Programmierschnittstelle (SDK)
der TS-RW Geräte mit 125 und 134,2 kHz

TS_LF_SDK

Version 4.38

GiS
Gesellschaft für Informatik
und Steuerungstechnik mbH

Höllochstrasse 1
D-73252 Lenningen
Tel. +49 (0)7026 606 0
Fax +49 (0)7026 606 66
Email rfid@gis-net.de
Homepage http://www.gis-net.de/rfid

GiS
Gesellschaft für Informatik
und Steuerungstechnik mbH

Programmierschnittstelle (SDK) der TS-RW Geräte

Eigentumsvorbehalt:

Programmierschnittstelle (SDK) der TS-RW Geräte

Inhaltsverzeichnis

Programmierschnittstelle (SDK) der TS-RW Geräte

Programmierschnittstelle (SDK) der TS-RW Geräte

Änderungsstand:

Doku	Firmware	Datum	Kapitel	Name	Info
4.02		04.02.2011	4.0	Blank	Zuordnung der Transpondertypen HDX+ und NCS1015 war vertauscht.
4.03	1.08	14.04.2011	2	Blank	Standard Filterwerte korrigiert Befehlsreihenfolge umgestellt Untergliederung hinzugefügt Befehle für LCD hinzugefügt
4.04		07.06.2011		Blank	FSK Formate korrigiert
4.05	1.13	03.11.2011		Blank	Write FDX B zusätzliche Parameter möglich
4.06	1.16	24.01.2012		Blank	Transpondertyp HDX+ HiQ ergänzt
4.07	1.18	05.03.2012		Blank	Transpondertyp TMS37124 ergänzt
4.08	1.20	18.05.2012		Blank	Transpondertypen SIC7900 und SIC7888 ergänzt.
4.09	1.21	01.08.2012		Blank	Transpondertyp SIC7999 ergänzt
4.09	1.21	05.11.2012	4.0 4.2	Blank	Korrektur der Typnummer für SIC7999, Beschreibung der Typerkennung ergänzt.
4.12	1.25	08.11.2013	2.6 3.1.1.	Blank	Funktion WriteTagKey ergänzt. Transpondertypen erweitert
4.13	1.27	11.04.2014	4.3	Blank	Hitag 1, Hitag S und SIC7888 erweitert auf UID Request mit Antikollisionsalgorithmus sowie Quiet Funktion. Zugriff auf Hitag S im Crypto Mode
4.14	1.31	24.07.2014	2.5 3.1	Blank	Filtereinstellungen ergänzt Readermode Parameter erweitert
4.15		02.12.2014	2.2	Blank	Einstellung der Gateway Adresse
4.15		09.02.2015	6	Blank	Aktualisierung der Fehlerliste
4.16		22.05.2015	4	Blank	Zusätzliches Kommando für AutoTune
4.16		08.06.2015		Blank	verschiedene fehlende Punkte korrigiert
4.17	1.35	10.08.2015	2.5	Blank	AnswerMode 7 für FSK2a 2kBit
4.18		09.09.2015	4.6, 4.7	Blank	Lesen und Schreiben NC34Bit
4.18		19.10.2015	2.8	Blank	Beschreibung LCD Beleuchtung
4.20		01.03.2016	4.6, 4.7	Blank	Funktionen für AWID Datenformate hinzugefügt
4.22		14.07.2016	2.5	Blank	Funktionen für Gerätekonfiguration TS-RW82AC hinzugefügt
4.23	1.39	19.12.2016	1.1, 3.7	Blank	Erweiterungen für vereinfachte Verwendung im Readermodus
4.25		19.06.2017	3.7	Blank	Ergänzung bei Fehler im Readermodus
4.27		01.12.2017	2.3	Blank	Gerätetypen ergänzt, neue Funktionen eingefügt
4.29		10.08.2018		Blank	Interne Funktionalität in SDK aktualisiert
4.30		18.10.2018		Blank	Zusätzliche Funktionen ergänzt
4.31		01.04.2019		Blank	Zusätzliche Funktionen ergänzt
4.32		29.08.2019		Blank	Zusätzliche Funktionen TSLF_SetTimeout
4.33		06.12.2019		Blank	mehrere gleichzeitig geöffnete Geräte erhalten jetzt immer unterschiedliche Handles

Programmierschnittstelle (SDK) der TS-RW Geräte

Änderungsstand:

Doku	Firmware	Datum	Kapitel	Name	Info
4.34		08.01.2020		Blank	die Befehle TSLF_Beep, TSLF_SetLED, TSLF_RawWrite werden nun korrekt ausgeführt
4.35		08.06.2020	2.3.	Blank	Befehl TSLF_KeepAppActive in Doku ergänzt.
4.35		06.08.2020	4.6, 4.7	Blank	Befehle für IdentiCard Format ergänzt.
4.36		07.01.2021	4.5	Blank	Informationen zur Länge ergänzt.
4.37		02.11.2021		Blank	Zusätzlich Befehle für TI HDX+
4.38		08.04.2022	4.8	Blank	Transpondertyp EL9265 hinzugefügt mit zusätzlichen Kommandos. Unterstützung von TS-RW380

1. Allgemeines

Die Programmierschnittstelle für TS-R3x/R6x und TS-W3x/W6x Geräte, dient der Vereinfachung der Ansprache der GiS RFID Geräte in beliebigen Softwareanwendungen.

Die Ansprache der Geräte erfolgt im GiS Standard Protokoll G100 oder G300.
Die Erkennung des Geräteprotokolls erfolgt automatisch beim Verbinden mit dem Gerät.
Dies wird durch eine Versionsabfrage und Unterstützung der Kommandos je nach erkanntem Gerät durchgeführt.

Es wird eine Dynamic Link Library (DLL) zur Verfügung gestellt, die, die gesamte Funktionalität des Moduls abbildet. Diese DLL kann von verschiedenen Programmiersprachen aus verwendet werden.
Es werden zwei Varianten der DLL für 32 Bit oder 64 Bit Programme bereitgestellt.

Typ	Dateiname	Importbibliothek	Import Header für C / C++	Import für C#	Import für VB.net
32Bit	TS_LF_SDK.dll	TS_LF_SDK.lib	ts_lf_import.h	ts_lf_import.cs	ts_lf_import.vb
64 Bit	TS_LF_SDK64.dll	TS_LF_SDK64.lib	ts_lf_import.h	ts_lf_import64.cs	ts_lf_import64.vb

Die DLL unterstützt viele verschiedene Transpondertypen. Welcher Transpondertyp bearbeitet werden kann, hängt vom Gerätetyp ab und kann mit **TSLF_GetDeviceVersion()** ermittelt werden.

Programmierschnittstelle (SDK) der TS-RW Geräte

1.1. Betriebsarten

Je nach Gerät stehen verschiedene Betriebsarten zur Verfügung. Es wird hier zwischen den Betriebsarten "Readermodus" und "Programmermodus" unterschieden.
Geräte der "R" Serie kennen nur den Readermodus, Geräte der W Serie nur den Programmermodus während Geräte der RW Serie beide Modi unterstützen.

1.1.1. Readermodus

Als Readermodus wird der automatische Lesemodus bezeichnet. Hier arbeitet der Leser autark und sendet über die Schnittstelle die gelesenen Transponderdaten im Klartext. Welche Daten er in welcher Formatierung sendet, wird über die Applikation "TS-R3x ReaderSetup" oder über die Funktionen aus Kapitel 3 eingestellt. Dieser Modus wird auch oft verwendet, wenn das Gerät ohne Zuhilfenahme des SDK verwendet werden soll, also wenn es an einer COM Schnittstelle direkt mit einer für COM Schnittstellen ausgelegten Applikation oder als HID Gerät im Tastaturmodus verwendet wird.
Zur Verwendung mit dem SDK stehen spezielle Zugriffsfunktionen zur Verfügung, die nicht mit dem GiS Standard Protokoll arbeiten. (Kapitel 3.7)
Insbesondere ist bei Verwendung anderer Kommandos im Readermodus Vorsicht walten zu lassen, da der Leser ja wenn ein Transponder ins Feld kommt automatisch Daten absetzt, die mit den Antworten auf andere Kommandos kollidieren können.
Aus diesem Grund wurden für den Zugriff auf den Summer sowie die LEDs zusätzliche Direktkommandos ohne Rückmeldung geschaffen. (Kapitel 3.7)

1.1.2. Programmermodus

Im Programmermodus wird der Transponderzugriff über die Kommandos ausgelöst.
Hier werden die Kommandos aus Kapitel 4 und 5 sowie die allgemeinen Kommandos aus Kapitel 2 verwendet.
In diesem Modus können beliebige Blöcke der verschiedenen Transpondertypen gelesen und geschrieben werden.

Programmierschnittstelle (SDK) der TS-RW Geräte

1.2. Fehlerbearbeitung

Alle Befehle liefern einen Rückgabewert –1, wenn ein Fehler aufgetreten ist. Die Fehlernummer kann mit **TSLF_GetLastError()** abgefragt werden.
Die Fehlerliste befindet sich im Anhang dieses Dokuments.

1.3. Definitionen

Folgende Datentypen werden bei der Übergabe zu den Funktionen verwendet:

int	32 Bit Integer
BYTE	8 Bit unsigned integer
BYTE *	Zeiger auf ein Feld mit 8 Bit unsigned integer Werten
char *	Zeiger auf ein Feld mit 8 Bit signed integer Werten, (ANSI String gemäß C Definition) meist 0-terminiert

Die Datenübergabe in Feldern erfolgt grundsätzlich LSB zuerst.

2. Allgemeine Befehle

int TSLF_LibVersion()
Rückgabewert: -1 Fehler, >0 DLL Versionsnummer z.B. 100 entspricht Version 1.00

Liefert die Version der DLL.
Diese Funktion benötigt kein TSLF_Open.

int TSLF_IsVirtualComInstalled()
Rückgabewert: 1 Virtual Com Treiber für TS-RW38 ist installiert
0 Virtual Com Treiber für TS-RW38 ist nicht installiert.

int TSLF_IsDeviceDriverMissing()
Rückgabewert: 0: Keine GiS Geräte mit fehlenden Treibern gefunden
1: Treiber für angeschlossenes RW34 USB Gerät fehlt
2: Treiber für angeschlossenes RW38 VCOM Gerät fehlt
3: Treiber für angeschlossene RW34 USB und RW38 VCOM Geräte fehlen

2.1. Ermitteln der angeschlossenen Geräte

int TSLF_GetCOMDeviceNames (char* NamenListe, int BufferSize**)**

NamenListe	Die Namens Liste besteht aus einer Folge von Null-terminierten Strings. Das Listen Ende besteht aus einem Leerstring.
BufferSize	BufferSize gibt an wie viel Bytes für die Namenliste bereit gestellt wird.
Rückgabewert:	< 0 Fehler, > 0 Länge der NamenListe in Bytes.

Mit der Funktion **TSLF_GetCOMDeviceNames** kann man sich die Namen aller verfügbaren COM Schnittstellen besorgen. Jeder Name besteht aus einem NULL terminierten String.
Beispiel: „COM1“. Es können maximal 255 serielle Schnittstellen sein. Die COM Schnittstellen können entweder als reale oder als Virtuelle (über USB realisierte) Schnittstellen vorliegen
Bei Virtuellen Ports wird die Bezeichnung des Virtuellen Ports mit dargestellt.

Beispiele:

"\\.\COM1"	Echte COM Schnittstelle 1
"\\.\COM4 [GiS Virtual COM]"	Virtueller COM Port 4 bereitgestellt durch den "GiS Virtual COM" Treiber
"\\.\COM14 [GiS/FTDI Virtual COM]"	Virtueller COM Port 14 bereitgestellt durch den "GiS/FTDI Virtual COM" Treiber

Die hier ermittelten Schnittstellenbezeichnungen können so direkt an TSLF_Open übergeben werden.

Programmierschnittstelle (SDK) der TS-RW Geräte

int TSLF_GetUSBDeviceNames(char* NamenListe, int BufferSize**)**

NamenListe	Die Namens Liste besteht aus einer Folge von Null-terminierten Strings. Das Listen Ende besteht aus einem Leerstring.
BufferSize	BufferSize gibt an wie viel Bytes für die Namenliste bereitgestellt wird.
Rückgabewert:	< 0 Fehler, > 0 Länge der NamenListe in Bytes.

Mit der Funktion **GetUSBDeviceNames** kann man sich die USB-Namen (Seriennummern) der USB-Geräte von GiS besorgen, die von diesem SDK direkt unterstützt werden. Jeder Name besteht aus einem NULL terminierten String mit mindestens 8 ASCII Zeichen.
Beispiel: „11360001“ oder „1460-0001 HID“.
Aufgrund der Zusatzangabe „HID“ kann erkannt werden, ob es sich um ein HID (HumanInterfaceDevice) Gerät (USB Tastatursimulation) handelt.

int TSLF_GetUSBDeviceNamesEx(char* NamenListe, int BufferSize**)**

NamenListe	Die Namens Liste besteht aus einer Folge von Null-terminierten Strings. Das Listen Ende besteht aus einem Leerstring.
BufferSize	BufferSize gibt an wie viel Bytes für die Namenliste bereitgestellt wird.
Rückgabewert:	< 0 Fehler, > 0 Länge der NamenListe in Bytes.

Mit der Funktion **GetUSBDeviceNamesEx** kann man sich die USB-Namen (Seriennummern) aller USB-Geräte von GiS besorgen. Also auch Geräte, die von diesem SDK nicht direkt unterstützt werden (z.B.: 13.56 MHz Lesegeräte). Jeder Name besteht aus einem NULL terminierten String mit mindestens 8 ASCII Zeichen.
Beispiel: „11360001“ oder „1460-0001 HID“.
Aufgrund der Zusatzangabe „HID“ kann erkannt werden, ob es sich um ein HID (HumanDeviceInterface) Gerät (USB Tastatursimulation) handelt.

Programmierschnittstelle (SDK) der TS-RW Geräte

2.2. Funktionen für Ethernet Geräte

int TSLF_GetLanDeviceNames(char* NamenListe, int nBufferSize**)**

NamenListe	Die Namens Liste besteht aus einer Folge von Null-terminierten Strings. Das Listen Ende besteht aus einem Leerstring.
BufferSize	BufferSize gibt an wie viel Bytes für die Namenliste bereitgestellt wird.
Rückgabewert:	< 0 Fehler, > 0 Länge der NamenListe in Bytes.

Mit der Funktion **GetLanDeviceNames** kann man sich die LAN-Namen
aller LAN-Geräte besorgen. Es werden nur die TS-R3x und TS-W3x Geräte von GiS
berücksichtigt. Jeder Name besteht aus einem NULL terminierten String.
Es muss genügend Speicher für alle Geräte im Netz bereitgestellt werden.
Beispiele:

192.168.0.100-10001
[TS-LAN01]

Die Gerätenamen können entweder aus der IP-Adresse des Gerätes gefolgt von der Portnummer oder bei DHCP Geräten aus dem DHCP Namen bestehen. Der DHCP Name ist in eckigen Klammern [ ] eingefasst. Hiermit werden nur Geräte erfasst, die mit den aktuellen Netzwerkeinstellungen erreichbar sind.

int TSLF_GetLanDeviceNamesEx(char* NamenListe, int nBufferSize**)**

NamenListe	Die Namens Liste besteht aus einer Folge von Null-terminierten Strings. Das Listen Ende besteht aus einem Leerstring.
BufferSize	BufferSize gibt an wie viel Bytes für die Namenliste bereitgestellt wird.
Rückgabewert:	< 0 Fehler, > 0 Länge der NamenListe in Bytes.

Mit der Funktion **GetAllLanDeviceNames** kann man sich die LAN-Namen
aller LAN-Geräte besorgen. Jeder Name besteht aus einem NULL terminierten String.
Es muss genügend Speicher für alle Geräte im Netz bereitgestellt werden.
Beispiele:

192.168.0.100-10001
[TS-LAN01]169.194.245.01-10001

Die Gerätenamen können entweder aus der IP-Adresse des Gerätes gefolgt von der Portnummer oder bei DHCP Geräten aus dem DHCP Namen gefolgt von IP-Adresse und Portnummer bestehen. Der DHCP Name ist in eckigen Klammern [ ] eingefasst.
Mit dieser Funktion werden alle Geräte erfasst, auch wenn sie nicht mit den aktuellen Netzwerkeinstellungen erreichbar sind.

Programmierschnittstelle (SDK) der TS-RW Geräte

int TSLF_ChangeLanIPAddress(LPCSTR OldAddress,
LPCSTR NewAddress,
LPCSTR IPMask**)**

OldAddress	Bisherige IP-Adresse
NewAddress	Neue IP-Adresse
IPMask	Neue IP-Maske

int TSLF_ChangeLanIPAddressEx(LPCSTR OldAddress,
LPCSTR NewAddress,
LPCSTR IPMask
LPCSTR Gateway**)**

OldAddress	Bisherige IP-Adresse
NewAddress	Neue IP-Adresse
IPMask	Neue IP-Maske
Gateway	Neue Gateway Adresse

Mit diesen Funktionen kann die IP-Adresse und die Portnummer eines Gerätes geändert werden.
Das Gerät muss hierzu mit den aktuellen Netzwerkeinstellungen erreichbar sein.
Die Übergabeparameter sind jeweils 0 terminierte ASCII Strings.
Die IP-Adressen bestehen entweder aus der IP-Adresse gefolgt von der Portnummer oder aus dem DHCP Namen eingeschlossen in [ ].
Beispiele:

192.168.0.100-10001
[TS-LAN01]

In der IP Maske werden signifikanten Bits als Bitmaske angegeben.
Beispiel:

255.255.255.0

Bei Gateway Adresse wird die IP Adresse des Gateways angegeben. Soll kein Gateway eingetragen werden, so wird hier 0.0.0.0 verwendet. Bei Gateway können keine DHCP Namen angegeben werden.

Mit Änderung der IP Adresse wird das Gerät zurückgesetzt. Es kann anschließend bis zu 25 Sekunden dauern, bis das Gerät wieder im Netz erreichbar ist.

Programmierschnittstelle (SDK) der TS-RW Geräte

int TSLF_GetLanIPAddressEx(LPCSTR Name,
LPSTR Address,
LPSTR IPMask
LPSTR Gateway**)**

Name	Name des Gerätes
Address	IP-Adresse
IPMask	IP-Maske
Gateway	Gateway Adresse

Mit dieser Funktion wird die IP-Adresse und die Portnummer eines Gerätes gelesen.
Das Gerät muss hierzu mit den aktuellen Netzwerkeinstellungen erreichbar sein.
Die Übergabeparameter sind jeweils 0 terminierte ASCII Strings.
In Address wird entweder der DHCP Name oder die IP Adresse geliefert.
Es muss in den Übergabestrings jeweils genügend Platz zum Speichern der Adresse vorhanden sein. (mindestens 30 Byte)

int TSLF_IsLanDeviceAvailable(LPCTSTR Address**)**

Adress	IP-Adresse

Rückgabewert: < 0 Fehler,
0 Gerät nicht vorhanden
> 0 Gerät ist vorhanden

Die IP-Adresse besteht entweder aus der IP-Adresse gefolgt von der Portnummer oder aus dem DHCP Namen eingeschlossen in [ ].
Beispiele:
192.168.0.100-10001
[TS-LAN01]

Mit dieser Funktion kann geprüft werden, ob das angegebene Gerät im Netz erreichbar ist.

Programmierschnittstelle (SDK) der TS-RW Geräte

2.3. Kommunikation mit einem Gerät aufbauen

Achtung: **Die Funktion TSLF_Open() muss grundsätzlich vor allen anderen Befehlen aufgerufen werden, da der zurückgegebene PortHandle für alle Schreib- und Lesefunktionen benötigt wird. Konnte die Schnittstelle nicht geöffnet werden, wird eine negative Fehlernummer entsprechend der Fehlerliste zurückgegeben.**

int TSLF_Open(LPCTSTR strInterfaceName, int Baudrate, int ParityMode, int Timeout**)**
Öffnen der Kommunikationsschnittstelle für alle RS2332, USB und LAN Geräte.

strInterfaceName	Name der Schnittstelle. z.B. COM1 oder Seriennummer des USB Geräts. Bei **USB** Geräten muss der Gerätename eingetragen werden. Bei **USB HID** Geräten muss der Gerätename gefolgt von "HID" eingetragen werden. Die verfügbaren USB-Namen, kann man mit der Funktion **TSLF_GetAllUSBDeviceNames** ermitteln. z.B. „10610011“ oder "1317-0001 HID" Bei **LAN** Geräten muss die Adresse eingetragen werden. Die verfügbaren LAN Geräte kann man mit der Funktion **TSLF_GetAllLanDeviceNames** ermitteln.
Baudrate	Gewünschte Übertragungsrate. (2400, 4800, 9600, 19200, 38400, 57600, 115200 und 230400 zulässig). Achtung: Nicht alle Geräte unterstützen alle Baudraten. Bitte sehen Sie in der Gerätespezifikation nach den verfügbaren Übertragungsraten. **RS232 Leser sind standardmäßig auf 19200 eingestellt.** **USB und LAN Leser ignorieren die Baudrate (intern fest auf 19200).**
ParityMode	0: 8 Datenbits, 1 Stoppbit, keine Parität 1: 8 Datenbits, 1 Stoppbit, gerade Parität 2: 8 Datenbits, 1 Stoppbit, ungerade Parität 3: 8 Datenbits, 2 Stoppbit, keine Parität
Timeout	Zeit nach der die Kommunikation abgebrochen wird. (in Millisekunden)
Rückgabewert:	>0: **PortHandle** -1: Ungültige Schnittstelle -5: Timeout ungültig -15: Gerät antwortet nicht, evtl. kein Gerät angeschlossen

Wahlweise kann beim Schnittstellenname noch die Geräteadresse mit angegeben werden. Also z.B.: COM1:4 öffnet an COM 1 das Gerät mit Adresse 4. Dies ist notwendig, wenn die Geräteadresse nicht 1 ist, da bei TSLF_Open der Verbindungsaufbau mit dem Gerät gleich durchgeführt wird und die Verbindung nur akzeptiert wird wenn das Gerät antwortet.
Die Funktion erkennt automatisch das zugrunde gelegte LowLevel Protokoll des Gerätes (normalerweise GiS LowLevel Protokoll TS-RW3x, bei MultiX Geräten MultiX Modbus Protokoll)

Programmierschnittstelle (SDK) der TS-RW Geräte

Achtung: **Die Funktion TSLF_Close() muss grundsätzlich am Ende einer Applikation aufgerufen werden, da diese die geöffnete Schnittstelle wieder schließt und die Ressourcen wieder frei gibt.**

int TSLF_Close(int PortHandle**)**
PortHandle Zugriffsnummer
Rückgabewert: -1 Fehler, 0 OK

Schließen der Kommunikationsschnittstelle zum Gerät. Anschließend ist der PortHandle ungültig.

int TSLF_KeepAppActive (int PortHandle, int bActive**)**
PortHandle Zugriffsnummer
bActive 0: Applikation wird inaktiv während eine SDK Funktion ausgeführt wird
1: Applikation bleibt aktiv während eine SDK Funktion ausgeführt wird

Hiermit kann das Verhalten der aufrufenden Applikation gesteuert werden. Dies ist insbesondere hilfreich wenn Funktionen in sehr kurzen Intervallen aufgerufen werden. Nach TSLF_Open steht die Funktion auf bActive = 0. Damit ist während des Aufrufs einer Funktion die Applikation blockiert. Nach Aufruf von KeepAppActive mit bActive = 1 bleibt die Applikation auch innerhalb eines Funktionsaufrufes aktiv. Allerdings ist dann darauf zu achten, dass innerhalb einer Funktion keine weiteren Funktionen für dieses PortHandle aufgerufen werden können.

int **TSLF_IsUSB(**int PortHandle)
PortHandle Logische Gerätenummer von OpenPort
Rückgabewert: -1 Fehler, 0 = **KEIN** USB Port, 1 = USB Port

int TSLF_SetReaderAdresse(int PortHandle, int Adresse**)**
PortHandle Zugriffsnummer
Adresse Adresse des Lesemoduls. Defaultwert nach öffnen des Ports ist 1 oder die bei OpenPort angegebene Adresse. Die Adresse wird nur bei RS485 Verbindungen verwendet, wenn mehr als ein Gerät angeschlossen ist.

Zugriff auf mehrere Geräte an einer Schnittstelle über die Adressbelegung. Bei Geräten mit mehreren Antennen wird diese Funktion verwendet um zwischen den einzelnen Antennen umzuschalten.

Programmierschnittstelle (SDK) der TS-RW Geräte

int **TSLF_IsReader(**int PortHandle)
PortHandle Logische Gerätenummer von OpenPort
Rückgabewert: -1 Fehler
0 = Reader Modus wird **nicht** unterstützt
1 = Reader Modus wird unterstützt
Diese Funktion meldet nur, ob der Reader Modus möglich ist, nicht jedoch ob der Reader Modus eingeschaltet ist.

int **TSLF_IsProgrammer(**int PortHandle)
PortHandle Logische Gerätenummer von OpenPort
Rückgabewert: -1 Fehler
0 = Programmer Modus wird **nicht** unterstützt
1 = Programmer Modus wird unterstützt
Diese Funktion meldet nur, ob der Programmer Modus möglich ist, und die entsprechenden Befehle unterstütz werden, jedoch nicht ob der Programmer Modus eingeschaltet ist.

int **TSLF_IsPlusProgrammer(**int PortHandle)
PortHandle Logische Gerätenummer von OpenPort
Rückgabewert: -1 Fehler
0 = Erweiterter Programmer Modus wird **nicht** unterstützt
1 = Erweiterter Programmer Modus wird unterstützt
Meldet, ob erweiterte Funktionen verfügbar sind, die nur bei Plus oder AC Geräten zur Verfügung stehen.

int **TSLF_IsAnimalProgrammer(**int PortHandle)
PortHandle Logische Gerätenummer von OpenPort
Rückgabewert: -1 Fehler
0 = Animal Programmer Modus wird **nicht** unterstützt
1 = Animal Programmer Modus wird unterstützt
Meldet ob das Gerät ein 134,2 kHz Gerät mit Funktionen für Animal Code ist.

int **TSLF_IsHDXProgrammer(**int PortHandle)
PortHandle Logische Gerätenummer von OpenPort
Rückgabewert: -1 Fehler
0 = HDX Programmer Modus wird **nicht** unterstützt
1 = HDX Programmer Modus wird unterstützt
Meldet, ob das Gerät ein 134,2 kHz Gerät mit HDX Unterstützung ist.

Programmierschnittstelle (SDK) der TS-RW Geräte

int TSLF_SetBaudrate(int PortHandle, int Baudrate, int ParityMode**)**

PortHandle	Zugriffsnummer
Baudrate	2400, 4800, 9600,19200 und 38400 (Default = 19200).
ParityMode	0: 8 Datenbits, 1 Stopbit, keine Parität
	1: 8 Datenbits, 1 Stopbit, gerade Parität
	2: 8 Datenbits, 1 Stopbit, ungerade Parität
	3: 8 Datenbits, 2 Stopbit, keine Parität
Rückgabewert:	-1 Fehler, 0 OK

Baudrate setzen (nur für RS232 Geräte; USB oder LAN Geräte liefern einen Fehler)
Die Baudrate 38400 wird nur von TS-RW38 Geräten unterstützt.

int TSLF_GetLastError(int PortHandle**)**

PortHandle	Zugriffsnummer
Rückgabewert:	Fehlernummer wie im Anhang beschrieben

Letzten aufgetretenen Fehler abrufen. Alle Funktionen zum Zugriff auf ein geöffnetes Geräte liefern den allgemeinen Rückgabewert -1 für Fehler. Mit der Funktion TSLF_GetLastError kann die genauere Fehlerursache abgefragt werden.

int TSLF_GetDeviceVersion(int PortHandle, char * pDevVer, int VerLen,
char * pDevName, int NameLen**)**

PortHandle	Zugriffsnummer
pDevVer	Zeiger auf die Versionsdaten
VerLen	Länge der Versionsdaten (4 Byte)
pDevName	Zeiger auf Gerätename
NameLen	Länge des Gerätenamens
Rückgabewert:	-1 Fehler
	>0 Länge der Versionsdaten

Gerätenummer und Gerätefirmware im ASCII Format.
Byte 1 – 3 Gerätenummer
Byte 4 Geräte Firmware (0 – 9,A – Z)

Gerätename:
Der Gerätename wird als 0-terminierter String zurückgegeben.
Wird als pDevName ein Nullzeiger übergeben, oder ist in NameLen eine zu kurze Länge angegeben, so wird der Gerätename nicht eingetragen.
Der Gerätename lautete wie in der folgenden Tabelle eingetragen.
Die maximale Länge eines Gerätenamens beträgt 32 Byte.

Programmierschnittstelle (SDK) der TS-RW Geräte

Folgende Geräte werden von diesem SDK unterstützt:

Geräte-nummer	Bezeichnung	Lesermodus	Programmer	Animal Code	Unterstützte Transpondertypen (ab Firmwareversion)											
					HDX *	Hitag 1	Hitag 2	Hitag S	Q5	Titan	EM4569	EM4305	ATA5577	Hitag Y	ATA5575	SIC7888	EL9265
001	TS-W30		X						X								
002	TS-W30		X					X									
003	TS-W30		X			X											
010	TS-W32		X			X	X	X	X	X							
011	TS-W32		X			X	X	X	X	X							
021	TS-W3x		X			X	X	X	X	X	11		19				
024	TS-R3x	X				X	X	X	X	X	11		19				
025	TS-RW3x	X	X			X	X	X	X	X	11		19				
029	TS-W3x FDX		X	X		X	X	X	X	X	11		19				
031	TS-W3x BEE 270kHz		X	X		X	X	X	X	X	X	4	5	7			
039	TS-W3x AC 134kHz		X	X		X	X	X	X	X	X	4	5	7			
041	TS-W36		X			X	X	X	X	X	11		19				
044	TS-R36	X				X	X	X	X	X	11		19				
045	TS-RW36	X	X			X	X	X	X	X	11		19				
049	TS-W36 FDX		X	X		X	X	X	X	X	11		19				
051	TS-R320	X				X	X	X	X	X	X	X	X	X	X	X	
052	TS-RW320	X	X			X	X	X	X	X	X	X	X	X	X	X	
053	TS-RW320+	X	X	X		X	X	X	X	X	X	X	X	X	X	X	
054	TS-RW320 AC 134kHz	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	
061	TS-W6x		X			X	X	X	X	X	11		19				
064	TS-R6x	X				X	X	X	X	X	11		19				
065	TS-RW6x	X	X			X	X	X	X	X	11		19				
069	TS-W6x AC		X	X		X	X	X	X	X	11		19				
071	TS-W64 II 1170		X					X									
081	TS-MultiX																
109	TS-W80 AC 8x		X	X		X	X	X	X	X	X	4	5	7			
211	TS-W21		X			X	X	X	X	X	11		19				
214	TS-R21	X				X	X	X	X	X	11		19				
215	TS-RW21	X	X			X	X	X	X	X	11		19				
219	TS-W21AC		X	X		X	X	X	X	X	X	4	5	7			
241	TS-R38	X				X	X	X	X	X	X	X	X	X	X	20	
242	TS-RW38	X	X			X	X	X	X	X	X	X	X	X	X	20	
243	TS-RW38+	X	X	X		X	X	X	X	X	X	X	X	X	X	20	50
244	TS-RW38 AC 134kHz	X	X	X	X	X	X	X	X	X	X	X	X	X	X	20	50
251	TS-R68	X				X	X	X	X	X	X	X	X	X	X	20	
252	TS-RW68	X	X			X	X	X	X	X	X	X	X	X	X	20	
253	TS-RW68+	X	X	X		X	X	X	X	X	X	X	X	X	X	20	50
254	TS-RW68 AC 134kHz	X	X	X	X	X	X	X	X	X	X	X	X	X	X	20	50
294	TS-RW82AC		X	X	X	X	X	X	X	X	X	X	X	X	X	X	
551	TS-R380	X				X	X	X	X	X	X	X	X	X	X	X	
552	TS-RW380	X	X			X	X	X	X	X	X	X	X	X	X	X	
553	TS-RW380+	X	X	X		X	X	X	X	X	X	X	X	X	X	X	X
554	TS-RW380AC	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

* Folgende HDX Transpondertypen werden unterstützt:
TI TMS37190, NCD1015
TI TMS37124, SIC7900, SIC 7900X, SIC7999 (Bei RW38/68 ab Firmware Version 20)

2.4. Betriebsart festlegen

int TSLF_SetReaderMode(int PortHandle, int Mode**)**

PortHandle	Zugriffsnummer
Mode	0 setzt das Gerät in dem Programmermodus.
	1 setzt das Gerät in den Readermodus.
	2 setzt das Gerät in den Einschaltmodus
	80H legt als Einschaltmodus den Programmermodus fest
	81H legt als Einschaltmodus den Readermodus fest.
Rückgabewert:	-1 Fehler, 0 OK

Bei Geräten die sowohl den Readermodus als auch den Programmermodus unterstützen kann mit diesem Befehl der Reader- bzw. Programmermodus aktiviert werden.
Im Readermodus sendet das Gerät automatisch die Daten des Transponders ohne Protokollrahmen wie in der Parametereinstellung für den Reader Mode eingestellt. Dies ist im Programmermodus äußerst störend, weshalb hier der Readermodus deaktiviert werden muss.
Außerdem kann es besonders bei HID (Tastatursimulation) Geräten störend sein, wenn immer die Tastatursimulation aktiviert wird sobald ein Transponder aufgelegt wird.

int TSLF_SetPowerUpMode(int PortHandle, int Mode**)**

PortHandle	Zugriffsnummer
Mode	0H legt als Einschaltmodus den Programmermodus fest
	1H legt als Einschaltmodus den Readermodus fest.
Rückgabewert:	-1 Fehler, 0 OK

Entspricht der TSLF_SetReaderMode Funktion mit Mode 80H bzw. 81H.

int TSLF_GetPowerUpMode(int PortHandle**)**

PortHandle	Zugriffsnummer
Rückgabewert:	-1 Fehler,
	0 Einschaltmodus ist Programmermodus
	1 Einschaltmodus ist Readermodus

List den PowerUp Modus aus dem Gerät aus.

int TSLF_SetRF(int PortHandle,int OnOff**)**

PortHandle	Zugriffsnummer
OnOff	Kommando, 0 = Antennenfeld ausschalten, 1 = Antennenfeld einschalten
Rückgabewert:	-1 Fehler, 0 OK

Ein und Ausschalten des Antennenfeldes.

Programmierschnittstelle (SDK) der TS-RW Geräte

2.5. Geräteparameter setzen

int TSLF_GetFilter(int PortHandle,int TTyp, BYTE * pParam, int nParamLen**)**

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
pParam	Zeiger auf die Parameterdaten (ein Filterparametersatz)
nParamLen	Länge der Parameterdaten
Rückgabewert:	-1 Fehler
	0 Gerät unterstützt Lesen der Filterparameter nicht.
	>0 Länge der Filterparameter

Achtung: Nur TS-W38 Geräte erlauben das Lesen der Filterparametersätze

int TSLF_SetFilter(int PortHandle,int TTyp, BYTE * pParam, int nParamLen**)**

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
pParam	Zeiger auf die Parameterdaten (ein Filterparametersatz)
nParamLen	Länge der Parameterdaten
Rückgabewert:	-1 Fehler, 0 OK

Die Filterparametersätze werden bei TS-W38 Geräten für jeden Transpondertyp getrennt im Gerät dauerhaft gespeichert. Bei Zugriff auf einen Transpondertyp wird automatisch der entsprechende Filterparametersatz geladen.
Bei allen älteren Geräten kann nur ein Filterwert dauerhaft gespeichert werden. Dieser ist nicht Transponderspezifisch. Es ist bei diesen Geräten also unerheblich welcher Transpondertyp angegeben wird. Es wird dort auch nur „Para1 Filter“ übertragen, die erweiterten Parameter sind nur bei TS-W38 Geräten wirksam.
Bei älteren Geräten muss beim Wechsel des Transpondertyps der passende Filter erneut gesetzt werden.

Programmierschnittstelle (SDK) der TS-RW Geräte

Definition der Filterparametersätze:

Transpondertyp	Para1 Filter	Para 2 StartGap	Para 3 GapTime	Para 4 0-Length	Para 5 1-Length	Para 6 Answer Read Manchester 2kBit	Para 7 Answer Write Manchester 2kBit	Para 8 Answer Read Biphase 4kBit	Para 9 Answer Write Biphase 4kBit	Para 10 Answer Read FSK2a 2.5kBit	Para 11 Answer Write FSK2a 2.5kBit
1 (TIRIS)	-	-	-	-	-	-	-	-	-	-	-
2 (Hitag 2)	02	-	6	14	22	0*	0*	-	-	-	-
3 (Q5)	03	30	14	24	56	0*	0*	0*	0*	0*	0*
4 (Hitag S)	02	-	6	14	22	0*	0*	-	-	-	-
5 (Hitag 1)	02	-	6	14	22	0*	0*	-	-	-	-
6 (Titan)	02	-	32	32	64	0*	0*	-	-	-	-
7 (EM4569)	00	34	22	18	32	0*	0*	0*	0*	-	-
8 (EM4305)	00	34	22	18	32	0*	0*	0*	0*	0*	0*
9 (ATA5577)	03	30	14	24	56	0*	0*	0*	0*	0*	0*
10 (Hitag µ)	02	28	8	12	20	0*	0*	-	-	-	-
11 (ATA5575)	03	30	14	24	56	0*	0*	0*	0*	0*	0*
12 (SIC7888)	02	-	6	14	22	0*	0*	-	-	-	-
13 (ATA5551)	03	30	18	20	52	0*	0*	0*	0*	0*	0*
14 (EL9265)	02	-	6	14	22	0*	0*	-	-	-	-

Die Zeiten sind immer in 1/RF angegeben,
also bei RF = 125kHz in 8µSek Einheiten, bei RF = 134,2kHz in 7,45µSek Einheiten.
- Der Wert ist für diesen Transpondertyp nicht relevant und wird als 0 übergeben.
* Die Antwortzeiten sind relativ zur vorgegebenen Antwortzeit in ± 127 Einheiten einstellbar.

Der Parameter 1 Filter gibt die Einstellung der internen Filter für die Signalaufbereitung wieder. Hierbei sind die Bits folgendermaßen belegt:

Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Fast Mode	RFU	RFU	LowPass Filter aktiv	Verstärkung		HighPass Filter	LowPass Filter
0 = Normal Mode 1 = Fast Mode	0	0	0 = aktiv 1 = inaktiv	0 = 100 2 = 500	1 = 200 3 = 1000	0 = 40 Hz 1 = 160 Hz	0 = 3kHz 1 = 6kHz

Fast Mode gibt es nur bei Atmel Transpondern, dort werden die Lese und Schreibzugriffe dann ohne Mehrfachauswertung durchgeführt.

Programmierschnittstelle (SDK) der TS-RW Geräte

int TSLF_SetIO(int PortHandle,int Maske, int Daten**)**
PortHandle Zugriffsnummer
Maske Maskenwert für die zu setzenden Ausgänge
Daten Werte der zu setzenden Ausgänge
Es werden die Bits aus Daten übernommen, die in Maske gesetzt sind.
Rückgabewert: -1 Fehler, 0 OK

Hiermit werden die Ausgänge des Gerätes geschaltet. Je nach Geräteausführung können die Ausgänge unterschiedlich vorhanden und belegt sein. Nach Einschalten des Gerätes werden die LED's automatisch gesetzt. Nach Verwendung dieses Kommandos werden nur noch die Ausgänge automatisch gesetzt, die nicht in der Maske enthalten sind.
Bedeutung der Bits:

Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
LED gelb	LED grün	LED rot	Summer	Out 3	Out 2	Out 1	Out 0

Ist das Gerät mit einem Relaisausgang ausgestattet, so wird dieser über Out 0 angesprochen.

Beispiel: Maske: C0H Daten: 40H setzt die grüne LED und löscht die gelbe LED, die rote LED und alle anderen Ausgänge bleiben unverändert und können weiterhin automatisch durch den Leser je nach Betriebszustand gesetzt werden.

int TSLF_ReadIO(int PortHandle, int * Daten**)**
PortHandle Zugriffsnummer
Daten Werte der gelesenen Eingänge
Rückgabewert: -1 Fehler, 0 OK

Je nach Geräteversion können die Eingänge unterschiedlich vorhanden und belegt sein.
Bedeutung der Bits:

Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
O	O	O	O	In 3	In 2	In 1	IN 0

Ist das Gerät mit einem Sensoreingang ausgestattet, so wird der In 0 als Sensoreingang verwendet.

int TSLF_SetDefault(int PortHandle**)**
PortHandle Zugriffsnummer
Rückgabewert: -1 Fehler, 0 OK

Standardeinstellung aktivieren.
Dieser Befehl wird nur von Geräten mit Gerätenummer 081 unterstützt.

Programmierschnittstelle (SDK) der TS-RW Geräte

int TSLF_SetConfig (int PortHandle, BYTE * pConfig, int ConfigLen**)**

PortHandle	Zugriffsnummer
pConfig	Zeiger auf Konfiguration
ConfigLen	Länge des Puffers
Rückgabewert:	-1 Fehler, 0 OK

Schreiben der Konfiguration in das Gerät. Mögliche Konfigurationswerte sind je nach Gerät unterschiedlich. Diese Funktion wird nicht von allen Geräten unterstützt.

int TSLF_GetConfig (int PortHandle, BYTE * pConfig, int ConfigLen**)**

PortHandle	Zugriffsnummer
pConfig	Zeiger auf den, vom Benutzer zur Verfügung gestellten, Lesepuffer
ConfigLen	Länge des Puffers
Rückgabewert:	-1 Fehler, Länge der tatsächlich gelesenen Daten.

Lesen der Konfiguration aus dem Gerät. Mögliche Konfigurationswerte sind je nach Gerät unterschiedlich. Diese Funktion wird nicht von allen Geräten unterstützt.

int TSLF_ReadSerialNumber(int PortHandle, BYTE * pSerial, int Buflen**)**

PortHandle	Zugriffsnummer
pSerial	Zeiger auf den, vom Benutzer zur Verfügung gestellten, Lesepuffer
Buflen	Länge des Puffers
Rückgabewert:	-1 Fehler, Länge der tatsächlich gelesenen Daten.

Lesen der Seriennummer aus dem Gerät. Dieser Befehl wird nur von neuen Geräten unterstützt. Die Seriennummer wird als 4 Byte Nummer beginnend mit dem niederwertigsten Byte übertragen.

Programmierschnittstelle (SDK) der TS-RW Geräte

int TSLF_GetAntennaCount(int PortHandle);
PortHandle Zugriffsnummer
Rückgabewert: -1 Fehler, >0 Anzahl der Antennen des Gerätes

Diese Funktion wird nur von TS-RW82AC unterstützt, bei allen anderen Geräten wird 1 Antenne zurückgeliefert.

int TSLF_SetAntennaPowerPreset(int PortHandle, int Power);
PortHandle Zugriffsnummer
Power Spannungs Sollwert in Volt
Rückgabewert: -1 Fehler, 0 OK

Einstellung des Vorgabewertes für die Antennenspannung.
Diese Funktion wird nur von TS-RW82AC unterstützt.

int TSLF_GetAntennaPowerPreset(int PortHandle);
PortHandle Zugriffsnummer
Rückgabewert: -1 Fehler, >=0 Vorgabewert für die Antennenspannung in V

Lesen des Vorgabewertes für die Antennenspannung.
Diese Funktion wird nur von TS-RW82AC unterstützt.

int TSLF_GetAntennaPower(int PortHandle);
PortHandle Zugriffsnummer
Rückgabewert: -1 Fehler, >=0 Aktueller Wert der Antennenspannung in V

Hiermit wird der aktuelle Messwert der Antennenspannung erfasst.
Diese Funktion wird nur von TS-RW82AC unterstützt.

Programmierschnittstelle (SDK) der TS-RW Geräte

int TSLF_SetAnswerMode(int PortHandle, int AnswerMode**)**

PortHandle	Zugriffsnummer	
AnswerMode	Antwortmodus	
	0= Manchester	2 kBit Data Rate
	1= Biphase	4 kBit Data Rate
	2= FSK2a NRZ	2,5 kBit Data Rate
	4= FSK2a Manchester	1.25 kBit Data Rate
	5= Manchester	**4 kBit** Data Rate
	6= Biphase	**2 kBit** Data Rate
	7= FSK2a	**2 kBit** Data Rate

Rückgabewert: -1 Fehler, 0 OK

Dieses Kommando wird nur bei Geräten mit Animal Code Aktivierung unterstützt.
Hierbei wird Biphase (1) bei allen AC Geräten und bei FDX Geräten ab Firmware Version 'E' (14) unterstützt. FSK2a (2) wird nur bei TS-RW38Plus (243) und TS-RW38AC (244) unterstützt.
Manchester 4kBit und Biphase 2kBit wird nur bei TS-RW38Plus (243) und TS-RW38AC (244) jeweils ab Firmware Version 3 unterstützt.
FSK2a (7) wird nur bei TS-RW38Plus (243) und TS-RW38AC (244) jeweils ab Firmware Version 35 unterstützt.

Normalerweise senden die Transponder im Manchester Code. Bei bestimmten Anwendungen wird jedoch der Biphase Code (Animal FDX-B) oder FSK2a (Animal FDX-A) verwendet.
Je nach Transpondertyp wird dann nur der Animal Code im TTF Datenstrom in diesem Format gesendet oder der Transponder antwortet auf alle Kommandos in diesem Format.
Damit auch Transponder die auf Biphase bzw. FSK2a Code programmiert sind korrekt gelesen und geschrieben werden können, wird mit diesem Kommando die Auswertung der Antwort umgeschaltet.

Das Kommando hat Auswirkung auf Befehle für Q5, EM4569, EM4305, ATA5577 und ATA5575 Transponder.

Programmierschnittstelle (SDK) der TS-RW Geräte

int TSLF_GetAnswerMode(int PortHandle**)**
PortHandle Zugriffsnummer
Rückgabewert: -1 Fehler, >= 0 AnswerMode

Der AnswerMode ist wie bei TSLF_SetAnswerMode beschrieben.
Nur bei TS-RW38 Geräten wird der AnswerMode aus dem Gerät geladen, ansonsten ist es immer der AnswerMode wie ihn das SDK zuletzt gesetzt hatte.

Anmerkung:
Unterschiede der Transpondertypen:

Hitag 2, Hitag S und SIC7888
Bei diesen Transpondern erfolgt die Antwort auf Befehle immer im Manchester Code, auch wenn der TTF Datenstrom in Biphase erfolgt.

Q5
Bei Q5 erfolgt die Antwort auf Befehle immer gleich wie der TTF Datenstrom, also entweder alles in Biphase oder alles in Manchester Code.
Ausnahme ist die UID auf Page 1. Ist die Ausgabe von Page1 eingestellt, so wird unabhängig von der sonstigen Einstellung immer im Manchester Code gesendet. Die UID ist UNIQUE kodiert.

EM4569, EM4305, ATA5577 und ATA5575
Bei diesen Transpondertypen erfolgt die Antwort auf Befehle immer gleich wie der TTF Datenstrom, also entweder alles in Biphase oder alles in Manchester Code.

Hitag 1 und Titan
Diese Transpondertypen sind hiervon nicht betroffen, da diese Transponder immer in Manchester Kodierung arbeiten.

Programmierschnittstelle (SDK) der TS-RW Geräte

2.6. Schlüsselwert schreiben

Mit diesen Funktionen wird der Standardschlüssel zum Zugriff auf einen Transponder im Reader Modus bzw. bei den Schreibfunktionen eingestellt.
Es wird damit KEIN Passwort im Transponder gespeichert!

int TSLF_WriteHitag2Key(int PortHandle, BYTE * pKey, int KeyLen**)**

PortHandle	Zugriffsnummer
pKey	Zeiger auf Schlüssel
BufLen	Länge des Schlüssels (4 Byte)
Rückgabewert:	-1 Fehler, 0 OK

Beim Hitag 2 Transponder wird die Selektierung mit einem Passwort geschützt.
Das Passwort für die Selektierung wird mit diesem Befehl im Gerät hinterlegt.
Dies wird im Reader Modus beim Lesen von Blöcken für Hitag 2 verwendet, sowie im Programmer Modus beim Schreiben von formatierten Daten auf Hitag 2 Transponder.
Der voreingestellte Standardwert für das Hitag 2 Passwort ist „RKIM“(52 4B 49 4D).

int TSLF_WriteTagKey(int PortHandle, int TTyp, BYTE * pKey, int KeyLen**)**

PortHandle	Zugriffsnummer
TTyp	Transpondertyp, hier nur 30^{Hex} für ATA5577 mit Passwort zugelassen
pKey	Zeiger auf Schlüssel
BufLen	Länge des Schlüssels (4 Byte)
Rückgabewert:	-1 Fehler, 0 OK

Hiermit wird das Passwort für den Reader Modus für diesen Transpondertyp gesetzt. Wirksam wird dieses Passwort nur, wenn in den Parametern für Reader Modus auch der entsprechende Transpondertyp gesetzt wurde.

Programmierschnittstelle (SDK) der TS-RW Geräte

2.7. Allgemeines Kommando absetzen

Manche Geräte unterstützen zusätzliche Kommandos, die hier über eine allgemeine Funktion angesprochen werden können.

int TSLF_Transfer(int PortHandle, int Cmd, BYTE * pSendBuf, int SendBufLen,
BYTE * pRecvBuf, int RecvBufLen)

PortHandle	Zugriffsnummer
Cmd	Befehl der gesendet werden soll
pSendBuf	Zeiger auf zu sendende Daten
SendBufLen	Länge der zu sendenden Daten
pRecvBuf	Zeiger auf Empfangsdaten
RecvBufLen	Max. Länge der Empfangsdaten
Rückgabewert:	-1 Fehler, >= 0 Länge der gelesenen Daten in pRecvBuf

Soll direkt mit dem Gerät kommuniziert werden, z.B.: im Reader Modus ohne ein Datenprotokoll, so ist dies mit den folgenden Funktionen möglich:

int TSLF_RawWrite(int PortHandle, BYTE * pBuffer, int BufLen)

PortHandle	Zugriffsnummer
pBuffer	Zeiger auf Schreibpuffer
BufLen	Länge des Puffers
Rückgabewert:	-1 Fehler, 0 OK

Mit dieser Funktion kann man beliebige Daten an die Schnittstelle schreiben.
Es wird kein Protokoll vorausgesetzt.

int TSLF_RawRead(int PortHandle, BYTE * pBuffer, int BufLen)

PortHandle	Zugriffsnummer
pBuffer	Zeiger auf den, vom Benutzer zur Verfügung gestellten, Lesepuffer
BufLen	Länge des Puffers
Rückgabewert:	-1 Fehler, Länge der tatsächlich gelesenen Daten.

Mit dieser Funktion kann man beliebige Daten von der Schnittstelle lesen.
Es wird kein Protokoll vorausgesetzt.

Programmierschnittstelle (SDK) der TS-RW Geräte

2.8. Textausgabe an LCD

Der TS-RW38 kann mit einer LCD Anzeige ausgestattet werden.
Diese Anzeige kann mit folgenden Befehlen angesprochen werden.

Display Größe (22 x 33 mm)
Es kann zwischen zwei verschiedenen Schriftgrößen gewählt werden.
Kleine Schrift mit 8 Zeilen á 17 Zeichen
oder
Große Schrift mit 4 Zeilen á 8 Zeichen

int TSLF_ClearLCD (int PortHandle)

PortHandle	Zugriffsnummer
Rückgabewert:	-1 Fehler, 0 OK

Mit dieser Funktion wird die gesamte Anzeige gelöscht.

int TSLF_SetLCDFont (int PortHandle, int FontNr)

PortHandle	Zugriffsnummer
FontNr	0: kleine Schrift
	1: große Schrift
Rückgabewert:	-1 Fehler, 0 OK

Mit dieser Funktion wird die Schriftart zur Ausgabe gesetzt.

int TSLF_SetLCDText (int PortHandle, int StartLine, char * strText)

PortHandle	Zugriffsnummer
StartLine	Startzeile für die Textausgabe,
	Wertebereich 1-8 bei kleiner Schrift, 1-4 bei großer Schrift
strText	Auszugebender Text als Nullterminierter String.
Rückgabewert:	-1 Fehler, 0 OK

Mit dieser Funktion wird der angegebene Text am Display angezeigt.
Ist der Text länger als die Zeilenlänge wird automatisch umgebrochen, bis die max. Zeile erreicht wurde.

int TSLF_SetLCDLight(int handle, int Helligkeit)

handle	Zugriffsnummer
Helligkeit	Helligkeit der Hintergrundbeleuchtung, 0 = aus, 5 = max. Helligkeit

3. Kommandos für Readermodus

Diese Kommandos dienen der Parametrierung des Gerätes zum Betrieb im Readermodus sowie zur Datenübernahme im Readermodus

3.1 Parameter lesen und schreiben

Es können mehrere Parameter Datenstrukturen übergeben werden. Dann werden die angegebenen Transpondertypen abwechselnd ausgeführt.
Wird nur eine Datenstruktur übergeben, so muss diese nicht komplett gefüllt sein. Werden mehrere Datenstrukturen übergeben, so müssen die einzelnen Datenstrukturen immer komplett mit 20 Byte pro Datenstruktur übergeben werden.
Achtung nur der Gerätetyp TS-R38 unterstützt mehrere Datenstrukturen. Bei allen älteren Geräten wird nur eine Datenstruktur, teilweise werden dort auch nicht alle Parameter verwendet.
Der Parameter "BytesMaske" nur bei TS-R38 ab Version 1.31 unterstützt.
Sind in BytesMaske alle Bits gesetzt ($FF^{Hex)}$, so gilt für dieses Register die Einstellung von Zeichenanzahl und ValidBytes. Ansonsten werden nur genau die aktivierten Bytes ausgegeben. Dies ist nur bei ASCII und Hexadezimal Ausgabe erlaubt.

int TSLF_ReadParam(int PortHandle, BYTE * pBuffer, int BufLen**)**

PortHandle	Zugriffsnummer
pBuffer	Zeiger auf Lesebuffer. Siehe: **3.1.1 Aufbau der Parameter**.
BufLen	Länge des Puffers (20 Byte pro Datenstruktur)
Rückgabewert:	-1 Fehler, Länge der tatsächlich gelesenen Daten.

int TSLF_WriteParam(int PortHandle, BYTE * pBuffer, int BufLen**)**

PortHandle	Zugriffsnummer
pBuffer	Zeiger auf Schreibbuffer. Siehe: **3.1.1 Aufbau der Parameter**.
BufLen	Länge des Buffers (20 Byte pro Datenstruktur)
Rückgabewert:	-1 Fehler, 0 OK

Programmierschnittstelle (SDK) der TS-RW Geräte

3.1.1 Aufbau der Parameter

Die Parameter werden beim Schreiben und Lesen, immer in derselben Reihenfolge übergeben.

Pos.	Länge	Name	*Beschreibung*
0	1	TTyp	*Transpondertyp:* 00^{Hex} = Default (TTF) 08^{Hex} = EM4305 21^{Hex} = Tiris HDX 02^{Hex} = Hitag 2 09^{Hex} = ATA5577 22^{Hex} = NCD1015 HDX 03^{Hex} = Q5 / ATA5555 $0A^{Hex}$ = Hitag µ 23^{Hex} = Tiris HDX Plus 04^{Hex} = Hitag S $0B^{Hex}$ = ATA5575 24^{Hex} = SIC7900 05^{Hex} = Hitag 1 $0C^{Hex}$ = SIC7888 26^{Hex} SIC7999 06^{Hex} = Titan $0D^{Hex}$ = ATA5551 07^{Hex} = EM4569 30^{Hex} = ATA5577 (PW)
1	5	Register	Der Eintrag besteht immer aus bis zu 4 Registernummern und der Endekennung 0xffH. Dieser Eintrag kann verschiedenes bewirken: Ist bei Transpondertyp Default (TTF) = 0 eingestellt, so wird hier im ersten Byte das Format der Transponderantwort eingestellt: 01^{Hex} = Unique (EM4102) 02^{Hex} = GiS 16 Bit Format 03^{Hex} = Unique XL 04^{Hex} = FDX-B Format 05^{Hex} = FDX-A Format 06^{Hex} = HDX Format 07^{Hex} = FDX Waste(EN14803) 08^{Hex} = HDX Waste (EN14803) Wobei die FDX Typen nur bei TS-RW38+ und TS-RW38AC, die HDX Typen nur bei TS-RW38AC möglich sind. Ist ein Transpondertyp gewählt, so werden hier die Blocknummern definiert, die im Transponder gelesen werden sollen. Eine Sonderstellung nimmt hier die Blocknummer FE^{Hex} ein, die für die UID des Transponders steht. Byte 5 ist immer die Endekennung ff^{Hex}. z.B.: 00^{Hex},01^{Hex},$0f^{Hex}$,05^{Hex},ff^{Hex}.
6	1	Ausrichtung	*Ausrichtung* Ausrichtung der Daten 0 = LSB first 2 = LSB first bitgedreht 1 = MSB first 3 = MSB first bitgedreht
7	1	DTyp	*Datentyp:* 0 = Hexadezimal, 3 = Dezimal ohne führenden Nullen, 1 = Dezimal, 4 = Hexadezimal mit Kleinbuchstaben 2 = ASCII,
8	1	Zeichen	*Zeichenanzahl.* (1–32). Hier wird definiert wie viel Zeichen auf einmal ausgegeben werden sollen.
9	1	ValidBytes	(1-16) Anzahl der Bytes, die aus der UID oder den Datenblöcken verwendet werden. Hiermit können z.B.: die oberen Bits der UID ausgeblendet werden. Standardwert ist 5
10	1	ValidFrom	(1-16) Startbyte ab dem die Daten übertragen werden.
11	1	TypKenn1	Erkennungszeichen für den Transpondertyp wird vor den Daten übertragen, 1 ASCII Zeichen, bei 0 wird nichts übertragen
12	1	TypKenn2	Erkennungszeichen für den Transpondertyp wird nach den Daten übertragen, 1 ASCII Zeichen, bei 0 wird nichts übertragen
13	1	ModSpeed	Einstellung der Modulationsart und Geschwindigkeit 0: Manchester Code, 2 kBit Data Rate 1: Biphase Code, 4 kBit Data Rate 2: FSK2a Kodierung, 2.5 kBit Data Rate, NRZ 4: FSK2a Kodierung, 1.25 kBit Data Rate, Manchester Kodierung 5: Manchester Code, **4 kBit** Data Rate 6: Biphase Code, **2 kBit** Data Rate
14	4	BytesMaske	Der Eintrag besteht aus 4 Masken für die 4 oben angegebenen Register. In jeder Maske ist enthalten, welche Bytes aus dem Register ausgegeben werden. Das unterste Bit steht hierbei für das niederwertigste Byte.
18	2	-	Reserve, Standardwert 0

Programmierschnittstelle (SDK) der TS-RW Geräte

3.2 Prefix

Der Prefix ist die Zeichenkette die vor den Transponderdaten ausgegeben wird.
Der Prefix besteht maximal aus 31 Zeichen als Endekennung wird FFh eingesetzt.

int TSLF_WritePrefix(int PortHandle, BYTE * pBuffer, int BufLen**)**

PortHandle	Zugriffsnummer
pBuffer	Zeiger auf Schreibpuffer
BufLen	Länge des Puffers
Rückgabewert:	-1 Fehler, 0 OK

int TSLF_ReadPrefix(int PortHandle, BYTE * pBuffer, int BufLen**)**

PortHandle	Zugriffsnummer
pBuffer	Zeiger auf Lesepuffer
BufLen	Länge des Puffers
Rückgabewert:	-1 Fehler, Länge der tatsächlich gelesenen Daten.

3.3 Suffix

Der Suffix ist die Zeichenkette die nach den Transponderdaten ausgegeben wird.
Der Suffix besteht maximal aus 31 Zeichen als Endekennung wird FFh eingesetzt.

int TSLF_WriteSuffix(int PortHandle, BYTE * pBuffer, int BufLen**)**

PortHandle	Zugriffsnummer
pBuffer	Zeiger auf Schreibpuffer
BufLen	Länge des Puffers
Rückgabewert:	-1 Fehler, 0 OK

int TSLF_ReadSuffix(int PortHandle, BYTE * pBuffer, int BufLen**)**

PortHandle	Zugriffsnummer
pBuffer	Zeiger auf den Lesepuffer
BufLen	Länge des Puffers
Rückgabewert:	-1 Fehler, Länge der tatsächlich gelesenen Daten.

Programmierschnittstelle (SDK) der TS-RW Geräte

3.4 Termix

Der Termix ist die Zeichenkette die zwischen zwei Transponder Registern ausgegeben wird.
Der Termix besteht maximal aus 31 Zeichen als Endekennung wird FFh eingesetzt.

int TSLF_WriteTermix(int PortHandle, BYTE * pBuffer, int BufLen**)**

PortHandle	Zugriffsnummer
pBuffer	Zeiger auf Schreibpuffer
BufLen	Länge des Puffers
Rückgabewert:	-1 Fehler, 0 OK

int TSLF_ReadTermix(int PortHandle, BYTE * pBuffer, int BufLen**)**

PortHandle	Zugriffsnummer
pBuffer	Zeiger auf den Lesepuffer
BufLen	Länge des Puffers
Rückgabewert:	-1 Fehler, Länge der tatsächlich gelesenen Daten.

3.5 Postcode

Der Postcode ist die Zeichenkette die beim Abziehen des Transponders ausgegeben wird.
Der Postcode besteht maximal aus 31 Zeichen als Endekennung wird FFh eingesetzt.

int TSLF_WritePostcode(int PortHandle, BYTE * pBuffer, int BufLen**)**

PortHandle	Zugriffsnummer
pBuffer	Zeiger auf Schreibpuffer
BufLen	Länge des Puffers
Rückgabewert:	-1 Fehler, 0 OK

int TSRLF_ReadPostcode(int PortHandle, BYTE * pBuffer, int BufLen**)**

PortHandle	Zugriffsnummer
pBuffer	Zeiger auf den Lesepuffer
BufLen	Länge des Puffers
Rückgabewert:	-1 Fehler, Länge der tatsächlich gelesenen Daten.

Programmierschnittstelle (SDK) der TS-RW Geräte

3.6 Reader Mode Parameter

Schreiben und lesen der Reader Mode Parameter

int TSLF_WriteReadModeParam(int PortHandle, BYTE * pBuffer, int BufLen**)**

PortHandle	Zugriffsnummer
Pbuffer	Zeiger auf Schreibpuffer. Siehe: **3.6.1 Aufbau der Reader Mode Parameter**
BufLen	Länge des Puffers
Rückgabewert:	-1 Fehler, 0 OK

int TSLF_ReadReadModeParam(int PortHandle,BYTE * pBuffer, int BufLen**)**

PortHandle	Zugriffsnummer
pBuffer	Zeiger auf den Lesepuffer Siehe: **3.6.1 Aufbau der Reader Mode Parameter**
BufLen	Länge des Puffers
Rückgabewert:	-1 Fehler, Länge der tatsächlich gelesenen Daten.

3.6.1. Aufbau der Reader Mode Parameter

Die Parameter werden beim Schreiben und Lesen, immer in derselben Reihenfolge übergeben.

Data 1	**Data 2**	**Data 3**	**Data 4**
Mode	Zykluszeit	Anforderung	Timeout

Mode: Betriebsart
0: Daten bei Auflegen und Abziehen des Transponders senden
1: Daten auf Anforderung senden.
Das Anforderungszeichen wird in **Anforderung** definiert.
Das Gerät sendet Daten, wenn die Anforderung zum Geräte gesendet wurde.
3: Daten zyklisch senden
5: Daten bei Auflegen und Abziehen des Transponders senden.
Zusätzlich werden bei Auflegen des Transponders die Ausgänge 1 und 2 gesetzt und nach der eingestellten Zykluszeit wieder rückgesetzt. Dieser Modus ist nur bei Geräten mit Ausgängen einstellbar.

Zykluszeit: Die Zeit wird in Zehntelsekunden eingestellt. Der Standardwert ist 10, das bedeutet also 1 Sekunde. Die Zykluszeit ist im Mode 3 und 5 wirksam.

Anforderung: Anforderungszeichen, das im Mode 1 wirksam ist. Standardwert ist '?' (3fH)

Timeout: Die Zeit wird in Zehntelsekunden eingestellt. Der Standardwert ist 20, das bedeutet als 2 Sekunden. Der Timeout ist im Mode 0 und 5 gültig. Der Timeout gibt an, wie lange ein Transponder aus dem Feld sein muss, um wieder erkannt zu werden.

3.7. Datenübernahme im Readermodus

Die Daten, die im Readermodus übermittelt werden, können auf verschiedene Arten übernommen werden.

3.7.1. Zyklische Abfrage

Die empfangenen Daten können mittels des TSLF_RawRead Kommandos zyklisch abgefragt und aus den empfangenen Daten der Datensatz zusammengebaut werden. Hierbei ist eine Überprüfung des Datenendes durch die aufrufende Instanz notwendig.

3.7.2. Callback Funktion einrichten

Mittels einer Anwendungsspezifischen Callback Funktion kann der Datensatz vom SDK empfangen und ausgewertet werden und dann der gesamte Dateninhalt an die Funktion als ein String übergeben werden. Hierzu ist es notwendig, dass die Übertragung mit einem eindeutigen definierten Zeichen endet. Üblicherweise wird hier CR = 0DH verwendet.

int **TSLF_StartAutoRead**(int PortHandle, int TermChar, AutoReadCallback pAutoReadProc)

PortHandle	Zugriffsnummer
TermChar	Endezeichen der Übertragung. Mit diesem Zeichen wird der Aufruf der Callback Funktion getriggert.
pAutoReadProc	Zeiger auf die Callback Funktion
Rückgabewert:	-1 Fehler, 0: OK

Definition der Callback Funktion:
'C'

```
typedef int(__stdcall* AutoReadCallback)(char * pData, int Len);
```

'C#'

```
delegate int AutoReadCallback(      [MarshalAsAttribute(UnmanagedType.LPStr)] string pData,
                                    int Len);
```

'VB'

```
Delegate Function AutoReadCallback ( <MarshalAs(UnmanagedType.LPStr)> Arr As String,
                                     ByVal Len As Integer) As Integer
```

Die Verwendung der Callback Funktion ist in der "Readermodus" Beispielapplikation beschrieben. Die Callback Funktion wird mit einem Leerstring und Len=0 aufgerufen, wenn das verwendete Gerät nicht mehr verfügbar ist. Z.B.: wenn ein USB Gerät während des Betriebes abgesteckt wurde.

int **TSLF_StopAutoRead**(int PortHandle)

PortHandle	Zugriffsnummer
Rückgabewert:	-1 Fehler, 0: OK

Beendet den Aufruf der Callback Funktion.

3.7.3. LED und Buzzer setzen

Diese Funktionen werden nur bei TS-R38/RW38 ab Version 1.38 unterstützt.

int **TSLF_SetLED**(int PortHandle, int red, int green, int yellow)

PortHandle	Zugriffsnummer
red	-1: Rote LED nicht verändern
	0: Rote LED ausschalten
	1: Rote LED einschalten
	2: Kontrolle über Rote LED an Gerät zurückgeben
green	-1: Grüne LED nicht verändern
	0: Grüne LED ausschalten
	1: Grüne LED einschalten
	2: Kontrolle über Grüne LED an Gerät zurückgeben
yellow	-1: Gelbe LED nicht verändern
	0: Gelbe LED ausschalten
	1: Gelbe LED einschalten
	2: Kontrolle über Gelbe LED an Gerät zurückgeben
Rückgabewert:	-1 Fehler, 0: OK

int **TSLF_Beep**(int PortHandle)

PortHandle	Zugriffsnummer
Rückgabewert:	-1 Fehler, 0: OK

Aktiviert den Summer für 200 mSek. Natürlich abhängig davon, ob im Gerät der optionale Summer eingebaut ist oder nicht.

4. Befehle für alle Transpondertypen im Programmer Modus

Liste der Transpondertypen

Nummer	Transpondertyp	Hersteller	FDX-A	FDX-B	TTF Unique	Kommentar
0	undefiniert	-				
2	Hitag 2	NXP		X	X	
3	Q5	Atmel / Sokymat	X	X	X	auch ATA5555
4	Hitag S	NXP		X	X	
5	Hitag 1	NXP				
6	Titan 1100	Sokymat				auch EM4050, EM4450, EM4550
7	EM4569	Electro Marin		X	X	
8	EM4305	Electro Marin	X	X	X	
9	ATA5577	Atmel	X	X	X	auch ATA5567
10 ($0a^{Hex}$)	Hitag µ	NXP		X	X	
11 ($0b^{Hex}$)	ATA5575	Atmel	X	X	X	
12 ($0c^{Hex}$)	SIC7888	Silicon Craft	X	X	X	ähnlich zu Hitag S
13 ($0d^{Hex}$)	ATA5551	Atmel	X	X	X	
14 ($0e^{Hex}$)	EL9265	Excelio		X	X	ähnlich zu Hitag S
15 ($0f^{Hex}$)	EM4582	Electro Marin				
33 (21^{Hex})	Tiris HDX	Texas Instruments				HDX Format TMS37124
34 (22^{Hex})	NCD1015	Incide				HDX Format
35 (23^{Hex})	Tiris HDX+	Texas Instruments				HDX Format TMS37190
36 (24^{Hex})	SIC7900	Silicon Craft				HDX Format
37 (25^{Hex})	Tiris HDX+	Texas Instruments				HDX TMS37190 mit High Q Factor
38 (26^{Hex})	SIC7999	Silicon Craft				HDX-Format

Programmierschnittstelle (SDK) der TS-RW Geräte

4.1. Rücksetzen des Transponders

int TSLF_ResetTransponder(int PortHandle, int TTyp**)**
PortHandle Zugriffsnummer
TTyp Transpondertyp aus Tabelle
Rückgabewert: -1 Fehler, 0 OK
Setzt den aufgelegten Transponder zurück, je nach Transpondertyp wird entweder der Reset Befehl (Q5, ATA5577 und Titan) oder eine Sequenz mit SetRF(0) … SetRF(1) verwendet. Hier darf als Transpondertyp auch 0 angegeben werden, dann wird immer die SetRF(..) Sequenz verwendet.

4.2. Erkennung des Transpondertyps

Diese Funktion dient der Erkennung des Typs eines Transponders.
Es wird dabei versucht aufgrund transponderspezifischer Eigenschaften den Typ des Transponder zu erkennen. Es gibt jedoch auch Fälle in denen der Transpondertyp nicht sicher erkannt werden kann. Bei HDX Transpondern wird nur HDX erkannt, und als Rückgabewert Tiris HDX (33) zurückgegeben. Eine Unterscheidung der HDX Transpondertypen wird hier nicht durchgeführt.

int TSLF_GetTagType(int PortHandle, int TTypExpected)
PortHandle Zugriffsnummer
TTypExpected Erwarteter Transpondertyp aus Tabelle
Rückgabewert: -1 Fehler,
0: Kein Transponder erkannt
>0 Gefundener Transpondertyp

Erkennung des Antwortmodus des Transpondertyps

int TSLF_GetTagAnswerMode(int PortHandle, int TTyp, int AnswerModeExpected)
PortHandle Zugriffsnummer
TTyp Transpondertyp aus Tabelle
AnswerModeExpected Erwarteter Antwortmodus des Transponders
Rückgabewert: -1 Fehler,
>=0 Gefundener Antwortmodus

Siehe auch die Beschreibung zur Funktion TSLF_SetAnswerMode.
Die Erkennung des Antwortmodus erfolgt nur bei den entsprechenden Transpondertypen, die verschiedene Antwortmodi unterstützten. Bei den anderen Transpondertypen wird immer der übergebene Wert AnswerModeExpected zurückgegeben.

Programmierschnittstelle (SDK) der TS-RW Geräte

4.3. Selektieren eines Transponders

Je nach Transpondertyp muss dieser vor dem Zugriff durch die Lese und Schreibfunktionen selektiert werden.

Bei Hitag 1, Hitag 2, Hitag S und SIC7888 Transpondern ist die Selektierung obligatorisch.
Auf Hitag µ Transponder kann mit oder ohne Selektierung zugegriffen werden.
Bei Titan, EM4569 und EM4305 sowie SIC7888 Transpondern kann ein Login notwendig sein wenn der Transponder Passwortgeschützt ist.
Bei Q5 und ATA5577 Transpondern müssen gesonderte Funktionen verwendet werden, wenn im Passwortmodus zugegriffen werden soll.

int TSLF_UID_Request(int PortHandle, int TTyp, BYTE * pUID, int UIDLen**)**

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
pUID	Zeiger auf die UID die mit diesem Befehle gelesen wird.
UIDLen	max. Länge der UID (abhängig vom Transpondertyp)
Rückgabewert	-1 Fehler, >0 Länge der gelesenen Daten

Mit diesem Befehl wird die UID (Unique Identifier = Eindeutige Seriennummer) des Transponders ermittelt. Die UID wird je nach Transpondertyp unterschiedlich ermittelt. Es darf dabei nur ein Transponder im Antennenfeld sein.
Die Länge der UID ist vom Transpondertyp abhängig.
Bei Hitag Transpondern ist diese Funktion vor Anwendung der TSLF_Select Funktion vorgeschrieben.

Transpondertyp	UID Länge
Hitag 1	4 Byte
Hitag 2	4 Byte
Hitag S	4 Byte
Hitag µ	6 Byte
SIC7888	4 Byte
Q5	5 Byte
ATA5577	8 Byte
ATA5575	-
ATA5551	-
EL9265	4 Byte

Transpondertyp	UID Länge
Titan 1100	4 Byte
EM4569	4 Byte
EM4305	4 Byte
EM4582	4 Byte
Tiris HDX	8 Byte
Tiris HDX+	8 Byte
NCD1015	8 Byte
SIC7900	8 Byte
SIC7999	8 Byte

Programmierschnittstelle (SDK) der TS-RW Geräte

int TSLF_UID_RequestAC(int PortHandle, int TTyp, BYTE * pUID, int UIDLen**)**

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
pUID	Zeiger auf die UID die mit diesem Befehle gelesen wird.
UIDLen	max. Länge der UID (abhängig vom Transpondertyp)
Rückgabewert	-1 Fehler, >0 Länge der gelesenen Daten

Mit diesem Befehl können auch mehrere Transponder im Feld erkannt werden. Diese Funktionalität wird nur von Hitag1, HitagS und SIC7888 Transpondern unterstützt. Es werden die UID aller gefundenen Transponder gemeldet. Da jede UID eine Länge von 4 Byte hat, ist der Rückgabewert also ein Vielfaches von 4. Eventuell werden nicht alle Transponder mit dem ersten Aufruf gefunden. Durch Selektieren des Transponders mit TSLF_Select und deaktivieren mit TSLF_Disable können die gefundenen Transponder stummgeschaltet und anschließend mit TSLF_UID_RequestNext weitere Transponder gefunden werden.
Diese Funktion ist nur mit TS-RW38+ ab Firmware 1.27 möglich.

int TSLF_UID_RequestNext(int PortHandle, int TTyp, BYTE * pUID, int UIDLen**)**

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
pUID	Zeiger auf die UID die mit diesem Befehle gelesen wird.
UIDLen	max. Länge der UID (abhängig vom Transpondertyp)
Rückgabewert	-1 Fehler, >0 Länge der gelesenen Daten

Mit diesem Befehl werden weitere Transponder nach einem Aufruf von TSLF_UID_RequestAC gefunden. Hierbei ist darauf zu achten, dass vorher gefundene Transponder mit TSLF_Disable deaktiviert wurden, da diese ansonsten immer wieder gefunden werden.
Diese Funktion ist nur mit TS-RW38+ ab Firmware 1.27 möglich.

int TSLF_Select(int PortHandle, int TTyp, BYTE * pSelData, int SelDataLen,
BYTE * pConfig, int ConfigLen**)**

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
pSelData	Zeiger auf UID oder Passwort
SelDataLen	Länge der UID bzw. Passswort
pConfig	Zeiger auf das Konfigurationswort, das mit diesem Befehl gelesen wird.
ConfigLen	Länge des Konfigurationswort (muss fest auf 4 Byte sein)
Rückgabewert	-1 Fehler, >0 Länge der gelesenen Daten

In dieser Funktion werden nur für Hitag und SIC7888 Transponder unterstützt.
Bei Hitag1, Hitag S, SIC7888 und Hitag µ wird in pSelData die UID aus TSLF_UID_Request übergeben, wobei die Länge der UID bei Hitag 1, Hitag S und SIC7888 4 Byte und bei Hitag µ 6 Byte beträgt.
Bei Hitag 2 wird in pSelData das Tag Passwort übergeben.
Das Standard Passwort lautet „RKIM“ (52 4B 49 4D).

Programmierschnittstelle (SDK) der TS-RW Geräte

int TSLF_GetTagInfo(int PortHandle, int TTyp, BYTE * pInfo, int InfoLen**)**

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
pInfo	Zeiger auf Information
InfoLen	Länge der Information
Rückgabewert	-1 Fehler, >0 Länge der gelesenen Daten

Mit dieser Funktion können zusätzliche Daten über den Chip ausgelesen werden.
Bei Hitag µ Transpondern wird die „SystemInfo“ des Transponders gelesen.
Bei SIC7999 HDX Transpondern wird die Chipkonfiguration bestehend aus den Traceability Data, Flags, Ctune, Manufacturer Data ausgelesen (siehe SIC7999 Datenblatt)
Andere Transpondertypen werden von dieser Funktion nicht unterstützt.

int TSLF_SetCTune(int PortHandle, int TTyp,
int CTuneVal, int bUseValue, int bOne, int bLock, int * pFrequenz)

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
CTuneVal	Tuning Wert der verwendet werden soll
bUseValue	0: Funktion mit bisher gesetztem Wert ausführen 1: Tuning Wert verwenden
bOne	0: Frequenz für 0-Bits bestimmen 1: Frequenz für 1-Bits bestimmen
bLock	1: Tuning Wert im Transponder irreversibel speichern
pFrequenz	Im Inhalt dieses Zeigers wird die gemessene Frequenz zurückgegeben. Frequenz in Hz.
Rückgabewert	-1 Fehler, 0 OK

Mit dieser Funktion kann der Tuning Wert geschrieben, sowie die Frequenz der Datenübertragung vom Transponder gemessen werden. Damit ist ein genauer Abgleich des Transponders möglich. Der optimale Tuning Wert muss für jeden Transponder getrennt, durch mehrere Aufrufe dieser Funktion ermittelt werden. Diese Funktion wird nur für SIC7999 unterstützt.

int TSLF_AutoTune(int Porthandle, int TType, int * pFreq0, int * pFreq1, int Lock)

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
pFreq0	Frequenz für '0' Bits, normalerweise 134600 Hz
pFreq1	Frequenz für '1' Bits, normalerweise 124200 Hz
bLock	1: Tuning Wert im Transponder irreversibel speichern
Rückgabewert	-1: Fehler, >0 verwendeter Tuning Wert

Diese Funktion führt den kompletten Tuning Prozess aus. Die angegebenen Frequenzen werden für das Tuning verwendet. Rückgabewert ist der verwendete Tuning Wert, außerdem wird in pFreq0 und pFreq1 die tatsächlich erzielte Frequenz zurückgeliefert. Diese Funktion wird nur für SIC7999 unterstützt.

Programmierschnittstelle (SDK) der TS-RW Geräte

int TSLF_GetTagConfig(int handle, int TTyp, int * pConfig)

handle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
pConfig	Zeiger auf gelesene Konfiguration

Diese Funktion liest die Konfiguration eines Transponders aus.
Diese Funktion wird nur für TMS 37190 Transponder unterstützt.

int TSLF_SetTagConfig(int handle, int TTyp, int Config)

handle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
Config	zu schreibende Konfiguration

Diese Funktion schreibt die Konfiguration eines Transponders.
Diese Funktion wird nur für TMS 37190 Transponder unterstützt.

Die Konfiguration eines TMS 37190 beinhaltet den Lock Status sowie die Kennung ob es sich um einen Low Q oder High Q Transponder handelt.

Config: Konfigurationsbyte
- Bit 0-2: IC Life Cycle
  - 011b: Configuration/Production Phase, Chip ist veränderbar
  - 000b: Application Phase, Chip ist nicht mehr veränderbar
- Bit 3: Q-Factor: (je nach Transpondertyp wählen)
  - 0: Low Q Factor
  - 1: High Q Factor
- Bit 4: Discharge (soll auf 0 gesetzt werden)
  - 0: keine Entladung
  - 1: Entladung nach Antwort
- Bit 5-7: RFU (immer auf 0 setzen)

Programmierschnittstelle (SDK) der TS-RW Geräte

int TSLF_Disable (int PortHandle, int TTyp**)**
PortHandle Zugriffsnummer
TTyp Transpondertyp aus Tabelle

Hiermit wird der Transponder inaktiv geschaltet.
Dieser Befehl wirkt bei EM4569 und EM4305 Transpondern nur wenn die Option in der Konfiguration des Transponders freigeschaltet ist. Bitte beachten sie das Datenblatt des Transponders.
Bei Hitag µ Transpondern wird hier das Stay Quiet Kommando, bei dem der Transponder selektiert sein muss, verwendet.
Bei Hitag S wird das Quiet Kommando, bei dem der Transponder selektiert sein muss, verwendet.
Bei Hitag 1 wird das Halt Kommando, bei dem der Transponder selektiert sein muss, verwendet.
Bei SIC7888 wird das Silent Kommando, bei dem der Transponder selektiert sein muss, verwendet.
Um den Transponder wieder zu aktivieren, muss er zunächst zurückgesetzt und dann wieder in den entsprechenden Zustand gebracht werden.

int TSLF_WakeUp(int PortHandle, int TTyp, BYTE * pPasswort, int PasswortLen**)**
PortHandle Zugriffsnummer
TTyp Transpondertyp aus Tabelle
pPasswort Zeiger auf Passwort mit dem auf den Transponder zugegriffen wird.
PasswortLen Länge des Passworts (muss fest auf 4 Byte sein)
Rückgabewert: -1 Fehler, 0 OK

Aktivieren eines Transponders, der im AOR Modus programmiert ist. Dieser Befehl kann bei Q5 und ATA5577 Transpondern verwendet werden. Der Transponder muss entsprechend konfiguriert sein. Bitte beachten sie das Datenblatt des Transponders.

4.4. Zugriff auf Passwortgeschützte Transponder

Bei verschiedenen Transpondern ist ein Passwortschutz möglich. Dieser Passwortschutz ist auf unterschiedliche Art implementiert.

4.4.1. Passwortschutz bei EM4569, EM4305, Hitag µ, SIC7888 und Titan

Bei diesen Transpondern kann der Zugriff auf Passwortgeschützte Inhalte durch die Funktion TSLF_Login freigegeben werden.

int TSLF_Login(int PortHandle, int TTyp, BYTE * pPW, int nPWLen**)**

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
pPW	Zeiger auf das Passwort mit dem auf den Transponder zugegriffen wird.
nPWLen	Länge des Passworts (muss fest auf 4 Byte sein)
Rückgabewert:	-1 Fehler, 0 OK

Die Login Funktion wird benötigt, wenn der Transponder passwortgeschützt ist. Zu den Möglichkeiten des Passwortschutzes der verschiedenen Transpondertypen beachten sie bitte deren Datenblätter.
Der Login Befehl wird von den Transpondertypen Titan, EM4569, EM4305, Hitag µ und SIC7888 direkt unterstützt. Beim SIC7888 wird üblicherweise ein Login Read durchgeführt. Durch Erweiterung des Transpondertyps um 100hex (SIC_LOGIN_RW) wird der Login für Read und Write verwendet.
Der Zugriff mit Passwort auf Q5 und ATA5577 Transponder wird durch die Login Funktion aktiviert. Hier wird das Passwort im SDK gemerkt und anschließende Lese und Schreibzugriffe auf die Transponder werden als Befehle mit Passwort ausgeführt. Der Passwortzugriff wird zurückgenommen sobald der Transponder zurückgesetzt (TSLF_ResetTransponder) wird.

Das Passwort ist bei den meisten Transpondertypen in einem definierten Block abgelegt und wird mit der TSLF_Write Funktion gesetzt.
Hierzu muss meist vorher ein Login mit dem alten Passwort gemacht werden.

Eine Ausnahme ist der Titan Transponder, bei diesem wird mit einem separaten Befehl das Passwort gesetzt. Hierbei muss das bisherige Passwort angegeben werden.

int TSLF_WritePW(int PortHandle, int TTyp, BYTE * pOldPW, int OldPWLen,
BYTE * pNewPW, int NewPWLen**)**

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
pOldPW	Zeiger auf altes Passwort
OldPWLen	Länge des alten Passworts (muss fest auf 4 Byte sein)
pNewPW	Zeiger auf neues Passwort
NewPWLen	Länge des neuen Passworts (muss fest auf 4 Byte sein)
Rückgabewert:	-1 Fehler, 0 OK

Zulässig nur für Titan Transponder!

4.4.2. Passwortschutz bei Hitag 2

Hitag 2 Transponder können in zwei verschiedenen Betriebsarten verwendet werden. Im Passwort Modus wird bereits mit dem TSLF_Select Kommando der Passwortgeschützte Zugriff aktiviert. Im Crypto Modus muss mit der Funktion TSLF_Authenticate anstelle der Funktion TSLF_Select die Selektierung durchgeführt werden.

4.4.3. Passwortschutz bei Hitag S und EL9265

Normalerweise werden Hitag S und EL9265 Transponder ohne Passwortschutz verwendet.
Es ist jedoch auch möglich diese Transponder im „Authentication Mode“ zu verwenden. Dann muss nach dem TSLF_Select Kommando noch zusätzlich mit TSLF_Authenticate der Transponder authentifiziert werden.
Anschließend kann dann auf den Transponder normal zugegriffen werden.

int TSLF_Authenticate(int PortHandle, int TTyp, BYTE * pKey, int nKeyLen,
BYTE * pConfig, int nConfigLen**)**

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
pKey	Zeiger auf den Key mit dem auf den Transponder zugegriffen wird.
nKeyLen	Länge des Keys (muss fest auf 6 Byte sein)
pConfig	Konfiguration (Antwort auf Challenge Kommando bei Hitag S und EL9265, bzw Config Byte/Password Tag bei Hitag 2)
nConfigLen	Max. Länge der Konfiguration (4 Byte)
Rückgabewert:	-1 Fehler, > 0 Länge der gelesenen Konfiguration

Diese Funktion wird nur bei Hitag S, EL9265 und Hitag 2 verwendet. Bei Hitag S oder EL9265 muss die Aufruffolge zwingend TSLF_UID_Request, TSLF_Select, TSLF_Authenticate sein.
Bei Hitag 2 muss die Aufruffolge TSLF_UID_Request, TSLF_Authenticate sein (ohne TSLF_Select).
Der Key wird mit LSB First angegeben.
Standardkey für Hitag S, EL9265 und Hitag 2 ist 4D 49 4B 52 4F 4E („MIKRON“)
Diese Funktion erhält den Key als Eingangsparameter und liefert die Konfiguration als Ausgangsparameter.

Programmierschnittstelle (SDK) der TS-RW Geräte

4.5. Blockweises Lesen und Schreiben

int TSLF_Read(int PortHandle, int TTyp, int StartBlock, BYTE * pBuffer, int BufLen**)**

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
StartBlock	Blocknummer
pBuffer	Zeiger auf die Transponderdaten
BufLen	Länge der Transponderdaten in Bytes
Rückgabewert:	-1 Fehler, >0 Länge der gelesenen Daten

Bei dieser Funktion werden so viele aufeinanderfolgende Blöcke gelesen wie in Länge der Transponderdaten angegeben ist. Hierbei werden immer nur ganze Blöcke gelesen.
Beispiel: Bei einer Blocklänge von 4 Bytes und BufLen = 12 werden 3 Blöcke gelesen, bei BufLen = 10 werden 2 Blöcke gelesen und der Rückgabewert ist dann 8.

int TSLF_Write(int PortHandle, int TTyp, int StartBlock,BYTE * pBuffer, int BufLen , int Lock**)**

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
StartBlock	Blocknummer
pBuffer	Zeiger auf die Transponderdaten
BufLen	Länge der Transponderdaten
Lock	0 = Standardwert. Wird nur bei **Q5** und **ATA5577** benötigt und ausgewertet. Wenn dieser Wert 1 ist, wird dieser Block des Q5 bzw ATA5577 Transponders schreibgeschützt.
Rückgabewert:	-1 Fehler, 0 OK

Es werden immer nur ganze Blöcke geschrieben, wird also als Länge nicht ein Vielfaches der Blockgröße angegeben, so wird nur bis zum letzten komplett gefüllten Block geschrieben.

Der TTyp kann mit der Option: „**Read after write“** verknüpft werden. Diese Option löst ein sofortiges Lesen nach dem Schreiben aus. Anschließend erfolgt ein Vergleich zwischen gelesenen und geschriebenen Daten. Ist der Vergleich negativ, kehrt die Funktion mit Fehler zurück. GetLastError ermittelt den Fehler 20. (siehe Anhang). Speziell beim Q5 ist es möglich, die Daten invers zu lesen. In diesem Falle muss **„Read after write invers“** benutzt werden.
Read after write = 100 Hex Read after write invers = 200 Hex

Beim Transpondertyp EM4305 wird das Protection Word in den Blöcken 14 und 15 abgelegt, wobei aufgrund der gelesenen Daten zu unterscheiden ist, welcher Block das gültige Wort enthält.
Ein Schreiben des Protection Wortes kann nur durch einen speziellen Befehl erreicht werden, der hier automatisch ausgeführt wird, wenn auf Block 14 oder 15 geschrieben wird.
Bei Transpondertyp ATA5577 erfolgt der Zugriff auf Page1 durch Angabe der Blocknummer 8 – 11.

4.6. Formatiertes Schreiben

int TSLF_Write_Unique(int PortHandle, int TTyp, BYTE * pBuffer, int BufLen, int Lock**)**

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
pBuffer	Zeiger auf die Transponderdaten
BufLen	Länge der Transponderdaten (muss fest auf 5 Byte sein)
Lock	0 = Standardwert. Wenn dieser Wert 1 ist, werden im Transponder die entsprechenden Blöcke schreibgeschützt.
Rückgabewert:	-1 Fehler, 0 OK

Mit dieser Funktion kann man Transponder im Unique-Format (EM4102 kompatibel) beschreiben. Die Schreibfunktion funktioniert mit Q5, Hitag2, Hitag S, EHitag µ, EM4569, EM4305, SIC7888, EL9265, ATA5577 und ATA5575 Transpondern. Der Transponder wird hierzu entsprechend konfiguriert. (64 Bit = 2 Blöcke Transponder Talks First in Manchester Modulierung, 2kBit). Ob der Transpondertyp unterstützt wird hängt vom Gerät ab.

int TSLF_Write_UniqueXL(int PortHandle, int TTyp, BYTE * pBuffer, int BufLen, int Lock**)**

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
pBuffer	Zeiger auf die Transponderdaten
BufLen	Länge der Transponderdaten (muss fest auf 11 Byte sein)
Lock	0 = Standardwert. Wenn dieser Wert 1 ist, werden im Transponder die entsprechenden Blöcke schreibgeschützt.
Rückgabewert:	-1 Fehler, 0 OK

Mit dieser Funktion kann man Transponder im Unique-XL Format beschreiben. Die Schreibfunktion funktioniert mit Q5, Hitag2, Hitag S, Hitag µ, EM4569, EM4305, SIC7888, EL9265, ATA5577 und ATA5575 Transpondern. Der Transponder wird hierzu entsprechend konfiguriert. (128 Bit = 4 Blöcke Transponder Talks First in Manchester Modulierung, 2kBit). Ob der Transpondertyp und dieses Format unterstützt werden hängt vom Gerät ab.

int TSLF_Write_FastUnique(int PortHandle, int TTyp, BYTE * pBuffer, int BufLen, int Lock**)**

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
pBuffer	Zeiger auf die Transponderdaten
BufLen	Länge der Transponderdaten (muss fest auf 5 Byte sein)
Lock	0 = Standardwert. Wenn dieser Wert 1 ist, werden im Transponder die entsprechenden Blöcke schreibgeschützt.
Rückgabewert:	-1 Fehler, 0 OK

Mit dieser Funktion kann man Transponder im Unique-Datenformat, aber mit Manchester 4kBit Übertragungsgeschwindigkeit beschreiben.
Diese Funktion wird nur bei TS-RW38Plus und TS-RW38AC unterstützt.
Unterstützte Transpondertypen sind Q5, ATA5577, EM4569 und EM4305

Programmierschnittstelle (SDK) der TS-RW Geräte

int TSLF_Write_GiS16Bit(int PortHandle, int TTyp, BYTE * pBuffer, int BufLen, int Lock**)**

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
pBuffer	Zeiger auf die Transponderdaten
BufLen	Länge der Transponderdaten (muss fest auf 2 Byte sein)
Lock	0 = Standardwert. Wenn dieser Wert 1 ist, werden im Transponder die entsprechenden Blöcke schreibgeschützt.
Rückgabewert:	-1 Fehler, 0 OK

Mit dieser Funktion kann man Transponder im GiS 16Bit Format beschreiben. Die Schreibfunktion funktioniert mit Q5, Hitag2, Hitag S, Hitag µ, EM4569, EM4305, SIC7888, EL9265, ATA5577 und ATA5575 Transpondern. Der Transponder wird hierzu entsprechend konfiguriert. (64 Bit = 2 Blöcke Transponder Talks First in Manchester Modulierung, 2kBit). Ob der Transpondertyp unterstützt wird hängt vom Gerät ab.

int TSLF_Write_GiS12Bit(int PortHandle, int TTyp, BYTE * pBuffer, int BufLen, int Lock**)**

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
pBuffer	Zeiger auf die Transponderdaten
BufLen	Länge der Transponderdaten (muss fest auf 2 Byte sein)
Lock	0 = Standardwert. Wenn dieser Wert 1 ist, werden im Transponder die entsprechenden Blöcke schreibgeschützt.
Rückgabewert:	-1 Fehler, 0 OK

Mit dieser Funktion kann man Transponder im GiS 12Bit Format beschreiben. Die Schreibfunktion funktioniert mit Q5, Hitag2, Hitag S, Hitag µ, EM4569, EM4305, SIC7888, EL9265, ATA5577 und ATA5575 Transpondern. Der Transponder wird hierzu entsprechend konfiguriert. (64 Bit = 2 Blöcke Transponder Talks First in Manchester Modulierung, 2kBit). Ob der Transpondertyp und dieses Format unterstützt wird hängt vom Gerät ab.

int TSLF_Write_GiS72Bit(int PortHandle, int TTyp, BYTE * pBuffer, int BufLen, int Lock**)**

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
pBuffer	Zeiger auf die Transponderdaten
BufLen	Länge der Transponderdaten (muss fest auf 9 Byte sein)
Lock	0 = Standardwert. Wenn dieser Wert 1 ist, werden im Transponder die entsprechenden Blöcke schreibgeschützt.
Rückgabewert:	-1 Fehler, 0 OK

Mit dieser Funktion kann man Transponder im GiS 72Bit Format beschreiben. Die Schreibfunktion funktioniert nur mit Hitag S Transpondern. Der Transponder wird hierzu entsprechend konfiguriert. (128 Bit = 4 Blöcke Transponder Talks First in Manchester Modulierung, 2kBit). Ob dieses Format unterstützt wird hängt vom Gerät ab.

Programmierschnittstelle (SDK) der TS-RW Geräte

int TSLF_Write_FSK26Bit (int PortHandle, int TTyp,
DWORD FacilityCode, DWORD CustomerNr, int Lock**)**

PortHandle	Zugriffsnummer	
TTyp	Transpondertyp aus Tabelle	
FacilityCode	Facility Code 0 - 255	
CustomerNr	Kundennummer 0 – 65535	
Lock	0 = kein Schutz	1 = Datenblöcke sperren
	2 = Konfiguration sperren	3 = Konfiguration und Daten sperren.
Rückgabewert:	-1 Fehler, 0 OK	

Mit dieser Funktion werden Transponder im FSK26Bit Format geschrieben und formatiert.
Diese Funktion wird nur bei TS-RW38Plus und TS-RW38AC unterstützt.
Unterstützte Transpondertypen sind Q5, ATA5577 und SIC7888

int TSLF_Write_FSK34Bit (int PortHandle, int TTyp,
DWORD FacilityCode, DWORD CustomerNr, int Lock**)**

PortHandle	Zugriffsnummer	
TTyp	Transpondertyp aus Tabelle	
FacilityCode	Facility Code 0 - 65535	
CustomerNr	Kundennummer 0 - 65535	
Lock	0 = kein Schutz	1 = Datenblöcke sperren
	2 = Konfiguration sperren	3 = Konfiguration und Daten sperren.
Rückgabewert:	-1 Fehler, 0 OK	

Mit dieser Funktion werden Transponder im FSK34Bit Format geschrieben und formatiert.
Diese Funktion wird nur bei TS-RW38Plus und TS-RW38AC unterstützt.
Unterstützte Transpondertypen sind Q5, ATA5577 und SIC7888

int TSLF_Write_NC34Bit (int PortHandle, int TTyp,
DWORD FacilityCode, DWORD CustomerNr, int Lock**)**

PortHandle	Zugriffsnummer	
TTyp	Transpondertyp aus Tabelle	
FacilityCode	Facility Code 0 - 65535	
CustomerNr	Kundennummer 0 - 65535	
Lock	0 = kein Schutz	1 = Datenblöcke sperren
	2 = Konfiguration sperren	3 = Konfiguration und Daten sperren.
Rückgabewert:	-1 Fehler, 0 OK	

Mit dieser Funktion werden Transponder im NC34Bit Format geschrieben und formatiert.
Der Unterschied zwischen FSK34Bit und NC34Bit besteht in unterschiedlicher Prüfsummenberechnung.
Diese Funktion wird nur bei TS-RW38Plus und TS-RW38AC unterstützt.
Unterstützte Transpondertypen sind Q5, ATA5577 und SIC7888

Programmierschnittstelle (SDK) der TS-RW Geräte

int TSLF_Write_FSK35Bit (int PortHandle, int TTyp,
DWORD FacilityCode, DWORD CustomerNr, int Lock**)**

PortHandle	Zugriffsnummer	
TTyp	Transpondertyp aus Tabelle	
FacilityCode	Facility Code 0 - 4095	
CustomerNr	Kundennummer 0 - 1048575	
Lock	0 = kein Schutz	1 = Datenblöcke sperren
	2 = Konfiguration sperren	3 = Konfiguration und Daten sperren.
Rückgabewert:	-1 Fehler, 0 OK	

Mit dieser Funktion werden Transponder im FSK35Bit Format geschrieben und formatiert.
Diese Funktion wird nur bei TS-RW38Plus und TS-RW38AC unterstützt.
Unterstützte Transpondertypen sind Q5, ATA5577 und SIC7888

int TSLF_Write_FSK36Bit (int PortHandle, int TTyp, DWORD FacilityCode,
DWORD CustomerNr, DWORD FixedField, int Lock**)**

PortHandle	Zugriffsnummer	
TTyp	Transpondertyp aus Tabelle	
FacilityCode	Facility Code 0 - 255	
CustomerNr	Kundennummer 0 – 65535	
FixedField	FixedField 0 - 1023	
Lock	0 = kein Schutz	1 = Datenblöcke sperren
	2 = Konfiguration sperren	3 = Konfiguration und Daten sperren.
Rückgabewert:	-1 Fehler, 0 OK	

Mit dieser Funktion werden Transponder im FSK36Bit Format geschrieben und formatiert.
Diese Funktion wird nur bei TS-RW38Plus und TS-RW38AC unterstützt.
Unterstützte Transpondertypen sind Q5, ATA5577 und SIC7888

Programmierschnittstelle (SDK) der TS-RW Geräte

int TSLF_Write_FSK37Bit (int PortHandle, int TTyp,
DWORD FacilityCode, DWORD CustomerNr, int Lock**)**

PortHandle	Zugriffsnummer	
TTyp	Transpondertyp aus Tabelle	
FacilityCode	Facility Code 0 - 65535	
CustomerNr	Kundennummer 0 - 524287	
Lock	0 = kein Schutz	1 = Datenblöcke sperren
	2 = Konfiguration sperren	3 = Konfiguration und Daten sperren.
Rückgabewert:	-1 Fehler, 0 OK	

Mit dieser Funktion werden Transponder im FSK37Bit Format geschrieben und formatiert.
Diese Funktion wird nur bei TS-RW38Plus und TS-RW38AC unterstützt.
Unterstützte Transpondertypen sind Q5, ATA5577 und SIC7888

int TSLF_Write_FSK37BitHuge (int PortHandle, int TTyp,
BYTE * pCode, int CodeLen, int Lock**)**

PortHandle	Zugriffsnummer	
TTyp	Transpondertyp aus Tabelle	
pCode	Zeiger auf 37 Bit Huge Code 0 - 34359738367	
CodeLen	Länge von pCode in Byte (5)	
Lock	0 = kein Schutz	1 = Datenblöcke sperren
	2 = Konfiguration sperren	3 = Konfiguration und Daten sperren.
Rückgabewert:	-1 Fehler, 0 OK	

Mit dieser Funktion werden Transponder im FSK37Bit Huge Format geschrieben und formatiert.
Diese Funktion wird nur bei TS-RW38Plus und TS-RW38AC unterstützt.
Unterstützte Transpondertypen sind Q5, ATA5577 und SIC7888

int TSLF_Write_IdentiCard37Bit (int PortHandle, int TTyp, DWORD FacilityCode,
DWORD CustomerNr, DWORD FixedField, int Lock**)**

PortHandle	Zugriffsnummer	
TTyp	Transpondertyp aus Tabelle	
FacilityCode	Facility Code 0 - 8191	
CustomerNr	Kundennummer 0 – 131071	
FixedField	FixedField 0 - 31	
Lock	0 = kein Schutz	1 = Datenblöcke sperren
	2 = Konfiguration sperren	3 = Konfiguration und Daten sperren.
Rückgabewert:	-1 Fehler, 0 OK	

Mit dieser Funktion werden Transponder im IdentiCard 37Bit Format geschrieben und formatiert.
Diese Funktion wird nur bei TS-RW38Plus und TS-RW38AC unterstützt.
Unterstützte Transpondertypen sind Q5, ATA5577 und SIC7888

Programmierschnittstelle (SDK) der TS-RW Geräte

int TSLF_Write_AWID26Bit (int PortHandle, int TTyp,
DWORD FacilityCode, DWORD CustomerNr, int Lock**)**

PortHandle	Zugriffsnummer	
TTyp	Transpondertyp aus Tabelle	
FacilityCode	Facility Code 0 - 255	
CustomerNr	Kundennummer 0 - 65535	
Lock	0 = kein Schutz	1 = Konfiguration und Daten sperren.
Rückgabewert:	-1 Fehler, 0 OK	

Mit dieser Funktion werden Transponder im AWID 26 Bit Format geschrieben und formatiert.
Diese Funktion wird nur bei TS-RW38Plus und TS-RW38AC unterstützt.
Unterstützte Transpondertypen sind Q5, ATA5577 und SIC7888

int TSLF_Write_AWID40Bit (int PortHandle, int TTyp,
DWORD FacilityCode, DWORD CustomerNr, int Lock**)**

PortHandle	Zugriffsnummer	
TTyp	Transpondertyp aus Tabelle	
FacilityCode	Facility Code 0 - 1023	
CustomerNr	Kundennummer 0 - 268435455	
Lock	0 = kein Schutz	1 = Konfiguration und Daten sperren.
Rückgabewert:	-1 Fehler, 0 OK	

Mit dieser Funktion werden Transponder im AWID 40 Bit Format geschrieben und formatiert.
Diese Funktion wird nur bei TS-RW38Plus und TS-RW38AC unterstützt.
Unterstützte Transpondertypen sind Q5, ATA5577 und SIC7888

4.7. Formatiertes Lesen

Mit den Befehlen dieses Kapitels können Transponder im TTF Modus gelesen werden.
Da die Transponderdaten von allen Transpondern, die TTF beherrschen gleich sind wird hier kein Transpondertyp angegeben. Da der Transponder nicht selektiert sein darf wird das Antennenfeld innerhalb dieser Funktion Aus- und wieder Eingeschaltet.

int TSLF_Read_TTFData (int PortHandle, int AnswerMode, BYTE * pBuffer, int BufLen**)**

PortHandle	Zugriffsnummer	
AnswerMode	Antwortmodus	
	0= Manchester	2,0 kBit Data Rate
	1= Biphase	4,0 kBit Data Rate
	2= FSK2a	2,5 kBit Data Rate
pBuffer	Zeiger auf die Transponderdaten	
BufLen	Länge der erwarteten Transponderdaten	
Rückgabewert	-1 Fehler, >0 Länge der gelesenen Daten	

Der Dateninhalt ist nicht ausgerichtet, da der Datenstrom bei TTF kontinuierlich kommt. Vor der Bewertung der Daten muss der Datenstrom je nach verwendetem Datenformat noch entsprechend verschoben werden.

int TSLF_Read_Unique(int PortHandle, BYTE * pBuffer, int BufLen**)**

PortHandle	Zugriffsnummer
pBuffer	Zeiger auf die Transponderdaten
BufLen	Länge der Transponderdaten (muss fest auf 5 Byte sein)
Rückgabewert	-1 Fehler, >0 Länge der gelesenen Daten

Dieses Kommando liest die Unique Daten aus. Der Transponder muss hierzu als Unique formatiert und beschrieben sein.

int TSLF_Read_UniqueXL(int PortHandle, BYTE * pBuffer, int BufLen**)**

PortHandle	Zugriffsnummer
pBuffer	Zeiger auf die Transponderdaten
BufLen	Länge der Transponderdaten (muss fest auf 11 Byte sein)
Rückgabewert	-1 Fehler, >0 Länge der gelesenen Daten

Dieses Kommando liest die Unique XL Daten aus. Der Transponder muss hierzu als Unique XL formatiert und beschrieben sein.

Programmierschnittstelle (SDK) der TS-RW Geräte

int TSLF_Read_FastUnique(int PortHandle, BYTE * pBuffer, int BufLen**)**

PortHandle	Zugriffsnummer
pBuffer	Zeiger auf die Transponderdaten
BufLen	Länge der Transponderdaten (muss fest auf 5 Byte sein)
Rückgabewert	-1 Fehler, >0 Länge der gelesenen Daten

Dieses Kommando liest die Fast Unique Daten aus. Der Transponder muss hierzu als Fast Unique formatiert und beschrieben sein.
Diese Funktion wird nur bei TS-RW38Plus und TS-RW38AC unterstützt.

int TSLF_Read_GiS16Bit(int PortHandle, BYTE * pBuffer, int BufLen**)**

PortHandle	Zugriffsnummer
pBuffer	Zeiger auf die Transponderdaten
BufLen	Länge der Transponderdaten (muss fest auf 2 Byte sein)
Rückgabewert	-1 Fehler, >0 Länge der gelesenen Daten

Dieses Kommando liest die „GiS 16 Bit Format“ Daten aus. Der Transponder muss hierzu als „GiS 16 Bit Format“ formatiert und beschrieben sein.

int TSLF_Read_GiS12Bit(int PortHandle, BYTE * pBuffer, int BufLen**)**

PortHandle	Zugriffsnummer
pBuffer	Zeiger auf die Transponderdaten
BufLen	Länge der Transponderdaten (muss fest auf 2 Byte sein)
Rückgabewert	-1 Fehler, >0 Länge der gelesenen Daten

Dieses Kommando liest die „GiS 12 Bit Format“ Daten aus. Der Transponder muss hierzu als „GiS 12 Bit Format“ formatiert und beschrieben sein.

int TSLF_Read_GiS72Bit(int PortHandle, BYTE * pBuffer, int BufLen**)**

PortHandle	Zugriffsnummer
pBuffer	Zeiger auf die Transponderdaten
BufLen	Länge der Transponderdaten (muss fest auf 9 Byte sein)
Rückgabewert	-1 Fehler, >0 Länge der gelesenen Daten

Dieses Kommando liest die „GiS 72 Bit Format“ Daten aus. Der Transponder muss hierzu als „GiS 72 Bit Format“ formatiert und beschrieben sein.

Programmierschnittstelle (SDK) der TS-RW Geräte

int TSLF_Read_FSK26Bit (int PortHandle, DWORD *pFacilityCode, DWORD *pCustomerNr**)**
PortHandle Zugriffsnummer
pFacilityCode Zeiger auf Facility Code 0 - 255
pCustomerNr Zeiger auf Kundennummer 0 - 65535
Rückgabewert: -1 Fehler, 0 OK

Mit dieser Funktion werden Transponder im FSK26Bit Format gelesen.
Diese Funktion wird nur bei TS-RW38Plus und TS-RW38AC unterstützt.

int TSLF_Read_FSK34Bit (int PortHandle, DWORD *pFacilityCode, DWORD *pCustomerNr**)**
PortHandle Zugriffsnummer
pFacilityCode Zeiger auf Facility Code 0 - 65535
pCustomerNr Zeiger auf Kundennummer 0 - 65535
Rückgabewert: -1 Fehler, 0 OK

Mit dieser Funktion werden Transponder im FSK34Bit Format gelesen.
Diese Funktion wird nur bei TS-RW38Plus und TS-RW38AC unterstützt.
Wird eine fehlerhafte Prüfsumme erkannt, so ist der Rückgabewert -1, TSLF_GetLastError() liefert TSLF_CHECKSUMMENFEHLER (14) und der Facility Code und die Customer Nummer werden trotzdem gesetzt.

int TSLF_Read_NC34Bit (int PortHandle, DWORD *pFacilityCode, DWORD *pCustomerNr**)**
PortHandle Zugriffsnummer
pFacilityCode Zeiger auf Facility Code 0 - 65535
pCustomerNr Zeiger auf Kundennummer 0 - 65535
Rückgabewert: -1 Fehler, 0 OK

Mit dieser Funktion werden Transponder im FSK34Bit Format gelesen.
Der Unterschied zwischen FSK34Bit und NC34Bit besteht in unterschiedlicher Prüfsummenberechnung.
Diese Funktion wird nur bei TS-RW38Plus und TS-RW38AC unterstützt.
Wird eine fehlerhafte Prüfsumme erkannt, so ist der Rückgabewert -1, TSLF_GetLastError() liefert TSLF_CHECKSUMMENFEHLER (14) und der Facility Code und die Customer Nummer werden trotzdem gesetzt.

int TSLF_Read_FSK35Bit (int PortHandle, DWORD *pFacilityCode, DWORD *pCustomerNr**)**
PortHandle Zugriffsnummer
pFacilityCode Zeiger auf Facility Code 0 - 1023
pCustomerNr Zeiger auf Kundennummer 0 - 1048575
Rückgabewert: -1 Fehler, 0 OK

Mit dieser Funktion werden Transponder im FSK35Bit Format gelesen.
Diese Funktion wird nur bei TS-RW38Plus und TS-RW38AC unterstützt.

Programmierschnittstelle (SDK) der TS-RW Geräte

int TSLF_Read_FSK36Bit (int PortHandle, DWORD *pFacilityCode,
DWORD *pCustomerNr, DWORD * pFixedField**)**

PortHandle	Zugriffsnummer
pFacilityCode	Zeiger auf Facility Code 0 - 255
pCustomerNr	Zeiger auf Kundennummer 0 – 65535
pFixedField	Zeiger auf FixedField 0 - 1023
Rückgabewert:	-1 Fehler, 0 OK

Mit dieser Funktion werden Transponder im FSK36Bit Format gelesen.
Diese Funktion wird nur bei TS-RW38Plus und TS-RW38AC unterstützt.

int TSLF_Read_FSK37Bit (int PortHandle, DWORD *pFacilityCode, DWORD *pCustomerNr**)**

PortHandle	Zugriffsnummer
pFacilityCode	Zeiger auf Facility Code 0 - 65535
pCustomerNr	Zeiger auf Kundennummer 0 - 524287
Rückgabewert:	-1 Fehler, 0 OK

Mit dieser Funktion werden Transponder im FSK37Bit Format gelesen.
Diese Funktion wird nur bei TS-RW38Plus und TS-RW38AC unterstützt.

int TSLF_Read_FSK37BitHuge (int PortHandle, BYTE *pCode, int CodeLen**)**

PortHandle	Zugriffsnummer
pCode	Zeiger auf Code 0 - 34359738367
CodeLen	Länge von pCode in Byte (5)
Rückgabewert:	-1 Fehler, >0 Länge der gelesenen Daten

Mit dieser Funktion werden Transponder im FSK37Bit Huge Format gelesen.
Diese Funktion wird nur bei TS-RW38Plus und TS-RW38AC unterstützt.

int TSLF_Read_IdentiCard37Bit (int PortHandle, DWORD *pFacilityCode,
DWORD *pCustomerNr, DWORD * pFixedField**)**

PortHandle	Zugriffsnummer
pFacilityCode	Zeiger auf Facility Code 0 - 8191
pCustomerNr	Zeiger auf Kundennummer 0 – 131071
pFixedField	Zeiger auf FixedField 0 - 31
Rückgabewert:	-1 Fehler, 0 OK

Mit dieser Funktion werden Transponder im IdentiCard 37Bit Format gelesen.
Diese Funktion wird nur bei TS-RW38Plus und TS-RW38AC unterstützt.

Programmierschnittstelle (SDK) der TS-RW Geräte

int TSLF_Read_AWID26Bit (int PortHandle,
DWORD *pFacilityCode, DWORD *pCustomerNr**)**

PortHandle	Zugriffsnummer
pFacilityCode	Zeiger auf Facility Code 0 - 255
pCustomerNr	Zeiger auf Kundennummer 0 - 65535
Rückgabewert:	-1 Fehler, 0 OK

Mit dieser Funktion werden Transponder im AWID 26 Bit Format gelesen.
Diese Funktion wird nur bei TS-RW38Plus und TS-RW38AC unterstützt.

int TSLF_Read_AWID40Bit (int PortHandle,
DWORD *pFacilityCode, DWORD *pCustomerNr**)**

PortHandle	Zugriffsnummer
pFacilityCode	Zeiger auf Facility Code 0 - 1023
pCustomerNr	Zeiger auf Kundennummer 0 - 268435455
Rückgabewert:	-1 Fehler, 0 OK

Mit dieser Funktion werden Transponder im AWID 40 Bit Format gelesen.
Diese Funktion wird nur bei TS-RW38Plus und TS-RW38AC unterstützt.

Programmierschnittstelle (SDK) der TS-RW Geräte

4.8. Zugriff auf Transponder mit Temperaturmessung

Die folgenden Funktionen sind nur für Transponder mit eingebauter Temperaturmessung verfügbar.
Es werden ausschließlich Transponder vom Typ EL9265 unterstützt.
Nur TS-RW38+ und TS-RW38AC jeweils mit Version 1.50 und höher unterstützen diese Funktionen.

int TSLF_GetTempSensor(int PortHandle, int TTyp, int *pSensorData);

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
pSensorData	Zeiger auf Sensordaten die vom Transponder gelesen werden.
Rückgabewert:	-1 Fehler, 0: OK

Hiermit wird eine Temperaturmessung im Transponder ausgelöst (GET TEMP) und die Sensordaten geliefert. Aus den Sensordaten kann dann der Temperaturwert mithilfe der Kalibrierungsdaten ermittelt werden. Hierzu bitte das Datenblatt des EN9265 Transponders beachten.
Der Transponder wird vor der Lesung automatisch selektiert.

int TSLF_StoreTempSensor(int PortHandle, int TTyp, int TimingInfo, int Page,
int *pSensorData);

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
TimingInfo	Zusatzinformation die mit den Sensordaten im Transponder gespeichert wird
Page	Nummer des Blockes in dem die Sensordaten gespeichert werden. (Blöcke 8 – 57 sind erlaubt)
pSensorData	Zeiger auf Sensordaten die vom Transponder gelesen werden.
Rückgabewert:	-1 Fehler, 0: OK

Hiermit wird eine Temperaturmessung im Transponder ausgelöst (EN INSEN) und die Sensordaten geliefert. Die Sensordaten werden in der angegebenen Page im Transponder zusammen mit der TimingInfo Information gespeichert. Aus den Sensordaten kann dann der Temperaturwert mithilfe der Kalibrierungsdaten ermittelt werden. Hierzu bitte das Datenblatt des EN9265 Transponders beachten.
Der Transponder wird vor der Lesung automatisch selektiert.

Programmierschnittstelle (SDK) der TS-RW Geräte

int TSLF_GetTempGrad(int PortHandle, int TTyp, int * pTemp);

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
pTemp	Zeiger auf Temperaturwert in $^{1}/_{65536}$ °C
Rückgabewert:	-1 Fehler, 0 OK

Hiermit wird eine Temperaturmessung im Transponder ausgelöst (GET TEMP). Aus den ermittelten Sensordaten wird dann der Temperaturwert mithilfe der Kalibrierungsdaten gemäß des Datenblattes des EN9265 Transponders ermittelt. Die zurückgegebene Temperatur muss durch 65536 geteilt werden um einen Wert in °C zu erhalten.
Der Transponder wird vor der Lesung automatisch selektiert.

int TSLF_ConvertTemp(int PortHandle, int TTyp, int SensorData);

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
SensorData	Sensordaten die aus einer früheren Temperaturmessung stammen
Rückgabewert:	-1 Fehler, Temperaturwert in °C*65536

Es werden die Sensordaten wie sie entweder mittels TSLF_GetTempSensor ermittelt wurden, oder mittels TSLF_Read_FDXB in pExtension erhalten wurden (siehe TSLF_ActivateTempSensor) übergeben, mittels der Kalibrierungsdaten des Chips wird die Temperatur ermittelt und zurückgegeben.
Hierbei muss der Transponder im Feld sein, damit die Kalibrierungsdaten gelesen werden können.
Der Rückgabewert muss durch 65536 geteilt werden um einen Wert in °C zu erhalten.
Der Transponder wird vor der Lesung automatisch selektiert.

int TSLF_ActivateTempSensor(int PortHandle, int TTyp, int Page);

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
Page	Nummer des Blockes der verwendet wird (8-57), 0: deaktivieren
Rückgabewert:	-1 Fehler, 0 OK

Beim EL9265 Chip gibt es die Möglichkeit einen gespeicherten Temperaturwert mit den Animal Code Daten zurückzuliefern (TTF_MODE = 01b). Mit dieser Funktion wird dieser Modus aktiviert, oder wenn eine Page 0 angegeben wird, deaktiviert. Hierzu bitte das Datenblatt des EN9265 Transponders beachten.
Der Transponder wird vor der Lesung automatisch selektiert.

Programmierschnittstelle (SDK) der TS-RW Geräte

4.9. Zugriff auf Transponder Typ EM4582

Die folgenden Funktionen sind nur für Transponder des Typs EM4582 verfügbar.
Nur TS-RW380+ und TS-RW380AC jeweils mit Version 1.04 und höher unterstützen diese Funktionen.
Zur Verwendung der Funktionen wird Kenntnis des EM4582 Datenblattes vorausgesetzt.
Der Zugriff zum Lesen und Schreiben der Blöcke des EM4582 erfolgt über die üblichen TSLF_Read und TSLF_Write Befehle.

int **TSLF_EM4582_GetIncrement**(int PortHandle, DWORD *pData, DWORD *pVal)

PortHandle	Zugriffsnummer
pData	Benutzerdaten zur Nummer
pVal	zuletzt gespeicherte Nummer
Rückgabewert:	-1 Fehler, 6 Daten empfangen

Die mit TSLF_EM4582_Increment gespeicherte Nummer wird hiermit ausgelesen. Es wird automatisch die letzte geschriebene Nummer gesucht.

int **TSLF_EM4582_Increment**(int PortHandle,
BYTE *pKey, int KeyLen,
DWORD Data, DWORD Val)

PortHandle	Zugriffsnummer
pKey	Zeiger auf Schlüssel
KeyLen	Länge des Schlüssels
Data	Benutzerdaten
Val	Nummer
Rückgabewert:	-1 Fehler, 0 erfolgreich geschrieben

Es wird mit dem WRITE INCREMENT Kommando eine Nummer mit Benutzerdaten im Transponder gespeichert. Dies geht nur, wenn die Nummer größer als die zuletzt gespeicherte Nummer ist. In pKey wird der im Transponder hinterlegte Schlüssel SKI mit LSB first übergeben. In KeyLen ist die Länge des Schlüssels gespeichert.
Ist der Schlüssel im TS-RW380 gespeichert, so kann der Schlüssel hier weggelassen werden. Wird pKey = 0 oder die KeyLen = 0 übergeben, so wird der im TS-RW380 gespeicherte Schlüssel verwendet.
Soll der Schlüssel verwendet werden, so muss er eine Länge von 128 Bit (16 Byte) haben.

Programmierschnittstelle (SDK) der TS-RW Geräte

int **TSLF_EM4582_SendAccess**(int PortHandle,
BYTE *pID, int IDLen,
BYTE *pPwd, int PWLen)

PortHandle Zugriffsnummer
pID UID des Transponders, LSB first
IDLen Länge der UID des Transponders (immer 4 Byte)
pPwd Passwort, LSB first
PWLen Länge des Passwortes (immer 4 Byte)
Rückgabewert: -1 Fehler, 0: Erfolg

Hiermit wird das SEND ACCESS Kommando ausgeführt. Die UID des Transponders kann mit TSLF_UID_Request bestimmt werden. Das SEND ACCESS Kommando erlaubt den Zugriff auf Register und Speicherbereiche die mit LB0 geschützt sind.

int **TSLF_EM4582_UnlockKey**(int PortHandle,
BYTE *pID, int IDLen,
BYTE *pPwd, int PWLen)

PortHandle	Zugriffsnummer
pID	UID des Transponders, LSB first
IDLen	Länge der UID des Transponders (immer 4 Byte)
pPwd	Passwort, LSB first
PWLen	Länge des Passwortes (immer 4 Byte)
Rückgabewert:	-1 Fehler, 0: Erfolg

Hiermit wird das UNLOCK KEY Kommando ausgeführt. Die UID des Transponders kann mit TSLF_UID_Request bestimmt werden. Der Befehl UNLOCK KEY ermöglicht das Entsperren eines geheimen Schlüssels zur Änderung, indem der bisherige geheime Schlüssel und die Chip-UID bereitgestellt werden.

int **TSLF_EM4582_GetRN**(int PortHandle, BYTE *pRN)

PortHandle	Zugriffsnummer
pRN	Zeiger auf Random Number, muss Platz für 16 Byte (128 Bit) haben.
Rückgabewert:	-1 Fehler, >0 Länge der Zufallszahl

Es wird eine Zufallszahl mittels des GET RN Kommandos des Transponders, die für die Authentifizierung verwendet werden kann, generiert. Die Zufallszahl wird im Speicherbereich pRN abgelegt. Die Länge der Zufallszahl ist abhängig von Einstellungen im Transponder.

Programmierschnittstelle (SDK) der TS-RW Geräte

int **TSLF_EM4582_SingleAuth**(int PortHandle,
BYTE *pRN, int RNLen,
BYTE *pTokenR, int TokenRLen,
BYTE *pTokenT, int MaxTokenTLen)

PortHandle	Zugriffsnummer
pRN	Zufallszahl
RNLen	Länge der Zufallszahl
pTokenR	TOKENR→T
TokenRLen	Länge von TOKENR→T
pTokenT	TOKENT→R
MaxTokenTLen	Maximale Länge von TOKENT→R
Rückgabewert:	-1 Fehler, > 0 Länge des TOKENT→R

SingleAuth erlaubt die Authentifizierung des Transponders mittels des SINGLE AUTHENTICATION Kommandos. Die Längen der Zufallszahl und der Token ist von der Einstellung des Transponders abhängig. Eine genauere Beschreibung ist dem Transponder Datenblatt zu entnehmen.

int TSLF_EM4582_MutualAuth(int PortHandle,
BYTE *pRN, int RNLen,
BYTE *pTokenR, int TokenRLen,
BYTE *pTokenT, int MaxTokenTLen)

PortHandle	Zugriffsnummer
pRN	Zufallszahl
RNLen	Länge der Zufallszahl
pTokenR	TOKENR→T
TokenRLen	Länge von TOKENR→T
pTokenT	TOKENT→R
MaxTokenTLen	Maximale Länge von TOKENT→R
Rückgabewert:	-1 Fehler, > 0 Länge des TOKENT→R

MutualAuth erlaubt die Authentifizierung des Transponders mittels des MUTUAL AUTHENTICATION Kommandos. Die Längen der Zufallszahl und der Token ist von der Einstellung des Transponders abhängig. Eine genauere Beschreibung ist dem Transponder Datenblatt zu entnehmen.

Programmierschnittstelle (SDK) der TS-RW Geräte

int **TSLF_EM4582_IsoMutualAuth**(int PortHandle,
BYTE *pTokenR, int TokenRLen,
BYTE *pTokenT, int MaxTokenTLen)

PortHandle	Zugriffsnummer
pTokenR	TOKENR→T
TokenRLen	Länge von TOKENR→T
pTokenT	TOKENT→R
MaxTokenTLen	Maximale Länge von TOKENT→R
Rückgabewert:	-1 Fehler, > 0 Länge des TOKENT→R

MutualAuthIso erlaubt die Authentifizierung des Transponders mittels des ISO MUTUAL AUTHENTICATION Kommandos. Die Längen der Token ist von der Einstellung des Transponders abhängig. Eine genauere Beschreibung ist dem Transponder Datenblatt zu entnehmen.

int **TSLF_EM4582_SingleAuthDirect**(int PortHandle, int RnTokenTyp,
BYTE *pKey, int KeyLen)

PortHandle	Zugriffsnummer
RNTokenTyp	Einstellung für Länge der Zufallszahl und Token Bedeutung siehe Bit 0 – 7 des Crypt Configuration Registers im Datenblatt
pKey	Zeiger auf Schlüssel
KeyLen	Länge des Schlüssels
Rückgabewert:	-1 Fehler, 0 Erfolg

Es wird der komplette SINGLE AUTHENTICATION Vorgang durchgeführt.
In pKey wird der im Transponder hinterlegte Schlüssel SKA mit LSB first übergeben.
In KeyLen ist die Länge des Schlüssels gespeichert.
Ist der Schlüssel im TS-RW380 gespeichert, so kann der Schlüssel hier weggelassen werden. Wird pKey = 0 oder die KeyLen = 0 übergeben, so wird der im TS-RW380 gespeicherte Schlüssel verwendet.
Soll der Schlüssel verwendet werden, so muss er eine Länge von 128 Bit (16 Byte) haben.
Diese Funktion vereinfacht die Anwendung der SINGLE AUTHENTICATION, da in der aufrufenden Software keine AES Verschlüsselung gemacht werden muss. Allerdings wird dadurch ggf. der Key im Klartext übertragen, was eine mögliche Sicherheitslücke darstellt.
Durch Speichern des Keys im TS-RW380 kann dies umgangen werden.

Programmierschnittstelle (SDK) der TS-RW Geräte

int **TSLF_EM4582_MutualAuthDirect**(int PortHandle, int RnTokenTyp,
BYTE *pKey, int KeyLen)

PortHandle	Zugriffsnummer
RNTokenTyp	Einstellung für Länge der Zufallszahl und Token
	Bedeutung siehe Bit 0 – 7 des Crypt Configuration Registers im Datenblatt
pKey	Zeiger auf Schlüssel
KeyLen	Länge des Schlüssels
Rückgabewert:	-1 Fehler, 0 Erfolg

Es wird der komplette MUTUAL AUTHENTICATION Vorgang durchgeführt.
In pKey wird der im Transponder hinterlegte Schlüssel SKA mit LSB first übergeben.
In KeyLen ist die Länge des Schlüssels gespeichert.
Ist der Schlüssel im TS-RW380 gespeichert, so kann der Schlüssel hier weggelassen werden. Wird pKey = 0 oder die KeyLen = 0 übergeben, so wird der im TS-RW380 gespeicherte Schlüssel verwendet.
Soll der Schlüssel verwendet werden, so muss er eine Länge von 128 Bit (16 Byte) haben.
Diese Funktion vereinfacht die Anwendung der MUTUAL AUTHENTICATION, da in der aufrufenden Software keine AES Verschlüsselung gemacht werden muss. Allerdings wird dadurch ggf. der Key im Klartext übertragen, was eine mögliche Sicherheitslücke darstellt.
Durch Speichern des Keys im TS-RW380 kann dies umgangen werden.

int **TSLF_EM4582_IsoMutualAuthDirect**(int PortHandle, int RnTokenTyp,
BYTE *pKey, int KeyLen)

PortHandle	Zugriffsnummer
RNTokenTyp	Einstellung für Länge der Zufallszahl und Token
	Bedeutung siehe Bit 0 – 7 des Crypt Configuration Registers im Datenblatt
pKey	Zeiger auf Schlüssel
KeyLen	Länge des Schlüssels
Rückgabewert:	-1 Fehler, 0 Erfolg

Es wird der komplette ISO MUTUAL AUTHENTICATION Vorgang durchgeführt.
In pKey wird der im Transponder hinterlegte Schlüssel SKA mit LSB first übergeben.
In KeyLen ist die Länge des Schlüssels gespeichert.
Ist der Schlüssel im TS-RW380 gespeichert, so kann der Schlüssel hier weggelassen werden. Wird pKey = 0 oder die KeyLen = 0 übergeben, so wird der im TS-RW380 gespeicherte Schlüssel verwendet.
Soll der Schlüssel verwendet werden, so muss er eine Länge von 128 Bit (16 Byte) haben.
Diese Funktion vereinfacht die Anwendung der ISO MUTUAL AUTHENTICATION, da in der aufrufenden Software keine AES Verschlüsselung gemacht werden muss. Allerdings wird dadurch ggf. der Key im Klartext übertragen, was eine mögliche Sicherheitslücke darstellt.
Durch Speichern des Keys im TS-RW380 kann dies umgangen werden.

Programmierschnittstelle (SDK) der TS-RW Geräte

int **TSLF_EM4582_SetSKA**(int PortHandle,
BYTE *pKey, int KeyLen)

PortHandle	Zugriffsnummer
pKey	Zeiger auf Schlüssel
KeyLen	Länge des Schlüssels (immer 16 Byte)
Rückgabewert:	-1 Fehler, 0 Erfolg

Speichern des Schlüsselwertes für SKA im TS-RW380. Damit kann bei den **…AuthDirect** Funktionen der Schlüssel weggelassen werden.

int **TSLF_EM4582_SetSKI**(int PortHandle,
BYTE *pKey, int KeyLen)

PortHandle	Zugriffsnummer
pKey	Zeiger auf Schlüssel
KeyLen	Länge des Schlüssels
Rückgabewert:	-1 Fehler, 0 Erfolg

Speichern des Schlüsselwertes für SKI im TS-RW380. Damit kann bei **TSLF_EM4582_Increment** der Schlüssel weggelassen werden.

int **TSLF_EM4582_SetET**(int PortHandle,
BYTE *pKey, int KeyLen)

PortHandle	Zugriffsnummer
pKey	Zeiger auf Schlüssel
KeyLen	Länge des Schlüssels (immer 12 Byte)
Rückgabewert:	-1 Fehler, 0 Erfolg

Speichern des Wertes für die Expansion Number Transponder ET im TS-RW380.
Diese wird bei den **…AuthDirect** Funktionen zur Bestimmung der Tokens benötigt.

int **TSLF_EM4582_SetER**(int PortHandle,
BYTE *pKey, int KeyLen)

PortHandle	Zugriffsnummer
pKey	Zeiger auf Schlüssel
KeyLen	Länge des Schlüssels (immer 4 Byte)
Rückgabewert:	-1 Fehler, 0 Erfolg

Speichern des Wertes für die Expansion Number Reader ER im TS-RW380.
Diese wird bei den **…AuthDirect** Funktionen zur Bestimmung der Tokens benötigt.

5. Animal (FDX-A, FDX-B, HDX) oder Waste (EN14803) Transponder

Diese Funktionen sind nur mit Geräten die Animal Code unterstützen möglich.

5.1. Animal Transponder (FDX-A, FDX-B, HDX) schreiben und lesen

FDX-A Format Kurzbeschreibung
Das FDX-A Format besteht aus einer 10 stelligen Zahl, die im OktHex Verfahren eingetragen wird Das heißt jedes der 5 Byte des Wertes kann Bereiche von 00 – $7F^{Hex}$ annehmen.
Häufig wird jedoch nur der Dezimalbereich verwendet.

FDX-B Format Kurzbeschreibung
Ein Animal (FDX-B) Tag hat 128 Bit Daten. Der Datenaufbau ist in ISO11785 beschrieben.
Hierin ist der Identifikation Code enthalten. Der Aufbau des Identifikation Codes wiederum ist in ISO11784 beschrieben.
Als Erweiterung zu ISO11784 wurde die Verwendung des Reserved Bereiches für die EU mit folgender Bedeutung festgelegt:

Reserved Bit 0 - 2	Retagging counter
Reserved Bit 3 - 7	User information field
Reserved Bit 8 - 13	reserved field

Der in ISO 11785 beschriebene Extension Bereich hat eine Länge von 24 Bit und wird als Extension angegeben.

HDX Format Kurzbeschreibung
Ein Animal (HDX) Transponder hat 80 Bit Daten. Der Datenaufbau ist in ISO11785 beschrieben.
Hierin ist der Identifikation Code enthalten. Der Aufbau des Identifikation Codes wiederum ist in ISO11784 beschrieben.
Als Erweiterung zu ISO11784 wurde die Verwendung des Reserved Bereiches für die EU mit folgender Bedeutung festgelegt:

Reserved Bit 0 - 2	Retagging counter
Reserved Bit 3 - 7	User information field
Reserved Bit 8 - 13	reserved field

Der in ISO 11785 beschriebene Extension Bereich hat eine Länge von 24 Bit und wird als Extension angegeben.

Programmierschnittstelle (SDK) der TS-RW Geräte

int TSLF_Read_FDXA(int PortHandle, BYTE * pIdNumber, int pIdLen)

PortHandle	Zugriffsnummer
pIdNumber	Identifikationsnummer (5 Byte Länge gemäß FDX-A Spezifikation)
pIdLen	Länge des Datenbereiches für Identifikationsnummer (muss mind. 5 Byte sein)
Rückgabewert	-1 Fehler, > 0 Länger der Daten in pIdNumber

Mit dieser Funktion kann man Animal FDX-A Transponder einlesen.
Achtung: Diese Funktion wird nur von TS-RW38 AC und TS-RW68 AC unterstützt!!!

int TSLF_Read_FDXB(int PortHandle, int * pCountryCode,
BYTE * pBuffer, int BufLen, int *pAnimalFlag,
int *pDatenblockFlag, int *pReserved, int *pExtension**)**

PortHandle	Zugriffsnummer
pCountryCode	CountryCode nach ISO 3166
pBuffer	Zeiger auf die ID-Nummer (1 – 274.877.906.944)
BufLen	Länge der Daten (muss fest auf 8 Byte sein)
pAnimalFlag	Zeiger auf AnimalFlag
pDatenblockFlag	Zeiger auf Datenblock Flag
pReserved	Zeiger auf Reserved
pExtension	Zeiger auf Extension Daten
Rückgabewert	-1 Fehler, > 0 Länger der Daten in pBuffer

Mit dieser Funktion kann man Animal FDX-B Transponder einlesen.

int TSLF_Read_HDX(int PortHandle, int * pCountryCode,
BYTE * pBuffer, int BufLen, int *pAnimalFlag,
int *pDatenblockFlag, int *pReserved, int *pExtension**)**

PortHandle	Zugriffsnummer
pCountryCode	CountryCode nach ISO 3166
pBuffer	Zeiger auf die ID-Nummer (1 – 274.877.906.944)
BufLen	Länge der Daten (muss fest auf 8 Byte sein)
pAnimalFlag	Zeiger auf AnimalFlag
pDatenblockFlag	Zeiger auf Datenblock Flag
pReserved	Zeiger auf Reserved
pExtension	Zeiger auf Extension Daten
Rückgabewert	-1 Fehler, > 0 Länger der Daten in pBuffer

Mit dieser Funktion kann man Animal HDX Transponder einlesen.
Achtung: Das HDX Format wird nur von TS-RW38 AC und TS-RW68 AC unterstützt!

Programmierschnittstelle (SDK) der TS-RW Geräte

int TSLF_Write_FDXA(int PortHandle, int TTyp, BYTE *pFactoryID,
BYTE * pIdNumber, int pIdLen, int Lock**)**

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
pFactoryID	Die FactoryID (UID) des selektierten Transponders. Je nach Transpondertyp hat diese einen unterschiedlichen Wertebereich. Der Speicherplatz für die FactoryID muss 8 Byte betragen.
pIdNumber	Identifikationsnummer (5 Byte Länge gemäß FDX-A Spezifikation)
pIdLen	Länge des Datenbereiches für Identifikationsnummer (muss 5 Byte sein)
Lock	0 = Standardwert. Wenn dieser Wert 1 ist, werden im Transponder die Blöcke schreibgeschützt.
Rückgabewert	-1 Fehler, 0 Erfolgreich geschrieben 2 Transponder erkannt, der Transponder ist schreibgeschützt, konnte nicht geschrieben werden 3 Transponder erkannt, der Transponder hat dieselbe Factory ID die übergeben wurde, der Transponder wurde nicht beschrieben. 4 Process pending, siehe nächste Seite.

Mit dieser Funktion wird ein Transponder selektiert, bei Bedarf die Konfiguration geändert (formatiert) und anschließend mit den Daten beschrieben und wenn gewünscht schreibgeschützt.
In pFactoryID wird die UID des beschriebenen Transponders zurückgeliefert.
Um zu vermeiden, dass derselbe Transponder mehrfach beschrieben wird, kann in pFactoryID auch die zuletzt verwendete UID übergeben werden. Es wird dann geprüft, ob der Transponder mit dieser UID erkannt wird und dann der Rückgabewert 3 zurückgegeben.
Das Schreiben der Blöcke wird bis zu 5x wiederholt, wenn ein Schreibfehler auftritt.

Programmierschnittstelle (SDK) der TS-RW Geräte

int TSLF_Write_FDXB(int PortHandle, int TTyp, BYTE *pFactoryID,
int CountryCode, BYTE * pIDNummer, int IDNummerLen,
int AnimalFlag, int DatenblockFlag, int Reserved,
int Extension, int Lock**)**

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
pFactoryID	Die FactoryID (UID) des selektierten Transponders. Je nach Transpondertyp hat diese einen unterschiedlichen Wertebereich. Der Speicherplatz für die FactoryID muss 8 Byte betragen.
CountryCode	Ländercode nach ISO 3166
pIDNummer	Zeiger auf die ID-Nummer (1 – 274.877.906.944)
IDNummerLen	Länge der Daten (muss fest auf 8 Byte sein)
AnimalFlag	AnimalFlag 0:Industrial, 1:Animal
DatenblockFlag	0: kein Data Block, 1: Data Block vorhanden
Reserved	Reservierter Bereich (14 Bit)
Extension	Erweiterter Bereich. (24 Bit)
Lock	0 = Standardwert. Wenn dieser Wert 1 ist, werden die entsprechenden Blöcke schreibgeschützt.
Rückgabewert	-1 Fehler, 0 Erfolgreich geschrieben 2 Transponder erkannt, der Transponder ist schreibgeschützt, konnte nicht geschrieben werden 3 Transponder erkannt, der Transponder hat dieselbe Factory ID die übergeben wurde, der Transponder wurde nicht beschrieben. 4 Process pending, siehe nächste Seite.

Mit dieser Funktion wird ein Transponder selektiert, bei Bedarf die Konfiguration geändert (formatiert) und anschließend mit den Daten beschrieben und wenn gewünscht schreibgeschützt.
In pFactoryID wird die UID des beschriebenen Transponders zurückgeliefert.
Um zu vermeiden, dass derselbe Transponder mehrfach beschrieben wird, kann in pFactoryID auch die zuletzt verwendete UID übergeben werden. Es wird dann geprüft, ob der Transponder mit dieser UID erkannt wird und dann der Rückgabewert 3 zurückgegeben.
Das Schreiben der Blöcke wird bis zu 5x wiederholt, wenn ein Schreibfehler auftritt.
Durch Angabe des Wertes NO_RETRY = 1000Hex = 4096 Dez. additiv zum Transpondertyp kann diese Schreibwiederholung unterdrückt werden.
Durch Angabe des Wertes READ_AFTER_WRITE = 100Hex = 256 Dez. additiv zum Transpondertyp wird ein Lesetest bei jedem geschriebenen Block durchgeführt und bei Fehlern automatisch der Block nochmals geschrieben. Hiermit kann die Ausfallrate nochmals gesenkt werden, jedoch wird die benötigte Zeitdauer erhöht.
Das zusätzliche Lesen der Blöcke dauert ca. 100 mSek.
Durch Angabe des Wertes NO_READ_AFTER_WRITE = 4000Hex = 16384 Dez additiv zum Transpondertyp wird die TTF Lesung nach erfolgreichem Schreiben unterdrückt um die Geschwindigkeit zu erhöhen. Wird der Wert NO_FACTORY_ID = 8000Hex = 32768 Dez zum Transpondertyp addiert, so wird die Lesung der FactoryID unterdrückt. Bei manchen Transpondertypen ist jedoch das Lesen der Factory ID obligatorisch, sodass hier diese nicht unterdrückt werden darf.

Programmierschnittstelle (SDK) der TS-RW Geräte

int TSLF_Write_HDX(int PortHandle, int TTyp, BYTE *pFactoryID,
int CountryCode, BYTE * pIDNummer, int IDNummerLen,
int AnimalFlag, int DatenblockFlag, int Reserved,
int Extension, int Lock**)**

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
pFactoryID	Die FactoryID (UID) des selektierten Transponders. Je nach Transpondertyp hat diese einen unterschiedlichen Wertebereich. Der Speicherplatz für die FactoryID muss 8 Byte betragen.
CountryCode	Ländercode nach ISO 3166
pIDNummer	Zeiger auf die ID-Nummer (1 – 274.877.906.944)
IDNummerLen	Länge der Daten (muss fest auf 8 Byte sein)
AnimalFlag	AnimalFlag 0:Industrial, 1:Animal
DatenblockFlag	0: kein Data Block, 1: Data Block vorhanden
Reserved	Reservierter Bereich (14 Bit)
Extension	Erweiterter Bereich. (24 Bit)
Lock	0 = Standardwert. Wenn dieser Wert 1 ist, werden die entsprechenden Blöcke schreibgeschützt.
Rückgabewert	-1 Fehler, 0 Erfolgreich geschrieben 2 Transponder erkannt, der Transponder ist schreibgeschützt, konnte nicht geschrieben werden 3 Transponder erkannt, der Transponder hat dieselbe Factory ID die übergeben wurde, der Transponder wurde nicht beschrieben. 4 Process pending, siehe nächste Seite.

Mit dieser Funktion wird ein HDX Transponder beschrieben. Es dürfen hier nur HDX Transpondertypen übergeben werden. Es ist zu beachten, dass HDX Transponder üblicherweise erst dann als Tiertransponder funktionieren, wenn sie gelockt wurden. Manche HDX Transponder sind auch nur einmal programmierbar.
Bei den meisten HDX Transpondertypen muss das Lock Flag gesetzt sein, damit sie korrekt beschrieben werden und anschließend auch von anderen Lesern gelesen werden können.

Achtung: Das HDX Format wird nur von TS-RW38 AC und TS-RW68 AC unterstützt!

Programmierschnittstelle (SDK) der TS-RW Geräte

Schreiben als Hintergrundprogramm
Da die Funktionen **TSLF_Write_FDXA**, **TSLF_Write_FDXB** bzw. **TSLF_Write_HDX** relativ lange dauert (ca. 500 – 700 mSek.) kann diese Funktion auch im "Overlapped Modus" aufgerufen werden. Hierzu wird der Transpondertyp um den Wert DO_OVERLAPPED = 2000Hex = 8192 Dez. erhöht. Dann wird die Funktion sofort beendet und als Rückgabewert TSLF_RETVAL_PROCESS_PENDING = 4 zurückgegeben.

Mit der Funktion
int TSLF_IsWriteFinished(int PortHandle, BYTE * pFactoryID**)**

PortHandle	Zugriffsnummer
pFactoryID	Die FactoryID (UID) des selektierten Transponders. Je nach Transpondertyp hat diese einen unterschiedlichen Wertebereich. Der Speicherplatz für die FactoryID muss 8 Byte betragen.

kann dann abgefragt werden, ob der Schreibvorgang abgeschlossen ist.

Die Rückgabewerte dieser Funktion sind wie bei TSLF_Write_FDXA bzw. TSLF_Write_FDXB beschrieben. Es ist darauf zu achten, dass auch in diesem Fall die FactoryID nur bestimmt wurde, wenn der Rückgabewert den Wert 0 (Erfolgreich geschrieben) erreicht hat.
Solange der Rückgabewert TSLF_RETVAL_PROCESS_PENDING = 4 geliefert wird ist der Schreibvorgang noch nicht abgeschlossen.
Durch geeignete Programmierung der Anwendung ist es hiermit möglich mehrere Transponder an mehreren gleichzeitig angeschlossenen Geräten simultan zu beschreiben.
Es muss jedoch sichergestellt sein, dass die Abstände zwischen den Antennen groß genug sind um ein Übersprechen zu vermeiden.

Programmierschnittstelle (SDK) der TS-RW Geräte

5.2. Anwendung der Befehle für FDX-A bzw. FDX-B Transponder:

1. Vorbereitung:
Zunächst muss die Geräteschnittstelle mit **TSLF_Open(….)** geöffnet werden.
Mit dem Befehle **TSLF_SetFilter(…)** wird der Filterwert für den gewählten Transpondertyp eingestellt. Da dieser Wert im Gerät gespeichert bleibt, ist dies nur erforderlich wenn der Transpondertyp geändert wird. Bei TS-RW38 Geräten muss der Filterwert nicht gesetzt werden, da diese Geräte Filtereinstellungen für alle Transpondertypen beinhalten.

2. Beschreiben des Transponders
Dies kann komfortabel mit der Funktion **TSLF_Write_FDXA(…)** bzw **TSLF_Write_FDXB(…)** erfolgen. Diese Funktion prüft zunächst ob ein Transponder vorhanden ist, formatiert ihn bei Bedarf und beschreibt in mit den gewünschten Daten. Da diese Funktion auch meldet wenn kein Transponder vorhanden war, kann diese Funktion auch zur automatischen Transpondererkennung und Beschreiben verwendet werden.

3. Lesen des Transponders
Die FDX Daten werden mit dem Befehl **TSLF_Read_FDXA(…)**bzw **TSLF_Read_FDXB(…)** ausgelesen. Diese Funktion liefert nur einen gefundenen Transponder, wenn der Transponder bereits richtig formatiert und beschrieben ist. Bei fabrikneuen Transpondern wird diese Funktion immer den Rückgabewert -1 mit Fehlercode "**21** Keine Antwort" melden. Das Lesen von FDX-A Transponder wird nur von TS-W38AC Geräten unterstützt!

4. Leser mit Mehrfachantennen (TS-W80-AC 8x)
Bei diesen Lesern werden mehrere Antennen abwechslungsweise betrieben. Die Umschaltung der Antennen erfolgt über den Befehl **TSLF_SetReaderAdresse(…)** Das Gerät verhält sich also so, wie 8 gleichzeitig angeschlossene Geräte mit einer Antenne.

Programmierschnittstelle (SDK) der TS-RW Geräte

5.3. Waste Transponder (EN14803) schreiben und lesen

Ein Waste Transponder (EN14803) hat 128 Bit Daten. Der Datenaufbau ist in ISO11785 beschrieben.
Hierin ist der Identifikation Code enthalten. Der Aufbau des Identifikation Codes wiederum ist in EN14803 beschrieben.

int TSLF_Read_EN14803(int PortHandle, int * pManufacturer, int * pSerialNumber**)**
int TSLF_Read_EN14803HDX(int PortHandle, int * pManufacturer, int * pSerialNumber**)**

PortHandle	Zugriffsnummer
pManufacturer	Herstellercode
pSerialNumber	Seriennummer
Rückgabewert	-1 Fehler, 0 Erfolgreich gelesen, 5 Ungültige Daten im Transponder

Mit dieser Funktion kann man Waste Transponder einlesen.
Die Daten werden dann entsprechend EN14803 und ISO 11785 interpretiert.
Mit der Funktion TSLF_Read_EN14803HDX können HDX Waste Transponder gelesen werden.

Achtung: Das HDX Format wird nur von TS-RW38 AC und TS-RW68 AC unterstützt!

Programmierschnittstelle (SDK) der TS-RW Geräte

int TSLF_Write_EN14803(int PortHandle, int TTyp, BYTE *pFactoryID,
int Manufacturer, int SerialNumber, int Lock**)**

PortHandle	Zugriffsnummer
TTyp	Transpondertyp aus Tabelle
pFactoryID	Die FactoryID (UID) des selektierten Transponders. Je nach Transpondertyp hat diese einen unterschiedlichen Wertebereich. Der Speicherplatz für die FactoryID muss 8 Byte betragen.
Manufacturer	Herstellercode
SerialNumber	Seriennummer
Lock	0 = Standardwert. Wenn dieser Wert 1 ist, werden die entsprechenden Blöcke schreibgeschützt.
Rückgabewert	-1 Fehler, 0 Erfolgreich geschrieben 2 Transponder erkannt, der Transponder ist schreibgeschützt, konnte nicht geschrieben werden 3 Transponder erkannt, der Transponder hat dieselbe Factory ID die übergeben wurde, der Transponder wurde nicht beschrieben. 4 Process pending, siehe nächste Seite.

Mit dieser Funktion wird ein Transponder selektiert, bei Bedarf die Konfiguration geändert (formatiert) und anschließend mit den Daten beschrieben und wenn gewünscht schreibgeschützt.
In pFactoryID wird die UID des beschriebenen Transponders zurückgeliefert.
Um zu vermeiden, dass derselbe Transponder mehrfach beschrieben wird, kann in pFactoryID auch die zuletzt verwendete UID übergeben werden. Es wird dann geprüft, ob der Transponder mit dieser UID erkannt wird und dann der Rückgabewert 3 zurückgegeben.
Das Schreiben der Blöcke wird bis zu 5x wiederholt, wenn ein Schreibfehler auftritt.
Durch Angabe des Wertes NO_RETRY = 1000Hex = 4096 Dez. additiv zum Transpondertyp kann diese Schreibwiederholung unterdrückt werden.
Durch Angabe des Wertes READ_AFTER_WRITE = 100Hex = 256 Dez. additiv zum Transpondertyp wird ein Lesetest bei jedem geschriebenen Block durchgeführt und bei Fehlern automatisch der Block nochmals geschrieben. Hiermit kann die Ausfallrate nochmals gesenkt werden, jedoch wird die benötigte Zeitdauer erhöht.
Das zusätzliche Lesen der Blöcke dauert ca. 100 mSek.
Mit dieser Funktion können sowohl FDX wie auch HDX Transponder beschrieben werden. Es muss der korrekte Transpondertyp gewählt sein.
Bei den meisten HDX Transpondertypen muss das Lock Flag gesetzt sein, damit sie korrekt beschrieben werden und anschließend auch von anderen Lesern gelesen werden können.

Achtung: HDX Formate werden nur von TS-RW38 AC und TS-RW68 AC unterstützt!

Programmierschnittstelle (SDK) der TS-RW Geräte

Schreiben als Hintergrundprogramm
Da die Funktion **TSLF_Write_EN14803** relativ lange dauert (ca. 500 – 700 mSek.) kann diese Funktion auch im "Overlapped Modus" aufgerufen werden. Hierzu wird der Transpondertyp um den Wert DO_OVERLAPPED = 2000Hex = 8192 Dez. erhöht. Dann wird die Funktion sofort beendet und als Rückgabewert
TSLF_RETVAL_PROCESS_PENDING = 4 zurückgegeben.

Mit der Funktion
int TSLF_IsWriteFinished(int PortHandle, BYTE * pFactoryID**)**

PortHandle	Zugriffsnummer
pFactoryID	Die FactoryID (UID) des selektierten Transponders. Je nach Transpondertyp hat diese einen unterschiedlichen Wertebereich. Der Speicherplatz für die FactoryID muss 8 Byte betragen.

kann dann abgefragt werden, ob der Schreibvorgang abgeschlossen ist.

Die Rückgabewerte dieser Funktion sind wie bei TSLF_Write_EN14803 beschrieben. Es ist darauf zu achten, dass auch in diesem Fall die FactoryID nur bestimmt wurde, wenn der Rückgabewert den Wert 0 (Erfolgreich geschrieben) erreicht hat.
Solange der Rückgabewert TSLF_RETVAL_PROCESS_PENDING = 4 geliefert wird ist der Schreibvorgang noch nicht abgeschlossen.
Durch geeignete Programmierung der Anwendung ist es hiermit möglich mehrere Transponder an mehreren gleichzeitig angeschlossenen Geräten simultan zu beschreiben.
Es muss jedoch sichergestellt sein, dass die Abstände zwischen den Antennen groß genug sind um ein Übersprechen zu vermeiden.

Programmierschnittstelle (SDK) der TS-RW Geräte

5.4. Anwendung der Befehle für EN14803 Transponder:

1. Vorbereitung:
Zunächst muss die Geräteschnittstelle mit **TSLF_Open(….)** geöffnet werden.
Mit dem Befehle **TSLF_SetFilter(…)** wird der Filterwert für den gewählten Transpondertyp eingestellt. Da dieser Wert im Gerät gespeichert bleibt, ist dies nur erforderlich wenn der Transpondertyp geändert wird. Bei TS-RW38 Geräten muss der Filterwert nicht gesetzt werden, da diese Geräte Filtereinstellungen für alle Transpondertypen beinhalten.

2. Beschreiben des Transponders
Dies kann komfortabel mit der Funktion **TSLF_Write_EN14803(…)** erfolgen. Diese Funktion prüft zunächst ob ein Transponder vorhanden ist, formatiert ihn bei Bedarf und beschreibt in mit den gewünschten Daten. Da diese Funktion auch meldet wenn kein Transponder vorhanden war, kann diese Funktion auch zur automatischen Transpondererkennung und Beschreiben verwendet werden.

3. Lesen des Transponders
Die EN14803 Daten werden mit dem Befehl **TSLF_Read_EN14803(…)** ausgelesen.
Diese Funktion liefert nur einen gefundenen Transponder, wenn der Transponder bereits richtig formatiert und beschrieben ist. Bei fabrikneuen Transpondern wird diese Funktion immer den Rückgabewert -1 mit Fehlercode "**21** Keine Antwort" melden.

4. Leser mit Mehrfachantennen (TS-W80-AC 8x)
Bei diesen Lesern werden mehrere Antennen abwechslungsweise betrieben. Die Umschaltung der Antennen erfolgt über den Befehl **TSLF_SetReaderAdresse(…)** Das Gerät verhält sich also so, wie 8 gleichzeitig angeschlossene Geräte mit einer Antenne.

Programmierschnittstelle (SDK) der TS-RW Geräte

6. Fehlerliste

Fehlernummer	Beschreibung
1	Fehler beim Öffnen des Ports, kein Gerät an dieser Schnittstelle vorhanden
5	Timeout-Wert ungültig
6	Porthandle ungültig, keine geöffnete Schnittstelle mit diesem Porthandle vorhanden
7	Timeout, innerhalb des bei TSLF_Open angegebenen Timeout ist keine Antwort vom Gerät eingegangen.
8	Datenlänge zu klein für empfangene Daten
9	Datenlänge der Sendedaten ungültig
10	Falscher Transpondertyp, für den angegebenen Transpondertyp ist diese Funktion nicht zulässig.
11	Keine Daten für Kommunikation
12	Blocknummer ungültig
13	Ungültiger Lockwert
14	Checksummenfehler, in den vom Transponder empfangenen Daten wurde eine fehlerhafte Prüfsumme erkannt. Entweder enthält der Transponder ungültige Daten, oder er ist zu weit vom Leser entfernt.
15	Keine Kommunikation zum Gerät
16	Checkbuffer leer, dieser Fehler sollte nie auftreten
17	NULL Zeiger übergeben, es können keine Antwortdaten gespeichert werden.
18	Ungültiger Mode angegeben, bitte Parameter prüfen
19	USB Gerät - Baudrate setzen nicht erlaubt.
20	Lese – Schreib Vergleich negativ bei Read after Write
21 (NACK = 15H)	Negative Antwort, Transponder antwortet nicht oder Transponder lehnt das Kommando ab.
22 (SYNC= 16H)	Das Gerät hat in seinen empfangenen Daten eine fehlerhafte Prüfsumme erkannt. Dies deutet auf Verbindungsprobleme, evtl. durch zu lange Kabel hin.
23	Transponder ist gesperrt
24 (CAN= 18H)	Ungültiges Kommando empfangen, das Gerät kann dieses Kommando nicht ausführen. Z. B.: wenn Programmer Kommandos (ab Kapitel 4) bei einem TS-R38 aufgerufen werden, der nur den ReaderMode beherrscht.
25	Ungültige Daten empfangen, die empfangenen Daten entsprechen nicht den Richtlinien für das erwartete Datenformat.