RS-Link

RS232 to Transputer OS-Link Converter

User's Manual

Date: February 02. 2004

Author: Ramona Beyerlein, Edgar Conzen

Important Notes!

The information in this document is subject to change without notice.

Contact

Ingenieurbuero fuer Elektronikdesign Ingo Mohnen

Rottstrasse 33

D-52068 Aachen

Fon: +49 (241) 949 24-0

Fax: +49 (241) 949 24-29

mailto: info@ib-mohnen.de

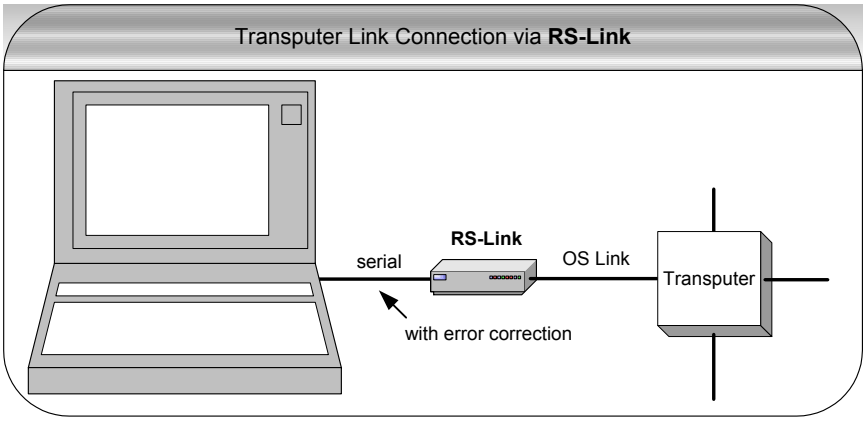
http://www.ib-mohnen.de

Table of Contents

1 General

1.1 Introduction

The RS-Link RS232 to transputer OS-Link converter is a highly developed adapter that enables interfacing a transputer OS-link via standard RS232 as it is found on nearly all computing machines nowadays.



The RS-Link offers many sophisticated features:

- Provides communication between a host computer with a standard serial port and a transputer
- Serial port baud rates from 2400 up to 230400 (fixed settings 8,N,1)
- Fault tolerant protocol through the serial connection
- Data transfer rates up to 8.5 KB/s (RS232) and 12 KB/s (USB)
- Link speed is software switchable between 10 Mbps and 20 Mbps
- Allows software switchable power supply of external devices via OS-Link, thus avoiding the need for an extra power supply for small peripherals
- Basic software package for easy integration into customers' applications under Windows
- Adapted Inmos iserver
- Optional fully opto-isolated OS link
- Optional USB interface instead of RS232

1.2 System Requirements

Before using RS-Link, make sure your system complies with the following requirements:

- One free standard RS232 (optionaly USB) interface
- Microsoft Windows NT version 4.0 or Windows 2000 as operating system

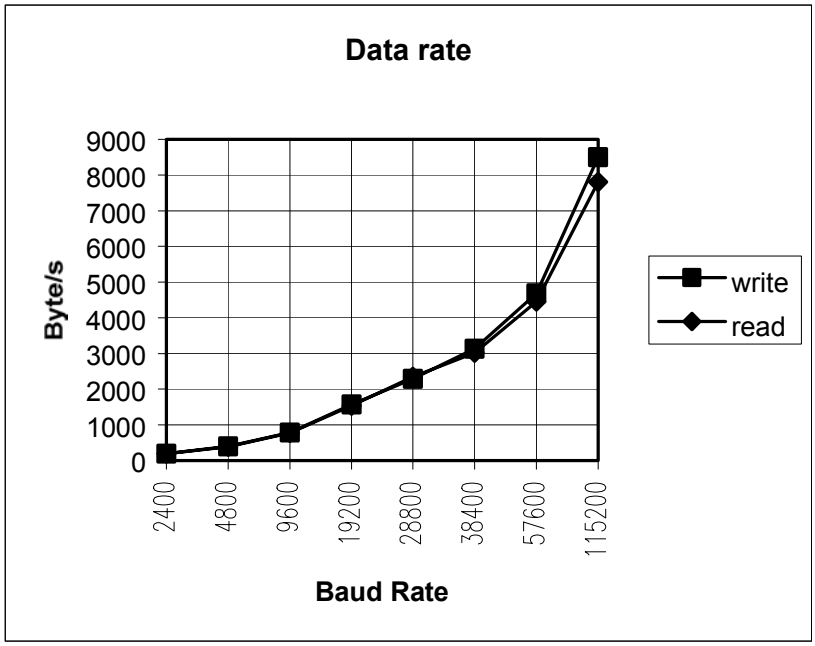
Before using this product, please carefully check that your package includes:

- RS-Link OS link interface
- software package "Ser2Link Access DLL for Windows NT 4.0/Windows 2000"

1.3 Performance

The following diagrams were measured on a ASUS P5A motherboard with a 400MHz AMD CPU and 128MB RAM. A single RS-Link link was connected to a BBK-PCI light interface card. The curves describe the following situations:

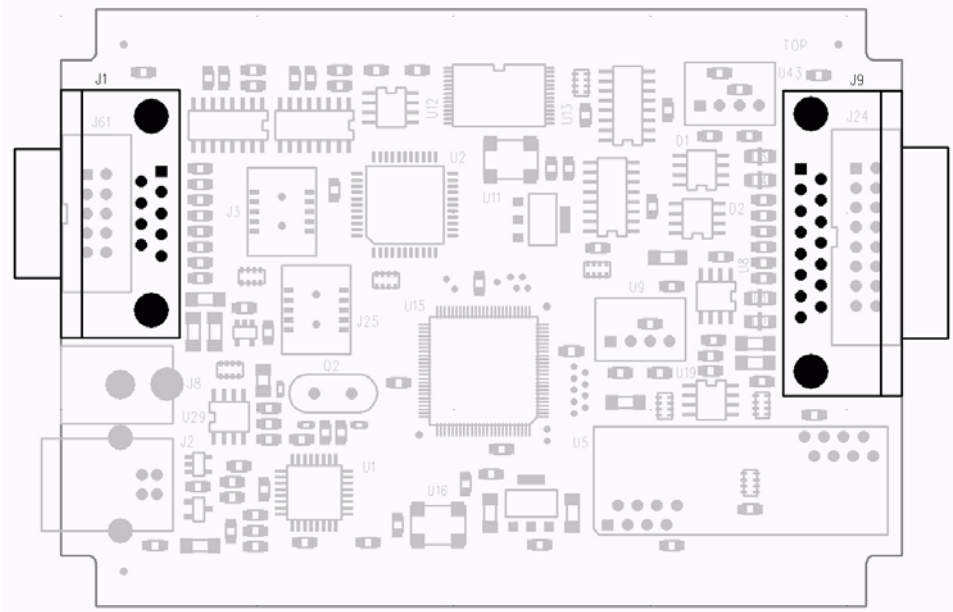
Write	RS-Link writes data to the OS link which is continually accepting data
Read	RS-Link reads data to the OS link which is continually trying to send data



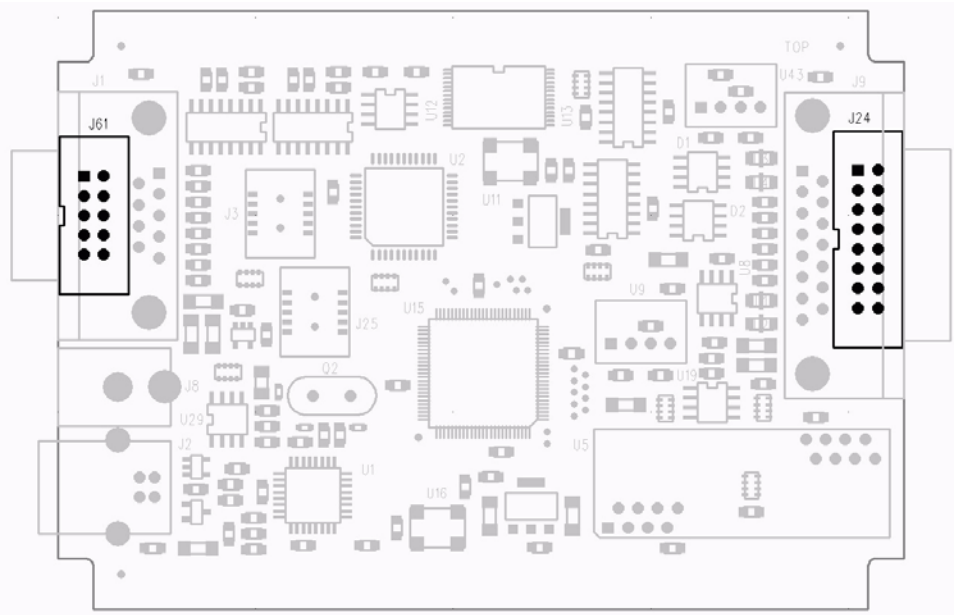
2 Hardware

2.1 General Description.

There are two versions of RS-Link available:
1.) The stand alone version is housed and provided with DSUB connectors.



2.) The OEM version is not housed and provided with pin headers.



2.2 Connector Assignments

COM-Connector

J1 DSUB9 male	J61 Header 2x5	Signal	Description
Pin 1	Pin 1	DCD	not connected
Pin 2	Pin 3	RXD	V.24 input, receive data
Pin 3	Pin 5	TXD	V.24 output, transmit data
Pin 4	Pin 7	DTR	not connected
Pin 5	Pin 9	GND	Signal Ground
Pin 6	Pin 2	DSR	not connected
Pin 7	Pin 4	RTS	V.24 output, not used by software
Pin 8	Pin 6	CTS	V.24 input, not used by software
Pin 9	Pin 8	RI	not connected
	Pin 10	-	not connected

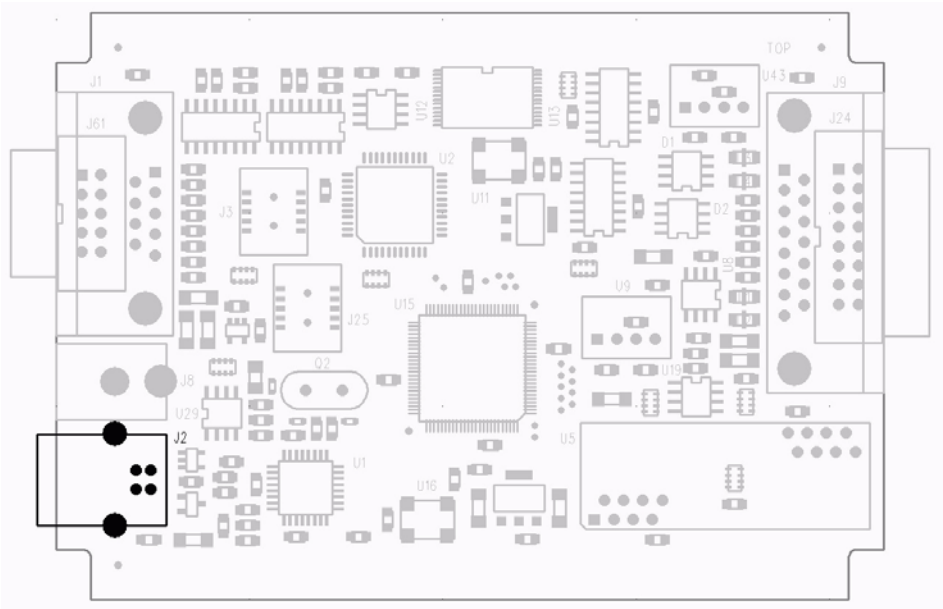
LINK-Connector

J1 DSUB15 female	J61 Header 2x8	Signal	Description
Pin 1	Pin 1	LINKIN+	RECEIVE DATA, RS422 input, true
Pin 2	Pin 3	GND	Signal Ground
Pin 3	Pin 5	n_RESET_OUT+	RESET (active low), RS422 output, true
Pin 4	Pin 7	n_ERROR_IN+	ERROR (active low), RS422 input, true
Pin 5	Pin 9	n_ANA_OUT+	ANALYSE (active low), RS422 output, true
Pin 6	Pin 11	GND	Signal Ground
Pin 7	Pin 13	LINK_OUT+	TRANSMIT DATA, RS422 output, true
Pin 8	Pin 15	VCC	Power supply input
Pin 9	Pin 2	LINKIN-	RECEIVE DATA, RS422 input, inverted
Pin 10	Pin 4	GND	Signal Ground
Pin 11	Pin 6	n_RESET_OUT-	RESET (active low), RS422 output, inverted

Pin 12	Pin 8	n_ERROR_IN-	ERROR (active low), RS422 input, inverted
Pin 13	Pin 10	n_ANA_OUT-	ANALYSE (active low), RS422 output, inverted
Pin 14	Pin 12	GND	Signal Ground
Pin 15	Pin 14	LINK_OUT-	TRANSMIT DATA, RS422 output, inverted
	Pin 16	-	not connected

2.3 Power Supply

There are two ways to provide the power supply to the RS-Link: First via J24 (header 2x8), pin 15 respective J9 (DSUB15) pin 8. Second via J2, an USB type B receptacle, that can be connected to the PC.



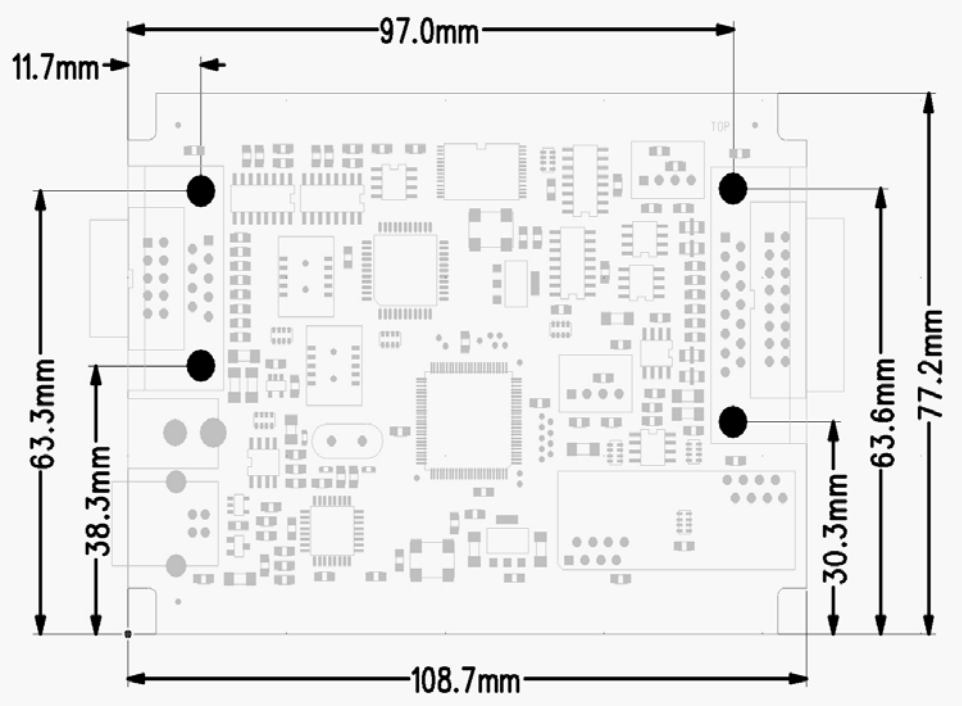
Attention must be paid not to connect the two power sources simultaneously, since this can destroy the power sources or the RS-Link!

2.4 Technical Data

DC supply voltage	5 V ± 5%
Current consumption	350mA typ
Size of OEM version (PCB)	108,7mm x 77,2mm x 16mm
Size of housed version	116mm x 85mm x 35 mm
Operating temperature	0 – 40 °C
Rel. humidity	20 – 80%, non-condensing
Storage temperature	0 – 70 °C
External power supply output	1 × 5V/150mA

2.5 Mechanical Data

In the OEM version the DSUB connectors are not assembled. In this case the holes for the mounting posts can serve as mounting holes



2.6 Installation

Especially when installating the OEM-Version be aware of static electricity. Under the right conditions, static electricity will build up. If you touch the board or its components it will discharge into the components and circuitry. Computer components are sensitive to damage from electrostatic discharge. They can be damaged or destroyed if the discharge is powerful enough. Static build-up is most likely to occur in dryer and cooler conditions, but it is always important to be cautious.

To protect the RS-Link and other components against damage from electrostatic discharge, you should follow some basic precautions whenever you handle them:

1. Use a grounding wrist strap. The strap will have an 'alligator' clip at the end of a shielded wire lead. Clip it to a grounded object. Any static electricity will then harmlessly discharge through the strap. Put on and connect the strap *before* you handle the components.
2. Use an anti-static pad. Put any components on the pad whenever you work on them outside the computer. If you don't have a pad, use the anti-static bag RS-Link came in.

Both the wrist strap and pad are inexpensive and are generally available from computer supply companies.

3 Software

3.1 Introduction

The basis software package includes a dynamic link library which provides all access functions to connect to the RS-Link device, e.g. to transmit data via the RS-Link device and to perform link special operations like resetting an OS link, switch the link speed, switch the external power etc. A host driven protocol with 2 byte CRC error checking is used to ensure reliable performance of all functions of the RS-Link device. It is assumed that the apropriate operating system is properly installed and running on your computer.

The following chapters describe the user interface of the Ser2Link Access DLL for Windows NT 4.0/Windows 2000.

For program development using the Ser2Link interface, the files `ser2link.h`, `ser2link.dll` are delivered with the RS-Link device. Note, that for proper operation the DLL must be in your system path or in the path of your application.

The following files ship with RS-Link device:

bin\ser2link.dll	Ser2Link Access DLL for Windows NT 4.0/Windows 2000
bin\reset.exe	reset link sample application
bin\getinfo.exe	get info sample application
bin\selftest.exe	data transfer sample application
lib\ser2link.lib	linkable library for application development
include\ser2link.h	function prototypes, error codes etc.
sample\	source of sample application files

3.2 User Interface of the Ser2Link Access DLL

3.2.1 Overview

The following API functions are provided will be explained in the following sections:

Function name	Parameter	Short description
HANDLE OpenLink	(char *name)	opens the link
int CloseLink	(HANDLE fd)	closes the link
int ReadLink	(HANDLE fd, void* buf, int size)	reads data from link
int WriteLink	(HANDLE fd, void* buf, int size)	writes data to the link
int ResetLink	(HANDLE fd)	performs a link reset
int SetReset	(HANDLE fd)	sets the link reset signal
int ResetInterface	(HANDLE fd)	resets the link engine
int ResetAdapter	(HANDLE fd)	restarts the firmware
int AnalyseLink	(HANDLE fd)	performs a link analyse
int SetAnalyse	(HANDLE fd)	sets the link analyse signal
int GetSpeed	(HANDLE fd)	retrieves the link speed
int SetSpeed	(HANDLE fd, int speed)	sets the link speed to speed
int TestRead	(HANDLE fd)	tests if a byte is avaliable for reading
int TestWrite	(HANDLE fd)	tests if one byte can be written
int TestError	(HANDLE fd)	retrieves the link error flag
int SetPower	(HANDLE fd, int flag)	sets the link power output on/off
int SetLoop	(HANDLE fd, int flag)	sets the internal link loopback on/off
int SetTimeout	(HANDLE fd, int timeout)	sets the link timeout
int GetTimeout	(HANDLE fd, int timeout)	retrieves the link timeout
int SetFirmwareDefaultBaud	(HANDLE fd, int baud)	sets the default baudrate of the serial connection permanently
int SetBaud	(HANDLE fd, int baud)	sets the baudrate temporarly
int GetBaud	(HANDLE fd)	gets the actual baudrate
char *GetInfo	(HANDLE fd)	get version information

The DLL has the following device defaults set:

```
Link Speed (Mbps)      = 20
Timeout (ms)           = 50000
Baud Rate              = 115200
```

You can use the DLL interface functions `SetSpeed()`, `SetTimeout()`, `SetFirmware-DefaultBaud()` and `SetBaud()` to manipulate these parameters at run-time.

In the following the C-API of the RS-Link device with the logical device name COM<number> is described. Note that error codes are retrieved by calling `GetLastError()`.

3.2.2 OpenLink

```
HANDLE OpenLink (const char *devname);
```

`OpenLink()` opens the device specified and returns a descriptor to use in the other interface functions like `ReadLink()`, `WriteLink()` and `CloseLink()`.

Parameters:

Name	Direction	Description
`name`	Input	a) name of standard serial interface as a '\0'-terminated string, e.g "COM1", or b) '\0'-terminated string containing the name of standard serial interface and the baudrate (if known) , e.g. "COM1 115200"

Return :

This function returns the new descriptor of the link or `INVALID_HANDLE_VALUE`, if an error ocurred. You can retrieve the error code by calling `GetLastError`(), see *Error Codes* for details.

Example:

```
fd = OpenLink("COM1");
if(fd == INVALID_HANDLE_VALUE)
  fprintf(stderr, "Open failed, LastError = %d \n",
                  GetLastError());
```

3.2.3 CloseLink

```
void CloseLink (HANDLE fd);
```

`CloseLink()` closes a file descriptor returned by `OpenLink()`, so that it no longer refers to any device and may not be reused.

Parameters:

Name	Direction	Description
`fd`	Input	descriptor returned by `OpenLink()`

3.2.4 ReadLink

```
int ReadLink (HANDLE fd, char *buf, int count);
```

Reads `size` bytes of data from the RS-Link interface and saves the data read at the memory position pointed to by `buf`. If `count` is zero, `ReadLink()` returns zero and has no other results.

Parameters:

Name	Direction	Description
`fd`	Input	descriptor returned by `OpenLink()`
`buf`	Input	pointer to data buffer
`size`	Input	data size in bytes

Return:

On success the function returns the number of bytes read, or a code < 0 if an error occurred. The error code can be retrieved by calling `GetLastError()`, see *Error Codes* for details.

3.2.5 WriteLink

```
int WriteLink (HANDLE fd, char *buf, int count);
```

`WriteLink()` writes `count` bytes to file descriptor `fd` from the buffer starting at `buf`. If `count` is zero, `WriteLink()` returns zero and has no other effects.

Parameters:

Name	Direction	Description
`fd`	Input	descriptor returned by `OpenLink()`
`buf`	Input	pointer to data buffer
`size`	Input	data size in bytes

Return Values

On success the function returns the number of bytes written, or a code < 0 if an error occurred. The error code can be retrieved by calling `GetLastError()`, see *Error Codes* for details.

3.2.6 ResetLink

```
int ResetLink(HANDLE fd);
```

Performes a reset of the OS-link connected to the RS-Link device. This has to be done prior of booting a transputer.

Parameters:

Name	Direction	Description
`fd`	Input	descriptor returned by `OpenLink()`

Return:

Zero on success or `-1`, if an error ocurred. You can retrieve the error code by calling `GetLastError()`, see *Error Codes* for details.

3.2.7 ResetAdapter

```
int ResetAdapter (HANDLE fd);
```

Resets the RS-Link device, this includes a firmware restart.

For *Parameters* and *Return Values* see `ResetLink()`.

3.2.8 ResetInterface

```
int ResetInterface (HANDLE fd);
```

Resets the OS-link engine. The input and output fifos will be cleared. This may be required if you got a timeout while writing to the OS-link and thus not all bytes were transferred. Furthermore pending read or write operations will be aborted.

For *Parameters* and *Return Values* see `ResetLink()`.

3.2.9 AnalyseLink

```
int AnalyseLink (HANDLE fd);
```

Performs the analyse link sequence: set reset – set analyse (100 ms) - clear analyse –clear reset

For *Parameters* and *Return Values* see `ResetLink()`.

3.2.10 SetReset

```
int SetReset (HANDLE fd, int flag);
```

Sets or clears the link reset signal.

Parameters:

Name	Direction	Description
`fd`	Input	descriptor returned by `OpenLink()`
`flag`	Input	0 – clear , 1 – set

Return:

Zero on success or `-1`, if an error ocurred. You can retrieve the error code by calling `GetLastError()`, see *Error Codes* for details.

3.2.11 SetAnalyse

```
int SetAnalyse (HANDLE fd, int flag);
```

Sets or clears the link analyse signal.

Parameters:

Name	Direction	Description
`fd`	Input	descriptor returned by `OpenLink()`
`flag`	Input	0 – clear , 1 – set

Return:

Zero on success or `-1`, if an error ocurred. You can retrieve the error code by calling `GetLastError()`, see *Error Codes* for details.

3.2.12 GetSpeed

```
int GetSpeed(HANDLE fd);
```

If this function succeeds, the transfer speed of the OS-link interface connected to the RS-Link device is returned. For *Parameters* and *Return Values* see `ResetLink()`.

3.2.13 SetSpeed

```
int SetSpeed(HANDLE fd, int speed);
```

Sets the transfer speed of the OS-link connected to the RS-Link device.

Parameters:

Name	Direction	Description
`fd`	Input	descriptor returned by `OpenLink()`
`speed`	Input	link speed, 10 or 20 can be specified here

Return:

Zero on success or `-1`, if an error ocurred. You can retrieve the error code by calling `GetLastError()`, see *Error Codes* for details.

3.2.14 TestRead

```
int TestRead(HANDLE fd);
```

Tests if data (1 byte) is available on the OS-link for reading.

Return:

Returns 0 if no data is available for reading, returnes 1 if data is available for reading or `-1` if an error occurred. For *Parameters* and *Return Values* see `ResetLink()`.

3.2.15 TestWrite

```
int TestWrite (HANDLE fd);
```

Tests if the 1 byte output fifo of the OS-link interface is empty.

Return Values

The function returns value >0 if a write could be initiated and 0 if a write operation is pending, further information concerning *Parameters* and *Return Values* see `ResetLink()`.

3.2.16 TestError

```
int TestError (HANDLE fd);
```

Retrieves the error flag of the OS-link connected to the RS-Link.

Return Values

The function returns 1 if the link error flag is set and 0 if the link error flag is clear, further information concerning *Parameters* and *Return Values* see `ResetLink()`.

3.2.17 SetPower

```
int SetPower (HANDLE fd, int flag);
```

Switchs the external power supply of the OS-link on or off.

Parameters:

Name	Direction	Description
`fd`	Input	descriptor returned by `OpenLink()`
`flag`	Input	0 – switch the power supply off 1 – switch the power supply on

Return:

Zero on success or `-1`, if an error ocurred. You can retrieve the error code by calling `GetLastError()`, see *Error Codes* for details.

3.2.18 SetLoop

```
int SetLoop (HANDLE fd, int flag);
```

Switchs the internal link loopback mode on or off.

Parameters:

Name	Direction	Description
`fd`	Input	descriptor returned by `OpenLink()`
`flag`	Input	0 – switch the link loopback off 1 – switch the link loopback on

Return:

Zero on success or `-1`, if an error ocurred. You can retrieve the error code by calling `GetLastError()`, see *Error Codes* for details.

3.2.19 SetFirmwareDefaultBaud

```
int SetFirmwareDefaultBaud (HANDLE fd, int baud);
```

Sets the RS-Link firmware default speed of the serial connection.

Parameters:

Name	Direction	Description
`fd`	Input	descriptor returned by `OpenLink()`
`baud`	Input	baud rate of serial connection, see `ser2link.h` for valid values

The default speed is written to RS-Link EEPROM. This baud rate would be configured for the serial connection during the next startup of the firmware, which can be initiated via `ResetAdapter()` or powerup of the RS-Link box.

Return:

Zero on success or `-1`, if an error ocurred. You can retrieve the error code by calling `GetLastError()`, see *Error Codes* for details.

3.2.20 SetBaud

```
int SetBaud (HANDLE fd, int baud);
```

Sets the speed of the serial connection.

Parameters:

Name	Direction	Description
`fd`	Input	descriptor returned by `OpenLink()`
`baud`	Input	baud rate of serial connection, see `ser2link.h` for valid values

Return:

Zero on success or `-1`, if an error ocurred. You can retrieve the error code by calling `GetLastError()`, see *Error Codes* for details.

3.2.21 GetBaud

```
int GetBaud (HANDLE fd);
```

Retrieves the speed of the serial connection.

Return:

If this function succeeds, the baud rateis returned or `-1`, if an error ocurred. You can retrieve the error code by calling `GetLastError()`, see *Error Codes* for details.

3.2.22 SetTimeout

```
int SetTimeout (HANDLE fd, int timeout);
```

Sets the link transfer timeout.

Parameters:

Name	Direction	Description
`fd`	Input	descriptor returned by `OpenLink()`
timeout	Input	timeout in milliseconds, a value of 0 indicates infinite wait for completion

Return:

Zero on success or `-1`, if an error ocurred. You can retrieve the error code by calling `GetLastError()`, see *Error Codes* for details.

3.2.23 GetInfo

```
const char *GetInfo (HANDLE fd);
```

Retrieves the version numbers of the ser2link access DLL and the version of the RS-Link device firmware.

Return:

The functions returns the version info in a '\0'-terminated string.

3.2.24 Error Codes

Error codes are retrieved by calling `GetLastError()`.

Possible Win32 system error codes are:

Error Code	Description
`0`	successful function termination
`ERROR_BAD_UNIT`	no such device
`ERROR_INVALID_HANDLE`	the handle is not a valid handle to a RS-Link device
`ERROR_OUTOFMEMORY`	not enough memory
`ERROR_ARENA_TRASHED`	error in storage control block
`ERROR_INVALID_PARAMETER`	a parameter pointer is `NULL` or a parameter is out of range
`ERROR_NOT_READY`	no firmware connection
`ERROR_TIMEOUT`	the timeout period expired during a communication
`ERROR_BUSY`	the Link interface is busy
`ERROR_OPERATION_ABORTED`	the operation was aborted
`ERROR_NO_MORE_ITEMS`	no more data available
`ERROR_SHARING_VIOLATION`	the device is already open

If you want to print an error description, you can use the Windows API function `FormatMessage()`.

Example:

```
LPTSTR lpBuf;

FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER |
              FORMAT_MESSAGE_FROM_SYSTEM |
              FORMAT_MESSAGE_IGNORE_INSERTS,
              NULL,  GetLastError()  ,
              MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
               (LPTSTR) &lpMsgBuf,  0,   NULL);
if(lpBuf) fprintf(stderr, "%s\n", lpBuf);
```

3.3 Getting started

Connect your RS-Link device via a null modem cable with your PC (e.g. on COM1). Open a dos shell window, change to the bin directory and type

```
selftest COM1
```

This test program resets the link interface, sets the internal link loop connection, tries to transfer some data blocks via the serial line through the internal link loop and back via the serial line to the PC. After completion some data transfer statistics are printed out.

4 Troubleshooting

Common pitfalls are:

- Physical connection incorrect or damaged
- The link speeds on either side of the link do not match