Programming Interface (SDK)
for TS-URW devices with 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

Programming Interface (SDK) of TS-URW devices

Revision level:

Docu	SDK Version	Date	Chapter	Name	Info
1.00	1.00	20.11.2012		Blank	First revision released
1.03	1.03	04.11.2013	4.4, 4.5	Blank	Additional commands for LockTag and Kill
1.04	1.04	01.04.2014	4.4	Blank	Description of Lock Tag command corrected
1.05	1.05	03.12.2014	2.2	Blank	Setup of Gateway address
1.05	1.05	27.01.2015	4.3	Blank	Read with Password
1.13	1.13	08.01.2020	3.7	Blank	Usage of Reader mode described Additional Commands included
1.14	1.14	08.06.2020	2.3	Blank	Command TSURW_KeepAppActive added to documentation

Table of contents

Programming Interface (SDK) of TS-URW devices

1. Introduction

The programming interface is build for simple integration of TS-UR38/R68 and TS-URW38/W68 devices in arbitrary applications.

The dynamic link library (DLL) provides all commands and maps the entire functionality of the module. You can use the DLL in several program languages.

1.1. Error treatment

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

1.2. Definitions

The following data types are used in the function declarations:

int	32 Bit Integer
BYTE	8 Bit unsigned integer
BYTE *	pointer to array with 8 Bit unsigned integer values
char *	pointer to array with 8 Bit signed integer values, mostly 0-terminated
PCTSTR	pointer to constant field with 8 Bit signed character values, 0-terminated
LPCTSTR	(ANSI C string)

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

2. General commands

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

Provides DLL version.
This function does not need TSURW_Open.

2.1. Investigate the attached devices

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

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

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

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

Programming Interface (SDK) of TS-URW devices

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

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

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

Examples:

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

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

Programming Interface (SDK) of TS-URW devices

2.2. Functions for Ethernet devices

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

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

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

192.168.0.100-10001
[TS-LAN01]

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

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

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

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

192.168.0.100-10001
[TS-LAN01]169.194.245.01-10001

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

Programming Interface (SDK) of TS-URW devices

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

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

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

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

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

192.168.0.100-10001
[TS-LAN01]

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

255.255.255.0

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

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

Programming Interface (SDK) of TS-URW devices

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

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

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

int TSURW_IsLanDeviceAvailable(LPCTSTR Address**)**

Address	IP-Address

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

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

192.168.0.100-10001
[TS-LAN01]

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

Programming Interface (SDK) of TS-URW devices

2.3. Establish connection to a device

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

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

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

Optionally the device address can be given at the interface name. For example COM1:4 opens at COM 1 the device with address 4. This is necessary if the device address is not 1, because TSURW_Open makes a connection to the device and this is only accepted if the device answers. The function automatically recognizes the under laying protocol of the device (normally GiS LowLevel Protocol G400)

Programming Interface (SDK) of TS-URW devices

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

int TSURW_Close(int PortHandle**)**

PortHandle	Access handle
Return:	-1 error, 0 OK

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

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

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

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

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

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

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

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

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

Set baud rate (only for RS232 devices; USB or LAN devices provide an error).

Programming Interface (SDK) of TS-URW devices

int TSURW_GetLastError(int PortHandle**)**

PortHandle	Access handle
Return:	See error numbers in appendix

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

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

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

Device number and device firmware version in ASCII Format.

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

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

The following devices are supported by this Dll:

Device namuber	Description	Reader mode	Programmer
301	TS-UR38	X	
302	TS-URW38	X	X
311	TS-UR68	X	
312	TS-URW68	X	X

Programming Interface (SDK) of TS-URW devices

2.4. Set operation mode

int TSURW_SetReaderMode(int PortHandle, int mode**)**

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

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

int TSURW_SetRF(int PortHandle,int OnOff**)**

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

Turn antenna field on or off.

Programming Interface (SDK) of TS-URW devices

2.5. Set device parameters

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

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

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

Definition of bits:

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

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

int TSURW_ReadIO(int PortHandle, int * Daten**)**

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

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

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

int TSURW_SetDefault (int PortHandle**)**

PortHandle	Access handle
Return:	-1 Error, 0 OK

Activate Factory default settings.
This command is reserved for future use.

Programming Interface (SDK) of TS-URW devices

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

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

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

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

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

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

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

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

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

2.6. Issue general command

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

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

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

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

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

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

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

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

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

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

2.7. Access to the registers of the AS3992 controller

For information about the meaning of the registers see data sheet for AS3992 controller.

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

PortHandle	Access handle
Register	Index of register
pVal	Pointer to register data
return value:	-1 error, >= 0 reading OK

The registers of the AS3992 controller are 1 to 3 Bytes. The data is set according to the register size

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

PortHandle	Access handle
Register	Index of register
Val	register data
return value:	-1 error, >= 0 writing OK

This command writes a 1 byte register of the AS3992 controller.

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

PortHandle	Access handle
Register	Index of register
Val	register data
return value:	-1 error, >= 0 writing OK

This command writes a 3 byte register of the AS3992 controller.

3. Parameter settings for reader mode

These commands are used for setting parameters of the device in reader mode

3.1. Parameter read and write

If multiple parameter data structures are given, the transponder types are polled alternating. If only one data structure is given, this does not have to be filled completely, if multiple data structures are given, these have to be set completely with 30 Byte per data structure.

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

PortHandle	Access handle
pBuffer	Pointer to read buffer. See: **3.1.1 Parameter structure**.
BufLen	Length of read buffer (30 Byte per data structure)
Return:	-1 error, length of actually read data

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

PortHandle	Access handle
pBuffer	Pointer to write buffer. See: **3.1.1 Parameter structure**.
BufLen	Length of write buffer (30 Byte per data structure)
Return:	-1 error, 0 OK

Programming Interface (SDK) of TS-URW devices

3.1.1. Parameter structure

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

Pos.	Length	Name	*Description*
0	1	TTyp	*Transponder type:* 00^{Hex} = EPC Gen2 Transponder ID 01^{Hex} = EPC Gen 2 Transponder blocks
1	16	Register	The entry consists always of 4 Values with 4 Bytes each. in Byte 0 the memory type if given in the upper 2 bits and the length in the lower 6 bits. in Byte 1 – 3 the starting address in memory is fiven. A register setting with $FFFFFFFF^{Hex}$ is used as end of registers. Coding of memory type: 0 = reserved 2 = TID 1 = EPC 3 = USER
17	1	alignment	Alignment of data 0 = LSB first 2 = LSB first bits flipped 1 = MSB first 3 = MSB first bits flippedt
18	1	DTyp	*Data type:* 0 = Hexadecimal, 3 = Decimal without leading zero, 1 = Decimal, 4 = Hexadecimal with lower case letters 2 = ASCII,
19	1	Characters	*Character count.* (1–32). It is defined how much characters have to be created per block.
20	1	ValidBytes	(1-16) Number of bytes which are used from the UID ord data blocks. Whit this the upper bits of the UID can be masked out. Default value is 5.
21	1	ValidFrom	(1-16) Start byte from which the data it transferred.
22	1	TypKenn1	Distinctive mark for the transponder type which is transferred in front of the data, 1 ASCII character, if set to 0, nothing is transferred
23	1	TypKenn2	Distinctive mark for the transponder type which is transferred after the data, 1 ASCII character, if set to 0, nothing is transferred
24	6	-	Reserved, default value 0

3.2. Prefix

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

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

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

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

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

3.3. Suffix

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

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

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

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

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

3.4. Termix

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

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

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

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

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

3.5. Postcode

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

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

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

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

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

Programming Interface (SDK) of TS-URW devices

3.6. Reader Mode Parameter

Read and write the reader mode parameters.

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

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

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

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

3.6.1. Reader mode parameter structure

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

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

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

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

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

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

3.7. Data reception in Reader mode

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

3.7.1. Cyclic query

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

3.7.2. Set up callback function

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

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

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

Definition of callback function:
'C'

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

'C#'

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

'VB'

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

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

int **TSURW_StopAutoRead**(int PortHandle)

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

Terminates usage of the callback function.

3.7.3. Set LED and buzzer

These functions are only supported at TS-UR38/URW38 Version 1.06
as well as TS-UR39/TS-URW39 Version 1.00 or higher version at these devices.

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

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

int **TSURW_Beep**(int PortHandle)

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

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

4. Commands for all transponder types

4.1. Inventory

This command returns the serial number (EPC ID) of the transponders in field.

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

PortHandle	Access handle
InvType	Inventory Type
	0 Request new inventory with frequency hopping
	1 Request new inventory
	2 Get next EPC from existing inventory
pBuffer	pointer to inventory data to be read
BufLen	max. length of inventory data
Rückgabewert	-1 Fehler, >0 Length of data read

Then inventory data consists of:

1. Byte	remaining tags to be read in field
2. and 3. Byte:	Protocol Control Word
4. Byte to End:	serial number (EPC) of the transponder, LSB first**.**

This command returns the serial number (EPC ID) of the transponders and addition information about the signal strength and frequency.

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

PortHandle	Access handle
InvType	Inventory Typ
	0 Request new inventory with frequency hopping
	1 Request new inventory
	2 Get next EPC from existing inventory
pBuffer	pointer to inventory data to be read
BufLen	max. length of inventory data
Rückgabewert	-1 Fehler, >0 Length of data read

Then inventory data consists of:

1. Byte	remaining tags to be read in field
2. Byte to Len-4:	serial number (EPC) of the transponder, LSB first**.**
4. Byte before End:	Received signal strength. Lower 4 Bit are I Channel, upper 4 Bit are Q Channel, 2dB per step.
3 last Bytes:	Base frequency for this tag. Transmitted with LSB First, value in kHz, 868000 means 868 MHz.

Programming Interface (SDK) of TS-URW devices

4.2. Selecting a transponder

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

PortHandle	Access handle
pUID	pointer to EPC of transponder (from inventory)
UIDLen	length of EPC
Return value	-1 error, 0 OK

With this command the transponder is selected. It keeps selected until it is removed from the field.

Programming Interface (SDK) of TS-URW devices

4.3. Read and write blocks

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

PortHandle	Access handle
MemType	Type of memory to be read 0: reserved Memory Bank 1: EPC Memory Bank 2: TID Memory Bank 3: USER Memory Bank
StartByte	Address of first 16 Bit Word to be read.
pBuffer	Pointer to transponder data
BufLen	Length of transponder data
Return:	-1 error, >0 length of read data

The data amount always has to be even, because the transponder is organized by 16 bit words. As much bytes are read, as given in BufLen.

The size of the memory banks differ on the different transponders.

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

PortHandle	Access handle
MemType	Type of memory to be read 0: reserved Memory Bank 1: EPC Memory Bank 2: TID Memory Bank 3: USER Memory Bank
StartByte	Address of first 16 Bit Word to be read.
pBuffer	Pointer to transponder data
BufLen	Length of transponder data
Return:	-1 error, >0 length of read data

With this command as much data from start byte is read as the transponder delivers data. With this the memory size of the transponder can be found out. But this method is not totally secure, because the transponder sometimes does not respond even if there is more data.

Programming Interface (SDK) of TS-URW devices

Memory bank 0 contains the passwords and can be read only with password if password protection is activated. For this case the following command has been included:

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

PortHandle	Access handle
MemType	Type of memory to be read 0: reserved Memory Bank
StartByte	Address of first 16 Bit Word to be read.
pPasswort	Pointer to password, the password always consists of 4 bytes
pBuffer	Pointer to transponder data
BufLen	Length of transponder data
Return:	-1 error, >0 length of read data

The data amount always has to be even, because the transponder is organized by 16 bit words. As much bytes are read, as given in BufLen.

Programming Interface (SDK) of TS-URW devices

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

PortHandle	Access handle
MemType	Type of memory to be read 0: reserved Memory Bank 1: EPC Memory Bank 2: TID Memory Bank 3: USER Memory Bank
StartByte	Address of first 16 Bit Word to be read.
pBuffer	Pointer to transponder data
BufLen	Length of transponder data
Return:	-1 error, 0 OK

To write transponder data with this command, the transponder has to be selected as described in "Select".

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

PortHandle	Access handle
MemType	Type of memory to be read 0: reserved Memory Bank 1: EPC Memory Bank 2: TID Memory Bank 3: USER Memory Bank
StartByte	Address of first 16 Bit Word to be read.
pPasswort	Pointer to password, the password always consists of 4 bytes
pBuffer	Pointer to transponder data
BufLen	Length of transponder data
Return:	-1 error, 0 OK

To write transponder data with this command, the transponder has to be selected as described in "Select". This command is used at password protected transponder.

4.4. Lock or Unlock

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

PortHandle	Access handle
MemType	Memory space to be addressed 0: Kill Password 1: Access Password 2: EPC Memory Bank 3: TID Memory Bank 4: User Memory Bank
LockType	Locking Type as described in following table.
pPassword	pointer to 4 Byte Password
Return value	-1 error, 0 OK

If memory bank is addressed, values are as followed:

LockType	Description
0	memory bank is writeable from either open or secured states
1	memory bank is permanently writeable from either open or secured states and may never be locked
2	memory bank is writeable from secured state but not from open state
3	memory bank is not writeable from any state

If password is addressed, values are as followed:

LockType	Description
0	password location is readable and writeable from either open or secured states
1	password location is permanently readable and writeable from either open or secured states and may never be locked
2	password location is readable and writeable from secured state but not from open state
3	password location is not readable or writeable from any state

4.5. Kill Tag

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

PortHandle	Access handle
RecomState	recommission state value as described in following table
pPassword	pointer to 4 Byte Password for Kill command
Return value	-1 error, 0 OK

See also GEN2 Standard table at "XPC_W1 LSBs and Tag's recommissioned state"

RecomState	Description
0:	Kill the tag dead so it will not answer anymore.
1:	Disable block permalocking uand unlock all user memory blocks which were previously permalocked.
2:	Make User memory inaccessible
3:	Make User memory inaccessible
4:	Unlock EPC, TID and User memory bank, permalocked banks keep permalocked state.
5:	Unlock and disable permalocking.

5. Error list

1 Error while opening port
2 Timeout span invalid
3 Port handle invalid
4 Timeout
5 Buffer too small
6 Buffer length invalid
8 No send data
10 Checksum error
11 No data for communication
12 Check buffer empty
13 Invalid mode
14 Command not allowed
15 Block number invalid
20 No data received

21 No acknowledge	(NACK = 15 Hex)
22 Checksum error from device	(SYNC = 15 Hex)
24 Cancel	(CANCEL = 18 Hex)