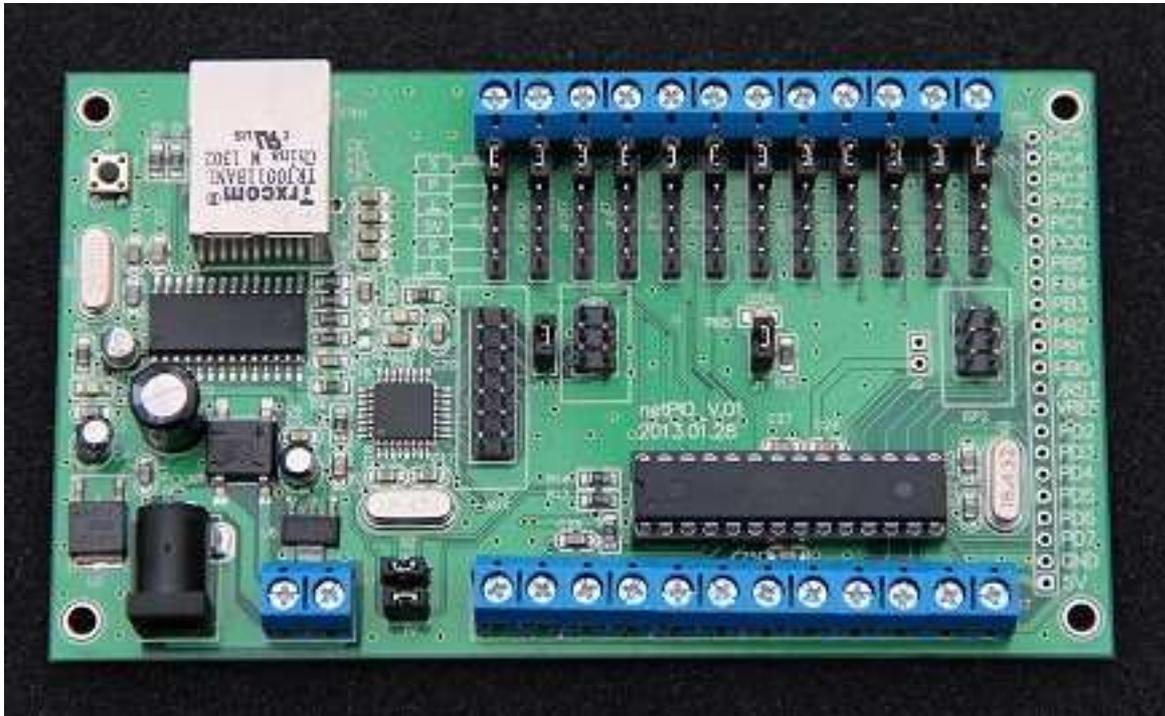


# ABACOM - netPIO



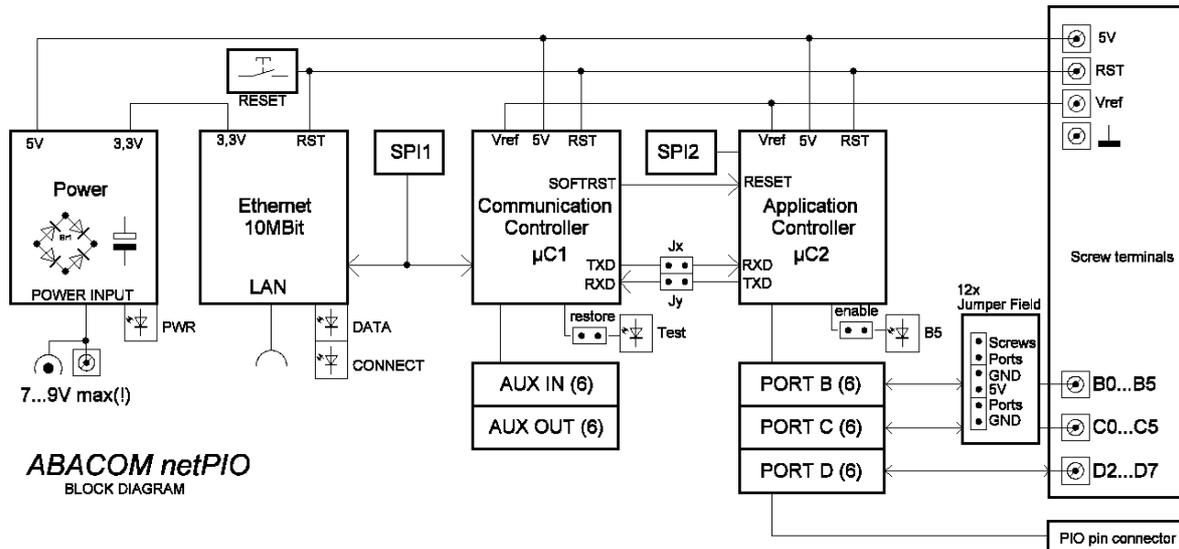
## Download

[http://www.abacom-online.de/div/setup\\_netPIO.exe](http://www.abacom-online.de/div/setup_netPIO.exe)

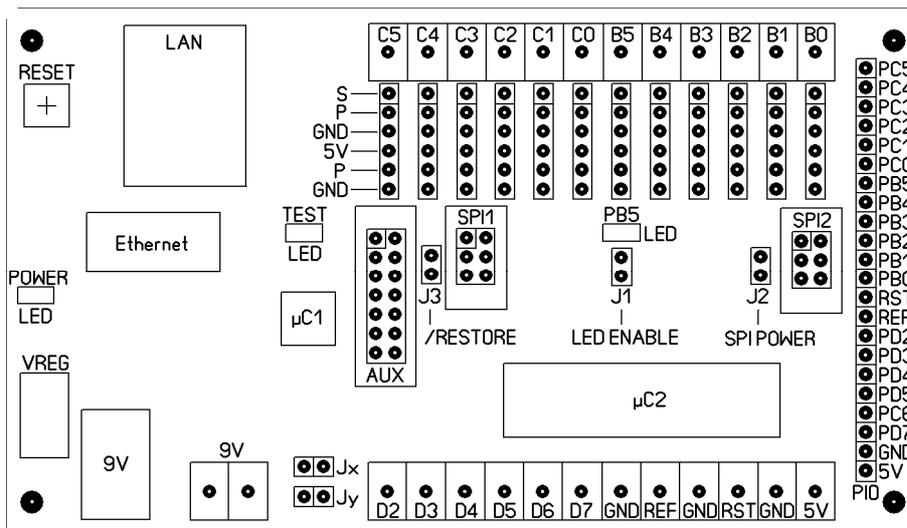
The ABACOM netPIO board is a 10Mbit network interface designed for measurement and control applications. The board is available with different firmware options which are also used with our USB- $\mu$ PIO modules. This document describes differences and additional features of the network based board version only. Information how to use the screw terminal port connections can be found in the document related to your firmware.

**Please read this document at least once completely before you start.**

## Block diagram



## Component placement



Dimensions 70mm x 120 mm (fits into WAGO DIN rail PCB carrier WAGO 288-600)

### **Voltage supply (Power)**

The module must be supplied by an external power adapter (7-9V), which can be connected to the DC jack or to the 9V screw terminal connector. The power adapter must be able to supply at least 500mA. The board uses approx. 200mA (without external circuits). A 5V voltage regulator is on board, as well as a protecting bridge rectifier.

A stabilized 9V power adapter is recommended. The supply voltage must not exceed 9V, to avoid excessive heat or damage. External circuits must not consume more than 200mA overall and I/O-port must not drive more than 10mA each.

Green power LED will light up as soon as the board is supplied with power. The voltage regulator (VREG) may become warm, which is normal, but it should never become hot.

### **Double controller - a powerful team**

The netPIO was designed with two microcontrollers on board, which we call the communication controller ( $\mu$ C1) and the application controller ( $\mu$ C2).

### **Ethernet connection and communication controller**

The LAN connector connects the device to your router. The green LED built into the connector indicates a physically correct connection. The yellow connector LED indicates incoming and outgoing network data.

The Ethernet interface is based on a MicroChip ENC28J70 chip that is controlled via SPI (SPI1) from an ATMEGA168-20 microcontroller ( $\mu$ C1 - Communication Controller).

A Windows configuration tool is used to configure network parameters, which is the same for all ABACOM netPIO products. The communication controller receives, sends and processes network data. The main part of communication is based on UDP network protocol.

The yellow TEST-LED is useful to identify devices and to debug network parameters configured with the Windows configuration tool. The communication controller offers an additional digital I/O port (AUX port), with six input and six output lines. A detailed explanation of the AUX port will follow later in this document.

## **Application controller**

The second microcontroller - the so-called application controller - offer three I/O ports (B, C and D), each with six I/O lines. These 18 I/O lines do the measurement and control jobs the device was designed for.

The application controller is connected to the communication controller, using the chips UART serial interface with a baud rate of 57600. The board internal serial connection is made available with jumpers Jx and Jy.

The application controller is again a ATMEGA168-20 (DIL socket version). Both controllers are clocked with a crystal frequency of 18.432 MHz.

Having two controllers on board makes it possible to split two heavy job, where one is the network communication and the other is the control application. Only relevant data is put through to the application controller. While the network job stays in the communication controller the application controller is free to control your application. As mentioned before the application controller is flashed with exactly the same firmware, which is used on our ABACOM- $\mu$ PIO boards. Refer to the document related to your firmware for details:

/ **GPIO18** ([http://www.abacom-online.de/div/FirmwareDocs/UK/Firmware\\_GPIO18.pdf](http://www.abacom-online.de/div/FirmwareDocs/UK/Firmware_GPIO18.pdf))

### **I/O interface with 18 general purpose digital I/O lines**

Use the board as general purpose digital IO. Compatible with ProfiLab Expert or make a software of your own.

/ **PWMIO18** ([http://www.abacom-online.de/div/FirmwareDocs/UK/Firmware\\_PWMIO18.pdf](http://www.abacom-online.de/div/FirmwareDocs/UK/Firmware_PWMIO18.pdf))

### **Comfortable interface with 18 general purpose digital I/O lines**

Digital I/O board with additional functions, like PWM, servo, blink / pulse.

Compatible with ProfiLab Expert or make a software of your own.

/ **FREQ** ([http://www.abacom-online.de/div/FirmwareDocs/UK/Firmware\\_FREQ.pdf](http://www.abacom-online.de/div/FirmwareDocs/UK/Firmware_FREQ.pdf))

### **9 MHz frequency counter / generator**

A powerful firmware with application software included.

Compatible with ProfiLab Expert

/ **INCR3** ([http://www.abacom-online.de/div/FirmwareDocs/UK/Firmware\\_INCR3.pdf](http://www.abacom-online.de/div/FirmwareDocs/UK/Firmware_INCR3.pdf))

### **Three incremental encode inputs**

Reads the position of up to three increment encoders.

Compatible with ProfiLab Expert.

/ **TEMP12** ([http://www.abacom-online.de/div/FirmwareDocs/UK/Firmware\\_TEMP12.pdf](http://www.abacom-online.de/div/FirmwareDocs/UK/Firmware_TEMP12.pdf))

### **Temperature system for Dallas DS18B20 sensors**

For use with up to 12x Dallas DS18B20 sensors.

The SPI2 interface of the application controller is used to flash the firmware during out production process (as well as SPI1 of the communication controller). There is not any usage of these interfaces the end user can take advantage of.

## I/O ports

These I/O lines are available for netPIO applications:

- Port B: Lines B0...B5
- Port C: Lines C0...C5
- Port D: Lines D2...D7

The line direction and purpose depends on the firmware option you chose to purchase with your netPIO. All I/O lines use TTL levels, which means that voltages at these line must never go above the 5V supply voltage and it must never go below ground level GND (negative).

All port I/O lines are made available on the PIO strip connector, allowing to solder and connect strip connectors, wires or other components as desired.

Port D lines are connected directly to the Port D screw terminals. For each line of port B and port C the board is equipped with a jumper field, simplifying external connections in many situations.

The jumper field pin-out is as follows (starting from the screw terminal):

- 1.) **S = Screw terminal** is connected to the screw terminal directly and only.
- 2.) **P - Port** leads to the corresponding port I/O line of the application controllers directly.
- 3.) **GND (Ground)** is ground level.
- 4.) **+5V** is connected to the board internal 5V power line.
- 5.) **P - Port** is the corresponding port line again.
- 6.) **GND - (Ground)** is ground once again.

There are many situations in which you may take advantage of the jumper field, but remember the limits please:

- max. 10 mA for each port
- max. external load 200mA over all

External circuits that consume more power must be supplied with its own power supply!

Some applications for the jumper field we can think of:

Example 1: Jumper set from S to P

S and P are connected, which is the factory setting. This connects the screw terminal directly to the port line.

Example 2: Jumper set from P to GND

A input(!) line is connected to ground to avoid noise from unused channels.

Example 3: Jumper set from +5V to P

A input(!) line gets a defined high level.

Example 4: A pluggable sensor with pin-out sequence 5V-Data-GND finds its place at pins 4, 5 and 6.

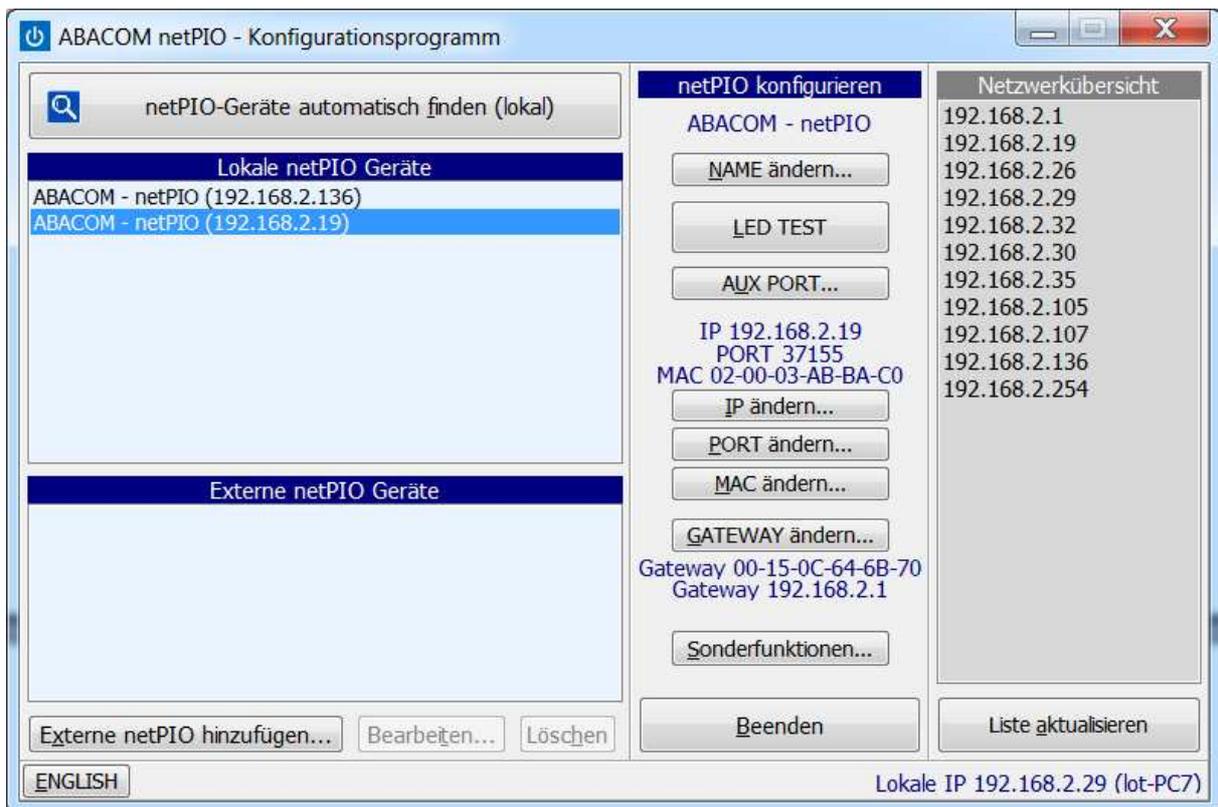
Example 5: A self-made input circuit (maybe an amplifier, input protection, low-pass filter, etc.) gets inserted between pin 1 (screw) and pin 2 (port).

Example 6: A sandwich PCB needs to be mounted and connected.

## Network configuration

Connect the netPIO to the your router and power up the device. The router must support 10Mbit connections. Green POWER-LED and green CONNECTOR-LED should light up. Yellow DATA-LED will indicate network traffic at the network connector, if any.

Adding new, not configured devices to your network, please do this one by one. Do not proceed with the next new device, before having configuration finished successfully for the actual one.



The network configuration tool (netPIO\_config.exe) is the same for any ABACOM netPIO product. It must be executed on a Windows-PC system which is connected to the same local network as the new netPIO device.

The configuration software finds and administrates all local netPIO devices. The software scans for devices when the software is started and each time the "find devices" button is pressed.

New devices with factory settings will be added to your network with a unused IP address. The new device will now appear in the local netPIO device list. It uses the default UDP port number 37155.

The configuration tool was designed to...

- change network parameters ( IP, Port, Gateway, MAC address )
- assign friendly-names to devices
- poll and display the AUX lines status
- setup master-slave configurations (traps)
- execute additional function

The configuration tool is able to change most settings of external device as well. Such devices may be located somewhere on the Internet, but must have been prepared (port forwarded) to be available from the Internet. IP, Port and MAC can not be change on external devices, but must be modified locally.

If you wish to change the port number or IP address for a device, select the device from the local list and press the corresponding CHANGE... button. The button TEST-LED toggles the yellow TEST-LED on the selected board. This may be helpful location a device and testing the network setup.

netPIO device have fixed IP addresses and do not use DHCP. A list of present IP devices can be found on the right, which may help you to find unused IP addresses. As soon as you finished the network setup for a device successfully, you can proceed with the next new device, if any.

The device name is just a helpful friendly-name and has not any influence on the device functions. You may prefer names like "Office #1", "Garage", "Heating" or similar. Usually it is not necessary to make changes of MAC address and port number, except that network conflicts appear in seldom cases. The gateway address is the local IP address of your router (192.168.1.1 for example), which the device uses for Internet access. You can change it if necessary.

All configuration settings are stored in the devices EEPROM memory. If a device can not be found any longer for some reasons, you should try to press the RESET button on the device. This will not change settings previously stored.

If still in trouble without any other reason, a bad configuration may have been be stored in the device EEPROM. In this case you can restore the original factory settings, and erase all individual settings you did before. To restore the factory settings do the following...

- remove jumper J3 (/Restore; located between SPI1 and AUX port)
- Press the RESET button (device must be powered during that)
- re-apply the jumper J3 to its former position
- press RESET button again

The device has now restored its factory settings, so you can assume it to be a new device and reconfigure it.

#### List of external devices

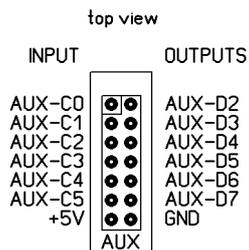
External devices can be added manually to this list, entering their hostnames and ports. To make devices available on the Internet, these devices must be published by forwarding their ports in their routers setup. Devices must be configured locally before you can forward them. You can not change IP and port of external devices remotely. (They would become unreachable if you did.)

## AUX-Port

The AUX port found on netPIO devices offers six additional digital input lines and six additional digital output lines. The port pins directly lead to the communication controller, so please be most careful using it. The communication controller is soldered to the board and can not be exchanged in case of damage. +5V and GND can be found on the AUX port as well.

Remember - All I/O lines use TTL levels, which means that voltages at these line must never go above the 5V supply voltage and it must never go below ground level GND (negative).

Do not confuse AUX channels with I/O lines of the application controller. D2 is NOT the same as AUX-D2 for example.



The AUX port pin-out from the component side,  
AUX-C inputs on the left - AUX-D outputs on the right.

AUX-C inputs that are left open have internal pull-up resistors activated, so these are high by default and can be pulled low by switches or open-collector-output circuits. The status of the AUX-D output lines is switchable using the configuration software. The status of all AUX lines can be polled and monitored using the configuration software.

So far an on first sight the AUX port offers some additional I/O lines only, but it is much more powerful, which will be described in the next chapter.

### **Signal triggered messages (traps)**

The network controller detect status changes at all AUX signal lines and on the TEST-LED. These changes can be configured to trigger messages, that are sent out to other netPIO devices (or to itself). The receiving device can change the output status of its AUX lines on such a notification message.

The device that sends the message is called master device, while the receiving device acts as slave. These kind of messages that are sent out without a former request are called traps. Each netPIO can handle and store six traps (notification messages).

The master devices sends out traps, when triggered by a rising AND falling edge of a selectable AUX signal channel. Any AUX signal input and output, as well as the TEST-LED signal can act as a trigger source for traps. For example a trap may be triggered by an external switch, but it could also be triggered when an output signal changes.

A trap is sent to a certain receiver in any case, which is the slave. The slave device can be any netPIO device available on the network and that is addressable with its IP or Hostname and UDP port. This means master devices can address local slaves with their local IP, but are also able to resolve DNS hostnames (like provided by DynDNS or other services).

A trap message contains a command that instruct the slave to switch one of its AUX outputs (ON, OFF or TOGGLE), or to FOLLOW the signal status of the trigger signal.

Example 1:

Device #1 detect a status change at input AUX-C0 (maybe a switch).

Device #1 sends a message to Device #2 saying "TOGGLE AUX-D0 OUTPUT!"

Device #2 receives the message and executes the instruction (maybe for be a relay).

Example 2:

Device #1 detect that its TEST-LED was turned on.

Device #1 sends a message to Device #2 saying "FOLLOW ME WITH YOUR TEST-LED!"

Device #2 receives the message and turns on its own TEST-LED.

You may get excited by the possibility to transfer signal from one place of earth to another, without any PC being involved.

But be careful as you can run into trouble building message loops causing endless feedback and flooding your network with messages, in cause of bad configuration. Plan your traps carefully!

Here an example what you better should NOT do..

Device #1 detect change at TEST-LED

Device #1 instructs Device #2 "TOGGLE YOUR TEST-LED!"

Device #2 detect change at its TEST-LED

Device #2 instructs Device #1 "TOGGLE YOUR TEST-LED!"

Device #1 detect change at TEST-LED

etc.

Once triggered this loop will never end. Now its time to unplug from network!

## Special function

There are some additional function that can be executed from the configuration software.

- RESET the module by software (like pressing the RESET button on the device)
- RESET the application controller
- RESTORE FACTORY SETTINGS, erases the EEPROM and removes all user configuration settings. Re-configure it like a new device.

## Application controller firmware

The netPIO board can be purchased with different firmware. Firmware is identical to the firmware of our USB- $\mu$ PIO modules. A detailed description of different firmware options is located in the USB- $\mu$ PIO datasheet. Firmware port names, signal directions and functionality are **exactly the same** for both, the netPIO and the USB- $\mu$ PIO devices.

## For programmers

### Data exchange with the application controller

The application controller uses the serial protocol as described for the USB- $\mu$ PIO modules.

The communication controller **encapsulates** these serial protocols, with few additional data and transfers the data using the UDP network protocol (User Datagram Protocol).

The request-response communication is still being used.

Programs made for the netPIO must build the original USB request data block, place it into a UDP datagram and send it through the network. The addressed device answers with an UDP package, containing the (original USB) response data block.

### Request via UDP

Some few additional data must be added to the original request (as build for the USB communication), to send it in one UDP datagram.

- a ASCII string "TXDATA" heads the request, telling the communication controller to pass the following request data to the communication controller, via the board internal UART connection

- The request data is followed by two additional bytes. The first one contains the LENGTH of the response that is expected. The second is a so-called SYNC byte.

Following above instruction, we get a UDP datagram which is eight bytes longer than the request itself, caused by six bytes of "TXDATA" string and the LENGHT and SYNC byte.

```
<"TXDATA"(6 bytes)> + <Request(n Bytes)> + <ResponseLen(1 Byte)> + <Sync(1 Byte) >
```

### Response via UDP

Receiving above UDP package, the controller returns a UDP package containing the response. The response datagram is the same as described for the USB modules.

### **Some remarks**

Before the UDP request package is sent, it needs to be addressed with destination IP and UDP port. Depending on the programming language, you will find ready-made components for that purpose, where INDY components are likely most well-known.

```
...  
IndyUDP.Host = "192.168.1.222"  
IndyUDP.Port = 37155  
IndyUDP.SendBuffer(RequestDatagramm)  
...  
IndyUDP.ReceiveBuffer(ResponseDatagramm)  
...
```

Please understand that we can not answer any questions on UDP components (like Indy) and we can not offer any support for programming languages, versions and dialects.

We are unable to answer questions like "How can I send a UDP package with Visual Basic 6.0?" or "How to build a request datagram in Java?". Programming hobbyists will find help in Internet forums and tutorials for their individual programming language.

## **Other functions**

The communication can process the following commands, that can be send in a UDP datagram. The command "TXDATA" we explained before

### **"LEDTOGGLE"**

Toggles the TEST-LED status (on/off).

### **"LEDON"**

Turns on the TEST-LED.

### **"LEDOFF"**

Turn off the TEST-LED.

### **"netPIO?"**

Device returns "netPIO!"

### **"SLFRST"**

The netPIO restarts (RESET).

### **"RSTCTRL"**

The application controller resets.

### **"RSTFAC"**

Restores the factory settings and erases all user configuration data.

### **"CHANGEIP" + 4 bytes** for the new IP

Changes the device IP.

### **"CHANGEPORT" + 2 bytes** with the new port number

Changes the device UDP port.

### **"CHANGEMAC" + 6 bytes** with the new MAC

Changes the device MAC.

### **"GATEWAY" + 4 bytes** for the new gateway IP

Changes the gateway IP (=local router IP) for Internet access.

### **"AUXD<sub>x</sub>ON"**

Turn on an AUX port output (output goes high)

("x" = "2" ... "7")

### **"AUXD<sub>x</sub>OFF"**

Turn off an AUX port output (output goes low)

### **"AUXD<sub>x</sub>TOGGLE"**

Toggles an AUX port output status

("x" = "2" ... "7")

## **"GETAUX"**

Polls the AUX port status and the TEST-LED. The response is 4 bytes long. The four bytes contain a readable ASCII-String (like "F3FC"). The value of the high byte (here "F3") represents the input states, the value of the lower Byte (here "FC") represents the output state of the AUX port:

High byte: AUX-C(Bit 0..5) input status

Low byte: AUX-D(Bit 2..7) output status and TEST-LED (Bit 0)

## **"TXDATA" + n-Bytes Request + 1 Byte ResponseLen + 1 Byte Sync**

Sends the request (n bytes) via the board internal UART (57600 Baud) to the application controller (without "TXDATA" and without ResponseLen / Sync-Byte).

ResponseLen<>0 (= 1 ... 254):

Expects a response from the application that has a length given with ResponseLen.

As soon as the application controller responds, its response is packed and returned in an UDP datagram by the communication controller, not changing the response in any way.

ResponseLen=255:

In that case the SYNC byte is used as a data block end character. A response from the application controller is expected and accepted until the application controller finally returns the SYNC byte value. An UDP datagram containing the response is returned immediately after that.

ResponseLen=0 is not allowed.