Programmierschnittstelle (SDK) der TS-URW Geräte mit 868 MHz

TS_URW_SDK

Version 1.14

GiS
Gesellschaft für Informatik und Steuerungstechnik mbH

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

GiS
Gesellschaft für Informatik
und Steuerungstechnik mbH

Programmierschnittstelle (SDK) der TS-URW Geräte

Änderungsstand:

Doku	SDK Version	Datum	Kapitel	Name	Info
1.00	1.00	20.11.2012		Blank	Erstversion freigegeben
1.03	1.03	04.11.2013	4.4, 4.5	Blank	Zusätzliche Kommandos für LockTag und Kill
1.04	1.04	01.04.2014	4.4	Blank	Beschreibung des LockTag Kommandos korrigiert
1.05	1.05	03.12.2014	2.2	Blank	Einstellung der Gateway Adresse
1.05	1.05	27.01.2015	4.3	Blank	Lesen mit Passwort
1.13	1.13	08.01.2020	3.7	Blank	Verwendung des Readermodus beschrieben, zusätzliche Kommandos eingfügt.
1.14	1.14	08.06.2020	2.3	Blank	Befehl TSURW_KeepAppActive in Doku ergänzt

Programmierschnittstelle (SDK) der TS-URW Geräte

Inhaltsverzeichnis

Programmierschnittstelle (SDK) der TS-URW Geräte

1. Allgemeines

Die Programmierschnittstelle für TS-UR38/UR68 und TS-URW38/URW68 Geräte, dient der Vereinfachung der Ansprache der GiS RFID Geräte in beliebigen Softwareanwendungen.

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.

1.1. Fehlerbearbeitung

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

1.2. 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, meist 0-terminiert
PCTSTR	Zeiger auf ein konstantes Feld mit 8 Bit signed character Werten, 0-terminiert.
LPCTSTR	(ANSI C Zeichenkette)

Die Datenübergabe in Feldern erfolgt grundsätzlich LSB zuerst.

Programmierschnittstelle (SDK) der TS-URW Geräte

2. Allgemeine Befehle

int TSURW_LibVersion()
Result -1 Fehler, >0 DLL Versionsnummer z.B. 100 entspricht Version 1.00

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

2.1. Ermitteln der angeschlossenen Geräte

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

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

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

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

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

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

Programmierschnittstelle (SDK) der TS-URW Geräte

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

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

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

Beispiele:

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

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

Programmierschnittstelle (SDK) der TS-URW Geräte

2.2. Funktionen für Ethernet Geräte

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

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

Mit der Funktion **GetLanDeviceNames** kann man sich die LAN-Namen aller LAN-Geräte besorgen. Es werden nur 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 TSURW_GetLanDeviceNamesEx(char* NamenListe, int nBufferSize**)**

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

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

192.168.0.100-10001
[TS-LAN01]169.194.245.01-10001

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

Programmierschnittstelle (SDK) der TS-URW Geräte

int TSURW_ChangeLanIPAddress(PCTSTR OldAddress,
LPCTSTR NewAddress,
LPCTSTR IPMask**)**

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

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

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

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

192.168.0.100-10001
[TS-LAN01]

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

255.255.255.0

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

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

Programmierschnittstelle (SDK) der TS-URW Geräte

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

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

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

int TSURW_IsLanDeviceAvailable(LPCTSTR Address**)**

Adress	IP-Adresse

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

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

192.168.0.100-10001
[TS-LAN01]

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

Programmierschnittstelle (SDK) der TS-URW Geräte

2.3. Kommunikation mit einem Gerät aufbauen

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

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

strInterfaceName	Name der Schnittstelle. z.B. COM1 oder Seriennummer des USB Geräts. Bei **USB** Geräten muss der Gerätename eingetragen werden. Bei **USB HID** Geräten muss der Gerätename gefolgt von "HID" eingetragen werden. Die verfügbaren USB-Namen, kann man mit der Funktion **TSURW_GetUSBDeviceNames** 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 **TSURW_GetLanDeviceNames** ermitteln.
Baudrate	Gewünschte Übertragungsrate. (2400, 4800, 9600, 19200 und 38400 zulässig). Achtung: Nicht alle Geräte unterstützen alle Baudraten. Bitte sehen Sie in der Gerätespezifikation nach den verfügbaren Übertragungsraten. **RS232 Leser sind standardmäßig auf 19200 eingestellt.** **USB und LAN Leser ignorieren die Baudrate (intern fest auf 19200).**
ParityMode	0: 8 Datenbits, 1 Stopbit, keine Parität 1: 8 Datenbits, 1 Stopbit, gerade Parität 2: 8 Datenbits, 1 Stopbit, ungerade Parität 3: 8 Datenbits, 2 Stopbit, keine Parität
Timeout	Zeit nach der die Kommunikation abgebrochen wird. (in Millisekunden)
Rückgabewert:	<0: Fehler, >0: **PortHandle**

Wahlweise kann beim Schnittstellenname noch die Geräteadresse mit angegeben werden. Also z.B.: COM1:4 öffnet an COM 1 das Gerät mit Adresse 4. Dies ist notwendig, wenn die Geräteadresse nicht 1 ist, da bei TSURW_Open 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 G400)

Programmierschnittstelle (SDK) der TS-URW Geräte

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

int TSURW_Close(int PortHandle**)**

PortHandle	Zugriffsnummer
Rückgabewert:	-1 Fehler, 0 OK

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

int TSURW_KeepAppActive (int PortHandle, int bActive**)**

PortHandle	Zugriffsnummer
bActive	0: Applikation wird inaktiv während eine SDK Funktion ausgeführt wird 1: Applikation bleibt aktiv während eine SDK Funktion ausgeführt wird

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

int TSURW_SetReaderAdresse(int PortHandle, int Adresse**)**

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

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

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

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

Baudrate setzen (nur für RS232 Geräte; USB oder LAN Geräte liefern einen Fehler)

Programmierschnittstelle (SDK) der TS-URW Geräte

int TSURW_GetLastError(int PortHandle**)**
PortHandle Zugriffsnummer
Rückgabewert: Fehlernummer wie im Anhang beschrieben

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

Programmierschnittstelle (SDK) der TS-URW Geräte

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

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

Gerätenummer und Gerätefirmware im ASCII Format.

Byte 1 – 3	Gerätenummer
Byte 4	Geräte Firmware (0 – 9,A – Z)

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

Folgende Geräte werden von diesem SDK unterstützt:

Geräte-nummer	Bezeichnung	Lesermodus	Programmer
301	TS-UR38	X	
302	TS-URW38	X	X
311	TS-UR68	X	
312	TS-URW68	X	X

Programmierschnittstelle (SDK) der TS-URW Geräte

2.4. Betriebsart festlegen

int TSURW_SetReaderMode(int PortHandle, int Mode**)**

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

Bei Geräten die sowohl den Lesermodus als auch den Programmiermodus unterstützen kann mit diesem Befehl der Leser bzw. Programmiermodus aktiviert werden.
Im Lesermodus 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 Lesermodus 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 TSURW_SetRF(int PortHandle,int OnOff**)**

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

Ein und Ausschalten des Antennenfeldes.

Programmierschnittstelle (SDK) der TS-URW Geräte

2.5. Geräteparameter setzen

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

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

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

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

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

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

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

Standardeinstellung aktivieren.
Dieser Befehl ist für zukünftige Nutzung reserviert.

Programmierschnittstelle (SDK) der TS-URW Geräte

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

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

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

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

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

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

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

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

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

Programmierschnittstelle (SDK) der TS-URW Geräte

2.6. Allgemeines Kommando absetzen

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

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

PortHandle	Zugriffsnummer
Cmd	Befehl der gesendet werden soll
pSendBuf	Zeiger auf zu sendende Daten
SendBufLen	Länge der zu sendenden Daten
pRecvBuf	Zeiger auf Empfangsdaten
RecvBufLen	Max. Länge der Emfangsdaten
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 TSURW_RawWrite(int PortHandle, BYTE * pBuffer, int BufLen)

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

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

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

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

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

Programmierschnittstelle (SDK) der TS-URW Geräte

2.7. Zugriff auf Register des AS3992 Controllers

Zur Bedeutung der Register siehe Datenblatt zum AS3992 Controller.

int **TSURW_ReadRegister** (int PortHandle, int Register, int *pVal)

PortHandle	Zugriffsnummer
Register	Nummer des Registers
pVal	Zeiger auf Registerdaten
Rückgabewert:	-1 Fehler, >= 0 Lesen erfolgreich

Die Register des AS3992 Controllers sind 1 bis 3 Byte lang, entsprechend werden hier die Daten eingetragen

int **TSURW_WriteRegister** (int PortHandle, int Register, int Val)

PortHandle	Zugriffsnummer
Register	Nummer des Registers
Val	Registerdaten
Rückgabewert:	-1 Fehler, >= 0 OK

Dieser Befehl schreibt ein 1 Byte Register des AS3992 Controllers.

int **TSURW_WriteRegister3**(int PortHandle, int Register, int Val)

PortHandle	Zugriffsnummer
Register	Nummer des Registers
Val	Registerdaten
Rückgabewert:	-1 Fehler, >= 0 OK

Dieser Befehl schreibt ein 3 Byte Register des AS3992 Controllers.

3. Parametereinstellung für Reader Mode

Diese Kommandos dienen der Parametrierung des Gerätes zum Betrieb im Reader Mode.

3.1. Parameter lesen und schreiben

Es können mehrere Parameter Datenstrukturen übergeben werden. Dann werden die angegebenen Transpondertypen abwechselnd ausgeführt.
Wird nur eine Datenstruktur übergeben, so muss diese nicht komplett gefüllt sein. Werden mehrere Datenstrukturen übergeben, so müssen die einzelnen Datenstrukturen immer komplett mit 30 Byte pro Datenstruktur übergeben werden.

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

PortHandle	Zugriffsnummer
pBuffer	Zeiger auf Lesebuffer. Siehe: **3.1.1 Aufbau der Parameter**.
BufLen	Länge des Puffers (30 Byte pro Datenstruktur)
Rückgabewert:	-1 Fehler, Länge der tatsächlich gelesenen Daten.

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

PortHandle	Zugriffsnummer
pBuffer	Zeiger auf Schreibbuffer. Siehe: **3.1.1 Aufbau der Parameter**.
BufLen	Länge des Buffers (30 Byte pro Datenstruktur)
Rückgabewert:	-1 Fehler, 0 OK

3.1.1. Aufbau der Parameter

Die Parameter werden beim schreiben und lesen, immer in derselben Reihenfolge übergeben.

Pos.	Länge	Name	*Beschreibung*
0	1	TTyp	*Transpondertyp:* 00^{Hex} = EPC Gen2 Transponder ID 01^{Hex} = EPC Gen2 Transponder Blöcke
1	16	Register	Der Eintrag besteht immer aus 4 Werten a 4 Byte. Im Byte 0 wird der Speichertyp in den oberen 2 Bit und die Länge in den unteren 6 Bit eingetragen, in Byte 1 – 3 wird die Startadresse im Speicher eingetragen. Ein Registereintrag mit $FFFFFFFF^{Hex}$ wird als Endekennung interpretiert. Kodierung des Speichertyps: 0 = reserviert 2 = TID 1 = EPC 3 = USER
17	1	Ausrichtung	Ausrichtung der Daten 0 = LSB first 2 = LSB first bitgedreht 1 = MSB first 3 = MSB first bitgedreht
18	1	DTyp	*Datentyp:* 0 = Hexadezimal, 3 = Dezimal ohne führenden Nullen, 1 = Dezimal, 4 = Hexadezimal mit Kleinbuchstaben 2 = ASCII,
19	1	Zeichen	*Zeichenanzahl.* (1–32). Hier wird definiert wie viel Zeichen auf einmal ausgegeben werden sollen.
20	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 8
21	1	ValidFrom	(1-16) Startbyte ab dem die Daten übertragen werden.
22	1	TypKenn1	Erkennungszeichen für den Transpondertyp wird vor den Daten übertragen, 1 ASCII Zeichen, bei 0 wird nichts übertragen
23	1	TypKenn2	Erkennungszeichen für den Transpondertyp wird nach den Daten übertragen, 1 ASCII Zeichen, bei 0 wird nichts übertragen
24	6	-	Reserve, Standardwert 0

Programmierschnittstelle (SDK) der TS-URW Geräte

3.2. Prefix

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

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

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

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

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

3.3. Suffix

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

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

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

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

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

Programmierschnittstelle (SDK) der TS-URW Geräte

3.4. Termix

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

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

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

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

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

3.5. Postcode

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

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

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

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

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

Programmierschnittstelle (SDK) der TS-URW Geräte

3.6. Reader Mode Parameter

Schreiben und lesen der Reader Mode Parameter

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

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

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

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

3.6.1. Aufbau der Reader Mode Parameter

Die Parameter werden beim schreiben und lesen, immer in der selben Reihenfolge übergeben.

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

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

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

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

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

Programmierschnittstelle (SDK) der TS-URW Geräte

3.7. Datenübernahme im Readermodus

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

3.7.1. Zyklische Abfrage

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

3.7.2. Callback Funktion einrichten

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

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

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

Definition der Callback Funktion:
'C'

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

'C#'

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

'VB'

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

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

int **TSURW_StopAutoRead**(int PortHandle)

PortHandle	Zugriffsnummer
Rückgabewert:	-1 Fehler, 0: OK

Beendet den Aufruf der Callback Funktion.

3.7.3. LED und Buzzer setzen

Diese Funktionen werden nur bei TS-UR38/URW38 ab Version 1.06 sowie bei TS_UR39/URW39 ab Version 1.00 unterstützt.

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

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

int **TSURW_Beep**(int PortHandle)

PortHandle	Zugriffsnummer
Rückgabewert:	-1 Fehler, 0: OK

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

4. Befehle für alle Transpondertypen

4.1. Inventory

Dieser Befehl liefert die Seriennummer (EPC ID) der vorhandenen Transponder.

int TSURW_Inventory (int PortHandle, int InvType, BYTE * pBuffer, int BufLen**)**

PortHandle	Zugriffsnummer
InvType	Inventory Typ
	0 Neues Inventory anfordern mit Frequenz Hopping
	1 Neues Inventory anfordern
	2 Nächsten EPC aus bestehendem Inventory abrufen
pBuffer	Zeiger auf die Inventory Daten die gelesen werden.
BufLen	max. Länge der Inventory Daten
Rückgabewert	-1 Fehler, >0 Länge der gelesenen Daten

Die Inventory Daten bestehen aus:

1. Byte	Anzahl noch zu lesender Tags im Feld
2. und 3. Byte:	Protocol Control Word
4. Byte bis Ende:	Seriennummer (EPC) des Transponders, LSB first**.**

Dieser Befehl liefert die Seriennummer (EPC ID) der vorhandenen Transponder sowie zusätzliche Informationen über die Signalstärke und Frequenz.

int TSURW_InventoryRSSI (int PortHandle, int InvType, BYTE * pBuffer, int BufLen**)**

PortHandle	Zugriffsnummer
InvType	Inventory Typ
	0 Neues Inventory anfordern mit Frequenz Hopping
	1 Neues Inventory anfordern
	2 Nächsten EPC aus bestehendem Inventory abrufen
pBuffer	Zeiger auf die Inventory Daten Befehl gelesen werden.
BufLen	max. Länge der Inventory Daten
Rückgabewert	-1 Fehler, >0 Länge der gelesenen Daten

Die Inventory Daten bestehen aus:

1. Byte	Anzahl noch zu lesender Tags im Feld
2. Byte bis Len-4:	Seriennummer (EPC) des Transponders, LSB first**.**
4. Byte vor Ende:	Empfangene Signalstärke untere 4 Bit sind I Kanal, obere 4 Bit sind Q Kanal, 2dB pro Schritt.
3 letzte Bytes:	Basisfrequenz auf der der Tag gefunden wurde. Übertragung mit LSB First, Wert in kHz, 868000 entspricht 868 MHz.

Programmierschnittstelle (SDK) der TS-URW Geräte

4.2. Selektieren eines Transponders

int TSURW_Select(int PortHandle, BYTE * pUID, int UIDLen**)**

PortHandle	Zugriffsnummer
pUIDData	Zeiger auf EPC des Transponder (von Inventory)
UIDLen	Länge des EPC
Rückgabewert	-1 Fehler, 0 OK

Mit diesem Befehl wird der Transponder selektiert. Er bleibt solange selektiert, bis er aus dem Feld genommen wird.

Programmierschnittstelle (SDK) der TS-URW Geräte

4.3. Blockweises Lesen und Schreiben

int TSURW_ReadData(int PortHandle, int MemType, int StartByte, BYTE * pBuffer, int BufLen**)**

PortHandle	Zugriffsnummer
MemType	Typ des Speichers der gelesen wird 0: reservierte Memory Bank 1: EPC Memory Bank 2: TID Memory Bank 3: USER Memory Bank
StartByte	Adresse des ersten 16 Bit Wortes das gelesen wird.
pBuffer	Zeiger auf die Transponderdaten
BufLen	Länge der Transponderdaten in Bytes
Rückgabewert:	-1 Fehler, >0 Länge der gelesenen Daten

Die Datenanzahl ist immer gerade, da wortweise gelesen wird. Es werden so viele Byte gelesen, wie in BufLen angegeben wird.

Die Größe der einzelnen Speicherbereiche kann je nach Transponder variieren.

int TSURW_ReadAllData(int PortHandle, int MemType, int StartByte,
BYTE * pBuffer, int BufLen**)**

PortHandle	Zugriffsnummer
MemType	Typ des Speichers der gelesen wird 0: reservierte Memory Bank 1: EPC Memory Bank 2: TID Memory Bank 3: USER Memory Bank
StartByte	Adresse des ersten 16 Bit Wortes das gelesen wird.
pBuffer	Zeiger auf die Transponderdaten
BufLen	Länge der Transponderdaten in Bytes
Rückgabewert:	-1 Fehler, >0 Länge der gelesenen Daten

Hier werden so viele Daten ab StartByte gelesen, wie der Transponder liefert. Damit kann die Speichergröße des Transponders ermittelt werden. Allerdings ist diese Methode nicht sicher, da der Transponder auch manchmal nicht reagiert, obwohl er noch Daten hätte.

Programmierschnittstelle (SDK) der TS-URW Geräte

Die Speicherbank 0 enthält die Passwörter und kann bei aktiviertem Passwortschutz auch nur mit Passwort gelesen werden. Daher kann für Speicherbank 0 folgender Befehl verwendet werden:

int TSURW_ReadDataPW(int PortHandle, int MemType, int StartByte, BYTE * pPasswort, BYTE * pBuffer, int BufLen**)**

PortHandle	Zugriffsnummer
MemType	Typ des Speichers der gelesen wird 0: reservierte Memory Bank
StartByte	Adresse des ersten 16 Bit Wortes das gelesen wird.
pPasswort	Zeiger auf Passwort, das Passwort besteht immer aus 4 Byte.
pBuffer	Zeiger auf die Transponderdaten
BufLen	Länge der Transponderdaten in Bytes
Rückgabewert:	-1 Fehler, >0 Länge der gelesenen Daten

Die Datenanzahl ist immer gerade, da wortweise gelesen wird. Es werden so viele Byte gelesen, wie in BufLen angegeben wird.

Programmierschnittstelle (SDK) der TS-URW Geräte

int TSURW_WriteData(int PortHandle, int MemType, int StartByte ,
BYTE * pBuffer, int BufLen**)**

PortHandle	Zugriffsnummer
MemType	Typ des Speichers der gelesen wird 0: reservierte Memory Bank 1: EPC Memory Bank 2: TID Memory Bank 3: USER Memory Bank
StartByte	Adresse des ersten 16 Bit Wortes das gelesen wird.
pBuffer	Zeiger auf die Transponderdaten
BufLen	Länge der Transponderdaten in Bytes
Rückgabewert:	-1 Fehler, 0 OK

Um Transponderdaten mit diesem Befehl schreiben zu können muss der Transponder selektiert sein wie bei "Select" beschrieben.

int TSURW_WriteDataPW(int PortHandle, int MemType, int StartByte, BYTE * pPasswort,
BYTE * pBuffer, int BufLen**)**

PortHandle	Zugriffsnummer
MemType	Typ des Speichers der gelesen wird 0: reservierte Memory Bank 1: EPC Memory Bank 2: TID Memory Bank 3: USER Memory Bank
StartByte	Adresse des ersten 16 Bit Wortes das gelesen wird.
pPasswort	Zeiger auf Passwort, das Passwort besteht immer aus 4 Byte.
pBuffer	Zeiger auf die Transponderdaten
BufLen	Länge der Transponderdaten in Bytes
Rückgabewert:	-1 Fehler, 0 OK

Um Transponderdaten mit diesem Befehl schreiben zu können muss der Transponder selektiert sein wie bei "Select" beschrieben. Dieses Kommando wird bei zusätzlich mit einem Passwort geschützten Transpondern verwendet.

4.4. Sperren des Transponders

int TSURW_Lock(int PortHandle, int MemType, int LockType, BYTE * pPassword)

PortHandle	Zugriffsnummer
MemType	Speicherbereich auf den das Kommando wirken soll 0: Kill Password 1: Access Password 2: EPC Memory Bank 3: TID Memory Bank 4: User Memory Bank
LockType	Lock Typ, wie in folgender Tabelle beschrieben
pPassword	Zeiger auf 4 Byte Passwort
Rückgabewert	-1 Fehler, 0 OK

Wenn ein Speicherbereich adressiert ist, gelten folgende Einstellungen:

LockType	Beschreibung
0	Speicherbereich ist immer beschreibbar
1	Speicherbereich ist permanent beschreibbar, und kann nicht gesperrt werden.
2	Speicherbereich ist nur im „secured state“ beschreibbar, aber nicht im „open state“.
3	Speicherbereich ist nicht beschreibbar.

Wenn ein Passwort adressiert ist, gelten folgende Einstellungen:

LockType	Beschreibung
0	Passwort Bereich ist immer lesbar und beschreibbar.
1	Passwort Bereich ist permanent lesbar und beschreibbar, und kann nicht gesperrt werden.
2	Passwort Bereich ist im „seucred state“ lesbar und beschreibbar, aber nicht im „open state“.
3	Passwort Bereich ist nicht lesbar oder beschreibbar.

Programmierschnittstelle (SDK) der TS-URW Geräte

4.5. Kill Transponder

int TSURW_KillTag(int PortHandle, int RecomState, BYTE * pPassword)

PortHandle	Zugriffsnummer
RecomState	Wiederinbetriebnahme Status Wert wie in untenstehender Tabelle beschrieben
pPassword	Zeiger auf 4 Byte Passwort für Kill Kommando
Rückgabewert	-1 Fehler, 0 OK

Beachten sie auch die GEN2 Standard Tabelle bei "XPC_W1 LSBs and Tag's recommissioned state"

RecomState	Beschreibung
0	Kill, damit antwortet der Tag nie mehr.
1	Block „permalocking“ ausschalten und alle User Memory Blöcke entsperren, die vorher „permalocked“ waren.
2	User Memory unzugänglich machen
3	User Memory unzugänglich machen
4	EPC, TID und User Speicherbank entsperren, „permalocked“ Speicher behalten den „permalocked“ Status.
5	Entsperren und „permalocking“ ausschalten.

Programmierschnittstelle (SDK) der TS-URW Geräte

5. Fehlerliste

1 Fehler beim Öffnen des Ports
2 Timeout ungültig
3 Porthandle ungültig
4 Timeout
5 Puffer zu klein
6 Puffergröße falsch
8 Keine Sendedaten
10 Checksummenfehler
11 Keine Daten für Kommunikation
12 Prüfpuffer leer
13 Modus ungültig
14 Befehl nicht erlaubt
15 Block Nummer ungültig
20 Keine Daten empfangen

21 Keine Antwort	(NACK	= 15 Hex)
22 Checksummenfehler vom Gerät	(SYNC	= 16 Hex)
24 Kommunikationsabbruch	(CANCEL	= 18 Hex)