



**Programmierschnittstelle (SDK)  
der TS-HRW Serie (13.56 MHz)**

**Version 2.19**



**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](mailto:rfid@gis-net.de)  
Homepage <http://www.gis-net.de/rfid>

**Programmierschnittstelle (SDK) der TS-HRW Serie****Änderungsstand:**

Doku	SDK Version	Datum	Kapitel	Name	Info
2.02	2.02	03.12.2014	2.1	Blank	Einstellung der Gateway Adresse
2.03	2.03	12.10.2015	5.5	Blank	ISO14443B Kommandos ergänzt
2.03	2.03	12.10.2015	2.3.	Blank	Erweiterte Fehlermeldungsanzeige
2.03	2.03	12.10.2015	5.6	Blank	Mifare DESFire Kommandos ergänzt
2.03	2.03	12.10.2015	6	Blank	Fehlermeldungen ergänzt
2.03	2.03	25.02.2016		Blank	Gerätetypen aktualisiert
2.05	2.05	22.07.2016	5.6	Scherzinger	Mifare DESFire zusätzliche Erläuterungen
2.06	2.06	04.01.2017	6	Scherzinger	NFC Kommandos ergänzt
2.06	2.06	05.01.2017	1.1, 3.8	Blank	Erweiterungen für vereinfachte Verwendung im Readermodus
2.07	2.07	02.05.2017	6	Blank	NFC Kommando für Text ergänzt
2.08	2.08	19.06.2017	3.8	Blank	Ergänzung bei Fehler im Readermodus
2.09	2.09	27.07.2017	6.4	Scherzinger	NFC Kommandos erweitert bzw. umbenannt
2.10	2.11	04.12.2018	3.2	Blank	Reader Mode Parameter ergänzt
2.12	2.12	08.07.2019	2.	Blank	Zusätzliche Funktionen ergänzt
2.13	2.13	06.12.2019		Blank	Verwendung mehrere Geräte jetzt unterschiedliche Handles
2.13	2.14	08.01.2020		Blank	die Befehle TSHRW_Beep, TSHRW_SetLED, TSHRW_RawWrite werden nun korrekt ausgeführt
2.14	2.14	08.06.2020	2.3	Blank	Befehl TSHRW_KeepAppActive in Doku ergänzt
2.17	2.17	21.09.2022	2.1 2.3	Blank	Befehle für LAN Client Mode ergänzt. Befehle zum Lesen des Gerätenamens und Portnamens ergänzt.
2.18	2.18	11.09.2023	2.3, 5.3	Blank	Einige Kommandos hinzugefügt.
2.19	2.19	25.10.2023		Blank	Kommandos für Transparent Mode hinzugefügt. Namenskonventionen bei allen Kommandos aktualisiert.

**Eigentumsvorbehalt:**

Dieses Dokument sowie die Software (SDK) ist Eigentum der Firma GiS, Gesellschaft für Informatik und Steuerungstechnik mbH und ist vertraulich zu behandeln. Alle Informationen aus diesem Dokument sowie das SDK dürfen ausschließlich nur im Zusammenhang mit RFID-Systemen der Firma GiS verwendet werden. Ohne Einverständnis der Firma GiS darf keine Vervielfältigung und insbesondere keine Weitergabe an Dritte auch nicht in Auszügen, durchgeführt werden.



## Programmierschnittstelle (SDK) der TS-HRW Serie

# Inhaltsverzeichnis

<b>1. Allgemeines .....</b>	<b>5</b>
1.1. Betriebsarten .....	6
1.1.1. Readermodus .....	6
1.1.2. Programmermodus .....	6
1.2. Fehlerbearbeitung .....	7
1.3. Definitionen .....	7
<b>2. Allgemeine Befehle .....</b>	<b>8</b>
2.1. Ermitteln der angeschlossenen Geräte .....	8
2.2. Funktionen für Ethernet Geräte .....	10
2.3. Funktionen für Geräte mit Bluetooth Schnittstelle .....	14
2.4. Kommunikation mit einem Gerät aufbauen .....	16
2.5. Betriebsart festlegen .....	20
2.6. Geräteparameter setzen .....	21
2.7. Allgemeines Kommando absetzen .....	23
<b>3. Kommandos für Readermodus .....</b>	<b>24</b>
3.1. Schlüsselwert schreiben .....	24
3.2. Parameter lesen und schreiben .....	24
3.2.1. Aufbau der Parameter .....	25
3.3. Prefix .....	26
3.4. Suffix .....	26
3.5. Termix .....	27
3.6. Postcode .....	27
3.7. Reader Mode Parameter (nicht für Geräte mit PS2 Schnittstelle) .....	28
3.7.1. Aufbau der Reader Mode Parameter .....	28
3.8. Datenübernahme im Readermodus .....	29
3.8.1. Zyklische Abfrage .....	29
3.8.2. Callback Funktion einrichten .....	29
3.8.3. LED und Buzzer setzen .....	30
<b>4. Befehle für Transponder Typ ISO 15693 .....</b>	<b>31</b>
4.1. Transponder im Antennenfeld bestimmen .....	32
4.2. Transponder selektieren .....	33
4.3. Transponderinformationen auslesen .....	34
4.4. Transponder ruhigstellen .....	35
4.5. Transponder reaktivieren .....	35
4.6. Einzelnen Block lesen .....	36
4.7. Mehrere Blöcke lesen .....	36
4.8. Security Status lesen .....	37
4.9. Einzelnen Block schreiben .....	37
4.10. Block sperren .....	38
4.11. AFI Typ setzen .....	38
4.12. AFI Typ sperren .....	38
4.13. DSFID schreiben .....	39
4.14. DSFID sperren .....	39
4.15. Allgemeiner Zugriff .....	40



<b>5. Befehle für Transponder Typ MIFARE® (ISO14443A) .....</b>	<b>41</b>
5.1. Anwendungshinweise .....	41
5.1.1. MIFARE® Ultralight (NFC Type 2).....	41
5.1.2. MIFARE® Classic.....	42
5.1.3. MIFARE® DESFire.....	42
5.2. Funktionsaufrufe MIFARE® allgemein.....	43
5.3. Funktionsaufrufe MIFARE® Classic und Ultralight.....	45
5.4. Funktionsaufrufe ISO14443A-4.....	48
5.4.1. ISO14443A-4 Aktivierung .....	48
5.5. Funktionsaufrufe ISO14443B.....	50
5.6. Funktionsaufrufe Mifare DESFire.....	52
5.6.1. Sicherheitsrelevante Befehle.....	53
5.6.2. Befehle auf Karten-Ebene .....	57
5.6.3. Befehle auf Kartenebene zur Applikationsverwaltung .....	60
5.6.4. Befehle auf File-Ebene .....	63
5.6.5. Befehle nach ISO 7816-4 .....	78
<b>6. Befehle für NFC (Near Field Communication).....</b>	<b>80</b>
6.1. Anwendungshinweise .....	80
6.2. Unterstützte Transpondertypen.....	81
6.2.1. NFC Typ 2 .....	81
6.2.2. NFC Typ 4 .....	81
6.2.3. NFC Typ 6 .....	81
6.2.4. NFC Typ 7 .....	81
6.3. Unterstützte Anwenderdaten .....	82
6.3.1. Text .....	82
6.3.2. WWW-Adresse .....	82
6.3.3. Telefon .....	83
6.3.4. SMS.....	83
6.3.5. E-Mail .....	84
6.3.6. Visitenkarte.....	85
6.4. Funktionsaufrufe NFC.....	86
<b>7. Befehle für Transparentmodus .....</b>	<b>90</b>
<b>8. Fehlerliste.....</b>	<b>93</b>
8.1. Allgemeine Fehler .....	93
8.2. Fehler bei Zugriff auf ISO15693 Transponder .....	94
8.3. Fehler bei Zugriff auf MIFARE® Transponder .....	94
8.4. SDK-interne Fehler bei DESFire Kommandos .....	94
8.5. Fehler von der DESFire-Karte .....	95
8.6. Fehler von der DESFire-Karte bei ISO7816-4 Kommandos .....	95
8.7. Fehler von NFC-Kommandos .....	96



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 1. Allgemeines

Die Programmierschnittstelle für TS-HRWxx Geräte, dient der Vereinfachung der Ansprache der GiS RFID Geräte in beliebigen Softwareanwendungen.

Die Ansprache der Geräte erfolgt im GiS Standard Protokoll G200 oder S002.

Die Erkennung des Geräteprotokolls erfolgt automatisch beim Verbinden mit dem Gerät.

Dies wird durch eine Versionsabfrage mit den verschiedenen Protokolltypen durchgeführt.

Es wird eine Dynamic Link Library (DLL) zur Verfügung gestellt, die, die gesamte Funktionalität des Moduls abbildet. Diese DLL kann von verschiedenen Programmiersprachen aus verwendet werden.

Es werden zwei Varianten der DLL für 32 Bit oder 64 Bit Programme bereitgestellt.

Dateiname	Importbibliothek	Typ	Import Header für C / C++	Import für C#	Import für VB
TS_HRW_SDK.dll	TS_HRW_SDK.lib	32Bit	ts_hrw_import.h	ts_hrw_import.cs	ts_hrw_import.vb
TS_HRW_SDK64.dll	TS_HRW_SDK64.lib	64 Bit	ts_hrw_import.h	ts_hrw_import64.cs	ts_hrw_import64.vb

Je nach verwendetem Gerätetyp werden nicht alle Befehle unterstützt. So werden z.B.: die Befehle für Mifare® nur von TS-HRW38 und TS-HRW32 unterstützt.

Insbesondere ist darauf zu achten, dass nur TS-HW sowie TS-HRW Geräte den Programmermodus, und damit die Kommandos aus Kapitel 4 – 6 unterstützen. TS-HR Geräte sind auf die Verwendung im Readermodus beschränkt und antworten auf Kommandos des Programmermodus grundsätzlich mit dem Fehlercode 24.



## **Programmierschnittstelle (SDK) der TS-HRW Serie**

### **1.1. Betriebsarten**

Je nach Gerät stehen verschiedene Betriebsarten zur Verfügung. Es wird hier zwischen den Betriebsarten "Readermodus" und "Programmermodus" unterschieden.

Geräte der "R" Serie kennen nur den Readermodus, Geräte der W Serie nur den Programmermodus während Geräte der RW Serie beide Modi unterstützen.

#### **1.1.1. Readermodus**

Als Readermodus wird der automatische Lesemodus bezeichnet. Hier arbeitet der Leser autark und sendet über die Schnittstelle die gelesenen Transponderdaten im Klartext. Welche Daten er in welcher Formatierung sendet, wird über die Applikation "TS-HRW ReaderSetup" oder über die Funktionen aus Kapitel 3 eingestellt. Dieser Modus wird auch oft verwendet, wenn das Gerät ohne Zuhilfenahme des SDK verwendet werden soll, also wenn es an einer COM Schnittstelle direkt mit einer für COM Schnittstellen ausgelegten Applikation oder als HID Gerät im Tastaturmodus verwendet wird.

Zur Verwendung mit dem SDK stehen spezielle Zugriffsfunktionen zur Verfügung, die nicht mit dem GiS Standard Protokoll arbeiten. (Kapitel 3.8)

Insbesondere ist bei Verwendung anderer Kommandos im Readermodus Vorsicht walten zu lassen, da der Leser ja wenn ein Transponder ins Feld kommt automatisch Daten absetzt, die mit den Antworten auf andere Kommandos kollidieren können.

Aus diesem Grund wurden für den Zugriff auf den Summer sowie die LEDs zusätzliche Direktkommandos ohne Rückmeldung geschaffen. (Kapitel 3.8)

#### **1.1.2. Programmermodus**

Im Programmermodus wird der Transponderzugriff über die Kommandos ausgelöst.

Hier werden die Kommandos aus Kapitel 4, 5 und 6 sowie die allgemeinen Kommandos aus Kapitel 2 verwendet.

In diesem Modus können beliebige Blöcke der verschiedenen Transpondertypen gelesen und geschrieben werden.



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 1.2. Fehlerbearbeitung

Alle Befehle liefern einen Rückgabewert –1, wenn ein Fehler aufgetreten ist. Die Fehlernummer kann mit **TSHRW\_GetLastError()** abgefragt werden. Eine zugehörige Fehlerbeschreibung erhält man mit **TSHRW\_GetLastErrorMessage()**

Die Fehlerliste befindet sich im Anhang dieses Dokuments.

### 1.3. Definitionen

Folgende Datentypen werden bei der Übergabe zu den Funktionen verwendet:

int	32 Bit Integer
BYTE	8 Bit unsigned integer
BYTE *	Zeiger auf ein Feld mit 8 Bit unsigned integer Werten
char *	Zeiger auf ein Feld mit 8 Bit signed integer Werten, (ANSI String gemäß C Definition) meist 0-terminiert
WORD	16 Bit unsigned integer
WORD *	Zeiger auf ein Feld mit 16 Bit unsigned integer Werten

Die Datenübergabe in Feldern erfolgt grundsätzlich LSB zuerst.



## **2. Allgemeine Befehle**

### **int TSHRW\_LibVersion()**

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

Liefert die Version der DLL.

Diese Funktion benötigt kein TSHRW\_OpenPort.

### **int TSHRW\_IsVirtualComInstalled()**

Rückgabewert: 1 Virtual Com Treiber für TS-HRW38 ist installiert

0 Virtual Com Treiber für TS-HRW38 ist nicht installiert.

### **int TSHRW\_IsDeviceDriverMissing()**

Rückgabewert: 0: Keine GiS Geräte mit fehlenden Treibern gefunden

1: Treiber für angeschlossenes HRW34 USB Gerät fehlt

2: Treiber für angeschlossenes HRW38 VCOM Gerät fehlt

3: Treiber für angeschlossene HRW34 USB und HRW38 VCOM Geräte fehlen

## **2.1. Ermitteln der angeschlossenen Geräte**

### **int TSHRW\_CountAllDevices()**

Rückgabewert: Anzahl der USB Geräte

Der Rückgabewert ist die Anzahl der tatsächlich angeschlossenen USB Geräte.

Hier werden nur die Geräte der Firma GiS berücksichtigt. Maximum 1000 Geräte.

### **int TSHRW\_ListAllDeviceNames(char\* NamenListe, int BufferSize)**

NamenListe Die Namens Liste besteht aus einer Folge von Null-terminierten Strings.  
Das Listen Ende besteht aus einem Leerstring.

BufferSize BufferSize gibt an wie viel Bytes für die Namenliste bereitgestellt wird.

Rückgabewert: < 0 Fehler, Anzahl der tatsächlich benötigte Buffergröße in Bytes.

Mit der Funktion **ListAllDeviceNames** kann man sich die USB-Namen (Seriennummern) aller USB-Geräte besorgen. Es werden nur die TS-HRW3x und TS-HW3x Geräte von GiS berücksichtigt. Jeder Name besteht aus einem NULL terminierten String mit mindestens 8 und max. 18 ASCII Zeichen.

Beispiel: "11360001" oder "1460-0001 HID".

Aufgrund der Zusatzangabe "HID" kann erkannt werden, ob es sich um ein HID Gerät, das heißt ein Gerät welches das USB-HID Geräte Interface unterstützt, handelt.





## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_GetAvailablePorts(char\* NamenListe, int BufferSize)**

**NamenListe** Die Namens Liste besteht aus einer Folge von Null-terminierten Strings.  
Das Listen Ende besteht aus einem Leerstring.

**BufferSize** BufferSize gibt an wie viel Bytes für die Namenliste bereitgestellt werden.

**Rückgabewert:** < 0 Fehler, Anzahl der tatsächlich benötigte Buffergröße in Bytes.

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

Beispiele:

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

Die hier ermittelten Schnittstellenbezeichnungen können so direkt an TSHRW\_OpenPort übergeben werden.



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 2.2. Funktionen für Ethernet Geräte

**int TSHRW\_LanListAllDeviceNames(char\* NamenListe, int nBufferSize)**

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

Mit der Funktion **LanListAllDeviceNames** kann man sich die LAN-Namen aller LAN-Geräte besorgen. Es werden nur die TS-HR3x und TS-HW3x Geräte von GiS berücksichtigt.

Jeder Name besteht aus einem NULL terminierten String.

Es muss genügend Speicher für alle Geräte im Netz bereitgestellt werden.

Beispiele:

```
192.168.0.100-10001  
[TS-LAN01]
```

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

**int TSHRW\_LanConfigListDevices(char\* NamenListe, int nBufferSize)**

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

Mit der Funktion **LanConfigListDevices** kann man sich die LAN-Namen aller LAN-Geräte besorgen. Jeder Name besteht aus einem NULL terminierten String.

Es muss genügend Speicher für alle Geräte im Netz bereitgestellt werden.

Beispiele:

```
192.168.0.100-10001  
[TS-LAN01]169.194.245.01-10001
```

Die Gerätenamen können entweder aus der IP-Adresse des Gerätes gefolgt von der Portnummer oder bei DHCP Geräten aus dem DHCP Namen gefolgt von IP-Adresse und Portnummer bestehen. Der DHCP Name ist in eckigen Klammern [ ] eingefasst.

Mit dieser Funktion werden alle Geräte erfasst, auch wenn sie nicht mit den aktuellen Netzwerkeinstellungen erreichbar sind.



## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_LanChangeIPAddress(** LPCSTR OldAddress,  
LPCSTR NewAddress,  
LPCSTR IPMask)

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

**int TSHRW\_LanChangeIPAddressEx(** LPCSTR OldAddress,  
LPCSTR NewAddress,  
LPCSTR IPMask  
LPCSTR Gateway)

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

Mit diesen Funktionen kann die IP-Adresse und die Portnummer eines Gerätes geändert werden. Das Gerät muss hierzu mit den aktuellen Netzwerkeinstellungen erreichbar sein.

Die Übergabeparameter sind jeweils 0 terminierte ASCII Strings.

Die IP-Adressen bestehen entweder aus der IP-Adresse gefolgt von der Portnummer oder aus dem DHCP Namen eingeschlossen in [ ].

Beispiele:

192.168.0.100-10001  
[TS-LAN01]

In der IP Maske werden signifikanten Bits als Bitmaske angegeben.

Beispiel:

255.255.255.0

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

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



## Programmierschnittstelle (SDK) der TS-HRW Serie

```
int TSHRW_LanGetIPAddressEx( LPCSTR Name,  
                             LPSTR Address,  
                             LPSTR IPMask  
                             LPSTR Gateway)
```

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

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

```
int TSHRW_LanGetInfo( LPCSTR strName, char * DHCPName, char * IPAddress,  
                     char * IPMask, char * GatewayAddress);
```

strName	Name des Gerätes
DHCPName	DHCPName
IPAddress	IP-Adresse
IPMask	IP-Maske
GatewayAddress	Gateway Adresse

Mit dieser Funktion wird DHCP Name und IP Adresseinstellungen aus dem Gerät geladen.  
Die Übergabeparameter sind jeweils 0 terminierte ASCII Strings.  
Im Gegensatz zu GetIPAddressEx werden hier sowohl der DHCP Name als auch die  
IP Adresse geliefert.



## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_LanGetRemoteIPAndPort(** LPCSTR strName, char \* pIPAddress,  
int MaxIPLen, int \* pPortNr);

strName	Name des Gerätes
pIPAddress	IP-Adresse
MaxIPLen	Maximale Länge von pIPAddress
pPortNr	Portnummer
Rückgabewert	-1: Fehler, $\geq 0$ Länge der IP-Adresse

Mit dieser Funktion wird die Remote IP Adresse als 0 terminierter String sowie die Remote Port Nummer eines Gerätes gelesen. Die Remote IP-Adresse/Port dient dazu, dass das Gerät sich automatisch mit einem Host als Service verbinden kann. In diesem Fall arbeitet das Gerät im Client Modus.

**int TSHRW\_LanWriteRemoteIPAndPort(** LPCSTR strName,  
const char \* pIPAddress, int PortNr)

strName	Name des Gerätes
pIPAddress	IP-Adresse
PortNr	Portnummer
Rückgabewert	-1: Fehler, $\geq 0$ Ok

Hiermit wird die Remote IP Adresse eingetragen. Die Übergabe erfolgt als 0 terminierter String. Wird als IPAddress ein Leerstring eingegeben, so wird der Client Mode deaktiviert.

**int TSHRW\_LanIsDeviceAvailable(LPCSTR Address)**

Address	IP-Adresse
---------	------------

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

Die IP-Adresse besteht entweder aus der IP-Adresse gefolgt von der Portnummer oder aus dem DHCP Namen eingeschlossen in [ ].

Beispiele:

192.168.0.100-10001  
[TS-LAN01]

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



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 2.3. Funktionen für Geräte mit Bluetooth Schnittstelle

Bei Geräten mit Bluetooth Schnittstelle ist ein zweistufiger Verbindungsaufbau gefordert. Zuerst muss das Lokale Bluetooth Gerät gefunden werden (GIS Bluetooth USB Adapter) Mit diesem Gerät wird eine Verbindung über TSHRW\_OpenPort(...) aufgebaut. Dann können über diese Verbindung externe Bluetooth Geräte gesucht und verbunden werden.

#### **int TSHRW\_BT\_ListAllDeviceNames(char\* Buffer, int BufferSize)**

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

Mit der Funktion **BT\_ListAllDeviceNames** kann man sich die USB-Namen (Seriennummern) aller vorhandenen GIS USB Bluetooth Adapter besorgen. Jeder Name besteht aus einem NULL terminierten String.

#### **int TSHRW\_BT\_Search(int hPort, BYTE \* DeviceList, int BufferSize, int Timeout)**

**hPort** Logische Gerätenummer  
**DeviceList** Die Beschreibung der DeviceList folgt unten.  
**BufferSize** BufferSize gibt an wie viel Bytes für die Namenliste bereitgestellt wird.  
**Timeout** Dauer, die für die Suche verwendet wird. Angabe ist in Millisekunden, üblicherweise 120000.  
**Rückgabewert:** < 0 Fehler, Anzahl der tatsächlich benötigte Buffergröße in Bytes.

Es werden immer nur so viele Geräte gemeldet, wie in der DeviceList Platz ist. Sind weitere Geräte vorhanden, so werden diese ignoriert.

Ein Eintrag in der DeviceList hat folgenden Aufbau:

Start	Länge	Bedeutung
0	6	Bluetooth Adresse des Gerätes
6	32	Local Name des Bluetooth Gerätes z.B.: "PN1450#0001" Der Name ist mit Nullbytes aufgefüllt. Der Name kann maximal 31 Byte lang sein, das 32. Byte ist der Stringterminator (Nullbyte)

Die DeviceList besteht aus einer Aneinanderreihung solcher Einträge.  
 Die Zurückgegebene Länge ist also immer ein Vielfaches von 38.



## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_BT\_Connect**(int hPort, BYTE \* Device, int Timeout)

hPort                      Logische Gerätenummer

Device                    Bluetooth Adresse des zu verbindenden Gerätes

Timeout                  max. Dauer für Verbindungsaufbau in Millisekunden, üblicherweise 3000.

Rückgabewert:          < 0 Fehler,  
                              0 Verbindung hergestellt.

Mit dieser Funktion wird die Verbindung zum angegebenen Gerät hergestellt.

Ist die Verbindung aufgebaut, so können die üblichen Funktionen zur Kommunikation mit dem Gerät verwendet werden, wie bei direkt angeschlossenen Geräten.

**int TSHRW\_BT\_CheckConnection**(int hPort)

hPort                      Logische Gerätenummer

Rückgabewert:          < 0 Fehler,  
                              0 Verbindung ist getrennt.  
                              1 Verbindung ist aktiv

Prüfen des Verbindungszustandes. Hiermit kann festgestellt werden, ob die Verbindung getrennt wurde.

**int TSHRW\_BT\_Disconnect**(int hPort)

hPort                      Logische Gerätenummer

Rückgabewert:          < 0 Fehler,  
                              0 Verbindung getrennt.

Trennen der Verbindung zum Gerät. Anschließend kann eine neue Verbindung auch zu einem anderen Bluetoothgerät aufgebaut werden.



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 2.4. Kommunikation mit einem Gerät aufbauen

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

**int TSHRW\_OpenPort(LPCTSTR strInterfaceName, int Baudrate, int Timeout)**

Öffnen der Kommunikationsschnittstelle für alle Geräte.

**strInterfaceName** Name der Schnittstelle. z.B. COM1 oder Seriennummer des USB Geräts.  
Bei **USB** Geräten muss der Gerätenamen eingetragen werden.  
Bei **USB HID** Geräten muss der Gerätenamen gefolgt von "HID" eingetragen werden.

Die verfügbaren USB-Namen, kann man mit der Funktion **TSHRW\_ListAllDeviceNames** ermitteln.

z.B. „10610011“ oder "1317-0001 HID"

Bei **LAN** Geräten muss die Adresse eingetragen werden.

Die verfügbaren LAN Geräte kann man mit der Funktion **TSHRW\_LanListAllDeviceNames** ermitteln.

**Baudrate** Gewünschte Übertragungsrate.  
(2400, 4800, 9600, 19200, 38400, 57600, 115200 und 230400 zulässig).  
Achtung: Nicht alle Geräte unterstützen alle Baudraten. Bitte sehen Sie in der Gerätespezifikation nach den verfügbaren Übertragungsraten.  
**RS232 Leser sind standardmäßig auf 19200 eingestellt.**  
**USB und LAN Leser ignorieren die Baudrate.**

**Timeout** Zeit nach der die Kommunikation abgebrochen wird. (in Millisekunden)

**Rückgabewert:** <0: Fehler, >0: **PortHandle** logische Gerätenummer

Wahlweise kann beim Schnittstellennamen noch die Geräteadresse mit angegeben werden. Also z.B.: COM1:4 öffnet an COM 1 das Gerät mit Adresse 4. Dies ist notwendig, wenn die Geräteadresse nicht 1 ist, da bei `TSHRW_OpenPort` der Verbindungsaufbau mit dem Gerät gleich durchgeführt wird und die Verbindung nur akzeptiert wird wenn das Gerät antwortet.

Die Funktion erkennt automatisch das zugrundegelegte LowLevel Protokoll des Gerätes (normalerweise GiS LowLevel Protokoll G200, bei MultiX Geräten MultiX Modbus Protokoll, bei S002 Geräten GiS LowLevel Protokoll S002, bei Medio P200u S004 Protokoll)

Medio P200u ist immer Virtual COM Mode und muss mit Baudrate 115200 betrieben werden.

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

**int TSHRW\_ClosePort(int hPort)**

**hPort** Logische Gerätenummer

**Rückgabewert:** -1 Fehler, 0 OK





## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_GetPortName**(int hPort, char \* pName, int MaxLen)

hPort                      Logische Gerätenummer  
pName                      Zeiger auf Speicher für Schnittstellennamen  
MaxLen                      Zur Verfügung stehende Länge in pName  
Rückgabewert:              -1 Fehler, > 0 Länge des Schnittstellennamens

Liefert den Namen der Schnittstelle wie bei OpenPort angegeben.

**int TSHRW\_KeepAppActive** (int hPort, int bActive)

hPort                      Logische Gerätenummer  
bActive                      0: Applikation wird inaktiv während eine SDK Funktion ausgeführt wird  
                                1: Applikation bleibt aktiv während eine SDK Funktion ausgeführt wird

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

**int TSHRW\_IsUSB**(int hPort)

hPort                      Logische Gerätenummer  
Rückgabewert:              -1 Fehler, 0 = **KEIN** USB Port, 1 = USB Port

**int TSHRW\_IsProgrammer**(int hPort)

hPort                      Logische Gerätenummer  
Rückgabewert:              -1 Fehler  
                                0 = Programmer Modus wird **nicht** unterstützt  
                                1 = Programmer Modus wird unterstützt

Diese Funktion meldet nur, ob der Programmer Modus möglich ist, und die entsprechenden Befehle unterstützt werden, jedoch nicht ob der Programmer Modus eingeschaltet ist.

**int TSHRW\_IsReader**(int hPort)

hPort                      Logische Gerätenummer  
Rückgabewert:              -1 Fehler  
                                0 = Reader Modus wird **nicht** unterstützt  
                                1 = Reader Modus wird unterstützt

Diese Funktion meldet nur, ob der Reader Modus möglich ist, nicht jedoch ob der Reader Modus eingeschaltet ist.



## Programmierschnittstelle (SDK) der TS-HRW Serie

int **TSHRW\_SetReaderAdresse**(int hPort, int Adresse)

hPort                      Logische Gerätenummer

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

Rückgabewert:            < 0 Fehler, 0 OK

int **TSHRW\_DeviceVersion**(int hPort, BYTE \* pBuffer, int BufLen)

hPort                      Logische Gerätenummer

pBuffer                    Zeiger auf die Versionsdaten

BufLen                     Länge der Versionsdaten (4 Byte)

Rückgabewert:            -1 Fehler

Gerätenummer und Gerätefirmware im ASCII Format.

Byte 1 – 3                Gerätenummer

Byte 4                    Geräte Firmware (0 – 9,A – Z,a – z)

### Folgende Gerätenummern werden von der DLL unterstützt:

Gerätenummern von TS-HR3x und TS-HW3x RFID Geräten

Die Gerätenummer dient zur Klassifizierung der Geräte Typen.

121 =	(TS-HW34)	nur Programmier
122 =	(TS-HR34 PS2)	nur Leser für PS2 Schnittstelle (konfigurierbar über RS232)
123 =	(TS-HRW34 PS2)	Leser für PS2 Schnittstelle und Programmier mit RS232 Interface
124 =	(TS-HR34)	nur Leser
125 =	(TS-HRW34)	Programmier und Leser
135 =	(TS-HRW90)	Programmier und Leser mit RS485 Interface
141 =	(TS-HW36)	nur Programmier mit HID oder LAN Interface
144 =	(TS-HR36)	nur Leser mit HID oder LAN Interface
145 =	(TS-HRW36)	Programmier und Leser
161 =	(TS-HW38)	nur Leser mit HID oder LAN Interface
164 =	(TS-HR38)	nur Leser mit HID oder LAN Interface
165 =	(TS-HRW38)	Programmier und Leser
171 =	(TS-HW34 S002)	Programmier mit S002 Protokoll
195 =	(TS-HRW32)	Programmier und Leser
117 =	(TS Medio P200u)	Programmier mit S004 Protokoll
405 =	(TS-HRW390)	Programmier und Leser
411 =	(TS-HW421)	nur Leser mit HID oder LAN Interface
414 =	(TS-HR421)	nur Leser mit HID oder LAN Interface
415 =	(TS-HRW421)	Programmier und Leser
421 =	(TS-HW420)	Programmier mit S002 Protokoll
431 =	(TS-HW480)	Programmier mit S002 Protokoll
451 =	(TS-HW380)	nur Leser mit HID oder LAN Interface
454 =	(TS-HR380)	nur Leser mit HID oder LAN Interface
455 =	(TS-HRW380)	Programmier und Leser

Die Geräte TS-HW34, HR34 und HRW34 gibt es als RS232, USB oder Bluetooth Ausführung.

Die Geräte TS-HW36, HR36 und HRW36 gibt es als USB-HID oder LAN Ausführung.

Die Geräte TS-HW38, HR38 und HRW38 gibt es als RS232, USB-HID, USB-VCOM oder LAN Ausführung.

Die Geräte TS-HRW32 gibt es als RS232, USB-HID oder USB-VCOM Ausführung.

Die Geräteserien TS-HRW38, TS-HRW32, TS-HRW390 und TS-HRW380 unterstützen außer den Transpondern im ISO15693 Standard auch MIFARE® classic, Ultralight® und DESFire®.





## Programmierschnittstelle (SDK) der TS-HRW Serie

### 2.5. Betriebsart festlegen

**int TSHRW\_SetReaderMode(int hPort, int mode)**

hPort                      Logische Geräteummer

Mode                      1 setzt das Gerät in den Lesermodus.

0 setzt das Gerät in den Programmiermodus.

2 setzt das Gerät in den Einschaltmodus

80H legt als Einschaltmodus den Programmiermodus fest

81H legt als Einschaltmodus den Readermodus fest.

Rückgabewert:            -1 Fehler, 0 OK

Bei Geräten die sowohl den Readermodus als auch den Programmiermodus unterstützen kann mit diesem Befehl der Reader- bzw. Programmiermodus aktiviert werden.

Im Readermodus sendet das Gerät automatisch die Daten des Transponders ohne Protokollrahmen wie in der Parametereinstellung für den Reader Mode eingestellt. Dies ist im Programmiermodus äußerst störend, weshalb hier der Readermodus deaktiviert werden muss.

Außerdem kann es besonders bei HID (Tastatursimulation) Geräten störend sein, wenn immer die Tastatursimulation aktiviert wird sobald ein Transponder aufgelegt wird.

**int TSHRW\_SetRF(int hPort, int OnOff)**

hPort                      Logische Geräteummer

OnOff                      Kommando, 0 = Antennenfeld ausschalten, 1 = Antennenfeld einschalten

Rückgabewert:            -1 Fehler, 0 OK

Ein und Ausschalten des Antennenfeldes



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 2.6. Geräteparameter setzen

**int TSHRW\_SetIO(int hPort, int Maske, int Daten)**

hPort                      Logische Gerätenummer

Maske                      Maskenwert für die zu setzenden Ausgänge

Daten                      Werte der zu setzenden Ausgänge

Es werden die Bits aus Daten übernommen, die in Maske gesetzt sind.

Rückgabewert:            -1 Fehler, 0 OK

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

Bedeutung der Bits:

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

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

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

**int TSHRW\_ReadIO(int hPort)**

hPort                      Logische Gerätenummer

Rückgabewert:            -1 Fehler, sonst Werte der gelesenen Eingänge

Je nach Geräteversion können die Eingänge unterschiedlich vorhanden und belegt sein.

Ist das Gerät mit einem Sensoreingang ausgestattet, so wird das unterste Bit für den Sensoreingang verwendet.

**int TSHRW\_SetDefault(int hPort)**

hPort                      Logische Gerätenummer

Rückgabewert:            -1 Fehler, 0 OK

Standardeinstellung aktivieren. Diese Funktion wird nicht von allen Geräten unterstützt.



## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_SetConfig**(int hPort, BYTE \* pConfig, int ConfigLen)

hPort                      Logische Geräteummer  
pConfig                    Zeiger auf Konfiguration  
ConfigLen                 Länge des Puffers  
Rückgabewert:            -1 Fehler, 0 OK

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

**int TSHRW\_GetConfig**(int hPort, BYTE \* pConfig, int ConfigLen)

**int TSHRW\_GetConfigEx**(int hPort, int Select, BYTE \* pConfig, int ConfigLen)

hPort                      Logische Geräteummer  
Select                     Bitkodierte Auswahl zur Selektierung der Antwort.  
pConfig                    Zeiger auf den, vom Benutzer zur Verfügung gestellten, Lesebuffer  
ConfigLen                 Länge des Puffers  
Rückgabewert:            -1 Fehler, Länge der tatsächlich gelesenen Daten.

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

**int TSHRW\_ReadSerialNumber**(int hPort, BYTE \* pSerial, int Buflen)

hPort                      Logische Geräteummer  
pSerial                    Zeiger auf den, vom Benutzer zur Verfügung gestellten, Lesebuffer  
Buflen                     Länge des Puffers  
Rückgabewert:            -1 Fehler, Länge der tatsächlich gelesenen Daten.

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

**int TSHRW\_SetCommunicationMode**(int hPort, int Mode)

hPort                      Logische Geräteummer  
Mode                       Kommunikationsmodus  
                              1: HID Schnittstelle  
                              2: Virtual COM Schnittstelle  
Rückgabewert:            -1 Fehler, 0 OK

Umschalten eines HID Gerätes in den Virtual COM Modus oder umgekehrt. Anschließend muss die Schnittstelle geschlossen und im neuen Mode wieder geöffnet werden.



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 2.7. Allgemeines Kommando absetzen

Manche Geräte unterstützen zusätzliche Kommandos, die hier über eine allgemeine Funktion angesprochen werden können. Dabei erfolgt der Zugriff immer über das vorgegebene Datenprotokoll.

**int TSHRW\_Transfer**(int hPort, BYTE \* pSendBuf, int SendBufLen,  
BYTE \* pRecvBuf, int RecvBufLen)

hPort	Logische Gerätenummer
pSendBuf	Zeiger auf zu sendende Daten beginnend mit Kommando.
SendBufLen	Länge der zu sendenden Daten
pRecvBuf	Zeiger auf Empfangsdaten
RecvBufLen	Max. Länge der Empfangsdaten
Rückgabewert:	-1 Fehler, >= 0 Länge der gelesenen Daten in pRecvBuf

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

**int TSHRW\_RawWrite**(int hPort, BYTE \* pBuffer, int BufLen)

hPort	Logische Gerätenummer
pBuffer	Zeiger auf Schreibpuffer
BufLen	Länge des Puffers
Rückgabewert:	-1 Fehler, 0 OK

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

**int TSHRW\_RawRead**(int hPort, BYTE \* pBuffer, int BufLen)

hPort	Logische Gerätenummer
pBuffer	Zeiger auf den, vom Benutzer zur Verfügung gestellten, Lesebuffer
BufLen	Länge des Puffers
Rückgabewert:	-1 Fehler, Länge der tatsächlich gelesenen Daten.

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



### **3. Kommandos für Readermodus**

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

#### **3.1. Schlüsselwert schreiben**

Diese Funktion wird nur bei MIFARE® Transpondern benötigt und wird daher auch nur bei Lesern unterstützt, welche die MIFARE® Betriebsart unterstützen

**int TSHRW\_WriteMifareKey**(int hPort, int KeyType, BYTE \* pKey, int KeyLen)

hPort	Logische Gerätenummer
KeyType	Schlüsselart
pKey	Zeiger auf Schlüssel. Der Schlüssel muss genau 6 Byte groß sein.
KeyLen	Länge des Schlüssels (6 Byte)
Rückgabewert:	-1 Fehler, 0 OK

#### **3.2. Parameter lesen und schreiben**

Es können mehrere Parameter Datenstrukturen übergeben werden. Dann werden die angegebenen Transpondertypen abwechselnd ausgeführt.

Wird nur eine Datenstruktur übergeben, so muss diese nicht komplett gefüllt sein. Werden mehrere Datenstrukturen übergeben, so müssen die einzelnen Datenstrukturen immer komplett mit 20 Byte pro Datenstruktur übergeben werden.

Achtung nur neuere Gerätetypen unterstützen mehrere Datenstrukturen. Bei allen älteren Geräten wird nur eine Datenstruktur verwendet, teilweise werden dort auch nicht alle Parameter verwendet.

**int TSHRW\_ReadParam**(int hPort, BYTE \* pBuffer, int BufLen)

hPort	Logische Gerätenummer
pBuffer	Zeiger auf Lesebuffer. Siehe: <b>3.2.1 Aufbau der Parameter.</b>
BufLen	Länge des Puffers (20 Byte pro Datenstruktur)
Rückgabewert:	-1 Fehler, Länge der tatsächlich gelesenen Daten.

**int TSHRW\_WriteParam**(int hPort, BYTE \* pBuffer, int BufLen)

hPort	Logische Gerätenummer
pBuffer	Zeiger auf Schreibbuffer. Siehe: <b>3.2.1 Aufbau der Parameter.</b>
BufLen	Länge des Buffers (20 Byte pro Datenstruktur)
Rückgabewert:	-1 Fehler, 0 OK



### 3.2.1. Aufbau der Parameter

Pos.	Länge	Name	Beschreibung
0	1	Wait	Wartezeit zwischen 2 Zeichen: (0 – 127) Nur für PS/2. Wartezeit = N/2 in Millisekunden. Standardwert = 10.
1	1	TTyp	Transpondertyp: Bit 0 – 5    0 = UID ISO15693                      1 = Einzelblöcke ISO15693 2 = UID Mifare                        3 = Einzelblöcke Mifare 4 = UID ICode1                        5 = Einzelblöcke ICode1 6 = UID DESFire                       7 = Dateiinhalte DESFire 8 = Blockfolge ISO15693    9 = Blockfolge Mifare Bit 6:        0 = normal                            1 = bitweise gespiegelt Bit 7         0 = LSB first                        1 = MSB first (Byteweise gedreht)
2	5	Register	Der Eintrag besteht immer aus 4 Registerdaten und der Endekennung 0xFFH (5. Byte). Dieser Eintrag wirkt nur, wenn bei TTyp Einzelblöcke (1, 3 oder 5) gewählt ist. Hier werden die Blocknummern definiert, die im Transponder gelesen werden sollen. Es werden maximal 4 Blocknummern übertragen. Byte 5 ist immer die Endekennung 0xFFH. z.B. 0x00,0x01,0x0F,0x05,0xFF. Ist als TTyp 7 Dateiinhalte DESFire angegeben, so wird in den ersten 3 Byte die ApplikationsID und im 4. Byte die FileID angegeben. Ist als TTYP 8 oder 9 angegeben, so steht in den ersten 2 Byte der Startblock und im 3. Byte die Länge der zu lesenden Daten in Byte.
7	1	DTyp	Datentyp: 0 = Hexadezimal,                                      3 = Dezimal ohne führenden Nullen, 1 = Dezimal,    4 = Hexadezimal mit Kleinbuchstaben 2 = ASCII,    5 = ASCII, 00H unterdrücken, nicht auffüllen mit SPACE
8	1	Zeichen	Zeichenanzahl. (1–32). Hier wird definiert wie viel Zeichen auf einmal ausgegeben werden sollen.
9	1	Frequenz	Sendefrequenz. Nur für PS/2. Hier wird eine spezielle Intervalldauer übergeben. Die dazugehörige Frequenz wird nach der Formel: $f[MHz] = \frac{1}{(64[\mu s] + 8[\mu s] * N)}$ berechnet. Der Wert N = 5 entspricht einer Frequenz von 10kHz. (Standardwert) Der Wert N = 17 entspricht einer Frequenz von 5 kHz. Die Werte sind gerundet.
10	1	Timeout	Die Timeoutzeit wird nach einer Formel berechnet (Nur für PS/2): T [ms] = 30[ms] + 30[ms] * Timeout. Standardwert ist 5, entspricht 180 ms. Nach Ablauf dieser Zeit kann derselbe Transponder erneut gelesen werden.
11	1	ValidBytes	(1-16) Anzahl der Bytes, die aus der UID oder den Datenblöcken verwendet werden. Hiermit können z.B.: die oberen Bits der UID ausgeblendet werden. Standardwert ist 5
12	1	ValidFrom	(1-16) Startbyte ab dem die Daten übertragen werden, dieser Parameter ist nur bei TS-HR38 ab Version 1.07 und bei TS-HR32 erlaubt
13	1	TypKenn1	Erkennungszeichen für den Transpondertyp wird vor den Daten übertragen, 1 ASCII Zeichen, bei 0 wird nichts übertragen
14	1	TypKenn2	Erkennungszeichen für den Transpondertyp wird nach den Daten übertragen, 1 ASCII Zeichen, bei 0 wird nichts übertragen
15	1	DesfireMode	0: plain, 1: Authentifiziert, 2 mit MAC, 3: voll verschlüsselt
16	4	-	Reserve, Standardwert 0



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 3.3. Prefix

Schreiben und lesen der Prefixeinstellung

Der Prefix besteht maximal aus 31 Zeichen als Endekennung wird FFh eingesetzt.

**int TSHRW\_WritePrefix**(int hPort, BYTE \* pBuffer, int BufLen)

hPort                      Logische Gerätenummer

pBuffer                    Zeiger auf Schreibpuffer

BufLen                    Länge des Puffers

Rückgabewert:            -1 Fehler, 0 OK

**int TSHRW\_ReadPrefix**(int hPort, BYTE \* pBuffer, int BufLen)

hPort                      Logische Gerätenummer

pBuffer                    Zeiger auf Lesebuffer

BufLen                    Länge des Puffers

Rückgabewert:            -1 Fehler, Länge der tatsächlich gelesenen Daten.

### 3.4. Suffix

Schreiben und lesen der Suffixeinstellung

Der Suffix besteht maximal aus 31 Zeichen als Endekennung wird FFh eingesetzt.

**int TSHRW\_WriteSuffix**(int hPort, BYTE \* pBuffer, int BufLen)

hPort                      Logische Gerätenummer

pBuffer                    Zeiger auf Schreibpuffer

BufLen                    Länge des Puffers

Rückgabewert:            -1 Fehler, 0 OK

**int TSHRW\_ReadSuffix**(int hPort, BYTE \* pBuffer, int BufLen)

hPort                      Logische Gerätenummer

pBuffer                    Zeiger auf den Lesebuffer

BufLen                    Länge des Puffers

Rückgabewert:            -1 Fehler, Länge der tatsächlich gelesenen Daten.



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 3.5. Termix

Schreiben und lesen der Termixeinstellung

Der Termix besteht maximal aus 31 Zeichen als Endekennung wird FFh eingesetzt.

**int TSHRW\_WriteTermix**(int hPort, BYTE \* pBuffer, int BufLen)

hPort	Logische Gerätenummer
pBuffer	Zeiger auf Schreibpuffer
BufLen	Länge des Puffers
Rückgabewert:	-1 Fehler, 0 OK

**int TSHRW\_ReadTermix**(int hPort, BYTE \* pBuffer, int BufLen)

hPort	Logische Gerätenummer
pBuffer	Zeiger auf den Lesebuffer
BufLen	Länge des Puffers
Rückgabewert:	-1 Fehler, Länge der tatsächlich gelesenen Daten.

### 3.6. Postcode

Schreiben und lesen der Postcodeeinstellung

Der Postcode besteht maximal aus 31 Zeichen als Endekennung wird FFh eingesetzt.

**int TSHRW\_WritePostcode**(int hPort, BYTE \* pBuffer, int BufLen)

hPort	Logische Gerätenummer
pBuffer	Zeiger auf Schreibpuffer
BufLen	Länge des Puffers
Rückgabewert:	-1 Fehler, 0 OK

**int TSHRW\_ReadPostcode**(int hPort, BYTE \* pBuffer, int BufLen)

hPort	Logische Gerätenummer
pBuffer	Zeiger auf den Lesebuffer
BufLen	Länge des Puffers
Rückgabewert:	-1 Fehler, Länge der tatsächlich gelesenen Daten.



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 3.7. Reader Mode Parameter (nicht für Geräte mit PS2 Schnittstelle)

Schreiben und lesen der Reader Mode Parameter

**int TSHRW\_WriteReadModeParam(int hPort, BYTE \* pBuffer, int BufLen)**

hPort                      Logische Gerätenummer

Pbuffer                    Zeiger auf Schreibpuffer.

Siehe: **3.7.1 Aufbau der Reader Mode Parameter**

BufLen                    Länge des Puffers

Rückgabewert:            -1 Fehler, 0 OK

**int TSHRW\_ReadReadModeParam(int hPort, BYTE \* pBuffer, int BufLen)**

hPort                      Logische Gerätenummer

pBuffer                    Zeiger auf den Lesebuffer

Siehe: **3.7.1 Aufbau der Reader Mode Parameter**

BufLen                    Länge des Puffers

Rückgabewert:            -1 Fehler, Länge der tatsächlich gelesenen Daten.

#### 3.7.1. Aufbau der Reader Mode Parameter

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

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

**Mode:**                    Betriebsart

0: Daten bei Auflegen und Abziehen des Transponders senden

1: Daten auf Anforderung senden.

Das Anforderungszeichen wird in **Anforderung** definiert.

Das Gerät sendet Daten, wenn die Anforderung zum Geräte gesendet wurde.

3: Daten zyklisch senden

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

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

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



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 3.8. Datenübernahme im Readermodus

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

#### 3.8.1. Zyklische Abfrage

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

#### 3.8.2. Callback Funktion einrichten

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

**int TSHRW\_StartAutoRead(int hPort, int TermChar, AutoReadCallback pAutoReadProc)**

hPort                      Logische Gerätenummer

TermChar                Endezeichen der Übertragung. Mit diesem Zeichen wird der Aufruf der Callback Funktion getriggert.

pAutoReadProc        Zeiger auf die Callback Funktion

Rückgabewert:        -1 Fehler, 0: OK

Definition der Callback Funktion:

'C'

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

'C#'

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

'VB'

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

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

**int TSHRW\_StopAutoRead(int hPort)**

hPort                      Logische Gerätenummer

Rückgabewert:        -1 Fehler, 0: OK

Beendet den Aufruf der Callback Funktion.



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 3.8.3. LED und Buzzer setzen

Diese Funktionen werden nur bei TS-HR38/HRW38 ab Version 1.23 sowie bei TS\_HR32/HRW32 ab Version 1.02 und bei TS-HR39/TS-HRW39 ab Version 1.02 unterstützt.

**int TSHRW\_SetLED(int hPort, int red, int green, int yellow)**

hPort                      Logische Geräteummer

red                        -1: Rote LED nicht verändern

0: Rote LED ausschalten

1: Rote LED einschalten

2: Kontrolle über Rote LED an Gerät zurückgeben

green                    -1: Grüne LED nicht verändern

0: Grüne LED ausschalten

1: Grüne LED einschalten

2: Kontrolle über Grüne LED an Gerät zurückgeben

yellow                   -1: Gelbe LED nicht verändern

0: Gelbe LED ausschalten

1: Gelbe LED einschalten

2: Kontrolle über Gelbe LED an Gerät zurückgeben

Rückgabewert:        -1 Fehler, 0: OK

**int TSHRW\_Beep(int hPort)**

hPort                    Logische Geräteummer

Rückgabewert:        -1 Fehler, 0: OK

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



#### **4. Befehle für Transponder Typ ISO 15693**

Grundsätzlich wird bei den Befehlen für ISO15693 das RequestFlag mit übergeben. Bitte beachten Sie die ISO15693 Norm zur Bedeutung des RequestFlags.

Innerhalb der folgenden Funktionen werden benötigte Bits des RequestFlags automatisch aufgrund des Transpondertyps gesetzt. Wird der Befehl mit UID ausgeführt, so ist der Transpondertyp in der UID enthalten. Bei Verwendung ohne UID kann das Transpondertyp-Flag zum RequestFlag hinzugefügt werden (logisches **ODER**).

Das Transpondertyp-Flag ist folgendermaßen definiert.

MYD1	=	0x0100
ICODE2	=	0x0200
TAGIT2	=	0x0400
ELECTRO MARIN	=	0x0800
STM	=	0x1000
FUJITSU	=	0x2000
KSW	=	0x4000
KEINE Automatik	=	0x8000

Das automatische Setzen von RequestFlags kann verhindert werden, indem IGNORE\_AUTOMATIC\_FLAGS = 0x8000 zum übergebenen Wert addiert wird. Dann ist es allerdings in der Verantwortung des Benutzers, das RequestFlag korrekt zu setzen.



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 4.1. Transponder im Antennenfeld bestimmen

**int TSHRW\_ISO\_Inventory**(int hPort, BYTE \* pBuffer, int BufLen, int RequestFlag)

**int TSHRW\_ISO\_InventoryAFI**(int hPort, BYTE \* pBuffer, int BufLen,  
int RequestFlag, int AFI)

Inventory Information nach ISO Norm 15693 lesen.

Dieser Befehl muss von allen Transpondern unterstützt werden.

Dieser Befehl liefert den Unified identifier (UID) zurück.

hPort                      Logische Gerätenummer

pBuffer                  Zeiger auf Datenblock

BufLen                    Länge des Blocks (fest x mal 11Byte).  
(x = Anzahl der erkannten Transponder)

RequestFlag              Request Flag nach ISO Norm 15693

AFI                        AFI Wert nach ISO Norm 15693

Rückgabewert:          < 0 Fehler, Länge der gelesenen Daten

Die TSHRW\_ISO\_InventoryAFI Funktion wird verwendet, um die Inventory Information nur von Transpondern mit einer speziellen AFI Einstellung zu bekommen.

Pro gefundenen Transponder werden 11Byte mit folgender Struktur erzeugt.

	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
<b>Byte0</b>	Timeslotnummer				ISO 15693 - Fehlercode			

Fehlercode = 0, kein Fehler

Byte1: Response Flags nach ISO 15693.

Byte2: DSFID nach ISO 15693.

Byte3 - Byte10: UID (Unified identifier)





## Programmierschnittstelle (SDK) der TS-HRW Serie

### 4.2. Transponder selektieren

**int TSHRW\_ISO\_Select**(int hPort, BYTE \* pUID,int UIDLen,int nRequestFlag)

hPort                      Logische Gerätenummer

pUID                      Zeiger auf UID. Das erforderliche Bit im RequestFlag wird automatisch gesetzt, wenn eine UID angegeben wird. Ist der Zeiger NULL, wird die UID ignoriert

Alle UID's können mit TSHRW\_ISO\_Inventory() ermittelt werden.

UIDLen                    Länge des Blocks (8 Byte). Ist die Länge != 8 wird der Parameter pUID ignoriert.

RequestFlag              Request Flag nach ISO Norm 15693

Rückgabewert:            < 0 Fehler, 0 OK

Wenn die UID mit der des Transponders übereinstimmt, wird er in den „Selected“-Zustand versetzt und man erhält eine Antwort. Stimmt die UID nicht überein, erhält man keine Antwort.



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 4.3. Transponderinformationen auslesen

**int TSHRW\_ISO\_GetSystemInfo**(int hPort, BYTE \* pBuffer, int BufLen, BYTE \* pUID, int UIDLen, int nRequestFlag)

hPort	Logische Gerätenummer
pBuffer	Zeiger auf "System Info Block" nach ISO Norm 15693
BufLen	Länge des Blocks
pUID	Zeiger auf UID. Das erforderliche Bit im RequestFlag wird automatisch gesetzt, wenn eine UID angegeben wird. Ist der Zeiger NULL, wird die UID ignoriert. Alle UID's können mit TSHRW_ISO_Inventory() ermittelt werden.
UIDLen	Länge des Blocks (8 Byte). Ist die Länge != 8 wird der Parameter pUID ignoriert.
RequestFlag	Request Flag nach ISO Norm 15693. Bei manchen Transpondern kann es notwendig sein, hier zusätzlich das Protocol Extension Flag (ISO15693_PROTOKOLL_EXTENSION = 08H) zu setzen.
Rückgabewert:	< 0 Fehler, Länge der gelesenen Daten

**int TSHRW\_ISO\_GetTagInfo**(int hPort, BYTE \* pUID, int UIDLen, int\* pBlockAnzahl, int\* pBytesProBlock, int\* pMfgCode, int nRequestFlag)

hPort	Logische Gerätenummer
pUID	Zeiger auf UID. Das erforderliche Bit im RequestFlag wird automatisch gesetzt, wenn eine UID angegeben wird. Ist der Zeiger NULL, wird die UID ignoriert. Alle UID's können mit TSHRW_ISO_Inventory() ermittelt werden.
UIDLen	Länge des Blocks (8 Byte). Ist die Länge != 8 wird der Parameter pUID ignoriert.
pBlockAnzahl	Zeiger auf Blockanzahl. Ist Zeiger NULL, wird die Info ignoriert.
pBytesProBlock	Zeiger auf BytesProBlock. Ist Zeiger NULL, wird die Info ignoriert.
pMfgCode	Zeiger auf Manufacturer code. Ist Zeiger NULL, wird die Info ignoriert.
RequestFlag	Request Flag nach ISO Norm 15693
Rückgabewert:	< 0 Fehler, 0 OK

Die Funktion GetTagInfo() liefert spezifische Informationen über den Transponder. Diese Funktion ist KEIN direkter ISO Befehl. Er wurde für Kunden gemacht, die nicht mit der ISO Norm vertraut sind. Es muss nur die UID bekannt sein. (die UID bekommt man mit TSHRW\_ISO\_Inventory()) und man erhält Blockanzahl, Bytes pro Block und Firmenkennung. Intern wird bei Bedarf die Funktion GetSystemInfo verwendet.



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 4.4. Transponder ruhigstellen

**int TSHRW\_ISO\_StayQuiet**(int hPort, BYTE \* pUID, int UIDLen, int nRequestFlag)

hPort                      Logische Gerätenummer

pUID                      Zeiger auf UID. Das erforderliche Bit im RequestFlag wird automatisch gesetzt, wenn eine UID angegeben wird. Ist der Zeiger NULL, wird die UID ignoriert.

Alle UID's können mit TSHRW\_ISO\_Inventory() ermittelt werden.

UIDLen                    Länge des Blocks (8 Byte). Ist die Länge != 8 wird der Parameter pUID ignoriert.

RequestFlag              Request Flag nach ISO Norm 15693

Rückgabewert:            < 0 Fehler, 0 OK

Durch „StayQuiet“ geht der Transponder in eine energiearme Wartestellung. „StayQuiet“ macht nur mit UID Sinn. Denn, es soll exakt nur dieser eine Transponder in Wartestellung gehen. Inventory sieht diesen Transponder dann nicht mehr. Der Zustand wird durch den Befehl: „ResetToReady()“ aufgehoben, oder durch Ansprechen des Transponders über einem Befehl mit UID. z.B. **ReadSingleBlock()** mit UID. Durch Übergabe der UID wird der Transponder gleichzeitig wieder selektiert.

### 4.5. Transponder reaktivieren

**int TSHRW\_ISO\_ResetToReady**(int hPort, BYTE \* pUID, int UIDLen, int nRequestFlag)

hPort                      Logische Gerätenummer

pUID                      Zeiger auf UID. Das erforderliche Bit im RequestFlag wird automatisch gesetzt, wenn eine UID angegeben wird. Ist der Zeiger NULL, wird die UID ignoriert

Alle UID's können mit TSHRW\_ISO\_Inventory() ermittelt werden.

UIDLen                    Länge des Blocks (8 Byte). Ist die Länge != 8 wird der Parameter pUID ignoriert.

RequestFlag              Request Flag nach ISO Norm 15693

Rückgabewert:            < 0 Fehler, 0 OK

Hebt den „Quiet“-Zustand wieder auf.



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 4.6. Einzelnen Block lesen

**int TSHRW\_ISO\_ReadSingleBlock**(int hPort, int BlockNr, BYTE \* pBuffer, int BufLen,  
BYTE \* pUID, int UIDLen, int nRequestFlag)

hPort	Logische Gerätenummer
BlockNr	Nummer des einzulesenden Blocks
pBuffer	Zeiger auf die Transponderdaten
BufLen	Länge der Transponderdaten
pUID	Zeiger auf UID. Das erforderliche Bit im RequestFlag wird automatisch gesetzt, wenn eine UID angegeben wird. Ist der Zeiger NULL, wird die UID ignoriert.
UIDLen	Länge des Blocks (8 Byte). Ist die Länge != 8 wird der Parameter pUID ignoriert.
RequestFlag	Request Flag nach ISO Norm 15693 Bei Transpondern von STM mit mehr als 256 Blöcken, muss hier zusätzlich das Protocol Extension Flag (ISO15693_PROTOKOLL_EXTENSION = 08H) gesetzt werden. Dies ist bei Zugriff auf alle Blöcke eines solchen Transponders notwendig.
Rückgabewert:	< 0 Fehler, Länge der gelesenen Daten

### 4.7. Mehrere Blöcke lesen

**int TSHRW\_ISO\_ReadMultipleBlocks**(int hPort, int nFirstBlock, int nNumberOfBlocks,  
BYTE \* pBuffer, int BufLen,  
BYTE \* pUID, int UIDLen, int nRequestFlag)

Liest mehrere Blöcke auf einmal ein	
hPort	Logische Gerätenummer
nFirstBlock	Nummer des ersten Blocks (0-254)
nNumberOfBlocks	Anzahl der einzulesenden Blöcke
pBuffer	Zeiger auf die Transponderdaten
BufLen	Länge der Transponderdaten
pUID	Zeiger auf UID. Das erforderliche Bit im RequestFlag wird automatisch gesetzt, wenn eine UID angegeben wird. Ist der Zeiger NULL, wird die UID ignoriert
UIDLen	Alle UID's können mit TSHRW_ISO_Inventory() ermittelt werden. Länge des Blocks (8 Byte). Ist die Länge != 8 wird der Parameter pUID ignoriert.
RequestFlag	Request Flag nach ISO Norm 15693 Bei Transpondern von STM mit mehr als 256 Blöcken, muss hier zusätzlich das Protocol Extension Flag (ISO15693_PROTOKOLL_EXTENSION = 08H) gesetzt werden. Dies ist bei Zugriff auf alle Blöcke eines solchen Transponders notwendig.
Rückgabewert:	< 0 Fehler, Länge der gelesenen Daten



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 4.8. Security Status lesen

**int TSHRW\_ISO\_GetMultipleBlockSecurityStatus**(int hPort, int FirstBlock,  
int NumberOfBlocks,  
BYTE \* pReceiveBuffer, int BufLen,  
BYTE \* pUID, int UIDLen,  
int nRequestFlag)

hPort	Logische Gerätenummer
FirstBlock	Startblock
NumberOfBlocks	Blockanzahl
pReceiveBuffer	Zeiger auf Ergebnisbuffer
BufLen	Länge des Ergebnisbuffers
pUID	Zeiger auf UID. Das erforderliche Bit im RequestFlag wird automatisch gesetzt, wenn eine UID angegeben wird. Ist der Zeiger NULL, wird die UID ignoriert. UID's können mit TSHRW_ISO_Inventory() ermittelt werden.
UIDLen	Länge des Blocks (8 Byte). Ist die Länge != 8 wird der Parameter pUID ignoriert.
RequestFlag	Request Flag nach ISO Norm 15693 Bei Transpondern von STM mit mehr als 256 Blöcken, muss hier zusätzlich das Protocol Extension Flag (ISO15693_PROTOKOLL_EXTENSION = 08H) gesetzt werden. Dies ist bei Zugriff auf alle Blöcke eines solchen Transponders notwendig.
Rückgabewert:	< 0 Fehler, Länge der gelesenen Daten

Liefert die BlockSecurity von mehreren Blöcken.

### 4.9. Einzelnen Block schreiben

**int TSHRW\_ISO\_WriteSingleBlock**(int hPort, int BlockNr, BYTE \* pBuffer, int BufLen,  
BYTE \* pUID, int UIDLen, int nRequestFlag)

hPort	Logische Gerätenummer
BlockNr	Nummer des Blocks
pBuffer	Zeiger auf die Transponderdaten
BufLen	Länge der Transponderdaten
pUID	Zeiger auf UID. Das erforderliche Bit im RequestFlag wird automatisch gesetzt, wenn eine UID angegeben wird. Ist der Zeiger NULL, wird die UID ignoriert Alle UID's können mit TSHRW_ISO_Inventory() ermittelt werden.
UIDLen	Länge des Blocks (8 Byte). Ist die Länge != 8 wird der Parameter pUID ignoriert.
RequestFlag	Request Flag nach ISO Norm 15693 Bei Transpondern von STM mit mehr als 256 Blöcken, muss hier zusätzlich das Protocol Extension Flag (ISO15693_PROTOKOLL_EXTENSION = 08H) gesetzt werden. Dies ist bei Zugriff auf alle Blöcke eines solchen Transponders notwendig.
Rückgabewert:	< 0 Fehler, 0 OK



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 4.10. Block sperren

**int TSHRW\_ISO\_LockBlock**(int hPort, int nBlockNr, BYTE \* pUID, int UIDLen, int nRequestFlag)

hPort	Logische Gerätenummer
BlockNr	Nummer des Blocks (0-254)
pUID	Zeiger auf UID. Das erforderliche Bit im RequestFlag wird automatisch gesetzt, wenn eine UID angegeben wird. Ist der Zeiger NULL, wird die UID ignoriert
UIDLen	Alle UID's können mit TSHRW_ISO_Inventory() ermittelt werden. Länge des Blocks (8 Byte). Ist die Länge != 8 wird der Parameter pUID ignoriert.
RequestFlag	Request Flag nach ISO Norm 15693
Rückgabewert:	< 0 Fehler, 0 OK

Sperrt den Block mit dieser Nummer. **Achtung: Das Sperren ist nicht umkehrbar.**

### 4.11. AFI Typ setzen

**int TSHRW\_ISO\_WriteAFI**(int hPort, int AFI, BYTE \* pUID, int UIDLen, int nRequestFlag)

hPort	Logische Gerätenummer
int AFI	AFI
pUID	Zeiger auf UID. Das erforderliche Bit im RequestFlag wird automatisch gesetzt, wenn eine UID angegeben wird. Ist der Zeiger NULL, wird die UID ignoriert
UIDLen	Alle UID's können mit TSHRW_ISO_Inventory() ermittelt werden. Länge des Blocks (8 Byte). Ist die Länge != 8 wird der Parameter pUID ignoriert.
RequestFlag	Request Flag nach ISO Norm 15693
Rückgabewert:	< 0 Fehler, 0 OK

Schreibt „Application family identifier“ (AFI) in den Transponder. Siehe Tabelle: AFI Coding im Normblatt **ISO/IEC 15693 – 3**. z.B: Transport oder Financial Chip.

### 4.12. AFI Typ sperren

**int TSHRW\_ISO\_LockAFI**(int hPort, BYTE \* pUID, int UIDLen, int nRequestFlag)

hPort	Logische Gerätenummer
pUID	Zeiger auf UID. Das erforderliche Bit im RequestFlag wird automatisch gesetzt, wenn eine UID angegeben wird. Ist der Zeiger NULL, wird die UID ignoriert
UIDLen	Alle UID's können mit TSHRW_ISO_Inventory() ermittelt werden. Länge des Blocks (8 Byte). Ist die Länge != 8 wird der Parameter pUID ignoriert.
RequestFlag	Request Flag nach ISO Norm 15693
Rückgabewert:	< 0 Fehler, 0 OK

Setzt den Schreibschutz für das AFI Feld (Application family identifier).



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 4.13. DSFID schreiben

**int TSHRW\_ISO\_WriteDSFID**(int hPort, int DSFID, BYTE \* pUID, int UIDLen,  
int nRequestFlag)

hPort	Logische Gerätenummer
int DSFID	DSFID
pUID	Zeiger auf UID. Das erforderliche Bit im RequestFlag wird automatisch gesetzt, wenn eine UID angegeben wird. Ist der Zeiger NULL, wird die UID ignoriert
UIDLen	Alle UID's können mit TSHRW_ISO_Inventory() ermittelt werden. Länge des Blocks (8 Byte). Ist die Länge != 8 wird UID ignoriert.
RequestFlag	Request Flag nach ISO Norm 15693
Rückgabewert:	< 0 Fehler, 0 OK

Schreibt selbst definierte “data storage format identifier” (DSFID) in den Transponder.

### 4.14. DSFID sperren

**int TSHRW\_ISO\_LockDSFID**(int hPort, BYTE \* pUID, int UIDLen, int nRequestFlag)

hPort	Logische Gerätenummer
pUID	Zeiger auf UID. Das erforderliche Bit im RequestFlag wird automatisch gesetzt, wenn eine UID angegeben wird. Ist der Zeiger NULL, wird die UID ignoriert
UIDLen	Alle UID's können mit TSHRW_ISO_Inventory() ermittelt werden. Länge des Blocks (8 Byte). Ist die Länge != 8 wird der Parameter pUID ignoriert.
RequestFlag	Request Flag nach ISO Norm 15693
Rückgabewert:	< 0 Fehler, 0 OK

Setzt den Schreibschutz für das DSFID Feld (data storage format identifier).



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 4.15. Allgemeiner Zugriff

**int TSHRW\_ISO\_RawRequest**(int hPort, int RequestType, int RequestFlag,  
int Kommando, BYTE \* pParam, int ParamLen, BYTE \* pData,  
int DataLen, BYTE \* pReceive, int ReceiveLen)

hPort	Logische Gerätenummer
RequestType	Request Typ Leserspezifisches Flag
RequestFlag	Request Flag nach ISO Norm 15693
Kommando	ISO Befehl
pParam	Zeiger auf Parameter
ParamLen	Anzahl der Parameter
pData	Zeiger auf Datenfeld
DataLen	Anzahl der Daten
pReceive	Zeiger auf Antwortpuffer
ReceiveLen	Größe des Antwortpuffers
Rückgabewert:	< 0 Fehler, Länge der gelesenen Daten

Mit Hilfe des Raw Request Befehls kann man einen beliebigen Befehl an den ISO Transponder senden. Die Antwort muss dementsprechend selbst ausgewertet werden.  
Siehe Datenblatt: **ISO/IEC 15693 – 3** im Kapitel 7.3 Request Format.





## **5. Befehle für Transponder Typ MIFARE® (ISO14443A)**

Die Befehle für MIFARE® Transponder werden nur bei manchen Geräten (zur Zeit nur TS-HRW38 und TS-HRW32) unterstützt.

### **5.1. Anwendungshinweise**

Es gibt unterschiedliche Arten von MIFARE® Transpondern. In dieser Programmierschnittstelle werden MIFARE® Classic, MIFARE® Ultralight (NFC Type 2) und MIFARE® DESFire Transponder unterstützt.

#### **5.1.1. MIFARE® Ultralight (NFC Type 2)**

Heutzutage sind verschiedene MIFARE® Ultralight kompatible Transponder von unterschiedlichen Herstellern erhältlich. Bitte prüfen sie anhand des Datenblattes des verwendeten Transponders die Speicherkonfiguration.

Original MIFARE® Ultralight Transpondern beinhalten keine Kryptographie-Einheit und sind daher vom Zugriff her einfacher zu verwenden. Die original MIFARE® Ultralight Transponder haben eine Blockstruktur mit 4 Bytes pro Block und insgesamt 16 Blöcke. Wobei nur die Blöcke 4 – 15 benutzerdefiniert verwendet werden können.

Der MIFARE® Ultralight C Transponder hat 44 Blöcke während ein NTAG216F Transponder bis zu 230 Blöcke hat. Alle diese und mehr Transponder werden unterstützt.

Um auf einen MIFARE® Ultralight kompatiblen Transponder zuzugreifen muss zunächst der Transponder selektiert werden.

Dies geschieht mit der Funktionsfolge

TSHRW\_Mifare\_Request()

TSHRW\_Mifare\_Select()

Anschließend kann mit TSHRW\_MifareRead() und TSHRW\_MifareWrite() auf die Blöcke zugegriffen werden.

Die Funktionen TSHRW\_Mifare\_Authenticate() sowie die TSHRW\_MifareGetValue(), TSHRW\_MifareSetValue() und TSHRW\_MifareChangeValue() werden von diesem Chip nicht unterstützt.



## **Programmierschnittstelle (SDK) der TS-HRW Serie**

### **5.1.2. MIFARE® Classic**

Der MIFARE® Classic Chip ist in 2 Varianten erhältlich, als MIFARE® 1K (1KByte EEPROM) und MIFARE® 4K (4KByte EEPROM)

MIFARE® Classic Transponder haben eine Kryptographie Einheit und der Zugriff ist nur nach Authentifizierung möglich.

Zuerst muss auch hier mit der Funktionsfolge

TSHRW\_Mifare\_Request()

TSHRW\_Mifare\_Select()

der Chip selektiert werden.

Anschließend muss der Speicherbereich auf den zugegriffen werden soll mit TSHRW\_MifareAuthenticate() oder TSHRW\_MifareAuthenticateDirect() freigeschaltet werden.

Die Aufrufe unterscheiden sich darin, wie der Zugriff auf den Schlüssel (Key) erfolgt.

Bei AuthenticateDirect wird der Schlüsselwert direkt angegeben, bei Authenticate wird ein vorab mit SetKey gespeicherter Schlüssel verwendet.

Der TS-HW38 kann bis zu je 15 Schlüssel des Schlüsseltyps A und B speichern.

Es ist damit eine effektive Nutzung der Schlüssel ohne die Notwendigkeit des ständigen Vorhaltens der Schlüssel möglich.

Es kann auch vorab der Schlüssel im Gerät gespeichert werden, sodass beim Endbenutzer der Schlüssel nicht offen vorliegen muss.

Die Authentifizierung gilt immer nur für den Sektor für den sie vorgenommen wurde.

Die Zugriffsrechte können pro Sektor eingestellt werden.

Dies wird im Datenblatt der Transponder beschrieben.

### **5.1.3. MIFARE® DESFire**

Die MIFARE® DESFire Transponder sind Mikrocontrollerbasiert. Die Sicherheitsmechanismen können je nach Bedarf verwendet werden.

MIFARE® DESFire unterstützt die verschiedenen Kryptographiemethoden DES, 3DES und AES.

Außerdem kann die Kommunikation offen (plain) oder verschlüsselt (enciphered) durchgeführt werden. Dies wird in der Konfiguration des MIFARE® DESFire Chips festgelegt.

Der MIFARE® DESFire Transponder hat eine eingebautes Dateisystem, es lassen sich mehrere Applikationen mit jeweils mehreren Dateien anlegen. Dies ist grundsätzlich anders zu den MIFARE® Classic und Ultralight Transpondern bei denen nur auf Blöcke zugegriffen werden kann.

Der Zugriff auf MIFARE® DESFire Transponder erfolgt nach ISO14443A-4 Spezifikation.



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 5.2. Funktionsaufrufe MIFARE® allgemein

Diese Funktionen dienen dem Zugriff auf eine MIFARE® (ISO14443A) Karte. Dies ist für alle MIFARE® Varianten gleich. Nach Selektierung der MIFARE® Karte unterscheiden sich dann die Kommandos je nach Kartentyp.

**int TSHRW\_Mifare\_Request(int hPort, WORD \*pATQA)**

hPort                      Logische Gerätenummer  
pATQA                      Zeiger auf die Answer To Request A  
Rückgabewert:            < 0 Fehler, 0 OK

Im ATQA ist nur das niederwertige Byte interessant.

Hier steht:

0x0044:                  MIFARE Ultralight® oder MIFARE® 1K (7BUID) erkannt  
0x0004:                  MIFARE® 1K (4BUID) erkannt  
0x0002:                  MIFARE® 4K (4BUID) erkannt  
0x0042:                  MIFARE® 4K (7BUID) erkannt  
0x0344:                  MIFARE® DESFire erkannt

**int TSHRW\_Mifare\_RequestAll(int hPort, WORD \*pATQA)**

hPort                      Logische Gerätenummer  
pATQA                      Zeiger auf die Answer To Request A  
Rückgabewert:            < 0 Fehler, 0 OK

Im ATQA ist nur das niederwertige Byte interessant.

Hier steht:

0x0044:                  MIFARE Ultralight® oder MIFARE® 1K (7BUID) erkannt  
0x0004:                  MIFARE® 1K (4BUID) erkannt  
0x0002:                  MIFARE® 4K (4BUID) erkannt  
0x0042:                  MIFARE® 4K (7BUID) erkannt  
0x0344:                  MIFARE® DESFire erkannt



## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_Mifare\_Select**(int hPort, BYTE \* pUID, int nUIDLen, BYTE \* pSAK)

hPort	Logische Gerätenummer
pUID	Zeiger auf die UID des Transponders
nUIDLen	Länge der UID
pSAK	Zeiger auf das Select Acknowledge
Rückgabewert:	< 0 Fehler, Länge der gelesenen UID

Dieser Befehl selektiert den ersten gefundenen Chip und liefert seine UID sowie den Select Acknowledge Wert der Aufschluss über den Chiptyp gibt.

Im SAK Wert steht eine Information über den gefundenen Chiptyp:

0x00:	MIFARE Ultralight® mit 7 Byte UID
0x08	MIFARE® 1K
0x18	MIFARE® 4K
0x20	MIFARE® DESFire 7 Byte UID

**int TSHRW\_Mifare\_SelectUID**(int hPort, BYTE \* pUID, int nUIDLen, BYTE \* pSAK)

hPort	Logische Gerätenummer
pUID	Zeiger auf die UID des Transponders
nUIDLen	Länge der UID
pSAK	Zeiger auf das Select Acknowledge
Rückgabewert:	< 0 Fehler, Länge der gelesenen Daten

Dieser Befehl selektiert den Chip mit der übergebenen UID wenn der Chip im Feld ist und liefert den Select Acknowledge Wert der Aufschluss über den Chiptyp gibt.

Im SAK Wert steht eine Information über den gefundenen Chiptyp:

0x00:	MIFARE Ultralight® mit 7 Byte UID
0x08	MIFARE® 1K
0x18	MIFARE® 4K
0x20	MIFARE® DESFire 7 Byte UID

**int TSHRW\_Mifare\_Halt**(int hPort)

hPort	Logische Gerätenummer
Rückgabewert:	< 0 Fehler, 0 OK

Der aktuell selektierte Chip wird gestoppt.

Es kann dann ein bekannter Chip über TSHRW\_Mifare\_SelectUID() aktiviert werden.



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 5.3. Funktionsaufrufe MIFARE® Classic und Ultralight

**int TSHRW\_MifareAuthenticate**(int hPort, BYTE \* pUID, int nUIDLen,  
int KeyType, int nAdresse, int BlockNr)

hPort	Logische Gerätenummer
pUID	Zeiger auf die UID des Transponders oder NULL wenn der selektierte Transponder verwendet werden soll.
nUIDLen	Länge der UID oder 0 wenn der selektierte Transponder verwendet werden soll.
KeyType	Art des Keys 0: KeyA 1: KeyB
nAdresse	Index des Keys im Key-Speicher
BlockNr	Nummer des Blockes für den die Authentifizierung gilt.
Rückgabewert:	< 0 Fehler, 0 OK

Authentifizierung des Sektors in dem der angegebene Block enthalten ist. Als Key wird hier ein gespeicherter Key verwendet. Keys können mit TSHRW\_MifareSetKey() dauerhaft gespeichert werden. MifareAuthenticate wird nur von MIFARE® Classic Transpondern unterstützt.

**int TSHRW\_MifareAuthenticateDirect**(int hPort, BYTE \* pUID, int nUIDLen,  
int KeyType, BYTE \* pKey, int nKeyLen, int BlockNr)

hPort	Logische Gerätenummer
pUID	Zeiger auf die UID des Transponders oder NULL wenn der selektierte Transponder verwendet werden soll.
nUIDLen	Länge der UID oder 0 wenn der selektierte Transponder verwendet werden soll.
KeyType	Art des Keys 0: KeyA 1: KeyB
pKey	Zeiger auf die Daten des Keys
nKeyLen	Länge der Key-Daten, immer 6
BlockNr	Nummer des Blockes für den die Authentifizierung gilt.
Rückgabewert:	< 0 Fehler, 0 OK

Authentifizierung des Sektors in dem der angegebene Block enthalten ist.

Als Key wird hier der übergebene Key verwendet.

MifareAuthenticateDirect wird nur von MIFARE® Classic Transpondern unterstützt.



## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_MifareSetKey**(int hPort, int KeyType, int nAdresse, BYTE \* pKey, int nKeyLen)

hPort	Logische Gerätenummer
KeyType	Art des Keys 0: KeyA 1: KeyB
nAdresse	Index des Keys im Key-Speicher
pKey	Zeiger auf die Daten des Keys
nKeyLen	Länge der Key-Daten, immer 6
Rückgabewert:	< 0 Fehler, 0 OK

Der angegebenen Key wird im Gerät unter der angegebenen Adresse gespeichert.  
Es können max. 15 Keys für KeyA und KeyB gespeichert werden.

**int TSHRW\_MifareRead**(int hPort, int BlockNr, BYTE \* pData, int nDataLen)

hPort	Logische Gerätenummer
BlockNr	Nummer des Blockes der gelesen wird.
pData	Daten des Blockes
nDataLen	Länge der Daten
Rückgabewert:	< 0 Fehler, Länge der gelesenen Daten

Lesen eines Blockes aus dem Transponder. Hierzu muss vorher die Selektierung des Transponders sowie ggf. die Authentifizierung durchgeführt worden sein.

Hier werden immer 16 Byte gelesen. Bei MIFARE Ultralight® beträgt die Blockgröße 4 Bytes, daher werden immer ab dem angegebenen Block die nächsten 4 Blöcke gelesen.

**int TSHRW\_MifareWrite** (int hPort, int BlockNr, BYTE \* pData, int nDataLen)

hPort	Logische Gerätenummer
BlockNr	Nummer des Blockes der geschrieben wird.
pData	Daten des Blockes
nDataLen	Länge der Daten
Rückgabewert:	< 0 Fehler, 0 OK

Schreiben eines Blockes in den Transponder. Hierzu muss vorher die Selektierung des Transponders sowie ggf. die Authentifizierung durchgeführt worden sein.

Die Länge der Daten ist abhängig vom Transpondertyp und beträgt beim MIFARE Ultralight® 4 Bytes und bei MIFARE® 1K und MIFARE® 4K 16 Bytes.



## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_MifareGetValue**(int hPort, int BlockNr, int \*pValue)

hPort                      Logische Gerätenummer  
BlockNr                    Nummer des Blockes der geschrieben wird.  
pValue                     Zeiger auf den Wert  
Rückgabewert:            < 0 Fehler, 0 OK

Lesen eines Blockes im Value Format. Diese Funktion kann nur verwendet werden, wenn der Block im Value Format beschrieben ist. Das Value Format wird nur von MIFARE® 1K und MIFARE® 4K Chips unterstützt.

Hierzu müssen vorher Selektierung sowie Authentifizierung des Transponders durchgeführt worden sein.

**int TSHRW\_MifareSetValue**(int hPort, int BlockNr, int Value)

hPort                      Logische Gerätenummer  
BlockNr                    Nummer des Blockes der geschrieben wird.  
Value                      zu schreibender Wert  
Rückgabewert:            < 0 Fehler, 0 OK

Schreiben eines Blockes im Value Format. Mit dieser Funktion wird der Block im Value Format formatiert. Das Value Format wird nur von MIFARE® 1K und MIFARE® 4K Chips unterstützt.

Hierzu müssen vorher Selektierung sowie Authentifizierung des Transponders durchgeführt worden sein.

**int TSHRW\_MifareChangeValue**(int hPort, int BlockNr, int Richtung, int Differenz)

hPort                      Logische Gerätenummer  
BlockNr                    Nummer des Blockes der geschrieben wird.  
Richtung                   Richtung der Werteänderung 0 = Dekrement, 1 = Inkrement  
Differenz                   Absoluter Wert der Werteänderung  
Rückgabewert:            < 0 Fehler, 0 OK

Ändern eines Blockes im Value Format. Diese Funktion kann nur verwendet werden, wenn der Block im Value Format beschrieben ist. Das Value Format wird nur von MIFARE® 1K und MIFARE® 4K Chips unterstützt. Hierzu müssen vorher Selektierung sowie Authentifizierung des Transponders durchgeführt worden sein.

Je nach eingestellter Richtung wird der Block um die angegebene Differenz inkrementiert oder dekrementiert.

Je nach Zugriffsrechten kann die aktuelle Authentifizierung nur Dekrementieren oder Inkrementieren und Dekrementieren zulassen.



## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_ISO14443A\_Transmit** (int hPort, BYTE \*pSendData, int nSendLen,  
BYTE \* pRecvData, int RecvLen);

hPort	Logische Gerätenummer
pSendData	zu sendende Daten
nSendLen	Länge der zu sendenden Daten
pRecvData	empfangene Daten
RecvLen	max. Länge der empfangenen Daten
Rückgabewert:	< 0 Fehler, Länge der empfangenen Daten

Hiermit können beliebige Befehle an eine aktivierte ISO14443A Karte gesendet werden. (Nur bei TS-HRW380 verfügbar)

### 5.4. Funktionsaufrufe ISO14443A-4

Nach der Selektierung der Karten gemäß ISO14443A-3 Kapitel 5.2. muss vor Zugriff auf die ISO14443A-4 Kommandos noch der ISO14443A-4 Kommandolevel aktiviert werden. Hierzu dienen die Aktivierungsbefehle in Kapitel 5.4.1.

#### 5.4.1. ISO14443A-4 Aktivierung

**int TSHRW\_ISO14443SetCID** (int hPort, int CID)

hPort	Logische Gerätenummer
CID	CardIdentifier, Identifikationsnummer der Karte, wird immer nur eine Karte aktiviert, so kann dies immer auf 1 bleiben.
Rückgabewert:	< 0 Fehler, 0 OK

Mit dieser Funktion wird der CardIdentifier gesetzt, der bei allen folgenden Befehlen Verwendung findet. Es können mehrere Karten selektiert sein, auf die mit unterschiedlichen Card Identifiern zugegriffen wird.

**int TSHRW\_ISO14443Rats** (int hPort, BYTE \* pData, int RecvBufLen)

hPort	Logische Gerätenummer
pData	Answer To Select Daten
RecvBufLen	Max. Länge des Datenpuffers
Rückgabewert:	< 0 Fehler, Länge der empfangenen Daten

Diese Funktion dient der Abfrage der Selektierungsantwort (Request for Answer to Select)  
Die Bedeutung der Bytes in der Selektierungsantwort ist im MIFARE® DESFire Datenblatt beschrieben.





## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_ISO14443PPS** (int hPort, int Speed)

hPort                      Logische Gerätenummer  
Speed                      gewünschte Übertragungsgeschwindigkeit  
Rückgabewert:            < 0 Fehler, 0 OK

Hiermit werden die Parameter der Übertragung definiert.

Es kann damit die Übertragungsgeschwindigkeit PCD → PICC und PICC → PCD definiert werden.

Es gilt folgende Tabelle:

Speed	PCD → PICC	PICC → PCD	Speed	PCD → PICC	PICC → PCD
0 (0000)	106 kBit	106 kBit	8 (1000)	106 kBit	424 kBit
1 (0001)	212 kBit	106 kBit	9 (1001)	212 kBit	424 kBit
2 (0010)	424 kBit	106 kBit	10 (1010)	424 kBit	424 kBit
3 (0011)	848 kBit	106 kBit	11 (1011)	848 kBit	424 kBit
4 (0100)	106 kBit	212 kBit	12 (1100)	106 kBit	848 kBit
5 (0101)	212 kBit	212 kBit	13 (1101)	212 kBit	848 kBit
6 (0110)	424 kBit	212 kBit	14 (1110)	424 kBit	848 kBit
7 (0111)	848 kBit	212 kBit	15 (1111)	848 kBit	848 kBit

alle weiteren Werte sind ungültig.

**int TSHRW\_ISO14443Transmit** (int hPort, BYTE \*pSendData, int nSendLen,  
BYTE \*pRecvData, int RecvLen)

hPort                      Logische Gerätenummer  
pSendData                zu sendende Daten  
nSendLen                Länge der zu sendenden Daten  
pRecvData                empfangene Daten  
RecvLen                max. Länge der empfangenen Daten  
Rückgabewert:            < 0 Fehler, Länge der empfangenen Daten

Hiermit können beliebige Befehle an aktivierte ISO14443-4 Karten gesendet werden.

Alle Befehle für MIFARE® DESFire Karten können über diese Funktion abgebildet werden.



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 5.5. Funktionsaufrufe ISO14443B

Diese Funktionen werden momentan nur von TS-HRW32 unterstützt.

**int TSHRW\_ISO14443B\_Request** (int hPort, BYTE \* pUID, int UIDLen,  
BYTE \* pAppData, int AppDataLen,  
BYTE \* pProtocolInfo, int ProtocolInfoLen);

hPort	Logische Gerätenummer
pUID	Zeiger auf den PUPI des Transponders
UIDLen	max. Länge des PUPI
pAppData	Zeiger auf Applikationsdaten
AppDataLen	max. Länge der Applikationsdaten
pProtocolInfo	Zeiger auf Protokollinformation
ProtocolInfoLen	max. Länge der Protokollinformation
Rückgabewert:	< 0 Fehler, 0 OK

Hiermit wird der REQB Request an den Transponder gesendet und die ATQB Antwort empfangen.  
pUID erhält die 4 Byte PUPI (Pseudo Unique PICC Identifier)  
pAppData erhält die 4 Byte Application Data mit AFI, CRC\_B und Applikationsanzahl  
pProtocolInfo enthält 3 Byte Protokollinformation.  
In der Dokumentation zu ISO14443-3 ist dies näher beschrieben.

**int TSHRW\_ISO14443B\_Wakeup** (int hPort, BYTE \* pUID, int UIDLen,  
BYTE \* pAppData, int AppDataLen,  
BYTE \* pProtocolInfo, int ProtocolInfoLen);

hPort	Logische Gerätenummer
pUID	Zeiger auf den PUPI des Transponders
UIDLen	max. Länge des PUPI
pAppData	Zeiger auf Applikationsdaten
AppDataLen	max. Länge der Applikationsdaten
pProtocolInfo	Zeiger auf Protokollinformation
ProtocolInfoLen	max. Länge der Protokollinformation
Rückgabewert:	< 0 Fehler, 0 OK

Hiermit wird der WUPB Request an den Transponder gesendet und die ATQB Antwort empfangen.  
pUID erhält die 4 Byte PUPI (Pseudo Unique PICC Identifier)  
pAppData erhält die 4 Byte Application Data mit AFI, CRC\_B und Applikationsanzahl  
pProtocolInfo enthält 3 Byte Protokollinformation.  
In der Dokumentation zu ISO14443-3 ist dies näher beschrieben.



## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_ISO14443B\_Select** (int hPort, int Speed, BYTE \* pUID, int UIDLen);

hPort                      Logische Gerätenummer  
 Speed                     Geschwindigkeitseinstellung  
 pUID                        Zeiger auf den PUPI des Transponders  
 UIDLen                    max. Länge des PUPI  
 Rückgabewert:          < 0 Fehler, Länge der empfangenen Daten

Hiermit wird das ATTRIB Kommando an den Transponder gesendet.

Mit Speed wird die Übertragungsgeschwindigkeit PCD → PICC und PICC → PCD definiert werden.

Es gilt folgende Tabelle:

Speed	PCD → PICC	PICC → PCD	Speed	PCD → PICC	PICC → PCD
0 (0000)	106 kBit	106 kBit	8 (1000)	106 kBit	424 kBit
1 (0001)	212 kBit	106 kBit	9 (1001)	212 kBit	424 kBit
2 (0010)	424 kBit	106 kBit	10 (1010)	424 kBit	424 kBit
3 (0011)	848 kBit	106 kBit	11 (1011)	848 kBit	424 kBit
4 (0100)	106 kBit	212 kBit	12 (1100)	106 kBit	848 kBit
5 (0101)	212 kBit	212 kBit	13 (1101)	212 kBit	848 kBit
6 (0110)	424 kBit	212 kBit	14 (1110)	424 kBit	848 kBit
7 (0111)	848 kBit	212 kBit	15 (1111)	848 kBit	848 kBit

alle weiteren Werte sind ungültig.

In pUID wird der PUPI übertragen, der von TSHRW\_ISO14443B\_Request übermittelt wurde.

**int TSHRW\_ISO14443B\_Transmit** (int hPort, BYTE \*pSendData, int nSendLen,  
 BYTE \* pRecvData, int RecvLen);

hPort                      Logische Gerätenummer  
 pSendData                zu sendende Daten  
 nSendLen                Länge der zu sendenden Daten  
 pRecvData                empfangene Daten  
 RecvLen                 max. Länge der empfangenen Daten  
 Rückgabewert:          < 0 Fehler, Länge der empfangenen Daten

Hiermit können beliebige Befehle an eine aktivierte ISO14443B Karte gesendet werden.



## **Programmierschnittstelle (SDK) der TS-HRW Serie**

### **5.6. Funktionsaufrufe Mifare DESFire**

Zur Verwendung der DESFire-Funktionalität muss die Karte aktiviert sein, siehe die Beschreibung in Kapitel 5.2 und 5.4.

Die DESFire-Karte benutzt ein flexibles Dateisystem. Es besteht aus Ordnern, die Applikationen genannt werden, und Dateien. Das Dateisystem erlaubt maximal 28 Applikationen. Jede Applikation kann bis zu 32 Dateien beinhalten. Es gibt fünf verschiedene Dateitypen. Eine Applikation ist kein ausführbares Programm. Eine Applikation ist ein Ordner mit Dateien für Daten, die von einem Programm verwendet werden, das auf einem externen Gerät läuft (z.B. PC) und über das Kartenlesegerät mit der Karte kommunizieren kann.

Um die Einstellungen der Karte zu ändern oder eine Applikation zu erzeugen, muss die Kartenebene (Applikation ID 0x00) selektiert sein.

Um die Einstellungen einer Applikation zu ändern oder ein File zu erzeugen oder darauf zuzugreifen, muss die zugehörige Applikation selektiert sein.

Nach der Aktivierung der Karte nach ISO14443-4 ist die DESFire-Kartenebene selektiert.

Vor der Datenübermittlung kann eine Authentifizierung durchgeführt werden. Nach der Authentifizierung wissen beide Seiten, sowohl die Karte als auch das Lesegerät, dass sie einander vertrauen können, weil sie beide den geheimen Schlüssel kennen mussten.

Die Authentifizierung erzeugt außerdem einen Sitzungsschlüssel, der automatisch sowohl in der Karte auch als in der DESFire-SDK gespeichert wird. Mit ihm kann die weitere Kommunikation abgesichert werden.

Jede weitere Kommunikation kann auf drei verschiedene Weisen erfolgen:

- Es werden nur die reinen unverschlüsselten Daten verschickt
- Es werden die unverschlüsselten Daten mit einer verschlüsselten Checksumme (CMAC) verschickt, um ein unbemerktes Austauschen der Karte oder des Lesegerätes oder eine Man-In-The-Middle-Attacke zu verhindern.
- Es werden die Daten verschlüsselt verschickt, um ein unbemerktes Austauschen der Karte oder des Lesegerätes oder eine Man-In-The-Middle-Attacke sowie zusätzlich unerwünschtes Mithören zu verhindern.

Die Verschlüsselungsart kann entweder sein:

- DES mit 8 Byte, oder 16 Byte mit zwei gleichen Hälften. Das niederwertigste Bit in jedem Byte wird nur für die Schlüsselversion verwendet
- 3DES mit 16 Byte. Das niederwertigste Bit in jedem Byte wird nur für die Schlüsselversion verwendet und bei der Verschlüsselung ignoriert.
- AES mit 16 Byte.

Es ist möglich, die Karte und die Applikationen so zu konfigurieren, dass sämtliche Änderungen der Einstellungen, das Erzeugen/Löschen von Applikationen und Dateien, das Lesen/Schreiben von Dateien zwingend eine vorhergehende Authentifizierung benötigen und dass alle Daten verschlüsselt übermittelt werden.



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 5.6.1. Sicherheitsrelevante Befehle

Abhängig von den Sicherheitseinstellungen muss sich ein Lesegerät authentisieren, bevor es auf der Karte die Architektur, Daten oder Einstellungen ändern kann. Nach einer Authentifizierung wissen beide Seiten, die Karte wie auch das Lesegerät, dass sie einander vertrauen können. Außerdem wird ein Sitzungsschlüssel erzeugt, mit dem die weitere Kommunikation abgesichert werden kann.

Eine Authentifizierung wird ungültig, wenn:

- Eine andere Applikation selektiert wird

- Der Key verändert wird, mit dem die Authentifizierung durchgeführt wurde

- Ein Kommando fehlschlägt

Der Sicherheitsmechanismus benutzt intern DES/3DES oder AES Verschlüsselung.

Schlüssel mit DES oder 3DES Verschlüsselung haben eine Länge von 16 Byte.

Das niederwertigste Bit in jedem Byte wird für die Schlüsselversion verwendet und bei der Verschlüsselung ignoriert. Es ist empfehlenswert, für die Version nicht die Parity-Bits einer Verschlüsselung zu verwenden.

Schlüssel für AES-Verschlüsselung haben eine Länge von 16 Byte mit separater Version.

Die Version eines Schlüssels wird mit dem Key auf der Karte gespeichert und kann für eigene Zwecke verwendet werden, die Karte selbst analysiert die Version nie.

Die Sicherheitsbefehle werden in der Kartenebene und in der Applikationsebene verwendet.

**int TSHRW\_Desfire\_Authenticate** (int hPort, BYTE KeyNo, BYTE\* pKey, int KeySize)

hPort                      Logische Gerätenummer

KeyNo                     Schlüsselnummer

pKey                        Zeiger auf Buffer mit Schlüssel. Der Schlüssel hat eine Länge von 16 Byte.

KeySize                   Länge des Schlüssels

Rückgabewert:           -1 Fehler, 0 OK

Authentifiziert mit dem Schlüssel mit der angegebenen Schlüsselnummer.

Die aktuell selektierte Applikation muss die Verschlüsselungsart DES oder 3DES verwenden.

Dieser Authentifizierungstyp dient zur Rückwärtskompatibilität mit älteren Desfire-Karten.

Es wird empfohlen, nach Möglichkeit AuthenticateIso zu verwenden.

Wenn der Wert in pKey falsch war, liefert TSHRW\_GetLastError den Fehler 100AE, „Der aktuelle Authentifizierungszustand erlaubt den Befehl nicht“.



## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_Desfire\_AuthenticateIso** (int hPort, BYTE KeyNo, BYTE\* pKey, int KeySize)  
hPort                      Logische Gerätenummer  
KeyNo                      Schlüsselnummer  
pKey                      Zeiger auf Buffer mit Schlüssel. Der Schlüssel hat eine Länge von 16 Byte.  
KeySize                    Länge des Schlüssels  
Rückgabewert:            -1 Fehler, 0 OK

Authentifiziert mit dem Schlüssel mit der angegebenen Schlüsselnummer.

Dieser Authentifizierungstyp wird verwendet, wenn die aktuell selektierte Applikation die Verschlüsselungsart DES oder 3DES benutzt.

Wenn der Wert in pKey falsch war, liefert TSHRW\_GetLastError den Fehler 100AE, „Der aktuelle Authentifizierungszustand erlaubt den Befehl nicht“.

**int TSHRW\_Desfire\_AuthenticateAES** (int hPort, BYTE KeyNo, BYTE\* pKey, int KeySize)  
hPort                      Logische Gerätenummer  
KeyNo                      Schlüsselnummer  
pKey                      Zeiger auf Buffer mit Schlüssel. Der Schlüssel hat eine Länge von 16 Byte.  
KeySize                    Länge des Schlüssels  
Rückgabewert:            -1 Fehler, 0 OK

Authentifiziert mit dem Schlüssel mit der angegebenen Schlüsselnummer.

Dieser Authentifizierungstyp wird verwendet, wenn die aktuell selektierte Applikation die Verschlüsselungsart AES benutzt.

Wenn der Wert in pKey falsch war, liefert TSHRW\_GetLastError den Fehler 100AE, „Der aktuelle Authentifizierungszustand erlaubt den Befehl nicht“.



## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_Desfire\_GetKeySettings** (int hPort,  
BYTE\* pKeySettings, int KeySettingsLen)

hPort	Logische Gerätenummer
pKeySettings	Byte 0: Verschlüsselungsverfahren in der Application (z.B. bei Authenticate(), CMAC), liefert auf Kartenebene immer 0 0: (3)DES 1: 3K3DES 2: AES Byte 1: maximale Anzahl von Schlüsseln in der Applikation Byte 2: Nummer des Keys, mit dem authentifiziert werden muss, um einen Key ändern zu können 0: mit Master Key (MK, Hauptschlüssel), 1 bis 13: mit diesem Key (außer bei MK und dem Key selbst), 14: mit dem Key, der geändert werden soll, 15: alle Keys außer dem Master Key sind unveränderlich Byte 3: Bit 0: Master Key ist veränderbar, Bit 1: File-Listen und App-Settings ohne App-Master-Key-Authent App-Listen und Karten-Settings ohne Karten-Master-Key-A. Bit 2: File erzeugen/löschen ohne App-Master-Key-Authent App erzeugen ohne Karten-Master-Key-Authent App löschen auch mit App-Master-Key-Authent Bit 3: Konfiguration veränderbar
SettingsLen	Länge von pKeySettings
Rückgabewert:	-1 Fehler, 0 OK

Liefert die Schlüssel-Sicherheitseinstellungen und maximale Schlüsselanzahl der aktuell selektierten Applikation bzw der Karte wenn die Kartenebene selektiert ist.

**int TSHRW\_Desfire\_ChangeKeySettings** (int hPort, BYTE ChangeKeyAuthentWith,  
BYTE OtherKeySettings)

hPort	Logische Gerätenummer
ChangeKeyAuthentWith	Nummer des Keys, mit dem authentifiziert werden muss, um einen Key ändern zu können. 0: mit Master Key (MK, Hauptschlüssel), 1 bis 13: mit diesem Key (außer bei MK und dem Key selbst) 14: mit dem Key, der geändert werden soll, 15: alle Keys außer dem Master Key sind unveränderlich
OtherKeySettings	Bit0: Master Key ist veränderbar Bit1: File/App-Listen und -Settings ohne Master-Key-Authentifizierung Bit2: File/App erzeugen/löschen ohne Master-Key-Authentifizierung Bit3: Konfiguration änderbar
Rückgabewert:	-1 Fehler, 0 OK

Setzt die Schlüssel-Sicherheitseinstellungen der aktuell selektierten Applikation (beziehungsweise der Karte, wenn Kartenebene selektiert). Die maximale Schlüsselanzahl der Applikation kann nach der Erzeugung nicht mehr geändert werden.





## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_Desfire\_ChangeKey** (int hPort, BYTE KeyNo,  
BYTE\* pNewKey, int NewKeyLen, BYTE NewKeyVersion,  
BYTE\* pCurrentKey, int CurrentKeyLen,  
BYTE NewMasterKeyCrypto)

hPort	Logische Gerätenummer
KeyNo	Nummer des Keys, dessen Wert geändert werden soll
pNewKey	Neuer Key-Wert. 16 Byte lang. Bei DES/3DES bereits mit KeyVersion
NewKeyLen	Länge des Schlüssels in pNewKey
NewKeyVersion	Version des Keys, für AES, bei DES/3DES ignoriert
pCurrentKey	Aktueller Inhalt von Schlüssel mit KeyNo. 16 Byte. Nur von Bedeutung wenn KeyNo anders als die Schlüsselnummer zur notwendigen Authentifizierung.
CurrentKeyLen	Länge des Schlüssels in pCurrentKey
MasterKeyCrypto	Nur verwendet, wenn Kartenebene selektiert (AID==0x00). Setzt das Verschlüsselungsverfahren des Karten-Hauptschlüssels 0: DES/3DES, 2: AES
Rückgabewert:	-1 Fehler, 0 OK

Ändert den Inhalt des Schlüssels und die Schlüsselversion. Wenn die Kartenebene selektiert ist, dann wird der Karten-Hauptschlüssel verändert. Wenn eine Applikation selektiert ist, dann wird ein Schlüssel der Applikation verändert.

Der Verschlüsselungstyp des Karten-Hauptschlüssels kann von 3DES nach AES geändert werden, aber nicht umgekehrt. Auf Applikationsebene wird der angegebene Schlüssel innerhalb der aktuell selektierten Applikation verändert, der Verschlüsselungstyp der Applikation ist nicht mehr veränderbar.

Vor dem Befehl ist eine Authentifizierung notwendig. Die Schlüsselnummer für die Authentifizierung ist abhängig von den Schlüssel-Sicherheitseinstellungen der Applikation. Wenn der Wert in pCurrentKey falsch war, liefert TSHRW\_GetLastError den Fehler 1001E, „Integritätsfehler, CRC oder MAC passen nicht zu den Daten“.

**int TSHRW\_Desfire\_GetKeyVersion** (int hPort, BYTE KeyNo)

hPort	Logische Gerätenummer
KeyNo	Nummer des Schlüssels
Rückgabewert:	-1 Fehler, >=0 Version des Schlüssels, 1 Byte

Liefert die Version des Keys. Es ist keine vorhergehende Authentifizierung notwendig.





## Programmierschnittstelle (SDK) der TS-HRW Serie

### 5.6.2. Befehle auf Karten-Ebene

Um diese Befehle auszuführen muss vorher die Kartenebene aktiviert worden sein

**int TSHRW\_Desfire\_GetVersion** (int hPort, BYTE\* pVersionData, int VersionDataLen)

hPort                      Logische Gerätenummer

pVersionData            Zeiger auf Buffer für Versionsdaten, 28 Byte  
Bytes 0 bis 6 Hardware-bezogene Informationen.

Byte 0: Hersteller-ID

Byte 1: Typ

Byte 2: Untertyp

Byte 3: Hauptversion

Byte 4: Unterversion

Byte 5: Speichergröße (z.B. 0x16 == 2048 Byte,  
0x18 == 4096 Byte, 0x1a == 8192 Byte)

Byte 6: Kommunikations-Protokolltyp

Bytes 7 bis 13 Software-bezogene Informationen.

Byte 7: Hersteller-ID

Byte 8: Typ

Byte 9: Untertyp

Byte 10: Hauptversion

Byte 11: Unterversion

Byte 12: Speichergröße (z.B. 0x16 == 2048 Byte,  
0x18 == 4096 Byte, 0x1a == 8192 Byte)

Byte 13: Kommunikations-Protokolltyp

Bytes 14 bis 20: UID der Karte

Bytes 21 bis 25: Produktions-Chargennummer

Byte 26: Kalenderwoche der Produktion (BCD, binary coded decimal)

Byte 27: Jahr der Produktion (BCD)

VersionDataLen        Länge von pVersionData

Rückgabewert:        -1 Fehler, 0 OK

Liest Eigenschaften des Transponders (28 Bytes).

**int TSHRW\_Desfire\_GetFreePICCMemory** (int hPort)

hPort                      Logische Gerätenummer

Rückgabewert:        -1 Fehler, 0>=0 Größe des Nutzdatenspeichers in Byte

Liefert die Speichergröße der Karte.



## Programmierschnittstelle (SDK) der TS-HRW Serie

**bool TSHRW\_Desfire\_FormatPICC** (int hPort)

hPort                      Logische Gerätenummer

Rückgabewert:            -1 Fehler, 0 OK

Gibt den verwendeten Nutzer-Speicher auf der Karte für neuen Gebrauch frei. Löscht alle Applications und Files. Das Kommando kann nicht rückgängig gemacht werden.

Gibt die Speicherbereiche der bereits früher gelöschten Applications und Files frei. Der Hauptschlüssel der Karte und die Sicherheitseinstellungen der Karte bleiben unverändert.

Der Befehl benötigt eine vorhergehende Authentifizierung mit dem Hauptschlüssel der Karte.

**int TSHRW\_Desfire\_GetCardUID** (int hPort, BYTE\* pUID, int UIDBufferLen)

hPort                      Logische Gerätenummer

pUIDBuffer                Zeiger auf Buffer für UID

UIDBufferLen             Länge von pUIDBuffer

Rückgabewert:            -1 Fehler, 0 OK

Liefert die UID der Karte. Diese hat eine Länge von 7 Byte.

Das Kommando wird benötigt, wenn die Karte auf Random-ID (zufällige UID) gesetzt ist.

Auf einer Karte ohne Random-ID ist das Kommando nicht empfehlenswert.



## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_Desfire\_SetConfiguration**(int hPort, BYTE Option,  
BYTE\* pSettings, int SettingsLen)

hPort	Logische Gerätenummer
Option	0: setzen des Konfigurationsbytes, 1: setzen des Default-Schlüssels, 2: setzen des ATS (Answer to Select)
pSettings	Zeiger auf Settings. Option=0: Konfigurationsbyte Bit 0 = 0: Formatieren der Karte ist möglich. Bit 0 = 1: Formatieren der Karte ist nicht möglich (kann nicht zurückgesetzt werden), Bit 1 = 0: Random ID aus Bit 1 = 1: Random ID an (kann nicht zurückgesetzt werden) Option=1: Default-Schlüssel mit Version, 17 Byte Byte 0-15: neuer Default-Schlüssel Byte 16 Schlüsselversion (3DES und AES). Option=2: benutzerdefinierter ATS (Answer to Select)
SettingsLen	Länge von pSettings
Rückgabewert:	-1 Fehler, 0 OK

Ändert die Konfiguration der Karte.

Das Kommando bietet drei verschiedene Optionen an:

Wenn „Random ID“ aktiviert ist, dann liefert die Karte beim TSHRW\_Mifare\_Select Kommando eine zufällig generierte UID. Um die echte UID der Karte zu lesen, muss das Kommando TSHRW\_Desfire\_GetCardUID verwendet werden, welches eine vorhergehende Authentifizierung erfordert.

Der Default-Schlüssel wird als Initialisierungswert für alle Schlüssel in neu erzeugten Applikation verwendet.

Mit Option = 2 kann die Antwort auf RATS gesetzt werden (answer to select).

Dies sollte nur von Experten durchgeführt werden!

Sehen Sie bitte die MIFARE-DESFIRE-Funktions-Spezifikation für detaillierte Informationen.



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 5.6.3. Befehle auf Kartenebene zur Applikationsverwaltung

Eine DESFire-Karte kann bis zu 28 Applikationen besitzen.

Eine Applikation beinhaltet Files, bis zu 32 Files sind in einer Applikation möglich.

Eine Applikation stellt Funktionalität zur Verfügung, um den Zugriff auf ihre Files abzusichern.

Die meisten Applikationsbefehle erwarten dass die Karten-Ebene selektiert ist.

Für einige Befehle ist eine vorhergehende Authentifizierung notwendig, dies ist abhängig von der Einstellung für den Schlüssel auf Karten-Ebene, den Master Key.

**int TSHRW\_Desfire\_GetApplicationIds** (int hPort, BYTE\* pAppIds, int AppIdsLen)

hPort                      Logische Gerätenummer

pAIDBuffer                Zeiger auf Puffer für die AIDs, es wird ein Feld von AIDs geschrieben mit 4  
Byte pro AID

AIDBufferLen             Länge in Byte des Buffers für die AIDs

Rückgabewert:            -1 Fehler, >=0 Anzahl der AIDs in pAIDBuffer

Liefert von allen Applikationen auf der Karte die Applikations ID (AID), abgesehen von der Kartenebene (AID==0). Für das Kommando muss die Kartenebene selektiert sein.

**int TSHRW\_Desfire\_GetApplicationIdsAndNames** (int hPort,  
BYTE\* pAppNamesArray,  
int AppNamesArrayLen)

hPort                      Logische Gerätenummer

pAppNamesArray          Zeiger auf Puffer für ein Feld von Applikation-ID-Datensätzen,  
für jede Applikation ein Datensatz 22 Byte:

Byte 0-2                  AID

Byte 3-4                  ISO AID

Byte 5-20                DF-Name

Byte 21                  Länge des DF-Names

AppNamesArrayLen        Länge in Byte von pAppNamesArray

Rückgabewert:            -1 Fehler, >=0 Anzahl Bytes genutzt in pAppNamesArray

Liefert für jede Application auf der Karte drei verschiedene Application-Identifizier.

Der DESFire spezifische Application-Identifizier (AID), ISO7816-4 Application-Identifizier (ISO AID) und ISO7816-4 DF-Name (DF-Name).

Applications ohne DF-Name werden ignoriert.

Der Befehl kann nicht in einem aktuellen Authentifizierungsstatus nach Authentifizierung mit AuthenticateISO oder AuthenticateAES verwendet werden.

**int TSHRW\_Desfire\_SelectApplication** (int hPort, int AppId)

Selektiert eine Applikation. Eine Selektion mit AppId 0 selektiert die Kartenebene.

hPort	Logische Gerätenummer
AppId	Applikation-ID. 3 Bytes. Muss auf der Karte einzigartig sein. 0 ist für die Kartenebene reserviert.
KeyCrypto	Verschlüsselungsverfahren in der Applikation (z.B. bei Authenticate(), CMAC) 0: DES/3DES, 2: AES
MaxNoOfKeys	Anzahl der Keys (maximal 14)
bFilesWithIsoID	0: Files mit ISO7816 ID sind nicht möglich. 1: später erzeugte Files dieser Applikation müssen eine ISO ID besitzen. Unabhängig davon, ob die Applikation mit ISO7816-Application-ID.
ChangeKeyAuthentWith	gibt an, mit welchem Key authentifiziert werden muss, um einen Key ändern zu können (change key). 0: mit Master Key (MK) 1 bis 13: mit diesem Key (change key) (außer bei MK & change key selbst, MK und change key benötigen Authent mit Master Key) 14: mit dem Key, der geändert werden soll 15: alle Keys außer Master Key sind unveränderlich
OtherKeySettings	Bit 0: Master Key ist veränderbar, Bit 1: File-Listen und -Settings ohne Master-Key-Authentifizierung Bit 2: File erzeugen/löschen ohne Master-Key-Authentifizierung Bit 3: Konfiguration veränderbar
pIso7816Data	Zeiger auf ISO7816-Application-ID, NULL: nicht verwenden
Iso7816DataLen	2 Byte ISO-Applikation-ID + DF-Name (kann leer sein)
Rückgabewert:	Länge von ISO7816-Application-ID, 0: nicht verwenden -1 Fehler, 0 OK

Seite 61 von 96  
Mai 2024



## **Programmierschnittstelle (SDK) der TS-HRW Serie**

**int TSHRW\_Desfire\_DeleteApplication** (int hPort, int AppId)

hPort                      Logische Gerätenummer

AppId                      Applikation-ID

Rückgabewert:            -1 Fehler, 0 OK

Löscht eine Applikation mit allen zugehörigen Files.

Der Befehl erfordert, dass die Kartenebene selektiert ist.

Die Speicherbereiche der gelöschten Applikation und der gelöschten Files wird nicht freigegeben.

Das Freigeben von Speicher kann nur durch Formatierung der Karte erreicht werden.



## **Programmierschnittstelle (SDK) der TS-HRW Serie**

### **5.6.4. Befehle auf File-Ebene**

Alle Befehle auf File-Ebene beziehen sich nur auf die Files der aktuell selektierten Applikation. Ein File wird über seinen File-Identifizier (File-ID) mit Länge von einem Byte angesprochen, dieser muss innerhalb der zugehörigen Applikation eindeutig sein.

Für einige Befehle ist eine vorhergehende Authentifizierung notwendig, dies ist abhängig von Einstellungen in der zugehörigen Applikation, deren Key-Settings.

Es gibt fünf verschiedene File-Typen:

- Standard Data Files und Backup Data Files speichern unformatierte Daten
- Backup Data Files besitzen zusätzlich einen internen Backup-Mechanismus, siehe unten
- Value Files werden verwendet, um eine 32bit Zahl zu speichern und zu verändern
- Linear Record Files und Cyclic Record Files werden verwendet, um Daten strukturiert zu speichern. Sie bestehen aus einem Feld von Datensätzen (Records) und einem aktuellen Datensatz. Cyclic und Linear Record Files haben nur einen Unterschied: wenn alle Datensätze gefüllt sind und ein zusätzlicher Datensatz übernommen werden soll, so liefert ein Linear Record File einen Fehler, ein Cyclic Record File überschreibt den ältesten Datensatz.

Alle File-Typen außer Standard Data Files arbeiten mit einem internen Backup-Mechanismus. Ein Schreibvorgang verändert ein Spiegel-File aber nicht das File selbst, ein folgender Lesevorgang würde den ursprünglichen Wert lesen.

Eine CommitTransaction wird benötigt, um die Änderungen zu bestätigen. Die Selektierung einer anderen Applikation ohne vorhergehender CommitTransaction, das Entfernen der Karte ohne vorhergehender CommitTransaction oder der Aufruf von AbortTransaction machen alle Änderungen seit der letzten CommitTransaction ungültig und die ursprünglichen Daten bleiben erhalten.



## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_Desfire\_GetFileIds** (int hPort, BYTE\* pFileIds, int FileIdsLen)

hPort	Logische Gerätenummer
pFileIds	Zeiger auf Buffer für ein Feld von IDs, 1 Byte pro ID
FileIdsLen	Länge des Buffers für die IDs
Rückgabewert:	-1 Fehler, >=0 Anzahl der IDs in pFileIds

Liefert die IDs aller Files in der aktuell selektierten Applikation.

**int TSHRW\_Desfire\_GetFileIsoIds** (int hPort, BYTE\* pFileIds, int FileIdsLen)

hPort	Logische Gerätenummer
pFileIds	Zeiger auf Buffer für die ISO IDs, 2 Byte pro ID
FileIdsLen	Länge des Buffers für die ISO IDs
Rückgabewert:	-1 Fehler, >=0 Anzahl der ISO IDs in pFileIds

Liefert die File-Identifizier nach ISO7816-4 (ISO File ID) von allen Files innerhalb der aktuell selektierten Applikation. Files ohne ISO File ID werden ignoriert.





## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_Desfire\_GetFileSettings** (int hPort, int FileId,  
BYTE\* pFileSettings, int FileSettingsBufferLen)

hPort	Logische Gerätenummer
FileId	File ID
pFileSettings	Byte 0: File-Typ 0: Standard Data File, 1: Backup Data File, 2: Value File, 3: Linear Record File, 4: Cyclic Record File Byte 1: Art der Kommunikation mit dem File 0: unverschlüsselt 1: unverschlüsselt, gesichert mit MAC 3: verschlüsselt Bytes 2-3: Zugriffsrechte Bits 0-3: Recht zum Ändern der Zugriffsrechte Bits 4-7: Lese- & Schreibrecht Bits 8-11: Schreibrecht Bits 12-15: Leserecht für jedes Zugriffsrecht: 0x0 - 0xd: Schlüsselnummer für vorhergehende Authentifizierung 0xe: Freier Zugriff 0xf: Kein Zugriff Für Standard und Backup Data File: Byte 4-6: nutzbare File-Größe Für Value File: Byte 4-7: untere Grenze Byte 8-11: obere Grenze Byte 12-15: aktueller maximaler Wert für LimitedCredit Byte 16: 0x00: LimitedCredit-Befehl erlaubt 0x01: LimitedCredit-Befehl deaktiviert Für Linear und Cyclic Record File: Byte 4-7: Größe eines Records Byte 8-11: maximale Anzahl an Records Byte 12-15: Aktuelle Anzahl an Records FileSettingsBufferLen Anzahl der Bytes, die in pFileSettings zur Verfügung stehen Rückgabewert: -1 Fehler, >=0 Länge der Daten in pFileSettings

Liefert Informationen und Einstellungen zu dem File mit File ID.

Geliefert wird der Dateityp, der Kommunikationsmodus, die Zugriffsrechte sowie die File-Nutzdatengröße.



## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_Desfire\_ChangeFileSettings** (int hPort, int FileId,  
int bChangeAccessRightsFree,  
BYTE\* pFileSettings, int FileSettingsLen)

hPort	Logische Gerätenummer
FileId	File ID
bChangeAccessRightsFree	1: Ausführung ohne Authentifizierung möglich 0: Ausführung nur mit Authentifizierung erlaubt
pFileSettings	Byte 0: Art der Kommunikation mit dem File 0: unverschlüsselt 1: unverschlüsselt, gesichert mit MAC 3: verschlüsselt Bytes 1-2: Zugriffsrechte Bits 0-3: Recht zum Ändern der Zugriffsrechte Bits 4-7: Lese- & Schreibrecht Bits 8-11: Schreibrecht Bits 12-15: Leserecht für jedes Zugriffsrecht: 0x0 - 0xd: Schlüsselnummer für vorhergehende Authentifizierung 0xe: Freier Zugriff 0xf: Kein Zugriff
FileSettingsLen	Länge der Daten in pFileSettings
Rückgabewert:	-1 Fehler, 0 OK

Setzt in dem File den Communication Mode und die Zugriffsrechte.

File-Typ und Größe können nach dem Erzeugen jedoch nicht mehr geändert werden.



## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_Desfire\_CreateStandardDataFile** (int hPort, BYTE FileId,  
int IsoFileId, int FileSize,  
BYTE CommunicationMode, int AccessRights)

hPort	Logische Gerätenummer
FileId	File ID, zwischen 0x00 und 0x1f, muss innerhalb der Applikation eindeutig sein.
IsoFileId	File-ID nach ISO7816-4 (ISO File ID). 2 Byte. Zwischen 0x0001 und 0xffff. Muss innerhalb der Applikation eindeutig sein. Für Kommandos nach ISO7816-4 wie z.B. TSHRW_Desfire_IsoSelectFile. Wenn dieser Wert 0 ist dann wird keine ISO File ID gesetzt. Ob notwendig oder nicht möglich ist abhängig von Applikations-Einstellung "Files haben ISO-ID".
FileSize	Nutzdatengröße. Kann später nicht mehr verändert werden.
CommunicationMode	Art der Kommunikation mit dem File 0: unverschlüsselt 1: unverschlüsselt, gesichert mit MAC 3: verschlüsselt
AccessRights	Bits 0-3: Recht zum Ändern der Zugriffsrechte Bits 4-7: Lese- & Schreibrecht Bits 8-11: Schreibrecht Bits 12-15 Leserecht für jedes Zugriffsrecht: 0 -D <sup>hex</sup> : Schlüsselnummer für vorhergehende Authentifizierung E <sup>hex</sup> : Freier Zugriff F <sup>hex</sup> : Kein Zugriff
Rückgabewert:	-1 Fehler, 0 OK

Erzeugt ein Standard Data File innerhalb der aktuell selektierten Application.  
Dieser File-Typ wird verwendet, Daten unformatiert zu speichern.  
Es verwendet keinen internen Backup-Mechanismus.



## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_Desfire\_CreateBackupDataFile** (int hPort, BYTE FileId,  
int IsoFileId, int FileSize,  
BYTE CommunicationMode, int AccessRights)

**hPort** Logische Gerätenummer

**FileId** File ID, zwischen 0x00 und 0x1f, muss innerhalb der Applikation eindeutig sein.

**IsoFileId** File-ID nach ISO7816-4 (ISO File ID).  
2 Byte. Zwischen 0x0001 und 0xffff.  
Muss innerhalb der Applikation eindeutig sein.  
Für Kommandos nach ISO7816-4 wie z.B. TSHRW\_Desfire\_IsoSelectFile.  
Wenn dieser Wert 0 ist dann wird keine ISO File ID gesetzt.  
Ob notwendig oder nicht möglich ist abhängig von Applikation-Einstellung "Files haben ISO-ID".

**FileSize** Nutzdatengröße. Kann später nicht mehr verändert werden.

**CommunicationMode** Art der Kommunikation mit dem File  
0: unverschlüsselt  
1: unverschlüsselt, gesichert mit MAC  
3: verschlüsselt

**AccessRights** Bits 0-3: Recht zum Ändern der Zugriffsrechte  
Bits 4-7: Lese- & Schreibrecht  
Bits 8-11: Schreibrecht  
Bits 12-15 Leserecht  
für jedes Zugriffsrecht:  
0 - D<sup>hex</sup>: Schlüsselnummer für vorhergehende Authentifizierung  
E<sup>hex</sup>: Freier Zugriff  
F<sup>hex</sup>: Kein Zugriff

**Rückgabewert:** -1 Fehler, 0 OK

Erzeugt ein Backup Data File innerhalb der aktuell selektierten Applikation.  
Dieser File-Typ wird verwendet, Daten unformatiert zu speichern.  
Es verwendet den internen Backup-Mechanismus.



## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_Desfire\_CreateValueDataFile** (int hPort, BYTE FileId,  
BYTE CommunicationMode, int AccessRights,  
int LowerLimit, int UpperLimit,  
int InitialValue, BYTE Config)

hPort	Logische Gerätenummer
FileId	File ID, zwischen 0x00 und 0x1f, muss innerhalb der Applikation eindeutig sein.
CommunicationMode	Art der Kommunikation mit dem File 0: unverschlüsselt 1: unverschlüsselt, gesichert mit MAC 3: verschlüsselt
AccessRights	Bits 0-3: Recht zum Ändern der Zugriffsrechte Bits 4-7: Lese- & Schreibrecht Bits 8-11: Schreibrecht Bits 12-15 Leserecht für jedes Zugriffsrecht: 0 - D <sup>hex</sup> : Schlüsselnummer für vorhergehende Authentifizierung E <sup>hex</sup> : Freier Zugriff F <sup>hex</sup> : Kein Zugriff
LowerLimit	minimaler Value
UpperLimit	minimaler Value
InitialValue	Anfangswert des Value nach File-Erzeugung
Config	Bit 0: „LimitedCredit“ Befehl, 1: erlaubt, 0: deaktiviert Bit 1: „Free GetValue“ Funktionalität, 1: erlaubt, 0: deaktiviert Rückgabewert: -1 Fehler, 0 OK

Erzeugt ein Value File innerhalb der aktuell selektierten Applikation.

Dieser File-Typ wird verwendet, um eine Zahl (Value) zu speichern und zu bearbeiten. Der Value ist eine Ganzzahl mit 32 Bit und Vorzeichen. Das File verwendet den internen Backup-Mechanismus.



## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_Desfire\_CreateLinearRecordFile** (int hPort, BYTE FileId, int IsoFileId, BYTE CommunicationMode, int AccessRights, int RecordSize, int MaxCountRecords)

hPort	Logische Gerätenummer
FileId	File ID, zwischen 0x00 und 0x1f, muss innerhalb der Applikation eindeutig sein.
IsoFileId	File-ID nach ISO7816-4 (ISO File ID). 2 Byte. Zwischen 0x0001 und 0xffff. Muss innerhalb der Applikation eindeutig sein. Für Kommandos nach ISO7816-4 wie z.B. TSHRW_Desfire_IsoSelectFile. Wenn dieser Wert 0 ist dann wird keine ISO File ID gesetzt. Ob notwendig oder nicht möglich ist abhängig von Applikation-Einstellung "Files haben ISO-ID".
CommunicationMode	Art der Kommunikation mit dem File 0: unverschlüsselt 1: unverschlüsselt, gesichert mit MAC 3: verschlüsselt
AccessRights	Bits 0-3: Recht zum Ändern der Zugriffsrechte Bits 4-7: Lese- & Schreibrecht Bits 8-11: Schreibrecht Bits 12-15 Leserecht für jedes Zugriffsrecht: 0 - D <sup>hex</sup> : Schlüsselnummer für vorhergehende Authentifizierung E <sup>hex</sup> : Freier Zugriff F <sup>hex</sup> : Kein Zugriff
RecordSize	Größe eines Records, im Bereich von 1 bis 0xfffff (16.777.215)
MaxCountRecords	Maximale Anzahl der Records, im Bereich von 1 bis 0xfffff (16.777.215).
Rückgabewert:	-1 Fehler, 0 OK

Erzeugt ein Linear Record File innerhalb der aktuell selektierten Applikation. Dieser File-Typ wird verwendet, um Daten strukturiert zu speichern. Das File besteht aus einem Feld von Datensätzen (Records). Wenn alle Records gefüllt sind, kann nicht mehr geschrieben werden.



## Programmierschnittstelle (SDK) der TS-HRW Serie

<b>int TSHRW_Desfire_CreateCyclicRecordFile</b> (int hPort, BYTE FileId, int IsoFileId, BYTE CommunicationMode, int AccessRights, int RecordSize, int MaxCountRecords)	
hPort	Logische Gerätenummer
FileId	File ID, zwischen 0x00 und 0x1f, muss innerhalb der Applikation eindeutig sein.
IsoFileId	File-ID nach ISO7816-4 (ISO File ID). 2 Byte. Zwischen 0x0001 und 0xffff. Muss innerhalb der Applikation eindeutig sein. Für Kommandos nach ISO7816-4 wie z.B. TSHRW_Desfire_IsoSelectFile. Wenn dieser Wert 0 ist dann wird keine ISO File ID gesetzt. Ob notwendig oder nicht möglich ist abhängig von Applikation-Einstellung "Files haben ISO-ID".
CommunicationMode	Art der Kommunikation mit dem File 0: unverschlüsselt 1: unverschlüsselt, gesichert mit MAC 3: verschlüsselt
AccessRights	Bits 0-3: Recht zum Ändern der Zugriffsrechte Bits 4-7: Lese- & Schreibrecht Bits 8-11: Schreibrecht Bits 12-15 Leserecht für jedes Zugriffsrecht: 0 - D <sup>hex</sup> : Schlüsselnummer für vorhergehende Authentifizierung E <sup>hex</sup> : Freier Zugriff F <sup>hex</sup> : Kein Zugriff
RecordSize	Nutzdatengröße eines Records, im Bereich von 1 bis 0xfffff (16.777.215)
MaxCountRecords	Maximale Anzahl der Records, im Bereich von 1 bis 0xfffff (16.777.215). Wegen einem aktuellen Record ist die maximale Anzahl an gültigen Records 1 weniger.
Rückgabewert:	-1 Fehler, 0 OK

Erzeugt ein Cyclic Record File innerhalb der aktuell selektierten Applikation.

Dieser File-Typ wird verwendet, um Daten strukturiert zu speichern. Das File besteht aus einem Feld von Datensätzen (Records) und einem aktuellen Record. Wenn alle möglichen Records gefüllt sind und ein zusätzlicher Record mit CommitTransaction übernommen werden soll, so wird der älteste Record mit dem Wert des aktuellen Records überschrieben.



## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_Desfire\_DeleteFile** (int hPort, BYTE FileId)

hPort                      Logische Gerätenummer  
FileId                      File ID

Löscht ein File innerhalb der aktuell selektierten Applikation.

Der Speicherbereich des gelöschten Files auf der Karte wird nicht freigegeben.

Die File ID des gelöschten Files kann in der Applikation für eine neues File wiederverwendet werden.

**int TSHRW\_Desfire\_ReadData** (int hPort, BYTE FileId, BYTE CommunicationMode,  
int StartIndex, int ReadCount,  
BYTE\* pReadDataBuffer, int ReadDataBufferLen)

hPort                      Logische Gerätenummer  
FileId                      File ID  
CommunicationMode      Art der Kommunikation mit dem File  
                                 0: unverschlüsselt  
                                 1: unverschlüsselt, gesichert mit MAC  
                                 3: verschlüsselt  
StartIndex                Index des 1.Byte, ab dem gelesen wird  
ReadCount                Anzahl der zu lesenden Byte. 0: ganzes File lesen, beginnend bei StartIndex  
pReadDataBuffer        Puffer, in den die gelesenen Bytes geschrieben werden  
ReadDataBufferLen      Länge von pReadDataBuffer  
Rückgabewert:            -1 Fehler, Länge der tatsächlich gelesenen Daten

Liest Daten aus einem Standard Data File oder einem Backup Data File.

Benötigt „Leserecht“ oder „Lese- & Schreibrecht“.

**int TSHRW\_Desfire\_WriteData** (int hPort, BYTE FileId, BYTE CommunicationMode,  
DWORD OffsetInFile, BYTE\* pWriteData, int WriteDataLen)

hPort                      Logische Gerätenummer  
FileId                      File ID  
CommunicationMode      Art der Kommunikation mit dem File  
                                 0: unverschlüsselt  
                                 1: unverschlüsselt, gesichert mit MAC  
                                 3: verschlüsselt  
OffsetInFile              Index des 1.Byte im File, das überschrieben wird  
pWriteData                Puffer, in dem die zu schreibenden Daten stehen  
WriteDataLen              Länge der zu schreibenden Daten  
Rückgabewert:            -1 Fehler, 0 OK

Schreibt Daten in ein Standard Data File oder ein Backup Data File.

Benötigt „Schreibrecht“ oder „Lese- & Schreibrecht“.





## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_Desfire\_GetValue** (int hPort, BYTE FileId,  
BYTE CommunicationMode, unsigned int \* pValue)

hPort	Logische Gerätenummer
FileId	File ID
CommunicationMode	Art der Kommunikation mit dem File 0: unverschlüsselt 1: unverschlüsselt, gesichert mit MAC 3: verschlüsselt Wenn „FreeGetValue“ in CreateValueDataFile gesetzt wurde, dann die Art “unverschlüsselt” verwenden, gegebenenfalls mit MAC nach Authentifizierung.
pValue	Puffer für den gelesenen Value des Files, ein Value ist 4 Byte lang
Rückgabewert:	-1 Fehler, 0 OK

Liest den aktuellen Wert (Value) des Value Files.

Benötigt „Leserecht“, „Schreibrecht“ oder „Lese- & Schreibrecht“.

Kann auch ohne diese Rechte verwendet werden, wenn das File die Eigenschaft „FreeGetValue“ besitzt.

**int TSHRW\_Desfire\_Credit** (int hPort, BYTE FileId,  
BYTE CommunicationMode, unsigned int Value)

hPort	Logische Gerätenummer
FileId	File ID
CommunicationMode	Art der Kommunikation mit dem File 0: unverschlüsselt 1: unverschlüsselt, gesichert mit MAC 3: verschlüsselt
Value	Wert, um den der Wert im Value File erhöht werden soll
Rückgabewert:	-1 Fehler, 0 OK

Erhöht den Wert eines Value Files.

Benötigt „Lese- & Schreibrecht“.



## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_Desfire\_LimitedCredit** (int hPort, BYTE FileId,  
BYTE CommunicationMode, unsigned int Value)

hPort	Logische Gerätenummer
FileId	File ID
CommunicationMode	Art der Kommunikation mit dem File 0: unverschlüsselt 1: unverschlüsselt, gesichert mit MAC 3: verschlüsselt
Value	Wert, um den der Wert im Value File erhöht werden soll
Rückgabewert:	-1 Fehler, 0 OK

Erhöht den Wert eines Value Files, ohne die volle Lese&Schreibrechte des Files zu besitzen, Diese Funktionalität kann bei der Erzeugung eines Files gesetzt oder ausgeschaltet werden. Der Wert ist beschränkt auf den Wert der letzten Debit-Transaktion. Benötigt „Schreibrecht“ oder „Lese- & Schreibrecht“.

**int TSHRW\_Desfire\_Debit** (int hPort, BYTE FileId,  
BYTE CommunicationMode, unsigned int Value)

hPort	Logische Gerätenummer
FileId	File ID
CommunicationMode	Art der Kommunikation mit dem File 0: unverschlüsselt 1: unverschlüsselt, gesichert mit MAC 3: verschlüsselt
Value	Wert, um den der Wert im Value File erniedrigt werden soll
Rückgabewert:	-1 Fehler, 0 OK

Erniedrigt den Wert (Value) eines Value Files.  
Benötigt „Leserecht“, „Schreibrecht“ oder „Lese- & Schreibrecht“.

**Programmierschnittstelle (SDK) der TS-HRW Serie**

**int TSHRW\_Desfire\_ReadRecord** (int hPort, BYTE FileId, BYTE CommunicationMode,  
int StartRecord, int ReadCountRecords, int RecordSize,  
BYTE\* pReadDataBuffer, int ReadDataBufferLen)

hPort	Logische Gerätenummer
FileId	File ID
CommunicationMode	0: unverschlüsselt 1: unverschlüsselt, gesichert mit MAC 3: verschlüsselt
StartRecord	Index des jüngsten zu lesenden Records, 0 ist der zuletzt erzeugte Record, 1 der vorletzte
ReadCountRecords	Anzahl der zu lesenden Records. 0: alle Records lesen
RecordSize	Größe eines Records
pReadDataBuffer	Puffer, in den die gelesenen Bytes geschrieben werden
ReadDataBufferLen	Länge von pReadDataBuffer
Rückgabewert:	-1 Fehler, Länge der tatsächlich gelesenen Daten

Liest aus einem Linear Record File oder einem Cyclic Record File einen Satz von Datensätzen (Records). Die Records werden in der Reihenfolge ihrer Erzeugung geliefert, beginnend mit dem ältesten Record. Der jüngste Record im File hat den Index 0, der älteste im File den Index AnzahlRecords-1. Der älteste gelesene Record hat den Index ReadCountRecords-1-StartRecord. Wenn StartRecord 0 ist, werden alle Records vom ältesten bis einschließlich dem jüngsten gelesen. Der aktuelle Record ist dagegen noch nicht gültig erklärt und wird nicht gelesen. Benötigt „Leserecht“ oder „Lese- & Schreibrecht“.



## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_Desfire\_WriteRecord** (int hPort, BYTE FileId,  
BYTE CommunicationMode, DWORD OffsetInRecord,  
BYTE\* pWriteData, int WriteDataLen)

hPort	Logische Gerätenummer
FileId	File ID
CommunicationMode	Art der Kommunikation mit dem File 0: unverschlüsselt 1: unverschlüsselt, gesichert mit MAC 3: verschlüsselt
OffsetInRecord	Byte-Offset im aktuellen Record
pWriteData	Puffer, in dem die zu schreibenden Daten stehen
WriteDataLen	Länge der zu schreibenden Daten
Rückgabewert:	-1 Fehler, 0 OK

Schreibt Daten in den aktuellen Record eines Linear oder Cyclic Record Files.

Mit einem folgenden CommitTransaction wird der aktuelle Record für gültig erklärt, wird zum jüngsten gültigen Record, kann von da an gelesen werden und nicht mehr beschrieben werden, und ein neuer aktueller Record wird angehängt und mit Nullen gefüllt. Sind dabei bereits alle möglichen Records gefüllt, gibt es bei einem Linear Record File eine Fehlermeldung, bei einem Cyclic Record File wird der älteste Record mit dem aktuellen Record überschrieben. Ein AbortTransaction macht alle Schreibvorgänge ungültig. Benötigt „Schreibrecht“ oder „Lese- & Schreibrecht“.

**int TSHRW\_Desfire\_ClearRecordFile** (int hPort, BYTE FileId)

hPort	Logische Gerätenummer
FileId	File ID
Rückgabewert:	-1 Fehler, 0 OK

Setzt in einem Cyclic oder Linear Record File alle Datensätze auf den Status ungefüllt.

Nach der Ausführung des Befehls ist ein CommitTransaction-Befehl notwendig.

Vor dem CommitTransaction-Befehl schlagen alle WriteRecord-Befehle fehl.

Ein AbortTransaction-Befehl macht die das Zurücksetzen des Files ungültig.

Benötigt „Lese- & Schreibrecht“.



## **Programmierschnittstelle (SDK) der TS-HRW Serie**

**int TSHRW\_Desfire\_CommitTransaction** (int hPort)

hPort                      Logische Gerätenummer

Rückgabewert:            -1 Fehler, 0 OK

Macht alle vorhergehenden Schreibvorgänge in Backup Data Files, Value Files und Record Files innerhalb der Applikation gültig und übernimmt die neuen Werte in das File.

**int TSHRW\_Desfire\_AbortTransaction** (int hPort)

hPort                      Logische Gerätenummer

Rückgabewert:            -1 Fehler, 0 OK

Macht alle vorhergehenden Schreibvorgänge in Backup Data Files, Value Files und Record Files innerhalb der Applikation ungültig. Es verwendet den internen Backup-Mechanismus. Es verändert nicht den Authentifizierungsstatus.



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 5.6.5. Befehle nach ISO 7816-4

Diese Befehle befolgen die Spezifikation ISO7816-4. Sie können nicht abwechselnd mit nativen DESFire-Befehlen aufgerufen werden. Nach einer Selektion des Transponders nach ISO 14443 ist es notwendig, entweder nur native-Funktionen oder nur ISO7816-Funktionen zu verwenden.

**int TSHRW\_Desfire\_IsoSelectApplication** (int hPort, BYTE\* pAppName,  
int AppNameLen)

hPort	Logische Gerätenummer
pAppName	Zeiger auf Buffer für Applikationsname, entspricht DF-Name
AppNamesArrayLen	Länge von Applikationsname
Rückgabewert:	-1 Fehler, 0 OK

Selektiert eine Applikation. Selektiert Kartenebene bei DF-Name 0x D2 76 00 00 85 01 00.

**int TSHRW\_Desfire\_IsoSelectFile** (int hPort, int IsoFileId)

hPort	Logische Gerätenummer
IsoFileId	ISO File ID
Rückgabewert:	-1 Fehler, 0 OK

Selektiert ein File innerhalb der aktuell selektierten Applikation.

**int TSHRW\_Desfire\_IsoReadData** (int hPort, int StartIndex, int ReadCount,  
BYTE\* pReadDataBuffer, int ReadDataBufferLen)

hPort	Logische Gerätenummer
StartIndex	Offset in Datei in Byte.
ReadCount	Anzahl der zu lesenden Bytes
pReadDataBuffer	Puffer, in den die gelesenen Bytes geschrieben werden
ReadDataBufferLen	Länge des Puffers für pReadData
Rückgabewert:	-1 Fehler, >=0 Länge der tatsächlich gelesenen Daten

Liest aus Standard Data File oder Backup Data File.



## **Programmierschnittstelle (SDK) der TS-HRW Serie**

**int TSHRW\_Desfire\_IsoWriteData** (int hPort, int StartIndex,  
BYTE\* WriteData, int WriteDataLen)

hPort	Logische Gerätenummer
StartIndex	Offset in Datei in Byte, ab dem geschrieben wird
pWriteData	Puffer, in dem die zu schreibenden Daten stehen
WriteDataLen	Länge der zu schreibenden Daten
Rückgabewert:	-1 Fehler, 0 OK

Schreibt Daten in ein Standard Data File oder ein Backup Data File.



## **6. Befehle für NFC (Near Field Communication)**

Mit dem SDK können Daten auf komfortable Art der NFC-Spezifikation entsprechend in einen Transponder geschrieben werden. Sie können dann von anderen NFC-fähigen Lesegeräten ausgelesen und verwendet werden, wie z.B. von vielen modernen Smartphones.

### **6.1. Anwendungshinweise**

NFC steht für „**Near Field Communication**“. Es ist ein internationaler Übertragungsstandard, an dem Firmen wie Google, Microsoft, Nokia, Samsung und viele andere beteiligt sind.

Die meisten neuen höherwertigen Smartphones mit Android oder Windows Phone Betriebssystem haben bereits einen NFC Leser eingebaut.

Wenn in dem Smartphone NFC eingeschaltet ist und an das Gerät ein NFC Transponder gehalten wird, wird der Inhalt des Transponders automatisch eingelesen.

Je nach Inhalt der NFC Daten startet das Betriebssystem die entsprechende App und übergibt ihr die Daten.

Zum Beispiel:

Im Transponder ist ein Kontakt im NFC Format gespeichert.

Ein Kontakt ist eine Art Visitenkarte und kann Namen, Telefonnummern, E-Mail-Adresse und vieles mehr enthalten.

Nach dem Einlesen des Transponders erkennt das Betriebssystem, dass die Daten ein Kontakt sind, und fragt den Anwender, ob er den Kontakt hinzufügen möchte. Bei Bestätigung wird der Kontakt automatisch den Kontakten (Telefonbuch) des Smartphones hinzugefügt.





## **Programmierschnittstelle (SDK) der TS-HRW Serie**

### **6.2. Unterstützte Transpondertypen**

Die hier angegebenen Speichergrößen sind die Benutzerdatenbereiche der Transponder, die tatsächlichen Nutzdaten eines NFC Datensatzes sind vom Datentyp abhängig etwas geringer, da zusätzliche Verwaltungsdaten gespeichert werden müssen.

#### **6.2.1. NFC Typ 2**

1. NXP MIFARE Ultralight® kann standardmäßig 48 Byte Nutzdaten speichern.
2. NXP MIFARE Ultralight® C kann standardmäßig 144 Byte Nutzdaten speichern.
3. NXP NTAG 203 kann standardmäßig 144 Byte Nutzdaten speichern.
4. Infineon my-d NFC kann standardmäßig 128 Byte Nutzdaten speichern.
5. Infineon my-d move NFC kann standardmäßig 128 Byte Nutzdaten speichern.
6. weitere NFC Typ 2 kompatible Transponder

#### **6.2.2. NFC Typ 4**

1. NXP MIFARE® DESFire mit verschiedenen Speichergrößen

#### **6.2.3. NFC Typ 6**

1. NXP ICode SLI / SLIX mit verschiedenen Speichergrößen
2. TI TagIT HF mit verschiedenen Speichergrößen
3. weitere ISO15693 kompatible Transponder

Speziell bei NFC Typ 6 werden nicht alle ISO15693 kompatiblen Transponder auch von allen NFC Geräten unterstützt. So sind viele NFC-Geräte bekannt, die nur die NXP Transponder unterstützen.

#### **6.2.4. NFC Typ 7**

1. NXP MIFARE® Classic 1k kann standardmäßig 720 Byte Nutzdaten speichern.
2. NXP MIFARE® Classic 4k. kann standardmäßig 3360 Byte Nutzdaten speichern.

Werden nicht von allen NFC-Geräten unterstützt.



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 6.3. Unterstützte Anwenderdaten

Die Anwenderdaten werden zum Lesen und Schreiben im JSON-Format ausgetauscht.

JSON steht für „JavaScript Object Notification“ und ist ein allgemeines Format für den Austausch von Daten zwischen Anwendungen.

Parser für JSON existieren in praktisch allen verbreiteten Programmiersprachen.

Anmerkung zu den folgenden Beispielen im JSON-Format:

**Leerzeichen** und **Zeilenumbrüche** um die Schlüsselwörter werden von den Parsern und dem SDK ignoriert.

Das SDK unterstützt folgende Arten von Anwenderdaten:

#### 6.3.1. Text

Ein NFC-Gerät, das einen so beschriebenen Transponder liest, zeigt diesen Text unmittelbar an.

Beispiel (JSON-Format):

```
{ "TEXT": { "TITLE": "Dies ist ein Beispieltext" } }
```

oder

```
{ "TEXT":  
  {  
    "TITLE": "Dies ist ein ""Beispieltext"" mit Anführungszeichen"  
  }  
}
```

#### 6.3.2. WWW-Adresse

Ein NFC-Gerät, das einen so beschriebenen Transponder liest, möchte die Webadresse aus dem Transponder im Internet-Browser laden und anzeigen.

Je nach NFC-Gerät wird z.B. die Adresse zusammen mit einem optionalen Beschreibungstext angezeigt und der Anwender muss das Öffnen bestätigen, oder es wird z.B. sofort der Browser geöffnet und die Internetseite geladen (bei aktiver Internetverbindung).

Beispiel (JSON-Format):

```
{ "BOOKMARK":  
  {  
    "URL": "https://www.BeispielAdresse.de",  
    "TITLE": "Dies ist ein optionaler Beischreibungstext"  
  }  
}
```



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 6.3.3. Telefon

Ein NFC-Gerät, das einen so beschriebenen Transponder liest, möchte die gespeicherte Telefonnummer in der Telefon-App anzeigen, von wo aus sie angerufen werden kann. Je nach NFC-Gerät wird zuerst die Telefonnummer zusammen mit einem optionalen Beschreibungstext angezeigt und der Anwender muss das Öffnen der Telefon-App bestätigen, oder es wird sofort die Telefon-App geöffnet und die Telefonnummer angezeigt (aber nicht angerufen).

Beispiel (JSON-Format):

```
{ "TELEPHON":  
  {  
    "NUMBER": "12345678",  
    "TITLE": "Dies ist ein optionaler Beschreibungstext"  
  }  
}
```

### 6.3.4. SMS

Ein NFC-Gerät, das einen so beschriebenen Transponder liest, möchte die SMS aus dem Transponder im SMS-Editor anzeigen, wo sie bearbeitet oder verschickt werden kann. Je nach NFC-Gerät wird zuerst der SMS-Name zusammen mit einem optionalen Beschreibungstext angezeigt und der Anwender muss das Öffnen bestätigen, oder es wird sofort der SMS-Editor geöffnet und die SMS angezeigt (aber nicht verschickt).

Beispiel (JSON-Format):

```
{ "SMS":  
  {  
    "ADDRESS": "12345678",  
    "MESSAGE": "Dies ist ein Nachrichttext",  
    "TITLE": "Dies ist ein optionaler Beschreibungstext"  
  }  
}
```



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 6.3.5. E-Mail

Ein NFC-Gerät, das einen so beschriebenen Transponder liest, öffnet die E-Mail-App und zeigt die E-Mail an. Auf Wunsch des Anwenders kann sie bearbeitet oder verschickt werden

Beispiel (JSON-Format):

```
{ "MAIL":  
  {  
    "ADDRESS": "BeispielName@BeispielAdresse.de",  
    "SUBJECT": "Dies ist ein Betreff",  
    "MESSAGE": "Dies ist ein Nachrichttext",  
    "TITLE": "Dies ist ein optionaler Beschreibungstext"  
  }  
}
```



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 6.3.6. Visitenkarte

Ein NFC-Gerät, das einen so beschriebenen Transponder liest, möchte den Kontakt aus dem Transponder zum Telefonbuch des NFC-Gerätes hinzufügen. Je nach NFC-Gerät wird zuerst der Kontaktname zusammen mit einem optionalen Beschreibungstext angezeigt und der Anwender muss das Hinzufügen bestätigen, oder es wird sofort das Telefonbuch geöffnet und der Kontakt hinzugefügt und angezeigt.

Ein Kontakt besteht aus Vorname, Nachname, Firma / Organisation und einer beliebigen Anzahl von Telefonnummern, E-Mail-Adressen und Internetseiten.

Eine Telefonnummer besteht aus Nummer, Anschlussort (0: unbestimmt, 1: Arbeit, 2: privat) und Geräteart (0: unbestimmt oder Festnetz, 1: mobil, 2: Fax).

Beispiel (JSON-Format):

```
{ "VCARD":
{
  "FIRSTNAME": "Dies ist der Vorname",
  "SURNAME": "Dies ist der Nachname",
  "ORGANISATION": "Dies ist die Firma oder Organisation",
  "MAILS":
  {
    "MAIL0": "BeispielName@BeispielAdresse.de",
    "MAIL1": "ZweitesBeispiel@ZweiteAdresse.com",
    "MAIL2": "WeiteresBeispiel@WeitereAdresse.eu"
  },
  "URLS":
  {
    "URL0": "https://www.BeiispielAdresse.de",
    "URL1": "http://www.ZweiteAdresse.com"
  },
  "PHONES":
  {
    "PHONE0":
    {
      "NUMBER": "12345678",
      "TYPE_PLACE": 2,
      "TYPE_DEVICE": 1
    },
    "PHONE1":
    {
      "NUMBER": "11223344"
    }
  }
}
```



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 6.4. Funktionsaufrufe NFC

**int TSHRW\_Nfc\_ReadTag** (int hPort, char\* pReadData, int ReadDataLen)

hPort                      Logische Gerätenummer  
pReadData                Puffer, in den der gelesene JSON-String mit terminierender 0 geschrieben wird  
ReadDataLen              Länge des Puffers für pReadData  
Rückgabewert:            -1 Fehler, >=0 Länge der tatsächlich gelesenen Daten (ohne terminierender 0)

Liest aus einem Transponder mit Anwenderdaten im NFC-Format die Anwenderdaten.  
Sie werden als ein String im JSON-Format geliefert.

**int TSHRW\_Nfc\_ReadUrl** (int hPort, char\* pUrl, int UrlLen, char\* pText, int TextLen)

hPort                      Logische Gerätenummer  
pUrl                        Puffer, in den die gelesene URL mit terminierender 0 geschrieben wird  
UrlLen                     Länge des Puffers für pUrl  
pText                        Puffer, in den der Beschreibungstext mit terminierender 0 geschrieben wird.  
                              NULL: Text ignorieren  
TextLen                    Länge des Puffers für pText. 0: Text ignorieren  
Rückgabewert:            -1 Fehler, >=0 Länge der tatsächlich gelesenen URL (ohne terminierender 0)

Liest aus einem Transponder im NFC-Format die gespeicherte URL (Uniform Resource Locator, z.B. eine Internetadresse) sowie den optionalen zugehörigen Beschreibungstext. Bei anderen Anwenderdaten als einer URL wird -1 zurückgegeben.

**int TSHRW\_Nfc\_ReadText** (int hPort, char\* pText, int TextLen)

hPort                      Logische Gerätenummer  
pText                        Puffer, in den der gelesene Text-String mit terminierender 0 geschrieben wird  
TextLen                    Länge des Puffers für pText  
Rückgabewert:            -1 Fehler, >=0 Länge des gelesenen Textes (ohne terminierender 0)

Liest aus einem Transponder im NFC-Format den gespeicherten Text.  
Bei anderen Anwenderdaten als einem Text wird -1 zurückgegeben.





## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_Nfc\_WriteUrl** (int hPort, char\* pUrl, int UrlLen,  
char\* pText, int TextLen,  
BYTE\* pKey, int KeySize, int Lock)

hPort                      Logische Gerätenummer  
pUrl                        Puffer, in dem die zu schreibenden URL steht,  
keine terminierende 0 notwendig  
UrlLen                     Länge der zu schreibenden URL  
pText                       Puffer, in dem der zu schreibende Beschreibungstext steht,  
keine terminierende 0 notwendig  
NULL: ignorieren  
TextLen                    Länge des zu schreibenden Beschreibungstextes (0: pText ignorieren)  
pKey                        Zeiger auf Buffer mit Schlüssel. Der Schlüssel hat eine Länge von 16 Byte.  
NULL: pKey ignorieren  
KeySize                    Länge des Schlüssels (0: pKey ignorieren)  
Lock                        1: Formatierung und Anwenderdaten des Transponders schreibschützen.  
**Achtung: Dies kann nicht rückgängig gemacht werden.**  
0: nicht schreibschützen  
Rückgabewert:            -1 Fehler, 0 OK

Beschreibt den Transponder im NFC-Format mit der URL (Uniform Resource Locator, z.B. eine Internetadresse) sowie einem optionalen zugehörigen Beschreibungstext. Wenn der Transponder noch nicht als NFC formatiert ist, dann wird er als NFC formatiert. Bei DESFire-Transpondern kann es sein, dass sie über einen Schlüssel schreibgeschützt sind. In diesem Fall muss auch der Schlüssel übergeben werden.

**int TSHRW\_Nfc\_WriteText** (int hPort, char\* pText, int TextLen,  
BYTE\* pKey, int KeySize, int Lock)

hPort                      Logische Gerätenummer  
pText                       Puffer, in dem der zu schreibende Text steht,  
keine terminierende 0 notwendig  
TextLen                    Länge des zu schreibenden Textes  
pKey                        Zeiger auf Buffer mit Schlüssel. Der Schlüssel hat eine Länge von 16 Byte.  
NULL: pKey ignorieren  
KeySize                    Länge des Schlüssels (0: pKey ignorieren)  
Lock                        1: Formatierung und Anwenderdaten des Transponders schreibschützen.  
**Achtung: Dies kann nicht rückgängig gemacht werden.**  
0: nicht schreibschützen  
Rückgabewert:            -1 Fehler, 0 OK

Beschreibt den Transponder im NFC-Format mit einem Text.  
Wenn der Transponder noch nicht als NFC formatiert ist, dann wird er als NFC formatiert.  
Bei DESFire-Transpondern kann es sein, dass sie über einen Schlüssel schreibgeschützt sind. In diesem Fall muss auch der Schlüssel übergeben werden.





## Programmierschnittstelle (SDK) der TS-HRW Serie

**int TSHRW\_Nfc\_FormatTag** (int hPort, BYTE\* pKey, int KeySize)

hPort                      Logische Gerätenummer  
pKey                        Zeiger auf Buffer mit Schlüssel. Der Schlüssel hat eine Länge von 16 Byte.  
                              NULL: pKey ignorieren  
KeySize                    Länge des Schlüssels (0: pKey ignorieren)  
Rückgabewert:            -1 Fehler, 0 OK

Formatiert den Transponder im NFC-Format.

Bei DESFire-Transpondern kann es sein, dass sie über einen Schlüssel schreibgeschützt sind. In diesem Fall muss auch der Schlüssel übergeben werden.

**int TSHRW\_Nfc\_LockTag** (int hPort, BYTE\* pKey, int KeySize)

hPort                      Logische Gerätenummer  
pKey                        Zeiger auf Buffer mit Schlüssel. Der Schlüssel hat eine Länge von 16 Byte.  
                              NULL: pKey ignorieren  
KeySize                    Länge des Schlüssels (0: pKey ignorieren)  
Rückgabewert:            -1 Fehler, 0 OK

Schützt die Formatierung und die Anwenderdaten des Transponders vor überschreiben.

Bei DESFire-Transpondern kann es sein, dass sie bereits über einen Schlüssel schreibgeschützt sind. In diesem Fall muss auch der Schlüssel übergeben werden (um sicherzugehen, dass alles, z.B. auch die Anwenderdaten, schreibgeschützt ist).

**int TSHRW\_Nfc\_SetTagPasswordProtected** (int hPort, int bUserDataToo,  
    BYTE\* pOldKey, int OldKeySize,  
    BYTE\* pNewKey, int NewKeySize)

hPort                      Logische Gerätenummer  
bUserDataToo            != 0: auch Anwenderdaten schreibschützen,  
                              0: nur Formatierung schreibschützen  
pOldKey                   Zeiger auf Buffer mit Schlüssel. Der Schlüssel hat eine Länge von 16 Byte.  
                              NULL: pKey ignorieren  
OldKeySize               Länge des Schlüssels (0: pKey ignorieren)  
pNewKey                   Zeiger auf Buffer mit neuem Schlüssel. Der Schlüssel hat eine Länge von 16  
                              Byte.  
NewKeySize               Länge des neuen Schlüssels  
Rückgabewert:            -1 Fehler, 0 OK

Schützt die Formatierung und die Anwenderdaten eines DESFire-Transponders mittels Passwort vor unbefugtem überschreiben.

Bei DESFire-Transpondern kann es sein, dass sie bereits über einen Schlüssel schreibgeschützt sind. In diesem Fall muss auch der bisherige Schlüssel übergeben werden (um sicherzugehen, dass alles, z.B. auch die Anwenderdaten, schreibgeschützt ist).



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 7. Befehle für Transparentmodus

Mit dem Transparentmodus ist es möglich auch Transponder anzusprechen, die nicht den vollen Standard unterstützen. Es sind dazu allerdings genauere Kenntnisse über das Verhalten der Transponder Voraussetzung.

#### **int TSHRW\_TM\_SetProtocol** (int hPort, int Protocol)

hPort                      Logische Gerätenummer  
Protocol                  Verwendetes Übertragungsprotokoll  
                              0: ISO15693  
                              1: ISO1443A  
                              2: ISO14443B  
Rückgabewert:            -1 Fehler, 0 OK

#### **int TSHRW\_TM\_GetProtocol** (int hPort)

hPort                      Logische Gerätenummer  
Rückgabewert:            -1 Fehler,  $\geq 0$  Verwendetes Übertragungsprotokoll, siehe oben

#### **int TSHRW\_TM\_SetFWTETU** (int hPort, int FwtETU)

hPort                      Logische Gerätenummer  
FwtETU                    FrameWaitTime Dauer in ETU (Elementary Time Units)  
                               $ETU = 128/f_c$ , wobei  $f_c = 13.56\text{MHz}$ , also  $ETU = 9,44 \mu\text{s}$ .  
Rückgabewert:            -1 Fehler, 0 OK

#### **int TSHRW\_TM\_GetFWTETU** (int hPort)

hPort                      Logische Gerätenummer  
Rückgabewert:            -1 Fehler,  $\geq 0$  FrameWaitTime, siehe oben



## Programmierschnittstelle (SDK) der TS-HRW Serie

Mit den folgenden Funktionen wird Datenrate für Senden und Empfangen eingestellt oder abgefragt.

Die Datenraten werden als Indexwerte für die Datenrate übergeben und sind abhängig vom gewählten Protokoll.

ISO15693: TxRate 0: 1 aus 256 (1,65kBit/s)

1: 1 aus 4 (26,48kBit/s)

RxRate 0: Single Subcarrier 6,26 kBit/s,

1: Single Subcarrier 26,48 kBit/s

2: Double Subcarrier 6,67 kBit/s

3: Double Subcarrier, 26,69 kBit/s

ISO14443: TxRate

RxRate 0: 106 kBit/s

1: 212 kBit/s

2: 424 kBit/s

3: 828 kBit/s

**int TSHRW\_TM\_SetDatarate** (int hPort, int TxRate, int RxRate)

hPort                      Logische Gerätenummer

TxRate                     Datenrate beim Senden

RxRate                     Datenrate beim Empfang

Rückgabewert:            -1 Fehler, 0 OK

**int TSHRW\_TM\_GetDatarate** (int hPort, int \* pTxRate, int \* pRxRate)

hPort                      Logische Gerätenummer

Rückgabewert:            -1 Fehler, 0 OK



## Programmierschnittstelle (SDK) der TS-HRW Serie

Die eigentliche Datenübertragung im TransparentModus erfolgt über die folgenden Funktionen:

**int TSHRW\_TM\_Transmit** (int hPort, BYTE \* pSendBuf, int SendBufLen,  
BYTE \* pRecvBuf, int RecvBufLen)

hPort	Logische Gerätenummer
pSendBuf	zu sendende Daten
SendBufLen	Länge der zu sendenden Daten
pRecvBuf	empfangene Daten
RecvBufLen	max. Länge der empfangenen Daten
Rückgabewert:	-1 Fehler, $\geq 0$ Länge der Daten in pRecvBuf

**int TSHRW\_TM\_Transmit4** (int hPort, BYTE \* pSendBuf, int SendBufLen,  
BYTE \* pRecvBuf, int RecvBufLen)

hPort	Logische Gerätenummer
pSendBuf	zu sendende Daten
SendBufLen	Länge der zu sendenden Daten
pRecvBuf	empfangene Daten
RecvBufLen	max. Länge der empfangenen Daten
Rückgabewert:	-1 Fehler, $\geq 0$ Länge der Daten in pRecvBuf

Diese Funktion wird verwendet, wenn ein 4 Bit Result erwartet wird. (ACK/NAK)

Dann wird das 4 Bit Ergebnis im 1. Byte des Empfangspuffers in den unteren 4 Bits abgelegt und Rückgabewert = 1

**int TSHRW\_TM\_TransmitP** (int hPort, BYTE \* pSendBuf, int SendBufLen,  
BYTE \* pRecvBuf, int RecvBufLen)

hPort	Logische Gerätenummer
pSendBuf	zu sendende Daten
SendBufLen	Länge der zu sendenden Daten
pRecvBuf	empfangene Daten
RecvBufLen	max. Länge der empfangenen Daten
Rückgabewert:	-1 Fehler, $\geq 0$ Länge der Daten in pRecvBuf

Bei Verwendung der TSHRW\_TM\_TransmitP Funktion, enthalten die übertragenen Daten auch die Paritätsbits. Jedes Datenelement wird durch zwei 8 Bit Werte repräsentiert, wobei im ersten Byte die Daten und im 2. Byte im niedrigsten 9. Bit die Paritätsinformation enthalten ist. Dies gilt für den Sende- und Empfangspuffer. Es wird immer die Anzahl Bytes angegeben.

Bei Datenrückgabe ist die Anzahl immer geradzahlig, da ja immer 2 Byte pro Datenelement zurückgegeben werden. Sollte ein ACK/NAK empfangen worden sein, so ist der Rückgabewert = 1 und es sind tatsächlich nur die unteren 4 Bit des 1. Bytes mit dem ACK/NAK Code belegt.



## 8. Fehlerliste

### 8.1. Allgemeine Fehler

Nr. (dez.)	Beschreibung
1	Fehler beim Öffnen des Ports
2	Timeout-Wert ungültig
3	Portnummer ungültig
4	Timeout
5	Übergebener Puffer zu klein
6	Puffergröße ungültig
7	Falscher Parameter
8	Keine Daten für Kommunikation
9	Keine Daten empfangen
10	Prüfsummenfehler
11	Keine Kommunikation zum Gerät
12	Prüfpuffer leer
13	Mode ungültig
14	Befehl nicht erlaubt
15	Blocknummer ungültig
16	Kommandofehler
17	Antenne ist ausgeschaltet, die Antenne muss zuerst mit TSHRW_SetRF eingeschaltet werden.
19	Block enthält keinen Value Eintrag
20	Datenanzahl falsch
21	Kein Transponder (NACK= 15H)
22	Checksummenfehler vom Gerät (SYNC= 16H)
23	Kollision ist aufgetreten
24	Ungültiges Kommando empfangen, das Gerät kann dieses Kommando nicht ausführen. z. B: wenn Write Kommandos (ab Kapitel 4) bei einem TS-HR38 aufgerufen werden, der nur den ReaderMode beherrscht. (CAN= 18H)
25	Kommunikationsfehler (CRC) bei Luftschnittstelle

## Programmierschnittstelle (SDK) der TS-HRW Serie

### 8.2. Fehler bei Zugriff auf ISO15693 Transponder

Nr. (dez.)	Beschreibung
32	ISO Fehler Kommando wird nicht unterstützt
33	ISO Fehler Kommando wurde nicht erkannt
34	ISO Fehler Option wird nicht unterstützt
35	ISO Fehler Keine Information
36	ISO Fehler Block nicht verfügbar
37	ISO Fehler Block bereits gesperrt
38	ISO Fehler Gesperrter Block kann nicht verändert werden
39	ISO Fehler Block konnte nicht geschrieben werden
40	ISO Fehler Block konnte nicht gesperrt werden
41	ISO Fehler unbekannter Fehler
42	ISO Fehler Länge nicht korrekt
43	ISO Fehler Unbekannter Transponder
44	ISO Fehler Framing Error

### 8.3. Fehler bei Zugriff auf MIFARE® Transponder

Nr. (dez.)	Beschreibung
250	MIFARE® Fehler ungültiger Wert
251	MIFARE® Fehler Parity Error
252	MIFARE® Fehler Authenticate Error,
254	MIFARE® Fehler CRC Error

### 8.4. SDK-interne Fehler bei DESFire Kommandos

Nr. (hex)	Beschreibung
10020	Schlüssellänge beträgt nicht 16 Byte
10021	Länge des bisherigen Schlüssels beträgt nicht 16 Byte
10022	Das Verschlüsselungsverfahren 3K3DES wird nicht unterstützt
10023	Befehl benötigt eine Authentifizierung
10024	MAC passt nicht zu den Daten
10025	CRC nach Entschlüsselung passt nicht zu den Daten
10026	Übergebenes Feld ist für die Daten zu klein
10027	Parameterfeld hat keine gültige Länge
10028	Befehl nicht möglich nach AuthenticateISO oder AuthenticateAES



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 8.5. Fehler von der DESFire-Karte

Die unteren zwei Stellen der Fehlernummer entsprechen der von DESFire gelieferten Fehlernummer.

Nr. (hex)	Beschreibung
1000C	Keine Änderungen durchgeführt
1000E	Interner Fehler, zu wenig NV-Memory
1001C	Befehl wird nicht unterstützt
1001E	Integritätsfehler, CRC oder MAC passen nicht zu den Daten
10040	Ungültige Schlüsselnummer
1007E	Interner Fehler, gesendetes Kommando-Feld hat die falsche Länge
1009D	Die aktuelle Konfiguration erlaubt den Befehl nicht
1009E	Interner Fehler, Parameter-Wert ungültig
100A0	Applikation mit angegebener AID nicht gefunden
100A1	Nicht behebbarer Fehler in Applikation, die Applikation wird blockiert
100AE	Der aktuelle Authentifizierungszustand erlaubt den Befehl nicht
100AF	Interner Fehler, es wird ein weiterer Daten-Rahmen erwartet
100BE	Es wurde versucht, jenseits der Dateigrenzen zu lesen oder zu schreiben
100C1	Karte wurde durch einen nicht behebbaren Fehler funktionsunfähig
100CA	Vorhergehender Befehl wurde nicht vollständig abgeschlossen
100CD	Karte wurde durch einen nicht behebbaren Fehler funktionsunfähig
100CE	Maximale Anzahl an Applikationen erreicht. Diese ist auf 28 begrenzt
100DE	Bereits verwendete Applikation oder Datei Nummer
100EE	Interner Fehler, EEPROM kann nicht beschrieben werden
100F0	Datei nicht gefunden
100F1	Nicht behebbarer Fehler in Datei

### 8.6. Fehler von der DESFire-Karte bei ISO7816-4 Kommandos

Die unteren 4 Stellen der Fehlernummer entsprechen den ISO7816-4 Fehlernummern

Nr. (hex)	Beschreibung
16282	Ende der Datei erreicht, bevor die gewünschten Länge gelesen werden konnte
16700	Interner Fehler, falsche Länge
16982	Zugriff auf Datei nicht gestattet
16985	Leere Datei
16A82	Interner Fehler, falscher Parameter P1 / P2
16A86	Interner Fehler, falscher Parameter P1 / P2
16C00	Datei nicht gefunden
16F00	Fehler ohne genauere Angabe



## Programmierschnittstelle (SDK) der TS-HRW Serie

### 8.7. Fehler von NFC-Kommandos

Nr. (hex)	Beschreibung
20001	Kein Transponder oder Transpondertyp wird in den NFC-Funktionen nicht unterstützt
20002	Dieser Mifare-Transpondertyp wird in den NFC-Funktionen nicht unterstützt
20003	Transponder ist nicht vom Typ DESFire
20004	Transponder ist nicht NFC-formatiert
20005	Authentifizierung des MifareClassic-Transponders fehlgeschlagen
20006	Authentifizierung der DESFire-Kartenebene fehlgeschlagen
20007	Authentifizierung der DESFire-NFC-Application fehlgeschlagen
20008	Fehler beim Ändern von Einstellungen
20009	Fehler beim Ändern des Schlüssels
2000A	Fehler beim Locken von Blöcken in ISO15693-Transponder
2000B	Fehler beim NFC-formatieren des Transponders
2000C	NFC in Transponder hat eine neuere nicht unterstützte NFC-Version
2000D	NFC in Transponder hat eine ältere nicht unterstützte NFC-Version
2000E	Transponder ist schreibgeschützt
2000F	Transponder ist NFC-lesegeschützt
20010	Transponder ist NFC-schreibgeschützt
20011	Fehler beim Lesen von Blöcken
20012	Fehler beim Schreiben in Blöcke
20013	Fehler beim Schreiben in DESFire-File
20014	Transponder hat nicht genügend freien Speicherplatz
20015	ID der DESFire-NFC-Application nicht gefunden
20016	Transponder ist NFC-formatiert, ist aber ohne Anwendernutzdaten
20017	Fehler in NFC-Struktur
20018	Fehler in Anwendernutzdaten vom Typ URL
20019	Fehler in Anwendernutzdaten vom Typ MIME
2001A	Fehler im übergebenen JSON-String
2001B	Transponder ist NFC-formatiert, aber der Typ der Anwendernutzdaten ist unbekannt
2001C	Transponder hat NFC-Anwendernutzdaten, aber nicht vom Typ URL
2001D	Transponder hat NFC-Anwendernutzdaten, aber nicht vom Typ Text