gb/us

Comprehensive User's Guide XLON® PC/104 Adapter

Version 1.0

DH electronics GmbH

Am Anger 8
83346 Bergen
Germany
Tel.: +49 8662 4882-0
Fax.: +49 8662 4882-99
E-Mail: info@xlon.de
www.xlon.de

Comprehensive User's Guide XLON® PC/104 Adapter

Version 1.0

Table of Contents

XLON® PC/104

gb/us

1 About this manual

This guide describes the hardware installation, software driver installation and the setup and configuration of the XLON® PC/104 adapter.

The developer is given information about creating suitable application software.

Used ideograms and symbols

In this manual the following ideograms and symbols are used to emphasize special points.

⚠	Attention! Very important point concerning safety.
⚡	Danger of injury caused by voltage.
⚠	Danger of damage to electronic components caused by static loading.
⚠	Danger of injury caused by mechanic components.
•	enumeration, working step
☞	remarkable instruction

2 Introduction / Product Information

The XLON® PC/104 LonTalk® adapter can be used to connect an embedded PC to a LonWorks® network via the PC/104 bus, according to P996 PC standard. It is designed for use in industrial control, process control and building automation.

The XLON® PC/104 supports not only the LNS Network Services Interface (NSI) for all LNS tools, but also the LonManager®-API- and DOS interface on older applications.

Thanks to its Client-Server-Architecture, the LNS network operating system provides simultaneous access to highly diverse applications on the Network-Services-Server (NSS). As a result LonWorks® network tools produced by different manufacturers can be simultaneously implemented for installation, maintenance, monitoring and control.

By utilizing the XLON® PC/104 it is also possible, to transform a PC into an extremely efficient LonWorks® node. In this case, the LonWorks® application runs on the PC and the XLON® PC/104 handles the operation of the LonTalk® protocol. This provides much more processing power for a LonWorks® application, in comparison to a Neuron® chip based node. In addition, the number of possible network variables is been considerably increased from 62 up to 4096, which can frequently play an important role when it comes to maintenance and monitoring applications.

The XLON® PC/104 has an integrated FTT-10A transceiver for Free Topology and Link Power networks or a RS485 transceiver for Twisted Pair networks.

In order to lower the expense for peripheral components in embedded PC applications, the user has additional 8 free configurable digital TTL IO's available. The inputs are able to generate an interrupt. More complex hardware functions like counters, encoders, digital filters etc. can be implemented on request.

It is possible to connect an external Service button and an external Service LED for use with manual installations and for visualization of the state of the LonTalk® adapter. To check the status of the XLON® PC/104 adapter a Service LED and a State LED is onboard. An extra LED displays any bus traffic on the network.

All available drivers are included in the XLON® PC/104 Kit. Sample programs for accessing the driver with C/C++ and VisualBasic can be downloaded from:

http://www.xlon.de

Scope of delivery

- XLON® PC/104 controller card
- Weidmueller-clamp for LonWorks® network connection
- Floppydisk/CD with device drivers
- User's manual for installation.

Available variants

- PC12-WM-FTT with integrated FTT-10A Transceiver
- PC12-WM-RS with integrated RS485 Transceiver

Find more informations about LonWorks® networks under:

www.echelon.com

3 Installation of the XLON® PC/104 adapter

3.1 Hints

For installation of the XLON® PC/104 adapter the embedded computer needs to have a PC/104 extension connector.

Look at the manual of your computer manufacturer for questions concerning running down your operation system and opening the PC or the arrangement of connectors.

Before opening the device please make sure:

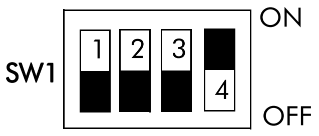
- Shut down the operating system and turn off the PC.
- Remove the power cord
- This product contains devices which are sensitive to static electricity. Before installing or removing the XLON® PC/104 adapter or the network cables, discharge any static electricity which may have accumulated to earth ground (for example by touching the metallic frame of the computer).

3.2 Hardware configuration

Before installing the hardware of the XLON® PC/104 adapter, you have to configure the base address. The XLON® PC/104 adapter occupies a block of 16 addresses in the I/O space of the host PC. You have the possibility to select between seven base addresses. The base address of this block may be set with DIP switches located on SW1 (see 3.3, (4)). Have a look at the User's Manual of your embedded PC, to get informations about usable base adresses for extension cards. If another device is allocated to an address range of the XLON® PC/104 adapter addresses, neither device will work properly.

To configure the base address of the XLON® PC/104 adapter registers, set the switches at SW1 as shown in the table below.

Set the IO address of the XLON® PC/104 card



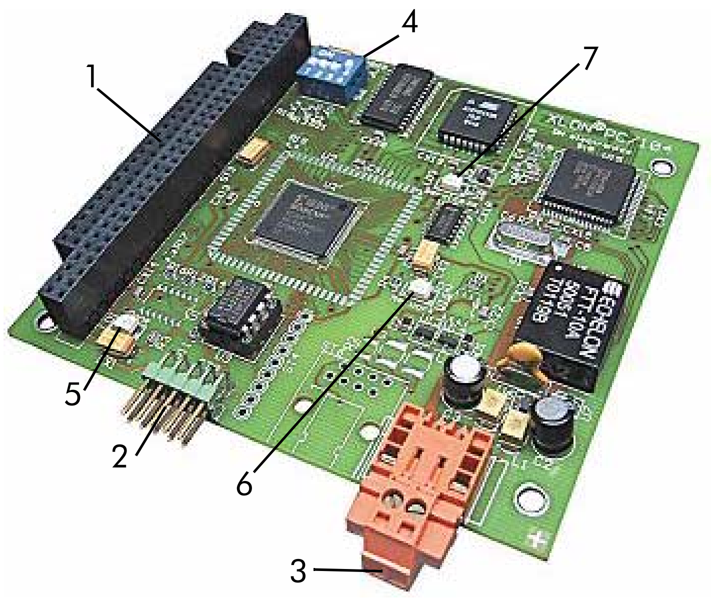
Base address	Settings of DIP switch SW1			
	1	2	3	4
0x200 (default)	OFF	OFF	OFF	ON
0x200	ON	ON	ON	ON
0x300	ON	OFF	OFF	ON
0x310	OFF	ON	OFF	ON
0x320	ON	ON	OFF	ON
0x330	OFF	OFF	ON	ON
0x340	ON	OFF	ON	ON
0x350	OFF	ON	ON	ON

The default assignment for the XLON® PC/104 adapter block of addresses is 0x200 - 0x20F.

The XLON® PC/104 adapter occupies one interrupt of the host PC. You have the possibility to select between four interrupts 5 ,10 (default) 11 and 15. The interrupt must be set during the software installation.

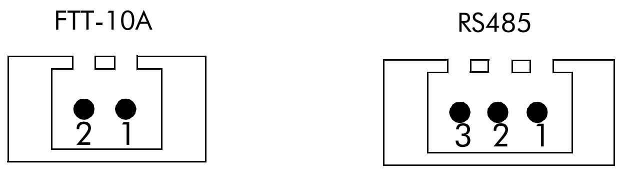
3.3 Hardware installation

Please be sure to observe all the embedded PC manufacturers instructions regarding the insertion of additional interface cards.



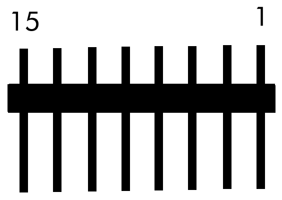
No.	Term	Remark
1	PC104 connector ST2, ST3	P996 PC standard
2	Extension connector ST8	Description see graphic and table below
3	LON® network connector ST5	2-pin: FTT-10A 3-pin: RS485
4	Address DIP switch SW1	Possible settings see 3.2
5	LED red „Done“ LD1	State of onboard logic
6	LED green „Network“ LD2	Display data traffic on the LonWorks® network
7	LED yellow „Service“ LD3	Display Service Pin Neuron processor

Pin allocation of the LonWorks® connector

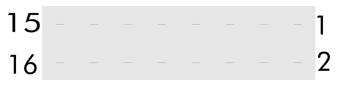


Pin	FTT-10A	RS485
1	NET A	RS485 A
2	NET B	RS485 B
3	not existing	GND

Pin allocation of the extension connector ST8



Topview



Frontview

Pin	Describtion	Pin	Describtion
1	power supply VCC (5.0V/200 mA DC)	9	programmable IN-/OUTPUT 8
2	programmable IN-/OUTPUT 1	10	power supply GND
3	programmable IN-/OUTPUT 2	11*	not connected
4	programmable IN-/OUTPUT 3	12*	not connected
5	programmable IN-/OUTPUT 4	13*	connection Service Pin Neuron
6	programmable IN-/OUTPUT 5	14*	connection LED 'Network'
7	programmable IN-/OUTPUT 6	15*	connection LED 'Done'
8	programmable IN-/OUTPUT 7	16*	not connected

*) available from hardware version -500

- Open PC, if neccessary
- Insert the XLON® PC/104 adapter into the PC/104 extension connector accurately
- Fix the XLON® PC/104 adapter with four screws to the PC/104 board below it
- Plug in the LonWorks® network cable
- Plug in the extension connector cable, if used
- Close PC, if neccessary
- Boot up the computer
- Under Windows based operating systems start 'Setup.exe' from the driver disk (see chapter 3.4)
- For installation of device drivers under Windows CE refer to chapter 3.4.2
- For installation of device drivers under LINUX refer to chapter 3.4.3
- For installation of device driver under DOS refer to chapter 3.4.4

De-installation

De-installation of the XLON® PC/104 adapter follows reverse order above. See chapter 3.5 for De-installation of driver software.

3.4 Software Installation

All device drivers supplied with the XLON® PC/104 adapter have been designed in accordance with the specifications of Echelon Corporation.

Before installing the software of the XLON® PC/104 adapter, you have to check possible interrupt settings for the XLON® PC/104 adapter. The XLON® PC/104 adapter occupies one interrupt of the host PC. You have the possibility to select between four interrupts 5 ,10 (default) 11 and 15. The interrupt must be set during the software installation. Have a look at the User's Manual of your embedded PC, to get informations about usable interrupts for extension cards. If another device is allocated to the same interrupt as the XLON® PC/104 adapter, neither device will work properly.

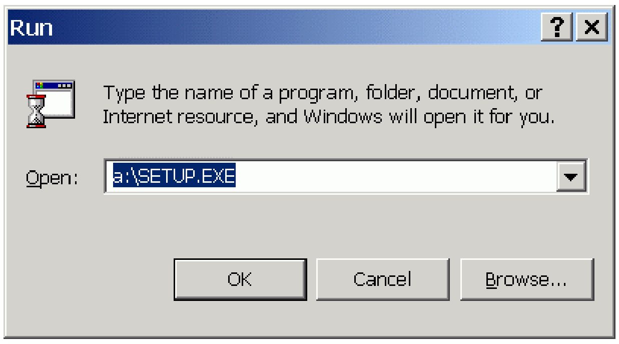
If you are using a host PC with an integrated PCI bus, you have to reserve the IRQ which you want to use in the BIOS. The reason for this is because the XLON® PC/104 is an ISA device. To get informations on how to do this, have a look at the User's Manual of your embedded PC. For example in an Award BIOS you find the setting under 'PNP/PCI Configuration -> IRQx assigned to: Legacy ISA'. In a Phoenix BIOS you find the setting under 'Advanced -> PCI Configuration -> PCI/PNP ISA IRQ Ressource Exclusion -> IRQx [Reserved].

3.4.1 Initial installation and configuration under Windows

3.4.1.1 Initial Installation

Initial installation under Windows 2000 is shown exemplary below. Installation under other Windows operating systems works analogously!

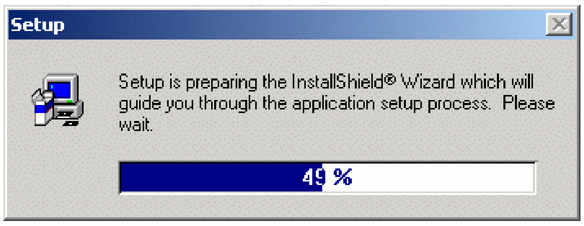
Insert the driver disk into your floppy drive. Start 'Setup.exe' as shown below:


Run

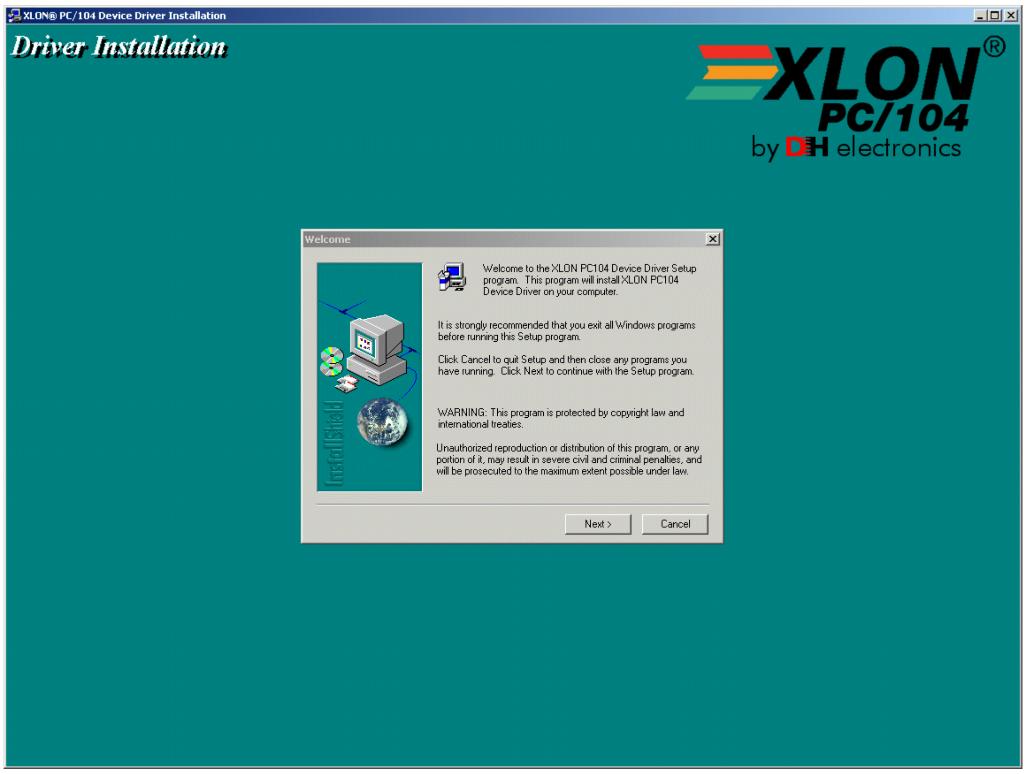
Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: a:\SETUP.EXE

The driver installation starts:


Setup

Setup is preparing the InstallShield® Wizard which will guide you through the application setup process. Please wait.


XLON® PC/104 Device Driver Installation
Driver Installation
XLON® PC/104 by DH electronics
Welcome
InstallShield
Next >
Cancel

Welcome to the XLON PC104 Device Driver Setup program. This program will install XLON PC104 Device Driver on your computer.

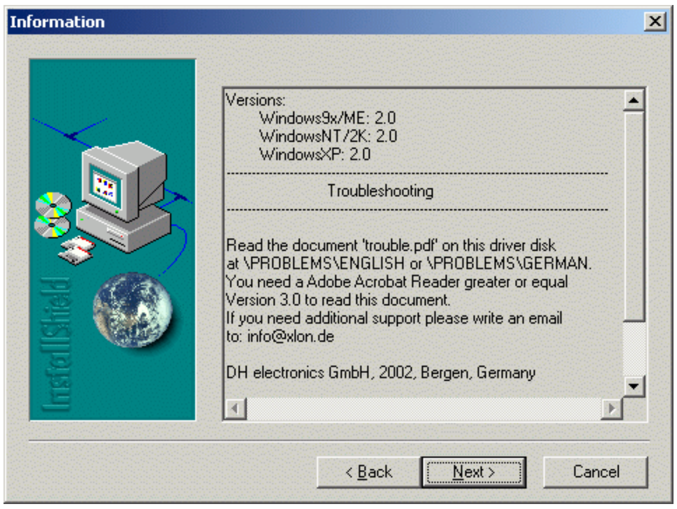
It is strongly recommended that you exit all Windows programs before running this Setup program.

Click Cancel to quit Setup and then close any programs you have running. Click Next to continue with the Setup program.

WARNING: This program is protected by copyright law and international treaties.

Unauthorized reproduction or distribution of this program, or any portion of it, may result in severe civil and criminal penalties, and will be prosecuted to the maximum extent possible under law.

- Click on „Next>“



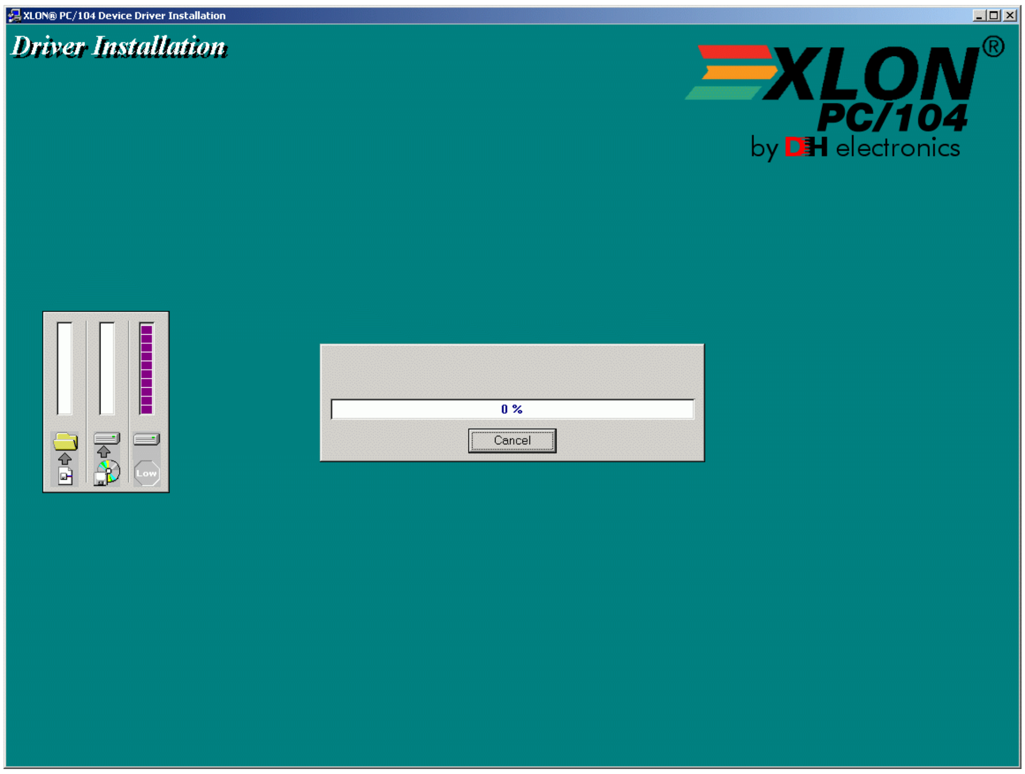
Versions:
    Windows9x/ME: 2.0
    WindowsNT/2K: 2.0
    WindowsXP: 2.0

Troubleshooting

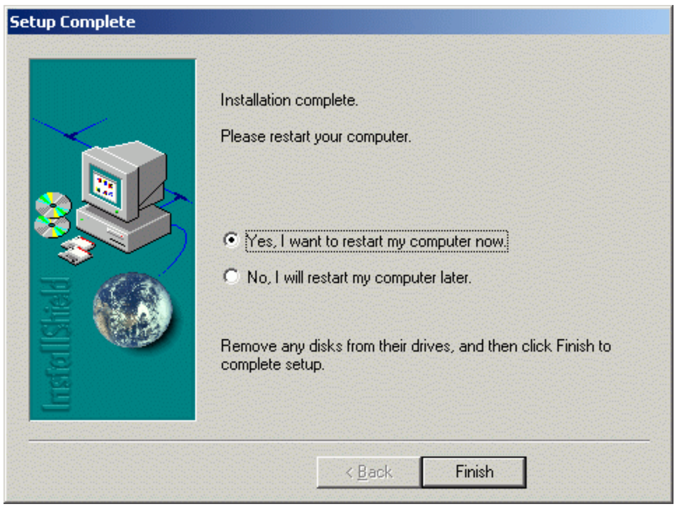
Read the document 'trouble.pdf' on this driver disk at \PROBLEMS\ENGLISH or \PROBLEMS\GERMAN.
You need a Adobe Acrobat Reader greater or equal Version 3.0 to read this document.
If you need additional support please write an email to: info@xlon.de

DH electronics GmbH, 2002, Bergen, Germany

- Read the informations and click on „Next>“



- The installation process copy the files to your system


Setup Complete

Installation complete.

Please restart your computer.

[✓ Yes, I want to restart my computer now.]

No, I will restart my computer later.

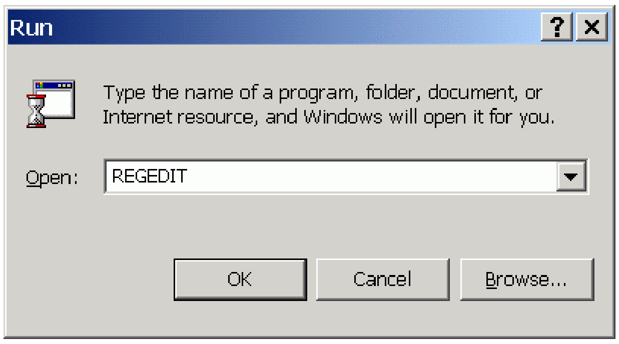
Remove any disks from their drives, and then click Finish to complete setup.

- Select between 'Yes, I want to restart my computer now' or 'No, I will restart my computer later'. For loading the device driver a restart of your system is neccessary.

- Click on „Finish"

3.4.1.2 Configuration of the device driver

- Start 'Regedit.exe' program as shown below



Run

Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: REGEDIT

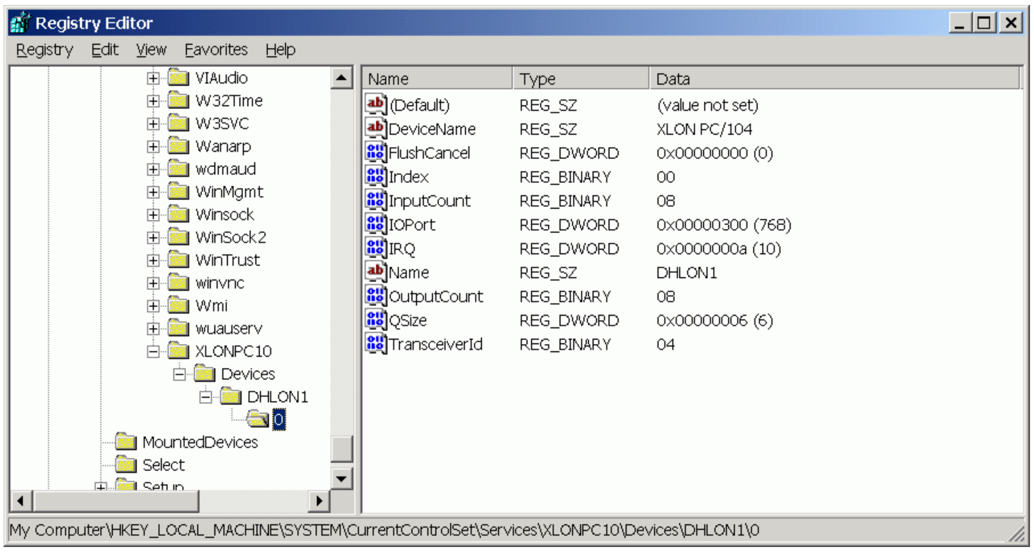
- Navigate to the following registry key:

  Windows9x/ME:

  HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Class\LonNIDH\0001\

  WindowsNT/2000/XP:

  HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\XLONPC10\Devices\DHLON1\0\



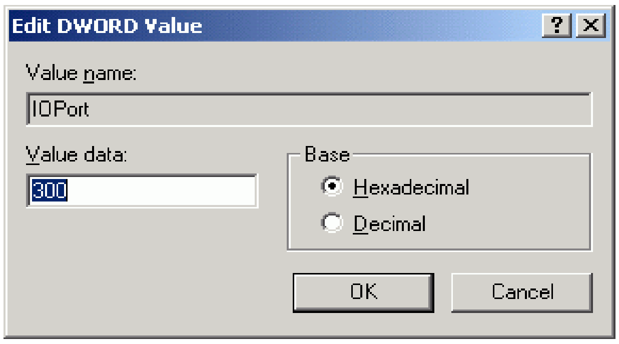
Registry Editor

Name	Type	Data
(Default)	REG_SZ	(value not set)
DeviceName	REG_SZ	XLON PC/104
FlushCancel	REG_DWORD	0x00000000 (0)
Index	REG_BINARY	00
InputCount	REG_BINARY	08
IOPort	REG_DWORD	0x00000300 (768)
IRQ	REG_DWORD	0x0000000a (10)
Name	REG_SZ	DHLON1
OutputCount	REG_BINARY	08
QSize	REG_DWORD	0x00000006 (6)
TransceiverId	REG_BINARY	04

My Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\XLONPC10\Devices\DHLON1\0

Under these registry keys you can modify the settings for 'IO Port' and 'IRQ'.

- Select 'IO Port' by doubleclick and configure the base address (as configured in 3.2):



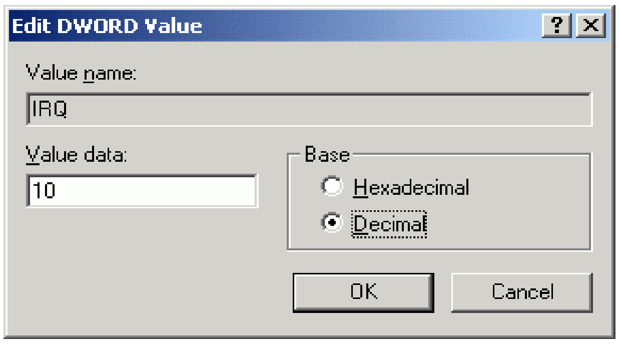
Value name: IOPort

Value data: 300

Base: (● Hexadecimal) ○ Decimal

Possbible settings are: 200, 300, 310, 320, 330, 340, 350 (hexadecimal)

- Select 'IRQ' by doubleclick and configure the interrupt (as described in 3.4):



Value name: IRQ

Value data: 10

Base: ○ Hexadecimal (● Decimal)

Possbible settings are: 5, 10, 11, 15 (decimal)

- You can configure other parameters (Flush Cancel, Input Buffer Count, Output Buffer Count, Transceiver ID) in the same way (see chapter 4.1.4).

- The software installation and configuration is finished now.

- After doing this settings Reboot your computer.

3.4.2 Initial Installation under Windows CE 3.0

The device driver for Windows CE 3.0 is implemented in the shape of a „Stream Interface Device Driver" as Dynamic Link Library (DLL) for the following processor platforms:

- x86
- SH3

The files required for installation under Windows CE 3.0 can be downloaded from the website www.xlon.de .

The device driver supports up to 7 XLON® PC/104 adapters within one system. The base addresses must be configured as described in 3.2. The interrupts needs to be reserved as described in 3.4 and are allocated automatically corresponding to the registry settings. Although Windows CE 3.0 and PC/104 (ISA) bus does not support a „native" interrupt sharing, the particular structure of the device driver enables interrupt sharing between any number of XLON® PC/104 adapters.

The device driver can either be added dynamically to a running Windows CE System or can be statically tied up in the Windows CE Image which has to be created. The second operation should only be done by experienced users who want to create a new Windows CE 3.0 Image. Both operations are explained below.

3.4.2.1 Installing the driver to a running Windows CE System

For XLON® PC/104 driver installation on a Windows CE device with static RAM the driver file „xlon_pc104.dll" has to be copied manually into the directory „\Windows". This has to be followed by registration entry as explained in chapter 3.4.2.3.

3.4.2.2 Creating a new Windows CE Image

For making the device driver available to the XLON® PC/104 adapter under „Microsoft Platform Builder 3.0" the below steps have to be followed. They refer to a x86 Hardware Platform. The procedure is analog under other hardware architectures. Depending on the specific platform directory paths may however differ.

- Copy device driver file „xlon_pc104.dll" to directory „_WINCEROOT\PLATFORM\CEPC\FILES".
- Copy component file „xlon.cec" to directory „CEPBDir\CEPB\ CEC".
- Start „Microsoft Platform Builder 3.0" and load your Platform working area.
- Open the „File" Menu and go to „Manage Platform Builder Components"
- Import the component file „xlon.cec" by clicking „Import New"
- For the following step you need informations about base addresses and interrupts available for the XLON® PC/104 adapter. Find detailed information under chapter 3.4.2.4.
- Under the „Platform Builder" manually add Windows CE Registry Information to file „platform.reg" as described in chapter 3.4.2.3.
- Under the „Platform Builder" manually add the content of file „xlon_pc104.bib" to file „platform.bib".
- Under the „Platform Builder" open menu „View", and click on „Catalog". The catalog view should open.
- In Catalog, open the Tree View „Catalog/Drivers/CEPC/XLON".
- Add the „XLON® PC/104" component to the current platform by clicking the right mouse button on „Add to Platform".
- Restart the „Platform Builder", create your new Windows CE 3.0 Image and load it on your target system. After the target system has been booted up the device driver for XLON® PC/104 adapter is loaded automatically by the „Device Manager" of Windows CE and can then be used for your application.

3.4.2.3 Registration Entry

Find an example for a correct registration entry in file „xlon_pc104.reg". The content is explained below.

[HKEY_LOCAL_MACHINE\Drivers\BuiltIn\XlonPC104x]
The registration key HKEY_LOCAL_MACHINE\Drivers\Builtin contains subkeys defining the device drivers that are loaded by the Device Manager after restarting the system. Each subcode holds the name of the device driver and contains the entries listed in the table below. The „x" in subcode „...\XlonPC104x" is identical to the entry „Index" in the table below.

.

Value name	**Value type**	**Description**
Dll	**REG_SZ**	This required entry specifies the file name for a driver DLL which is loaded by the Device Manager, e.g. xlon_pc104.dll
Prefix	**REG_SZ**	This required value specifies the driver's device file name prefix. It is a three character identifier, such as LON.
Index	**REG_DWORD**	This optional value specifies the device index, a value from 0 to 9, that the Device Manger should assign to the driver.
Order	**REG_DWORD**	This value specifies the load order for a driver. If two drivers have the same load order value, the drivers load in the order they occur in the registry. This entry is useful when one driver must load after another.
FlushCancel	**REG_DWORD**	This value specifies wether the device driver sends a niFLUSHCANCEL command down to the network interface after receiving a niRESET command from the network interface. A value equal to 0 enables this feature, a value other than 0 disables this feature.
TranceiverID	**REG_DWORD**	This value specifies the Transceiver ID of the network transceiver used on the XLON® PC/104 adapter as specified by Echelon. Refer to chapter 5.1.5 for more information about Transceiver ID's.
BaseAddress	**REG_DWORD**	This value specifies the base address on which the XLON® PC/104 adapter is located. Refer to chapter 3.4.2.4 for additional information.
IRQ	**REG_DWORD**	This value specifies the interrupt number which is configured for the XLON® PC/104 adapter. Refer to chapter 3.4.2.4 for additional information.
FriendlyName	**REG_SZ**	A optional string containing a human-readable description of name for the device driver.

3.4.2.4 Hardware Configuration

With the Windows CE 3.0 device driver you can use up to 7 XLON® PC/104 adapters in one system. If this should not be sufficient for you, please contact DH electronics. The special software design of the device driver enables interrupt sharing between any number of XLON® PC/104 adapters. This is very important because neither Windows CE nor PC/104 (ISA) bus supports „native" interrupt sharing.

Please refer to chapter 3.2 and 3.4 on how to find and configure available base addresses and interrupts. Often this can also be found in the manual enclosed by the manufacturer of the motherboard.

3.4.3 Initial Installation under Linux

The device driver for Linux is implemented in the form of a loadable kernel module. Currently the kernel versions 2.0, 2.2 und 2.4 are supported. The device driver supports up to 7 XLON® PC/104 adapters on different base addresses in one system. Interrupt sharing is not supported.

After making sure that the kernel module fits with the current kernel version, loading can be tried. Please notice that the kernel module can only be loaded with the rights of a Superuser. The device driver module can be added to the kernel by the below command. As additional parameters, the I/O base address and a interrupt number must be specified.

```
# /sbin/insmod pclta.o probe_io=<card_io> probe_irq=<card_irq>
```

You must specify a valid base address for <card_io> (e.g. 0x200) and a valid interrupt number for <card_irq> (e.g. 10). Please refer to chapter 3.2 and 3.4 on how to find and configure available base addresses and interrupts. Often this can also be found in the manual enclosed by the manufacturer of the motherboard.

If the module can't be loaded and you get message „Kernel version mismatch" and if the first two digits are identical with the kernel version (i.e. 2.2 or 2.4) loading can be enforced by the below command including the option "-f". The given warning has no effects on the function of the device driver.

```
# /sbin/insmod –f pclta.o probe_io=<card_io> probe_irq=<card_irq>
```

By the following command you can check if loading of the device driver module has been successful:

```
# /sbin/lsmod
```

After this order the messages on the screen should contain the below lines:

```
Module          Size        Used by
pclta.o         6960        0   (unused)
```

If the module pclta.o does not appear or if you want to check how many XLON® PC/104 adapters have been recognized you can switch to the console indicating the kernel messages. If this console is not available the kernel messages can also be found in the file /var/log/messages.
To remove the device driver module from the kernel please use the following command:

```
# sbin/rmmod pclta
```

To use the XLON® PC/104 adapter in your own application you must create a „Character Device File“. Please use the following commands:

```
# mknod /dev/pclta c 40 0
# chmod 666 /dev/pclta
```

In the first line a „Character Device File“ called „pclta“ is created. This name enables access to the device driver. The parameter „c“ defines that the device driver works unbuffered. It is followed by the major- and minor device numbers of the device driver which enables access to the device. In this case it is the device driver for the XLON® PC/104 adapter.

If you are intending to use more than one XLON® PC/104 adapter a „Character Device File“ has to be created for each physical LON channel. Please use the below series of commands (in this case two XLON® PC/104 adapters).

```
# mknod /dev/pcltaa c 40 0
# chmod 666 /dev/pcltaa
# mknod /dev/pcltab c 40 1
# chmod 666 /dev/pcltab
```

The device /dev/pcltaa appears as first LON channel, the device /dev/pcltab stands for the second LON channel. The example shows that the minor device number stand for the specific LON channel. This index starts with zero and is raised by one for each additional LON channel.

For the „Character Device File“ you can basically use any name. Under Liux applications the name „pclta“ is used as standard. It is also possible to create several „Character Device Files“ for one XLON® PC/104 adapter.

If you want to automize loading and unloading of the device driver the orders above can be incased in Shell-Scripts or added to the rc-Scripts.

3.4.4 Initial Installation under DOS

The XLON® PC/104 adapter network driver is installed by adding a 'DEVICE' command to the DOS 'Config.sys' file.

Use the following procedure to install the DOS driver:

1.) Copy the driver file 'Lonpc104.sys' from the driver disk/CD (subdirectory \DOS) into your root directory or any other appropriate directory, e.g. c:\lon\drivers.

2.) Add the following line into your Config.sys file:

```
device=<drive>:\<path>\lonpc104.sys /p200 /d1 /u10
or
devicehigh=<drive>:\<path>\lonpc104.sys /p200 /d1 /u10
```

drive: drive on which the driver is located. For example: C:
path: directory name in which the driver is located. For example: lon\drivers
The directory names have to be <= 8 characters!

Command line options:
/p<nnn>   --> used base adress
                       /p200 [default]; p300; p310; p320; p330; p340; p350;
Sets the device address to <nnn>. This address must match the XLON® PC/104 adapter address set in the DIP switch. See chapter 3.2 for information on setting these switches. The default device address used by the driver is 0x200.

/d<n>   --> Device unit number
                       /d1 -> LON1 [default]
                       /d2 -> LON2 ...
Defines the device unit number as <n>, where <n> is between 1 and 9, so that the DOS device name is LON1 through LON9. The default is 1 for LON1. The option is used to support multiple network interfaces on a single PC. Device numbers must be unique.

/u<n>   --> used interrupt
                       /u10   interrupt 10 [default]
                       /u5; /u10; /u11; /u15
Sets the XLON® PC/104 adapter interrupt request number (IRQ) to the value <n>. The default IRQ is 10. The specified IRQ must not be used by any other device installed in the PC.

/O<nn>   --> Sets the number of output (downlink) buffer within the driver.
                       /O10 -> 10 output buffer in the device driver
                       /O8 -> 8 output buffer in the device driver [default]
Sets the number of output (downlink) buffer within the driver to <nn>. The buffer sizes within the driver are pre-set to accomodate 255 byte packets.

/I<nn>   --> Sets the number of input (uplink) buffer within the driver.
                       /I10 -> 10 input buffer in the device driver
                       /I8 -> 8 input buffer in the device driver [default]
Sets the number of input (uplink) buffer within the driver to <nn>. The buffer sizes within the driver are pre-set to accomodate 255 byte packets.

/z   --> Disables an automatic flush cancel.
Disables automatic cancellation of the FLUSH state. If this flag is not specified, the driver automatically cancels the FLUSH state. If the flag is specified, the host application must cancel the FLUSH state.

3.) Save your modified Config.sys file.

4.) Reboot your system!

When the system boot up the following message appear:

```
*** Device Driver for the XLON PC/104 Board ***
                    Version 2.1
Copyright (C) DH electronics GmbH 1998, 2003. All rights reserved.
```

In this message you find the driver version of the installed DOS driver.

Now you can access the XLON PC/104 with under the switch /d given name (e.g. LON1).

⚠ This DOS driver works in the DOS Box of Windows9x too!
This DOS driver doesn't work under WindowsNT/2K/XP/ME!

3.4.5 Driver update

☞ Find the current drivers for download under:

www.xlon.de

- Download the driver corresponding to your operating system and store it at any place

3.4.5.1 Windows

- Unistall the device driver as described in chapter 3.5
- Install the new device driver as desribed in chapter 3.4

3.4.5.2 Windows CE 3.0

Driver update corresponds to initial installation as described in chapter 3.4.2.

3.4.5.3 Linux

Driver update corresponds to initial installation as described in chapter 3.4.3.

3.4.5.4 DOS

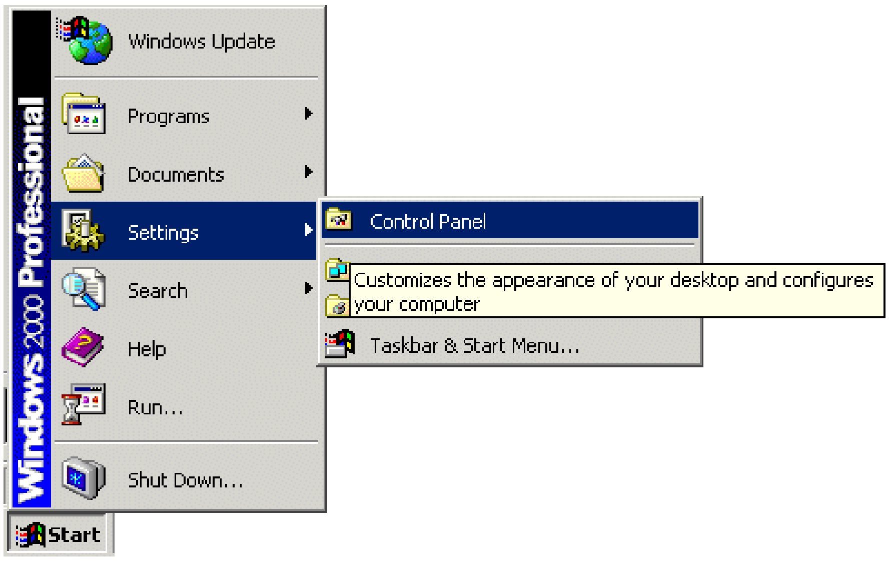
Driver update corresponds to initial installation as described in chapter 3.4.4.

3.5 Software De-installation

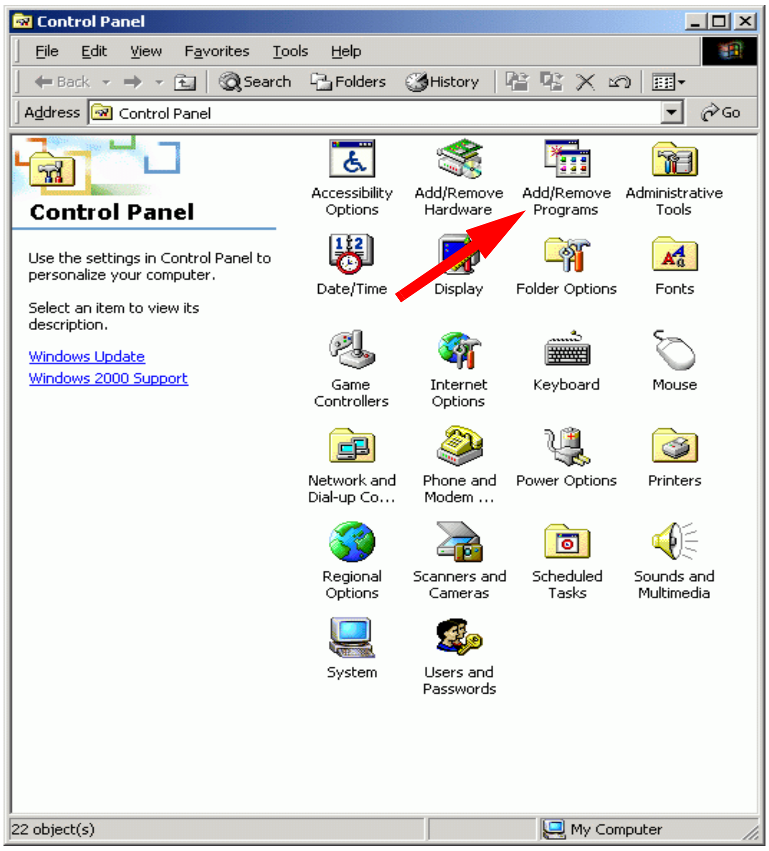
3.5.1 Windows

Initial De-installation under Windows 2000 is shown exemplary below. De-installation under other Windows operating systems works analogously!

- Open the Control Panel
  - Click on „Start“
  - Go to „Settings“ with the mousepointer
  - Click on „Control Panel“



Customizes the appearance of your desktop and configures your computer

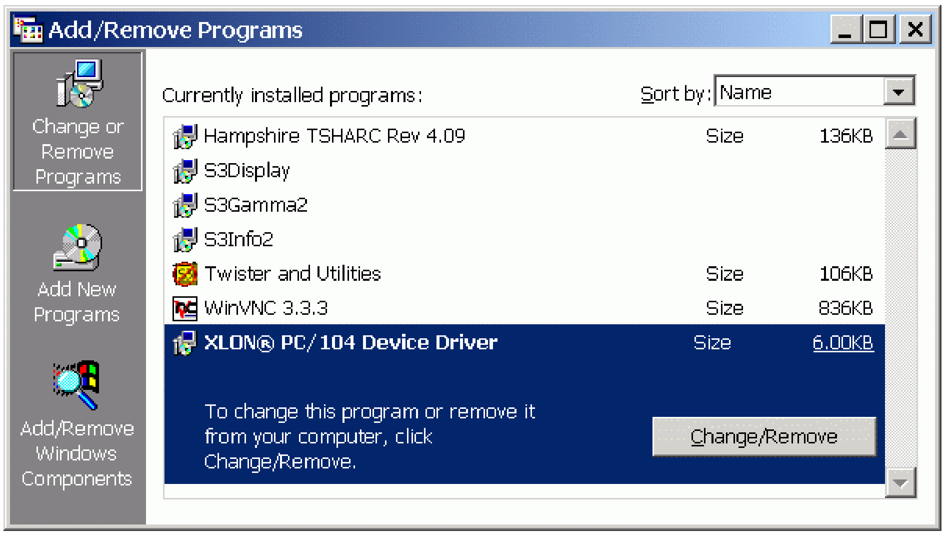


Control Panel

Use the settings in Control Panel to personalize your computer.

Select an item to view its description.

- Select „Add/Remove Programs“ by doubleclick

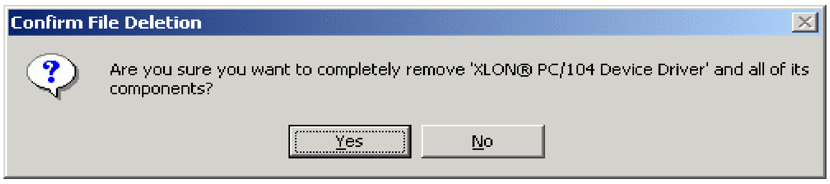


Currently installed programs:

Hampshire TSHARC Rev 4.09	Size	136KB
S3Display		
S3Gamma2		
S3Info2		
Twister and Utilities	Size	106KB
WinVNC 3.3.3	Size	836KB
XLON® PC/104 Device Driver	Size	6.00KB

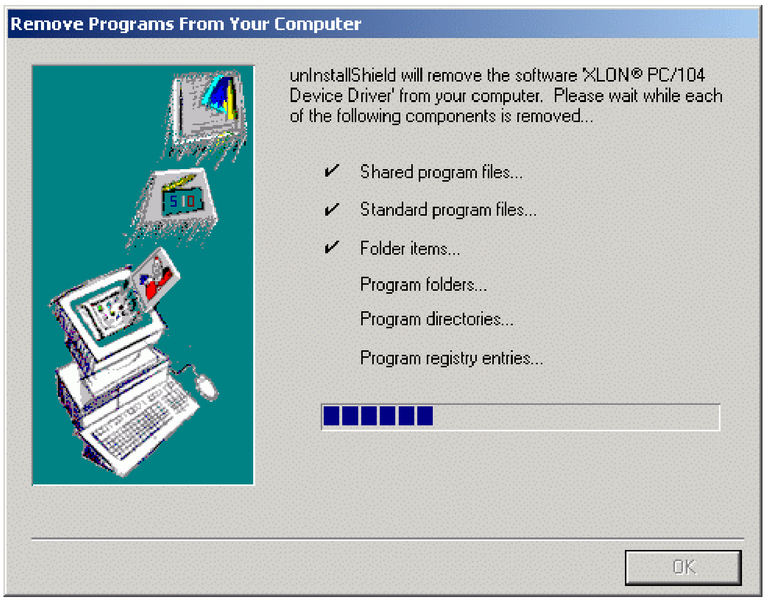
To change this program or remove it from your computer, click Change/Remove.

- Select „XLON PC/104 Device Driver“ via mouse click
- Click on „Change/Remove“ Button



Are you sure you want to completely remove 'XLON® PC/104 Device Driver' and all of its components?

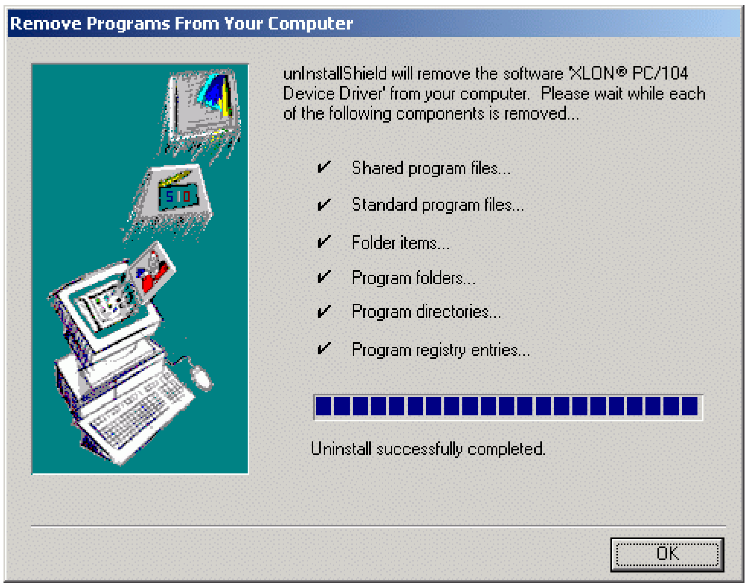
- If you are sure you want to de-install the device driver click on „YES“. You can cancel the de-installation with button „No“.



unInstallShield will remove the software 'XLON® PC/104 Device Driver' from your computer. Please wait while each of the following components is removed...

✔ Shared program files...
✔ Standard program files...
✔ Folder items...
Program folders...
Program directories...
Program registry entries...

The de-installation procced.



Remove Programs From Your Computer

unInstallShield will remove the software 'XLON® PC/104 Device Driver' from your computer. Please wait while each of the following components is removed...

- ✓ Shared program files...
- ✓ Standard program files...
- ✓ Folder items...
- ✓ Program folders...
- ✓ Program directories...
- ✓ Program registry entries...

Uninstall successfully completed.

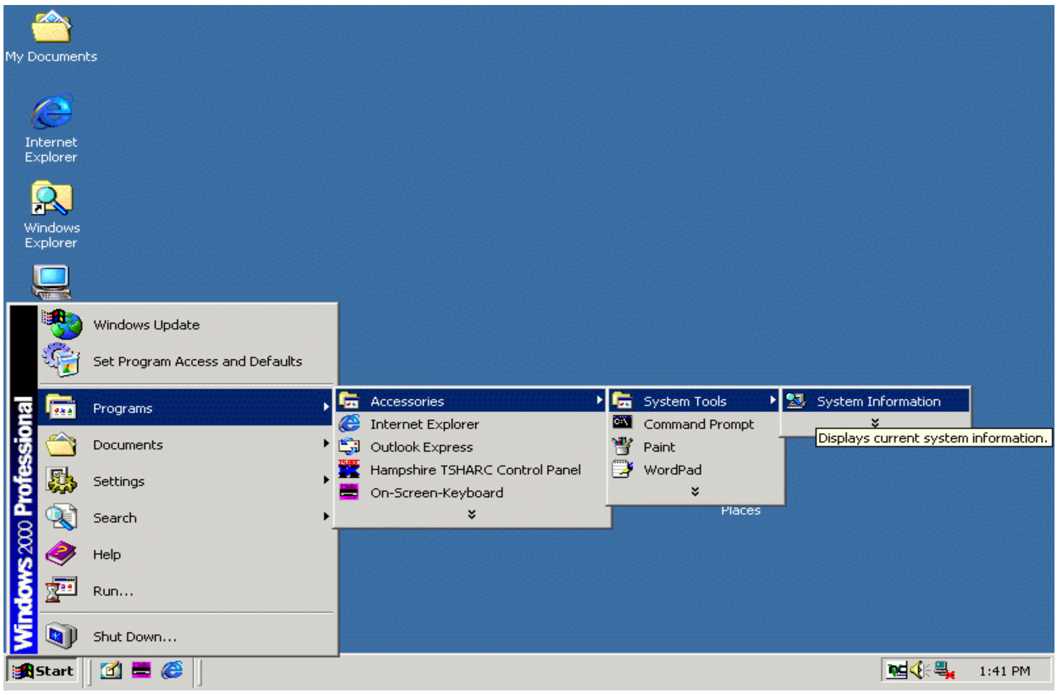
- Click on „OK“. The de-installation of driver software is finished.

4 Configuring and Testing

4.1 Check of settings under Windows

4.1.1 General settings

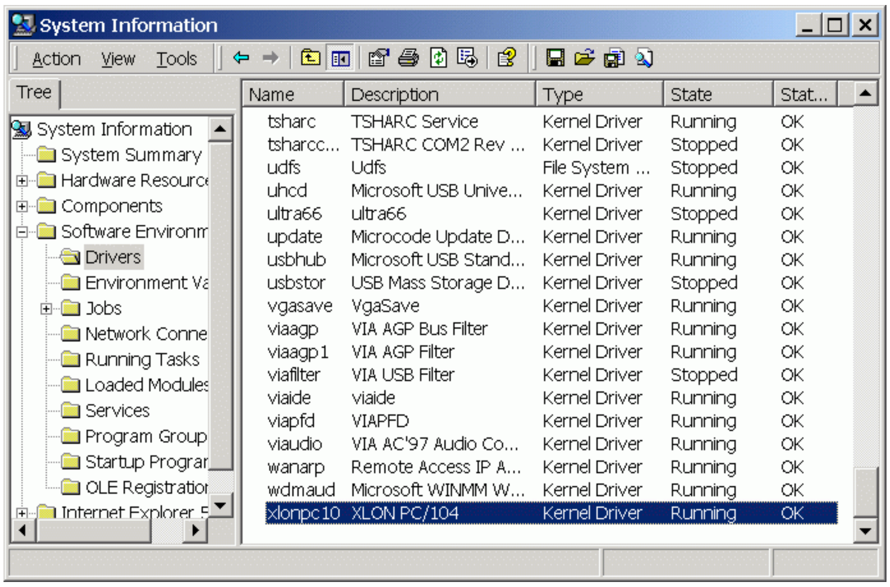
To get informations about the installed device driver use the „System Information" Tool:



gb/us

4.1.2 Driver information

- Select to the path 'Software Environment\Drivers' and scroll to Name „xlonpc10" as shown below.



Name	Description	Type	State	Stat...
tsharc	TSHARC Service	Kernel Driver	Running	OK
tsharcc...	TSHARC COM2 Rev ...	Kernel Driver	Stopped	OK
udfs	Udfs	File System ...	Stopped	OK
uhcd	Microsoft USB Unive...	Kernel Driver	Running	OK
ultra66	ultra66	Kernel Driver	Stopped	OK
update	Microcode Update D...	Kernel Driver	Running	OK
usbhub	Microsoft USB Stand...	Kernel Driver	Running	OK
usbstor	USB Mass Storage D...	Kernel Driver	Stopped	OK
vgasave	VgaSave	Kernel Driver	Running	OK
viaagp	VIA AGP Bus Filter	Kernel Driver	Running	OK
viaagp1	VIA AGP Filter	Kernel Driver	Running	OK
viafilter	VIA USB Filter	Kernel Driver	Stopped	OK
viaide	viaide	Kernel Driver	Running	OK
viapfd	VIAPFD	Kernel Driver	Running	OK
viaudio	VIA AC'97 Audio Co...	Kernel Driver	Running	OK
wanarp	Remote Access IP A...	Kernel Driver	Running	OK
wdmaud	Microsoft WINMM W...	Kernel Driver	Running	OK
xlonpc10	XLON PC/104	Kernel Driver	Running	OK

State of the current XLON® PC/104 adapter Windows driver is shown.

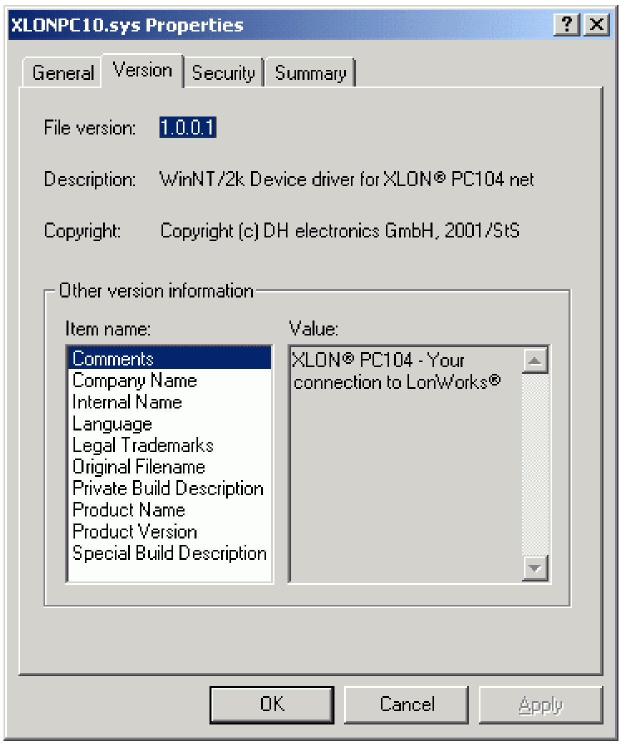
- To find out the version number of the device driver use the Windows 'Explorer' and go to the following directories:

Windows9x/ME:	<Windows-Drive>:\<Windows-Path>\system
	e.g.: c:\windows\system
WindowsNT/2K/XP:	<Windows-Drive>:\<Windows-Path>\system32\drivers
	e.g.: c:\winnt\system32\drivers

- Select the following files, press right mouse button and select menu „Properties":

Windows9x/ME:	'lonpc104.vxd'
WindowsNT/2K/XP:	'xlonpc10.sys'

- Click on tab „Version":



File version: 1.0.0.1

Description: WinNT/2k Device driver for XLON® PC104 net

Copyright: Copyright (c) DH electronics GmbH, 2001/StS

Other version information

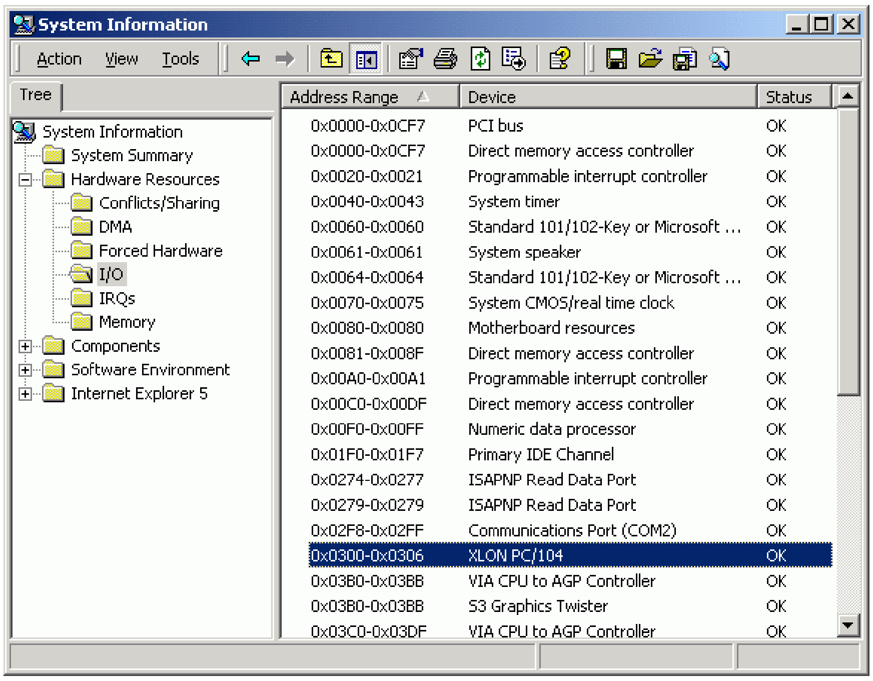
Item name:	Value:
Comments	XLON® PC104 - Your connection to LonWorks®
Company Name	
Internal Name	
Language	
Legal Trademarks	
Original Filename	
Private Build Description	
Product Name	
Product Version	
Special Build Description	

You get information about the File version, Company name (Provider) and Copyright.

☞ **For enquiries concerning support you should always hold ready this information!**

4.1.3 Occupied Hardware-Resources

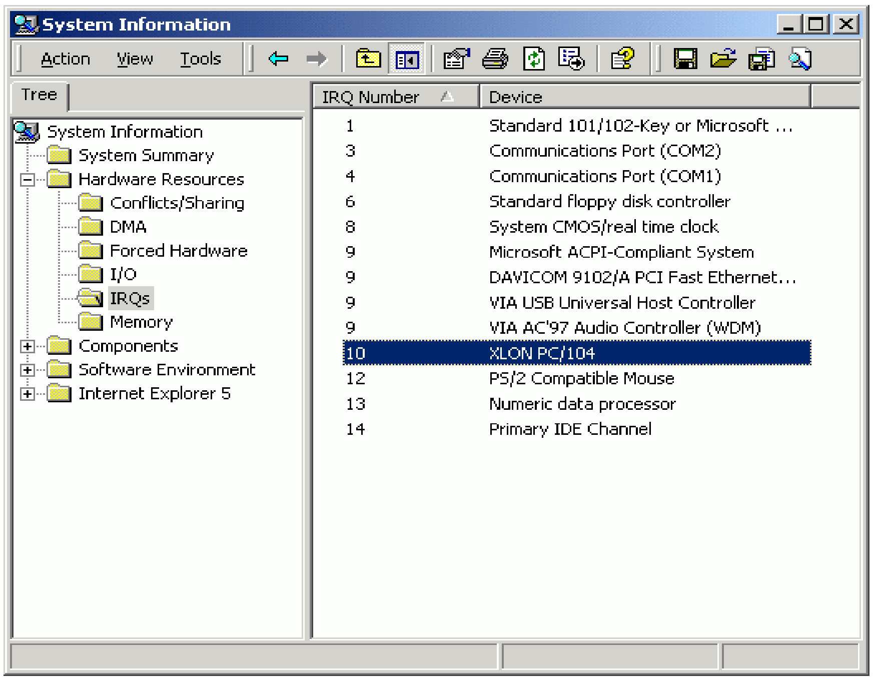
- Start the System Information Tool as shown in chapter 4.1.1
- Click on path „Hardware Resources\I/O“



Address Range	Device	Status
0x0000-0x0CF7	PCI bus	OK
0x0000-0x0CF7	Direct memory access controller	OK
0x0020-0x0021	Programmable interrupt controller	OK
0x0040-0x0043	System timer	OK
0x0060-0x0060	Standard 101/102-Key or Microsoft ...	OK
0x0061-0x0061	System speaker	OK
0x0064-0x0064	Standard 101/102-Key or Microsoft ...	OK
0x0070-0x0075	System CMOS/real time clock	OK
0x0080-0x0080	Motherboard resources	OK
0x0081-0x008F	Direct memory access controller	OK
0x00A0-0x00A1	Programmable interrupt controller	OK
0x00C0-0x00DF	Direct memory access controller	OK
0x00F0-0x00FF	Numeric data processor	OK
0x01F0-0x01F7	Primary IDE Channel	OK
0x0274-0x0277	ISAPNP Read Data Port	OK
0x0279-0x0279	ISAPNP Read Data Port	OK
0x02F8-0x02FF	Communications Port (COM2)	OK
0x0300-0x0306	XLON PC/104	OK
0x03B0-0x03BB	VIA CPU to AGP Controller	OK
0x03B0-0x03BB	S3 Graphics Twister	OK
0x03C0-0x03DF	VIA CPU to AGP Controller	OK

☞ In this window you find information about occupied IO address range.

- Click on path „Hardware Resources\IRQs“



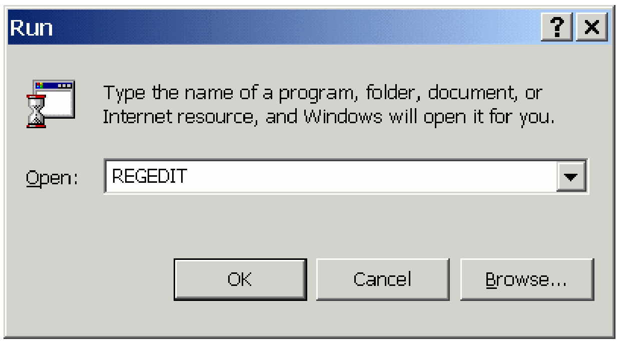
IRQ Number	Device
1	Standard 101/102-Key or Microsoft ...
3	Communications Port (COM2)
4	Communications Port (COM1)
6	Standard floppy disk controller
8	System CMOS/real time clock
9	Microsoft ACPI-Compliant System
9	DAVICOM 9102/A PCI Fast Ethernet...
9	VIA USB Universal Host Controller
9	VIA AC'97 Audio Controller (WDM)
10	XLON PC/104
12	PS/2 Compatible Mouse
13	Numeric data processor
14	Primary IDE Channel

☞ In this window you find information about occupied interrupt.

4.1.4 Properties of the XLON® PC/104 adapter

4.1.4.1 Configuration of the device driver

- Start 'Regedit.exe' program as shown below

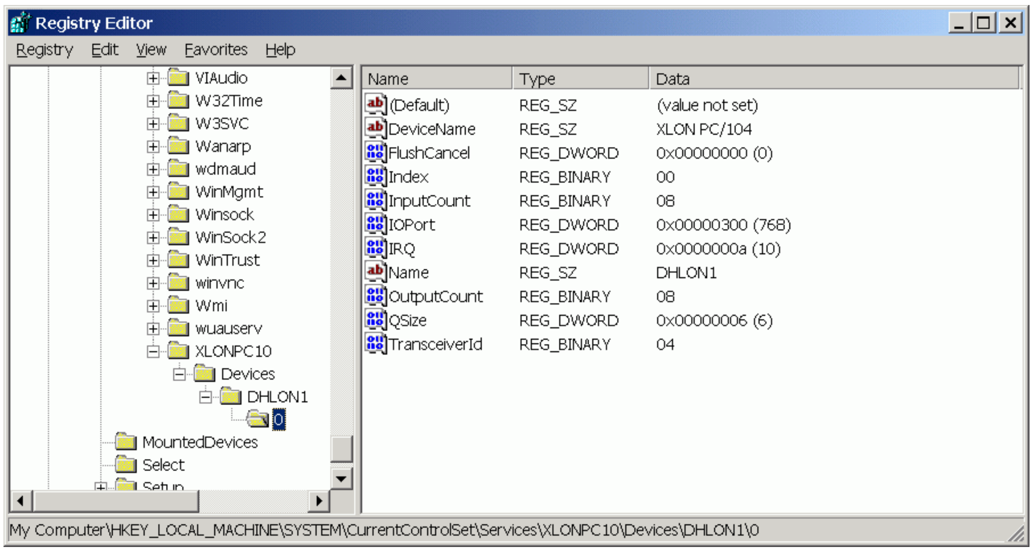


Type the name of a program, folder, document, or Internet resource, and Windows will open it for you.

Open: REGEDIT

- Navigate to the following registry key:

  Windows9x/ME:
  HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\Class\LonNIDH\0001\
  WindowsNT/2000/XP:
  HKEY_LOCAL_MACHINE\System\CurrentControlSet\Services\XLONPC10\Devices\DHLON1\0\

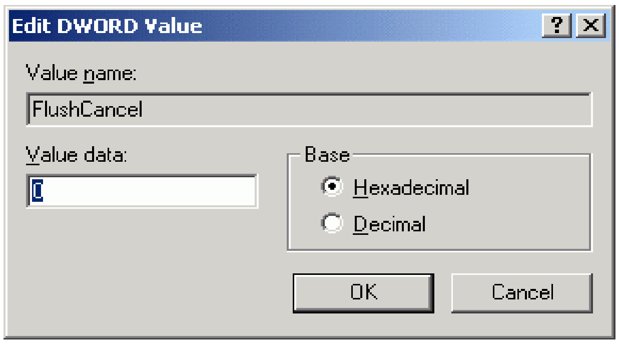


Name	Type	Data
(Default)	REG_SZ	(value not set)
DeviceName	REG_SZ	XLON PC/104
FlushCancel	REG_DWORD	0x00000000 (0)
Index	REG_BINARY	00
InputCount	REG_BINARY	08
IOPort	REG_DWORD	0x00000300 (768)
IRQ	REG_DWORD	0x0000000a (10)
Name	REG_SZ	DHLON1
OutputCount	REG_BINARY	08
QSize	REG_DWORD	0x00000006 (6)
TransceiverId	REG_BINARY	04

My Computer\HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\XLONPC10\Devices\DHLON1\0

Under this registry key you can modify the settings for 'IO Port', 'IRQ' (described in chapter 3.4.1.2), 'Flush Cancel', 'Transceiver ID', 'Input Count' and 'Output Count'.

- Select 'FlushCancel' by doubleclick



Value name: FlushCancel
Value data: 0
Base: [✓ Hexadecimal] Decimal

- Actvating/De-activating Automatic Flush Cancel

Possible values for 'Flush Cancel':

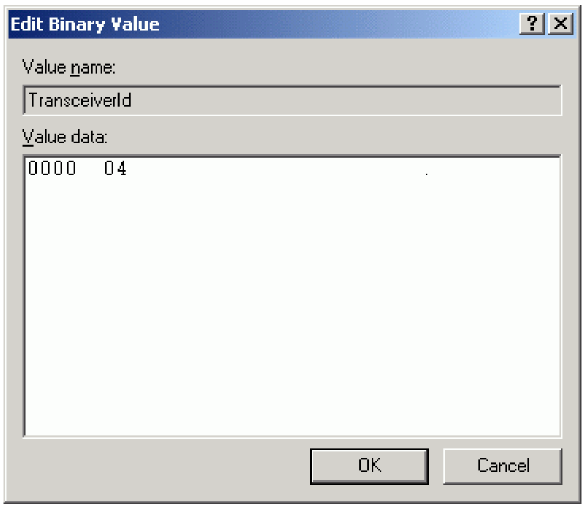
Automatic Flush Cancel activated:	0
Automatic Flush Cancel deactivated:	1

After each reset of the XLON® PC/104 adapter communication via the LonWorks® network is locked by standard. Receiving or sending of data is only possible after a Flush Cancel command has been sent to the XLON® PC/104 adapter.

If the Automatic Flush Cancel function is activated the device driver doesn't sends a Flush Cancel command to the XLON® PC/104 adapter after every reset. Then the application is responsible for sending a Flush Cancel command to the XLON® PC/104 adapter . After sending the Flush Cancel command, communication via the LonWorks® network is possible.

☞ **Please notice that if the Automatic Flush Cancel function is de-activated a Flush Cancel Command has to be sent from the application to the XLON® PC/104 adapter!**

- Select 'Transceiver ID' by doubleclick and configure the Transceiver ID



Value name: TransceiverId
Value data:

```
0000   04
```

Alteration of Transceiver ID

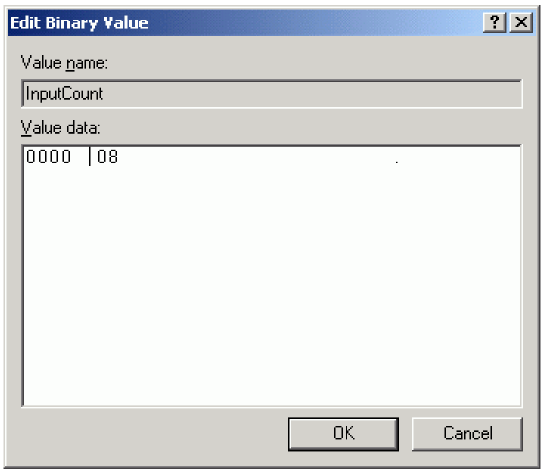
Make sure that possible transceiver settings match with the hardware.The XLON® PC/104 adapter will not work with incorrect settings!

Possible values for 'TransceiverID':

ID	NAME	Media	Network Bit Rate
04	FTT-10A	Free topology/link power	78 kbps
05	RS485-39	RS-485 twisted pair	39 kbps
10	RS485-625	RS-485 twisted pair	625 kbps
11	RS485-1250	RS-485 twisted pair	1250 kbps
12	RS485-78	RS-485 twisted pair	78 kbps
30	Custom	Custom	N/A

Use Transceiver ID '30' for custom specific network bit rate of RS485 transceiver versions. For more informations have a look at the Neuron processor data book.

- Select 'Input Count' or 'Output Count' by doubleclick and configure the Input or Output Buffer of the device driver



Value name: InputCount

Value data:

```
0000  08
```

The value range of the value for 'Input Count' and 'Output Count' is 1 < value < 256.

The following settings are made by state of delivery:

Transceiver ID:	04 (TP/FT-10 78kbps)
Automatic Flush Cancel:	activated
Input Count:	8
Output Count:	8

- After modifing this settings Reboot your computer.

4.2 Test of the XLON® PC/104 adapter under Windows

4.2.1 Diagnosis by Software

- Use an usual LonWorks Software product for testing the XLON® PC/104 adapter.

4.2.2 Diagnosis by LED

As described in chapter 3.2 the XLON® PC/104 adapter is fitted with three LED's for visualisation. A green LonWorks® network traffic LED, a yellow Service-Pin LED and a red State LED of the onboard logic.

Green LonWorks®-network-traffic LED:

By blinking (frequency is not defined) the green LED indicates whether there is data traffic on the LonWorks®-network.
☞ Initial state is not defined, this means that in the basic state the LED can be activated or de-activated. Moreover this LED can blink with the frequency of the yellow Service-Pin LED (1,25 Hz). This happens if no application has accessed the XLON® PC/104 adapter before.

Yellow Service-Pin LED:

The yellow LED indicates the state of the Neuron Processor Service Pin line.

- If the device driver is installed correctly and access to the XLON® PC/104 adapter is made from an application the LED is de-activated.
- If the LED blinks with a frequency oft ½ Hz the state of the XLON® PC/104 adapter is „Unconfigured".
- If a „Reset"-command from an application is executed the LED flashes for a moment

☞ If the LED blinks with a frequency of 1,25 Hz and an application has no access to the XLON® PC/104 adapter there is a problem with the driver installation.

If access from an application is not possible and the state of the LED doesn´t change probably a hardware error is the problem.

Yellow Service-Pin LED	**Describtion**
Constantly off	- Successful access from the application to the XLON® PC/104 adapter
Constantly on / Constantly off	- Hardware error
Flash frequency1,25 Hz	- No application has accessed the XLON® PC/104 adapter
Flash frequency ½ Hz	- State of the XLON® PC/104 adapter is „Unconfigured", this means the XLON® PC/104 adapter has no network address
Flashes for a moment	- A „Reset" to the XLON® PC/104 adapter has been made or - A device test has been made

Red Onboard-Logic-State-LED:

- When the LED is off the onboard logic device is loaded successful.
- When the LED is on the onboard logic is not loaded until yet, or a problem occured when loading the onboard logic.

When the power goes up on the XLON® PC/104 adapter, the state of this LED should switch from on to off.

5 Technical Specification

5.1 Hardware

5.1.1 General Information

Bus Interface		PC/104 conform, in accordance with P996 PC standard
Network Connection	FTT-10A: RS485:	2-conductor Weidmueller connector with tension clamp connection and screw flanges 3-conductor Weidmueller connector with tension clamp connection and screw flanges
Power Supply		Via the PC/104 bus
Service Pin Function		Controlled by host or external Service button
Configuration State		Displayed on host and via Service LED
Network Transceiver		FTT-10A or RS485 (integrated)
Network-Topologies		FTT-10A: Free Topology and Link Power RS485: Twisted Pair
Power Supply Data		5 V DC, ±5%, 200 mA typical
Operating temperature		0°C to +70°C (+32°F to +158°F) standard -40°C to +85°C (-40°F to +185°F) extended
Non-operating temperature		-45°C to +85°C (-49°F to +185°F)
Maximum humidity		90%@+50°C (90%@+122°F), non condensing
EMI		EN55022 Level B, EN61000-4-2, EN61000-4-4, EN50140, EN50141
Listings		CE
Processor		Neuron® 3150 - Chip@10 MHz
Dimensions		90 x 96 mm (3.6" x 3.8") (lenght x width)
Weight		55 g

The hardware of the XLON® PC/104 adapter supports up to 4 adapters per PC104 Bus System in the PC (Multiple Device Support).

5.1.2 Plug Connection

As plug connection for the LonWorks®-connection of the XLON® PC/104 adapter a Weidmueller plug connection (Series BL3.5) is used.

Type of network	**Pin #**	**Weidmueller Termination**	**Order number**
Free Topologie FTT	2 pin	BL3.5/2F SN OR	160664
Twisted Pair RS485	3 pin	BL3.5/3F SN OR	160665

The cross-section of the line which can be clamped at maximum is 1.5 mm^2.

Find order information as well as further detailled information about this connector under:

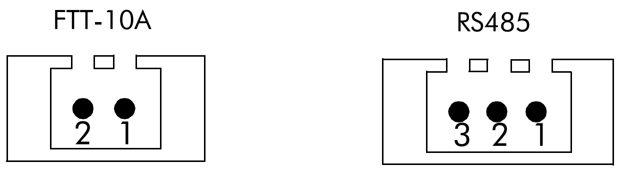
www.weidmueller.de

As plug connection for the extension-connection of the XLON® PC/104 adapter a socket (2.54 mm grid) is used.

There are many manufacturers of such connectors on the market.

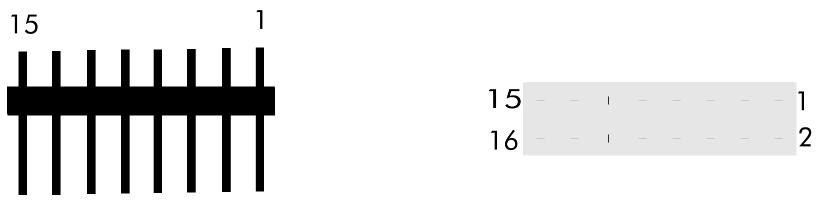
Plug Allocation:

Pin allocation of the LonWorks® connector



Pin	**FTT-10A**	**RS485**
1	NET A	RS485 A
2	NET B	RS485 B
3	not existing	GND

Pin allocation of the extension connector



Topview

Frontview

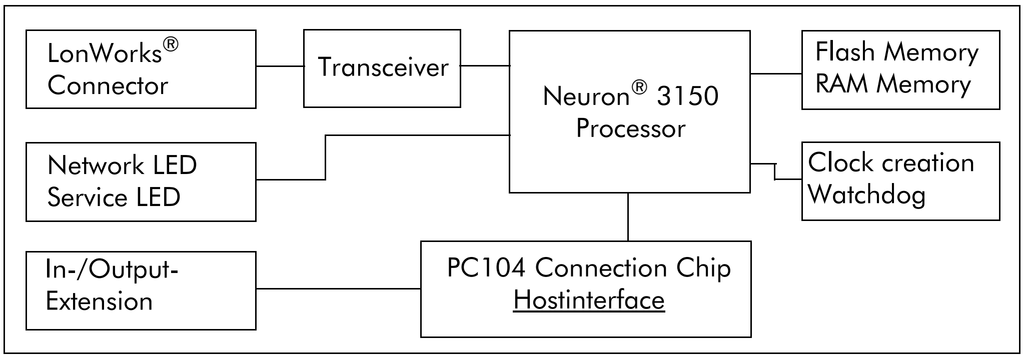
Pin	Describtion	Pin	Describtion
1	power supply VCC (5,0V DC)	9	programmable IN-/OUTPUT 8
2	programmable IN-/OUTPUT 1	10	power supply GND
3	programmable IN-/OUTPUT 2	11*	not connected
4	programmable IN-/OUTPUT 3	12*	not connected
5	programmable IN-/OUTPUT 4	13*	connection Service Pin Neuron
6	programmable IN-/OUTPUT 5	14*	connection LED 'Network'
7	programmable IN-/OUTPUT 6	15*	connection LED 'Done'
8	programmable IN-/OUTPUT 7	16*	not connected

*) available from hardware version -500

Pin 13 of the extension connector is the Service Pin of the Neuron® Processor. You can connect a Service Pin Button as described in the Neuron® Processor data book to this pin.

You can connect a LED via a series resistor of 470 Ohm on pin 14 and 15 of the extension connector. The 'Network' LED have to be connected to ground and the 'Done' LED to VCC.

5.1.3 Block model



5.1.4 Technical Details of the Hardware

Connection to the Host System via PC104 Bus:

Connection to the PC104 Bus follows PC standard P996. The width of data bus is 8 Bit.

LonWorks®-Network Interface:

There are two different Transceiver variants for the LonWorks®-Network at disposal: Free Topology Transceiver FTT-10A (2 pin connector) and RS485 Transceiver (3 pin connector). The transmission rate of the FTT-10A Transceiver is 78,5 kBit/s. If the XLON® PC/104 adapter is fitted with RS485 Transceiver different transmission rates can be set by software (compare chapter 4.1.4). The maximum datarate is 250kBit/s.

Neuron Processor Core:

A 3150® Neuron Processor with external storage interface is used. A one time programable PROM memory is used as program memory. A SRAM memory is used as data memory. The Neuron Processor is connected to the PC104-bus by the Neuron ‚Parallel IO Model'.

Address Register of the Neuron Processor:

Type of memory	Address Zone	Memory Size
ROM-memory	0x0000 - 0xC2FF	49919 Byte (48.75 kB)
RAM-memory, read and write	0xC300 - 0xE6FF	9215 Byte (9.00 kB)
IO range for Interrupt-Generation	0xE700 - 0xE7FF	
Reserved Neuron® Processor intern	0xE800 - 0xFFFF	

Digital Input-/Output Extension:

The Input-/Output-Extension contains eight free configurable digital TTL IO's. You find informations how to program this IO's in chapter 6.4.

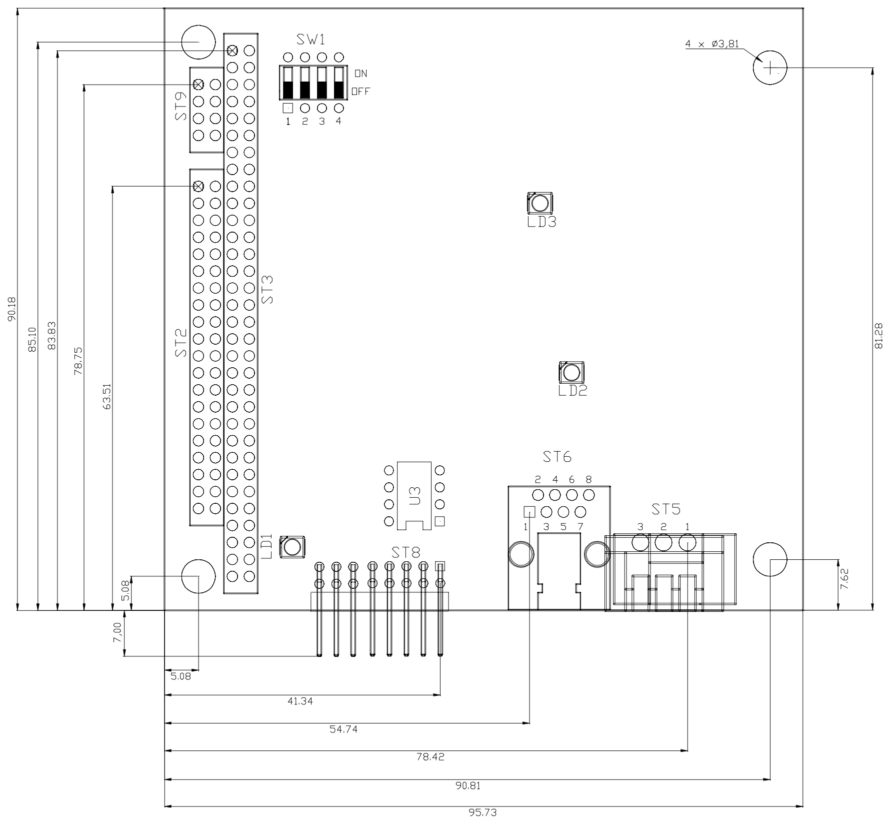
Describtion	Min [V]	Max [V]
High-level input voltage	2.0	5.0
Low-level input voltage	0	0.8
High-level output voltage @ High-level output current = -4.0 mA	2.4	
Low-level output voltage Low-level output current (sinking) = 12.0 mA	0	0.4

5.1.5 Supported Transceivers

Generally the transceiver configuration doesn´t have to be changed. If required it can be changed in the Registry with the Regedit program under Microsoft Windows Desktop operating systems. For operating systems like Microsoft Windows CE, Linux or DOS manual setting of the transceiver ID into certain configuration files might be necessary. Find more information about this in chapter „Driver Installation" of the corresponding operating system. The register below shows the LON Transceiver IDs which are generally supported by the XLON® PC/104 adapter. Depending on the transceiver(TP/FT-10 oder TP-RS485) which is physically existing on the adapter not all of the transceiver operating types are supported.

ID	Name	Media	Network Bit Rate
04	TP/FT-10	Free topology/link power	78 kbps
05	TP-RS485-39	RS-485 twisted pair	39 kbps
12	TP-RS485-78	RS-485 twisted pair	78 kbps

5.1.6 Mechanical Drawing



6 Software Access

6.1 Application Interface under Windows

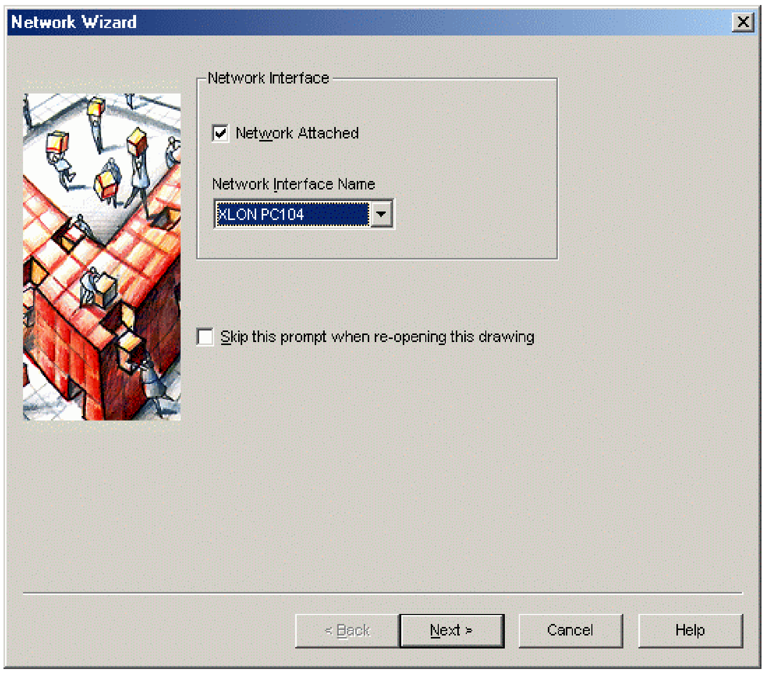
6.1.1 LNS-applications

If you want to access the XLON® PC/104 adapter via a LNS application as network interface you only have to select the XLON® PC/104 adapter you are intending to use.

The LNS-application shows a „Select" menu for the Network Interface Name. Find the XLON® PC/104 adapter under the Network Interface Name:

XLON PC104

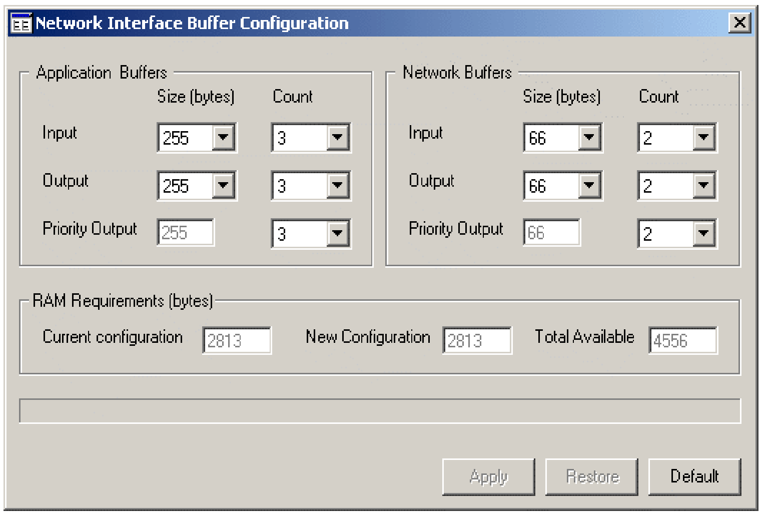
The Window Screen below shows configuration in the application ‚LonMaker for Windows':



Network Attached: ✓
Network Interface Name: XLON PC104

6.1.2 Configuration of the Network Interface Buffer

The Network- and Applicaton Buffer of the Neuron Processor can be changed by the Echelon LNS Plug-In ‚Network Interface Buffer Configuration' (www.echelon.com).


Network Interface Buffer Configuration

Application Buffers

	Size (bytes)	Count
Input	255	3
Output	255	3
Priority Output	255	3

Network Buffers

	Size (bytes)	Count
Input	66	2
Output	66	2
Priority Output	66	2

RAM Requirements (bytes)

Current configuration 2813 New Configuration 2813 Total Available 4556

6.1.3 Programming your own application

Based on the information of the ‚LonWorks Host Application Programmer's Guide' it is possible to program your own LonWorks Host application. The programming instruction can be obtained from Echelon (www.echelon.com) .

The process of coding access to the device driver of the XLON® PC/104 adapter is explained in this chapter. As the Echelon Standard Driver Interface of systems based on Windows 9x/ ME and Windows NT/2k/XP is different, you have to distinguish between the two types of operating systems.

All functions listed below are Windows 32-bit API-functions.

Find a more detailed example for download under the following internet site:

http://www.xlon.de

6.1.3.1 Opening the device driver

Before the driver can be accessed it has to be opened. If the device driver has been opened successfully it delivers a handle which enables access to the driver.

The name of the device driver for the XLON® PC/104 adapter under Windows9x/ME operating system is: ‚\\.\lonpc104.0'

The name of the device driver for the XLON® PC/104 adapter under WindowsNT/2k/XP operating system is: ‚\\.\dhlon10'

This name can also be read from the Registry by an Alias. If the Network Interface which must be used is intended to be for selection it is advisable to turn access selectable from the Registry by an Alias Name.
For this device indicated by registration code:

<HKEY_LOCAL_MACHINE\SOFTWARE\LonWorks\DeviceDrivers\>

can be offered for selection.

Find the XLON® PC/104 adapter under the Alias Name: XLON PC104

Necessary Command and Type Definitions:

Commands under Windows9x/ME:

```
#define LDV_Acquire                  1      // Sign access to the driver
#define LDV_Release                  2      // Release access to the driver
#define LDV_Register_Event_Handle    7      // Register an event handle
#define LDV_Read                     10     // Read from the driver
#define LDV_Write                    11     // Write to the driver
```

Type Definition of the application buffer:

```
#define MAXLONMSG                    253    // Maximum length of message data
typedef struct APILNI_Message_Struct
{
        BYTE NiCmd;                         // Network Interface Command
        BYTE Length;                        // Size of ExpAppBuffer
        BYTE ExpAppBuffer[MAXLONMSG];       // Buffer for Data
} APILNI_Message;
```

Structure of the application buffer (API LNI Message)

	Command	2 Byte
	Length	
Length	Begin of Data	3 Byte
	Network Adress	11 Byte
	Data	variable Length

Definition of a Access Handle

```
1.)     HANDLE* phandle = new HANDLE;       // Handle to access the device driver
```

Definition of the Application Buffers

```
2.)     APILNI_Message* ni_in_msg = new APILNI_Message;     // NSI Message In structure
        APILNI_Message* ni_out_msg = new APILNI_Message;    // NSI Message Out structure
```

Windows9x/ME:

The function ‚CreateFile' opens the device driver of the XLON® PC/104 adapter and passes back a handle for access to the device driver. ‚\\.\lonpc104.0' has to be used as name for the device driver.

2.) *pHandle = CreateFile("\\\\.\\lonpc104.0", GENERIC_READ | GENERIC_WRITE, 0, 0, OPEN_EXISTING, 0, 0);

Possible error codes passed back by the driver:

error:	Not enough memory for allocation of driver buffer
error code:	ERROR_NOT_ENOUGH_MEMORY

Under Windows9x/ME operating systems the command 'LDV_Aquire' has to be obeyed after opening the driver. Hereby the device driver is informed that the driver has been accessed.

3.) DeviceIoControl(*pHandle, MAKELONG(LDV_Acquire, 0), &inBuf, sizeof(char), &RetInfo, sizeof(RetInfo), &nBytesReturned, NULL);

Possible error codes passed back by the driver:

error:	none

WindowsNT/2000/XP:

The function ‚CreateFile' opens the device driver for the XLON® PC/104 adapter and passes back a handle for access to the device driver. ‚\\.\dhlon10' has to be used as name for the device driver.

2.) *pHandle = CreateFile("\\\\.\\dhlon10", GENERIC_READ | GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE, (LPSECURITY_ATTRIBUTES) NULL, OPEN_EXISTING, 0, (HANDLE) NULL);

Possible error codes passed back by the driver:

error:	Not enough memory for allocation of the driver bufferr
error code:	ERROR_NOT_ENOUGH_MEMORY

6.1.3.2 Registration of an Event-Handle

For communication between driver and application a common Event-Handle can be registered. Therefore the application has to demand a Handle from the operating system. The application subsequently delivers the handle to the driver. When the driver has got data for the host application it creates an event.

Windows9x/ME:

Creation of an Synchronisation Event Handle:
1.) CreateCommonEvent(&hEventR3, &hEventR0, FALSE, FALSE);

Delivery of the Event Handle to the Driver:
2.) DeviceIoControl(pHandle, MAKELONG(LDV_Register_Event_Handle,0), hEventR0, sizeof(HANDLE), &RetInfo, sizeof(RetInfo), &nBytesReturned, NULL));

WindowsNT/2000/XP:

Hereby another Event mechanism is in action. Access is made via Overlapped IO.

6.1.3.3 Reading of data from the device driver

In order to read data the command ‚LDV_Read' in the Windows API function ‚DeviceIoControl' is used under Windows98/ME.

Under WindowsNT/2000/XP the Windows API function ‚ReadFile' is used.

Windows9x/ME:

```
DeviceIoControl(pHandle, MAKELONG(LDV_Read, 0), NULL, sizeof(char), ni_in_Msg, length,
&nBytesReturned, NULL) ;
```

Possible error codes passed back by the driver:

error:	No LDV_ACQUIRE was executed before the driver has been accessed.
error code:	ERROR_ACCESS_DENIED

WindowsNT/2000/XP:

```
ReadFile( pHandle, ni_in_Msg, length+1, &nBytesReturned, NULL );
```

Possible error codes passed back by the driver:

error:	Overlapped IO and no data available.
errorcode:	STATUS_PENDING
error:	Non Overlapped IO and no data available.
errorcode:	STATUS_UNSUCCESSFUL

6.1.3.4 Writing of data on the device drivers

In order to write data the command ‚LDV_Write' in the Windows API function ‚DeviceIoControl' is used under Windows9x/ME. Under WindowsNT/2000/XP the Windows API function ‚WriteFile' is used.

Windows9x/ME:

```
DeviceIoControl(pHandle, MAKELONG(LDV_Write, 0) , ni_out_Msg, length, NULL, sizeof(char),
&nBytesReturned, NULL);
```

Possible error codes passed back by the driver:

error:	No LDV_ACQUIRE was executed before the driver has been accessed.
error code:	ERROR_ACCESS_DENIED
error:	No free application buffer available in the driver
error code:	ERROR_NOT_ENOUGH_MEMORY
error:	Data record sent before application has been too big
error code:	ERROR_ACCESS_DENIED

WindowsNT/2000/XP:

```
WriteFile( pHandle, ni_out_Msg, length, &nBytesReturned, NULL );
```

Possible error codes passed back by the driver:

error	Application Buffer exhausted for writing
error code:	ERROR_NOT_ENOUGH_MEMORY

6.1.3.5 Closing the device driver

If you intend to end the application the driver has to be closed.

Windows9x/ME:

The command ‚LDV_Release' has to be obeyed before the driver can finally be closed. Hereby the occupancy of the device driver is cancelled.

```
1.)    DeviceIoControl( pHandle, MAKELONG(LDV_Release, 0), &inBuf, sizeof(char), &RetInfo, sizeof(RetInfo),
       &nBytesReturned, NULL);
```

The driver has to be closed subsequently.

```
2.)    CloseHandle(pHandle);
```

WindowsNT/2000/XP:

```
2.)    CloseHandle(pHandle);
```

6.1.3.6 Important Programing Information

If you program your own application for the XLON® PC/104 adapter a Program ID has to be programed into the adapter during the process of initialisation. The Program ID which has to be used can be defined by the application.

The network and application buffers of the Neuron® Processor can be changed. The maximum valid amount of bytes for all buffers used must not be higher than 4556 bytes. Admissible buffer settings should be located by the LNS Plug-In ‚Network Interface Buffer Configuration' (compare chapter 6.1.2.)
In some cases the XLON® PC/104 adapter can lose its function because of false buffer settings. This can only be canceled by a 'Reboot' of the adapter or in some cases it is not possible to revoke.

If the RS485 variant requires a special Transmission Rate which can not be set as Registry parameter (compare chapter 4.1.4) the Transceiver ID ‚Custom Transceiver' must be parameterized in the Registry. Please notice that then the application is responsible for correct settings of the Transceiver and the Transmission Rate.

6.2 Application interface under Windows CE 3.0

For creating a C/C++ LON Host Application under Windows CE 3.0 basically the documentation under chapter 6.1.3 can be used. The Application Programming Interface (API) for access to device driver under Desktop Windows and Windows CE 3.0 however differs.

For access from an own C/C++ LON Host Application to the device driver under Windows CE 3.0 the following Standard-Operating-System-Commands (Windows CE 3.0 API) for „Stream Interface Devices“ are available. The following subsections offer a survey of different API-functions. For a more detailed description look up the Windows CE 3.0 documentation which can be found in the MSDN Library or in the Microsoft Windows CE Platform Builder 3.0 Library.

CreateFile()	ReadFile()
CloseHandle()	DeviceIoControl()
WriteFile()	GetLastError()

6.2.1 CreateFile()

Prototype:

```
HANDLE CreateFile(    LPCTSTR lpFileName,
                      DWORD dwDesiredAccess,
                      DWORD dwShareMode,
                      LPSECURITY_ATTRIBUTES lpSecurityAttributes,
                      DWORD dwCreationDisposition,
                      DWORD dwFlagsAndAttributes,
                      HANDLE hTemplateFile );
```

Description:

This function opens the device driver which is specified by „lpFileName“. The name of device driver is composed of the device prefix and the device index followed by a doublepoint, for example "LON1: ", "LON2: ", "LON3: ", and so on. Chapter 3.3.2.1 delivers detailed information about the composition of the device driver name . Find more information about further function parameters and about the function Create File() in general in the Windows CE 3.0 documentation.

Example:

```
HANDLE myHandle;

myHandle = CreateFile(  TEXT("LON1:"),
                        GENERIC_READ | GENERIC_WRITE,
                        0x00, NULL
                        OPEN_EXISTING,
                        FILE_ATTRIBUTE_NORMAL,
                        NULL )
```

Return value:

If the device driver has been opened successfully a Handle is passed back to the device driver. By this Handle further operations can be made on the device driver or the device driver can be closed again. In case of an error INVALID_HANDLE_VALUE will be passed back. More detailed information about error cause can be obtained by calling the function GetLastError().

Error cause:

Possible error causes for a failure of this function are:
A device driver which has not been loaded, a false name of the device driver, a device driver which has not been closed orderly beforehand or function parameters which have not been set correctly.

6.2.2 CloseHandle()

Prototype:

BOOL **CloseHandle**(HANDLE hObject);

Description:

This function closes the device driver which is specified by the Handle „hObject" . As function parameter the handle passed back after successfully calling the function Create File() has to be passed to the device driver. Find more detailed information about the function CloseHandle() in the CE 3.0 documentation.

Example:

BOOL bResult;

bResult = **CloseHandle**(myHandle);

Return value:

If the device driver has been closed successfully the value „TRUE" is passed back otherwise the value „FALSE" is passed back. In this case detailed information about error cause can be obtained by calling the function LastError(). After a successful call of CloseHandle() the handle that has been passed on is not valid any longer and can no longer be used for operations on the device driver.

Error causes:

An invalid Handle on the device driver might be the error cause of a failure of this function.

6.2.3 WriteFile()

Prototype:

BOOL **WriteFile**(HANDLE *hFile*,
LPCVOID *lpBuffer*,
DWORD *nNumberOfBytesToWrite*,
LPDWORD *lpNumberOfBytesWritten*,
LPOVERLAPPED *lpOverlapped*);

Description:

By this function data from a LON application is written on the device driver and consequently on the XLON®-Network Interface. Calls made by this function are asynchronous. This means that the function returns as soon as data has been taken into the internal buffer of the device driver or an error has occured. Processing of data is then running in parallel to the

LON application in the background.

Each XLON®-Network Interface is specified by its „hFile". The cursor „lpBuffer" must point to a data structure of the type APILNI_Message which is also noted as Application Layer Buffer. The size of this data structure is variable. The structure is shown in the point below. For each specific call the size of this variable data structure is set by the parameter „nNumberOfBytesToWrite". By the parameter „lpNumberOfBytesWritten" the actual amount of bytes which has been transmitted to the Network Interface is passed back. In case of success these two values are identical and not zero. The parameter „lpOverlapped" has no usage under Windows CE 3.0. Find detailed information about the different function parameters and about function WriteFile() in general in the Windows CE 3.0 documentation.

Application Layer Buffer:

The below C-Code defines the data type „APILNI_Message" for Application Layer Buffer as specified in the Echelon NSI Firmware User's Guide. Find details on the structure of the Structure Element „ExpAppBuffer[]" in the „LonWorks Host Application Programmer's Guide".

```
#define MAXLONMSG  253

typedef struct APILNI_Message_Struct {
        BYTE NiCmd;                         // NSI command
        BYTE Length;                        // Size of  ExpAppBuffer
        BYTE ExpAppBuffer[MAXLONMSG];       // Message data
} APILNI_Message;
```

The graphics below shows again the structure of the Application Layer Buffer

command	2 Bytes	Application Layer Header
length		
message header	3 Bytes	ExpAppBuffer[MAXLONMSG], maximum size 253 Bytes
network address	11 Bytes	
message data	variable length	

Example:

```
BOOL bResult;
DWORD dwBytesWritten;
APILNI_Message lni_msg = { niRESET, 0x00 }        // Buffer with Reset Command to NSI

bResult = WriteFile(        myHandle,
                            (LPCVOID)&lni_msg,
                            0x02,
                            &dwBytesWritten,
                            NULL );
```

Return value:

If the writing operation on the device driver was successful the value „TRUE" is passed back otherwise the value „FALSE" is passed back. In this case detailed information about error cause can be obtained by calling the function GetLastError().

Error causes:

Possible error causes for a failure of this function:
An invalid Handle on the device driver, an Application Layer Buffer which has not been built

up correctly or the parameter „nNumberOfBytesToWrite" which is located outside the valid zone or does not conform with the actual length of the Application Layer Buffer.

6.2.4 ReadFile()

Prototype:

```
BOOL ReadFile(      HANDLE hFile,
                    LPVOID lpBuffer,
                    DWORD nNumberOfBytesToRead,
                    LPDWORD lpNumberOfBytesRead,
                    LPOVERLAPPED lpOverlapped );
```

Description:

By this function data is read from the device driver and consequently from the XLON®-Network Interface by a LON application. Calls made by this function are asynchronous. This means that the function returns as soon as data has been taken from the internal buffer of the device driver. If no data is available or if there has been an error the function also returns immediately. This means that data from the XLON®-Network Interface is not being waited for.

The XLON®-Network Interface is specified by its Handle „hFile". The cursor „lpBuffer" must point to a data structure of the type APILNI_Message which is also noted as Application Layer Buffer. The size of this data structure is variable. During a reading operation it should however be set at maximum value as the actual amount of data which has to be read is not known by time of calling the function.The structure is shown in the point below. For each specific call the size of this variable data structure is set by the parameter „nNumberOfBytesToRead". By the parameter „lpNumberOfBytesRead" the actual amount of bytes which has been transmitted to the Network Interface is passed back. In case of success this value equals „nNumber ofBytesToRead" or is lower than that and differs from zero. The parameter „lpOverlapped" has no usage under Windows CE 3.0. Find detailed information about the different function parameters and about function ReadFile() in general in the Windows CE 3.0 documentation.

Application Layer Buffer:

The below C-Code defines the data type „APILNI_Message" for Application Layer Buffer as specified in the Echelon NSI Firmware User's Guide. Find details on the structure of the Structure Element „ExpAppBuffer[]" in the „LonWorks Host Application Programmer's Guide".

```
#define MAXLONMSG  253

typedef struct APILNI_Message_Struct {
        BYTE NiCmd;                         // NSI command
        BYTE Length;                        // Size of  ExpAppBuffer
        BYTE ExpAppBuffer[MAXLONMSG];       // Message data
} APILNI_Message;
```

The graphics below shows again the structure of the Application Layer Buffer.

command	2 Bytes	Application Layer Header
length		
message header	3 Bytes	ExpAppBuffer[MAXLONMSG], maximum size 253 Bytes
network address	11 Bytes	
message data	variable length	

Example:

```
BOOL bResult;
DWORD dwBytesRead;
APILNI_Message lni_msg                 // Application Layer Buffer for message from NSI

bResult = ReadFile(          myHandle,
                             (LPCVOID)&lni_msg,
                             255,
                             &dwBytesRead,
                             NULL );
```

Return value:

If the reading operation on the device driver has been successful the value „TRUE“ is passed back. If the amount of read bytes passed back in „lpNumberOfBytesRead“ equals zero no data had been available. If the reading operation on the device driver has not been successful the value „FALSE“ is passed back. In this case detailed information about error causes can be obtained by calling the function GetLastError().

Error causes:

Possible error causes for a failure of this function:
An invalid Handle on the device driver or a parameter „nNumberOfBytesToRead“ which is located outside the valid zone or does not conform with the actual length of the Application Layer Buffer.

6.2.5 DeviceIo Control()

Prototype:

BOOL **DeviceIoControl**(HANDLE *hFile*,
DWORD *dwIoControlCode*
LPVOID *lpInBuffer*,
DWORD *nInBufferSize*,
LPVOID *lpOutBuffer*,
DWORD *nOutBufferSize*,
LPDWORD *lpBytesReturned*,
LPOVERLAPPED *lpOverlapped*);

Description:

This function enables certain operations on the device driver which are not practicable by the functions listed above. At present these operations are „GetVersion“ und „ReadWait“ which can be called by the commands „IOCTL_XLON_GETVERSION“ or „IOCTL_XLON_READWAIT“.
The operation „GetVersion“ enables read out of a version from the device driver. The operation „ReadWait“ is the synchronous (blocking) variant of the function ReadFile(). These two functions which can be operated by DeviceIoControl() are explained below.

6.2.5.1 „GetVersion“ by DeviceIoControl()

Description:

The operation „GetVersion“ is realized by the API-Function DeviceIoControl() and enables read out of a version code from a device driver. The IO-Control-Code for this operation is defined as „IOCTL_XLON_GETVERSION“. The driver version is coded by a DWORD. Each of the four bytes represents a decimal digit. The Major-Version is coded in Bit 16 to Bit 23 (Byte 2) and the Minor-Version in Bit 8 to Bit 15 (Byte 1) the remaining Bits (Byte 0 and Byte 3) have no meaning at the moment. A read out DWORD from 0x00010200 has the value 0.1.2.0 this means that the driver version is 1.2.

Calling DeviceIoControl() the XLON®-Network Interface has to be specified by its Handle „hFile“. The parameter „dwIoControlCode“ has to be occupied by the command „IOCTL_XLON_GETVERSION“. The parameters „lpInBuffer“ or „nInBufferSize“ are not needed. For the parameter „lpOutBuffer“ a pointer is passed on to a DWORD where the driver version is put later. In „nOutBufferSize“ the size of the DWORD is passed into Byte. In the parameter „lpBytesReturned“ the amount of Bytes which has been read is passed back. The parameter „lpOverlapped“ is not used under CE 3.0. Find detailed information about the specific function parameters and about the function DeviceIoControl() in general in the Windows CE 3.0 documentation.

Example:

```
#define IOCTL_XLON_GETVERSION  (DWORD)0x01 // IOCTL-Code for command
                                                   „GetVersion“
BOOL bResult;
DWORD dwVersion, dwBytesReturned;

bResult = DeviceIoControl(      myHandle,
                                IOCTL_XLON_GETVERSION,
                                NULL, 0,
                                &dwVersion,
                                sizeof( dwVersion ),
                                &dwBytesReturned
                                NULL );
```

Return value:

If read out of the driver version from the device driver has been successful the value „TRUE“ is passed back otherwise the value „FALSE“ is passed back. In this case detailed information about the error cause will be obtained by calling the function GetLastError().

Error causes:

A possible error cause for a failure of this function is an invalid Handle on the device driver.

6.2.5.2 „ReadWait" by DeviceIoControl()

Description:

The operation „ReadWait" is the synchronous (blocking) variant of the function ReadFile(). By this operation a LON Application reads data from the device driver and consequently from XLON®-Network Interface. If there is no data in the internal buffer of the device driver data is waited for within a free defined period until the call returns (Blocking Call). If the waiting time is defined as „Zero" the function is identical to the call of the function ReadFile(). If the waiting time is defined as „INFINITE" there is no Timeout while waiting.

Calling DeviceIoControl() the XLON®-Network Interface has to be specified by its Handle „hFile". The parameter „dwIoControlCode" has to be occupied by the command „IOCTL_XLON_READWAIT". The waiting time for the operation „ReadWait" is defined in the parameter „lpInBuffer" this has to be a pointer to a DWORD. The value of this DWORD specifies waiting time in milliseconds . If the value is „INFINITE" the process of waiting lasts until data has arrived from the XLON®-Network Interface or until the driver has been closed. The parameter „nInBufferSize" defines the length of the preceding DWORD including waiting period. The pointer „lpOutBuffer" has to indicate a data structure of the type APILNI_Message which is also noted as Application Layer Buffer. The size of this data structure is variable during a reading operation it should however be put at maximum as the actual amount of data which has to be read is not known by the time of calling the function. The structure is shown below. For each specific call the valid size of this variable data structure is defined by the parameter "nOutBufferSize". The actual amount of Bytes read from the Network Interface is passed back by the parameter „lpBytesReturned". In case of success this value lays below or equals „nOutBufferSize" and is not zero. The parameter „lpOverlapped" is not used under Windows CE 3.0. Find detailed information about the different functions and about the function ReadFile() in general in the Windows CE 3.0 documentation.

Application Layer Buffer:

The below C-Code defines the data type „APILNI_Message" for Application Layer Buffer as specified in the Echelon NSI Firmware User's Guide. Find details on the structure of the Structure Element „ExpAppBuffer[]" in the „LonWorks Host Application Programmer's Guide".

```
#define MAXLONMSG  253

typedef struct APILNI_Message_Struct {
        BYTE NiCmd;                          // NSI command
        BYTE Length;                         // Size of  ExpAppBuffer
        BYTE ExpAppBuffer[MAXLONMSG];        // Message data
} APILNI_Message;
```

The graphics below shows again the structure of the Application Layer Buffer

command	2 Bytes	Application Layer Header
length		
message header	3 Bytes	ExpAppBuffer[MAXLONMSG], Maximum size. 253 Bytes
network address	11 Bytes	
message data	variable length	

Example:

```
#define IOCTL_XLON_READWAIT  (DWORD)0x00 // IOCTL-Code for Command
                                                „ReadWait“
BOOL bResult;
DWORD dwTimeout = INFINITE;
DWORD dwBytesReturned;
APILNI_Message Ini_msg              // Application Layer Buffer for message from NSI

bResult = DeviceIoControl(      myHandle,
                                IOCTL_XLON_READWAIT,
                                &dwTimeout,
                                sizeof( dwTimeout ),
                                (LPVOID)&Ini_msg,
                                255,
                                &dwBytesReturned
                                NULL );
```

Return value:

If the reading operation on the device driver has been successful the value „TRUE“ is passed back. If the amount of read bytes passed back in „lpBytesReturned“ equals zero no data had been available even after the waiting period has run down. If the reading operation on the device driver has not been successful the value „FALSE“ is passed back. In this case detailed information about error causes can be obtained by calling the function GetLastError()..

Error cause:

Possible error causes for a failure of this function:
An invalid Handle on the device driver, or a parameter „nOutBuffersize“ which is located outside the valid zone or does not conform with the actual length of the Application Layer Buffer.

6.2.6 GetLastError()

Prototype:

DWORD **GetLastError**(<u>void</u>);

Description:

This function gives detailed information about error cause if there is a failure in the functions CreateFile(), CloseHandle(), ReadFile(), WriteFile() or DeviceIoControl(). Find detailed information about the function GetLastError() in the Windows CE 3.0 documentation.

Example:

```
DWORD dwLastError;

dwLastError = GetLastError();
```

Return value:

Error code of the last operation.The below error codes are defined for the XLON device driver under Windows CE 3.0:

```
typedef enum
{
        LONDEV_SUCCESS = 0,             // No error detected
        LONDEV_NOT_FOUND,               // Device not found
        LONDEV_ALREADY_OPEN,            // Device already open
        LONDEV_NOT_OPEN,                // Not a handle to a open device
        LONDEV_DEVICE_ERR,              // Device detect error
        LONDEV_INVALID_DEVICE_ID,       // Invalid device id was detected
        LONDEV_NO_MSG_AVAIL,            // No message available
        LONDEV_NO_BUFF_AVAIL,           // Buffer is full
        LONDEV_NO_RESOURCES,            // No more resources available
        LONDEV_INVALID_BUF_LEN,         // Invalid buffer length
        LONDEV_DEVICE_BUSY              // Device is busy
} LonDevCode;
```

6.3 Application Interface under Linux

For the creation of a LON Host application under Linux you can basically refer to the documentation under 5.2.1. For access to the device driver from a own LON Host application under Linux commands of the operating system correspond to the commands used for other file operations under Linux. Find detailed information in the specific Linux documentation.

open()	Open driver of a specific device
close()	Close driver
read()	Read an „Uplink“ packet from the driver
write()	Write a „Downlink“ packet to the device

The structure of the data packet used for data exchange between LON Host application and device driver is explained in the „LonWorks Host Application Programmer's Guide“ as already mentioned in chapter 6.1.3. If there are no data packets available the read()-function blocks. This doesn´t happen if the device driver has been opened with the „O_NONBLOCK“ Flag. In this case an error is passed back with the value „EWOULDBLOCK“. The write()-function works in the same way.

6.4 Accessing the digital In- and Outputs

There are four 8 bit registers for accessing and configuring the eight digital IO's are 8. They are located at base address (as configured, see chapter 3.2) +4 ,+6 (data) and +5, +7 (configuration).

Register describtion of digital In- and Outputs:

address	**describtion**	**IO location**				
		Bit0	**Bit1**	**Bit2**	**Bit3**	**Bit4..7**
BASE+4	Dataregister 1	IO0	IO1	IO2	IO3	XXXX
BASE+5	Configuration for Dataretgister1*)	IO0	IO1	IO2	IO3	XXXX
BASE+6	Dataregister 2	IO4	IO5	IO6	IO7	XXXX
BASE+7	Configuration for Dataretgister2*)	IO4	IO5	IO6	IO7	XXXX

*) Configuration for Dataregisters: 0 = INPUT; 1= OUTPUT

Example code 1:
Base address 0x200; IO0 is used as Service Pin input

```
        unsigned char io_register_1 = 0x00;
...
        //Direction of IO0..IO3 INPUT, do this in the initialization of your software
        outport(0x205, 0x00);
        // Read IO register 1, poll this IO for the Service Pin button
        io_register_1 = inport(0x204);
        // If Service Pin Button is pressed
        if (io_register_1 & 0x01)
        // Send a Service Pin Message
                ni_send_immediate(niSERVICE);
```

Example code 2:
Base address 0x300; IO5 is used as input

```
        unsigned char io_register_2 = 0x00;
...
        //Direction of IO4..IO7 OUTPUT, do this in the initialization of your software
        outport(0x307, 0xFF);
        // Switch IO5 on
        io_register_2 = io_register_2 | 0x02;
        // Write to the IO register
        outport(0x307, io_register_2);
        // Switch IO5 off
        io_register_2 = io_register_2 & 0xFD;
        // Write to the IO register
        outport(0x307, io_register_2);
```

7 Appendix

8 Revision History

Version	Date	Describtion	State	By	Comments
1.0	11/06/03	Initial version	released	StS	