Etherlink

Ethernet to Transputer OS-Link Converter

User's Manual

Date: January 30, 2004

Authors: Ramona Beyerlein, Borut Ivanuscha

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

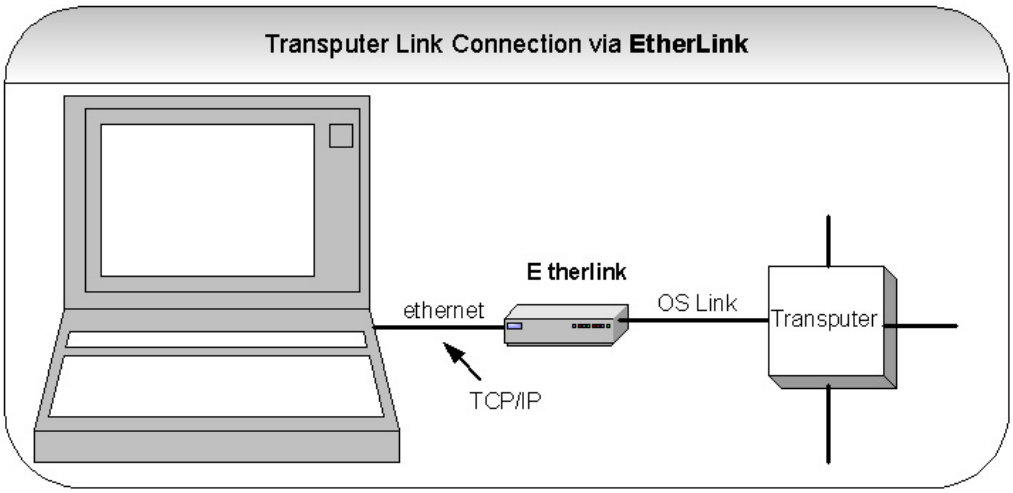
1 General 5
1.1 Introduction 5
1.2 System Requirements 5
1.3 Performance 6
2 Hardware 7
2.1 Standalone Version 7
2.1.1 Technical Data 7
2.1.2 Front Panel 7
2.1.3 Rear Panel 8
2.2 19” Version 8
2.2.1 Technical Data 8
2.2.2 Front View 8
2.2.3 Rear View 8
2.2.4 Top View 9
2.3 Connector Pin Assignments 9
3 Software 10
3.1 Getting Started 10
3.1.1 IP Address 10
3.1.2 Etherlink-Transputer Communication Test 11
3.2 EtherLink DLL API 12
3.2.1 Overview 13
3.2.2 OpenLink 14
3.2.3 CloseLink 14
3.2.4 ReadLink 15
3.2.5 WriteLink 16
3.2.6 ResetLink 17
3.2.7 ResetInterface 17
3.2.8 AnalyseLink 18
3.2.9 GetSpeed 18
3.2.10 SetSpeed 19
3.2.11 TestRead 19
3.2.12 TestWrite 20
3.2.13 TestError 20

3.2.14 SetPower 21
3.2.15 SetLoop 21
3.2.16 SetTimeout 22
3.2.17 GetTimeout 22
3.2.18 Error Codes 23
4 Firmware Update 24
5 Appendices 25
5.1 Connector Pin Assignments 25
5.1.1 Etherlink-10/100-LAN 25
5.1.2 RS-232 Serial Connector 25
5.1.3 Module Power Supply 26
5.1.4 HEMA-Link (Master) 26
5.1.5 External Power Supply 27
5.1.6 External HEMA-Link (Master) 27

1 General

1.1 Introduction

The Etherlink to transputer OS-Link converter is a highly developed adapter that enables interfacing a transputer OS-link via Ethernet as it is found on nearly all computing machines nowadays. The Etherlink firmware operates based on an ARM processor node.



The Etherlink offers many sophisticated features:

- provides communication between a host computer with a standard ethernet port and a transputer
- supports 10/100BaseT Ethernet
- uses the TCP/IP protocol over the ethernet connection
- data transfer rates up to 1 MB/s for large data packets
- 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 (Microsoft Windows)
- adapted Inmos iserver for Windows
- no additional software on transputer needed

1.2 System Requirements

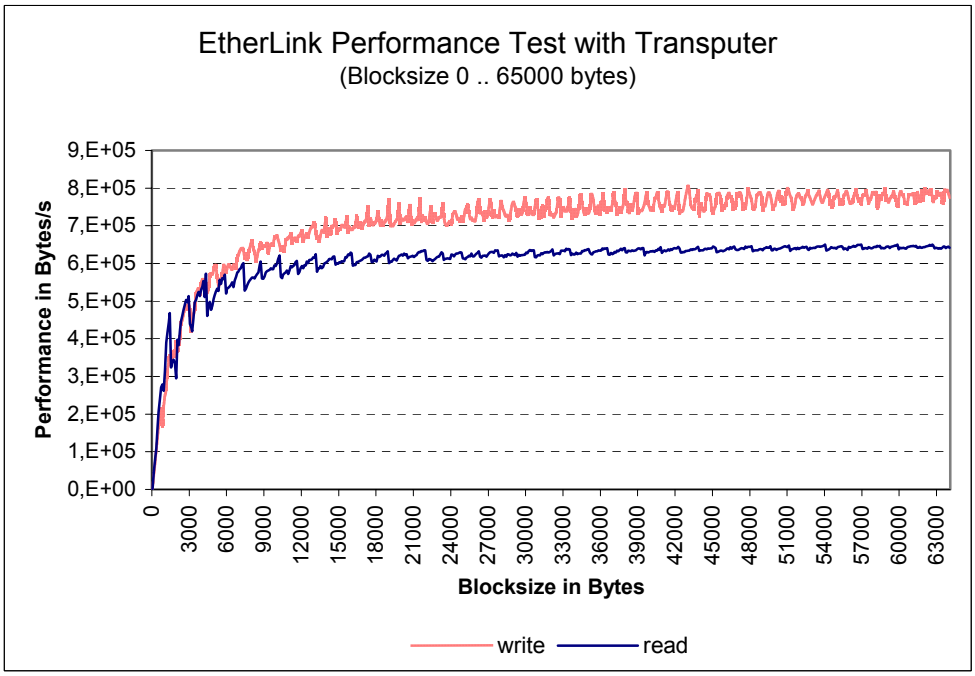
The host operating system is recommended to be Microsoft Windows 2000, but it should work on other Windows platforms as well. Your host PC has to be connected via 10/100BaseT Ethernet to the Etherlink device. You have to provide one IP address per Etherlink device. Before using this product, please carefully check that your package includes:

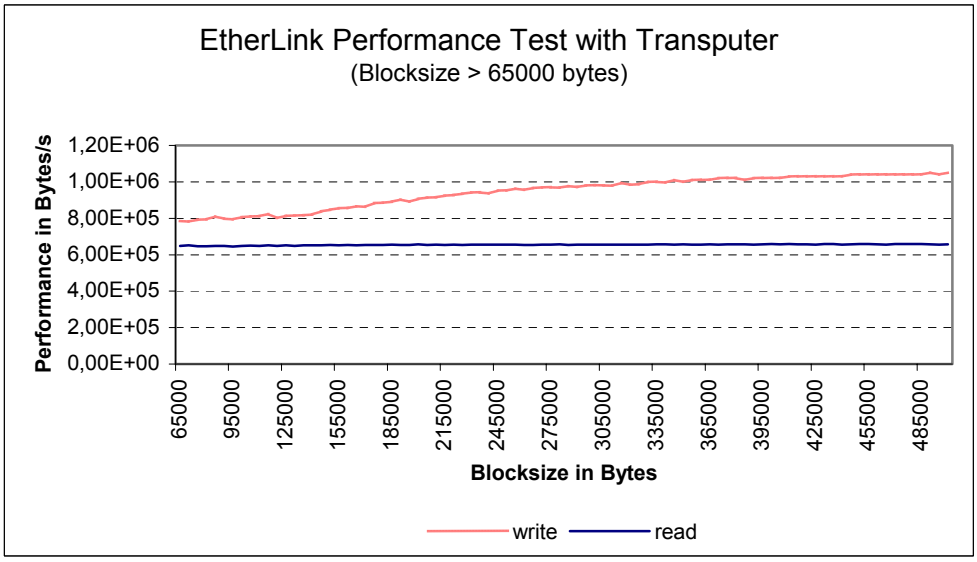
- EtherLink OS link device
- software package "Eth2Link DLL for Windows NT 4.0/2000"

1.3 Performance

The Etherlink performance is restricted to the ARM chip limits, the OS-link speed and it depends on the block size of the data to be transferred. For large data packets you achieve data rates up to 1 MByte per second. For shorter the data packets the protocol overhead causes lower performance values. The following diagram was recorded under Microsoft Windows 2000 with a transputer (T8) connected to the EtherLink device over OS-link with a link speed of 20 Mb/s. Both transfer directions were measured unidirectional.

Note that Ethernet is not real-time-capable!





2 Hardware

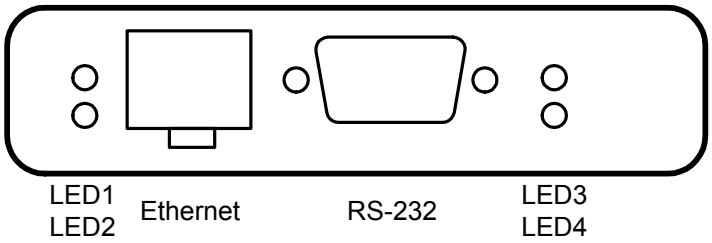
The Etherlink module is shipped either in a case for standalone operation or without case in order to be integrated in customer environment. Due to its dimensions the module is suitable for insertion or assembly in 19” slots and racks.

2.1 Standalone Version

2.1.1 Technical Data

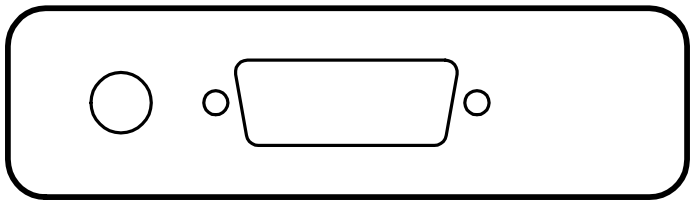
Power supply requirements:	supply voltage	12 VDC (9.5 V - 15 V allowed)
	current consumption	0.25 A typ. / 0.45 A max. (at 12 V)
Dimensions:	length/width/height	175 mm x 110 mm x 35 mm
	weight	350 g

2.1.2 Front Panel



Connector / Indicator	Description
Ethernet	10 or 100 MBit/s RJ45 Ethernet jack
RS232	Serial connector, required for setup purposes
LED1	Ethernet traffic indicator
LED2	Ethernet connection indicator
LED3	Boot/Error indicator
LED4	Functional indicator

2.1.3 Rear Panel



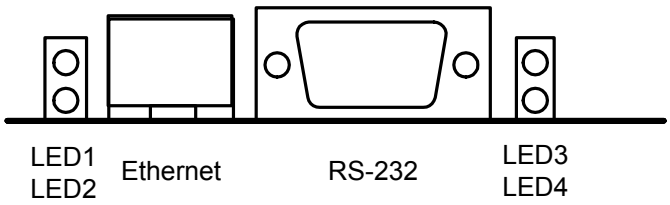
Connector	Description
Power	Power supply for the module (not included in delivery)
Link	HEMA Link Master Port

2.2 19” Version

2.2.1 Technical Data

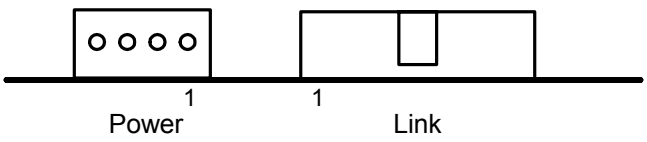
Power Supply Requirements	supply voltage	5 VDC (4.5 - 5.5 V allowed)
	current consumption	0.45 A typ. / 0.85 A max.
Dimensions	length/width/height (above PCB)	120 mm x 100 mm x 14 mm

2.2.2 Front View



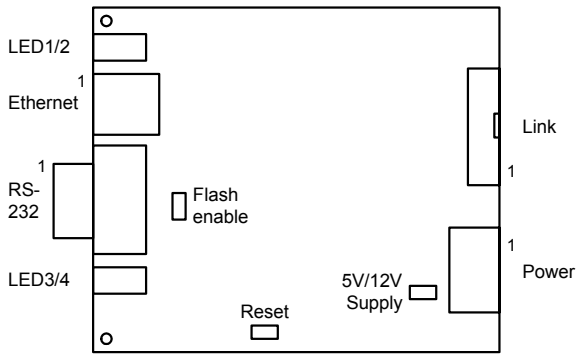
Connector / Indicator / Description	-> same as "Standalone Version"

2.2.3 Rear View



Connector / Indicator / Description	-> same as "Standalone Version"

2.2.4 Top View



Jumper	Description
Flash enable	Must be be closed for normal operation
Reset	Must be left open for normal operation; Shortening these Pins forces a hardware reset (for testing purposes only!)
5 V / 12 V Supply	Must be left open for 5 V Supply (by pins 1+2 of Power connector) Must be closed when supplying 12 V (by pins 3+4 of Power connector)

2.3 Connector Pin Assignments

The detailed connector pin descriptions can be found in appendix:
"Connector Pin Assignments"

3 Software

3.1 Getting Started

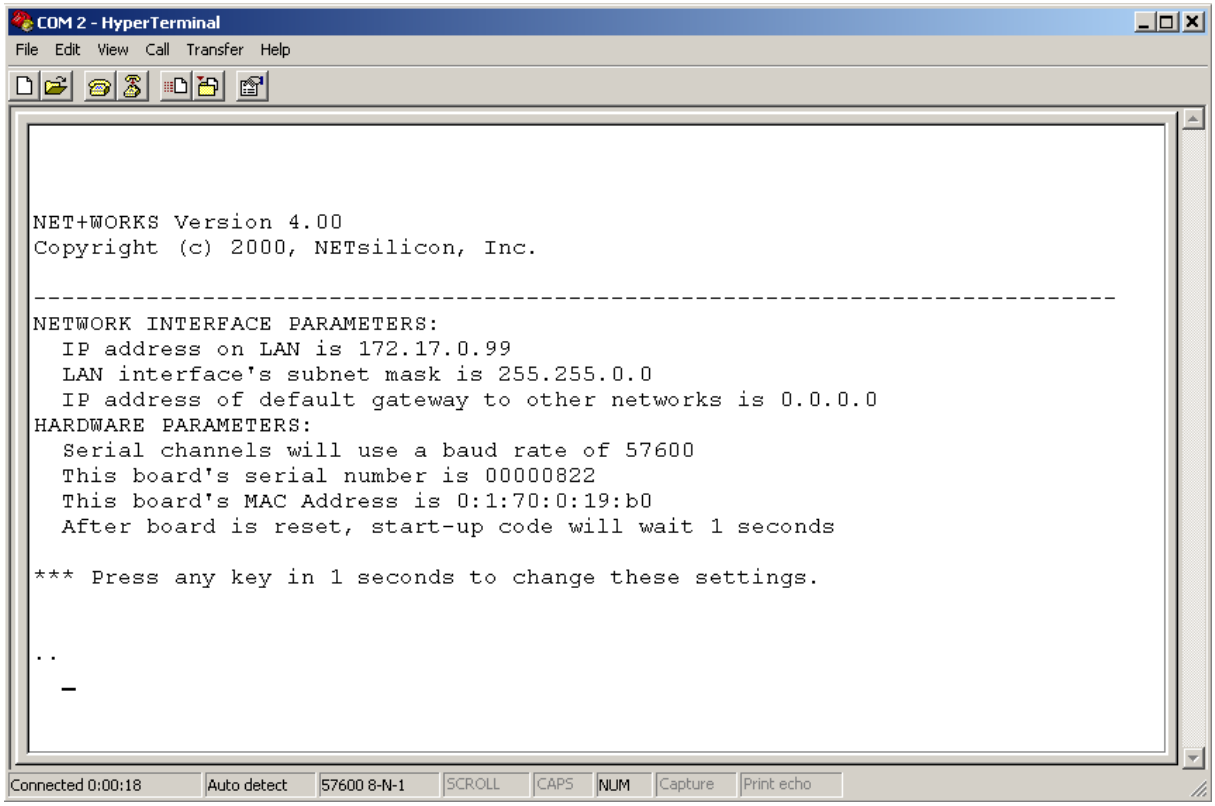
3.1.1 IP Address

The manufacturer has loaded a default IP address into the EtherLink device. Consult your network administrator for a unique IP address to integrate the EtherLink into your company's network. Most likely, you will need to reconfigure the Etherlink default settings for IP address, along with the subnet mask and the gateway, to correspond your specific needs.

The first configuration of the Etherlink networking parameters is done via serial cable, further changes can be done via ethernet.

For the first configuration of the Etherlink parameters, do the following:

1. Connect a terminal or terminal emulator to the serial port of the Etherlink device. (57600 baud, 8-N-1, no flow control)
2. Power up the Etherlink device. The current configuration is displayed on the terminal emulator. The following is an example of the boot up screen of the Etherlink:



1. Press <ENTER> before the timeout expires.
2. Press <M> to modify the configuration settings.
3. Set the desired network parameters (IP address, subnet mask and gateway).
4. Press <ENTER> to continue. The new configuration settings were saved.

Further configurations are done via ethernet with the tool `ethconf.exe`:

1. List all Etherlink devices in your network:

```
c:\etherlink\bin\ethconf -a ETHLINK

Device-Identifier    IP-Address     Subnet Mask   Gateway    DHCP     FW
-------------------------------------------------------------------------
* 00-00-07-92-E8-23 192.168.2.23   255.255.0.0   0.0.0.0    OFF      2.6
```

2. You can change the IP address with:

```
c:\etherlink\bin\ethconf -u 00-00-07-92-E8-23 -i 192.168.2.9
```

For all options available call `ethconf` without parameters.

Note: If the gateway is not set correctly, this method of IP configuration does not work.

3.1.2 Etherlink-Transputer Communication Test

As an example the IP address 192.168.2.23 is used. For a first test of the function of Etherlink, do the following:

1. Power up the Etherlink device. Wait until the firmware is booted (the green LEDs are on).
2. Ping 192.168.2.23 should output something like that:

```
C:\ping 192.168.2.23
Pinging 192.168.2.23 with 32 bytes of data:

Reply from 192.168.2.23: bytes=32 time=10ms TTL=30
Reply from 192.168.2.23: bytes=32 time<10ms TTL=30
Reply from 192.168.2.23: bytes=32 time<10ms TTL=30
Reply from 192.168.2.23: bytes=32 time<10ms TTL=30
```

3. Connect the link port of the Etherlink to a transputer.
4. Boot the example program to the transputer:

```
c:\etherlink\bin\boot 192.168.2.23 drain_source.btl or
c:\etherlink\bin\iserver /sl 192.168.2.23 /sr /sc drain_source.btl
```

   This transputer program reads data from link and writes data to link in an endless loop.

5. Start the communication test for reading data from link (with a block size of 1000 bytes, 10 blocks):
```
c:\etherlink\bin\comtest -d 192.168.2.23 -s 1000 -n 10 -t r
```

6. Start the communication test for writing data to link (with a block size of 1000 bytes, 10 blocks):
```
c:\etherlink\bin\comtest -d 192.168.2.23 -s 1000 -n 10 -t w
```

3.2 EtherLink DLL API

Currently an Etherlink dynamic library for Windows NT4.0/2000 is available. This software should work on other Windows platforms as well but this is an issue to be tested. It will be referred to as EtherLink DLL in the following sections.

For program development under Microsoft Windows using the Etherlink interface, the files `eth2link.h` and `eth2link.dll` (resp. `eth2link.lib` for static linkage) are required.

The software package contains the following files:

`bin/drain_source_t2.btl` `bin/drain_source_t8.btl`	Transputer test program compiled for T2 and T8, which sends and receives data in an endless loop
`bin/iserver.exe`	adapted iserver program (from Inmos)
`bin/eth2link.dll`	EtherLink DLL
`bin/boot.exe`	example program for booting a transputer via EtherLink
`bin/comtest.exe`	simple communications test program
`bin/ethconf.exe`	EtherLink IP address configuration tool
`bin/fwload.exe`	tool for preparing the Etherlink for a firmware update
`bin/ftpdl.exe`	firmare download program (from Netsilicon)
`lib/eth2link.lib`	EtherLink static library
`include/eth2link.h`	header for the EtherLink DLL
`example/boot.c`	example implementation for booting a transputer via EtherLink
`example/resetlink.c`	example implementation for resetting a link via EtherLink
`doc/usermanual.pdf`	this manual

All function prototypes, data structure definitions and macros referred to in the following can be found in `eth2link.h`.

Copy all files in a directory of your choice for installation and make sure, that the EtherLink DLL is in the system path when using the iserver or the other programs.

In the following the implemented C-API calls are described. Note that error codes are retrieved by calling `GetLastError()` in Windows NT or by analysing the error data structure provided within this API.

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)`	open the communication channel to the firmware
`int CloseLink`	`(HANDLE fd)`	close the communication channel opened with `OpenLink`
`int ReadLink`	`(HANDLE fd, void* buf, int size)`	reads `size` bytes of data from link and saves it into the buffer pointed to by `buf`
`int WriteLink`	`(HANDLE fd, void* buf, int size)`	writes `size` bytes of data from the buffer pointed to by `buf` to the link
`int ResetLink`	`(HANDLE fd)`	performs a link reset
`int ResetInterface`	`(HANDLE fd)`	resets the link engine
`int AnalyseLink`	`(HANDLE fd)`	performs a link analyse
`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 communication timeout
`int GetTimeout`	`(HANDLE fd, int timeout)`	retrieves the link communication timeout

3.2.2 OpenLink

```
HANDLE OpenLink(char *name);
```

Opens the communication channels to the EtherLink interface and returns a descriptor (handle), which is to be used with all other functions of the EtherLink API.

Parameters:

Name	Direction	Description
`name`	Input	name or IP-address of the EtherLink device

Return :

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("192.168.0.1");
if(fd == INVALID_HANDLE_VALUE)
  fprintf(stderr, "Open failed, LastError = %d WSALastError = %x\n",
                  GetLastError(), WSAGetLastError());
```

3.2.3 CloseLink

```
int CloseLink(HANDLE fd);
```

Closes the communication channels to the EtherLink interface.

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.

3.2.4 ReadLink

```
int ReadLink(HANDLE fd, void *buf, int size);
```

Reads `size` bytes of data from the EtherLink interface and saves the data read at the memory position pointed to by `buf`.

Parameters:

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

Return:

Count of bytes transferred or `-1`, if an error ocurred. You retrieve the error code by calling `GetLastError()`, see *Error Codes* for details.

Example:

```
char message[100];
int  rc;
rc = ReadLink(fd, message, 100);
if(rc < 100) fprintf(stderr, "Read incomplete or failed.");
```

3.2.5 WriteLink

```
int WriteLink(HANDLE fd, void *buf, int size);
```

Writes `size` bytes of data from `buf` to the EtherLink interface.

Parameters:

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

Return:

Count of bytes transferred or `-1`, if an error ocurred. You retrieve the error code by calling `GetLastError()`, see *Error Codes* for details.

Example:

```
  /* .. */
 char *message = "This is an important message!";

  rc = WriteLink(fd, message, sizeof(message));
  if(rc < 0)
    fprintf(stderr, "Write failed, LastError = %d\n",GetLastError());
  else if(rc < sizeof(message))
    fprintf(stderr, "Write incomplete, only %d bytes of %d written\n",
            rc, sizeof(message));
```

3.2.6 ResetLink

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

Performes a reset of the OS-link connected to the EtherLink 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 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.

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.

3.2.8 AnalyseLink

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

Performes the analyse sequence for the OS-link connected to the Etherlink device.

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.

3.2.9 GetSpeed

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

Returns the transfer speed of the OS-link interface connected to the EtherLink device.

Parameters:

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

Return:

Returns the transfer speed of the OS-link connected to the Etherlink device or `-1`, if an error ocurred. You can retrieve the error code by calling `GetLastError()`, see *Error Codes* for details.

3.2.10 SetSpeed

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

Sets the transfer speed of the OS-link connected to the Etherlink 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.

3.2.11 TestRead

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

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

Parameters:

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

Return:

Returns 0 if no data is available for reading, returnes 1 if data is available for reading or `-1` if an error occurred. You can get the error code by calling `GetLastError`(), see *Error Codes* for details.

3.2.12 TestWrite

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

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

Parameters:

Name	Direction	Description
fd	Input	descriptor returned by OpenLink()

Return:

Returns 0 if no data can be written to the OS-link output fifo, returnes 1 if data can be written to the OS-link output fifo or -1 if an error occurred. You can get the error code by calling GetLastError(), see *Error Codes* for details.

3.2.13 TestError

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

Retrieves the error flag of the OS-link connected to the EtherLink.

Parameters:

Name	Direction	Description
fd	Input	descriptor returned by OpenLink()

Return:

Returns 0 if the error flag is clear, returnes 1 if the error flag is set or -1 if an error occurred. You can get the error code by calling GetLastError(), see *Error Codes* for details.

3.2.14 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.

3.2.15 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.

3.2.16 SetTimeout

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

Sets the transfer timeout of the OS-link interface.

Parameters:

Name	Direction	Description
`fd`	Input	descriptor returned by `OpenLink()`
`timeout`	Input	transfer timeout in milliseconds

Return:

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

3.2.17 GetTimeout

```
int GetTimeout(HANDLE fd);
```

Returns the transfer timeout of the OS-link interface.

Parameters:

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

Return:

Returns the transfer timeout in milliseconds.

3.2.18 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	the IP-address is invalid or the EtherLink firmware is down
ERROR_INVALID_HANDLE	the handle fd is not a valid handle to a EtherLink 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_READ_FAULT	reading a parameter from the EtherLink firmware failed
ERROR_WRITE_FAULT	Sending a command to the EtherLink firmware failed.
ERROR_TIMEOUT	The timeout period expired during a communication.
ERROR_BUSY	The EtherLink interface is busy.

The OpenLink() function initializes the Windows Sockets Library. You can call the `WSAGetLastError()` function to get the error status for the last sockets library operation in case of error. 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,
              WSAGetLastError()  ,
              MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),
              (LPTSTR) &lpMsgBuf,
              0,
              NULL);

if(lpBuf) fprintf(stderr, "%s\n", lpBuf);
```

4 Firmware Update

Firmware updates of the Etherlink device are done via ftp. Therefor the device has to be changed to programming mode using the `fwload` and `ftpdl` utilities:

```
1. c:\etherlink\bin\fwload 192.168.2.23
2. c:\etherlink\bin\ftpdl
```

A GUI starts and has to be supported with parameters like the device IP address and the file name of the firmware binary. Change all settings conforming your needs and click the "Download" button. It will take some minutes to burn the new firmware into the flash and to restart with the updated version.

5 Appendices

5.1 Connector Pin Assignments

5.1.1 Etherlink-10/100-LAN

(RJ45 Jack, Locking = bottom)

Pin No.	Signal / Description
1	TxD +
2	TxD -
3	RxD +
4	not used
5	not used
6	RxD -
7	not used
8	not used

Counting direction: Jack is positioned horizontal, view into opening, Gold contact are up
=> Pin 1 = left

5.1.2 RS-232 Serial Connector

(D-Subminiature, 9 Pins, male)

Pin No.	Signal / Description
1	not used
2	RxD (Input from PC)
3	TxD (Output to PC)
4	not used
5	GND
6	not used
7	RTS (Output to PC)
8	CTS (Input from PC)
9	not used

Counting direction: Connector positioned horizontal, frontal view on pins, long pin row is up
=> Pin 1 = top left

5.1.3 Module Power Supply

Pin No.	Signal / Description
1	+ 5 V
2	GND (5 V)
3	GND (12 V)
4	+ 12 V

Counting direction: Connector positioned horiz., frontal view on pins, ripple profile is down => Pin 1 = right

5.1.4 HEMA-Link (Master)

(Shroud Male Header, 16 Pins)

Pin No.	Signal / Description
1	LINKIN +
2	LINKIN -
3	GND
4	GND
5	\RESOUT +
6	\RESOUT -
7	\ERRIN +
8	\ERRIN -
9	\ANAOUT +
10	\ANAOUT -
11	GND
12	GND
13	LINKOUT +
14	LINKOUT -
15	VCC (Output, switched on by software)
16	not used

Counting direction: Connector positioned horiz., view on Pins from above, notch is down => Pin 1 = bottom left

5.1.5 External Power Supply

(Flange Male Header, 4 Pins)

Pin No.	Signal / Description
1	+12 V
2	+12 V
3	0 V
4	0 V

Counting direction: Connector positioned horiz., frontal view on pins, track/tongue is up
=> Pin 1 = top left, succeeding pins are numbered counter clockwise

5.1.6 External HEMA-Link (Master)

(D-Subminiature female socket, 15 Pins, with attached flat cable)

Pin No.	Signal / Description
1	LINKIN +
2	GND
3	\RESOUT +
4	\ERRIN +
5	\ANAOUT +
6	GND
7	LINKOUT +
8	VCC (internally switched 5 V-output)
9	LINKIN -
10	GND
11	\RESOUT -
12	\ERRIN +
13	\ANAOUT -
14	GND
15	LINKOUT -

Counting direction: Connector positioned horiz., frontal view on sockets, short pin row is up
=> Pin 1 = bottom left