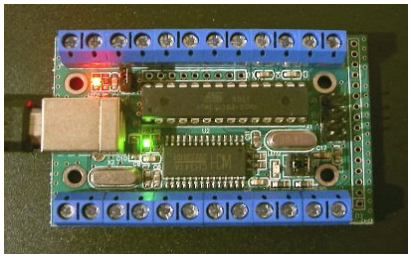
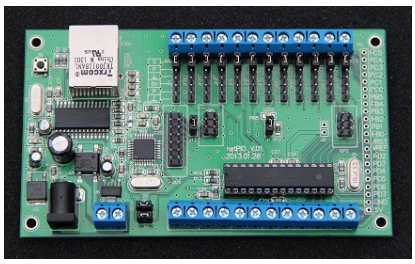
ABACOM USB-µPIO and netPIO boards





These boards are the base for a product series we developed for some frequent measurement and control applications. A USB and a network based platform is available.

We deliver the boards ready-made with one of the following firmware options:

/ TEMP12
Temperature system for Dallas DS18B20 sensors
For use with up to 12x Dallas DS18B20 sensors.

/ GPIO18
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
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
9 MHz frequency counter / generator
A powerful firmware with application software included.
Compatible with ProfiLab Expert

/ INCR3
Three incremental encode inputs
Reads the position of up to three increment encoders.
Compatible with ProfiLab Expert.

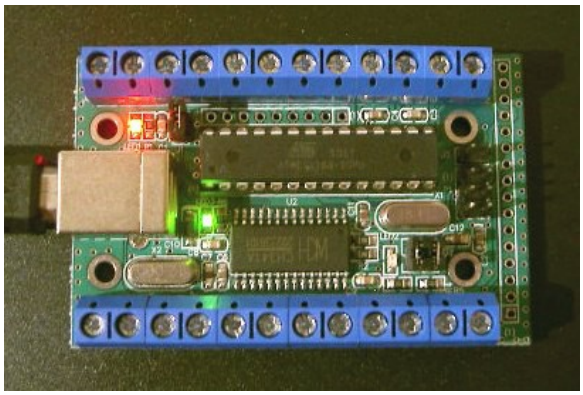
Developer can grow their DIY projects on or hardware platform as well:

/ MCS
incl. MCS BOOTLOADER for BASCOM AVR (USB only)
BASCOM AVR is your favorite AVR programming language. We equip the board with the MCS bootloader. A ISP programmer is not required as you can upload your programs directly from the BASCOM IDE via USB.

/ CLEAR
not programmed
You are experienced in AVR programming and like to flash the chip yourself with your own application. Requires ISP programmer.

To understand your product and all its details you need to read the hardware descriptions as well as the firmware description in following chaptesd, which is related to the device you purchased.

USB-µPIO



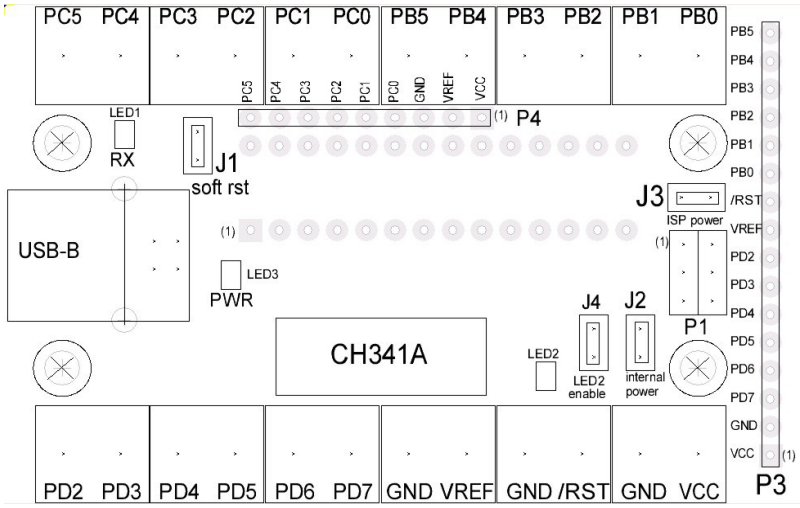
USB AVR board

- Compact AVR board with Atmel ATmega168-20
- High speed clock frequency 18.432000 MHz
- 100% error free High baud rates
- Screw terminal and pin connections
- 6 pin ISP connector
- Power supply (+5V) via USB or external
- 3 LED: USB power, COM (RxD) , USER (Port B.5; can be disabled)
- SOFT-RESET via DTR signal (can be disabled)
- USB-B connector
- USB chipset: CH341A
- Virtual (RS232-)-COM driver (VCP)
- System requirements: XP, 2000, Vista, WIN 7, 32/64 Bit
- Dimensions: approx. 70 x 45 x 15 mm

Download

http://www.abacom-online.de/div/setup_usb_µPIO.exe

Screw terminals, LEDs and jumpers



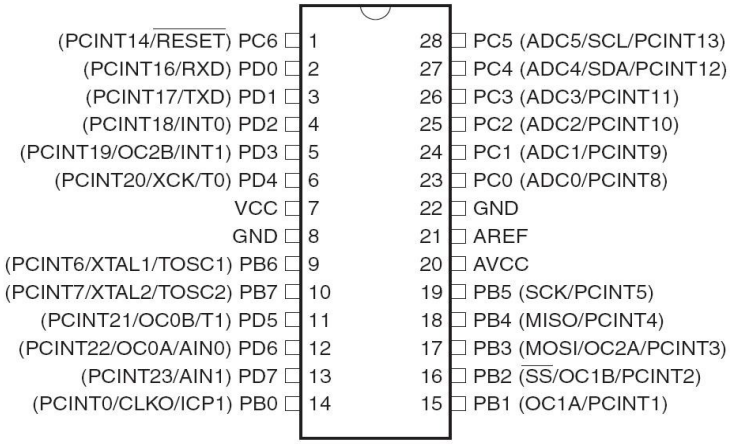
P1 – ISP connector for programmer
P2 – USB-B connector
P3 – Pin strip for Port B, Port D, VREF, RST, VCC, GND (Digital port)
P4 – Pin strip Port C, VREF, VCC, GND (Digital port und ADC port)

J1 – Jumper SOFT RST connects RESET(PC6) with DTR signal from CH341A
J2 – Jumper INTERNAL POWER connects VCC with USB supply voltage
J3 – Jumper ISP POWER connects VCC to ISP
J4 – Jumper LED2 ENABLE connects LED2 with PB5

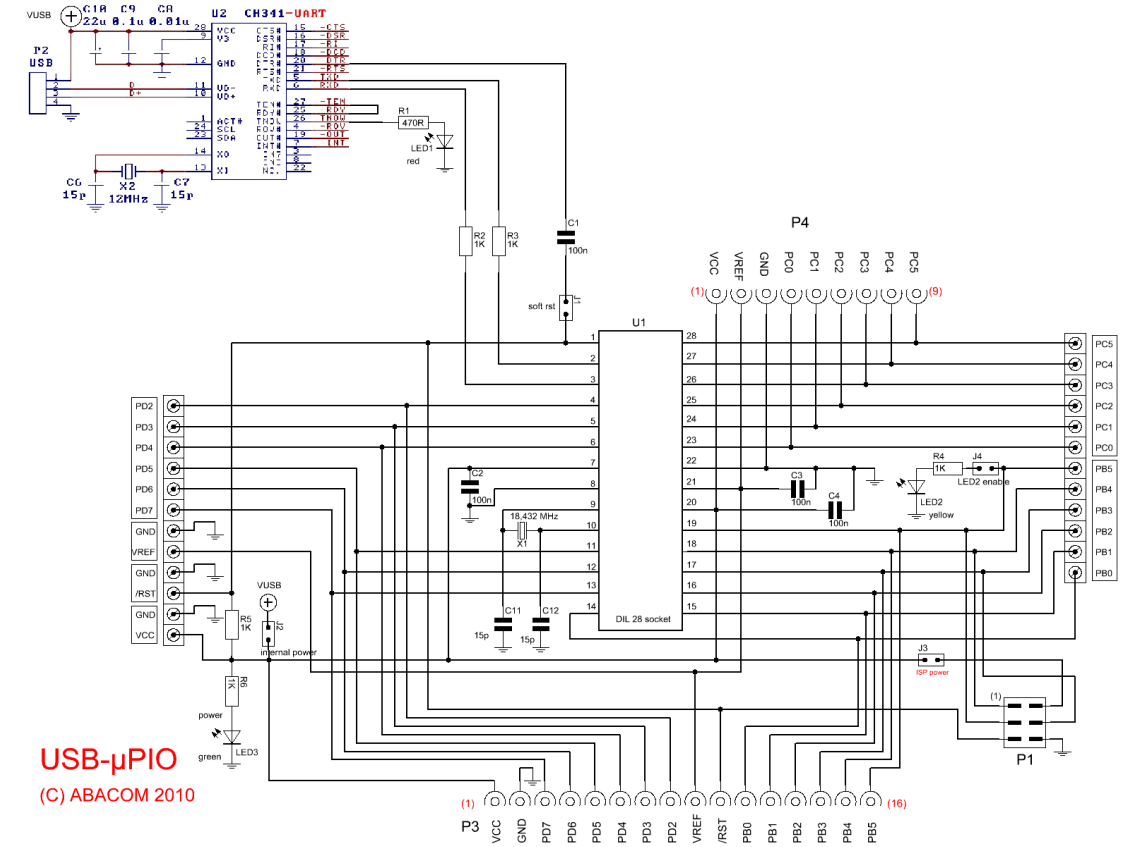
LED1 – RX indicator (red; Data received from PC)
LED2 – LED at PB5 (yellow; for user purpose)
LED3 – USB-Power (green; USB power supplied)

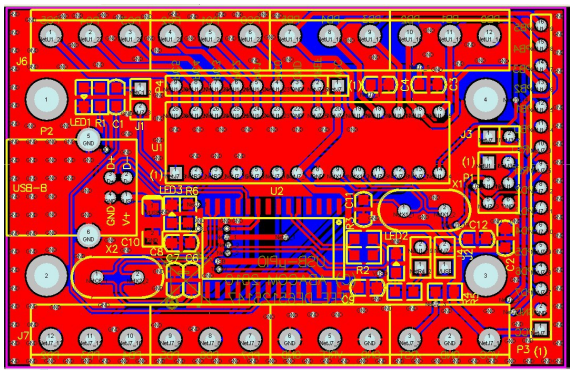
Screw terminals for Port B, Port C and Port D

Terminal names refer to ATmega168 pin names



Circuit and Dimensions

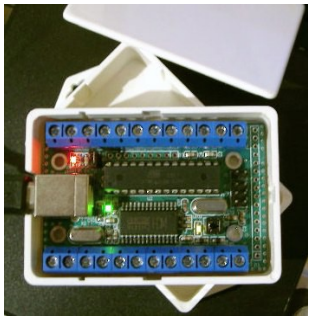




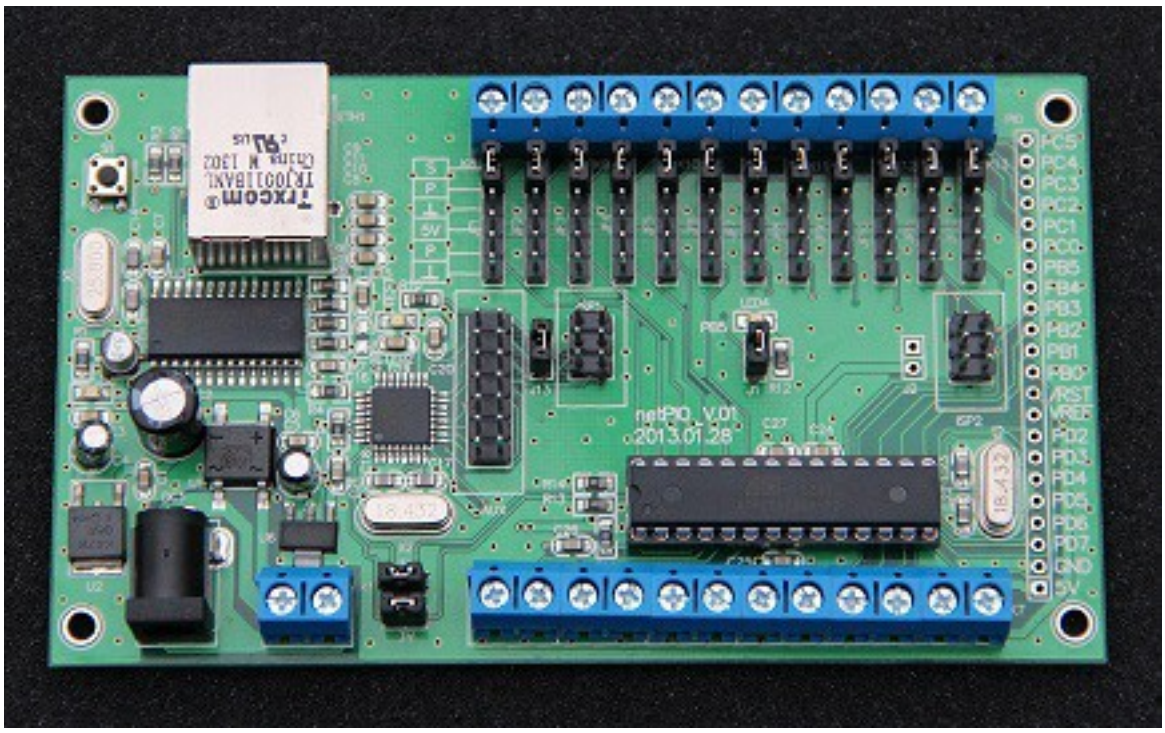
Dimensions approx. 70 x 45 x 15 mm

Available accessories:

- Enclosure (unfinished)
- USB ISP programmer with ISP cable



ABACOM - netPIO

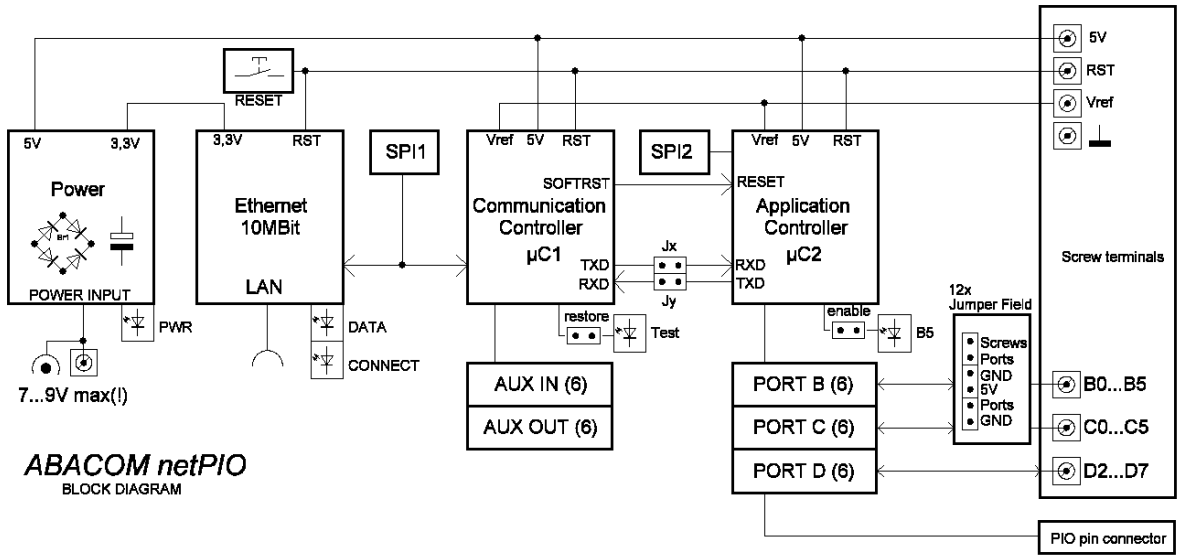


Download

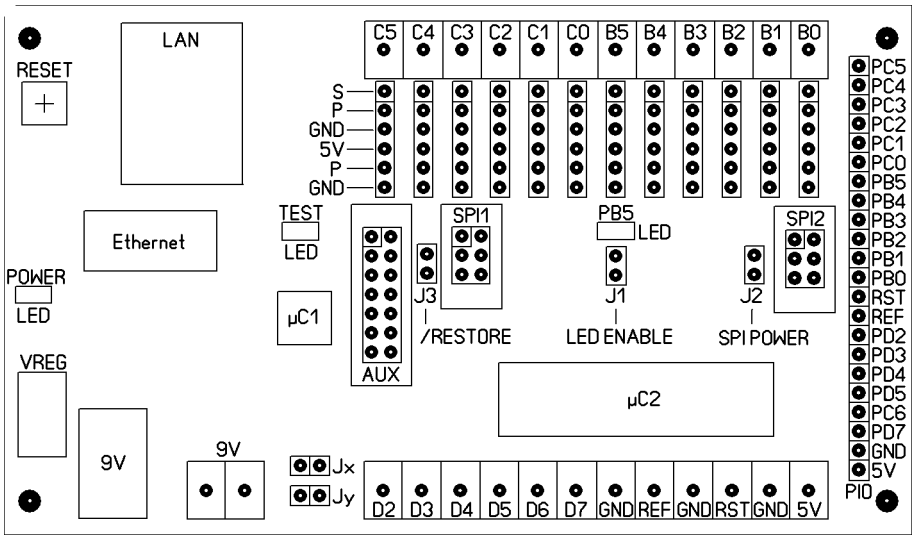
http://www.abacom-online.de/div/setup_netPIO.exe

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-µPIO modules. This chapter 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 chapter related to your firmware.

Block diagram



Connections, LEDs and jumpers



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

Voltage supply (Power)

The module must by supply 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 over all 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

The netPIO was designed with two microcontrollers on board, which we call the communication controller (µC1) and the application controller (µC2).

Ethernet connection an 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 a ATMEGA168-20 microcontroller (µC1 - Communication Controller).

A Windows configuration tool is used to configure network parameter, which is the same for all ABACOM netPIO product. 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 an 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-µPIO boards.

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 TLL 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 may 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 an 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.

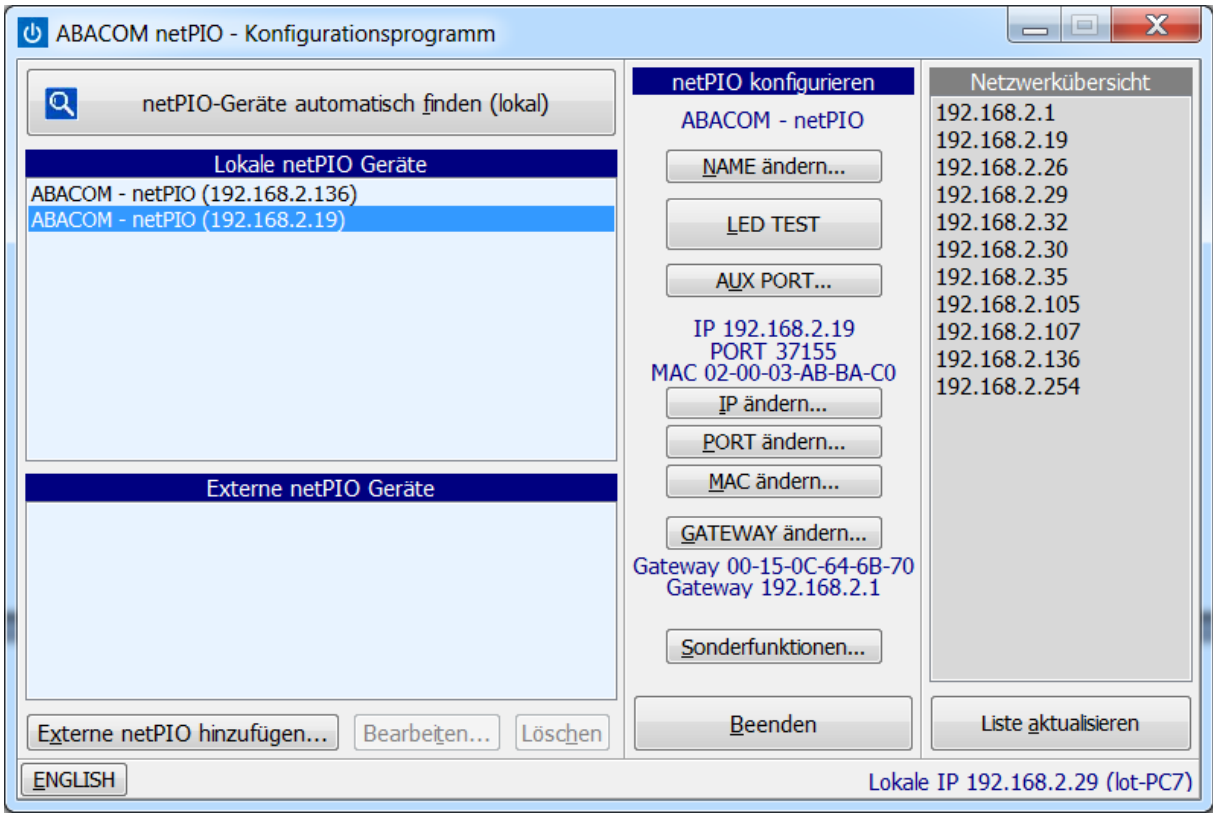
There many situation in which you may take advantage of the jumper field, but remember the limits please:
- max. 10 mA for each port
- max. external load 200mA over all
External circuits that consume more power must be supplied with its own power supply!

Some application 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.

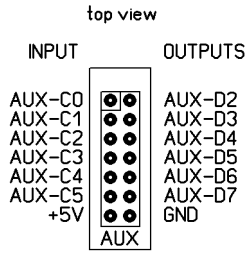
List of external devices
External devices can be added manually to this list, entering their hostnames and ports. To make devices available on the Internet, these devices must be published by forwarding their ports in their routers setup. Devices must be configured locally before you can forward them.
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 TLL 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 exited 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 functions

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-µPIO modules. Firmware port names, signal directions and functionality are **exactly the same** for both, the netPIO and the USB-µPIO devices.

Programming

Data exchange with the application controller

The application controller uses the serial protocol as described for the USB-µPIO 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**x**ON**"
Turn on an AUX port output (output goes high)
("x" = "2" ... "**7**")

"**AUXD**x**OFF**"
Turn off an AUX port output (output goes low)

"**AUXD**x**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.
A soon as the application controller responses, its response is packed and retuned 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 an 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.

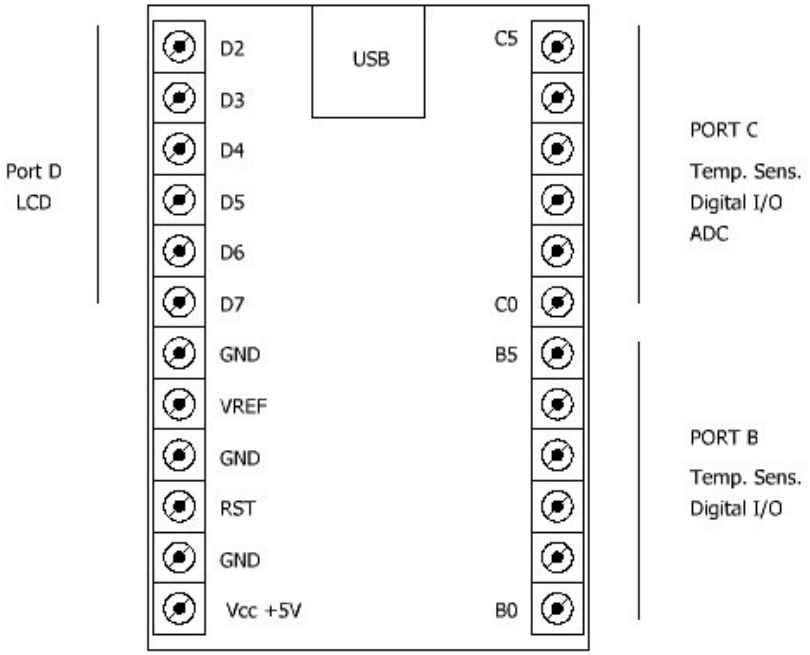
Firmware /TEMP12

The /TEMP12 firmware allows you to connect up to twelve digital temperature sensors (type Dallas DS18B20). Data from twelve channels is transferred to your PC via USB. ONE temperature sensor can be connected to each of the screw terminals B0…B5 and C0…C5.

/TEMP12 connections

Channels that are not occupied by temperature sensors, can take over alternative functions:

- DS18B20 temperature sensor
- Digital input
- Digital output
- Alarm output
- Sensor input (ADC) (channels C0…C5 only)



This chapter shows wiring diagrams using the USB-µPIO module. The can be found as well on the netPIO network module, with same functionality and arrangement.

The device is delivered with a configuration software, which allows you to setup each channel. The configuration is transferred to the device and stored temporarily or permanently. So the device can work as PC interface as well as in a stand-alone application.

A “HIGH” sense limit and a “LOW” sense limit can be configured for each analogue channel (temperature or sensor). If a channel value gets classified as HIGH or LOW, this can trigger an depending alarm output. For example a cooler could be activated if a temperature channel indicates “HIGH”. Channels that exceed its limits are also indicated on an optional LCD display.

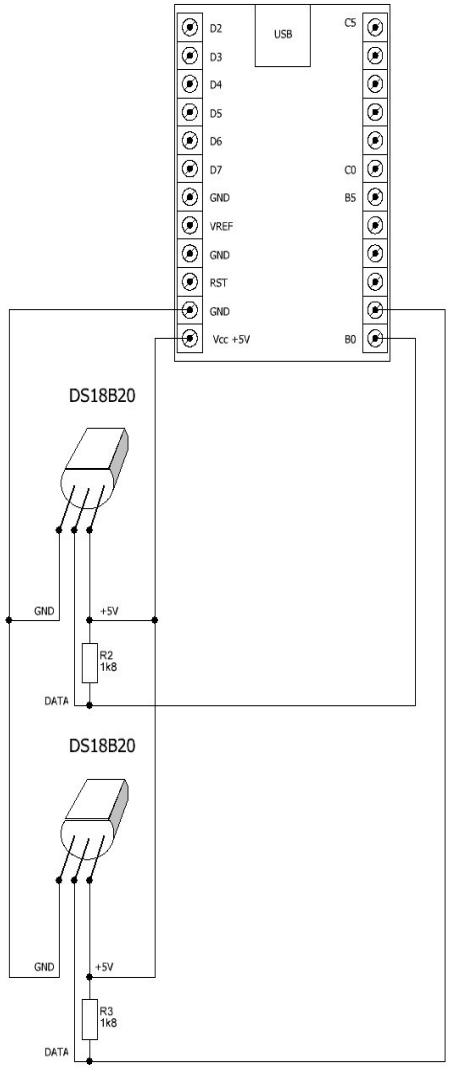
DS18B20 temperature sensor

The picture below shows how to connect temperature sensors to the board.
Temperature sensors can be operated on the channel B0…B5 and C0…C5. Use the configuration software to setup the mode for each channel. The DS18B20 can measure temperatures from -55 °C und 125° C. In case you need to change the unit (for example from °C to °F) you can enter a linear sensor scale factor and offset.

Example – Scale °C to °F:
Unit = „°F“
Factor = 1,8
Offset = 32

Understand that scaling a sensor does NOT change its physical range (-55 °C to 125° C), but only changes its representation. Limits for „HIGH“ and „LOW“ detection must be entered in SCALED units.

Wiring diagram



This chapter shows wiring diagrams using the USB-µPIO module. The can be found as well on the netPIO network module, with same functionality and arrangement.

ADC sensor input

Channels C0…C5 can be configured as ADC sensor inputs. In this channel mode an analogue-to-digital-converter measures an input voltage, which may be from a light sensor for example..

The ADC uses a reference voltage VREF to measure the input voltage in volt units. There are three possible sources for the reference voltage. First is the supply voltage (5V USB), second is a internal voltage source (1.1V) and third is an external reference voltage supplied at REF input. The reference voltage source can be selected for each ADC channel input separately. The reference voltage VREF must never exceed the supply voltage! The input voltage must be between 0V and VREF, which defines the input range. Voltages at all pins must never go below 0 Volt (GND) or above supply voltage (VCC)!

With a correct VREF setting the ADC will report the input status in volt units. ADC channels can be scaled to different units with linear scale factor and offset.

Example – Scale voltage (with VREF = 5V) into percentage

Unit = „%“
Factor = 20
Offset = 0

The input channel is now represented as percentage, as 5(V) * 20 = 100%. Understand that scaling a sensor does NOT change its physical range (0V … VREF), but only changes its representation. Different ranges of sensors must be adapted with suitable input circuits (voltage divider, amplifier or similar)

Limits for „HIGH“ and „LOW“ detection must be entered in SCALED units.

Alarm output

In this channel mode a channel works as digital output, that outputs high level (5V) if the channel is ON or low level (0V) if the channel is OFF. The output can drive up to 20 mA loads.The alarm output will turn ON and OFF automatically, in case a certain CONDITION is true or false.
Configurable conditions are the “HIGH” and “LOW” states of other analogue channels (temperature or ADC inputs). To avoid flickering of alarm outputs in case the inputs value swings slightly around a limit, a fixed hysteresis of +/-2 LSB was built in.

Digital output

In this channel mode a channel works as digital output, that outputs high level (5V) if the channel is ON or low level (0V) if the channel is OFF. The output can drive up to 20 mA loads. The output is controlled by the PC.

Digital input

This channel mode sets the channel direction to input. A high level (5V) is interpreted as “ON”. A low level (0V) represents “OFF”. Open (not connected) inputs are internally pulled high. This makes

it possible to sample the status of a switch, which connects the input to ground (GND) if contacts a closed.

Using the LCD option, the LCD will show the status of digital channels (digital inputs, digital outputs, alarm outputs) as “ON” or “OFF”

Measuring differences between sensor values

In practice you may need to measure differences between sensors, like indoor temperature – outdoor temperature. For that purpose the /TEMP12 firmware offers four additional (virtual) channels. The values for these channels are calculated from the difference of two sensors as follows:

- DIFF1 = Channel C0 – Channel C1
- DIFF2 = Channel C0 – Channel C2
- DIFF3 = Channel C0 – Channel C3
- DIFF4 = Channel C4 – Channel C5

Both channels of a pair must be configured identically, to enable the differential channels. (Otherwise the difference would not make sense.) The only configurable parameters for the differential channels are the “HIGH” and “LOW” limits, which can be used as conditions for alarm outputs as any other sensor channel.

Device setup

All configured parameters must be transferred to the device and are stored on the device for normal operation. The configuration software must be used for this configuration upload. The configuration can be stored temporary in the RAM memory or permanently in the EEPROM memory of the device. A device reset will read the configuration stored in the EEPROM memory. A configuration stored temporary in the RAM memory will be overwritten in this case. A reset occurs when the device is powered up or is triggered by a low pulse at the reset input (RST). If jumper J1 (SOFT-RESET) is set, a reset also occurs when PC software establishes a data connection, opening the COM port. Avoid this behaviour removing Jumper J1 if necessary.

During upload process a internal clock is synchronized with the PC time and date setting, Allowing to display tome and date information on the optional LCD display. Time information gets lost for technical reasons, when a device reset occurs. Time and date will not be displayed in that case. A permanent power supply is needed to display time information.

/TEMP12 protocol

Parameters

Baud 57600
8 data bits
No parity
1 stop bit

Request / Response communication is used. A request consists of 21 data bytes. After receiving 21 bytes, the device responses with a response data block in any case. A new request must not be send, before the response on the previous request was received completely.

Request: 21 bytes

The request data block is 21 bytes long and transfers all control data.

Byte 1:	Chr(66) =„B" in any case
Byte 2…Byte 13:	Digital output states for channels B0…B5, C0…C5
Byte 14…21:	Reserved

A request must start with a character „B". The following characters (Byte 2 … 13) control the output state of channels that have been configured as digital outputs. Channels that are not configured as digital output will not be affected. A character "0" = chr(48) turns the corresponding channel OFF. A character "1" = chr(49) turn the corresponding channel ON.

Any other character will be ignored.

Programming example (with PRINT command on serial port):

```
Print „B111111111111--------";
```

Above command will turn all (configured) outputs ON. Note the semicolon at the end of each line. A chr(13) chr(10) must not be sent after the 21 bytes of the request!

```
Print „B000000000000--------";   // Turn off all digital outputs (low)
Print „B100000000000--------";   // Set B0 HIGH; all others LOW
Print „Bxxxxx11xxxxx--------";   // Set B5 and C0 HIGH; do not change others
Print „Bxxxxx01xxxxx--------";   // Set B5 LOW; Set C0 HIGH; do not change others
Print „Bxxxxx01xxxxx--------";   // Set B5 LOW; Set C0 HIGH; do not change others
Print „Bxxxxxxxxxxxx--------";   // No change of outputs (request only)
```

The device will send a response after each of above PRINT commands.

Response

After receiving 21 request bytes the device will send a response to the PC. The response is a readable character chain (String). The response string consists of several so-called “cells”. Each cell is six characters long, followed by a semicolon as separation character. The first cell contains the channel name of the first channel (B0). The second cell contains the channel value for channel B0. Two cells are sent for each channel. A pair of two cells (Name and value) is sent for each channel. Channels B0…B5, C0 … C5 and DIFF1 …4 are sent one after each other.

Example response:

```
„ ALARM;  EIN ;B1-OUT;  EIN ;B2-OUT;  EIN ;B3-OUT;  EIN ;B4-OUT;  EIN ; PC-B3;  EIN ;
INNEN; 20.9; TANK ;96.77%;C2-ADC; 1.07;C3-ADC;4.24;C4-ADC; 4.29 ;C5-ADC; 4.38 ;  --  ;
--  ;  --  ;  --  ;  --  ;  --  ;  --  ;  --  ;" +CHR(13)CHR(10)
```

Finally the response is terminated with chr(13)chr(10). Altogether 32 cells (á 6 characters + Semicolon) and two terminating character are sent. This makes 32 * 7+2 = 226 characters.

The cell content is used to display information on the optional LCD display as well. Therefore cells contain some special characters. To get a readable representation on the PC, the character must be replaced as follows:

Chr(1)	=	µ
Chr(2)	=	°
Chr(3)	=	°C
Chr(4)	=	°F
Chr(5)	=	°K
Chr(6)	=	€

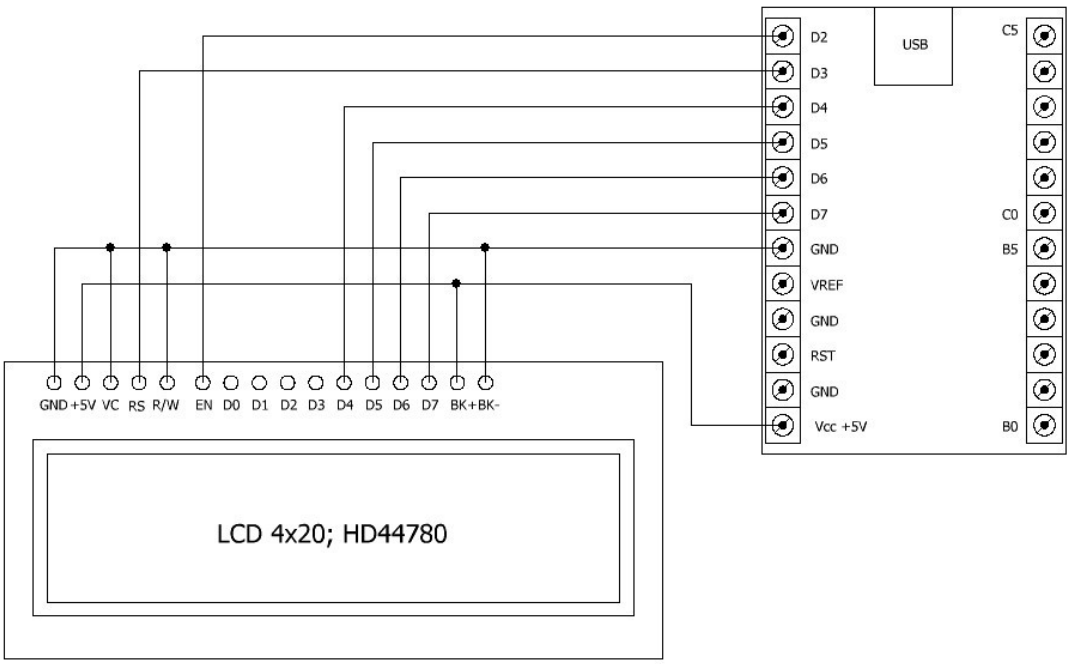
LCD option

A standard LCD text display (HD44780) with 4x20 characters can be connected on Port D2..D7. Other display formats are NOT supported. The display will show value and status of call channels automatically. Connections between the board and the LCD display must be made as follows:

BOARD	LCD 4x20 characters
PORT D	HD44780
D2	EN
D3	RS
D4	D4
D5	D5
D6	D6
D7	D7

LCD input R/W must be connected to ground (GND). The remaining data lines **<u>of the LCD</u>** D0…D3 are not connected.

Wiring diagram



This chapter shows wiring diagrams using the USB-µPIO module. The can be found as well on the netPIO network module, with same functionality and arrangement.

Firmware /GPIO18

The /GPIO18 firmware works as PC interface with 18 digital I/O lines. The data direction for each line is programmable (18 GPIO).

/GPIO18 connections

I/O lines are combined to ports B, C an D. Each port offers six I/O lines.

Port B0…PortB5	Digital I/O
Port C0…PortC5	Digital I/O = ADC 0..5
Port D2…PortD7	Digital I/O

A internal pull-up resistor can be activated by software on all lines that are programmed as input. This could be useful to bind open inputs to a defined potential.

In addition voltages at port C lines can be read with an internal 10 bit ADC. The ADC reference voltage can be taken from three sources:

External reference voltage
A individual reference voltage is supplied trough the VREF input (0<Vref<Vcc).

Internal reference voltage VCC
The positive supply voltage (Vcc) is used as reference voltage. (approx. 5V)

Internal reference voltage 1.1 V
A 1.1 Volt reference is generated internally.

/GPIO18 protocol

Parameters

Baud 57600
8 data bits
No parity
1 stop bit

Request / Response communication is used. A request consists of seven data bytes. After receiving seven bytes, the device responses with a 15 bytes long response data block
A sample rate of approx. 150 samples/sec can be achieved.

Request: 7 bytes

Byte1: ADC reference selection (0=EXTERN; 1=INTERN VCC; 3=INTERN 1.1V)
Byte2: Direction Port B
Byte3: Direction Port C
Byte4: Direction Port D
Byte5: Data Port B
Byte6: Data Port C
Byte7: Data Port D

Bytes 2..4 set the data direction of each port line. A SET bits sets the corresponding line to output, a CLEAR bit sets the line to INPUT.

Bytes 5..7 set the output state for each port line. A SET bit sets the output HIGH, a CLEAR bit sets the output LOW. In case a line was set to INPUT direction, a SET bit ACTIVATES the pull-up for the input line and a CLEAR bit deactivates the pull-up for the input line.

The bit numbers in the data bytes correspond with port line numbers. For example a SET BIT 2 in request byte 4, defines port line D2 as output.

Response: 15 Bytes

Byte 1: Status port B
Byte 2: Status port C
Byte 3: Status port D
Byte 4: ADC0 LSB
Byte 5: ADC0 MSB
Byte 6: ADC1 LSB
Byte 7: ADC1 MSB
Byte 8: ADC2 LSB
Byte 9: ADC2 MSB
Byte 10: ADC3 LSB
Byte 11: ADC3 MSB
Byte 12: ADC4 LSB
Byte 13: ADC4 MSB
Byte 14: ADC5 LSB
Byte 15: ADC5 MSB

Bytes 1…3 represent the digtial state of the I/O lines. A SET bit indicates a HIGH level at the corresponding line. This is NOT depending on the lines data direction.

Bytes 4...15 are paired to six words, containing the 10 bit conversion result of the ADC channels (Port C0…C5). Voltage can be calculated as follows: Voltage = VREF * (MSB * 256 + LSB) / 1023

Firmware /FREQ

The /FREQ firmware is a comfortable 9-MHz frequency counter with additional features, like digital I/O, PWM output, pulse and clock generator and six ADC channels.

Digital-I/O

All I/O lines have a fixed data direction, that can not be changed. All lines use TTL levels (5V) and can be used as digital input or output, depending on their direction. Lines overtake alternative functions depending on the application and configuration, but they never change their direction.

9-MHz frequency counter

D2 and D5 are used as frequency measure inputs. Usually the input signal gets connected to BOTH inputs (bind together). The measurement principle changes automatically depending on the input frequency. High frequencies are measured by edge counting during a gate time interval (D5; F>250 Hz). Low frequencies are measured by edge detection and measurement (D2; F<250 Hz).

Gate time intervals can be 2000ms / 1000 ms / 500ms / 100 ms / 50 ms / 20 ms / 10ms.
The gate clock can be output through digital output D7

For low frequency signals (< 250 Hz) the elapsed time since last falling edge on D2 is measured and reported. Frequency measurement, counter function and time measurement can be used for low frequencies simultaneously. For high frequencies only frequency measurement and counter function are available alternatively.

32-bit counter

With gate time generator deactivated, the frequency counter works as simple 32-bit up counter, counting falling edges at D5. Digital input B0 can be configured as external RESET input for the counter. Digital input B4 can be configured as external ENABLE input for the counter.

Generator 1

A periodic output signal is generated at digital output D6. An input clock drives a internal 8 bit counter. The counter value gets compared and the compare result generates the output clock at D6. Depending on the configuration the output signal can be a clock with adjustable frequency or pulse width, The input clock signal can be internal or external.

Internals clock sources are devided system clocks (18.432000 MHz) with divisors of 1, 8, 64, 256 or 1024. Input D4 can be configured as external ENABLE input for the internal clock source.

External clocks are supplied through input D4 as well. External clocks at D4 can be enabled or disabled by software.

Generator 2

A pulse with adjustable pulse length is output at D3. Duration of HIGH and LOW signal output is adjusted separately in multiples of a common time base value (16 bits resolution). Available time bases are 125µs, 500µs, 5ms und 500 ms. The output pulse can be generated continuously or single shot. A single shot can be triggered by software. Digital input B1 can be configured and used as trigger as well. In continuous mode a trigger will cause the output clock to synchronize with the trigger (software or B1).

Generator 3

A adjustable (8 bit) PWM signal with a fixed frequency of 72kHz is generated at digital output B3.

/FREQ connections

B0 Digital Input (Counter RESET)
B1 Digital Input (Sync/Trigger Generator 2)
B2 Digital Output
B3 Digital Output (Generator 3 PWM)
B4 Digital Input (Counter enable)
B5 Digital Output (LED)

D2 Digtal Input (Frequency input 0-250Hz)
D3 Digital Output (Generator 2)
D4 Digtal Input (EXT CLK ENABLE)
D5 Digtal Input (Frequency input 250 Hz - 8MHz)
D6 Digital Output (Generator 1)
D7 Digital Output (Gate Clock Output)

C0…C5 Digital Inputs / ADC 0…5

Inputs of Port B and Port D are pulled up to Vcc internally. Port C provides digital inputs without pull-ups. Voltages at port C can be read with the internal ADC. The reference voltage for the ADC can be taken from three different sources:

External reference voltage:
A individual reference voltage is supplied trough the VREF input (0<Vref<Vcc).

Internal reference voltage VCC
The positive supply voltage (Vcc) is used as reference voltage. (approx. 5V)

Internal reference voltage 1.1 V
A 1.1 Volt reference is generated internally.

/FREQ protocol

Parameters

Baud 57600
8 data bits
No parity
1 stop bit

Request / Response communication is used. A request consists of 10 data bytes. After receiving 10 bytes, the device responses with a 33 bytes long response data block

Request: 10 Bytes

The request data block is 10 bytes long and transfers all control data.

Byte 1 Gate time and Digital Out
Bits 0..2 select the gate time interval fort he frequency measurement:

Bit 210	Hex	Dec	Gate time
000	00h	0	2 s
001	01h	1	1s
010	02h	2	500 ms
011	03h	3	100 ms
100	04h	4	50 ms
101	05h	5	20 ms
110	06h	6	10 ms
111	07h	7	OFF / COUNTER

Bits 3…7 are assigned to the digital outputs B2, B3, B5, D3 and D6, one after each other. These bits control the output state and activate alternative functions for the corresponding output.

Byte 2 Compare value for generator 1
This byte adjust the compare value of generator 1 and varies output frequency or PWM depending on the generator mode.

Byte 3 Compare value generator 3
This byte adjust the compare value of generator 3 and varies PWM.

Byte 4 Mode
Following functions are encoded in byte 4 bitwise:

- Mode and clock source for generator 1
- Time base for generator 2
- Counter reset by software

The value for byte can be calculated, adding up the decimal of HEX values of the desired functions.

Bits 0...2 select the clock source for **generator 1**.

Bit [210]	Hex	Dec	Clock source	
000	00h	0	NONE	
001	01h	1	CLK	18.432000 MHz
010	02h	2	CLK / 8	2.304000 MHz
011	03h	3	CLK / 64	288.000 KHz
100	04h	4	CLK / 256	72.000 KHz
101	05h	5	CLK / 1024	18.000 KHz
110	06h	6	EXTERNAL CLOCK (D4), falling edge	
111	07h	7	EXTERNAL CLOCK (D4), rising edge	

Bit 3 resets the (frequency) counter to zero (**Counter Software Reset**), if bit is SET.

Bit	Hex	Dec	
[3]	08h	8	Counter Reset

Bit 4 and Bit 5 select the **mode of generator 1**

Bit [54]	Hex	Dec	Mode Generator 1
00	00h	0	Normal
01	10h	16	PWM
10	20h	32	CTC
11	30h	48	Fast PWM

Mode: Normal
The generator devides the input frequency by 512.

Fout = (Fin / 2) / 256

Fixed output frequencies with internal clock source:

36.000 kHz
4.500 kHz
562,5 Hz
140,625 Hz
35,15625 Hz

Mode: PWM
The generator devides input frequency by 2 and by 255 after that.
The pulse width is varies with the compare value.

Fout = (Fin / 2) / 255

PWM frequencies with internal clock sources:

36.141 kHz
4.518 kHz
564,7 Hz
141,17 Hz
35,294 Hz

Modus: CTC
The input frequency is devided by 2 and then devided by compare value +1.
So the output frequency varies with the compare value.

Fout = (Fin / 2) / (CMP1 + 1)

Frequency ranges with internal clock sources:

36.141 kHz – 9.216 MHz
4.518 kHz – 1.152 MHz
564,7 Hz - 144.0 kHz
141,17 Hz - 36.0 kHz-
35,294 Hz – 9.0 kHz

Modus: Fast PWM
The input frequency is devided by 256.
The pulse width is varies with the compare value.

Fout = Fin / 256

PWM frequencies with internal clock sources:

72.000 kHz
9.0 kHz
1.125 kHz
281,25 Hz
70,3125 Hz

Bit 6 und Bit 7 select the **time base for generator 2** ein.

Bit [76]	Hex	Dec	Time base Generator 2
00	00h	0	125 µs
01	40h	64	500 µs
10	80h	128	5 ms
11	C0h	192	500 ms

Byte 5…6 TH; Generator 2 ; Duration HIGH (INT16 / WORD)
Byte 7…8 TL; Generator 2; Duration LOW (INT16 / WORD)

Two bytes make up a 16 bit value, which sets the pulse HIGH/LOW duration of generator 2.
Time is the result of the multiplication of the duration value and the tme base value:

T high [s] = TH * Time base
T low [s] = TL * Time base

With TH = TL = 1 and time base = 125µs we get the
maximum output frequency of 4 kHz. f = 1/ ((TL + TH) * Time base)

With TH = TL = FFFFh and time base = 500ms we get the
maximum period length of 65535 seconds, which is more than 18 hours
T = (TL + TH) * time base

Byte 9 Configuration And Digital Out (Flags)
Bits in this Byte are flags for some configurations and functions.

Bit 0 If Bit 0 is SET, B0 becomes RESET input for the (frequency) counter.
Bit 1 If Bit 1 is SET, B1 becomes SYNC/TRIGGER input for generator 2.
Bit 2 If Bit 2 is SET, B4 becomes ENABLE input for the (frequency) counter.
Bit 3 Bit 3 sets the mode of generator 2: Continuous / Single-shot
Bit 4 Bit 4 is SYNC/TRIGGER for generator 2 (software trigger)
Bit 5 In case of **external** clock source for generator 2:

Bit 5 SET enables the clock input from D4
Bit 5 CLEAR disables the clock input from D4
In case of **internal** clock source for generator 2:
Bit 5 SET: D4 is enable input for internal clock.
Bit 5 CLEAR: Internal clock enabled; independent from D4

Bit 6 Bit 6 activates Gate Clock Output at D7.
Bit 7 Bit 7 turns output D7 on/off.

Byte 10 **ADC reference selection (VREF SELECT)**

Bit 0..1 select the reference voltage source for the ADC conversion (C0…C5)

Bit 10	Hex	Dec	VREF
00	00h	0	External from VREF terminal
01	01h	1	Internal (Vcc)
10	02h	2	undefined
11	03h	3	Internal 1.1 Volt
Bit 2..7	Reserved		Set to zero

Response: 33 Bytes

The device sends a 33 Byte response after a 10 Byte request.

Byte 1…4 Edge measurement T1 (INT32 / DWORD)
Byte 5…8 Edge measurement T2 (INT32 / DWORD)
Byte 9…12 Edge measurement T3 (INT32 / DWORD)

Four bytes make up an 32 bit value (LSB first).
Three words T1, T2 and T3 deliver the result of the edge measurement. A falling edge on D2 triggers the readout of a free running 32 bit counter, while the counter is clocked with a fixed 72kHz clock (T = 13,8µs). The readout result is reported in data word T2, while the previous readout value is shifted to data word T1.

The signal frequency and period can be calculated as follows:

T = (T2 – T1) * 13,8 µs
F = 1 / T

As the counter will overflow at $FFFFFFFF and start from zero, it is recommended checking the T2>=T1 condition.

Data word T3 is filled with the current counter value. So the difference (T3-T2) delivers the time elapsed since last falling edge at D2.

T elapsed = (T3 – T2) * 13,8 µs

Check T3>=T2 condition.

Byte 13…16 Frequency counter / Counter CNT (INT32 / DWORD)
Four bytes make up an 32 bit value (LSB first). The data word CNT contains a counter value of the 32 bit counter clocked from D5. In case a gate time is activated, the CNT value is readout from the counter when gate time elapses and the counter is reset. (For calculation of frequency: see Byte 20)

If no gate time is active the data word contains the current counter value.

Byte 17 Digital state Port B (Byte)
Bits 0..5 indicate the digital state of lines B0…B5.

Byte 18 Digital state Port C (Byte)
Bits 0..5 indicate the digital state of lines C0…C5.

Byte 19 Digital state Port D (Byte)
Bits 2..7 indicate the digital state of lines D20…D7.
The digital state is independent from the data direction.

Byte 20 Gate time GT [1/100 s]
This byte contains the gate time (in 1/100 s units) that was active for above CNT value.
Frequency at D5 can easily be calculated by software:

F [Hz] = CNT / (GT / 100)

If no gate time is active, a frequency can not be calculated, as GT is zero.

Byte 21..22 ADC0 (Word)
Byte 23..24 ADC1 (Word)
Byte 25..26 ADC2 (Word)
Byte 27..28 ADC3 (Word)
Byte 29..20 ADC4 (Word)
Byte 31..32 ADC5 (Word)

Bytes 21...32 are paired to six words, containing the 10 bit conversion results of the ADC channels (Port C0…C5). Voltage can be calculated as follows:

Voltage = VREF * (MSB * 256 + LSB) / 1023

U0 [Volt] = ADC0 / 3FFh * Vref

Byte 33 Reserved
Byte 33 is reserved and currently 13.

Firmware /INCR3

The /INCR3 firmware can decode positions of three incremental encoders (TTL level). The encoder signals (A; B) are counted with 32 bit (position) counters. A falling edge at input A is the clock for the counter. Counter direction is depending on the status of B input. (high = up / low = down).
Each of three input channels is equipped with an ENABLE input.
Maximum count rate is approx 20 kHz (ratio 1:1).
A index signal Z can be used, if available from the encoder. A falling edge on input Z will reset the position counter to zero, so the position counter is synchronized with the Z index. In addition the Z signal is the clock for a 16 bit (cycle) counter. The cycle count direction taken from the position counter direction.

/INCR3 connections

The encoder signals are connected to port B and port C as follows:

Encoder 1

Port C0	Input	1A	CLK
Port C1	Input	1B	DIR
Port B0	Input	1Z	/RES
Port B3	Input	1EN	EN

Encoder 2

Port C2	Input	2A	CLK
Port C3	Input	2B	DIR
Port B1	Input	2Z	/RES
Port B4	Input	2EN	EN

Encoder 3

Port C4	Input	3A	CLK
Port C5	Input	3B	DIR
Port B2	Input	3Z	/RES
Port B5	Input	3EN	EN (remove Jumper J4 LED!)

All inputs use TTL level (5V) and are bound to Vcc with internal pull-up resistors.

Port D is a general purpose digital I/O port, with programmable direction for each line.
Line D6 delvers a clock signal if set to output. The base clock frequency is 9 kHz which can be devided through 1,2,3,….,256. A pull-up can be activated for each input line of port D.

The device gets programmed with commands, send out with the request data block of data protocol./INCR3 protocol

Parameters

Baud 57600
8 data bits
No Parity
1 stop bit

Request / Response communication is used. A request consists of 5 data bytes. After receiving 5 bytes, the device responses with a 21 bytes long response data block
A sample rate of approx. 150 samples/sec can be achieved.

Request: 5 Bytes

Byte 1: Command
Byte 2…5: Parameter

The command (Byte1) contains a function number, that cause some certain device actions. Undefined function numbers are answered with an resonse block as well, but do not cause any further action. A parameter (Byte2…Byte4) must be sent in any case, even if it is only recognised with some of the commands. The four bytes of the parameter are interpreted as a value of 32 bits, 16 bits or 8 bits, where Byte 2 is least significant and Byte 5 is most significant.
The following Counter commands can be sent out as a request block:

Decimal	HEX	CHR	Description	Parameter
65dez	41h	„A“	Reset Counter 0	don´t care
66dez	42h	„B“	Reset Counter 1	don´t care
67dez	43h	„C“	Reset Counter 2	don´t care
68dez	44h	„D“	Reset Z-Counter 0	don´t care
69dez	45h	„E“	Reset Z-Counter 1	don´t care
70dez	46h	„F“	Reset Z-Counter 2	don´t care
71dez	47h	„G“	Load Counter 0	INT32 / DWORD
72dez	48h	„H“	Load Counter 1	INT32 / DWORD
73dez	49h	„I“	Load Counter 2	INT32 / DWORD
74dez	4Ah	„J“	Load Z-Counter 0	INT16 / WORD
75dez	4Bh	„K“	Load Z-Counter 1	INT16 / WORD
76dez	4Ch	„L“	Load Z-Counter 2	INT16 / WORD

Port D commands

88dez	58h	„X“	Clock devisor f=9 KHz / (N+1)	Byte
89dez	59h	„Y“	PORTD = Status / Pull-up	Byte
90dez	5Ah	„Z“	DDRD = Data direction	Byte

Request example
Byte1…Byte 5: <5A><FC><00><00><00>
<5A> = Command „Data direction Port D“
<FC> = %1111 1100 = Bit2…Bit7 SET = D2…D7 Direction set to OUTPUT

After receiving the five request bytes, the device will send a response. Do not send a new request before having received the response.

Response: 21 Bytes

Port status:

Byte1:	Status Port B	Bit 0..5 = B0..B5
Byte2:	Status Port C	Bit 0..5 = C0..C5
Byte3:	Status Port D	Bit 2..7 = D2..D7

Position counters:

Byte4:	Counter 1	Bit 0…7 (LSB)
Byte5:	Counter 1	Bit 8…15
Byte6:	Counter 1	Bit 16…23
Byte7:	Counter 1	Bit 24…32 (MSB)
Byte8:	Counter 2	Bit 0…7 (LSB)
Byte9:	Counter 2	Bit 8…15
Byte10:	Counter 2	Bit 16…23
Byte11:	Counter 2	Bit 24…32 (MSB)
Byte12:	Counter 3	Bit 0…7 (LSB)
Byte13:	Counter 3	Bit 8…15
Byte14:	Counter 3	Bit 16…23
Byte15:	Counter 3	Bit 24…32 (MSB)

Cycle counters (Z):

Byte16:	Counter 1	Bit 8…15 (LSB)
Byte17:	Counter 1	Bit 0…7 (MSB)
Byte18:	Counter 2	Bit 8…15 (LSB)
Byte19:	Counter 2	Bit 0…7 (MSB)
Byte20:	Counter 3	Bit 8…15 (LSB)
Byte21:	Counter 3	Bit 0…7 (MSB)

Bytes 1…3 deliver the digital line status. A SET bit indicates a HIGH level. This is independent from the lines direction.

Bytes 4..15 contain the 32 bits counter values (position).
Bytes 16…21 contain the 16 bits counter values (cycles; Z)

Firmware /PWMIO18

The /PWMIO18 firmware works PC interface with 18 digital I/O lines with adjustable data direction (18 GPIO). All I/O lines can generate a 8 bit / 50 Hz PWM signal, as well as blink an pulse signals. Port D can be used to control servo drives. It generates a 50 Hz-PWM-signal with a pulse duration of 1..2 ms for that purpose.
Servo and PWM outputs are controlled by software, by ADC inputs or by an internal ramp generator. Ramps and pulses can be triggered with Port B signals.

/PWMIO18 connections

I/O lines are grouped to six ports (Port B, Port C, Port D). Each I/O line can be configured for alternative purposes:

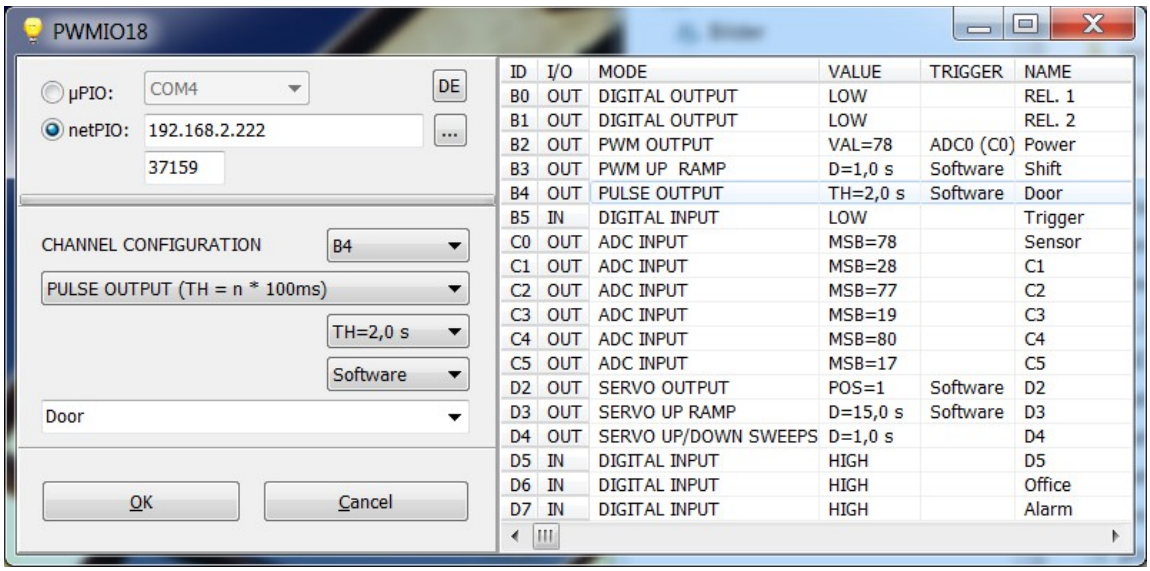
Port B0...Port B5	Digital I/O, PWM, Blink/Pulse output, 6x Trigger input
Port C0...Port C5	Digital I/O, PWM, Blink/Pulse output, 6x ADC
Port D2...Port D7	Digital I/O, PWM, Blink/Pulse output, 6x Servo

In addition Port C voltages can be read with a 10 bit ADC, using three different reference options:

External reference voltage: External Vref is supplied through VREF screw terminal.
0 < Vref < Vcc must not be overdriven.

Internal reference VCC: Positive 5V supply voltage is used as reference.
Internal reference 1.1 V: An internal reference voltage of 1.1 V is used.

Input channels can be configured to be "pulled high" by an internal pull-up resistor, to have a defined high level that could be pulled low by switch contacts.
The module powers up with an individual configuration storable in the modules EEPROM, so it is useful for some small stand-alone control applications as well.
User- defined friendly names (17 chars / channel) can be assigned to each channel.



A configuration software helps you to setup the desired configuration.

/PWMIO18 protocol

Parameters

Baud 57600
8 Data bits
No parity
1 Stop bit

Request / response communication is being used. A request is 20 bytes long in any case. A response with same length and format will be returned on any request.
The sample rate is about 10 samples/sec..

Request / Response: 20 bytes

The first of the 20 request bytes defines a command to be executed. The second request byte addresses a channel and is followed by 18 parameter bytes,

/ Byte1 = command // Byte 2 = channel // Byte 3..20 = parameters /

Each command triggers a response with same data structure.

Command (Byte 1):

The following commands are defined:

Mode Read/Write
    cmd_WriteChannelModes = 'D';
    cmd_ReadChannelModes = 'E';

Status Read/Write
    cmd_WriteChannelStats = 'S';
    cmd_ReadChannelStats = 'T';

Trigger/Source select
    cmd_WriteChannelSources = 'F';
    cmd_ReadChannelSources = 'G';

Namen Read/Write
    cmd_WriteChannelname = 'N';
    cmd_ReadChannelname = 'O';

Reference voltage and ADC
    cmd_WriteVref = 'V';
    cmd_ReadVref = 'W';
    cmd_ReadADCs = 'A';

Memory functions
cmd_ConfigFromEeprom = 'X';
cmd_ConfigToEeprom = 'Y';

Channel number (Byte 2):
The channel number is located in Byte 2 of the request.
The channel assignment is as follows (decimal):

ch_ALL = 0; (All channels addressed, if supported by command)

ch_B0 = 1;
ch_B1 = 2;
ch_B2 = 3;
ch_B3 = 4;
ch_B4 = 5;
ch_B5 = 6;

ch_C0 = 7;
ch_C1 = 8;
ch_C2 = 9;
ch_C3 = 10;
ch_C4 = 11;
ch_C5 = 12;

ch_D2 = 13;
ch_D3 = 14;
ch_D4 = 15;
ch_D5 = 16;
ch_D6 = 17;
ch_D7 = 18;

Parameters (Bytes 3..20)
The 18 parameter bytes are assigned to the channels B0 to D7 one after each other.

Examples:
The first parameter byte (P1 = Byte 3 of requests) is assigned to channel B0.
The last parameter byte (P18 = Byte 20 of requests) is assigned to channel D7.
The 8th parameter byte (P8 = Byte 10 of request) is assigned to channel C1.

```
// 1 /   2   / 3/ 4/ 5/ 6/ 7/ 8/ 9/10/11/ 12/ 13/ 14/ 15/ 16/ 17/ 18/ 19/ 20//
//CMD/CHANNEL/P1/P2/P3/P4/P5/P6/P7/P8/P9/P10/P11/P12/P13/P14/P15/P16/P17/P18//
//CMD/CHANNEL/B0/B1/B2/B3/B4/B5/C0/C1/C2/ C3/ C4/ C5/ D2/ D3/ D4/ D5/ D6/ D7//
```

---

cmd_WriteChannelModes = 'D'
This command selects the desired channel mode (input, output, blink, etc.)
All 18 channels could be set in one go, or one channel can be selected individually.
(see above "channel number")

The byte values of the 18 parameter bytes (bytes 3..20 of request) specify the channel mode for each channel. Mode for channel B0 is located in first parameter byte (Byte 3 of request), while last parameter byte refers to channel D7. If a certain channel is addressed parameter bytes for other channels are ignored.

The module will echo the request as soon it received it.

The following modes are selectable with the parameter bytes:

Pm_disabled = 0
The channel is deactivated (high impedance) .

Pm_adc_input = 1
Data direction is set to input for use as ADC input channel. This mode can be used on Port C channels only!

Pm_open_input = 2
Data direction is set to input. Channel work as 5V TTL logic input.
(level floating if left open).

Pm_pullup_input = 3
Data direction is set to input an channel is pulled high with internal pullup resistor.
Input may be pulled down to ground with switch contacts or similar.

Pm_output = 4
Data direction is set to output. Output acts as general purpose logic output with 5V high level.

Pm_pwm = 5
Channel mode is set to PWM output.

Pm_servo = 6
Channel mode is set to servo output.
Only possible on Port D channels!

Pm_blink100ms = 7
Channel mode is programmed to output blink/clock signals, with intervals ranging from
1 * 100ms (0,1 sec.) to 255 * 100 ms (25,5 sec.).

Pm_blink1s = 8
Channel mode is programmed to output blink/clock signals, with intervals ranging from
1 * 1s (1 sec.) to 255 * 1s (4min. 15sec.).

Pm_pulse100ms = 9
Channel mode is programmed to output single pulses, ranging from
1 * 100ms (0,1 sec.) to 255 * 100 ms (25,5 sec.).
(Triggered by software or Port B channels)

Pm_pulse1s = 10
Channel mode is programmed to output single pulses, ranging from
1 * 1s (1 sec.) to 255 * 1s (4min. 15sec.).
(Triggerable by software or Port B channels)

Pm_pwmRampUp = 11
Output mode is set to PWM. The PWM gets modulated by a (single shot) ramp (0...100%).
(Triggered by software or Port B channels)

Pm_pwmSweepUp = 12
Output mode is set to PWM. The PWM gets modulated by a (continious) ramp (0...100%).

Pm_ServoRampUp = 13
Only available on Port D!
Output mode is set to servo. The servo position gets shifted by a (single shot) ramp (0...100%).
(Triggered by software or Port B channels)

Pm_ServoSweepUp = 14
Only available on Port D!
Output mode is set to servo. The servo position gets shifted by a (continuous) ramp (0...100%).
(Triggered by software or Port B channels)

Pm_pwmRampDown = 15
Pm_pwmSweepDown = 16
Pm_servoRampDown = 17 Only available on Port D!
Pm_servoSweepDown = 18 Only available on Port D!
Same modes as 11,12,13,14, but opposite ramp direction (down).

Pm_pwmRampUpDown = 19
Pm_pwmSweepUpDown = 20
Pm_servoRampUpDown = 21 Only available on Port D!
Pm_servoSweepUpDown= 22 Only available on Port D!
Same as above,but with alternating ramp directions (up/down).

cmd_ReadChannelModes = 'E'
This command request channel modes being set for all channels. Channel no and parameters are ignored.
Modes are returned as described under **cmd_WriteChannelModes.**

cmd_WriteChannelStats = 'S'
This command is used to change channel conditions. Depending on the channel mode the parameter bytes will affect individual channel conditions, like output state, blink interval, etc. All 18 channels could be set in one go, or one channel can be selected individually.
(see above "channel number") A response with conditions of all channels will be returned

Parameter meanings depend on channel modes as follows:

Pm_output:
Parameter byte = 0 => Output goes LOW
Parameter byte <> 0 => Output goes HIGH

Pm_blink100ms,
Pm_pulse100ms:
Parameter byte = 0 => Output goes LOW
Parameter byte = 1 ...255 => Sets the duration (Parameter * 100ms)

Pm_blink1s,
Pm_pulse1s:
Parameter byte = 0 => Output goes LOW
Parameter byte = 1 ...255 => Sets the duration (Parameter * 1s)

Pm_pwm:
Parameter byte = 0 ...255 => Sets the PWM amount (0..100%)

Pm_servo:
Parameter byte = 0 ...255 => Adjusts the servo position

Pm_xxRampxx,
Pm_xxSweepxx:
Parameter byte = 0 => Ramp off
Parameter byte = 1 ...255 => Adjusts ramp duration (1...255 s)

Pulses and ramps are triggered by software writing the duration value, but just in case a certain channel is addressed.

cmd_ReadChannelStats = 'T'
Reads condition parameters of all channels. Channel and parameter bytes are ignored.
The response contains the channel conditions as described for **cmd_WriteChannelStats.**
Conditions of input channels are reported as well:

Pm_adc_input :
Parameter byte => Byte value read by ADC (MSB)

Pm_open_input ,
Pm_pullup_input = 3
Parameter byte = 0 => Input is LOW
Parameter byte <> 0 => Input is HIGH

cmd_WriteChannelSources = 'F'
Some modes can be controlled by other channels instead of controlling these by software.
This is done by assigning a source channel to the channel being controlled, using this command.

Pm_pwm, Pm_servo:
PWM amounts and servo positions can be controlled by a ADC channel (available on Port C). The control source for a channel is selected by one of the following values, to be placed in the parameter bytes.

```
src_sw    = 0; // software controlled
src_adc0 = 1; // controlled by ADC channel
src_adc1 = 2;
src_adc2 = 3;
src_adc3 = 4;
src_adc4 = 5;
src_adc5 = 6;
```

Pm_pulse100ms, Pm_pulse1s,
Pm_pwmrampup, Pm_servorampup,
Pm_pwmrampdown,Pm_servorampdown,
Pm_pwmrampupdown, Pm_servorampupdown,

Pulses and ramps can be triggered by port B channels (falling edge). The trigger source is selected by the following values of the parameter bytes.

```
src_sw = 0;          // Software trigger
src_B0 = 1;          // triggered by falling edge on port B
src_B1 = 2;
src_B2 = 3;
src_B3 = 4;
src_B4 = 5;
src_B5 = 6;
```

cmd_ReadChannelSources = 'G'
Some modes can be controlled by other channels instead of controlling these by software.
This commands reads the current control source settings of all channels.
(refer to cmd_WriteChannelSource)

cmd_WriteVref = 'V'
Thos command selcects one of three possible voltage reference options. Reference voltage is common for all ADC channels.

```
VrefExt = 0; // supply external reference voltage
VrefVcc  = 1; // reference voltage = 5V (internal supply voltage)
Vref1V1  = 3; // reference voltage = 1,1V (internal)
```

This command does not need any channel number, so the option byte replaces the channel byte in the request.

cmd_ReadVref = 'W'
Reads the selected reference voltage option from the board.
(see cmd_WriteVref)

cmd_ReadADCs = 'A'
This commands reads the ADC raw values from Port C, whatever channel mode is set for these channels. It does not change the data direction or channel mode in any way.
The leading twelve parameter bytes of the response contain the ADC raw values (words) of the six ADC channels. The words bits are left-aligned, so bit 0..5 are not valid, while bits 6...15 represent the 10 bit ADC result.

Pairs of two bytes are combined to one word value.

First parameter byte is LByte of ADC0.
2nd parameter byte is HByte of ADC0.
3rd parameter byte is LByte von ADC1.
...
12th parameter byte is HByte of ADC5.

cmd_WriteChannelName = 'N'
Command assigns a friendly name to a channel. The channel name is a null-terminated ASCII string, the must be placed in the parameter bytes. 17 character + CHR(0) = 18 bytes.
Unused characters at the end should be filled with CHR(0).
The response echoes the request.

cmd_ReadChannelName = 'O'
Reads the channel name of a channel. A certain channel no must be specified. Parameter bytes are ignored. The response contains a null terminated ASCII string in the parameter bytes.

cmd_ConfigToEeprom = 'Y'
All configuration settings (modes, states, names, etc.) that are made with above commands are temporary stored in the modules RAM memory. For permanent storage current settings can be transferred to the modules EEPROM memory. After power-up the module recalls the settings stored permanently in the EEPROM.
The response echoes the request.

cmd_ConfigFromEeprom = 'X'
Recalls the start-up settings stored in the devices EEPROM memory (see ***cmd_ConfigToEeprom***).
The response echoes the request.

/PWMIO18 appendix

Commands (Byte 1)

```
cmd_Writechannelname = 'N';
cmd_Readchannelname = 'O';
cmd_Writechannelmodes = 'D';
cmd_Readchannelmodes = 'E';
cmd_Writechannelstats = 'S';
cmd_Readchannelstats = 'T';
cmd_WritechannelSources = 'F';
cmd_ReadchannelSources = 'G';
cmd_Writevref = 'V';
cmd_Readvref = 'W';
cmd_Readadcs = 'A';
cmd_Configfromeeprom = 'X';
cmd_Configtoeeprom = 'Y';
```

Channel address (Byte 2)

```
ch_ALL = 0;
ch_B0 = 1;
ch_B1 = 2;
ch_B2 = 3;
ch_B3 = 4;
ch_B4 = 5;
ch_B5 = 6;
ch_C0 = 7;
ch_C1 = 8;
ch_C2 = 9;
ch_C3 = 10;
ch_C4 = 11;
ch_C5 = 12;
ch_D2 = 13;
ch_D3 = 14;
ch_D4 = 15;
ch_D5 = 16;
ch_D6 = 17;
ch_D7 = 18;
```

ADC source

```
src_sw   = 0;
src_adc0 = 1;
src_adc1 = 2;
src_adc2 = 3;
src_adc3 = 4;
src_adc4 = 5;
src_adc5 = 6;
```

Trigger source

```
src_sw = 0;
src_B0 = 1;
src_B1 = 2;
src_B2 = 3;
src_B3 = 4;
src_B4 = 5;
src_B5 = 6;
```

USB serial protocol structure

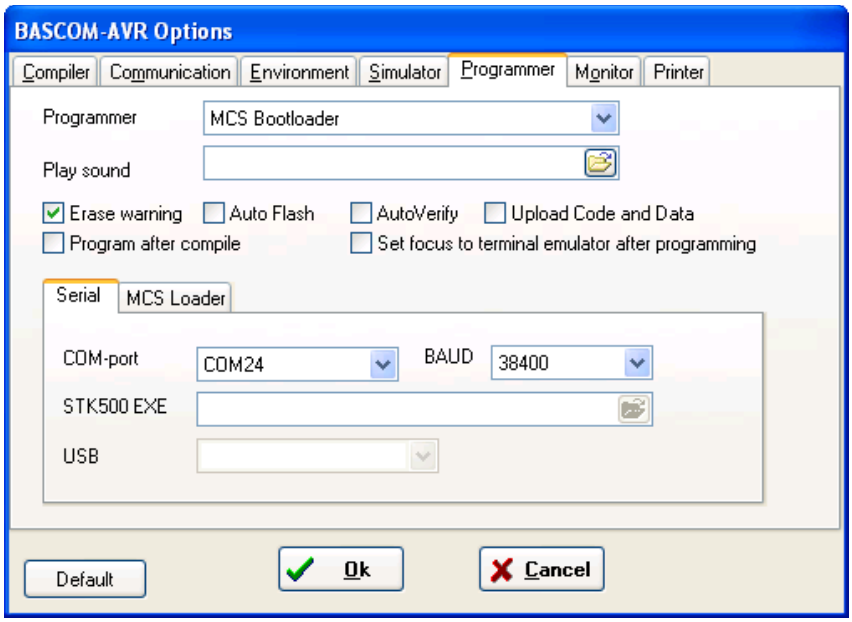
```
RequestSize = 20;
TRequest =  packed record
                Command: AnsiChar;
                ChannelNo: Byte;
                Params: array[0..17] of Byte;
            end;

ResponseSize = 20;
TResponse = TRequest;
```

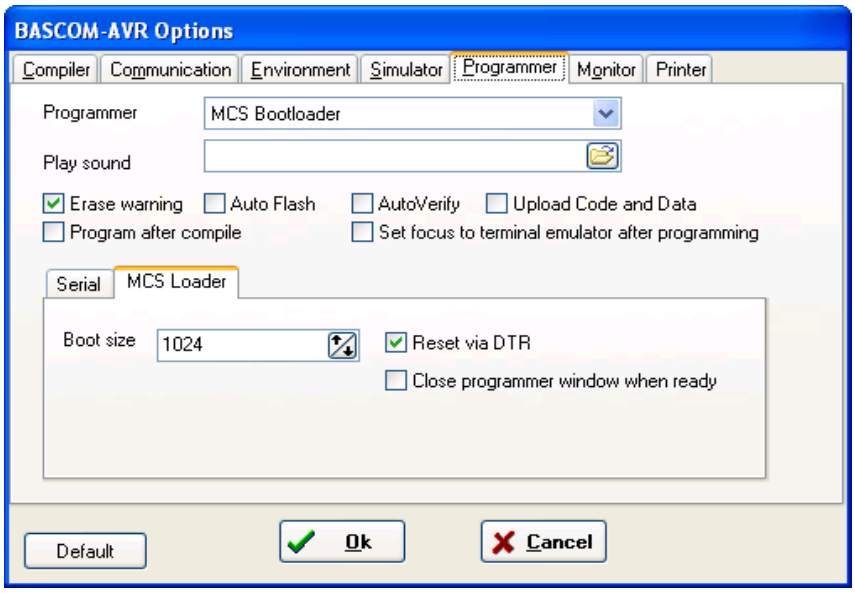
Firmware /MCS

The /MCS firmware is ready-made for use with BASCOM AVR (MCS Electronics). Applications designed with BASCOM AVR can easily be transferred to the board via USB. So you do not need any hardware programmer. The installed MCS bootloader will overtake this work. The bootloader occupies 2kB from the total 16kB flash memory.

In BASCOM AVR select the MCS bootloader (Options-Programmer).



The virtual COM port (VCP) is installed by the device driver and can be found in the Windows Device Manager. It uses 38400 Baud transfer rate. All software options like 'Reset via DTR' can be used.



Further information:
http://www.mcselec.com
AN #143 - MCS Bootloader

Firmware /CLEAR

We deliver the board equipped with a clear and empty ATMega168-20.
You will need a hardware ISP programmer to flash the board, which is not included.
The board offer a 6 pin ISP connection.

We recommend the following basic fuse setting:

Low Fuse: 0xFF
High Fuse: 0xDF
Extended Fuse: Fuse:0xF9

Programmer suggestion
The „mySmartUSB light“ made by myAVR was tested successfully with the µPIO board and can be ordered separately. A ISP cable is included and fits to out board.