001 angebunden.

Variablen:

Legen Sie global einen Sende- sowie einen Empfangspuffer (gEL6ComTxBuffer, gEL6ComRxBuffer) vom Type ComBuffer an.

Legen Sie außerdem noch zwei Daten Strukturen an, wie sie im TwinCAT System Manager zur seriellen Kommunikation verwendet werden:

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

Diese Strukturen verlinken Sie im System Manager mit den Kanälen des seriellen Ports.

Zur seriellen Kommunikation legen Sie eine Instanz des 'SerialLineControl' an. Rufen Sie diese in einer schnellen Task zyklisch auf.

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

Mode: Als Handle geben Sie in unserem Beispiel die serielle EtherCAT Klemme mit 22Byte Nutzdaten an.

Weiterführende Hinweise finden Sie im Kapitel [Serielle RFID Reader Anbindung \[▶ 14\]](#).

[nächste Tutorial-Seite \[▶ 32\]](#)

8.4 Baustein Deklaration

Der Funktionsbaustein FB_RFIDReader ist das Herzstück der ganzen RFID Reader Kommunikation. Die Deklaration und Initialisierung dieses Bausteines wird im folgenden Kapitel beschrieben.

FB_RFIDReader	
bExecute	bBusy
eCommand	bResponseRcv
stAccessData	eResponse
stCtrl	bError
stCfg	iErrorID
eManufacturer	iErrCodeRcv
tTimeOut	stReaderCfg
RxBuffer ▶	stReaderInfo
TxBuffer ▶	stTranspInfo
	stRawData

Legen Sie eine Instanz des FB_RFIDReader an.

Übergeben Sie ihr den Hersteller Ihres RFID Modells am Eingang *eManufacturer*.

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

Der FB_RFIDReader verfügt über 7 Eingänge (6 bei den spezifischen FBs aufgrund der fehlenden Herstellerangabe), 2 Ein-Ausgänge und 10 Ausgänge.

Zum Empfangen von Nachrichten, welche seitens des RFID Readers zur Steuerung gesendet werden, ist es ausreichend den Funktionsbaustein zyklisch aufzurufen. Der Eingang bExecute muss dabei FALSE bleiben.

Dies wird in diesem Beispiel genutzt, um vorerst eine einfache Präsenzerkennung zu implementieren. Dazu rufen Sie den Baustein wie folgt auf:

```
fbRFIDReader (
  bExecute      := FALSE,

  RxBuffer      := RxBuffer,
  TxBuffer      := TxBuffer,

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

Nun wird die zuletzt gelesene Seriennummer eines RFID Transponders in Ihrer String-Variablen dargestellt.

Es sollten zur Fehleranalys ebenfalls die Ausgänge bError und iErrorID etc. ausgewertet werden.

[nächste Tutorial-Seite \[▶ 33\]](#)

8.5 Baustein Verwendung

Eine effektivere Auswertung der empfangenen Daten können Sie mit folgenden Anweisungen erreichen:

Deklarationen:

```
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;
```

Programmablauf:

```
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
```

Empfangene Fehlercodes können online als Enumerationswert dargestellt werden, indem die Integer Variablen *iErrorID* und *iErrCodeRcv* direkt zugewiesen werden.

Mit Hilfe eines Triggers werden weitere Daten nur ausgewertet, falls eine neue Nachricht empfangen wird. Ihre String Variable *sTranspSerialNumber* gibt nun immer die Seriennummer des zuletzt detektierten Transponders wieder. Diese ist in diesem Fall ebenso am Funktionsbaustein[ausgang fbRFIDReader.stTranspInfo.sSerialNumber](#) zu sehen.

Weitere Informationen können je nach Anwendung von den Ausgängen des Funktionsbausteines übernommen werden.

Um eine empfangene Nachricht als komplette Bytefolge anzuzeigen, nutzen Sie beispielsweise die MEMCPY Funktion und kopieren die Rohdaten in ihr deklariertes Bytearray.

Jede Meldung Ihres RFID Readers wird nun empfangen und in obiger Weise ausgewertet.

[nächste Tutorial-Seite \[▶ 34\]](#)

8.6 Test

Sobald Sie das Programm gemäß den letzten Kapiteln erstellt haben, können Sie nun die aktuelle Konfiguration mit den verlinkten Variablen im System Manager aktivieren und TwinCAT in den RunMode versetzen.

Daraufhin folgt der Login seitens TwinCAT PLC Control und der Start der Applikation.

Wenn Sie einen Transponder vor Ihren RFID Reader bewegen, wird dieser detektiert und die empfangene Nachricht sowie Seriennummer im Programmcode übernommen. Die Werte werden online im Programmcode oder in einer zusätzlichen Visualisierung dargestellt.



Der Balluff RFID Reader muss für diese Funktionalität passend konfiguriert sein. 'Typ und serial number bei CT pres.' muss aktiviert sein. Auf diese und andere Konfigurationsparameter wird im Kapitel [RFID Reader Einstellungen - Balluff \[▶ 22\]](#) näher eingegangen. Nicht jeder RFID Reader unterstützt diese Einstellung.

Das komplette Projekt können Sie ebenso herunterladen: <https://infosys.beckhoff.com/content/1031/tcpplibrfid/Resources/11421957643.zip>

Weitere und umfangreichere Beispiele sind im [Kapitel Beispiele \[► 70\]](#) zu finden.

9 FB_RFIDReader

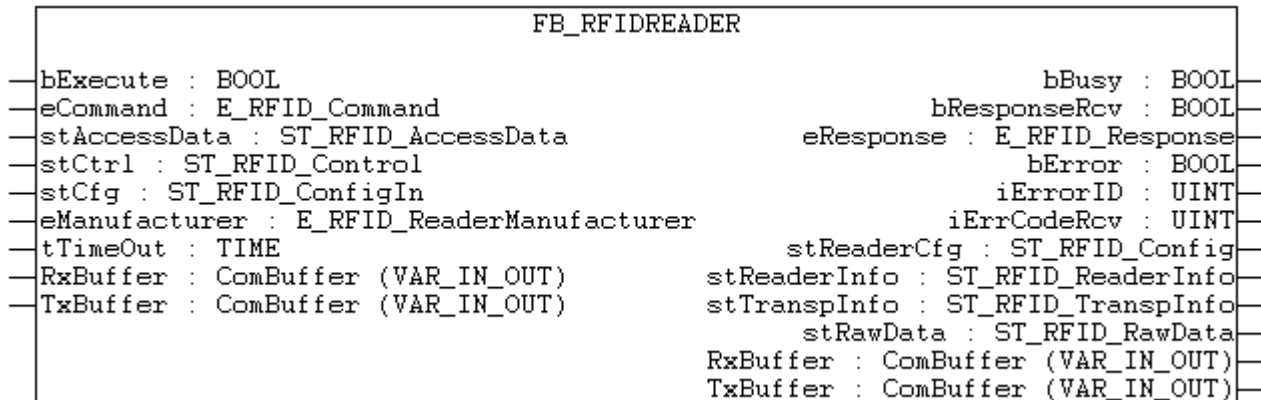
Die TwinCAT RFID Bibliothek besteht lediglich aus einem Funktionsbaustein.

Dessen Schnittstelle ist weitgehend selbsterklärend. Für einen schnellen Einstieg in die Handhabung der Bibliothek folgt eine Erläuterung der Schnittstellenvariablen.

Ebenso sei auf das [Tutorial \[► 31\]](#) und die Beispiele hingewiesen.

Die einheitliche Handhabung für alle RFID Reader Modelle und die damit verbundenen aufbereiteten Schnittstellendeklarationen sind besonders anwenderfreundlich.

Allerdings sei darauf hingewiesen, dass der Funktionsbaustein der RFID Bibliothek aufgrund der Unterschiede einiger RFID Reader Modelle einen geringfügigen Overheat besitzt. Diese unabdingbare Eigenschaft wird jedoch stark durch die Vorteile überwogen, welche die verfügbare Flexibilität bietet.



Eingangsvariablen

```

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

Um Nachrichten des RFID Readers zu empfangen wird der Funktionsbaustein mit FALSE an diesem Eingang aufgerufen.

Der Funktionsbaustein reagiert auf eine positive Flanke von *bExecute* indem der ausgewählte Befehl *eCommand* ausgeführt bzw. beim RFID Reader angefragt wird.

eCommand

Der Eingang *eCommand* bietet eine Auswahl an Befehlen, wie beispielsweise Lesen oder Schreiben eines Transponders, in Form einer Enumeration an. Siehe Beschreibung des [Befehlssatzes \[► 17\]](#). Ein Befehl wird ausgeführt, indem der Eingang *bExecute* gesetzt wird.

stAccessData

Falls ein Schreib- oder Lesebefehl ausgeführt werden soll, müssen mit dieser Eingangsstruktur Parameter übergeben werden. Details zu dieser Struktur: [ST_RFID_AccessData \[► 51\]](#)

stCtrl

Mit *stCtrl* können unterschiedliche Kontrollparameter am Eingang übergeben werden. Dazu gehört unter anderem auch die Möglichkeit, Verzögerungszeiten anzugeben. Details zu dieser Struktur: [ST_RFID_Control \[► 45\]](#)

stCfg

Ein RFID Reader besitzt eine interne Konfiguration. Diese lässt sich bei einigen Geräten auslesen und ändern. An dem Eingang *stCfg* werden Konfigurationsparameter übergeben, die auf den RFID Reader übertragen werden sollen. Siehe Beschreibung der [Konfigurationsmöglichkeiten](#) [► 39].

Details zu dieser Struktur: [ST_RFID_ConfigIn](#) [► 49]

eManufacturer

An diesem Eingang wird der Hersteller des verwendeten RFID Reader Modells angegeben. Details zu dieser Enumeration: [E_RFID_ReaderManufacturer](#) [► 63]

tTimeout

Gibt eine maximale Zeitdauer für die Ausführung des Funktionsbausteines an. Der Defaultwert ist 5 Sekunden.



Es gilt die Bedingung $tTimeout > tPreSendDelay + tPostSendDelay$. Andernfalls wird ein Fehler am Ausgang ausgegeben. Siehe Details zu den Verzögerungszeiten in [ST_RFID_Control](#) [► 45].

Ein-/Ausgangsvariablen

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

RxBuffer

Es wird der Empfangspuffer angegeben, welcher als Eingangsvariable deklariert und im TwinCAT System Manager mit der seriellen Klemme verlinkt wurde.

Siehe dazu die Beschreibung der [seriellen Anbindung eines RFID Readers](#) [► 14].

TxBuffer

Es wird der Ausgangspuffer angegeben, welcher als Ausgangsvariable deklariert und im TwinCAT System Manager mit der seriellen Klemme verlinkt wurde.

Siehe dazu die Beschreibung der [seriellen Anbindung eines RFID Readers](#) [► 14].

Ausgangsvariablen

```
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

Der Ausgang *bBusy* wird bei einem gültigen Kommandoaufruf für mindestens einen Takt TRUE. Der Funktionsbaustein darf erst wieder mit *bExecute* = TRUE für einen erneuten Befehl aufgerufen werden, wenn *bBusy* zu FALSE gewechselt ist und der Funktionsbaustein somit nicht mehr im aktiven Sendezustand ist. Somit können, wenn *bBusy* = FALSE erkannt wird, wiederum alle weiteren Ausgangsvariablen *bResponseRcv*, *eResponse*, *bError*, *iErrCodeRcv*, ... und *stRawData* ausgewertet werden. Wird zu einem getätigten Kommandoaufruf eine Response erwartet, so bleibt der Funktionsbaustein so lange *bBusy* = TRUE bis ein Telegram empfangen wird oder das Timeout *tTimeout* erreicht wird.

Falls dem Funktionsbaustein Verzögerungszeiten *tPreSendDelay* und/oder *tPostSendDelay* mitgegeben wurden, ist *bBusy* mindestens so lange TRUE wie die Summe dieser Zeiten. Siehe zu dieser Konfigurationseinstellung [ST_RFID_Control \[► 45\]](#).

bResponseRcv

Sobald eine Response vom RFID Reader bei der Steuerung eingetroffen ist, wird dieses Flag für mindestens einen Zyklus gesetzt. Mit steigender Flanke zu *bResponseRcv* = TRUE wird allgemein das Eintreffen eines Telegramms signalisiert. Somit können, wenn dieses Flag erkannt wird, wiederum auch unerwartete Telegramme und die dazugehörigen Ausgangsvariablen *eResponse*, *bError*, *iErrCodeRcv*, ... und *stRawData* ausgewertet werden.

Wird zu einem getätigten Kommandoaufruf eine Response erwartet, so bleibt der Funktionsbaustein so lange *bBusy* = TRUE bis ein Telegramm empfangen wird. Je nach Kommandoaufruf kann mehr als eine Response eintreffen bevor die Aktion abgeschlossen und *bBusy* = FALSE ist.

Je nach Konfigurationseinstellung der Verzögerungszeiten in [ST_RFID_Control \[► 45\]](#) kann *bResponseRcv* bereits TRUE werden, bevor *bBusy* wieder auf FALSE wechselt.

eResponse

Sobald *bResponseRcv* TRUE anzeigt, gibt diese Enumeration die Art der empfangenen Nachricht an. Je nach Art kann beispielsweise die entsprechende Auswertung folgen.

Details zu dieser Enumeration: [E_RFID_Response \[► 61\]](#)

bError

Der Ausgang *bError* wird TRUE, sobald ein Fehler auftritt. Dabei kann es sich um fehlerhafte Eingangsparameter, um Übertragungsfehler, um Fehler seitens des RFID Readers oder um ein Timeout handeln.

Welche Art von Fehler aufgetreten ist wird mit der folgenden Ausgangsvariablen *iErrorID* angezeigt. Details zur Fehlerdarstellung sind im Kapitel [Fehlercodes \[► 66\]](#) angegeben.

iErrorID

Falls ein Fehler auftritt, wird die Art dieser Fehler am Ausgang *iErrorID* angezeigt.

Details zu den möglichen FehlerIDs sind im Kapitel [Fehlercodes \[► 66\]](#) angegeben.

iErrCodeRcv

Der am Ausgang *iErrCodeRcv* angegebene Fehlercode entspricht dem vom RFID Reader an die Steuerung gesendetem Fehlercode.

Details zur Fehlerdarstellung sind im Kapitel [Fehlercodes \[► 66\]](#) angegeben.

stReaderCfg

Ein RFID Reader besitzt eine interne Konfiguration. Diese lässt sich bei einigen Geräten auslesen und ändern. Am Ausgang *stReaderCfg* werden diese ausgelesenen Konfigurationsparameter zur Verfügung gestellt. Details zu dieser Struktur: [ST_RFID_Config \[► 50\]](#)

stReaderInfo

Jeder RFID Reader besitzt eigene Kenndaten wie Bezeichnung, Hardwareversion etc. Diese Werte, welche unter anderem über den Befehl 'GetReaderVersion' abgefragt werden können, sind in der Ausgangsstruktur *stReaderInfo* angegeben. Details zu dieser Struktur: [ST_RFID_ReaderInfo \[► 43\]](#)

stTranspInfo

Die Struktur *stTranspInfo* enthält Informationen zu dem zuletzt gelesenen Transponder. Hier wird unter anderem die Seriennummer des Transponders ausgegeben. Details zu dieser Struktur: [ST_RFID_TranspInfo \[► 43\]](#)

stRawData

Die Ausgangsstruktur *stRawData* gibt die gesendeten sowie die empfangenen Rohdaten aus. Details zu dieser Struktur: [ST_RFID_RawData \[► 45\]](#)

9.1 Handhabungshinweise

RFID Bibliothek Handhabung

Wenn Sie die Bibliotheksdatei TcRFID.lib eingebunden haben, erhalten Sie Zugriff auf alle Funktionen. Die Bibliothek stellt zur Kommunikation mit einem RFID Reader einen Funktionsbaustein zur Verfügung. Es kann der generelle Funktionsbaustein FB_RFIDReader genutzt werden, welcher für alle RFID Reader Modelle verwendbar ist oder einer der herstellerspezifischen Funktionsbausteine. Diese bieten den identischen Funktionsumfang, annähernd das gleiche Interface, das gleiche Handling und sind zudem code- und performanceoptimiert.

Der, zur RFID Reader Kommunikation zur Verfügung gestellte, Funktionsbaustein bietet High-Level Kommunikation mit High-Level Interface. Ein [Befehlssatz \[► 17\]](#) stellt unterschiedlichste Kommandos zur Verfügung.

Zusätzlich ermöglicht die integrierte [Low-Level Kommunikation \[► 41\]](#) das Senden und Empfangen von Rohdaten.



Die TcRFID Bibliothek stellt an die RFID Reader die Erwartung, dass eine auf einen Befehl folgende Response unmittelbar nach dem Befehl erfolgt und der Dialog nicht durch ein anderes Telegram unterbrochen wird. Andernfalls ist eine Auswertung ggf. nicht möglich.

Allgemeine Handhabung des Funktionsbausteines

Je nach RFID Reader Modell kann das Gerät ohne vorherige Aufforderung ein Telegram zur Steuerung senden. Zum Empfang reicht ein zyklischer Aufruf des RFID Funktionsbausteines mit *bExecute* = FALSE. Alle möglichen aktiven Zugriffe auf das RFID Gerät sind im [Befehlssatz \[► 17\]](#) gelistet. Allen Befehlen ist folgende Vorgehensweise gleich. Der Funktionsbaustein wird mit einer positiven Flanke am Eingang *bExecute* aufgerufen. Danach liefert zyklisches Aufrufen des Funktionsbausteines (*bExecute* = FALSE) das Ergebnis der Abfrage am Ausgang, sobald die Bearbeitung der Abfrage abgeschlossen ist (*bBusy* = FALSE). Weitere Handhabungshinweise liefert die [Beschreibung der Ein- und Ausgangsvariablen \[► 36\]](#) des Funktionsbausteines sowie das Tutorial/Beispiel in dieser Dokumentation. Der Funktionsbaustein muss so lange aufgerufen werden (*bExecute* = FALSE) bis die interne Bearbeitung abgeschlossen (*bBusy* = FALSE) ist. Währenddessen sind alle Eingänge des Funktionsbausteines unverändert zu belassen.



Bei Systemstart sind folgende Aktionen zur Initialisierung eines über die TwinCAT Bibliothek integrierten RFID Readers notwendig. Soweit laut Befehlssatz verfügbar müssen die Modellinformationen (Befehl 'GetReaderVersion') und die aktuelle Reader Konfiguration (Befehl 'GetConfig') ausgewertet werden. Weil eine erfolgreiche Kommunikation mit dem RFID Reader von diesen Daten abhängig ist, muss sichergestellt werden, dass immer die aktuellen Werte vorliegen und bei Bedarf abgefragt werden.

Alle empfangenen Nachrichten werden zusätzlich als Rohdaten in nicht aufbereiteter Form komplett am Ausgang zur Verfügung gestellt.

RFID Reader Handhabung

Im Kapitel 'RFID Reader Einstellungen und Handhabung' werden Eigenarten der unterstützten Reader Modelle genannt.

Die dort aufgeführten Hinweise sind den speziellen RFID Reader Herstellern zugeordnet:

9.2 Konfiguration

Alle unterstützten RFID Reader lassen sich mit demselben Befehl konfigurieren. Dieser muss gemäß dem [Befehlssatz \[► 17\]](#) für das spezielle Modell verfügbar sein.

Zu jeden Programmstart sollte neben der Reader Version auch die aktuelle Konfiguration des Readers angefordert werden.

Weil die RFID Reader unterschiedlicher Hersteller nie identische Konfigurationsmöglichkeiten besitzen, bietet die PLC RFID Library neben der Eingangs-Konfigurationsstruktur jeweils eine Unterstruktur pro Hersteller mit den spezifischen Parametern. (ST_RFID_CfgStruct_DeisterUDL, ST_RFID_CfgStruct_LeuzeRFM, ...) Die dort gelisteten Parameter sind vom Nutzer im Rahmen der gültigen Wertebereiche beliebig zu parametrieren. Die Bedeutung der Parameter ist entweder der Strukturdeklaration oder den proprietären Spezifikationen zu entnehmen.

Konfiguration lesen

Um die aktuelle RFID Reader Konfiguration auszulesen wird der Befehl 'Get Config' aus dem [Befehlssatz \[▶ 17\]](#) verwendet. Daraufhin können bei erfolgreicher Abfrage die Konfigurationsdaten am Ausgang des Funktionsbausteines entnommen werden. Sie liegen dort in der Struktur [ST_RFID_Config \[▶ 50\]](#) als Konfigurationsstruktur sowie auch als Konfigurationsregister vor.

Konfiguration ändern

Um eine RFID Reader Konfiguration zu schreiben wird der Befehl 'Set Config' aus dem [Befehlssatz \[▶ 17\]](#) verwendet.

Nach einem 'Set Config' Befehl muss die aktuelle Konfiguration einmal mit dem Befehl 'Get Config' ausgelesen werden.

Falls der Nutzer weitergehende spezielle Konfigurationsparameter über ein externes Tool einstellt und diese beibehalten will, sollte das Flag für 'Default Values' *bUseCfgDefault* in der Struktur [ST_RFID_ConfigIn \[▶ 49\]](#) deaktiviert werden.



Hinweis

Teilweise sind bestimmte Kombinationen von Konfigurationsparametern unzulässig. Welche Parameterwerte sich bei welcher Kombination ausschließen ist den proprietären Protokollspezifikationen der RFID Reader Hersteller zu entnehmen. Bei fehlerhafter Eingabe der Parameter wird entweder bereits vor Konfigurationsanfrage ein Fehler generiert oder der RFID Reader signalisiert durch seine Response, dass die Konfigurationsdaten nicht übernommen werden konnten.

Konfigurationsdaten

Jede Konfiguration kann als Register (Byte Array) oder als Struktur gesehen werden. Dabei handelt es sich nicht um die Parametrierung der PLC RFID Bibliothek sondern um die proprietäre Konfiguration des RFID Readers. So gibt es in der PLC RFID Bibliothek verschiedene Konfigurationsstrukturen, welche die Rohdaten der Konfigurationsregister unterschiedlicher RFID Reader aufarbeiten. Am Ausgang des Funktionsbausteines der Bibliothek werden sofern verfügbar beide Varianten in [ST_RFID_Config \[▶ 50\]](#) zur Verfügung gestellt. Dies geschieht über Pointer.

Baltech

Die Konfigurationsdaten werden für Baltech RFID Reader als Struktur verwendet.

[ST_RFID_CfgStruct_BaltechMifVHLFile \[▶ 52\]](#)

Die Struktur ist für das Schreiben mit dem Befehl `eRFC_SetConfig` geeignet. (siehe [Befehlssatz \[▶ 17\]](#))

Balluff

Es wird keine Möglichkeit der Konfiguration angeboten.

Deister

Die Konfigurationsdaten können für Deister RFID Reader sowohl als Struktur als auch als Register verwendet werden.

Falls ein Register (Byte Array) verwendet wird, muss dieses immer die Größe der vollständigen Konfigurationsdaten besitzen. Bei den unterstützten Deister RDL Geräten ist dies 88 Byte und bei den UDL Geräten 117 Byte.

[ST_RFID_CfgStruct_DeisterRDL \[► 54\]](#)

[ST_RFID_CfgStruct_DeisterUDL \[► 56\]](#)

Die Strukturen sind für das Schreiben mit `eRFC_SetConfig` sowie das Lesen mit `eRFC_GetConfig` geeignet. (siehe [Befehlssatz \[► 17\]](#))

Leuze

Die Konfigurationsdaten können für Leuze RFID Reader sowohl als Struktur als auch als Register verwendet werden.

Falls ein Register (Byte Array) verwendet wird, muss dieses immer die Größe der vollständigen Konfigurationsdaten besitzen. Bei den unterstützten Leuze Geräten ist dies 88 Byte.

[ST_RFID_CfgStruct_LeuzeRFM \[► 57\]](#)

Die Struktur ist für das Schreiben mit `eRFC_SetConfig` sowie das Lesen mit `eRFC_GetConfig` geeignet. (siehe [Befehlssatz \[► 17\]](#))

Pepperl+Fuchs

Die Konfigurationsdaten werden für Pepperl+Fuchs RFID Reader als Struktur verwendet.

[ST_RFID_CfgStruct_PepperlFuchsIDENT \[► 59\]](#)

Die Struktur ist für das Lesen mit `eRFC_GetConfig` geeignet. (siehe [Befehlssatz \[► 17\]](#))

9.3 Low Level Kommunikation

Die PLC RFID Bibliothek bietet neben dem High Level [Befehlssatz \[► 17\]](#) auch die Möglichkeit der Low Level Kommunikation. Dies ist implizit gelöst. Es wird derselbe Funktionsbaustein verwendet.

Es können beliebige Telegramme bis zu einer maximalen Größe von 1024 Bytes empfangen und bis zu einer Größe von 300 Bytes versendet werden.

Ein gesamtes Telegramm setzt sich dabei wie folgt zusammen:

| Prefix | Adressierung | **Rohdaten** | CRC | Suffix |

Je nach proprietärer Protokollspezifikation können einzelne Bestandteile fehlen. Generell ist die Zusammensetzung aber gleich.

Senden

Dazu wird als Befehl `eRFC_SendRawData` aus dem Befehlssatz verwendet und die zu sendenden Rohdaten in der Eingangsstruktur [ST_RFID_Control \[► 45\]](#) angegeben.

Um ein Low Level Telegramm abzusenden werden *nur* die Rohdaten angegeben. Die anderen Bestandteile des Telegramms werden automatisch von der PLC RFID Library ergänzt. Ebenso werden Prüfdaten wie CRC intern erzeugt und eingefügt.



Falls das Protokoll eine Umkodierung von bestimmten Bytes innerhalb der Rohdaten verlangt, wird dies ebenfalls automatisch von der PLC RFID Library vorgenommen und muss nicht mitbedacht werden.

i Falls es sich um eine RS485 Schnittstelle handelt, so muss die Adressierung gesondert angegeben werden. Sie darf nicht mit in den angegebenen Rohdaten enthalten sein. Per Default wird die Adressierung automatisch von der Bibliothek übernommen. Sie kann allerdings über Eingangsvariablen in [ST_RFID_Control \[▶ 45\]](#) parametrisiert werden.

Die zuletzt gesendeten Rohdaten können jederzeit am Ausgang des Funktionsbausteines mit Hilfe der Struktur [ST_RFID_RawData \[▶ 45\]](#) eingesehen werden. Dies ist unabhängig vom verwendeten Befehl.

Empfangen

Die zuletzt empfangenen Rohdaten können jederzeit am Ausgang des Funktionsbausteines mit Hilfe der Struktur [ST_RFID_RawData \[▶ 45\]](#) eingesehen werden.

Die zugehörige Adressierung ist in der Struktur [ST_RFID_ReaderInfo \[▶ 43\]](#) ausgegeben.

Bei Nutzung des Kommandos `SendRawData` kann eine direkte Auswertung der empfangenen Antwort nicht garantiert werden.

Beispiel: Wird ein Lesebefehl manuell als Bytefolge mittels des Kommandos `SendRawData` versandt, so werden empfangene Transponderdaten nicht auf eine in [ST_RFID_AccessData \[▶ 51\]](#) angegebene Adresse geschrieben. Eine Auswertung/Speicherung der Daten sollte demnach auch manuell mit Hilfe der immer angegebenen Rohdaten Struktur [ST_RFID_RawData \[▶ 45\]](#) geschehen. Bei den angegebenen Rohdaten handelt es sich um das empfangene Telegramm, allerdings ohne Prefix, Suffix, Checksum, CRC oder Shift-Sequence-Kodierung.

Falls das empfangene Telegramm nicht regulär von dem Funktionsbaustein ausgewertet wurde, wird dies zudem über einen Fehler signalisiert.

Um die empfangenen Daten nutzen zu können, müssen diese beispielsweise auf ein Bytearray kopiert werden.

Beispiel einer Zuweisung von empfangenen Daten mit Hilfe der Funktion `MEMCPY()`:

```
fbRFIDReader      : FB_RFIDReader;
arrReceivedData  : ARRAY [0..511] OF BYTE;
MEMSET( ADR(arrReceivedData), 0, SIZEOF(arrReceivedData) );
MEMCPY( ADR(arrReceivedData), fbRFIDReader.stRawData.pReceivedRsp, MIN(fbRFIDReader.stRawData.iReceivedRspLen, SIZEOF(arrReceivedData)) );
```

i Balluff RFID Reader: Die Endekennung (LF CR) wird als Suffix zur Erkennung von Telegrammen genutzt. Sobald diese 2 Bytes im Datenstrom erkannt werden, werden vorherige Daten zu einem Telegramm zusammengefasst.

10 Datentypen

10.1 Strukturen

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
    
```

Die Struktur ST_RFID_TranspInfo gibt den Typ [▶ 63] und die Seriennummer [▶ 64] des zuletzt detektierten RFID Transponders an.

Mit dem Befehl 'GetInventory' werden diese Informationen abgefragt und aktualisiert.

alle Hersteller:

sSerialNumber	Die Seriennummer des Transponders (häufig 8 Byte) wird im String <i>sSerialNumber</i> in hexadezimaler Darstellung angegeben. Bei Verwendung von Balluff RFID Readern wird die Seriennummer bei 13,56Mhz Transpondern im Gesamten byteweise vom Bibliotheksbaustein gedreht. Dies geschieht, weil die ausgelesene Seriennummer eines Transponders andernfalls nicht mit der an einem anderen Reader ausgelesenen Seriennummer übereinstimmen würde. So lassen sich Geräte verschiedener Hersteller gemeinsam in einem Verbund betreiben.
eType	Der Typ des Transponders wird als Enumerationswert der Enumeration E_RFID_TranspType angegeben.

Pepperl+Fuchs:

iHeadNumber	Falls ein RFID Reader mit mehreren Leseköpfen angeschlossen ist, so gibt <i>iHeadNumber</i> an von welchem Kopf der RFID Transponder detektiert wurde.
--------------------	--------------------------------------------------------------------------------------------------------------------------------------------------------

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
    
```

Nach dem Befehl GetReaderVersion aus dem Befehlssatz [▶ 17] werden die empfangenen Daten in dieser Ausgangsstruktur aufbereitet.

Nicht jede Variable wird dabei von jedem RFID Reader Modell in Form der Versionsinformation bedient. So gibt ein Reader Modell beispielsweise das Produktionsdatum bekannt, während ein anderes Reader Modell die Hardwareversion übermittelt.

Nähere Informationen zu den Angaben finden sich in den herstellereigenen Protokollspezifikationen und -handbüchern.

alle Hersteller:

eType	An diesem Ausgang wird der RFID Reader Typ als Enumeration angegeben.
eGroup	An diesem Ausgang wird die RFID Reader Gruppe/Serie angegeben. Durch diese Gruppenzuordnung wird die bibliotheksinterne Verarbeitung aller Telegramme festgelegt.
eManufacturer	An diesem Ausgang wird der Hersteller des angeschlossenen RFID Readers angegeben. Die Enumeration gibt folgende Auswahl: TYPE E_RFID_ReaderManufacturer : (eRFRM_Unknown, eRFRM_Balluff, eRFRM_Deister, eRFRM_Leuze, eRFRM_PepperlFuchs, eRFRM_Baltech); END_TYPE
iSrcAddrRcv	Im Falle der RS485 Schnittstelle wird hier die empfangene Quelladresse angegeben.
iDstAddrRcv	Im Falle der RS485 Schnittstelle wird hier die empfangene Zieladresse angegeben.

Baltech:

dDate	An diesem Ausgang wird das Produktionsdatum des RFID Readers angegeben. Das Datum 01.01.1970 bedeutet, dass kein Produktionsdatum übermittelt wurde.
sSWVersion	Gibt die Softwareversion als Text an.
sCode	Der spezielle Typ des RFID Readers wird als Zahlencode übermittelt. Dieser ist am Ausgang <i>sCode</i> als String ausgegeben.
sSerialNumber	Die Seriennummer des RFID Readers wird am Ausgang <i>sSerialNumber</i> als String in hexadezimaler Darstellung ausgegeben. Diese ist nicht zu verwechseln mit der Transponderseriennummer.

Deister:

sSWVersion	Gibt die Softwareversion als Text an.
sHWVersion	Gibt die Hardwareversion als Text an.
sCode	Der spezielle Typ des RFID Readers wird als Zahlencode übermittelt. Dieser ist am Ausgang <i>sCode</i> als String ausgegeben.
sSerialNumber	Die Seriennummer des RFID Readers wird am Ausgang <i>sSerialNumber</i> als String in hexadezimaler Darstellung ausgegeben. Diese ist nicht zu verwechseln mit der Transponderseriennummer.

Leuze:

dDate	An diesem Ausgang wird das Produktionsdatum des RFID Readers angegeben. Das Datum 01.01.1970 bedeutet, dass kein Produktionsdatum übermittelt wurde.
sCode	Der spezielle Typ des RFID Readers wird als Zahlencode übermittelt. Dieser ist am Ausgang <i>sCode</i> als String ausgegeben.

Pepperl+Fuchs:

dDate	An diesem Ausgang wird das Produktionsdatum des RFID Readers angegeben. Das Datum 01.01.1970 bedeutet, dass kein Produktionsdatum übermittelt wurde.
sSWVersion	Gibt die Softwareversion als Text an.
sCode	Der spezielle Typ des RFID Readers wird als Zahlencode übermittelt. Dieser ist am Ausgang <i>sCode</i> als String ausgegeben.

10.1.3 ST_RFID_RawData

```

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

  iReceivedRspLen   :UINT;
  iSentCommandLen   :UINT;
END_STRUCT
END_TYPE

```

Diese Struktur gibt die gesendeten und empfangenen Daten als Rohdaten aus. Dabei handelt es sich immer um das komplette Telegramm, allerdings ohne Prefix, Suffix, Checksum, CRC oder Shift-Sequence-Kodierung. Die Low Level Kommunikation ist näher im Kapitel [Low Level Kommunikation](#) [► 41] beschrieben.

Über die angegebenen Pointer können die Bytefolgen eingesehen werden.



Zur Auswertung kann die Funktion MEMCPY genutzt werden.

pReceivedRsp : Ein empfangenes Telegramm ist als Bytefolge abgelegt und der Pointer *pReceivedRsp* zeigt auf diese Bytefolge.

pSentCommand : Ein gesendetes Telegramm ist als Bytefolge abgelegt und der Pointer *pSentCommand* zeigt auf diese Bytefolge.

iReceivedRspLen : Gibt die Länge der abgelegten Bytefolge in Bytes an.

iSentCommandLen : Gibt die Länge der abgelegten Bytefolge in Bytes an.

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 addresses are handled automatically and set addresses are not used. *)

  bLogging           :BOOL;

  iHeadNumber        :USINT := 1;           (* if multiple read heads are installed at the reader unit, 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 VHL file of vhlSetup) *)

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

```

Die Eingangsstruktur *stCtrl* beinhaltet Variablen, u.a. die [Seriennummer](#) [► 64], mit denen sich das Verhalten des Funktionsbausteines parametrieren lässt.

Die zwei Variablen *tPreSendDelay* und *tPostSendDelay* bieten optional die Möglichkeit Verzögerungszeiten zu parametrieren. Weitere Hinweise finden sich am Ende der Seite.

● RS485

Eine Unterstützung mehrerer RFID Reader an einem RS485 Netz ist mit der PLC RFID Library vorerst nicht gegeben. Es muss je RFID Reader eine separate Verbindung zu einer separaten Klemme erfolgen. In diesem Stand Alone Betrieb ist eine individuelle Adressierung mit *iSrcAddrSnd* und *iDstAddrSnd* nicht nötig. Die Adressierung kann demnach automatisch von der PLC RFID Library übernommen werden, wozu die Eingangsvariable *bAddrAutoMode* auf TRUE belassen werden kann.

alle Hersteller:

tPreSendDelay	Vor dem Senden eines Kommandos an den RFID Reader wartet der Funktionsbaustein die mit <i>tPreSendDelay</i> angegebene Zeit ab.																								
tPostSendDelay	Nach dem Senden eines Kommandos an den RFID Reader wartet der Funktionsbaustein mindestens die mit <i>tPostSendDelay</i> angegebene Zeit ab. Falls bis dahin die erwartete Response noch nicht eingetroffen ist, wartet der Funktionsbaustein weiter auf die Response bis maximal die angegebene Timeout Zeit <i>tTimeOut</i> abgelaufen ist.																								
iSrcAddrSnd	Quelladresse im Falle der RS485 Kommunikation. Diese Adresse wird verwendet, falls <i>bAddrAutoMode</i> nicht gesetzt ist.																								
iDstAddrSnd	Zieladresse im Falle der RS485 Kommunikation. Diese Adresse wird verwendet, falls <i>bAddrAutoMode</i> nicht gesetzt ist.																								
bAddrAutoMode	Auswahlmöglichkeit im Falle der RS485 Kommunikation. Falls <i>bAddrAutoMode</i> gesetzt ist (Default = TRUE), werden die RS485 Adressen automatisch zugewiesen. Die oben angegebenen Adressen werden nicht verwendet. Falls <i>bAddrAutoMode</i> deaktiviert (FALSE) ist, so werden oben angegebene Adressen verwendet.																								
bLogging	<p>Mit <i>bLogging</i> lässt sich eine zusätzliche Ausgabe aktivieren. Mit Hilfe von Log View Messages kann so die serielle Kommunikation nachvollzogen werden. Diese Messages werden im Windows Event Viewer sowie im TwinCAT System Manager sichtbar. Dies ist sinnvoll für Test- und Analysezwecke. Das Format der Ausgabe ist nicht festgelegt, um Erweiterungen zu ermöglichen.</p> <table border="1"> <thead> <tr> <th>Server (Port)</th> <th>Timestamp</th> <th>Message</th> </tr> </thead> <tbody> <tr> <td>TCPLC.PlcAuxTask (801)</td> <td>27.10.2010 11:29:12 50 ms</td> <td>TcPclLibrary RFID (Baltech IDE LP) Rx data (3 bytes): 00007F</td> </tr> <tr> <td>TCPLC.PlcAuxTask (801)</td> <td>27.10.2010 11:29:12 46 ms</td> <td>TcPclLibrary RFID (Baltech IDE LP) Rx data (16 bytes): 1C01020D0048616C6C6F2057656C7421</td> </tr> <tr> <td>TCPLC.PlcAuxTask (801)</td> <td>27.10.2010 11:29:12 2 ms</td> <td>TcPclLibrary RFID (Baltech IDE LP) Tx data (11 bytes): 1C01020500FF000A000DE2</td> </tr> <tr> <td>TCPLC.PlcAuxTask (801)</td> <td>27.10.2010 11:29:11 57 ms</td> <td>TcPclLibrary RFID (Baltech IDE LP) Rx data (6 bytes): 1C010300001E</td> </tr> <tr> <td>TCPLC.PlcAuxTask (801)</td> <td>27.10.2010 11:29:10 993 ms</td> <td>TcPclLibrary RFID (Baltech IDE LP) Tx data (23 bytes): 1C01031100FF000A000C48616C6C6F2057656C7421009B</td> </tr> <tr> <td>TCPLC.PlcAuxTask (801)</td> <td>27.10.2010 11:28:54 610 ms</td> <td>TcPclLibrary RFID (Baltech IDE LP) Rx data (10 bytes): 1C0101040057F4E98BD9</td> </tr> <tr> <td>TCPLC.PlcAuxTask (801)</td> <td>27.10.2010 11:28:54 602 ms</td> <td>TcPclLibrary RFID (Baltech IDE LP) Tx data (6 bytes): 1C010100001C</td> </tr> </tbody> </table> <p>Beim ersten Aufruf des Funktionsbausteines mit <i>bLogging=TRUE</i> wird die Ausgabe 'Logging initialized.' erzeugt. In diesem ersten Zyklus werden keine Kommunikationsdaten überwacht.</p>	Server (Port)	Timestamp	Message	TCPLC.PlcAuxTask (801)	27.10.2010 11:29:12 50 ms	TcPclLibrary RFID (Baltech IDE LP) Rx data (3 bytes): 00007F	TCPLC.PlcAuxTask (801)	27.10.2010 11:29:12 46 ms	TcPclLibrary RFID (Baltech IDE LP) Rx data (16 bytes): 1C01020D0048616C6C6F2057656C7421	TCPLC.PlcAuxTask (801)	27.10.2010 11:29:12 2 ms	TcPclLibrary RFID (Baltech IDE LP) Tx data (11 bytes): 1C01020500FF000A000DE2	TCPLC.PlcAuxTask (801)	27.10.2010 11:29:11 57 ms	TcPclLibrary RFID (Baltech IDE LP) Rx data (6 bytes): 1C010300001E	TCPLC.PlcAuxTask (801)	27.10.2010 11:29:10 993 ms	TcPclLibrary RFID (Baltech IDE LP) Tx data (23 bytes): 1C01031100FF000A000C48616C6C6F2057656C7421009B	TCPLC.PlcAuxTask (801)	27.10.2010 11:28:54 610 ms	TcPclLibrary RFID (Baltech IDE LP) Rx data (10 bytes): 1C0101040057F4E98BD9	TCPLC.PlcAuxTask (801)	27.10.2010 11:28:54 602 ms	TcPclLibrary RFID (Baltech IDE LP) Tx data (6 bytes): 1C010100001C
Server (Port)	Timestamp	Message																							
TCPLC.PlcAuxTask (801)	27.10.2010 11:29:12 50 ms	TcPclLibrary RFID (Baltech IDE LP) Rx data (3 bytes): 00007F																							
TCPLC.PlcAuxTask (801)	27.10.2010 11:29:12 46 ms	TcPclLibrary RFID (Baltech IDE LP) Rx data (16 bytes): 1C01020D0048616C6C6F2057656C7421																							
TCPLC.PlcAuxTask (801)	27.10.2010 11:29:12 2 ms	TcPclLibrary RFID (Baltech IDE LP) Tx data (11 bytes): 1C01020500FF000A000DE2																							
TCPLC.PlcAuxTask (801)	27.10.2010 11:29:11 57 ms	TcPclLibrary RFID (Baltech IDE LP) Rx data (6 bytes): 1C010300001E																							
TCPLC.PlcAuxTask (801)	27.10.2010 11:29:10 993 ms	TcPclLibrary RFID (Baltech IDE LP) Tx data (23 bytes): 1C01031100FF000A000C48616C6C6F2057656C7421009B																							
TCPLC.PlcAuxTask (801)	27.10.2010 11:28:54 610 ms	TcPclLibrary RFID (Baltech IDE LP) Rx data (10 bytes): 1C0101040057F4E98BD9																							
TCPLC.PlcAuxTask (801)	27.10.2010 11:28:54 602 ms	TcPclLibrary RFID (Baltech IDE LP) Tx data (6 bytes): 1C010100001C																							
pRawDataCommand	Der reguläre Funktionsbaustein der PLC RFID Library kann ebenso zur LowLevel Kommunikation verwendet werden. In dem Fall können Rohdaten direkt an den RFID Reader gesendet werden. Die zu sendenden Rohdaten (z.B. als Bytearray) müssen in dieser Eingangsvariablen angegeben werden. Dabei handelt es sich um einen Pointer auf die zu sendenden Rohdaten. Auf diese Adresse wird von der TcRFID Bibliothek nur lesend zugegriffen. Weitere Informationen zum Ablauf der Low Level Kommunikation sind in dem Kapitel Low Level Kommunikation [► 41] zusammengefasst. Am Ausgang können die gesendeten sowie die empfangenen Rohdaten jederzeit mittels der Struktur ST RFID RawData [► 45] eingesehen werden.																								
iRawDataCommandLen	Die Eingangsvariable <i>iRawDataCommandLen</i> gibt die Länge in Bytes der mittels dem Pointer <i>pRawDataCommand</i> angegebenen Rohdaten an.																								

Balluff:

iHeadNumber	Falls ein RFID Lesegerät mit mehreren Leseköpfen angesprochen wird, so wird in der Eingangsvariablen <i>iHeadNumber</i> eine Wahl des Lesekopfes angegeben.
--------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------

iDCType	Es kann bei Verwendung von Balluff Readern über <i>iDCType</i> die Blockgröße des Chipspeichers (0->64bytes, 1->32bytes) angegeben werden.
----------------	--------------------------------------------------------------------------------------------------------------------------------------------

Baltech:

iVHLFileID	Schreib- und Lesebefehle werden mit der hier ausgewählten VHL Datei ausgeführt. Diese muss in der Konfiguration des RFID Gerätes hinterlegt sein. Falls <i>iVHLFileID</i> den Wert 0xFF [default] hat, so wird der RFID Leser keine normale VHL Datei aus seiner Konfiguration nehmen, sondern eine einfache Ad-Hoc-Datei, welche gewählt werden kann, um nicht konfigurierte/unverschlüsselte Blankokarten zu verwenden.																					
iCardTypeMask [ab Bibliothek v.1.2.4]	Mit Hilfe der <i>iCardTypeMask</i> lassen sich bestimmte Transponderkartenfamilien einstellen. Nur die gesetzten Kartentypen werden mit dem Befehl GetInventory erkannt. Alle anderen Kartentypen werden herausgefiltert. Sollen alle Kartentypen erkannt werden, muss dieser Wert nicht verändert werden [default: 0xFFFF]. Für jede akzeptierte Kartenfamilie wird in der <i>iCardTypeMask</i> das jeweilige Bit gesetzt.																					
	<table border="1"> <thead> <tr> <th>Card Type</th> <th>Card Family</th> <th>CardTypeMask</th> </tr> </thead> <tbody> <tr> <td>Mifare / ISO 14443-A</td> <td>1</td> <td>0x0001 (Bit 1)</td> </tr> <tr> <td>LEGIC</td> <td>2</td> <td>0x0002 (Bit 2)</td> </tr> <tr> <td>ISO 15693</td> <td>3</td> <td>0x0004 (Bit 3)</td> </tr> <tr> <td>ISO 14443-B</td> <td>4</td> <td>0x0008 (Bit 4)</td> </tr> <tr> <td>PICOPASS via ISO 14443-2-B</td> <td>5</td> <td>0x0010 (Bit 5)</td> </tr> <tr> <td>PICOPASS via ISO 15693</td> <td>6</td> <td>0x0020 (Bit 6)</td> </tr> </tbody> </table>	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)
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 [ab Bibliothek v.1.2.4]	Wenn <i>bReselect</i> gesetzt ist [default: TRUE], wird bei jeder Abfrage mit dem Befehl GetInventory ein Transponder am Ausgang angegeben, sofern sich ein Transponder im HF Feld befindet. Falls sichergestellt ist, dass sich jeweils nur eine Karte vor dem RFID Reader befindet, sollte diese Einstellung beibehalten werden. Wenn sich mehr als eine Transponderkarte im HF Feld befindet, existiert im RFID Gerät eine Liste erkannter Karten. Um alle Transponder mit dem Befehl GetInventory anwählen zu können, sollte <i>bReselect</i> ausgeschaltet [FALSE] werden. Der Befehl GetInventory geht diese Liste dann durch mehrfaches Aufrufen einem nach dem anderen durch. Nachdem die letzte Karte angewählt wurde, folgt die Rückmeldung <i>eRFR_NoTransponder</i> . Erst wenn eine Karte aus den HF Feld herausgenommen und erneut hineingeführt wird, gibt der Befehl GetInventory den Transponder wieder samt Seriennummer am Ausgang an.																					
bAcceptConfCard [ab Bibliothek v.1.2.4]	Falls <i>bAcceptConfCard</i> gesetzt ist [default: TRUE], werden mit dem Befehl GetInventory auch Konfigurationskarten erkannt. Es wird automatisch versucht die darauf vorhandene Konfiguration auf den RFID Reader zu übertragen. Um eine Beeinflussung durch fremde Konfigurationskarten zu verhindern kann dieser Parameter deaktiviert werden.																					

Leuze:

sSelTranspSRN	An der Eingangsvariablen <i>sSelTranspSRN</i> kann die Seriennummer des Transponders, auf den der Befehl (z.B. Read/Write) angewendet werden soll, als String angegeben werden. Dies ist in den meisten Fällen nicht notwendig. Falls eine bestimmte Reader-Konfiguration dies notwendig macht, sind Details hierzu bei der diesbezüglichen Beschreibung in der proprietären Spezifikation zu finden.
----------------------	-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Pepperl+Fuchs:

iHeadNumber	Falls ein RFID Lesegerät mit mehreren Leseköpfen angesprochen wird, so wird in der Eingangsvariablen <i>iHeadNumber</i> eine Wahl des Lesekopfes angegeben. <i>iHeadNumber</i> wird auch als IdentChannel bezeichnet.
iDCType	Es kann bei Pepperl+Fuchs Readern mit dieser Variablen und dem Befehl 'ChangeDCType' der Transpondertyp am Lesekopf eingestellt werden. Mögliche Werte dafür sind im proprietären Protokoll beim Befehl 'ChangeTag' gelistet. (Sie stimmen nicht mit den Werten der Enumeration E_RFID_TranspType überein.) Auszug unterstützter Transpondertypen: 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 Die meisten Befehle werden umgehend nach dem Aufruf einmal bearbeitet. Je nach RFID Reader können Befehle dauerhaft anstehen. Dies ermöglicht unter anderem einen Lesevorgang, sobald der RFID Transponder ins Lesefeld kommt, ohne einen extra Befehl dazu abzuschicken. Entweder ist dies in der RFID Reader Konfiguration konfigurierbar (Balluff, Deister, Leuze) oder mit dieser Eingangsvariablen auswählbar (Pepperl+Fuchs).

Ist an einem Lesekopf ein solch gepufferter Befehl aktiv, darf der Trigger Mode nicht für diesen Kanal aktiviert werden bzw. aktiv sein! Ebenso darf kein Rohdatenbefehl abgesetzt werden, welcher diesen Kanal betrifft!

Verzögerungszeiten

Die zwei Variablen *tPreSendDelay* und *tPostSendDelay* bieten optional die Möglichkeit Verzögerungszeiten zu parametrieren.

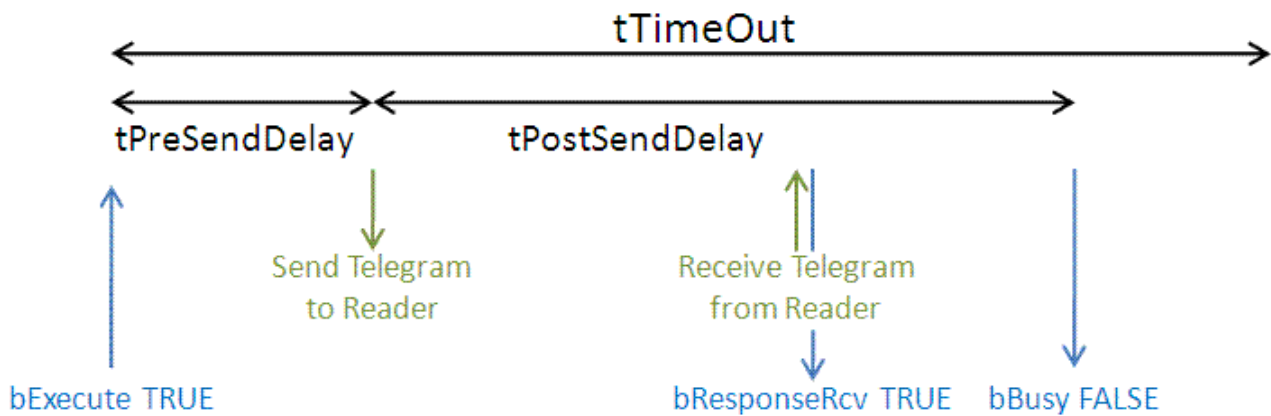
Beide Variablen stellen sicher, dass zwischen zwei Anfragen an den RFID Reader eine Verzögerung abgewartet wird. Wird die Verzögerungszeit als *tPreSendDelay* angegeben, so ist eine Verzögerung zwischen dem letzten Antworttelegramm und dem nächsten Anfragetelegramm sichergestellt. Soll das Anfragetelegramm möglichst direkt abgeschickt werden, so kann *tPostSendDelay* verwendet werden. Es gilt die Bedingung ' $tTimeOut > tPreSendDelay + tPostSendDelay$ '. Andernfalls wird ein Fehler am Ausgang ausgegeben.



Im proprietären Protokoll der Balluff RFID Reader ist eine Mindestverzögerungszeit zwischen zwei Befehlen von 300 ms spezifiziert.



Im proprietären Protokoll der Leuze electronic RFID Reader ist eine Mindestverzögerungszeit zwischen Empfang und Befehl von 150 ms spezifiziert.



10.1.5 ST_RFID_ConfigIn

```

TYPE ST_RFID_ConfigIn :
(* defines the configuration input parameters.
The data can be set via Config structure or Config register.
Different RFID Reader in different ReaderGroups can differ in their configuration data. *)
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 CfgStructure *)
)

(* 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
    
```

Am Eingang des RFID Funktionsbausteines gibt diese Struktur die Möglichkeit eine beliebige Konfiguration auf den RFID Reader zu übertragen.

Die zuletzt gelesene RFID Reader Konfiguration ist am Ausgang mit der Struktur **ST_RFID_Config** [► 50] angegeben. In deren Beschreibung finden sich auch ergänzende Informationen zu den Konfigurationen. Konfigurationsdaten können in Form einer spezifischen Konfigurationsstruktur (ST_RFID_CfgStruct_DeisterUDL, ST_RFID_CfgStruct_LeuzeRFM, ...) oder auch in Form eines Konfigurationsregisters (Byte Array) vorliegen. Mit der Variable **bUseCfgReg** lässt sich diese Auswahl treffen.

Weitere Informationen zur RFID Reader Konfiguration sind im [Kapitel Konfiguration](#) [► 39] zusammengefasst.

Baltech:

pCfg	Dieser Pointer muss die Speicheradresse der zu schreibenden Konfiguration beinhalten. Dabei handelt es sich um die Konfigurationsstruktur ST_RFID_CfgStruct_BaltechMifVHLFile .
iCfgSize	Diese Eingangsvariable gibt die Länge in Bytes der über den Pointer angegebenen Konfigurationsdaten an.

Deister:

pCfg	Dieser Pointer muss die Speicheradresse der zu schreibenden Konfiguration beinhalten. Dabei kann es sich um eine Konfigurationsstruktur sowie auch ein Konfigurationsregister handeln.
-------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

iCfgSize	Diese Eingangsvariable gibt die Länge in Bytes der über den Pointer angegebenen Konfigurationsdaten an.
bUseCfgReg	Wird die Eingangsvariable <i>bUseCfgReg</i> gesetzt (TRUE), so kann über den Pointer <i>pCfg</i> ein Konfigurationsregister (Byte Array) adressiert werden anstatt einer Konfigurationsstruktur. Per Default wird eine spezifische Konfigurationsstruktur angegeben.
bUseCfgDefault	Dieser Parameter ist nur relevant, wenn die Konfigurationsdaten in Form einer spezifischen Konfigurationsstruktur anliegen. Eine Konfigurationsstruktur ist keine namentliche Gesamtdarstellung des Konfigurationsregister. Die Struktur beinhaltet nur die wichtigsten Konfigurationsparameter. Wird die Eingangsvariable <i>bUseCfgDefault</i> gesetzt (TRUE), so werden für die nicht angegebenen Konfigurationsparameter Standardwerte verwendet. Andernfalls wird der Wert dieser Konfigurationsparameter nicht verändert, weil die zuletzt gelesenen Werte wiederverwendet werden.

Leuze:

pCfg	Dieser Pointer muss die Speicheradresse der zu schreibenden Konfiguration beinhalten. Dabei kann es sich um eine Konfigurationsstruktur sowie auch ein Konfigurationsregister handeln.
iCfgSize	Diese Eingangsvariable gibt die Länge in Bytes der über den Pointer angegebenen Konfigurationsdaten an.
bUseCfgReg	Wird die Eingangsvariable <i>bUseCfgReg</i> gesetzt (TRUE), so kann über den Pointer <i>pCfg</i> ein Konfigurationsregister (Byte Array) adressiert werden anstatt einer Konfigurationsstruktur. Per Default wird eine spezifische Konfigurationsstruktur angegeben.
bUseCfgDefault	Dieser Parameter ist nur relevant, wenn die Konfigurationsdaten in Form einer spezifischen Konfigurationsstruktur anliegen. Eine Konfigurationsstruktur ist keine namentliche Gesamtdarstellung des Konfigurationsregister. Die Struktur beinhaltet nur die wichtigsten Konfigurationsparameter. Wird die Eingangsvariable <i>bUseCfgDefault</i> gesetzt (TRUE), so werden für die nicht angegebenen Konfigurationsparameter Standardwerte verwendet. Andernfalls wird der Wert dieser Konfigurationsparameter nicht verändert, weil die zuletzt gelesenen Werte wiederverwendet werden.

Folgende Konfigurationsvariante ist nur für Leuze electronic RFID Reader verfügbar.

iRegIdx	Wird an <i>iRegIdx</i> ein spezieller Index angegeben, so wird alleinig dieser Index des Konfigurationsregisters geändert/gelesen. Dazu muss <i>iRegGroup</i> zudem als SingleRegister angegeben sein.
iRegGroup	Drei Werte stehen zur Verfügung: 0 um das gesamte Konfigurationsregister zu ändern/lesen; 1 um die Indizes 16#00-16#0F des Registers zu ändern/lesen; 2 um einen einzelnen Index des Registers zu ändern/lesen, wozu dieser mit <i>iRegIdx</i> angegeben werden muss.

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

```

Die Struktur gibt die zuletzt gelesene RFID Reader Konfiguration an. Dabei handelt es sich nicht um die Parametrierung der PLC RFID Bibliothek, sondern um die proprietäre Konfiguration des RFID Readers. Diese kann mit dem Befehl *eRFC_GetConfig* abgefragt werden (siehe [Befehlssatz](#) [\[17\]](#)).

Jede Konfiguration kann als Register (Byte Array) oder als Struktur gesehen werden. So gibt es in der PLC RFID Bibliothek verschiedene Konfigurationsstrukturen (ST_RFID_CfgStruct_DeisterUDL, ST_RFID_CfgStruct_LeuzeRFM, ...), welche die Rohdaten der Konfigurationsregister unterschiedlicher RFID

Reader aufarbeiten. Am Ausgang des Funktionsbausteines der Bibliothek werden beide Varianten zur Verfügung gestellt. Dies geschieht über Pointer. Zur weiteren Auswertung kann die Funktion MEMCPY() genutzt werden mit der angegebenen Datenlänge in Bytes.

pCfgStruct : Dieser Pointer gibt die Speicheradresse der spezifischen Konfigurationsstruktur an.

pCfgReg : Dieser Pointer gibt die Speicheradresse des spezifischen Konfigurationsregisters an.

iCfgStructSize : Diese Ausgangsvariable gibt die Länge in Bytes der spezifischen Konfigurationsstruktur an.

iCfgRegSize : Diese Ausgangsvariable gibt die Länge in Bytes des spezifischen Konfigurationsregisters an. Falls *iCfgRegSize* = 0 sind die Konfigurationsdaten nicht als Register (Byte Array) verfügbar.

Weitere Informationen zur RFID Reader Konfiguration sind im [Kapitel Konfiguration](#) [▶ 39] zusammengefasst.

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
    
```

Falls ein Lese- oder Schreibbefehl ausgeführt werden soll, ist es notwendig die Eingangsstruktur *stAccessData* anzugeben.

Mit dieser Struktur wird angegeben, wie viele und welche Daten gelesen und wo diese abgespeichert werden sollen bzw. wie viele und welche Daten geschrieben werden sollen.

pData	Der Pointer <i>pData</i> zeigt auf die zu schreibenden Daten bzw. auf den freien Speicherplatz für die zu lesenden Daten.
iDataSize	Gibt die Größe der zu schreibenden/lesenden Daten in Bytes an.
iStartBlock	Gibt den ersten Blockindex an, ab dem gelesen bzw. auf den Speicher des Transponders geschrieben werden soll.
iBlockCount	Gibt die Anzahl von Blöcken an, die gelesen bzw. geschrieben werden soll.
iBlockSize	Mit der Variablen <i>iBlockSize</i> kann die Blockgröße der Nutzdaten (in Bytes) angegeben werden. Je nach Transponder und RFID Reader Modell sind hier nur bestimmte Einstellungen möglich (gebräuchlich sind z.B. 8, 4 oder 1 Byte). Dies sollte vorab aus den Transponderinformationen in Erfahrung gebracht und getestet werden. Ebenso ist die Variable <i>iBlockSize</i> in Übereinstimmung mit der Einstellung in der RFID Reader Konfiguration zu wählen. Andernfalls ist es teilweise möglich, dass ein Zugriff auf den Transponder oder die Auswertung der empfangenen Daten nicht erfolgen kann.



Falls unterschiedliche RFID Reader auf denselben Transponder zugreifen sollen, so muss der Speicherzugriff auf dem Transponder vorher getestet werden. Es ist möglich, dass ein Reader-Modell die Datenblöcke in verdrehter Bytereihenfolge auf dem Transponder ablegt im Vergleich zu einem anderen Reader Modell. Oder von einem Reader-Modell wird der gesamte Speicherbereich in umgekehrter Reihenfolge gesehen als es ein anderes Reader Modell tut. Ebenso kann die lesbare Speichergröße des Transponders zwischen verschiedenen Reader-Modellen leicht variieren. Dies ist zusätzlich abhängig vom Transpondertypen. Die RFID Library nimmt darauf keinen Einfluss. Der Anwender muss obige Eingangsparameter dementsprechend wählen.

Die mögliche Blockgröße ist abhängig vom Transpondertypen. Je nach Transponder können dessen erste Blöcke für Systemdaten wie der Seriennummer reserviert sein. Der Bereich der Nutzdaten kann dementsprechend bei Index 0 beginnen oder auch bei einem höheren Wert. Falls dies der Fall ist kann die Variable *iUserDataStartBlock* genutzt werden, um diesen zusätzlichen Parameter anzugeben und den eigentlichen Index *iStartBlock* gleich zu belassen. Intern werden beide Werte zusammenaddiert. Verschiedene RFID Reader Modelle interpretieren diesen Index allerdings teilweise unterschiedlich. Beispiel: Auf einen Lesebefehl mit Startindex 0 gibt Reader A den ersten Block Nutzdaten zurück und Reader B die Seriennummer.

iUserDataStartBlock	Mit der Variablen <i>iUserDataStartBlock</i> kann der Startblock (als Index von Blöcken) der Nutzdaten auf dem Transponder angegeben werden. Die Blockgröße ist zu beachten.
----------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

10.1.8 Konfigurationsdaten

10.1.8.1 ST_RFID_CfgStruct_BaltechMifVHLFile

[verfügbar ab Bibliothek 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 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
    
```

Die Struktur ist für das Schreiben mit dem Befehl `eRFC_SetConfig` geeignet. (siehe [Befehlssatz](#) | 171) Dabei handelt es sich nicht um die Parametrierung der PLC RFID Bibliothek, sondern um die proprietäre Konfiguration des RFID Readers.

Struktur einer Mifare Karte (bis 2 KB Speicher):

Eine Mifare Karte mit 1 KB Speicher besitzt 16 Sektoren a 64 Byte. Jeder Sektor beinhaltet 4 Blöcke. Sektor 0 besteht aus Block 0-3, Sektor 1 aus Block 4-7 und folgende Sektoren bilden sich analog dazu. In der Darstellung entspricht eine Spalte jeweils einem Sektor, während ein Kästchen einen Block a 16 Byte repräsentiert.

0	4	*	*	*	*	*	*	*	*	*	*	*	*	*	*
1	5	*	*	*	*	*	*	*	*	*	*	*	*	*	*
2	6	*	*	*	*	*	*	*	*	*	*	*	*	*	*
3	7														

Nur die mit dem Sternchensymbol (*) dargestellten Blöcke beinhalten für den Nutzer verwendbaren Speicherbereich. Die maximale Größe der Nutzdaten ist demnach 752 Bytes (47 x 16 Byte) bei einer 1024 Byte großen Mifare Karte.

iVHLFile	Mit <i>iVHLFile</i> wird die Nummer der zu konfigurierenden VHL Datei angegeben. In der Konfiguration des RFID Gerätes können mehrere VHL Dateien nebeneinander existieren.
-----------------	-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------

iNrOfKeys	Mit <i>iNrOfKeys</i> wird die erforderliche Anzahl an Schlüsseln vorgegeben. Es können 1 bis 8 Schlüssel definiert werden.
iNrOfSectors	Mit <i>iNrOfSectors</i> wird die Anzahl an Sektoren angegeben, die für Nutzdaten verwendet werden sollen. Eine 1 KB Mifare Karte besitzt 16 Sektoren (default = 16). Beispiel: Sollen nur die Sektoren 4-6 verwendet werden, so wird <i>iNrOfSectors</i> =3 angegeben.
iRC500EEPOffset	Dieser Parameter betrifft die interne Übertragung der Schlüssel innerhalb der Hardware des RFID Gerätes. Die Standardeinstellung (default = 16#FF) garantiert erhöhte Sicherheit. Eine Änderung wird nicht empfohlen.
arrKeyList	In dem Array <i>arrKeyList</i> werden alle Schlüssel hinterlegt. Ein Schlüssel ist vom Typ T_RFID_MifareKey und besteht aus 6 Byte. Beispiel: Sollen zwei Schlüssel verwendet werden, so werden bei <i>arrKeyList</i> [0] und <i>arrKeyList</i> [1] die jeweiligen Schlüssel hinterlegt. TYPE T_RFID_MifareKey : ARRAY[0..5] OF BYTE; END_TYPE
arrSectorList	In dem Array <i>arrSectorList</i> werden alle Sektoren hinterlegt, die für Nutzdaten verwendet werden sollen. Beispiel: Sollen nur die Sektoren 4-6 verwendet werden, so wird <i>arrSectorList</i> [0]=4, <i>arrSectorList</i> [1]=5, <i>arrSectorList</i> [2]=6 angegeben. Das Array ist bereits mit durchlaufender Nummerierung initialisiert. In den meisten Fällen muss deshalb hier keine Änderung vorgenommen werden.
arrRdKeyAssign	In dem Array <i>arrRdKeyAssign</i> wird der Schlüsselindex für jeden verwendeten Sektor hinterlegt. Beispiel: Bei einer 1 KB Mifare Karte sollen alle Sektoren verwendet werden (<i>iNrOfSectors</i> = 16). Zwei Schlüssel werden verwendet (<i>iNrOfKeys</i> = 2) Die erste Hälfte der Sektoren auf dieser Karte ist mit dem ersten Schlüssel (<i>arrKeyList</i> [0]) zu lesen und die zweite Hälfte mit dem zweiten Schlüssel (<i>arrKeyList</i> [1]). In dem Array <i>arrRdKeyAssign</i> müssen demnach die Indizes <code>stCfg:ST_RFID_CfgStruct_BaltechMifVHLFile:=(arrRdKeyAssign:=0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1);</code> hinterlegt werden. Soll für alle Sektoren ein Schlüssel verwendet werden, handelt es sich für alle Sektoren um den Schlüsselindex 0. Das Array ist bereits mit 0 initialisiert. Deshalb muss in diesem Fall hier keine Änderung vorgenommen werden.
arrWrKeyAssign	In dem Array <i>arrWrKeyAssign</i> wird der Schlüsselindex für jeden verwendeten Sektor hinterlegt. Beispiel: Bei einer 1 KB Mifare Karte sollen alle Sektoren verwendet werden (<i>iNrOfSectors</i> = 16). Zwei Schlüssel werden verwendet (<i>iNrOfKeys</i> = 2) Die erste Hälfte der Sektoren auf dieser Karte ist mit dem ersten Schlüssel (<i>arrKeyList</i> [0]) zu beschreiben und die zweite Hälfte mit dem zweiten Schlüssel (<i>arrKeyList</i> [1]). In dem Array <i>arrWrKeyAssign</i> müssen demnach die Indizes <code>stCfg:ST_RFID_CfgStruct_BaltechMifVHLFile:=(arrWrKeyAssign:=0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1);</code> hinterlegt werden. Soll für alle Sektoren ein Schlüssel verwendet werden, handelt es sich für alle Sektoren um den Schlüsselindex 0. Das Array ist bereits mit 0 initialisiert. Deshalb muss in diesem Fall hier keine Änderung vorgenommen werden.
bMAD_Mode	Falls MAD (Mifare Application Directory) mit AIDs (Application Identifiers) anstatt der Sektorzuweisung verwendet werden soll, so muss <i>bMAD_Mode</i> gesetzt (TRUE) werden. Als Standard wird jedoch die Sektorzuweisung verwendet (default = FALSE).
iMAD_AID	Der Eingang wird nur benötigt, falls MAD (Mifare Application Directory) verwendet wird (<i>bMAD_Mode</i> = TRUE). Am Konfigurationseingang <i>iMAD_AID</i> wird die MAD AID (Application Identifier) für die VHL Datei angegeben.

Weitere Informationen zum Ablauf der RFID Reader Konfiguration sind im [Kapitel Konfiguration](#) [▶ 39] zusammengefasst.

Detaillierte Informationen zum Thema VHL Datei und Konfiguration von Baltech RFID Geräten finden sich zudem in den herstellereigenen Dokumentationen [Mifare.pdf](#) und [ConfigurationValues.pdf](#).

10.1.8.2 ST_RFID_CfgStruct_DeisterRDL

Die Struktur ist für das Schreiben mit eRFC_SetConfig sowie das Lesen mit eRFC_GetConfig geeignet. (siehe Befehlssatz [▶ 17])

Dabei handelt es sich nicht um die Parametrierung der PLC RFID Bibliothek, sondern um die proprietäre Konfiguration des RFID Readers.

```

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

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

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

  tOutputPulseTime  : TIME                   := T#300ms;

  eTranspType       : E_RFID_TranspType      := eRFTT_TagItHfi;

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

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

<p>eOpMode</p>	<p>Die Betriebsart legt fest, welche Funktion durch einen Triggerimpuls ausgelöst wird.</p> <p>Der Befehl eRFC_TriggerOn oder ein Impuls am optionalen Triggereingang löst die hier eingestellte Aktion aus.</p> <p>Falls eRFOP_WriteData eingestellt ist, wird ein Schreibzugriff ausgeführt. Falls eRFOP_ReadData eingestellt ist, wird ein Lesezugriff ausgeführt. [default]</p> <p>Das darauffolgende Antworttelegramm wird vom Funktionsbaustein der Tc RFID Bibliothek empfangen. Eine Zuweisung von gelesenen Transponderdaten ist in dem Fall nicht gegeben. Die empfangenen Rohdaten können zur weiteren Verarbeitung dem Bausteininterface entnommen werden.</p> <p>TYPE E_RFID_OpMode : (eRFOP_WriteData, eRFOP_ReadData, eRFOP_ReadSerialNumber (* This option is not possible at all RFID Reader types. *));END_TYPE</p>
<p>eTriggerMode</p>	<p>Falls eRFTR_ImmediateRead eingestellt ist, ist das Gerät immer lesebereit. Durch diese Einstellung gilt die Triggerbedingung immer als erfüllt. [default]</p> <p>Falls eRFTR_ReadWithTrigger eingestellt ist, liest das Gerät nur bei Triggerbedingung. Dazu kann der Befehl eRFC_TriggerOn genutzt werden. (siehe Befehlssatz [▶ 17])</p> <p>Das darauffolgende Antworttelegramm wird vom Funktionsbaustein der Tc RFID Bibliothek empfangen. Eine Zuweisung von gelesenen Transponderdaten ist in dem Fall nicht gegeben. Die empfangenen Rohdaten können zur weiteren Verarbeitung dem Baustein-Interface entnommen werden.</p> <p>TYPE E_RFID_TriggerMode : (eRFTR_ImmediateRead, eRFTR_ReadWithTrigger);END_TYPE</p>
<p>eReadMode</p>	<p>Falls eRFRD_ContinuousRead eingestellt ist, liest das Gerät dauerhaft und gibt ebenfalls dauerhaft gelesene Daten aus.</p> <p>Falls eRFRD_SingleShot eingestellt ist, liest das Gerät genau einmal. [default]</p>

	TYPE E_RFID_ReadMode : (eRFRD_ContinuousRead, eRFRD_SingleShot);END_TYPE
eWriteMode	Falls eRFWR_ImmediateWrite eingestellt ist, muß der Transponder im Feld sein, um einen Schreib- oder Lesebefehl korrekt auszuführen. [default] Falls eRFWR_WriteToNextTag eingestellt ist, werden die Daten eines Schreibbefehles in den nächstfolgenden Transponder geschrieben. ('Vorspannen') TYPE E_RFID_WriteMode : (eRFWR_ImmediateWrite, eRFWR_WriteToNextTag);END_TYPE
eNetworkMode	Falls eRFNM_Network eingestellt ist, können mehrere Geräte in einem RS485 Netzwerk eingebunden sein. Falls eRFNM_StandAlone eingestellt ist, befindet sich das Gerät im Stand-Alone Betrieb. [default] Der Betrieb von mehreren Geräten innerhalb eines RS485 Netzwerkes wird von der Bibliothek nicht unterstützt. TYPE E_RFID_NetworkMode :(eRFNM_Network, eRFNM_StandAlone);END_TYPE
bSerialNumberMode	Falls <i>bSerialNumberMode</i> TRUE ist, wird die Seriennummer bei Schreib- und Lesebefehlen mit übertragen. Im Standardfall entspricht dies der zuletzt mit dem Befehl GetInventory detektierten Transponderseriennummer. Andernfalls wird die Transponderseriennummer durch Angabe in <u>ST RFID Control</u> [▶ 45] festgelegt.
bMultiTranspMode	Falls <i>bMultiTranspMode</i> TRUE ist, so ist Antikollision aktiv, wenn mehrere Transponder im Feld sind.
bOutputAutomatic	Falls <i>bOutputAutomatic</i> TRUE ist, wird der Schaltausgang automatisch geschaltet.
iBlockSize	Die Blockgröße kann als 4 Byte oder 8 Byte eingestellt werden. Sie muss mit der zum Lesen und Schreiben in <u>ST RFID AccessData</u> [▶ 51] verwendeten Blockgröße übereinstimmen.
tOutputPulseTime	Mit <i>tOutputPulseTime</i> wird die Aktionszeit des Ausganges konfiguriert. Die Impulsdauer des optionalen Ausgangssignales kann zwischen 30ms und 9000ms eingestellt werden.
eTranspType	Soll das RFID Gerät nur Transponder eines Types erkennen, kann dieser mit <i>eTranspType</i> [▶ 63] eingestellt werden. Mit dem Wert 16#FE wird keine Einschränkung vorgenommen. Folgende Werte sind möglich: eRFTT_ICode eRFTT_STmLRI512 eRFTT_TagIt eRFTT_ICodeSli eRFTT_InfineonSRF55 eRFTT_Inside eRFTT_TagItHfi
iCountBlocksRead	Mit <i>iCountBlocksRead</i> wird die Anzahl der automatisch zu lesenden Blöcke konfiguriert. Das Produkt mit <i>iBlockSize</i> ergibt die Anzahl an Bytes. Die maximale Anzahl an Blöcken beträgt je nach Blockgröße und anderen Einstellungen zwischen 4 und 9 Blöcke.
iCountBlocksWrite	Mit <i>iCountBlocksRead</i> wird die Anzahl der automatisch zu schreibenden Blöcke konfiguriert. Das Produkt mit <i>iBlockSize</i> ergibt die Anzahl an Bytes. Die maximale Anzahl an Blöcken beträgt je nach Blockgröße und anderen Einstellungen zwischen 4 und 9 Blöcke.
iStartBlockRead	Mit <i>iStartBlockRead</i> wird die Startadresse für das automatische Lesen konfiguriert.
iStartBlockWrite	Mit <i>iStartBlockRead</i> wird die Startadresse für das automatische Schreiben konfiguriert.
arrWriteData	Es können maximal 72 Bytes als Schreibdaten angegeben werden.



Es gibt Kombinationen von Werten, welche unzulässig sind. Die bestehenden Abhängigkeiten sind in der proprietären Spezifikation des Herstellers dargelegt. Falls versucht wird eine unzulässige Konfiguration zu schreiben, tritt der Fehler eRFERR_InvalidCfg ein oder es wird ein Fehlercode vom RFID Gerät empfangen.

Weitere Informationen zum Ablauf der RFID Reader Konfiguration sind im [Kapitel Konfiguration \[▶ 39\]](#) zusammengefasst.

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
    
```

Die Struktur ist für das Schreiben mit eRFC_SetConfig sowie das Lesen mit eRFC_GetConfig geeignet. (siehe [Befehlssatz \[▶ 17\]](#))

Dabei handelt es sich nicht um die Parametrierung der PLC RFID Bibliothek, sondern um die proprietäre Konfiguration des RFID Readers.



Wie bereits erwähnt muss ggf. der Unterschied zwischen Polling und Trigger beachtet werden. Hinzu kommt in diesem Kontext, dass neben dem PollingMode dennoch der TriggerMode vorhanden sein kann.

ePollingMode	<p>Falls eRFPO_PollingMode eingestellt ist, sendet das RFID Gerät nur auf Anfrage Daten. [default]</p> <p>Falls eRFPO_ReportMode eingestellt ist, darf das RFID Gerät jederzeit von sich aus Daten übertragen.</p> <p>TYPE E_RFID_PollingMode :(eRFPO_ReportMode, eRFPO_PollingMode);END_TYPE</p>
eTriggerMode	<p>Falls eRFTR_ImmediateRead eingestellt ist, ist das Gerät immer lesebereit. Durch diese Einstellung gilt die Triggerbedingung immer als erfüllt. [default]</p> <p>Falls eRFTR_ReadWithTrigger eingestellt ist, liest das Gerät nur bei Triggerbedingung. Dazu kann der Befehl eRFC_TriggerOn genutzt werden. (siehe Befehlssatz [▶ 17])</p> <p>Das darauffolgende Antworttelegramm wird vom Funktionsbaustein der Tc RFID Bibliothek empfangen. Eine Zuweisung von gelesenen Transponderdaten ist in dem Fall nicht gegeben. Die empfangenen Rohdaten können zur weiteren Verarbeitung dem Baustein-Interface entnommen werden.</p> <p>TYPE E_RFID_TriggerMode : (eRFTR_ImmediateRead, eRFTR_ReadWithTrigger);END_TYPE</p>
eOpMode	<p>Diese Betriebs-Modi sind nicht mit jedem Transpondertyp möglich.</p>

	<p>Falls eRFOP_WriteData eingestellt ist, wird ein Schreibzugriff ausgeführt sobald ein Transponder erkannt wird. Falls eRFOP_ReadData eingestellt ist, wird ein Lesezugriff ausgeführt sobald ein Transponder erkannt wird. Falls eRFOP_ReadSerialNumber eingestellt ist, wird keine Aktion ausgeführt. Der Befehl Polling liefert die Seriennummer. [default]</p> <p>Das darauffolgende Antworttelegramm wird vom Funktionsbaustein der Tc RFID Bibliothek empfangen. Eine Zuweisung von gelesenen Transponderdaten ist in dem Fall nicht gegeben. Die empfangenen Rohdaten können zur weiteren Verarbeitung dem Bausteininterface entnommen werden.</p> <p>TYPE E_RFID_OpMode : (eRFOP_WriteData, eRFOP_ReadData, eRFOP_ReadSerialNumber (* This option is not possible at all RFID Reader types. *));END_TYPE</p>
eTranspType	<p>Soll das RFID Gerät nur Transponder eines Types erkennen, kann dieser mit eTranspType [► 63] eingestellt werden. Mit dem Wert 16#FE wird keine Einschränkung vorgenommen. Folgende Werte sind möglich: eRFTT_EPC1Gen1 eRFTT_EPC1Gen2</p>
tOutputPulseTime	<p>Mit tOutputPulseTime wird die Aktionszeit des Ausgangs konfiguriert. Die Impulsdauer des optionalen Ausgangssignals kann zwischen 30ms und 9000ms eingestellt werden.</p>
bOutputLevel	<p>Mit bOutputLevel wird die Kontrolle des optionalen digitalen Ausgangs beeinflusst. Nach einem erfolgreichen Lesen kann der Ausgang auf HighLevel (bOutputLevel=TRUE) oder Lowlevel (bOutputLevel=FALSE) gesetzt werden.</p>
iCountBlocksRead	<p>Mit iCountBlocksRead wird die Anzahl der automatisch zu lesenden Blöcke konfiguriert. Das Produkt mit iBlockSize ergibt die Anzahl an Bytes. Die maximale Anzahl an Blöcken beträgt je nach Blockgröße und anderen Einstellungen zwischen 4 und 9 Blöcke. Die Blockgröße ist abhängig vom Transpondertyp.</p>
iCountBlocksWrite	<p>Mit iCountBlocksWrite wird die Anzahl der automatisch zu schreibenden Blöcke konfiguriert. Das Produkt mit iBlockSize ergibt die Anzahl an Bytes. Die maximale Anzahl an Blöcken beträgt je nach Blockgröße und anderen Einstellungen zwischen 4 und 9 Blöcke. Die Blockgröße ist abhängig vom Transpondertyp.</p>
iStartBlockRead	<p>Mit iStartBlockRead wird die Startadresse für das automatische Lesen konfiguriert.</p>
iStartBlockWrite	<p>Mit iStartBlockWrite wird die Startadresse für das automatische Schreiben konfiguriert.</p>
arrWriteData	<p>Es können maximal 32 Bytes als Schreibdaten angegeben werden.</p>

Weitere Informationen zum Ablauf der RFID Reader Konfiguration sind im [Kapitel Konfiguration \[► 39\]](#) zusammengefasst.

10.1.8.4 ST_RFID_CfgStruct_LeuzeRFM

Die Struktur ist für das Schreiben mit eRFC_SetConfig sowie das Lesen mit eRFC_GetConfig geeignet. (siehe [Befehlssatz \[► 17\]](#))

Dabei handelt es sich nicht um die Parametrierung der PLC RFID Bibliothek, sondern um die proprietäre Konfiguration des RFID Readers.

```

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

    eNetworkMode     : E_RFID_NetworkMode     := eRFNM_Network;
    bAFI             : BOOL                   := FALSE; (* not implemented; ready for future extention *)
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	<p>Die Betriebsart legt fest, welche Funktion durch einen Triggerimpuls ausgelöst wird.</p> <p>Der Befehl eRFC_TriggerOn oder ein Impuls am optionalen Triggereingang löst die hier eingestellte Aktion aus.</p> <p>Falls eRFOP_WriteData eingestellt ist, wird ein Schreibzugriff ausgeführt. Falls eRFOP_ReadData eingestellt ist, wird ein Lesezugriff ausgeführt. [default]</p> <p>Das darauffolgende Antworttelegramm wird vom Funktionsbaustein der Tc RFID Bibliothek empfangen. Eine Zuweisung von gelesenen Transponderdaten ist in dem Fall nicht gegeben. Die empfangenen Rohdaten können zur weiteren Verarbeitung dem Bausteininterface entnommen werden.</p> <p>TYPE E_RFID_OpMode : (eRFOP_WriteData, eRFOP_ReadData, eRFOP_ReadSerialNumber (* This option is not possible at all RFID Reader types. *));END_TYPE</p>
eTriggerMode	<p>Falls eRFTR_ImmediateRead eingestellt ist, ist das Gerät immer lesebereit. Durch diese Einstellung gilt die Triggerbedingung immer als erfüllt. [default]</p> <p>Falls eRFTR_ReadWithTrigger eingestellt ist, liest das Gerät nur bei Triggerbedingung. Dazu kann der Befehl eRFC_TriggerOn genutzt werden. (siehe Befehlssatz [▶ 17])</p> <p>Das darauffolgende Antworttelegramm wird vom Funktionsbaustein der Tc RFID Bibliothek empfangen. Eine Zuweisung von gelesenen Transponderdaten ist in dem Fall nicht gegeben. Die empfangenen Rohdaten können zur weiteren Verarbeitung dem Bausteininterface entnommen werden.</p> <p>TYPE E_RFID_TriggerMode : (eRFTR_ImmediateRead, eRFTR_ReadWithTrigger);END_TYPE</p>
eReadMode	<p>Falls eRFRD_ContinuousRead eingestellt ist, liest das Gerät dauerhaft und gibt ebenfalls dauerhaft gelesene Daten aus.</p> <p>Falls eRFRD_SingleShot eingestellt ist, liest das Gerät genau einmal. [default]</p> <p>TYPE E_RFID_ReadMode : (eRFRD_ContinuousRead, eRFRD_SingleShot);END_TYPE</p>
eWriteMode	<p>Falls eRFWR_ImmediateWrite eingestellt ist, muss der Transponder im Feld sein, um einen Schreib- oder Lesebefehl korrekt auszuführen. [default]</p> <p>Falls eRFWR_WriteToNextTag eingestellt ist, werden die Daten eines Schreibbefehles in den nächst folgenden Transponder geschrieben. ('Vorspannen')</p> <p>TYPE E_RFID_WriteMode : (eRFWR_ImmediateWrite, eRFWR_WriteToNextTag);END_TYPE</p>
eNetworkMode	<p>Falls eRFNM_Network eingestellt ist, können mehrere Geräte in einem RS485 Netzwerk eingebunden sein. [default]</p>

	<p>Falls eRFNM_StandAlone eingestellt ist, befindet sich das Gerät im Stand-Alone Betrieb.</p> <p>Der Betrieb von mehreren Geräten innerhalb eines RS485 Netzwerkes wird von der Bibliothek nicht unterstützt.</p> <p>TYPE E_RFID_NetworkMode :(eRFNM_Network, eRFNM_StandAlone);END_TYPE</p>
bSerialNumberMode	<p>Falls <i>bSerialNumberMode</i> TRUE ist, wird die Seriennummer bei Schreib- und Lesebefehlen mit übertragen.</p> <p>Im Standardfall entspricht dies der zuletzt mit dem Befehl GetInventory detektierten Transponder Seriennummer. Andernfalls wird die Transponder Seriennummer durch Angabe in ST_RFID_Control [▶ 45] festgelegt.</p>
bMultiTranspMode	<p>Falls <i>bMultiTranspMode</i> TRUE ist, so ist Antikollision aktiv, wenn mehrere Transponder im Feld sind.</p>
bOutputAutomatic	<p>Falls <i>bOutputAutomatic</i> TRUE ist, wird der Schaltausgang automatisch geschaltet.</p>
iBlockSize	<p>Die Blockgröße kann als 4 Byte oder 8 Byte eingestellt werden. Sie muss mit der zum Lesen und Schreiben in ST_RFID_AccessData [▶ 51] verwendeten Blockgröße übereinstimmen.</p>
tOutputPulseTime	<p>Mit <i>tOutputPulseTime</i> wird die Aktionszeit des Ausganges konfiguriert. Die Impulsdauer des optionalen Ausgangssignals kann zwischen 30 ms und 9000 ms eingestellt werden.</p>
eTranspType	<p>Soll das RFID Gerät nur Transponder eines Types erkennen, kann dieser mit eTranspType [▶ 63] eingestellt werden. Mit dem Wert 16#FE wird keine Einschränkung vorgenommen.</p> <p>Folgende Werte sind möglich:</p> <ul style="list-style-type: none"> eRFTT_ICode eRFTT_STmLRI512 eRFTT_TagIt eRFTT_ICodeSli eRFTT_InfineonSRF55 eRFTT_Inside eRFTT_TagItHfi
iCountBlocksRead	<p>Mit <i>iCountBlocksRead</i> wird die Anzahl der automatisch zu lesenden Blöcke konfiguriert. Das Produkt mit <i>iBlockSize</i> ergibt die Anzahl an Bytes. Die maximale Anzahl an Blöcken beträgt je nach Blockgröße und anderen Einstellungen zwischen 4 und 9 Blöcke.</p>
iCountBlocksWrite	<p>Mit <i>iCountBlocksWrite</i> wird die Anzahl der automatisch zu schreibenden Blöcke konfiguriert. Das Produkt mit <i>iBlockSize</i> ergibt die Anzahl an Bytes. Die maximale Anzahl an Blöcken beträgt je nach Blockgröße und anderen Einstellungen zwischen 4 und 9 Blöcke.</p>
iStartBlockRead	<p>Mit <i>iStartBlockRead</i> wird die Startadresse für das automatische Lesen konfiguriert.</p>
iStartBlockWrite	<p>Mit <i>iStartBlockWrite</i> wird die Startadresse für das automatische Schreiben konfiguriert.</p>
arrWriteData	<p>Es können maximal 72 Bytes als Schreibdaten angegeben werden.</p>



Es gibt Kombinationen von Werten, welche unzulässig sind. Die bestehenden Abhängigkeiten sind in der proprietären Spezifikation des Herstellers dargelegt. Falls versucht wird eine unzulässige Konfiguration zu schreiben, tritt der Fehler eRFERR_InvalidCfg ein oder es wird ein Fehlercode vom RFID-Gerät empfangen.

Weitere Informationen zum Ablauf der RFID Reader Konfiguration sind im [Kapitel Konfiguration \[▶ 39\]](#) zusammengefasst.

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

```

Die Struktur ist für das Lesen mit `eRFC_GetConfig` geeignet. (siehe [Befehlssatz \[▶ 17\]](#))

Dabei handelt es sich nicht um die Parametrierung der PLC RFID Bibliothek, sondern um die proprietäre Konfiguration des RFID Readers.

Das Pepper+Fuchs Gerät Ident Control Compact besteht aus einer Zentraleinheit und 1-4 Schreib-/Leseköpfen. Jedes dieser fünf Elemente erhält einen Identifikationskanal (Ident Channel), mit dem sich Befehle zu einzelnen Elementen zuordnen lassen. Im Standardfall ist die Zentraleinheit mit dem Kanal 0 versehen und die Schreib-/Leseköpfe mit den Kanälen 1-4.

Mit dem Befehl `eRFC_GetConfig` und den Ausgaben am Ausgang `stReaderCfg` können die Einstellungen für alle Identifikationskanäle überprüft werden.

tTimeout	tTimeout gibt die Dauer an, die das RFID Gerät auf weitere Zeichen eines Telegramms wartet. Hat das Gerät nach dieser Dauer keinen verständlichen Befehl erkannt, folgt eine Fehlermeldung. (Standard ist 0 ms)
iBaudrate	Mit <i>iBaudrate</i> wird die aktuell verwendete Baudrate des RFID Gerätes angezeigt. Die unterstützten RFID Geräte verfügen über eine maximale Übertragungsrate von 38400 Baud.
identChannel	Identifikationskanal der Zentraleinheit
bMultiplexMode	Im Multiplex-Modus ist eine gegenseitige Beeinflussung der Schreib-/Leseköpfe minimiert, weil immer nur ein Kopf gleichzeitig aktiv ist.
arrHeadCfg	Es existieren Geräte mit bis zu vier Schreib-/Leseköpfen. Jeder Kopf hat einen Status und einen DataCarrierType. Diese Information ist je Kopf in einer Struktur vom Typ <code>ST_RFID_HeadCfg</code> hinterlegt. Der Status der Zentraleinheit wird in <i>iErrCodeRcv</i> direkt am Ausgang von FB_RFIDReader [▶ 36] ausgegeben. Mögliche Werte für <i>iDCType</i> sind in ST_RFID_Control [▶ 45] erläutert. TYPE <code>ST_RFID_HeadCfg</code> : STRUCT eStatus : <code>E_RFID_ErrCodeRcv_PepperFuchs</code> ; iDCType : <code>USINT</code> ; (* not equal to <code>E_RFID_TranspType</code> enumeration *) iReserved : <code>USINT</code> ; END_STRUCT END_TYPE
arrTriggerCfg	Es existieren Geräte mit bis zu vier Schreib-/Leseköpfen. Jeder Kopf besitzt einen <i>identChannel</i> und <i>bTriggerMode</i> . Diese Information ist je Triggersensor in einer Struktur vom Typ <code>ST_RFID_TriggerCfg</code> hinterlegt. <i>identChannel</i> bezeichnet den Schreib- Leskopf für den der Triggersensor konfiguriert ist. Falls <i>iTriggerMode</i> TRUE ist, ist der Trigger Modus ist für einen Triggersensor aktiv. Falls zudem <i>bInverted</i> TRUE ist, so handelt es sich um ein invertiertes Triggersignal. Weitere Informationen zum Thema TriggerMode befinden sich im Kapitel RFID Reader Einstellungen Pepper+Fuchs [▶ 28]. TYPE <code>ST_RFID_TriggerCfg</code> : STRUCT iIdentChannel : <code>USINT</code> ; bTriggerMode : <code>BOOL</code> ; bInverted : <code>BOOL</code> ; bReserved : <code>BOOL</code> ; END_STRUCT END_TYPE

Weitere Informationen zum Ablauf der RFID Reader Konfiguration sind im [Kapitel Konfiguration \[▶ 39\]](#) zusammengefasst.

10.2 Enumerationen

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
```

Der Funktionsbaustein *FB_RFIDReader* der TwinCAT PLC RFID Bibliothek bietet am Eingang *eCommand* obige Enumerationswerte an.

Dabei handelt es sich um eine Auswahl an Befehlen, wie beispielsweise Lesen oder Schreiben eines Transponders. Ausführliche Erläuterungen zu den Befehlen finden sich im Kapitel [Befehlssatz \[► 17\]](#).

10.2.2 E_RFID_Response

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

Der Funktionsbaustein *FB_RFIDReader* der TwinCAT PLC RFID Bibliothek bietet am Ausgang *eResponse* obige Enumerationswerte an.

Diese werden im Folgenden kurz erläutert. Teils finden sich Analogien zu den Telegrammantworttypen der herstellerproprietären Protokolle.

Welche herstellerproprietäre MessageID der hier gelisteten Response entspricht, wird im Folgenden jeweils *kursiv* angegeben. Aufgrund der Auswertungskomplexität sind nicht alle Entsprechungen aufgeführt.

Detaillierten Aufschluss kann bei Bedarf die Rohdatendarstellung [ST_RFID_RawData \[► 45\]](#) am Ausgang des Funktionsbausteines geben.

eRFR_NoRsp :

Dieser Wert signalisiert, dass zuletzt keine Antwort eingetroffen ist.

eRFR_Unknown :

Dieser Wert gibt an, dass das angekommene Telegramm nicht einem bestimmten Typ zugeordnet werden konnte und somit auch nicht ausgewertet wurde. Meist geht dies einher mit einer Fehlermeldung (*bError = TRUE*).

eRFR_Ready :

Manche RFID Reader Modelle signalisieren ihre Betriebsbereitschaft, beispielsweise nach einem Reset, mit einem extra Telegramm. In diesem Fall nimmt *eResponse* den Wert *eRFR_Ready* an.

Entsprechung im proprietären Protokoll:

Leuze: 'S'

Pepperl+Fuchs: Status='2'

eRFR_CmdConfirmation :

Mit dieser Antwort wird der gesendete Befehl bestätigt. Dies kann bei vielen Befehlen auftreten. Ausnahmen sind die Befehle 'Konfiguration ändern' und 'Daten schreiben', denn auf diese beiden Befehle folgen spezielle Bestätigungen, welche durch die folgenden zwei Enumerationswerte repräsentiert werden.

Entsprechung im proprietären Protokoll:

Leuze: 'Q2', 'Q4'

eRFR_CfgChangeExecuted :

Wenn die RFID Reader Konfiguration geändert wurde, sendet der RFID Reader dieses Telegramm zur Bestätigung der Aktion.

Entsprechung im proprietären Protokoll:

Leuze: 'Q1'

eRFR_WriteCmdSucceeded :

Sobald Daten auf den Transponder geschrieben wurden, sendet der RFID Reader diese Bestätigung.

Entsprechung im proprietären Protokoll:

Leuze: 'Q5'

eRFR_NoTransponder :

Diese Antwort wird gegeben, falls sich kein Transponder im Lesefeld befindet. Dies wird nicht zwangsläufig als Fehler gewertet, so dass der Ausgang *bError* auch nicht gesetzt wird.

Entsprechung im proprietären Protokoll:

Leuze: '\$18'

Pepperl+Fuchs: Status='5'

eRFR_Error :

Falls ein Telegramm empfangen wurde, welches einen Fehlercode übermittelt hat, so wird *eRFR_Error* am Ausgang *eResponse* ausgegeben. Der übermittelte Fehlercode wird in der Ausgangsvariablen *iErrCodeRcv* angegeben, wozu im Kapitel [RFID Fehlercodes \[► 66\]](#) näheres beschrieben ist. Falls *eResponse* den Wert *eRFR_Error* einnimmt, so wird auch ein Fehler mittels *bError* TRUE am Ausgang des Funktionsbausteines signalisiert.

Entsprechung im proprietären Protokoll:

Balluff: <NAK>+Failurenumber

Deister: MessageErrorCode=>16#20

Leuze: 'Exx', 'Q0'

eRFR_Data_ReaderVersion :

Durch eine Versionsabfrage wird ein RFID Reader dazu aufgefordert, Modellinformationen zu senden. Diese Art von empfangenen Daten wird mit dem Wert *eRFR_Data_ReaderVersion* am Ausgang *eResponse* bezeichnet.

eRFR_Data_Config :

Eine ausgelesene RFID Reader Konfiguration wird mittels dem Enumerationswert *eRFR_Data_Config* signalisiert.

Entsprechung im proprietären Protokoll:

Leuze: 'G'

eRFR_Data_Inventory :

Dieser Telegrammtyp wird angezeigt, wenn ein Transponder erkannt wurde bzw. die Seriennummer eines Transponders ausgelesen wurde.

eRFR_Data_ReadData :

Ein Empfang von ausgelesenen Daten aus dem Transponderspeicher wird mit dem Wert *eRFR_Data_ReadData* signalisiert.

Entsprechung im proprietären Protokoll:

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_ReaderManufacturer

```
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
```

Der Datentyp beinhaltet eine RFID Transponder Seriennummer.

Die Seriennummer des Transponders (häufig 8 Byte) wird als String in hexadezimaler Darstellung angegeben. (iRFID_MAXSRNLENGTH = 51)

11 Funktionen

11.1 F_GetVersionTcRFID

Mit dieser Funktion können Versionsinformationen der SPS-Bibliothek ausgelesen werden.

FUNCTION F_GetVersionTcRFID: UINT

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

nVersionElement : Versionselement, das gelesen werden soll. Mögliche Parameter:

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

12 Fehlercodes

Fehlerausgaben werden an zwei Ausgängen des RFID PLC Funktionsbausteines zur Verfügung gestellt. Diese zwei Ausgangsvariablen `iErrorID` [► 66] und `iErrCodeRcv` [► 69] werden im Folgenden erläutert.

Error ID - iErrorID

Falls ein Fehler vorliegt, zeigt diese Ausgabe die Art des Fehlers. Folgende Liste gibt die möglichen Werte wieder.

```

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

```

Wenn `iErrorID` den Wert `eRFERR_ErrorRcv` besitzt, wurde ein Fehlercode von dem RFID Reader empfangen. Dieser empfangene Fehlercode wird in der Ausgangsvariablen `iErrCodeRcv` angegeben.



Bei einem anderen Wert für `iErrorID` kann dennoch zusätzlich vom RFID Reader ein Fehler empfangen worden sein, welcher auch an `iErrCodeRcv` angezeigt wird.

	ID (hex)	ID (dec)	Beschreibung
<code>eRFERR_TimeOutElapsed</code>	0x4001	16385	Dieser Fehler tritt ein, wenn die als Timeout am Eingang (Default = 5sec) angegebene Zeitdauer abgelaufen ist. Eine noch in Bearbeitung befindliche Aktion wird damit abgebrochen. Es kommt auch zu einem Timeout Fehler, falls die serielle Hintergrundkommunikation nicht funktioniert. Dies kann beispielsweise der Fall sein, falls die falsche Baudrate eingestellt ist oder die Taskzykluszeit nicht ausreicht, um die Daten zu verarbeiten. Nähere Informationen hierzu finden sich im Kapitel RFID Reader Anbindung [► 14] und in der Dokumentation der SPS Bibliothek Serielle Kommunikation .
<code>eRFERR_InvalidCommand</code>	0x4101	16641	Der Befehl wurde nicht ausgeführt. Der gewählte Befehl kann mit diesem RFID Reader Modell nicht ausgeführt werden. Im Befehlssatz [► 17] sind die zur Verfügung stehenden Befehle gelistet.

	ID (hex)	ID (dec)	Beschreibung
			Um sicher zu stellen, dass das vorliegende RFID Reader Modell dem Funktionsbaustein bekannt ist, muss falls möglich zuallererst der Befehl 'GetReaderVersion' ausgeführt werden.
eRFERR_IncompatibleCfg	0x4102	16642	Der Befehl wurde nicht ausgeführt. Die aktuelle RFID Reader Konfiguration (siehe Ausgangsstruktur ST_RFID_Config [► 50]) ist nicht kompatibel mit dem ausgewählten Befehl (und den dazu gehörigen Eingangsparametern). Es muss sichergestellt werden, dass die Konfiguration zuvor einmal gelesen wurde. Andernfalls kann ebenso dieser Fehlercode auftreten.
eRFERR_InvalidManufacturer	0x4103	16643	Am Eingang des generischen Funktionsbausteines <code>FB_RFIDReader</code> muss mit der Variablen <code>eManufacturer</code> der RFID Hersteller des RFID Reader Modells angegeben werden. Der Fehler <code>eRFERR_InvalidManufacturer</code> signalisiert eine ungültige Angabe.
eRFERR_InvalidTimeOutParam	0x4104	16644	Dieser Fehler wird ausgegeben, falls die Angabe der Eingangsvariablen <code>tTimeOut</code> ungültig ist. Es gilt die Bedingung ' $tTimeOut > tPreSendDelay + tPostSendDelay$ '.
eRFERR_InvalidRawDataParam	0x4105	16645	Sollen Rohdaten gesendet werden, so müssen diese am Eingang des Funktionsbausteines angegeben werden. Der Fehler <code>eRFERR_InvalidRawDataParam</code> wird ausgegeben, falls die Angabe der Eingangsvariablen <code>pRawDataCommand</code> oder <code>iRawDataCommandLen</code> ungültig ist. Näheres zum Thema im Kapitel LowLevel Kommunikation [► 41].
eRFERR_InvalidAccessData	0x4106	16646	Der Befehl wurde nicht ausgeführt, weil die am Eingang angegebenen Parameter in der Struktur ST_RFID_AccessData [► 51] ungültig sind.
eRFERR_InvalidCfg	0x4107	16647	Der Befehl 'Konfiguration ändern' wurde nicht ausgeführt, weil die angegebenen Konfigurationsdaten ungültig sind. Die Konfigurationsdaten liegen als Konfigurationsstruktur an (<code>bUseCfgReg = FALSE</code>). Im Zweifelsfall sind die genauen gültigen Wertebereiche der Daten in der proprietären Protokollspezifikation durch den Hersteller gegeben. Falls die zuletzt gelesene Konfiguration für die in der Konfigurationsstruktur nicht verfügbaren Parameter genutzt wird (<code>bUseCfgDefault = FALSE</code> in ST_RFID_ConfigIn [► 49]), muss sicher gestellt werden, dass die Konfiguration zuvor gelesen wurde. Andernfalls kann ebenso dieser Fehlercode auftreten.
eRFERR_InvalidCfgParam	0x4108	16648	Der Befehl 'Konfiguration ändern' wurde nicht ausgeführt, weil die angegebenen Eingangsparameter der Konfigurationsdaten ungültig sind. Die Konfigurationsdaten sollen als Konfigurationsregister oder Konfigurationsstruktur

	ID (hex)	ID (dec)	Beschreibung
			anliegen. Dessen Länge oder ein anderer Parameter in ST_RFID_ConfigIn [▶ 49] sind ungültig.
eRFERR_InvalidCtrlHeadNumber	0x4109	16649	Falls ein RFID Lesegerät mit mehreren Leseköpfen angesprochen wird, so wird in der Eingangsstruktur ST_RFID_Control [▶ 45] eine Wahl des Lesekopfes angegeben. Ist für den angegebenen Wert kein Lesekopf verfügbar, so wird dieser Fehlerwert ausgegeben.
eRFERR_InvalidResponse	0x4201	16897	Falls die Bytefolge der empfangenen Response keiner bekannten Nachrichtenart entspricht oder einzelne Bytes nicht die erforderlichen Werte aufweisen, wird dieser Fehler ausgegeben. Die empfangenen Daten können als Rohdatenblock am Ausgang ST_RFID_RawData [▶ 45] analysiert werden.
eRFERR_InvalidRspLen	0x4202	16898	Falls die Bytefolge theoretisch einer bekannten Nachrichtenart entspricht, die Länge allerdings nicht der Erwartung entspricht, wird diese Fehlermeldung erzeugt. Die empfangenen Daten können als Rohdatenblock am Ausgang ST_RFID_RawData [▶ 45] analysiert werden.
eRFERR_InvalidBlockSize	0x4203	16899	Tritt dieser Fehlerwert ein, so konnten die empfangenen vom Transponder gelesenen Daten nicht ausgewertet werden. Die Anzahl der empfangenen Bytes signalisiert, dass die konfigurierte Blocksize nicht mit der Eingabe beim Befehlsaufruf übereinstimmt.
eRFERR_ErrorRcv	0x4204	16900	Ein Fehlercode wurde mit der empfangenen Nachricht gesendet. Der vom RFID Reader angezeigte Fehler wird ebenso ausgegeben und von der Ausgangsvariablen <i>iErrCodeRcv</i> repräsentiert (Erläuterung am Ende dieses Kapitels).
eRFERR_ChecksumError	0x4205	16901	Je nach Protokollspezifikation wird eine Checksumme, beispielsweise eine CRC Prüfsumme, im Telegramm mitgesendet. Wird von der Steuerung ein Telegramm empfangen mit fehlerhafter Checksumme, so wird dieser Fehler ausgegeben.
eRFERR_UnknownReaderGroup	0x4401	17409	Die RFID Bibliothek teilt die RFID Reader Modelle intern in Gruppen ein. Der Fehler <i>eRFERR_UnknownReaderGroup</i> kann auftreten, falls der RFID Reader noch keiner Gruppe zugeordnet ist. Deshalb muss je nach RFID Reader Modell bei Programmstart als erste Abfrage der Befehl <code>GetReaderVersion</code> getätigt werden.
eRFERR_CreatedTelegramTooBig	0x4402	17410	Es wurde versucht ein Telegramm zu senden, welches die maximal mögliche Größe von 300 Bytes überschritten hat. Es können nur Telegramme mit bis zu 300 Bytes versendet werden.



In einigen wenigen Fällen werden vom RFID-Gerät mehrere Telegramme unmittelbar hintereinander versendet. Es ist deshalb wichtig, immer den Fehler zu erkennen und zu beheben, welcher als erstes eintraf.

Error Code Received - iErrCodeRcv

Falls vom RFID Reader ein Fehlercode mitgeschickt wird, wird dieser an der Stelle ausgegeben.



Teils werden Statusmeldungen vom RFID Reader mitgeschickt und dann an iErrCodeRcv ausgegeben, welche nicht zu einem Fehler führen (bError bleibt FALSE und iErrorID zeigt keinen Fehler).

Eine Liste möglicher Werte ist entweder der Datentypdeklaration der PLC RFID Library im Library Manager oder direkt in der Protokollspezifikation zu entnehmen.

Diesbezügliche Datentypen sind die Enumerationen *E_RFID_ErrCodeRcv_Balluff*, *E_RFID_ErrCodeRcv_Deister*, *E_RFID_ErrCodeRcv_Leuze* usw.

13 Beispiele

Die folgenden Beispiele wurden mit unterschiedlichen RFID Reader Modellen entwickelt. Weil grundsätzlich kaum Unterschiede in der Handhabung der RFID Reader mit der TwinCAT SPS Bibliothek bestehen, kann auch ein Beispiel welches mit einem anderen Modell entwickelt wurde, zur Einarbeitung herangezogen werden.

Es wird empfohlen sich mindestens zwei Beispiele näher anzuschauen, um ein Verständnis für die Handhabung zu erlangen.

Tutorial

Das [Tutorial \[► 31\]](#) beschreibt, wie ein RFID Reader in Betrieb genommen wird. Dabei wird Schritt für Schritt von der Einbindung der TwinCAT Bibliothek bis hin zur Präsenzerkennung von RFID Transpondern vorgegangen.

Projekt Download: <https://infosys.beckhoff.com/content/1031/tcplclibrfid/Resources/11421957643.zip>

Beispiel 1

Dieses Beispiel kann für unterschiedliche RFID Reader genutzt werden (Balluff, Baltech, Deister, Leuze, Pepper+Fuchs).

Getestet ist das Beispiel mit den Modellen Balluff BIS M 401 und Leuze electronic RFM32.

Im Projekt wurde ein RFID Reader an eine einkanalige serielle EL6001 (an einem EK1100) angeschlossen.

Es lassen sich ebenso andere serielle Klemmen nutzen.

Das Projekt kann ebenso für zwei RFID Reader genutzt werden. Das Beispielprogramm ist bereits für zwei RFID Reader vorbereitet. Es muss lediglich die zweite Verlinkung im TwinCAT System Manager erfolgen.

[zum Beispiel \[► 70\]](#)

Beispiel 2

Dieses Beispiel ist mit einem Baltech RFID Reader, welcher in den Beckhoff Control-Panels sowie Panel-PCs optional verbaut ist, entwickelt worden. Das Gerät wird an einen seriellen Com Port oder einen USB Port angeschlossen.

Es kann genutzt werden, um das Gerät komfortabel in Betrieb zu nehmen und zu testen. Das Beispiel verfügt über eine einfache Visualisierung.

[zum Beispiel \[► 71\]](#)

Beispiel 3

Dieses Beispiel entspricht einer kleinen Applikation. Die Anwendung umfasst das Erkennen, Lesen und Schreiben eines Transponders in einem automatischen Ablauf.

Erstellt wurde das Beispiel mit einem Pepperl+Fuchs RFID Reader. Es kann sowohl das 2-kanalige als auch das 4-kanalige Modell genutzt werden.

[zum Beispiel \[► 72\]](#)

13.1 Beispiel 1

Dieses Beispiel kann für unterschiedliche RFID Reader genutzt werden (Balluff, Baltech, Deister, Leuze, Pepper+Fuchs).

Getestet ist das Beispiel mit den Modellen Balluff BIS M 401 und Leuze electronic RFM32.

Im Projekt wurde ein RFID Reader an eine einkanalige serielle EL6001 (an einem EK1100) angeschlossen.

Es lassen sich ebenso andere serielle Klemmen nutzen. Bei Verwendung von KL Klemmen muss der Aufruf des Serial Line Control im Programmcode angepasst werden. (siehe Kapitel [RFID Reader Anbindung \[► 14\]](#))

Das Projekt kann ebenso für zwei RFID Reader genutzt werden. Das Beispielprogramm ist bereits für zwei RFID Reader vorbereitet. Es muss lediglich die zweite Verlinkung im TwinCAT System Manager erfolgen.

Das Beispielprojekt beinhaltet den Aufruf des RFID Funktionsbausteines mit unterschiedlichen Befehlen. Die wichtigsten Befehle wurden in diesem Beispiel implementiert. Dazu gehört unter anderem das Lesen und Schreiben vom RFID Transponderspeicher.

Nach Programmstart muss über die lokale Enumeration eManufacturer der passende RFID Reader Hersteller ausgewählt werden.

Mittels der lokalen Enumeration eCommand kann der Befehlstyp ausgewählt werden.

Um den Aufruf zu starten muss die lokale Variable bExecute einmal auf TRUE gesetzt werden.

Daraufhin sind an den Ausgängen des RFID Funktionsbausteines die Ergebnisse der Abfrage angegeben.

Alternativ kann die Bedienung mit der im Beispiel integrierten Visualisierung geschehen:

fbRFID1

Manufacturer: eRFRM_Balluff

<-
chosen command: eRFC_ReadBlock
->

Execute to send the command

access data:

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

last response:

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

last detected tag:

tag serial number: E00780ACDDEB563D
tag type: eRFTT_TagItHfi



Je nach RFID Reader Modell müssen zuerst die Befehle GetReaderVersion und GetConfig (ggf. auch SetConfig) ausgeführt werden, um eine korrekte Kommunikation mit dem RFID Reader zu ermöglichen.

Für weitere Informationen zum Ablauf der RFID Reader Kommunikation wird auf das Kapitel [RFID Funktionsbaustein Handhabung \[▶ 39\]](#) verwiesen.

Projekt Download: <https://infosys.beckhoff.com/content/1031/tcplclibrfid/Resources/11421959051.zip>

13.2 Beispiel 2

Dieses Beispiel ist mit einem Baltech RFID Reader, welcher in den Beckhoff Control-Panels sowie Panel-PCs optional verbaut ist, entwickelt worden. Das Gerät wird an einen seriellen Com Port oder den USB-Port angeschlossen.

Es kann genutzt werden, um das Gerät komfortabel in Betrieb zu nehmen und zu testen.

Wird ein Baltech RFID Reader verwendet, der an eine serielle Beckhoff Klemme anstatt an den Com Port angeschlossen ist, muss die serielle Hintergrundkommunikation im SPS Code geändert werden und dies im TwinCAT System Manager neu konfiguriert werden. (siehe Kapitel [RFID Reader Anbindung \[▶ 14\]](#))

Das Beispielprojekt beinhaltet den Aufruf des RFID Funktionsbausteines mit unterschiedlichen Befehlen. Mit Hilfe der integrierten Visualisierung können die Befehle ausgeführt werden. Implementiert sind die Befehle GetReaderVersion, GetInventory, ReadBlock und WriteBlock. So kann auch ein RFID Transponder getestet werden und Daten in Form eines ASCII Strings auf diesen geschrieben werden sowie ebenso gelesen werden.

Rfid Demo Application for serial connected Rfid Reader in Beckhoff Panels

Inputs / Commands	Outputs					
Init Rfid Reader	Reader Info: <table border="1"> <tr> <td>Reader Type: eRFRT_BaltechIDE_LP</td> <td>1034</td> </tr> <tr> <td colspan="2">Reader's SW version : 1.08.01</td> </tr> </table>	Reader Type: eRFRT_BaltechIDE_LP	1034	Reader's SW version : 1.08.01		
Reader Type: eRFRT_BaltechIDE_LP	1034					
Reader's SW version : 1.08.01						
Detect Transponder	Transponder Info: <table border="1"> <tr> <td colspan="2">Transp.Type: eRFTT_LegicPrime</td> </tr> <tr> <td colspan="2">Transp.SerialNbr: 57F4E98B</td> </tr> </table>	Transp.Type: eRFTT_LegicPrime		Transp.SerialNbr: 57F4E98B		
Transp.Type: eRFTT_LegicPrime						
Transp.SerialNbr: 57F4E98B						
Select Data To Read: <table border="1"> <tr> <td>Byteldx: 10</td> <td>ByteCount: 13</td> </tr> </table>	Byteldx: 10	ByteCount: 13	Response Type: <table border="1"> <tr> <td>eResponse: eRFR_Data_ReadData</td> <td>Busy</td> </tr> </table>	eResponse: eRFR_Data_ReadData	Busy	
Byteldx: 10	ByteCount: 13					
eResponse: eRFR_Data_ReadData	Busy					
Read Data	Read Data: <table border="1"> <tr> <td>ReadData(hex): 48 61 6C 6C 6F 20 57 65 6C 74 21 00 00</td> </tr> <tr> <td>ReadData(ASCII): Hallo Welt!</td> </tr> </table>	ReadData(hex): 48 61 6C 6C 6F 20 57 65 6C 74 21 00 00	ReadData(ASCII): Hallo Welt!			
ReadData(hex): 48 61 6C 6C 6F 20 57 65 6C 74 21 00 00						
ReadData(ASCII): Hallo Welt!						
Select Data To Write: <table border="1"> <tr> <td>Byteldx: 10</td> </tr> <tr> <td>WriteData(ASCII): Hallo Welt!</td> </tr> </table>	Byteldx: 10	WriteData(ASCII): Hallo Welt!				
Byteldx: 10						
WriteData(ASCII): Hallo Welt!						
Write Data						
<table border="1"> <tr> <td>Byteldx: 10</td> <td>ByteCount: 12</td> </tr> </table>	Byteldx: 10	ByteCount: 12				
Byteldx: 10	ByteCount: 12					
Erase Data (Write 0x00 00 00 ...)	Error Detection: <table border="1"> <tr> <td>RfidError: FALSE</td> </tr> <tr> <td>ErrorID: eRFERR_NoError</td> <td>0</td> </tr> <tr> <td>ErrCodeRcv: eRFERRBaltech_NoError</td> <td>0</td> </tr> </table>	RfidError: FALSE	ErrorID: eRFERR_NoError	0	ErrCodeRcv: eRFERRBaltech_NoError	0
RfidError: FALSE						
ErrorID: eRFERR_NoError	0					
ErrCodeRcv: eRFERRBaltech_NoError	0					
Activate LogView Output	<table border="1"> <tr> <td>ComError: FALSE</td> </tr> <tr> <td>ComErrorID: COMERROR_NOERROR</td> </tr> <tr> <td>LastDetectedComErrorID: COMERROR_NOERROR</td> </tr> </table>	ComError: FALSE	ComErrorID: COMERROR_NOERROR	LastDetectedComErrorID: COMERROR_NOERROR		
ComError: FALSE						
ComErrorID: COMERROR_NOERROR						
LastDetectedComErrorID: COMERROR_NOERROR						

Der RFID Reader muss zuerst initialisiert werden, um eine korrekte Kommunikation mit dem RFID Reader zu ermöglichen. Der Button InitRfidReader führt dazu den Befehl GetReaderVersion aus.

Für weitere Informationen zum Ablauf der RFID Reader Kommunikation wird auf das Kapitel [RFID Funktionsbaustein Handhabung](#) [► 39] verwiesen.

Durch Aktivierung der LogView Ausgabe wird im TwinCAT System Manager LoggerView die komplette serielle Übertragung dargestellt.

Projekt Download: <https://infosys.beckhoff.com/content/1031/tcplclibrfid/Resources/11421960459.zip> or <https://infosys.beckhoff.com/content/1031/tcplclibrfid/Resources/11421961867.zip>

13.3 Beispiel 3

Dieses Beispiel entspricht einer kleinen Applikation. Die Anwendung umfasst das Erkennen, Lesen und Schreiben eines Transponders in einem automatischen Ablauf.

Erstellt wurde das Beispiel mit einem Pepperl+Fuchs RFID Reader. Es kann sowohl das 2-kanalige als auch

das 4-kanalige Modell genutzt werden.

Im Beispiel ist das Gerät direkt an den Com Port angeschlossen. Wird ein Pepperl+Fuchs RFID Reader verwendet, der an eine serielle Beckhoff Klemme anstatt an den Com Port angeschlossen ist, muss die serielle Hintergrundkommunikation im SPS Code geändert werden und dies im TwinCAT System Manager neu konfiguriert werden. (siehe Kapitel [RFID Reader Anbindung \[► 14\]](#))

Ablauf der implementierten Applikation:



Mit dem RFID Gerät sind zwei Schreib-/Leseköpfe verbunden. Beide erkennen vollautomatisch im Feld eintreffende Transponder. Nach Erkennung wird ein Speicherblock aus dem Datenspeicher des Transponders ausgelesen. Der sich darin befindliche 4-byte Wert wird vom ersten Lesekopf um eins addiert bzw. vom zweiten Lesekopf um eins subtrahiert. Der neue Wert wird sofort zurück auf den Transponder geschrieben. Dieser Ablauf des Erkennens, Lesens und Schreibens dauert in Summe ca. eine halbe Sekunde. Zwischen zwei solchen Vorgängen am selben Lesekopf müssen mindestens 3 Sekunden liegen, um eine ungewollte Mehrfachausführung zu vermeiden. (Dies liesse sich ebenso mittels Prüfung der Tag-Seriennummer lösen.)

Das Programm beinhaltet im Wesentlichen eine Zustandsmaschine mit 6 Zuständen:

- 0: Initialisierung - Ausführung von GetReaderVersion, GetConfig, etc.
- 1: Tag-Erkennung an Lesekopf 1- buffered GetInventory
- 2: Tag-Erkennung an Lesekopf 2- buffered GetInventory
- 3: Warten auf Tag-Erkennung
- 4: Aktion an Lesekopf 1 - ReadBlock und WriteBlock
- 5: Aktion an Lesekopf 2 - ReadBlock und WriteBlock



Der data carrier type (iUSEDCTYPE) sollte auf den jeweils verwendeten Transpondertypen angepasst werden.

Das Beispielprojekt beinhaltet den Aufruf des RFID Funktionsbausteines mit unterschiedlichen Befehlen. Für weitere Informationen zum Ablauf der RFID Reader Kommunikation wird auf das Kapitel [RFID Funktionsbaustein Handhabung \[► 39\]](#) verwiesen.

Projekt Download: <https://infosys.beckhoff.com/content/1031/tcplclibrfid/Resources/11421963275.zip>



Es muss die aktuelle Version der TwinCAT Bibliothek verwendet werden.

Mehr Informationen:
www.beckhoff.de/ts6600

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

