Programming Interface (SDK)
for TS-RW devices with 125 and 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

Programming Interface (SDK) of TS-RW devices

Ownership conditions:

Programming Interface (SDK) of TS-RW devices

Table of contents

Programming Interface (SDK) of TS-RW devices

Revision level:

Docu	Firmware	Date	Chapter	Name	Info
4.02		04.02.2011	4.0	Blank	Definition of transponder types HDX+ and NCS1015 was exchanged.
4.03	1.08	14.04.2011		Blank	Standard filter values corrected Command order changed structuring added Commands for LCD added
4.04	1.08	07.06.2011		Blank	FSK Formats corrected
4.05	1.13	03.11.2011		Blank	At Write FDX B additional parameter possible
4.06	1.16	24.01.2012		Blank	Additional Tag type HDX+ HiQ added
4.07	1.18	05.03.2012		Blank	Additional Tag type TMS37124 added
4.08	1.20	18.05.2012		Blank	Additional tag types for SIC7900 and SIC7888 added.
4.09	1.21	01.08.2012		Blank	Additional tag type for SIC7999 added
4.09	1.21	05.11.2012	4.0 4.2	Blank	Correction of Tag Type for SIC7999 Description of Tag Detection changed.
4.12	1.25	08.11.2013	2.6 3.1.1.	Blank	Function WriteTagKey added. Extended tag types.
4.13	1.27	11.04.2014	4.3	Blank	Hitag 1, Hitag S and SIC7888 extended to UID Request with anti-collision algorithm and Quiet function. Access to Hitag S in Crypto Mode
4.14	1.31	24.07.2014	2.5 3.1	Blank	Filter settings updated Reader mode parameter extended
4.15		02.12.2014	2.2	Blank	Setup of Gateway address
4.15		09.02.2015	6	Blank	Update of error list
4.16		22.05.2015	4	Blank	Additional command for AutoTune
4.16		08.06.2015		Blank	different missing items corrected
4.17	1.35	10.08.2015	2.5	Blank	Answer mode 7 added for FSK2a 2kBit
4.18		09.09.2015	4.6, 4.7	Blank	Read and Write NC34Bit implemented
4.18		19.10.2015	2.8	Blank	Description LCD Light added
4.20		01.03.2016	4.6, 4.7	Blank	Functions for AWID Data format added.
4.22		14.07.2016	2.5	Blank	Functions for device configuration for TS-RW82AC added
4.23	1.39	19.12.2016	1.1, 3.7	Blank	Extensions for simplified usage in Reader mode
4.25		19.06.2017	3.7	Blank	Amendment at errors in Reader mode
4.27		01.12.2017	2.3	Blank	Reader types updated, new functions included
4.29		10.08.2018		Blank	internal functionality of SDK updated
4.30		18.10.2018		Blank	additional functions added
4.31		01.04.2019		Blank	additional functions added
4.32		29.08.2018		Blank	additional functions TSLF_SetTimeout
4.33		06.12.2019		Blank	several devices opened simultaneously now always get different handles

Programming Interface (SDK) of TS-RW devices

Revision level:

Docu	Firmware	Date	Chapter	Name	Info
4.34		08.01.2020		Blank	the commands TSLF_Beep, TSLF_SetLED, TSLF_RawWrite are now executed correctly
4.35		08.06.2020	2.3	Blank	Command TSLF_KeepAppActive added to documentation
4.35		06.08.2020	4.6, 4.7	Blank	Commands added for IdentiCard format.
4.36		07.01.2021	4.5	Blank	Information for length added.
4.37		02.11.2021		Blank	Additional commands for TI HDX+
4.38		08.04.2022	4.8	Blank	Tag Type EL9265 added with special commands Support of TS-RW380
4.38		30.01.2023	4.9	Blank	Tag Type EM4582 added with special commands

1. Introduction

The programming interface is built for simple integration of TS-R3x/R6x and TS-W3x/W6x devices in arbitrary applications.

The interface to the devices is in GiS Standard Protocol G100 or G300.
The recognition of the device protocol is done automatically when the connection is established. This is done by making the version query and supporting the different commands depending on the found device.

The dynamic link library (DLL) provides all commands and maps the entire functionality of the module. You can use the DLL in several program languages.
Two variants of the DLL are given for 32 Bit or 64 Bit applications.

Type	Filename	Import library	Import Header for C / C++	Import for C#	Import for 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

The DLL supports many different types of transponders. Depending on the device type different transponder types are supported. You can get the type with **TSLF_GetDeviceVersion().**

Programming Interface (SDK) of TS-RW devices

1.1. Operation modes

Depending on the device, different operation modes are available. It is to be distinguished between the operation modes "Reader mode" and "Programmer mode".
Devices of "R" series work only in "Reader mode", Devices of "W" series only in "Programmer mode", while devices of "RW" series support both modes.

1.1.1. Reader mode

Automatic reading is called "reader mode". The reader works autonomous and sends the read transponder data through its interface as ASCII characters. Which data in which format is to be sent is adjusted using the "TS-R3x ReaderSetup" application or the functions described in chapter 3.
This mode is also often used, if the device has to be used without usage of the SDK, especially if used with serial interface in a third party application designed for serial interface connection or as HID (Human Interface device) in keyboard mode.
To be used with the SDK special access functions are available which do not work with GiS Standard protocol. (Chapter 3.7.)
Take care using other commands in reader mode, the reader will send data automatically if a tag is presented to the device, so this may cause collisions at the reply to standard commands.
This was the reason why special direct access commands had been created to access the buzzer and the LED which do not send any response. (Chapter 3.7.)

1.1.3. Programmer mode

In programmer mode all access to the transponder is done through commands.
The commands from Chapter 4 and 5 as well as the common commands from chapter 2 might be used.
In this mode it is possible to read and write all the blocks at different transponder types.

Programming Interface (SDK) of TS-RW devices

1.2. Error treatment

All function give a return value of -1 if an error occurred. To get the error code, call **TSLF_GetLastError()**.
Error list is at the end of the document.

1.3. Definitions

The following data types are used in the function declarations:

int	32 Bit Integer
BYTE	8 Bit unsigned integer
BYTE *	pointer to array with 8 Bit unsigned integer values
char *	pointer to array with 8 Bit signed integer values, (ANSI Strings in C definition) mostly 0-terminated

The order of data in the fields is always LSB first.

2. General commands

int TSLF_LibVersion()
Return: -1 error, >0 Version number with factor 100. E.g. 100 is 1.00

Provides DLL version.
This function does not need TSLF_Open.

int TSLF_IsVirtualComInstalled()
Return: 1 Virtual Com Driver for TS-RW38 is installed
0 Virtual Com Driver for TS-RW38 is not installed.

int TSLF_IsDeviceDriverMissing()
Return: 0: There are no GiS Devices with missing drivers attached
1: driver for attached RW34 USB device is missing
2: driver for attached RW38 VCOM device is missing
3: driver for attached RW34 USB and RW38 VCOM device is missing

2.1. Get attached devices

int TSLF_GetCOMDeviceNames (char* NameList, int BufferSize**)**
NameList Name list is a number of null terminated strings.
End of the list is an empty string.
BufferSize Size of the buffer the user allocates.
Return: <0 error, or the value is the size of actually used buffer

The function **TSLF_GetCOMDeviceNames** provides the names of all available COM-interfaces.
Each name consists of a NULL terminated string. Example: “COM1”.
The maximum number of serial interfaces is 255.
The serial interfaces can exist as real or virtual (through USB) interfaces.
At virtual COM ports the name of the virtual com port is also displayed.

Examples:

"\\.\COM1"	real COM port 1
"\\.\COM4 [GiS Virtual COM]"	Virtual COM Port 4 appropriated through the "GiS Virtual COM" driver
"\\.\COM14 [GiS/FTDI Virtual COM]"	Virtual COM Port 14 appropriated through the "GiS/FTDI Virtual COM" driver

The found names can be used as input for TSLF_Open directly.

Programming Interface (SDK) of TS-RW devices

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

NameList	Name list is a number of null terminated strings. End of the list is an empty string.
BufferSize	Size of the buffer the user allocates.
Return:	<0 error, or the value is the size of actually used buffer

The function **GetUSBDeviceNames** provides the USB names (serial numbers) of the USB devices from GiS which are to be used directly with this SDK. Each name consist of a NULL terminated string with at least 8 ASCII characters. Example: “11360001” or “1460-0001 HID”.
Because of the addition “HID” it can be recognized if the device is a HID (HumanInterfaceDevice) Device (USB Keyboard Simulation).

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

NameList	Name list is a number of null terminated strings. End of the list is an empty string.
BufferSize	Size of the buffer the user allocates.
Return:	<0 error, or the value is the size of actually used buffer

The function **GetUSBDeviceNamesEx** provides the USB names (serial numbers) of the USB devices from GiS. Also devices which are not to be used directly with this SDK are listed (p.e.: Reader for 13.56 MHz). Each name consist of a NULL terminated string with at least 8 ASCII characters. Example: “11360001” or “1460-0001 HID”.
Because of the addition “HID” it can be recognized if the device is a HID (HumanInterfaceDevice) Device (USB Keyboard Simulation).

Programming Interface (SDK) of TS-RW devices

2.2. Functions for Ethernet devices

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

NameList	Name list is a number of null terminated strings.
	End of the list is an empty string.
BufferSize	Size of the buffer the user allocates.
Return:	<0 error, or the value is the size of actually used buffer

The function **GetLanDeviceNames** provides the LAN names of the LAN devices.
Only GiS TS-R3x and TS-W3x devices are factored in. Each name consists of a NULL terminated string. The buffer space must be big enough to fit for all devices in the network.
Examples:

192.168.0.100-10001
[TS-LAN01]

The device name can be the IP-Address of the device followed by the port number or at DHCP devices, the DHCP name. The DHCP Name is written in [ ]. With this function only devices are found, which are reachable with the actual network settings.

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

NameList	Name list is a number of null terminated strings.
	End of the list is an empty string.
BufferSize	Size of the buffer the user allocates.
Return:	<0 error, or the value is the size of actually used buffer

The function **GetLanDeviceNamesEx** provides the LAN names of all the LAN devices.
Each name consists of a NULL terminated string. The buffer space must be big enough to fit for all devices in the network.
Examples:

192.168.0.100-10001
[TS-LAN01]169.194.245.01-10001

The device name can be the IP-Address of the device followed by the port number or at DHCP devices, the DHCP name followed by IP-Address and Port number. The DHCP Name is written in [ ]. With this function all devices are found, also if they are not reachable with the actual network settings.

Programming Interface (SDK) of TS-RW devices

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

OldAddress	Old IP-Address
NewAddress	New IP-Address
IPMask	New IP-Mask

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

OldAddress	Old IP-Address
NewAddress	New IP-Address
IPMask	New IP-Mask
Gateway	New gateway address

With this function the IP-Address and port number of a device can be changed.
To do this, the device has to be reachable with the actual Network settings.
The parameters are all 0 terminated ASCII Strings.
The IP-Addresses are either the IP-Address followed by the port number or the DHCP Name embedded in [ ].
Example:

192.168.0.100-10001
[TS-LAN01]

In the IP Mask the significant bits are set as bit mask.
Example:

255.255.255.0

At Gateway address the IP Address of the gateway is given. If no gateway shall be set, please use 0.0.0.0 . At gateway no DHCP Name can be used.

With changing the IP-Address the device is restarted. It can last up to 25 Seconds until the device is available in the network.

Programming Interface (SDK) of TS-RW devices

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

Name	Name of device
Address	IP-Address
IPMask	IP-Mask
Gateway	Gateway address

With this function the IP-Address and the port number of a device is read.
To do this, the device has to be reachable with the actual Network settings.
The parameters are all 0 terminated ASCII Strings.
In Address either the DHCP Name or the IP Address is delivered.
In each passed string enough space to store the address has to be available (at least 30 Byte)

int TSLF_IsLanDeviceAvailable(LPCSTR Address**)**

Address	IP-Address

Return value: < 0 Error,
0 Device not available
> 0 Device available

The IP-Address is either the IP-Address followed by the port number or the DHCP Name embedded in [ ].
Example:

192.168.0.100-10001
[TS-LAN01]

With this function can be tested if the device is available in the network.

Programming Interface (SDK) of TS-RW devices

2.3. Establish connection to a device

Attention: **The function TSLF_Open() must be called before using any other command. The returned handle is necessary for all read and write commands. If there is an interface open error, the return value is a negative error number. See error list.**

int TSLF_Open(LPCTSTR strInterfaceName, int Baudrate, int ParityMode, int Timeout**)**
Opens the interface for all RS232, USB and LAN devices.

strInterfaceName	Name of the interface. e.g. “COM1” or serial number of USB devices. For **USB** devices you have to put in the device name. At **USB HID** devices you have to put in the device name followed by "HID" You can get the device name with the function **TSLF_GetAllUSBDeviceNames.** E.g. “10610011”. or "1317-0001 HID" At **LAN** devices you have to give the address of the device. The available LAN devices can be found with function **TSLF_GetAllLanDeviceNames**.
Baudrate	Set the baud rate. Valid values are 2400, 4800, 9600, 19200, 38400, 57600, 115200 and 230400. Attention: Not all devices support every baud rate. Please read device specification. for valid baud rates. **Default setting for RS232 readers is 19200.** **USB and LAN devices do ignore the baud rate. (internally fixed 19200)**
ParityMode	0: 8 Data Bits, 1 Stop bit, no Parity 1: 8 Data Bits, 1 Stop bit, even Parity 2: 8 Data Bits, 1 Stop bit, odd Parity 3: 8 Data Bits, 2 Stop bit, no Parity
Timeout	Time span after which communication shall break off. (in milliseconds)
Return:	>0: **PortHandle** -1: Invalid port -5: Timeout invalid value -15: Device does not respond, maybe no device connected

Optionally the device address can be given at the interface name. For example COM1:4 opens at COM 1 the device with address 4. This is necessary if the device address is not 1, because TSLF_Open makes a connection to the device and this is only accepted if the device answers.
The function automatically recognizes the under laying protocol of the device
(normally GiS LowLevel Protocol TS-RW3x, at MultiX devices MultiX Modbus Protocol)

Programming Interface (SDK) of TS-RW devices

Attention: **You generally have to call TSLF_Close() at the end of the application. Because this function closes the interface and frees all resources.**

int TSLF_Close(int PortHandle**)**

PortHandle	Port handle
Return:	-1 error, 0 OK

Closing the communication interface to the device. After this action the PortHandle is invalid.

int TSLF_KeepAppActive (int PortHandle, int bActive**)**

PortHandle	Port handle
bActive	0: Application is inactive while a SDK function is running 1: Application keeps active while a SDK function is running

This is used to control the behavior of a calling application. It is helpful if functions are called in very short interval. After TSLF_Open it is set to bActive = 0. So while a function is called the application is blocked. After calling KeepAppActive with bActive = 1 the application stays active also while inside a function call. It is to be ensured by the user that no more functions for this port handle are called while a function is running.

int **TSLF_IsUSB(**int PortHandle)

PortHandle	Port handle from OpenPort
Return:	-1 error, 0 = **NO** USB Port, 1 = USB Port

int TSLF_SetReaderAdresse(int PortHandle, int Adresse**)**

PortHandle	Port handle
Adresse	Address of module, default value after OpenPort is 1 or the given address at OpenPort. The address is only used in RS485 connections if more than one device is attached.

Access to multiple devices at on interface with address selection. At devices with multiple antennas this function is used to select between the antennas.

Programming Interface (SDK) of TS-RW devices

int **TSLF_IsReader(**int PortHandle)
PortHandle Port handle from OpenPort
Return: -1 error
0 = Reader Mode is **not** supported
1 = Reader Mode is supported
This function returns if the reader mode is supported by the device, not if the reader mode is activated.

int **TSLF_IsProgrammer(**int PortHandle)
PortHandle Port handle from OpenPort
Return: -1 error
0 = Programmer Mode is **not** supported
1 = Programmer Mode is supported
This function returns if the programmer mode is supported by the device and the corresponding commands are allowed, not if the programmer mode is activated.

int **TSLF_IsPlusProgrammer(**int PortHandle)
PortHandle Port handle from OpenPort
Return: -1 error
0 = Extended Programmer Mode is **not** supported
1 = Extended Programmer Mode is supported
Returns if extended functions are available, which are only supported at Plus or AC devices.

int **TSLF_IsAnimalProgrammer(**int PortHandle)
PortHandle Port handle from OpenPort
Return: -1 error
0 = Animal Programmer Mode is **not** supported
1 = Animal Programmer Mode is supported
Returns if the device is a 134,2 kHz device with functions for Animal Code.

int **TSLF_IsHDXProgrammer(**int PortHandle)
PortHandle Port handle from OpenPort
Return: -1 error
0 = HDX Programmer Mode is **not** supported
1 = HDX Programmer Mode is supported
Returns if the device is a 134,2 kHz device with HDX support.

Programming Interface (SDK) of TS-RW devices

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

PortHandle	Port handle
Baudrate	2400, 4800, 9600, 19200 and 38400 (Default = 19200).
ParityMode	0: 8 Data Bits, 1 Stop bit, no Parity
	1: 8 Data Bits, 1 Stop bit, even Parity
	2: 8 Data Bits, 1 Stop bit, odd Parity
	3: 8 Data Bits, 2 Stop bit, no Parity
Return:	-1 error, 0 OK

Set baud rate (only for RS232 devices; USB or LAN devices provide an error).
The baud rate 38400 is only supported at TS-RW38 devices.

int TSLF_GetLastError(int PortHandle**)**

PortHandle	Port handle
Return:	See error numbers in appendix

This recalls the last occurred error. All functions which access the opened device return the common value -1 for errors. With the Function TSLF_GetLastError the exact error reason can be obtained.

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

PortHandle	Port handle
pDevVer	Pointer to version data
VerLen	Length of version data (4 Byte)
pDevName	Pointer to device name
NameLen	Length for device name
Return:	-1 Error
	> 0 Length of version data

Device number and device firmware version in ASCII Format.

Byte 1 – 3	Device number
Byte 4	Firmware version (0 – 9, A – Z)

Device name:
The device name is returned as 0-terminated string.
If a Null pointer is given as pDevName or NameLen is too short, the device name is not registered.
The device name is as given in the following table.
The maximum length of a device name is 32 Byte.

Programming Interface (SDK) of TS-RW devices

The following devices are supported by this Dll:

Device number	Description	Reader mode	Programmer	Animal Code	supported Transponder types (ex Firmware version)											
					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

* The following HDX transponder types are supported:
TI TMS37190, NCD1015
TI TMS37124, SIC7900, SIC 7900X, SIC7999 (At RW38/68 from firmware version 20)

Programming Interface (SDK) of TS-RW devices

2.4. Set operation mode

int TSLF_SetReaderMode(int PortHandle, int mode**)**

PortHandle	Port handle
Mode	0 set device to programmer mode.
	1 set device to reader mode.
	2 set device to power up mode
	80H set power up mode as programmer mode
	81H set power up mode as read mode
Return:	-1 error, 0 OK

For devices which can operate in reader or programmer mode, this command is used to activate the reader resp. programmer mode.
In reader mode the device send the data of the transponder without protocol header as set up in the parameter settings for the reader mode. This interferes communication in programmer mode and so the reader mode has to be deactivated during programmer usage.
Also especially at HID (Keyboard emulation) devices it can be very uncomfortable if the keyboard emulation is activated at the time a transponder is attached.

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

PortHandle	Port handle
Mode	0H set power up mode as programmer mode
	1H set power up mode as reader mode.
Rückgabewert:	-1 error, 0 OK

Same as TSLF_SetReaderMode function with modes 80H resp. 81H.

int TSLF_GetPowerUpMode(int PortHandle**)**

PortHandle	Port handle
Rückgabewert:	-1 error,
	0 Power up mode is programmer mode
	1 Power up mode is reader mode

Reads power up mode from the device.

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

PortHandle	Port handle
OnOff	0 = turn antenna field off, 1 = turn antenna field on
Return:	-1 error, 0 OK

Turn antenna field on or off.

Programming Interface (SDK) of TS-RW devices

2.5. Set device parameters

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

PortHandle	port handle
TTyp	Transponder type from table
pParam	pointer to parameter data (one filter parameter set)
nParamLen	Length of parameter data
Rückgabewert:	-1 error
	0 Device does not support reading of filter parameter.
	>0 length of filter parameter

Attention: Only TS-RW38 device allow the reading of filter parameter sets.

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

PortHandle	port handle
TTyp	Transponder type from table
pParam	pointer to parameter data (one filter parameter set)
nParamLen	length of parameter data
Rückgabewert:	-1 error, 0 OK

At TS-RW38 devices the filter parameter sets are permanently stored separately for each transponder type in the device. At access of a transponder type automatically the correlating filter parameter set is used.
At all other devices only one filter parameter can be stored permanently. So this parameter is valid for all transponder types. At these devices the transponder type has no effect. Only the para1 (Filter) is transmitted, the extended parameters are only valid at TS-RW38 devices.
At all former devices the correct filter has to be set at each change of transponder type.

Programming Interface (SDK) of TS-RW devices

Definition of filter parameter sets:

Transponder type	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*	-	-	-	-

The times are always given in 1/RF,
this means at RF = 125kHz units are 8µSek, at RF = 134,2kHz units are 7,45µSek.
- the value is not relevant for this transponder type and has to be set to 0.
* the answer time is relative to the predefined answer time adjustable in ± 127 units.

The parameter 1 Filter gives the setting of the internal filters for the signal processing.
The bits are used as follows:

Bit7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Fast Mode	RFU	RFU	Low pass Filter active	Gain factor		High pass Filter	Low pass Filter
0 = Normal mode 1 = Fast mode	0	0	0 = active 1 = inactive	0 = 100 2 = 500	1 = 200 3 = 1000	0 = 40 Hz 1 = 160 Hz	0 = 3kHz 1 = 6kHz

Fast mode is only available for Atmel Transponder, read and write access will be done without multiple evaluation.

Programming Interface (SDK) of TS-RW devices

int TSLF_SetIO(int PortHandle,int Maske, int Daten**)**

PortHandle	Port handle
Maske	Mask values for outputs to set
Daten	Values for outputs all bits are used, which are set in the mask.
Return:	-1 error, 0 OK

With this outputs of the device are set. Depending on the device, different outputs are available. After power on at the device the LED's are set automatically. After using this command only those outputs are set automatically which are not included in the mask.

Definition of bits:

Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
led yellow	Led green	Led red	Buzzer	Out 3	Out 2	Out 1	Out 0

If the device is equipped with a relay output, this is connected through Out 0.

Example: Mask: C0H Data: 40H sets the green led and turns the yellow led off, the red led and all other outputs are kept unattended and are set through the reader regarding to the operation mode.

int TSLF_ReadIO(int PortHandle, int * Daten**)**

PortHandle	Port handle
Daten	Value of the inputs read
Return:	-1 error, 0 OK

It depends on the device version which inputs are available.
Definition of 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

If the device is equipped with a sensor input, this is available in In 0.

int TSLF_SetDefault (int PortHandle**)**

PortHandle	Port handle
Return:	-1 Error, 0 OK

Activate Factory default settings.
This command is only supported at devices with device number 081.

Programming Interface (SDK) of TS-RW devices

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

PortHandle	Port handle
pConfig	pointer to configuration
ConfigLen	Length of buffer
Return:	-1 Error, 0 OK

Write the configuration to the device. The configuration is dependent of the device. This command is not supported at all devices.

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

PortHandle	Port handle
pConfig	pointer to read buffer
ConfigLen	Length of buffer
Return:	-1 Error, length of actually read data.

Read the configuration from the device. The configuration is dependent of the device. This command is not supported at all devices.

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

PortHandle	Port handle
pSerial	pointer to read buffer
Bufken	Length of buffer
Return:	-1 Error, length of actually read data.

Read serial number of the device. This command is only supported at new devices.
Serial number is reported as 4 Byte binary Number with LSB first.

Programming Interface (SDK) of TS-RW devices

int TSLF_GetAntennaCount(int PortHandle);
PortHandle Port handle
Return: -1 Error, >0 Number of antennas of the device

This function is supported only by TS-RW82AC, at all other devices 1 antenna is returned.

int TSLF_SetAntennaPowerPreset(int PortHandle, int Power);
PortHandle Port handle
Power Antenna power set point in Volt
Return: -1 Error, 0 OK

Setup of the setpoint for the antenna power.
This function is supported only by TS-RW82AC.

int TSLF_GetAntennaPowerPreset(int PortHandle);
PortHandle Port handle
Return: -1 Error, >=0 set point of antenna power in V

Reads set point of antenna power.
This function is supported only by TS-RW82AC.

int TSLF_GetAntennaPower(int PortHandle);
PortHandle Port handle
Return: -1 Error, >=0 actual value of antenna power in V

With this the actual measured value of antenna power is returned.
This function is supported only by TS-RW82AC.

Programming Interface (SDK) of TS-RW devices

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

PortHandle	Port handle	
AnswerMode	Answer mode	
	0= Manchester	2 kBit Data Rate
	1= Biphase	4 kBit Data Rate
	2= FSK2a	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

Return: -1 Error, 0 OK

This command is supported only at devices with animal coding support.
At this Biphase (1) is supported at all AC devices and at FDX devices beginning with Firmware Version 'E' (14). FSK2a (2) is only supported at TS-RW38Plus (243) and TS-RW38AC (244).
Manchester 4kBit and Biphase 2kBit ist only supported at TS-RW38Plus (243) and TS-RW38AC (244) beginning with Firmware version 3.
FSK2a (7) is only supported at TS-RW38Plus (243) and TS-RW38AC (244) beginning with Firmware version 35.

Normally, the transponder sends in Manchester Code. At special applications the Biphase Code (Animal FDX-B) or FSK2a (Animal FDX-A) is used.
Dependent of the transponder type only the animal code in the TTF data stream is sent in Biphase coding or the transponder answers to all commands in Biphase coding.
To make it possible to read and write also transponders configured to biphase or FSK2a coding correctly, this command is used to switch the interpretation of the transponder answers.

The command works together with commands for Q5, EM4569, EM4305, ATA5577 and ATA5575 transponders.

Programming Interface (SDK) of TS-RW devices

int TSLF_GetAnswerMode(int PortHandle**)**
PortHandle Port handle
Return: -1 Error, >= 0 AnswerMode

The AnswerMode returned is as described at TSLF_SetAnswerMode.
Only TS-RW38 devices can return the actual answer mode. At all other devices the answer mode last set in the SDK is returned.

Remark:
Differences in transponder types:

Hitag 2, Hitag S and SIC7888
At these transponders the answer to commands is always send in Manchester Code, also if the TTF data stream operates in Biphase coding.

Q5
At Q5 transponders the answer to commands is always working the same way as the TTF data stream does. So either all is in Biphase or all is in Manchester Code.
The only exception is the UID at page 1. Is the output set to page 1, so the transponder sends always in Manchester code, independent from other settings. The UID is UNIQUE coded.

EM4569, EM4305, ATA5577 and ATA5575
At these transponders the answer to commands is always working the same way as the TTF data stream does. So either all is in Biphase or all is in Manchester Code.

Hitag 1 and Titan
These transponder types are not concerned, because these transponder types are working always in Manchester coding.

2.6. Write key value

With this function the standard key to access Hitag 2 Chips in reader mode and at the writing functions is set.
It does NOT save a password in the transponder!

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

PortHandle	Port handle
pKey	Pointer to Key.
KeyLen	Length of Key (4 Byte)
Return:	-1 error, 0 OK

At Hitag2 transponders the selection is protected with a password. The password for selecting the transponder is stored with this command in the device.
This is used in reader mode when reading blocks of Hitag 2 and in programmer mode when writing formatted data to Hitag 2 transponders.
The predefined password for Hitag 2 transponders is “RKIM” (52 4B 49 4D)

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

PortHandle	Port handle
TTyp	transponder type, only 30^{Hex} for ATA5577 with password is valid.
pKey	Pointer to Key.
KeyLen	Length of Key (4 Byte)
Return:	-1 error, 0 OK

Using this function the password for reader mode is set for the given transponder type. This password is only used, if in the parameters for reader mode the corresponding transponder type is used.

Programming Interface (SDK) of TS-RW devices

2.7. Issue general command

Some devices accept additional commandos, which can be accessed by this general command function.

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

PortHandle	Port handle
Cmd	Command to be sent
pSendBuf	Pointer to data to send
SendBufLen	length of data to send
pRecvBuf	pointer to data to receive
RecvBufLen	max. length of data to receive
Rückgabewert:	-1 error, >= 0 length of received data in pRecvBuf

If direct communication to the device is needed, for example in reader mode without any data protocol, this can be done with the following functions:

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

PortHandle	Port handle
pBuffer	Pointer to write buffer
BufLen	Length of buffer
Return:	-1 Error, 0 OK

This function writes arbitrary data to the interface.
No protocol implied.

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

PortHandle	Port handle
pBuffer	Pointer to read buffer
BufLen	Length of buffer
Return:	-1 Error, length of actually read data.

This function reads arbitrary data to the interface.
No protocol implied.

Programming Interface (SDK) of TS-RW devices

2.8. Text output to LCD

The TS-RW38 can be equipped with a LCD display.
This display can be accessed using the following commands.

Display size (22 x 33 mm)
Two different fonts can be chosen.
Small font with 8 lines with 17 characters
or
Large font with 4 lines with 8 characters

int TSLF_ClearLCD (int PortHandle)

PortHandle	Port handle
Return value:	-1 Error, 0 OK

With this function the display is cleared.

int TSLF_SetLCDFont (int PortHandle, int FontNr)

PortHandle	Port handle
FontNr	0: small font
	1: large font
Return value:	-1 Error, 0 OK

With this function the font to be used for the next output will be set.

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

PortHandle	Port handle
StartLine	First line of text output,
	valid from 1-8 at small font, 1-4 at large font
strText	Zero terminated text to be displayed.
Return value:	-1 Error, 0 OK

With this function the given text is shown at the display.
If the text is larger than the line length, automatic line break is done until the last line is reached.

int TSLF_SetLCDLight(int handle, int Helligkeit)

handle	Port handle
Helligkeit	Brightness of background light, 0 = off, 5 = max. Brightness

3. Parameter settings for reader mode

These commands are used for setting parameters of the device in reader mode as well as to manage data transmission in reader mode.

3.1 Parameter read and write

If multiple parameter data structures are given, the transponder types are polled alternating.
If only one data structure is given, this does not have to be filled completely, if multiple data structures are given, these have to be set completely with 20 Byte per data structure.
Attention: only device type TS-R38 supports multiple data structures. Al all former devices only one data structure with maybe less parameters is supported.
The parameter "BytesMask" is supported only at TS-R38 Version 1.31 or newer. If in BytesMask all bits are set ($FF^{Hex)}$, so the values for number of characters and valid bytes are valid for this register . Otherwise only the activated bytes are returned. This is only allowed for ASCII and Hexadecimal output.

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

PortHandle	Port handle
pBuffer	Pointer to read buffer. Read buffer must have length 11 or 12 bytes. See: **3.1.1 Parameter structure**.
BufLen	Length of read buffer (20 Byte per data structure)
Return:	-1 error, length of actually read data

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

PortHandle	Port handle
pBuffer	Pointer to write buffer. See: **3.1.1 Parameter structure**.
BufLen	Length of write buffer (20 Byte per data structure)
Return:	-1 error, 0 OK

3.1.1 Parameter structure

The parameter for reading and writing are passed always in the same order.

Pos.	Length	Name	*Description*
0	1	TTyp	*Transponder type:* 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	The entry consist always of 4 register numbers and the end marker 0xFF Hex The entry can have some special meanings: Is transponder type Default (TTF) = 0 given, the first byte of the registers give the format of the transponder answer: 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) The FDX Types are only valid at TS-RW38+ and TS-RW38AC, the HDX Types are only valid at TS-RW38AC. If a transponder type is used, the block numbers to be read are defined here. A special meaning has block number $FE^{Hex,}$ which is used for the UID of the transponder Byte 5 is always the end sign ff^{Hex}. z.B.: 00^{Hex},01^{Hex},$0f^{Hex}$,05^{Hex},ff^{Hex}.
6	1	Ausrichtung	*Byte order* *Ordering of data* 0 = LSB first 2 = LSB first bits swapped 1 = MSB first 3 = MSB first bits swapped
7	1	DTyp	*Data type:* 0 = Hexadecimal, 3 = Decimal without leading zeros, 1 = Decimal, 4 = Hexadecimal with lower case letters 2 = ASCII,
8	1	Zeichen	*Number of characters.* (1–32). This defines how much characters shall be emitted.
9	1	ValidBytes	(1-16) Number of bytes to be used out of the UID or data block. With this for example the upper bits of the UID can be masked. Default value 5
10	1	ValidFrom	(1-16) Start byte for data transfer.
11	1	TypKenn1	Identification mark for the transponder type to be transmitted ahead of the data, 1 ASCII character, if 0 nothing is transmitted
12	1	TypKenn2	Identification mark for the Transponder type to be transmitted after the data, 1 ASCII character, if 0 nothing is transmitted
13	1	ModSpeed	Setting of Modulation type and speed 0: Manchester Code, 2 kBit data rate 1: Biphase Code, 4 kBit data rate 2: FSK2a Code, 2.5 kBit data rate 4: FSK2a Code, 1.25kBit data rate, Manchester coding 5: Manchester Code, **4 kBit** data rate 6: Biphase Code, **2 kBit** data rate
14	4	BytesMask	The entry consists of 4 masks for the 4 above given registers. Each mask contains which bytes of register data are used for output. The lowest bit marks he lowest significant byte.
18	2	-	Reserved, default value 0

3.2 Prefix

The prefix is the string to be transmitted ahead of the transponder data.
The prefix consists of 31 characters maximum. End marker is 0xFF Hex.

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

PortHandle	Port handle
pBuffer	Pointer to write buffer.
BufLen	Length of write buffer
Return:	-1 error, 0 OK

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

PortHandle	Port handle
pBuffer	Pointer to read buffer.
BufLen	Length of read buffer
Return:	-1 error, length of actually read data

3.3 Suffix

The suffix is the string to be transmitted after the transponder data.
The suffix consists of 31 characters maximum. End marker is 0xFF Hex.

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

PortHandle	Port handle
pBuffer	Pointer to write buffer.
BufLen	Length of write buffer
Return:	-1 error, 0 OK

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

PortHandle	Port handle
pBuffer	Pointer to read buffer.
BufLen	Length of read buffer
Return:	-1 error, length of actually read data

3.4 Termix

The Termix is the string to be transmitted between two blocks of transponder data.
The Termix consists of 31 characters maximum. End marker is 0xFF Hex.

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

PortHandle	Port handle
pBuffer	Pointer to write buffer.
BufLen	Length of write buffer
Return:	-1 error, 0 OK

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

PortHandle	Port handle
pBuffer	Pointer to read buffer.
BufLen	Length of read buffer
Return:	-1 error, length of actually read data

3.5 Postcode

The postcode is the string to be transmitted when the transponder is removed.
The postcode consists of 31 characters maximum. End marker is 0xFF Hex.

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

PortHandle	Port handle
pBuffer	Pointer to write buffer.
BufLen	Length of write buffer
Return:	-1 error, 0 OK

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

PortHandle	Port handle
pBuffer	Pointer to read buffer.
BufLen	Length of read buffer
Return:	-1 error, length of actually read data

3.6 Reader Mode Parameter

Read and write the reader mode parameters.

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

PortHandle	Port handle
pBuffer	Pointer to write buffer. See: **3.6.1 Read mode parameter structure**
BufLen	Length of write buffer
Return:	-1 error, 0 OK

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

PortHandle	Port handle
pBuffer	Pointer to read buffer. See: **3.6.1 Read mode parameter structure**
BufLen	Length of read buffer
Return:	-1 error, length of actually read data

3.6.1 Reader mode parameter structure

The parameter for reading and writing are passed always in the same order.

Data 1	**Data 2**	**Data 3**	**Data 4**
Mode	Cycle time	Prompt	Timeout

Mode: Operation mode
0: Send data when transponder goes in field and when transponder leaves field.
1: Send data when prompt is sent.
The prompt is defined in **Prompt**.
The device sends data if the prompt is sent to the device.
3: Send data cyclic
5: Send data when transponder goes in field and when transponder leaves field.
In addition when transponder goes in field the outputs 1 and 2 are set and after the given Cycle time removed. This mode is only at TS-R6x II devices useful, because the TS-R3x devices do not have outputs.

Cycle time: The time is given in 1/10 seconds. The default value is 10 which means 1 second. The cycle time is valid in mode 3 and 5.

Prompt: Character for prompting which is needed only in Mode 1. Default value is '?' (3fH)

Timeout The time is given in 1/10 seconds. The default value is 20 which means 2 seconds. The Timeout is valid in mode 0 and 5. It defines how long a transponder has to be out of field to be recognized again as new transponder.

3.7. Data reception in Reader mode

The data received in reader mode can be accepted in different ways.

3.7.1. cyclic query

The received items can be loaded using multiple calls of the TSLF_RawRead command.
Then the data set has to be connected out of all the received parts. The check for end of data has to be done by the calling instance.

3.7.2. Set up callback function

Using an application specific call back function, the data set can be received and evaluated by the SDK. The complete data set is given to the callback function as one string of data. It is essentially needed, that transmission ends with a unique character. Normally CR = 0DH is used for this.

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

PortHandle	Port handle
TermChar	Terminating character for the transmission. This character triggers the call of the callback function.
pAutoReadProc	pointer to callback function
Return value:	-1 Error, 0: OK

Definition of callback function:
'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
```

The usage of the callback function is described in the "Readermodus" sample application.
The callback function is called with an empty string and Len=0, if the used device is no longer available, for example, if the USB Device is removed during work.

int **TSLF_StopAutoRead**(int PortHandle)

PortHandle	port handle
Return value:	-1 Error, 0: OK

Terminates usage of the callback function.

3.7.3. Set LED and buzzer

This function is only supported at TS-R38/RW38 Version 1.38 or higher version at these devices.

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

PortHandle	port handle
red	-1: do not change red LED
	0: turn off red LED
	1: turn on read LED
	2: let the device control the red LED
green	-1: do not change green LED
	0: turn off green LED
	1: turn on green LED
	2: let the device control the green LED
yellow	-1: do not change yellow LED
	0: turn off yellow LED
	1: turn on yellow LED
	2: let the device control the yellow LED
Return value:	-1 Error, 0: OK

int **TSLF_Beep**(int PortHandle)

PortHandle	port handle
Return value:	-1 Error, 0: OK

Activate the buzzer for 200 mSek. This of course works only if the optional buzzer is available in the connected device.

4. Commands for all transponder types

List of transponder types

Number	Transponder type	Manufacturer	FDX-A	FDX-B	TTF Unique	Comment
0	undefined	-				
2	Hitag 2	NXP		X	X	
3	Q5	Atmel / Sokymat	X	X	X	also ATA5555
4	Hitag S	NXP		X	X	
5	Hitag 1	NXP				
6	Titan 1100	Sokymat				also EM4050, EM4450, EM4550
7	EM4569	Electro Marin		X	X	
8	EM4305	Electro Marin	X	X	X	
9	ATA5577	Atmel	X	X	X	also 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	similar to Hitag S
13 ($0d^{Hex}$)	ATA5551	Atmel	X	X	X	
14 ($0e^{Hex}$)	EL9265	Excelio		X	X	similar to 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 with High Q Factor
38 (26^{Hex})	SIC7999	Silicon Craft				HDX Format

Programming Interface (SDK) of TS-RW devices

4.1. Reset a transponder

int TSLF_ResetTransponder(int PortHandle, int TransponderTyp**)**
PortHandle Port handle
TransponderTyp transponder type from table
Return: -1 Error, 0 OK
Reset the transponder, depending on the transponder type either the Reset command (Q5, AT5577 and Titan) or a sequence with SetRF(0) … SetRF(1) is used. Here also Transpondertyp 0 is allowed. This uses always the SetRF(..) sequence.

4.2. Detection of a transponder type

This function is used to find the type of a transponder.
This tries to get the transponder type because of specific characteristics of the transponder. But there are also cases where the transponder type cannot be detected safely. At HDX Tag types only HDX recognition is supported, return value is always Tiris HDX (33). The different HDX types are not distinguished.

int TSLF_GetTagType(int PortHandle, int TTypExpected)
PortHandle Port handle
TTypExpected Expected transponder type from table
Rückgabewert: -1 Fehler,
0: No transponder detected
>0 detected transponder type

Detection of the answer mode of a transponder

int TSLF_GetTagAnswerMode(int PortHandle, int TTyp, int AnswerModeExpected)
PortHandle Port handle
TTyp transponder type from table
AnswerModeExpected Expected answer mode of the transponder
Rückgabewert: -1 error,
>=0 found answer mode

See also the description to TSLF_SetAnswerMode.
The recognition of the answer mode works only at transponder types which support different answer modes. At all other transponder types the return value is the AnswerModeExpected.

Programming Interface (SDK) of TS-RW devices

4.3. Selecting a transponder

Depending on the type, select the transponder before using read or write functions.

At Hitag 1, Hitag 2, Hitag S and SIC7888 transponder the selection is obligatory.
At Hitag µ transponders access is possible with or without selection.
At Titan, EM4569 und EM4305 transponders a login can be necessary if the transponder is password protected.
At Q5 and ATA5577 transponders special functions have to be used is access is granted in password mode.

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

PortHandle	Port handle
TTyp	transponder type from table
pUID	Pointer to UID to be read with this command.
UIDLen	max. Length of the UID (depending on transponder type)
Return value	-1 error, >0 length of read data

With this command the UID (Unique Identifier) of the transponder is detected.
The UID is detected differently depending on the transponder type. Only one transponder is allowed to be in the antenna field.
The length of the UID depends on transponder type.
At Hitag transponders this function is prescribed before using the TSLF_Select function.

Transponder type	UID Length
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

Transponder type	UID Length
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

Programming Interface (SDK) of TS-RW devices

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

PortHandle	Port handle
TTyp	transponder type from table
pUID	Pointer to UID to be read with this command.
UIDLen	max. Length of the UID (depending on transponder type)
Return value	-1 error, >0 length of read data

Using this command also multiple transponders in field can be found. This function is supported only by Hitag1, HitagS and SIC7888 transponders. The UID of all found transponders are reported. Because each UID has 4 Bytes, the return value is a multiple of 4. There might be not all transponders found in first call. So select the found transponders using TSLF_Select and deactivate using TSLF_Disable. After this call TSLF_UID_RequestNext to get more tags in field.
This function is available with TS-RW38+ Firmware 1.27 and later.

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

PortHandle	Port handle
TTyp	transponder type from table
pUID	Pointer to UID to be read with this command.
UIDLen	max. Length of the UID (depending on transponder type)
Return value	-1 error, >0 length of read data

This command is used to find more tags in fields after TSLF_UID_RequestAC. It is important, that the previously found tags are deactivated using TSLF_Disable, otherwise these tags are found again.
This function is available with TS-RW38+ Firmware 1.27 and later.

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

PortHandle	Port handle
TTyp	transponder type from table
pSelData	pointer to UID or password
SelDataLen	length of UID or. passsword
pConfig	Pointer to Configuration word, which is read by this command
ConfigLen	length of configurations word (fixed 4 Byte)
Return value	-1 error, >0 length of read data

This function is only for Hitag and SIC7888 transponders.
At Hitag1, Hitag S, SIC7888 and Hitag µ in pSelData the UID from TSLF_UID_Request is given, where the length of the UID is 4 Byte at Hitag 1, Hitag S and SIC7888 and 6 Byte at Hitag µ.
At Hitag 2 in pSelData the Tag Password is given.
Das default password is „RKIM“ (52 4B 49 4D).

Programming Interface (SDK) of TS-RW devices

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

PortHandle	Port handle
TTyp	transponder type from table
pInfo	Pointer to Information
InfoLen	length of Information
Return value	-1 error, >0 length of read data

This function is used to get additional information about the chip.
At Hitag µ transponders the „SystemInfo“ of the transponder is read.
At SIC7999 HDX Transponders the chip configuration containing the Traceability Data, Flags, Ctune, Manufacturer Data are read (see SIC7999 data sheet)
Other transponder types are not supported by this function.

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

PortHandle	Port handle
TTyp	transponder type from table
CTuneVal	Tuning value to be used
bUseValue	0: use function with already set value 1: use new tuning value
bOne	0: get frequency for 0-Bits 1: get frequency for 1-Bits
bLock	1: store tuning value in transponder irreversible
pFrequenz	In this variable the measured frequency is returned. Frequency returned in Hz.
Return:	-1 Error, 0 OK

With this function the tuning value is written and the frequency of the data transmission from the transponder is measured. So a very exact alignment of the transponders is possible.
The optimal tuning value has to be computed for every transponder separately by using multiple calls of this function. This function is supported only for SIC7999.

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

PortHandle	Port handle
TTyp	transponder type from table
pFreq0	Frequency for '0' Bits, normally 134600 Hz
pFreq1	Frequency for '1' Bits, normally 124200 Hz
bLock	1: store tuning value in transponder irreversible
return value	-1: error, >0 used tuning value

This function does the complete tuning process, the given frequencies are used for tuning. Result is the used tuning value. Also the real tuned frequencies are returned in pFreq0 and pFreq1.
This function is supported only for SIC7999

Programming Interface (SDK) of TS-RW devices

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

handle	Port handle
TTyp	transponder type from table
pConfig	Pointer to Configuration to be read

This function reads the configuration of a transponder.
This function is supported only for TMS 37190.

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

handle	Port handle
TTyp	transponder type from table
Config	Configuration to be written

This function writes the configuration to a transponder.
This function is supported only for TMS 37190.

The configuration of a TMS 37190 contains the lock state and the Low Q or High Q style.

Config: Configuration byte
- Bit 0-2: IC Life Cycle
  - 011b: Configuration/Production Phase, Chip is changeable
  - 000b: Application Phase, Chip is not changeable
- Bit 3: Q-Factor: (decide depending on tag style)
  - 0: Low Q Factor
  - 1: High Q Factor
- Bit 4: Discharge (shall be set to 0)
  - 0: no discharge
  - 1: discharge after response
- Bit 5-7: RFU (shall be set to 0)

Programming Interface (SDK) of TS-RW devices

int TSLF_Disable (int PortHandle, int TTyp**)**

PortHandle	Port handle
TTyp	transponder type from table

Using this command the Transponder is set inactive.
This command works at EM4569 and EM4305 transponder only if this option if set in the configuration of the transponder. Please respect the data sheet of the transponder.
At Hitag µ transponders the Stay Quiet Command, where the transponder has to be selected is used.
At Hitag S transponders the Quiet Command, where the transponder has to be selected is used.
At Hitag 1 transponders the Halt Command, where the transponder has to be selected is used.
At SIC7888 transponders the Silent Command, where the transponder has to be selected is used.
To reactivate the transponder it has to be reset and then brought to the active state.

int TSLF_WakeUp(int PortHandle, int TTyp, BYTE * pPasswort, int PasswortLen**)**

PortHandle	Port handle
TTyp	transponder type from table
pPasswort	Pointer to password for access to the transponder.
PasswortLen	Length of password (fixed 4 Byte)
Return value	-1 error, 0 OK

Activation of a transponder programmed in AOR Mode. This command can be used only at Q5 and ATA5577 transponders. The transponder has to have a matching configuration. Please respect the data sheet of the transponder.

4.4. Access to password protected transponder

Different transponder types can be password protected. This password protection is implemented in different manners.

4.4.1. Password protection at EM4569, EM4305, Hitag µ, SIC7888 and Titan

At these transponders the access to password protected data can be granted by TSLF_Login.

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

PortHandle	Port handle
TTyp	transponder type from table
pPW	Pointer to password for transponder access.
nPWLen	Length of password (fixed 4 Byte)
Return value	-1 error, 0 OK

The Login function is needed if the transponder is password protected. To get the possibilities of the password protection of the different transponder types please refer to the data sheets of the transponders.
The Login command is supported by the transponder types Titan, EM4569, EM4305, Hitag µ and SIC7888 directly. At the SIC7888 a read only login is made normally. By extending the tag type with 100hex (SIC_LOGIN_RW) the login for read and write is used.
The access with password at Q5 and ATA5577 transponders is activated by the login function. Hereby the password is registered in the SDK and all following read and write access is done as commands with password. The password is lost when the communication port is closed or the transponder is reset using TSLF_ResetTransponder.

The password is stored in a defined data block at most of the transponder types and is stored with the TSLF_Write function.
Therefore mostly a login with the old password has to be made.

One exception is the Titan transponder, at this a separate command is used to set the password, where the old and new password has to be given.

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

PortHandle	Port handle
TTyp	transponder type from table
pOldPW	Pointer to old password
OldPWLen	length of old password (fixed 4 Byte)
pNewPW	Pointer to new password
NewPWLen	length of new password (fixed 4 Byte)
Return value	-1 error, 0 OK

Valid only for Titan transponder!

4.4.2. Password protection at Hitag 2

Hitag 2 transponders can be used in two different operation modes. In password mode, the password protected access is granted with the TSLF_Select command.
In Crypto mode the selection has to be done using TSLF_Authenticate instead f TSLF_Select.

4.4.3. Password protection at Hitag S and EL9265

Normally Hitag S and EL9265 transponders are used without password protection.
It is possible to use Hitag S transponder in “Authentication mode”. If this is used, after TSLF_Select command the transponder has to be authenticated using TSLF_Authenticate.
Once authenticated the transponder access can be done normally.

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

PortHandle	Port handle
TTyp	transponder type from table
pKey	pointer to key used for access to the transponder
nKeyLen	key length (has to be fixed 6 bytes)
pConfig	Pointer to Configuration (Answer to challenge command at HitagS and EL9265, Config Byte / Password Tag at Hitag 2)
nConfigLen	max. length of configuration (fixed 4 Byte)
Return value	-1 error, >0 length of read configuration

This function is only used at Hitag S, EL9265 and Hitag 2. Using Hitag S or EL9265 the calling sequence has to be TSLF_UID_Request, TSLF_Select, TSLF_Authenticate.
At Hitag 2 the calling sequence has to be TSLF_Request, TSLF_Authenticate (without using TSLF_Select).
The key is given LSB first.
Default key for Hitag S. EL9265 and Hitag 2 is 4D 49 4B 52 4F 4E (“MIKRON”)
This function gets the key as input parameter and gives the configuration as output parameter.

4.5. Read and write blocks

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

PortHandle	Port handle
TTyp	transponder type from table
StartBlock	block number
pBuffer	Pointer to transponder data
BufLen	Length of transponder data
Return:	-1 error, >0 length of read data

At this function as much successive blocks are read as given in the length of transponder data. Only complete blocks are read.
If for example block length is 4 Bytes and BufLen = 12 is given, 3 blocks are read, if BufLen = 10 is given, 2 blocks are read and return value is 8.

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

PortHandle	Port handle
TTyp	transponder type from table
StartBlock	block number
pBuffer	Pointer to transponder data
BufLen	Length of transponder data
Lock	0 = Default. Only used for **Q5** and **ATA5577** transponder. If this value is 1, this block of the Q5 resp. ATA5577 transponder is locked.
Return:	-1 error, 0 OK

Only complete blocks are written, if the length is not a multiple of block length only up to the last completely filled block is written.

The TTyp can be concatenated with the option: **“read after write”**. This option causes an immediate read after write. Followed by comparing the read and written data. If the compare is negative, the function returns with error. GetLastError gives error number 20 (see appendix)
Especially with the Q5 it is possible to read inverse data. In this case use **“read after write invers”**

Read after write = 100 Hex Read after write invers = 200 Hex

At Transponder type EM4305 the protection word is stored in the blocks 14 and 15, where it has to be distinguished out of the read data which block has the valid word. Writing of the protection word can be done only with a special command which is used automatically if writing to block 14 or 15 is done.
At Transponder type ATA5577 the access to page 1 is done using block number 8 – 11.

4.6. Formatted writing

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

PortHandle	Port handle
TTyp	transponder type from table
pBuffer	Pointer to transponder data
BufLen	Length of transponder data (must be fixed 5 bytes)
Lock	0 = Default. If this value is 1, the blocks in the transponder are locked.
Return:	-1 error, 0 OK

Using this function transponders are written in Unique-Format (EM4102 compatible) .
The function can be used with Q5, Hitag2, Hitag S, Hitag µ, EM4569, EM4305, SIC7888, EL9265, ATA5577 and ATA5575 Transponders. The transponder is automatically configured as described in the specification. (64 Bit = 2 Blocks Transponder Talks First in Manchester Modulation, 2kBit). If the transponder type is supported depends on the device.

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

PortHandle	Port handle
TTyp	transponder type from table
pBuffer	Pointer to transponder data
BufLen	Length of transponder data (must be fixed 11 bytes)
Lock	0 = Default. If this value is 1, the blocks in the transponder are locked.
Return:	-1 error, 0 OK

Using this function transponders are written in UniqueXL-Format.
The function can be used with Q5, Hitag2, Hitag S, Hitag µ, EM4569, EM4305, SIC7888, EL9265, ATA5577 and ATA5575 Transponders. The transponder is automatically configured as described in the specification. (128 Bit = 4 Blocks Transponder Talks First in Manchester Modulation, 2kBit). If the transponder type is supported depends on the device.

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

PortHandle	Port handle
TTyp	transponder type from table
pBuffer	Pointer to transponder data
BufLen	Length of transponder data (must be fixed 5 bytes)
Lock	0 = Default. If this value is 1, the blocks in the transponder are locked.
Return:	-1 error, 0 OK

Using this function transponders are written in Unique-data format but with Manchester 4kBit speed.
This function is supported only by TS-RW38Plus and TS-RW38AC devices.
Supported transponder types are Q5, ATA5577, EM4569 and EM4305

Programming Interface (SDK) of TS-RW devices

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

PortHandle	Port handle
TTyp	transponder type from table
pBuffer	Pointer to transponder data
BufLen	Length of transponder data (must be fixed 2 bytes)
Lock	0 = Default. If this value is 1, the blocks in the transponder are locked.
Return:	-1 error, 0 OK

Using this function transponders are written in GiS 16Bit Format.
The function can be used with Q5, Hitag2, Hitag S, Hitag µ, EM4569, EM4305, SIC7888, EL9265, ATA5577 and ATA5575 Transponders. The transponder is automatically configured as described in the specification. (64 Bit = 2 Blocks Transponder Talks First in Manchester Modulation, 2kBit). If the transponder type is supported depends on the device.

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

PortHandle	Port handle
TTyp	transponder type from table
pBuffer	Pointer to transponder data
BufLen	Length of transponder data (must be fixed 2 bytes)
Lock	0 = Default. If this value is 1, the blocks in the transponder are locked.
Return:	-1 error, 0 OK

Using this function transponders are written in GiS 12Bit Format.
The function can be used with Q5, Hitag2, Hitag S, Hitag µ, EM4569, EM4305, SIC7888, EL9265, ATA5577 and ATA5575 Transponders. The transponder is automatically configured as described in the specification. (64 Bit = 2 Blocks Transponder Talks First in Manchester Modulation, 2kBit). If the transponder type is supported depends on the device.

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

PortHandle	Port handle
TTyp	transponder type from table
pBuffer	Pointer to transponder data
BufLen	Length of transponder data (must be fixed 9 bytes)
Lock	0 = Default. If this value is 1, the blocks in the transponder are locked.
Return:	-1 error, 0 OK

Using this function transponders are written in GiS 72Bit-Format. The function can be used only with Hitag S Transponders. The transponder will be also configured with this function as described in the specification. (128 Bit = 4 Blocks Transponder Talks First in Manchester Modulation, 2kBit) If this format is supported depends on the device.

Programming Interface (SDK) of TS-RW devices

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

PortHandle	Port handle	
TTyp	transponder type from table	
FacilityCode	Facility Code 0 - 255	
CustomerNr	Customer number 0 - 65535	
Lock	0 = no Lock	1 = lock data area
	2 = lock configuration	3 = lock configuration and data.
Return:	-1 error, 0 OK	

With this function the transponder is written in FSK26Bit Format.
This function is supported only by TS-RW38Plus and TS-RW38AC devices.
Supported transponder types are Q5, ATA5577 and SIC7888

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

PortHandle	Port handle	
TTyp	transponder type from table	
FacilityCode	Facility Code 0 - 65535	
CustomerNr	Customer number 0 - 65535	
Lock	0 = no Lock	1 = lock data area
	2 = lock configuration	3 = lock configuration and data.
Return:	-1 error, 0 OK	

With this function the transponder is written in FSK34Bit Format.
This function is supported only by TS-RW38Plus and TS-RW38AC devices.
Supported transponder types are Q5, ATA5577 and SIC7888

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

PortHandle	Port handle	
TTyp	transponder type from table	
FacilityCode	Facility Code 0 - 65535	
CustomerNr	Customer number 0 - 65535	
Lock	0 = no Lock	1 = lock data area
	2 = lock configuration	3 = lock configuration and data.
Return:	-1 error, 0 OK	

With this function the transponder is written in NC34Bit Format.
Difference between FSK34Bit and NC34Bit Format is in calculation of parity only.
This function is supported only by TS-RW38Plus and TS-RW38AC devices.
Supported transponder types are Q5, ATA5577 and SIC7888

Programming Interface (SDK) of TS-RW devices

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

PortHandle	Port handle	
TTyp	transponder type from table	
FacilityCode	Facility Code 0 - 4095	
CustomerNr	Customer number 0 - 1048575	
Lock	0 = no Lock	1 = lock data area
	2 = lock configuration	3 = lock configuration and data.
Return:	-1 error, 0 OK	

With this function the transponder is written in FSK35Bit Format.
This function is supported only by TS-RW38Plus and TS-RW38AC devices.
Supported transponder types are Q5, ATA5577 and SIC7888

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

PortHandle	Port handle	
TTyp	transponder type from table	
FacilityCode	Facility Code 0 - 255	
CustomerNr	Customer number 0 – 65535	
FixedField	Fixed field 0 - 1023	
Lock	0 = no Lock	1 = lock data area
	2 = lock configuration	3 = lock configuration and data.
Return:	-1 error, 0 OK	

With this function the transponder is written in FSK36Bit Format.
This function is supported only by TS-RW38Plus and TS-RW38AC devices.
Supported transponder types are Q5, ATA5577 and SIC7888

Programming Interface (SDK) of TS-RW devices

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

PortHandle	Port handle	
TTyp	transponder type from table	
FacilityCode	Facility Code 0 - 65535	
CustomerNr	Customer number 0 - 524287	
Lock	0 = no Lock	1 = lock data area
	2 = lock configuration	3 = lock configuration and data.
Return:	-1 error, 0 OK	

With this function the transponder is written in FSK37Bit Format.
This function is supported only by TS-RW38Plus and TS-RW38AC devices.
Supported transponder types are Q5, ATA5577 and SIC7888

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

PortHandle	Port handle	
TTyp	transponder type from table	
pCode	Pointer to 37 Bit Huge Code 0 – 34359738367	
CodeLen	Length of pCode in Bytes (5)	
Lock	0 = no Lock	1 = lock data area
	2 = lock configuration	3 = lock configuration and data.
Return:	-1 error, 0 OK	

With this function the transponder is written in FSK37Bit Huge Format.
This function is supported only by TS-RW38Plus and TS-RW38AC devices.
Supported transponder types are Q5, ATA5577 and SIC7888

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

PortHandle	Port handle	
TTyp	transponder type from table	
FacilityCode	Facility Code 0 - 8191	
CustomerNr	Customer number 0 – 131071	
FixedField	Fixed field 0 - 31	
Lock	0 = no Lock	1 = lock data area
	2 = lock configuration	3 = lock configuration and data.
Return:	-1 error, 0 OK	

With this function the transponder is written in IdentiCard 37Bit Format.
This function is supported only by TS-RW38Plus and TS-RW38AC devices.
Supported transponder types are Q5, ATA5577 and SIC7888

Programming Interface (SDK) of TS-RW devices

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

PortHandle	Port handle	
TTyp	transponder type from table	
FacilityCode	Facility Code 0 - 255	
CustomerNr	Customer number 0 - 65535	
Lock	0 = no Lock	1 = lock configuration and data.
Return:	-1 error, 0 OK	

With this function the transponder is written in AWID 26 Bit Format.
This function is supported only by TS-RW38Plus and TS-RW38AC devices.
Supported transponder types are Q5, ATA5577 and SIC7888

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

PortHandle	Port handle	
TTyp	transponder type from table	
FacilityCode	Facility Code 0 - 1023	
CustomerNr	Customer number 0 - 268435455	
Lock	0 = no Lock	1 = lock configuration and data.
Return:	-1 error, 0 OK	

With this function the transponder is written in AWID 40 Bit Format.
This function is supported only by TS-RW38Plus and TS-RW38AC devices.
Supported transponder types are Q5, ATA5577 and SIC7888

4.7. Formatted read

With the commands of this chapter it is possible to read transponders in TTF mode.
Because the transponder data is the same with all transponders which are able to operate in TTF mode, no transponder type is given. Because it is not allowed that the transponder is selected when this command is used, the antenna field is turned of and on inside of this function.

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

PortHandle	Port handle	
AnswerMode	Answer mode	
	0= Manchester	2,0 kBit Data Rate
	1= Biphase	4,0 kBit Data Rate
	2= FSK2a	2,5 kBit Data Rate
pBuffer	Pointer to transponder data	
BufLen	Length of expected transponder data	
Return:	-1 error, >0 length of read data	

The transmitted data is not justified, because the data stream at TTF is continually. To classify the data, the data stream has to be shifted to the right condition depending of the used data format.

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

PortHandle	Port handle
pBuffer	Pointer to transponder data
BufLen	Length of transponder data (must be fixed 5 bytes)
Return:	-1 error, >0 length of read data

This command reads the Unique data stream. The transponder has to be formatted and written in Unique format.

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

PortHandle	Port handle
pBuffer	Pointer to transponder data
BufLen	Length of transponder data (must be fixed 11 bytes)
Return:	-1 error, >0 length of read data

This command reads the Unique XL data stream. The transponder has to be formatted and written in Unique XL format.

Programming Interface (SDK) of TS-RW devices

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

PortHandle	Port handle
pBuffer	Pointer to transponder data
BufLen	Length of transponder data (must be fixed 5 bytes)
Return:	-1 error, >0 length of read data

This command reads the Fast Unique data stream. The transponder has to be formatted and written in Fast Unique format.
This function is only supported at TS-RW38Plus and TS-RW38AC.

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

PortHandle	Port handle
pBuffer	Pointer to transponder data
BufLen	Length of transponder data (must be fixed 2 bytes)
Return:	-1 error, >0 length of read data

This command reads the „GiS 16 Bit Format“ data stream. The transponder has to be formatted and written in „GiS 16 Bit Format“.

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

PortHandle	Port handle
pBuffer	Pointer to transponder data
BufLen	Length of transponder data (must be fixed 2 bytes)
Return:	-1 error, >0 length of read data

This command reads the „GiS 12 Bit Format“ data stream. The transponder has to be formatted and written in „GiS 12 Bit Format“.

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

PortHandle	Port handle
pBuffer	Pointer to transponder data
BufLen	Length of transponder data (must be fixed 9 bytes)
Return:	-1 error, >0 length of read data

This command reads the „GiS 72 Bit Format“ data stream. The transponder has to be formatted and written in „GiS 72 Bit Format“.

Programming Interface (SDK) of TS-RW devices

int TSLF_Read_FSK26Bit (int PortHandle, DWORD *pFacilityCode, DWORD *pCustomerNr**)**

PortHandle	Port handle
pFacilityCode	Pointer to Facility Code 0 - 255
pCustomerNr	Pointer to Customer Number 0 - 65535
Return:	-1 error, 0 OK

With this function Transponder in FSK26Bit Format are read.
This function is only supported at TS-RW38Plus and TS-RW38AC.

int TSLF_Read_FSK34Bit (int PortHandle, DWORD *pFacilityCode, DWORD *pCustomerNr**)**

PortHandle	Port handle
pFacilityCode	Pointer to Facility Code 0 - 65535
pCustomerNr	Pointer to Customer Number 0 - 65535
Return:	-1 error, 0 OK

With this function Transponder in FSK34Bit Format are read.
This function is only supported at TS-RW38Plus and TS-RW38AC.
If invalid checksum is found, Return value is -1, TSLF_GetLastError() returns TSLF_CHECKSUMMENFEHLER (14) but Facility code and Customer number are set anyway.

int TSLF_Read_NC34Bit (int PortHandle, DWORD *pFacilityCode, DWORD *pCustomerNr**)**

PortHandle	Port handle
pFacilityCode	Pointer to Facility Code 0 - 65535
pCustomerNr	Pointer to Customer Number 0 - 65535
Return:	-1 error, 0 OK

With this function Transponder in NC34Bit Format are read.
Difference between FSK34Bit and NC34Bit Format is in calculation of parity only.
This function is only supported at TS-RW38Plus and TS-RW38AC.
If invalid checksum is found, Return value is -1, TSLF_GetLastError() returns TSLF_CHECKSUMMENFEHLER (14) but Facility code and Customer number are set anyway.

int TSLF_Read_FSK35Bit (int PortHandle, DWORD *pFacilityCode, DWORD *pCustomerNr**)**

PortHandle	Port handle
pFacilityCode	Pointer to Facility Code 0 - 1023
pCustomerNr	Pointer to Customer Number 0 - 1048575
Return:	-1 error, 0 OK

With this function Transponder in FSK34Bit Format are read.
This function is only supported at TS-RW38Plus and TS-RW38AC.

Programming Interface (SDK) of TS-RW devices

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

PortHandle	Port handle
pFacilityCode	Pointer to Facility Code 0 - 255
pCustomerNr	Pointer to Customer Number 0 – 65535
pFixedField	Pointer to Fixed field 0 - 1023
Return:	-1 error, 0 OK

With this function Transponder in FSK36Bit Format are read.
This function is only supported at TS-RW38Plus and TS-RW38AC.

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

PortHandle	Port handle
pFacilityCode	Pointer to Facility Code 0 - 65535
pCustomerNr	Pointer to Customer Number 0 - 524287
Return:	-1 error, 0 OK

With this function Transponder in FSK37Bit Format are read.
This function is only supported at TS-RW38Plus and TS-RW38AC.

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

PortHandle	Port handle
pCode	Pointer to Code 0 - 34359738367
CodeLen	Length of Code in Byte (5)
Return:	-1 error, >0 Length of read Data

With this function Transponder in FSK37Bit Huge Format are read.
This function is only supported at TS-RW38Plus and TS-RW38AC.

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

PortHandle	Port handle
pFacilityCode	Pointer to Facility Code 0 - 8191
pCustomerNr	Pointer to Customer Number 0 – 131071
pFixedField	Pointer to Fixed field 0 - 31
Return:	-1 error, 0 OK

With this function Transponder in IdentiCard 37Bit Format are read.
This function is only supported at TS-RW38Plus and TS-RW38AC.

Programming Interface (SDK) of TS-RW devices

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

PortHandle	Port handle
pFacilityCode	Pointer to Facility Code 0 - 255
pCustomerNr	Pointer to Customer Number 0 - 65535
Return:	-1 error, 0 OK

With this function Transponder in AWID 26 Bit Format are read.
This function is only supported at TS-RW38Plus and TS-RW38AC.

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

PortHandle	Port handle
pFacilityCode	Pointer to Facility Code 0 - 1023
pCustomerNr	Pointer to Customer Number 0 - 268435455
Return:	-1 error, 0 OK

With this function Transponder in AWID 40 Bit Format are read.
This function is only supported at TS-RW38Plus and TS-RW38AC.

4.8. Access to transponders with temperature measurement

The following functions are only for transpoders with integrated temperature measurement available.
Only tag type EL9265 is supported.
Only TS-RW38+ and TS-RW38AC each with V1.50 and above support these functions.

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

PortHandle	Port handle
TTyp	transponder type from table
pSensorData	pointer to sensor data from tag.
Return value:	-1 Error, 0: OK

A temperature measurement is triggered in the transponder (GET TEMP) and the sensor value is reported. The temperature value can be calculated from sensor data using the calibration data. Please refer to the data sheet of EN9265.
The tag is automatically selected before reading.

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

PortHandle	Port handle
TTyp	transponder type from table
TimingInfo	Additional information to be stored with sensor data in the tag.
Page	page number to store sensor data. (pages 8 – 57 are allowed)
pSensorData	pointer to sensor data from tag.
Return value:	-1 Error, 0: OK

A temperature measurement is triggered in the transponder (EN INSEN) and the sensor value is reported. The sensor value is stored in the given page in the tag together with TimingInfo Temperature value can be calculated from sensor data using the calibration data. Please refer to the data sheet of EN9265.
The tag is automatically selected before reading.

Programming Interface (SDK) of TS-RW devices

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

PortHandle	Port handle
TTyp	transponder type from table
pTemp	Pointer to temperature value in $^{1}/_{65536}$ °C
Return value:	-1 Error, 0 OK

A temperature measurement is triggered in the transponder (GET TEMP). Using the sensor data reported, the temperature value is calculated using the calibration data as described in data sheet of EN9265. The returned temperature has to be divided by 65536 to get the value in °C.
The tag is automatically selected before reading.

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

PortHandle	Port handle
TTyp	transponder type from table
SensorData	Sensordaten die aus einer früheren Temperaturmessung stammen
Return value::	-1 Error, Temperature value in $^{1}/_{65536}$ °C

The sensor value as reported by TSLF_GetTempSensor or by TSLF_Read_FDXB in pExtension (see also TSLF_ActivateTempSensor) is given, using the calibration data of the tag the temperature is calculated and returned.
The tag has to be in field so the calibration data can be read.
The return value has to be divided by 65536 to get the value in °C.
The tag is automatically selected before reading.

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

PortHandle	Port handle
TTyp	transponder type from table
Page	block number to be used (8-57), 0: deactivate
Return value:	-1 Error, 0 OK

At EL9265 chip it is possibl to return a stored temperature value together with the Animal Code Data (TTF_MODE = 01b). Using this function this mode is activated. Of 0 is given as page number this function is deactivated. Please refer to the data sheet of EN9265.
The tag is automatically selected before reading.

4.9. Access to Transponder Type EM4582

The following functions are only available for Transponder of type EM4582.
Only TS-RW380+ and TS-RW380AC with Version 1.04 and higher support these functions.
To operate with the functions knowledge of EM4582 data sheet is obligated.
The access to read and write the blocks of EM4582 is done using the standard TSLF_Read and TSLF_Write commands.

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

PortHandle	port handle
pData	user data stored with value
pVal	last recorded value
Return value:	-1 Error, 6 data received

The last with TSLF_EM4582_Increment recorded number is read.

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

PortHandle	port handle
pKey	pointer to key
KeyLen	length of key
Data	user data
Val	number
Return value:	-1 Error, 0 successfully written

A number with user data is stored in the transponder using the WRITE INCREMENT command.
This is only possible if the number is larger than the last recorded number. In pKey the key SKI (as stored in the transponder) is given with LSB first.
KeyLen gives the length of the key.
If the key is also stored inside of TS-RW380 you can skip the key here. If pKey = 0 or KeyLen = 0, the TS-RW380 inside key is used.
If the pKey shall be used it has to have a length of 128 Bit (16 Byte).

Programming Interface (SDK) of TS-RW devices

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

PortHandle	port handle
pID	UID of transponder, LSB first
IDLen	Length of UID of transponders (always 4 Byte)
pPwd	Password, LSB first
PWLen	Length of Password (always 4 Byte)
Return value:	-1 Error, 0: Success

The SEND ACCESS command is executed. The UID of the transponder can be determined with TSLF_UID_Request. The SEND ACCESS command allows unlocking of registers and memory areas protected with LB0.

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

PortHandle	port handle
pID	UID of transponder, LSB first
IDLen	Length of UID of transponders (always 4 Byte)
pPwd	Password, LSB first
PWLen	Length of Password (always 4 Byte)
Return value:	-1 Error, 0: Success

The UNLOCK KEY commando is executed. The UID of the transponder can be determined with TSLF_UID_Request. The UNLOCK KEY command allows the unlocking of a Secret Key for modification by providing the actual Secret key and the chip UID.

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

PortHandle	port handle
pRN	pointer to random number, has to have space for 16 Byte (128 Bit).
Return value:	-1 Error, >0 length of random number

The GET RN command is used to generate a random number using the embedded True Random Number Generator (TRNG). The random number is stored in pRN. The length of the random number depends on the settings of the transponder.

Programming Interface (SDK) of TS-RW devices

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

PortHandle	port handle
pRN	random number
RNLen	length of random number
pTokenR	$TOKEN_{R \rightarrow T}$
TokenRLen	length of $TOKEN_{R \rightarrow T}$
pTokenT	$TOKEN_{T \rightarrow R}$
MaxTokenTLen	maximum length of $TOKEN_{T \rightarrow R}$
Return value:	-1 Error, > 0 Length of $TOKEN_{T \rightarrow R}$

SingleAuth does the authentication of the transponder using the SINGLE AUTHENTICATION command. Length of random number and token depends on the setting of the transponder. See the transponder datasheet for exact description.

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

PortHandle	port handle
pRN	random number
RNLen	length of random number
pTokenR	$TOKEN_{R \rightarrow T}$
TokenRLen	length of $TOKEN_{R \rightarrow T}$
pTokenT	$TOKEN_{T \rightarrow R}$
MaxTokenTLen	maximum length of $TOKEN_{T \rightarrow R}$
Return value:	-1 Error, > 0 Length of $TOKEN_{T \rightarrow R}$

MutualAuth does the authentication of the transponder using the MUTUAL AUTHENTICATION command. Length of random number and token depends on the setting of the transponder. See the transponder datasheet for exact description.

Programming Interface (SDK) of TS-RW devices

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

PortHandle	port handle
pTokenR	$TOKEN_{R \rightarrow T}$
TokenRLen	length of $TOKEN_{R \rightarrow T}$
pTokenT	$TOKEN_{T \rightarrow R}$
MaxTokenTLen	maximum length of $TOKEN_{T \rightarrow R}$
Return value:	-1 Error, > 0 Length of $TOKEN_{T \rightarrow R}$

MutualAuthIso does the authentication of the transponder using the ISO MUTUAL AUTHENTICATION command. Length of the tokens depends on the setting of the transponder. See the transponder datasheet for exact description.

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

PortHandle	port handle
RNTokenTyp	Setting for length of random numbers and tokens see Bit 0 – 7 of Crypt Configuration Registers in datasheet for description.
pKey	pointer to key
KeyLen	length of key.
Return value:	-1 Error, 0 Success

The full SINGLE AUTHENTICATION process is done.
The key SKA is given in pKey with LSB first.
The length of the key is stored in KeyLen.
If the key is also stored inside of TS-RW380 you can skip the key here. If pKey = 0 or KeyLen = 0, the TS-RW380 inside key is used.
If pKey shall be used it has to have a length of 128 Bit (16 Byte).
This function eases the usage of SINGLE AUTHENTICATION, so in the calling software no AES encryption has to be done. But because of this the Key is transmitted in clear text, which might give a security breach.
By storing the key in TS-RW380 this can be avoided.

Programming Interface (SDK) of TS-RW devices

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

PortHandle	port handle
RNTokenTyp	Setting for length of random numbers and tokens see Bit 0 – 7 of Crypt Configuration Registers in datasheet for description.
pKey	pointer to key
KeyLen	length of key.
Return value:	-1 Error, 0 Success

The full MUTUAL AUTHENTICATION process is done.
The key SKA is given in pKey with LSB first.
The length of the key is stored in KeyLen.
If the key is also stored inside of TS-RW380 you can skip the key here. If pKey = 0 or KeyLen = 0, the TS-RW380 inside key is used.
If pKey shall be used it has to have a length of 128 Bit (16 Byte).
This function eases the usage of MUTUAL AUTHENTICATION, so in the calling software no AES encryption has to be done. But because of this the Key is transmitted in clear text, which might give a security breach.
By storing the key in TS-RW380 this can be avoided.

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

PortHandle	port handle
RNTokenTyp	Setting for length of random numbers and tokens see Bit 0 – 7 of Crypt Configuration Registers in datasheet for description.
pKey	pointer to key
KeyLen	length of key.
Return value:	-1 Error, 0 Success

The full ISO MUTUAL AUTHENTICATION process is done.
The key SKA is given in pKey with LSB first.
The length of the key is stored in KeyLen.
If the key is also stored inside of TS-RW380 you can skip the key here. If pKey = 0 or KeyLen = 0, the TS-RW380 inside key is used.
If pKey shall be used it has to have a length of 128 Bit (16 Byte).
This function eases the usage of ISO MUTUAL AUTHENTICATION, so in the calling software no AES encryption has to be done. But because of this the Key is transmitted in clear text, which might give a security breach.
By storing the key in TS-RW380 this can be avoided.

Programming Interface (SDK) of TS-RW devices

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

PortHandle	port handle
pKey	pointer to key
KeyLen	length of key. (always 16 Byte)
Return value:	-1 Error, 0 Success

Storing the key value for SKA in TS-RW380. So the key can be skipped at the **…AuthDirect** functions.

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

PortHandle	port handle
pKey	pointer to key
KeyLen	length of key. (always 16 Byte)
Return value:	-1 Error, 0 Success

Storing the key value for SKI in TS-RW380.
So the key can be skipped at **TSLF_EM4582_Increment**.

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

PortHandle	port handle
pKey	pointer to key
KeyLen	length of key. (always 12 Byte)
Return value:	-1 Error, 0 Success

Storing the value for Expansion Number Transponder E_T in TS-RW380.
This is needed at the **…AuthDirect** functions to calculate the token.

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

PortHandle	port handle
pKey	pointer to key
KeyLen	length of key. (always 4 Byte)
Return value:	-1 Error, 0 Success

Storing the value for Expansion Number Reader E_R in TS-RW380.
This is needed at …**AuthDirect** functions to calculate the token.

5. Animal (FDX-A, FDX-B, HDX) or Waste (EN14803) Tags

These functions are possible only with devices that support animal coding.

5.1. Read and write Animal tags (FDX-A, FDX-B, HDX)

FDX-A Format short description
The FDX-A format includes a 10 digits number, which is defined in OktHex coding.
This means that each of the 5 Bytes can have a range of 00 – $7F^{Hex}$.
Mostly only the decimal values are used in each byte.

FDX-B Format short description
The animal (FDX-B) tag has 128 Bit data. The data structure is described in ISO11785.
Herein the identification code is included. The structure of the identification code is described in ISO11784.
As Extension to ISO11784 the usage of the reserved part is fixed for the EU with the following meaning:

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

The Extension part described in ISO11785 has a length of 24 Bit and is given as Extension.

HDX Format short description
The animal (HDX) tag has 80 Bit data. The data structure is described in ISO11785.
Herein the identification code is included. The structure of the identification code is described in ISO11784.
As Extension to ISO11784 the usage of the reserved part is fixed for the EU with the following meaning:

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

The Extension part described in ISO11785 has a length of 24 Bit and is given as Extension.

Programming Interface (SDK) of TS-RW devices

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

PortHandle	Port handle
pIdNumber	Identification number (5 Byte length as defined by FDX-A specification)
pIdLen	Length of data area for identification number (has to be at least 5 Bytes)
Return:	-1 error, >0 length of read data in pIdNumber

With this function animal FDX-A tags can be read.
Attention: This function is only supported at TS-RW38AC and TS-RW68 AC.

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

PortHandle	Port handle
pCountryCode	Country Code as ISO 3166
pBuffer	Pointer to ID Number (1 – 274.877.906.944)
BufLen	Length of data (must be fixed 8 bytes)
pAnimalFlag	Pointer to animal flag
pDatenblockFlag	Pointer to data block flag
pReserved	pointer to reserved part
pExtension	pointer to extension part
Return:	-1 error, , >0 length of read data

With this function animal FDX-B tags can be read.

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

PortHandle	Port handle
pCountryCode	Country Code as ISO 3166
pBuffer	Pointer to ID Number (1 – 274.877.906.944)
BufLen	Length of data (must be fixed 8 bytes)
pAnimalFlag	Pointer to animal flag
pDatenblockFlag	Pointer to data block flag
pReserved	pointer to reserved part
pExtension	pointer to extension part
Return:	-1 error, , >0 length of read data

With this function animal HDX tags can be read.
Attention: The HDX Format is only supported by TS-HW38 AC and TS-RW68 AC!

Programming Interface (SDK) of TS-RW devices

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

PortHandle	Port handle
TTyp	transponder type from table
pFactoryID	The FactoryID (UID) of the selected transponder. Depending on the transponder type, this has a different value range. Memory space for the FactoryID has to be 8 bytes.
pIdNumber	identification number (5 Byte length as defined in FDX-A specification)
pIdLen	length of data area for identification number (has to be 5 bytes)
Lock	0 = Default. If this value is 1, the blocks of the transponder are locked.
Rückgabewert	-1 Error, 0 Successfully written 2 Transponder detected, the transponder is write protected and could not be written. 3 Transponder detected, the transponder has same factory ID as committed, the transponder was not written. 4 Process Pending, see next page!

With this function a transponder is selected, if necessary the configuration is changed (formatted) and after this the data is written and if selected the tag is write protected.
In pFactoryID the UID of the written transponder is returned.
To avoid that the same transponder is written multiple times, in pFactoryID the last used UID can be given. With this it will be checked if the transponder has this UID and then return value 3 will be given.
The writing of the blocks is repeated up to 5 times if a writing error occurs.

Programming Interface (SDK) of TS-RW devices

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	Port handle
TTyp	transponder type from table
pFactoryID	FactoryID (UID) of the selected transponder. Depending on the transponder type, this has a different value range. The length of the FactoryID must be 8 Byte.
CountryCode	Country Code as ISO 3166
pIDNummer	Pointer to ID-Number (1 – 274.877.906.944)
IDNummerLen	Length of data (must be fixed 8 bytes)
AnimalFlag	animal flag 0: Industrial, 1: Animal
DatenblockFlag	0: no Data Block, 1: Data block available
Reserved	reserved part (14 Bit)
Extension	Extension part (24 Bit)
Lock	0 = Default. If this value is 1, the blocks are locked.
Return:	-1 Error. 0 Successfully written 2 Transponder detected, the transponder is write protected and could not be written. 3 Transponder detected, the transponder has same factory ID as committed, the transponder was not written. 4 Process Pending, see next page!

With this function the transponder is selected, if necessary the configuration is changed (formatted) and then the data is written and if selected write protected.
In pFactoryID is the UID of the written transponders returned.
To avoid that the same transponder is written multiple times, in pFactoryID the last used UID can be given. Then it will check if the selected transponder has this UID and then return value 3 is returned.
The writing of the blocks is repeated up to 5 times if a writing error occurs.
By giving the value NO_RETRY = 1000Hex = 4096 Dez. in addition to the Transponder type the write retry can be suppressed.
By giving the value READ_AFTER_WRITE = 100Hex = 256 Dez. in addition to the Transponder type a read check will be made at each block and if an error occurs the block will be rewritten automatically. With this the failure rate can be lowered again, but the duration will be increased. The additional reading of block needs about 100 mSec.
By giving the value NO_READ_AFTER_WRITE = 4000Hex = 16384 Dez in addition to the Transponder type the TTF read check after write completion is skipped to increase speed.
By giving the value NO_FACTORY_ID = 8000Hex = 32768 Dez in addition to the Transponder type the reading of the factory Id can be skipped. Depending on the Tag Type the reading of the factory Id is obligatory, so it should not be skipped.

Programming Interface (SDK) of TS-RW devices

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	Port handle
TTyp	transponder type from table
pFactoryID	FactoryID (UID) of the selected transponder. Depending on the transponder type, this has a different value range. The length of the FactoryID must be 8 Byte.
CountryCode	CountryCode as ISO 3166
pIDNummer	Pointer to ID-Number (1 – 274.877.906.944)
IDNummerLen	Length of data (must be fixed 8 bytes)
AnimalFlag	animal flag 0: Industrial, 1: Animal
DatenblockFlag	0: no Data Block, 1: Data block available
Reserved	Reserved part (14 Bit)
Extension	Extension part (24 Bit)
Lock	0 = Default. If this value is 1, the blocks are locked.
Return:	-1 Error. 0 Successfully written 2 Transponder detected, the transponder is write protected and could not be written. 3 Transponder detected, the transponder has same factory ID as committed, the transponder was not written. 4 Process Pending, see next page!

With this function a HDX Transponder is written. Only HDX Transponder types are supported here. It is to be respected, that HDX Transponders are mostly only functioning as Animal tags if they are locked. Some HDX Transponders are only one time programmable.
At most HDX transponder types the Lock Flag has to be set so they are written correctly to be read also with other reading devices.

Attention: The HDX Format is only supported by TS-HW38 AC and TS-RW68 AC!

Programming Interface (SDK) of TS-RW devices

Writing as Background task
Because the functions **TSLF_Write_FDXA**, **TSLF_Write_FDXB** and **TSLF_Write_HDX** need more time (about 500 – 700 mSec) this function can be called in "Overlapped Mode" also. For this the value of DO_OVERLAPPED = 2000Hex = 8192 Dez. is added to TTyp. With this the function will be finished immediately and return value will be TSLF_RETVAL_PROCESS_PENDING = 4.

The function
int TSLF_IsWriteFinished(int PortHandle, BYTE * pFactoryID**)**

PortHandle	Port handle
pFactoryID	FactoryID (UID) of the selected transponder. Depending on the transponder type, this has a different value range. The length of the FactoryID must be 8 Byte.

is used to check if the write process is finished.

The return values of this function are the same as at TSLF_Write_FDXA and TSLF_Write_FDXB described. Also in this case the Factory ID is only valid when the return value has reached 0 (successfully written). While the return value is TSLF_RETVAL_PROCESS_PENDING = 4, the write process is pending.
With qualified programming of the application it is possible with this to program multiple transponders at multiple connected devices simultaneously.
It has to be ensured that the distance between the antennas is big enough to avoid disturbance.

Programming Interface (SDK) of TS-RW devices

5.2. Usage of commands for FDX-A resp. FDX-B Transponders:

1. Preparation:
First of all, the device interface is opened with command **TSLF_Open(….)**.
With command **TSLF_SetFilter(…)** the filter value for the selected transponder type is set. This value is saved in the device, so it is only necessary to set the value if the transponder type is changed. Setting the filter is not needed at TS-RW38 devices, because this devices have built in filter settings for all transponder types.

2. Writing to the transponder
This can be done comfortably with **TSLF_Write_FDXA(..)** resp. **TSLF_Write_FDXB(…)**. This functions checks first if a transponder is found, formats the transponder if needed and writes the desired data. Because this function also signals if no tag is in field, if can also be used to recognize and write transponders automatically.

3. Reading the transponder
The ISO11784 data is read with command **TSLF_Read_FDXA(…)** resp. **TSLF_Read_FDXB(…)**. This function does only give a positive result, if the found transponder is correctly formatted and the data is valid! New transponders will always return -1 wit error code "**21** no Answer"

4. Reader with multiplexed antenna (TS-W80-AC 8x)
At these readers multiple antennas are used alternating. Switching between antennas is done through the command **TSLF_SetReaderAdresse(…)** The device acts as if 8 readers are connected simultaneously with one antenna each.

Programming Interface (SDK) of TS-RW devices

5.3. Read and write Waste tags (EN14803)

The Waste tag (EN14803) has 128 Bit data. The data structure is described in ISO11785.
Herein the identification code is included. The structure of the identification code is described in EN14803.

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

PortHandle	Port handle
pManufacturer	Manufacturer code
pSerialNumber	Serial number
Return:	-1 error, 0 successfully read, 5 Invalid data in tag

With this function waste tags can be read.
The Data stream is interpreted by EN14803 and ISO 11785.
The TSLF_Read_EN14803HDX function is to read HDX Waste tags.

Attention: The HDX Format is only supported by TS-HW38 AC and TS-RW68 AC!

Programming Interface (SDK) of TS-RW devices

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

PortHandle	Port handle
TTyp	transponder type from table
pFactoryID	FactoryID (UID) of the selected transponder. Depending on the transponder type, this has a different value range. The length of the FactoryID must be 8 Byte.
Manufacturer	Manufacturer code
SerialNumber	Serial number
Lock	0 = Default. If this value is 1, the blocks are locked.
Return:	-1 Error. 0 Successfully written 2 Transponder detected, the transponder is write protected and could not be written. 3 Transponder detected, the transponder has same factory ID as committed, the transponder was not written. 4 Process Pending, see next page!

Whit this function the transponder is selected, if necessary the configuration is changed (formatted) and then the data is written and if selected write protected.
In pFactoryID is the UID of the written transponders returned.
To avoid that the same transponder is written multiple times, in pFactoryID the last used UID can be given. Then it will check if the selected transponder has this UID and then return value 3 is returned.
The writing of the blocks is repeated up to 5 times if a writing error occurs.
By giving the value NO_RETRY = 1000Hex = 4096 Dez. in addition to the Transponder type the write retry can be suppressed.
By giving the value READ_AFTER_WRITE = 100Hex = 256 Dez. in addition to the Transponder type a read check will be made at each block and if an error occurs the block will be rewritten automatically. With this the failure rate can be lowered again, but the duration will be increased. The additional reading of block needs about 100 mSec.
With this function also FDX and HDX transponder can be written. The correct transponder type has to be selected.
At most HDX transponder types the Lock Flag has to be set so they are written correctly to be read also with other reading devices.

Attention: The HDX Format is only supported by TS-HW38 AC and TS-RW68 AC!

Programming Interface (SDK) of TS-RW devices

Writing as Background task
Because the function **TSLF_Write_EN14803** needs more time (about 500 – 700 mSec) this function can be called in "Overlapped Mode" also. For this the value of DO_OVERLAPPED = 2000Hex = 8192 Dez. is added to Transponder type. With this the function will be finished immediately and Return value will be
TSLF_RETVAL_PROCESS_PENDING = 4.

The function
int TSLF_IsWriteFinished(int PortHandle, BYTE * pFactoryID**)**

PortHandle	Port handle
pFactoryID	FactoryID (UID) of the selected transponder. Depending on the transponder type, this has a different value range. The length of the FactoryID must be 8 Byte.

is used to check if the write process is finished.

The return values of this function are the same as at TSLF_CheckFormatAndWrite_EN14803 described. Also in this case the Factory ID is only valid when the return value has reached 0 (successfully written). While the return value is TSLF_RETVAL_PROCESS_PENDING = 4, the write process is pending.
With qualified programming of the application it is possible with this to program multiple transponders at multiple connected devices simultaneously.
It has to be ensured that the distance between the antennas is big enough to avoid disturbance.

Programming Interface (SDK) of TS-RW devices

5.4. Usage of commands for EN14803Transponders:

1. Preparation:
First of all, the device interface is opened with command **TSLF_Open(….)**.
With command **TSLF_SetFilter(…)** the filter value for the selected transponder type is set. This value is saved in the device, so it is only necessary to set the value if the transponder type is changed. Setting the filter is not needed at TS-RW38 devices, because this devices have built in filter settings for all transponder types.

2. Writing to the transponder
This can be done comfortably with command **TSLF_Write_EN14803(…)**. This function checks first if a transponder is found, formats the transponder if needed and writes the desired data. Because this function also signals if no tag is in field, if can also be used to recognize and write transponders automatically.

3. Reading the transponder
The EN14803 data is read with command **TSLF_Read_EN14803(…)**.
This function does only give a positive result, if the found transponder is correctly formatted and the data is valid! New transponders will always return -1 with error code "**21** no Answer"

4. Reader with multiplexed antenna (TS-W80-AC 8x)
At these readers multiple antennas are used alternating. Switching between antennas is done through the command **TSLF_SetReaderAdresse(…)** The device acts as if 8 readers are connected simultaneously with one antenna each.

6. Error list

Error number	Description
1	Error while opening port, no device found at this port.
5	Timeout value invalid
6	Port handle invalid, no opened port with this handle found
7	Time out, no answer received during the given timeout from TSLF_Open
8	Data length too small for received data
9	Data length of data to send invalid
10	Invalid transponder type, this function is not allowed for the given transponder type
11	No data for communication
12	Block number invalid
13	Invalid lock value
14	Checksum error, incorrect checksum found in the data received from the transponder. Either the transponder contains invalid data or it is located too far from the reader.
15	No communication to device
16	Buffer empty, this error should never occur
17	Null pointer given, result cannot be stored.
18	Invalid mode given, please check parameters
19	It is not allowed to set baud rate for USB device
20	Read after write compare was negative
21 (NACK = 15H)	No acknowledge, Transponder does not respond, or Transponder does not accept the command.
22 (SYNC = 16H)	The device found an invalid checksum in its data. This might be caused by connection problem or invalid cable length.
23	Transponder is locked
24 (CAN = 18H)	Invalid command received, the device cannot execute this command. For example if programmer commands (Chapter 4) are used at TS-R38 which is only capable of reader mode.
25	Invalid data received, the received data does not confirm to the guidelines for the expected data format.