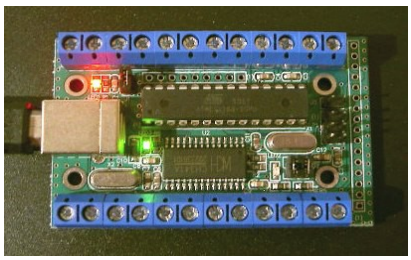
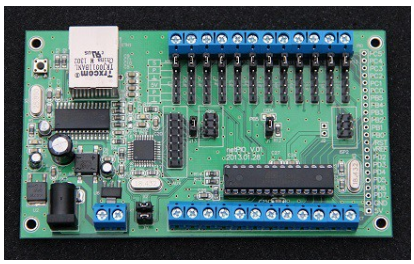
ABACOM USB-µPIO und netPIO-Boards





Diese Boards bilden die Basis für eine preiswerte Produktreihe, die wir für verschiedene, häufig vorkommende Mess- und Steuerungsaufgaben entwickelt haben. Sie haben somit die Wahl zwischen einem Gerät für den USB- oder Netzwerkanschluss.

Folgende Varianten sind derzeit erhältlich:

/ TEMP12
Temperatursystem für bis zu 12x Dallas DS18B20 Temperatur-Sensoren

/ GPIO18
Interface mit 18 programmierbaren I/O
Sie verwenden das Board als I/O-Interface z.B. mit unserer ProfiLab-Software oder eigenen Programmen.

/ PWMIO18
Komfortables Interface mit 18 programmierbaren I/O
mit erweiterten Funktionen, wie PWM, Servo, Blink-/ Pulsfunktionen.
Sie verwenden das Board als I/O-Interface z.B. mit unserer ProfiLab-Software oder eigenen Programmen.

/ FREQ
9-MHz.Frequenzzähler / Generator
Sie verwenden das Board als Frequenzzähler z.B. mit unserer ProfiLab-Software oder eigenen Programmen.

/ INCR3
Drei Inkrementalgeber-Eingänge
Sie verwenden das Board mit bis zu drei Inkrementalgebern z.B. mit unserer ProfiLab-Software oder eigenen Programmen.

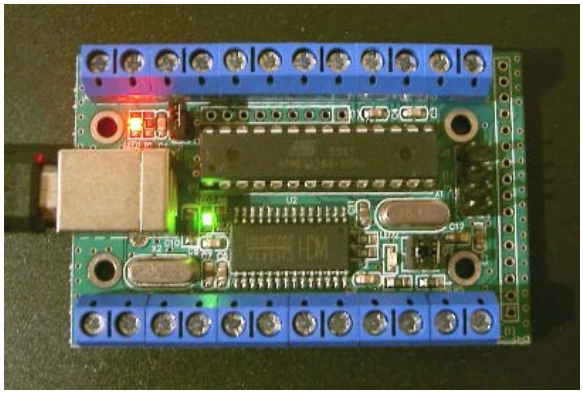
Für Eigenentwicklungen bieten wir Ihnen unsere Boards auch ohne fertige Firmware an:

/MCS
incl. MCS BOOTLOADER für BASCOM AVR (nur USB)
Sie verwenden BASCOM-AVR für die Programmierung eigener Anwendungen. Durch die Verwendung des MCS-Bootloaders benötigen KEINEN ISP-Programmer. Ihre Anwendung wird einfach per USB direkt auf den Chip übertragen.

/CLEAR
Nur CHIP - unprogrammiert
Sie haben Erfahrung mit der Programmierung des Atmel ATmega168 und programmieren / flashen Ihre Firmware selbst. Für die Programmierung ist ein ISP-Programmer erforderlich.

Zum vollständigen Verständnis der Funktionsweise Ihres Gerätes lesen Sie bitte sowohl die Beschreibung der Hardware (USB oder Netzwerk) und das passende Kapitel zur Firmware mit der Sie das Boards erworben haben.

USB-µPIO



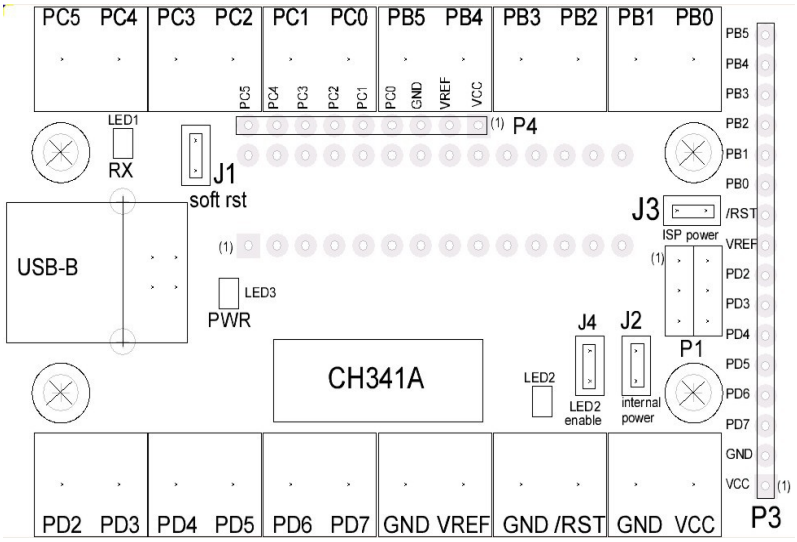
USB-AVR-Board

- Kompaktes AVR-Board mit Atmel ATmega168-20
- High Speed Taktfrequenz 18.432000 MHz
- 100% error free High Baudrates
- Komfortabler Anschluss über Schraubklemmen oder Stiftleisten
- 6-poliger ISP-Anschluss
- Betriebsspannung (+5V) wahlweise vom USB oder extern
- 3 LED: USB-Power, COM (RxD) , USER (Port B.5; deaktivierbar)
- SOFT-RESET per DTR-Signal (deaktivierbar)
- Standard USB-B-Anschluss
- USB Chipsatz: CH341A
- Virtueller (RS232-)-COM-Treiber (VCP)
- Systemvoraussetzungen: XP, 2000, Vista, WIN7, 32/64 Bit
- Abmessungen ca. 70 x 45 x 15 mm

Download

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

Anschlüsse, LED und Jumper



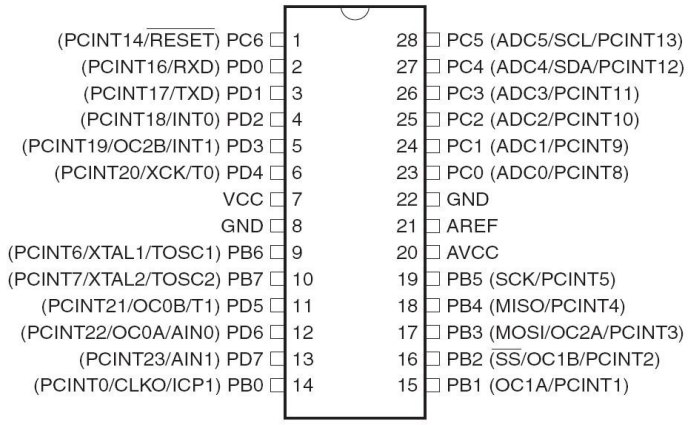
P1 – ISP-Anschluss für Programmer
P2 – USB-B-Anschluss
P3 – Anschlussleiste Port B, Port D, VREF, RST, VCC, GND (Digitalport)
P4 – Anschlussleiste Port C, VREF, VCC, GND (Digitalport und ADC-Analogport)

J1 – Jumper SOFT RST verbindet RESET(PC6) mit DTR-Signal vom CH341A
J2 – Jumper INTERNAL POWER verbindet VCC mit USB-Versorgungspannung
J3 – Jumper ISP POWER verbindet VCC mit ISP
J4 – Jumper LED2 ENABLE verbindet LED2 mit PB5

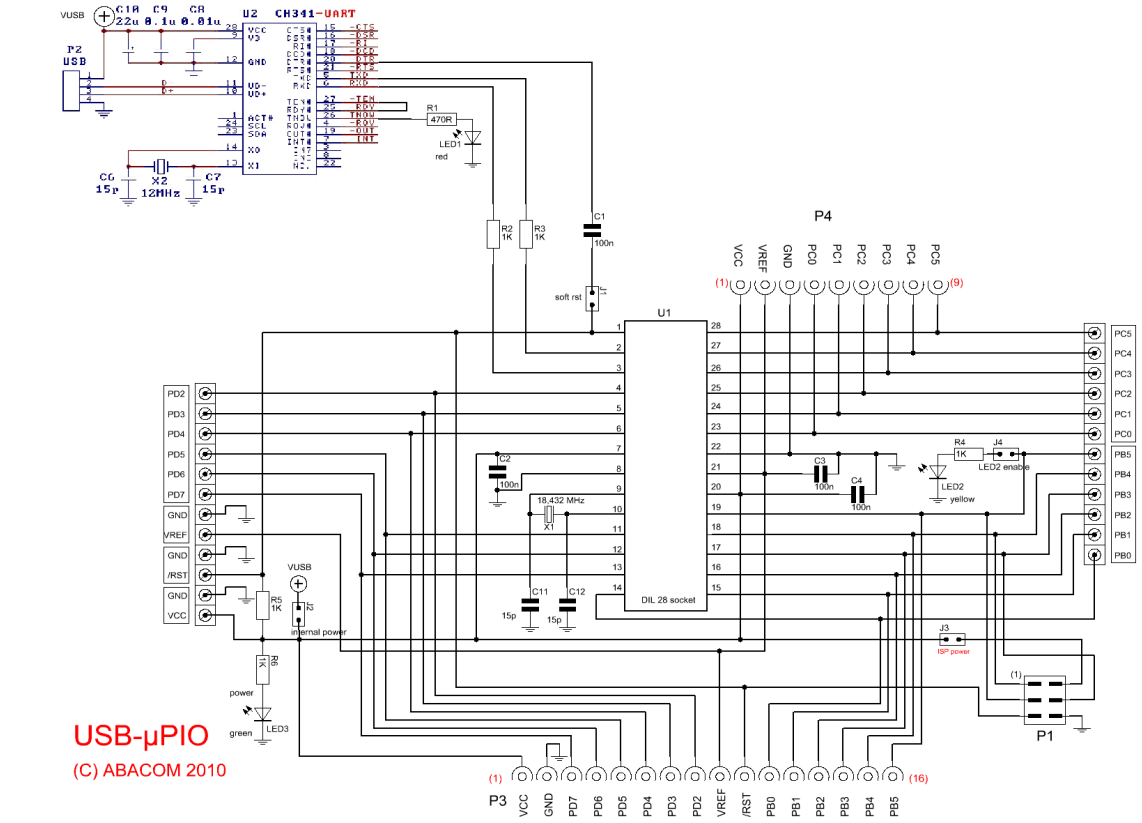
LED1 – RX Indikator (rot; Datenempfang vom PC)
LED2 – LED an PB5 (gelb; für eigene Zwecke)
LED3 – USB-Power (grün; USB-Spannungsversorgung)

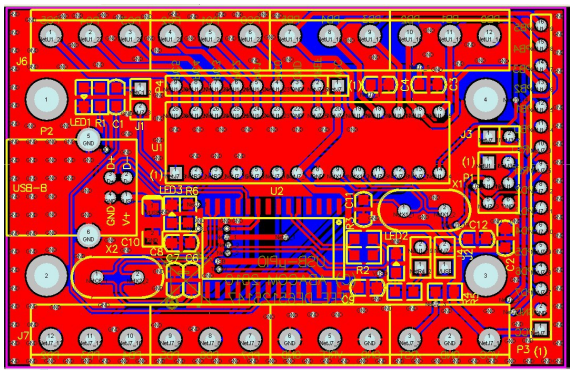
Schraubklemmen für Port B, Port C und Port D

Die Anschlussbezeichnungen ATmega168



Schaltung und Abmessungen

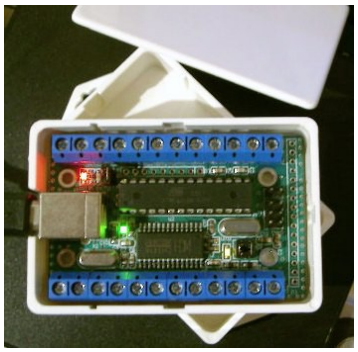




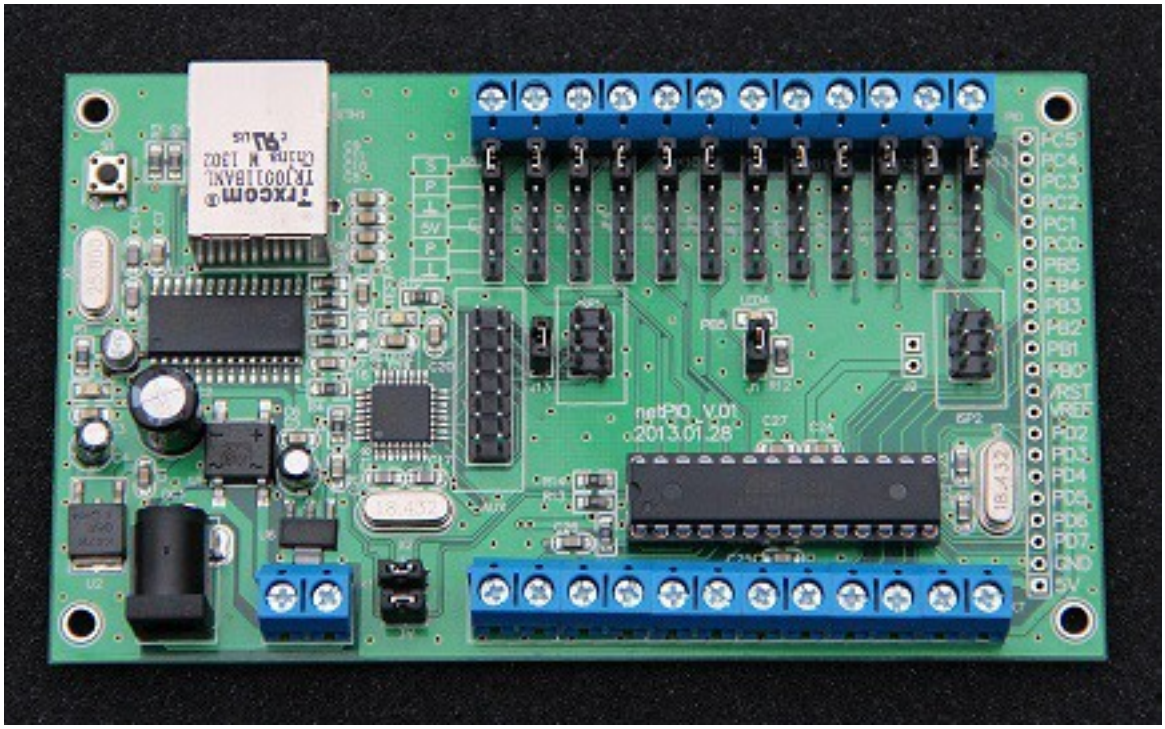
Abmessungen ca. 70 x 45 x 15 mm

Als Zubehör erhältlich:

- Modulgehäuse (unbearbeitet)
- USB-Programmer inkl. ISP-Kabel



netPIO

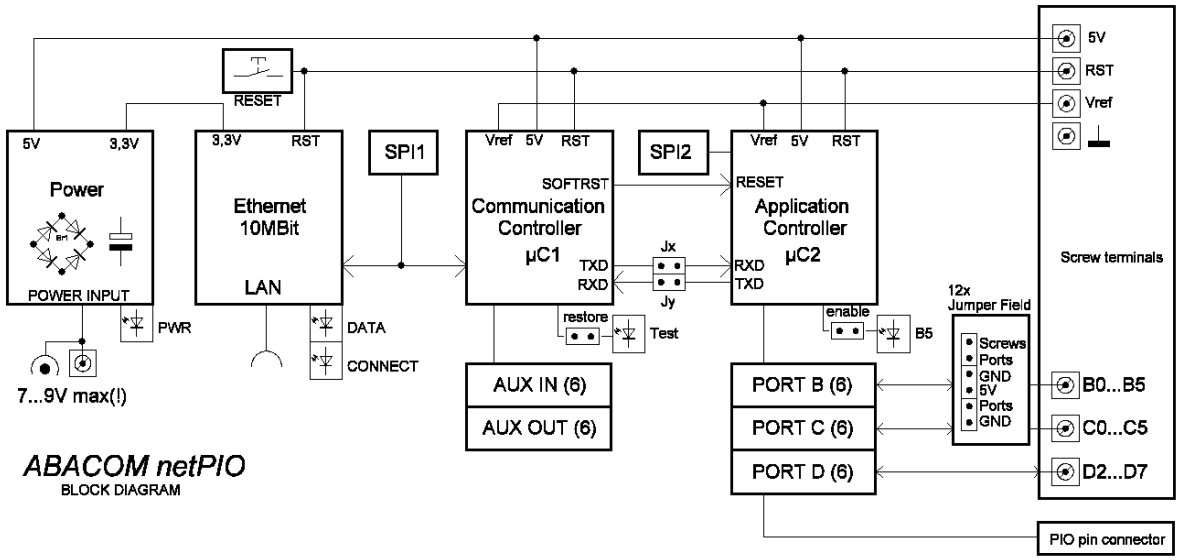


Download

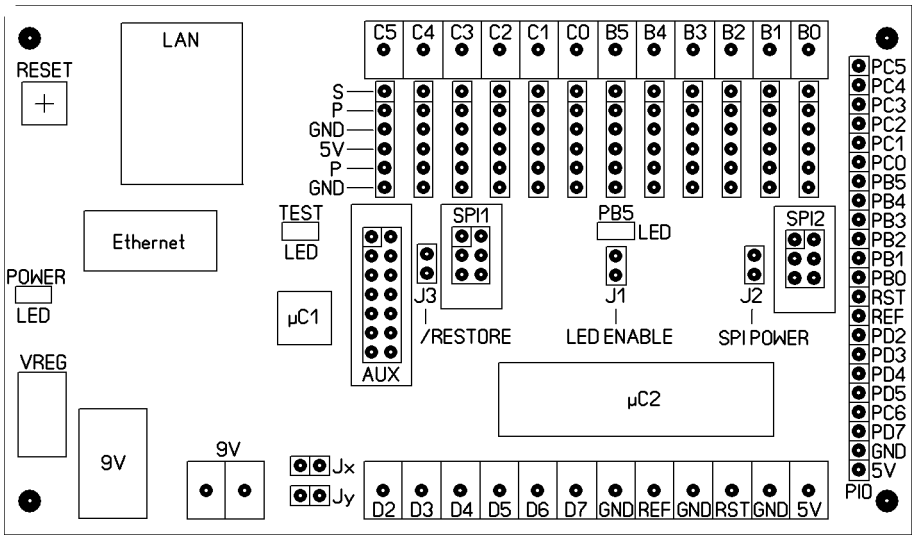
http://www.abacom-online.de/div/setup_netPIO.exe

Das ABACOM netPIO-Bord ist ein 10-MBit-LAN-Netzwerkinterface mit der sich verschiedene Mess- und Steuerungsaufgaben durchführen lassen. Das Bord ist mit verschiedenen Firmware-Varianten erhältlich, die auch auf unseren USB-µPIO-Modulen zum Einsatz kommen. Dieses Kapitel beschreibt lediglich die Unterschiede und zusätzlichen Funktionen der Netzwerk-Variante. Informationen zur Verwendung der Port-Schraubanschlüsse finden Sie im Kapitel zu Ihrer Firmware.

Blockschaltbild



Anschlüsse, LED und Jumper



Bordgrösse 70mm x 120 mm (passt in WAGO-Hutschienenprofil WAGO 288-600)

Spannungsversorgung

Das Modul benötigt eine externe Versorgungsspannung von 7-9V, die über eine übliche DC-Steckverbindung oder alternativ über einen Klemmenanschluss zugeführt werden kann.
Das Netzteil sollte wenigstens 500 mA liefern können. Die Ruhestromausnahme liegt bei typ. 200mA (ohne externe Beschaltung). Das Board besitzt eine eigene Spannungsstabilisierung (5V) mit vor geschaltetem Brückengleichrichter (Verpolungsschutz).
Die Verwendung stabilisierter DC-Netzteile ist zu empfehlen. Die Versorgungsspannung darf 9V nicht übersteigen, damit es nicht zu einer übermässigen Erwärmung kommt. Externe Komponenten dürfen in der Summe das Board maximal mit 200mA belasten. Die Stromaufnahme pro Port darf 10mA nicht übersteigen. Sobald das Board versorgt wird leuchtet die grüne POWER-LED. Das sich die Platine - insbesondere der Spannungsregler (VREG) - erwärmt ist normal.

Wenn anderweitig eine externe, STABILISIERTE 5V-Spannungsvervorgung zur Verfügung steht, kann das Bord alternativ auch über die Anschlussklemmen GND/VCC(5V) versorgt werden. Ein Verpolungsschutz besteht hier nicht. Diese Methode ist vorteilhaft, denn sie vermeidet die Erwärmung, die sonst durch die Verlustleistung der Spannungsregelung entsteht.

Doppelcontroller

Das netPIO verwendet zwei Mikrocontroller die nachfolgend als Kommunikationscontroller (µC1) und Anwendungscontroller (µC2) bezeichnet werden.

Netzwerkanschluss und Kommunikationscontroller

Für den 10MBit-Netzwerkanschluss wird das Gerät mit Ihrem Router verbunden. Die grüne LED im LAN-Anschluss signalisiert den physikalisch korrekten Netzwerkanschluss. Eine gelbe LED im Anschluss signalisiert eingehende und ausgehende Datenpakete.
Als Interface-Chip dient ein MicroChip ENC28J70 Ethernet-Controller, der intern per SPI-Schnittstelle (SPI1) mit einem ATMEGA168-20 Microcontroller (µC1 - Communication Controller) verbunden ist. Dieser Kommunikationskontroller steuert die Netzwerkkommunikation. Für die Einstellung der Netzwerkparameter aller ABACOM netPIO Produkte steht eine gemeinsame einheitliche Windows-Konfigurationssoftware zur Verfügung. Der Kommunikationskontroller sendet und empfängt Netzwerk-Datenpakete und bereitet diese auf. Die Kommunikation erfolgt im wesentlichen über das UDP-Netzwerk-Protokoll.

Eine gelbe TEST-LED erleichtert die Geräte-Identifikation und Diagnose der Netzwerkeinstellungen in Verbindung mit der Konfigurationssoftware. Darüber hinaus stellt der Netzwerkkontroller einen digitalen IO-Port (AUX) mit jeweils sechs Ein- und Ausgängen zur Verfügung. Die genaue Funktion des AUX-Port wird später noch erläutert.

Die Schnittstelle SPI2 des Anwendungscontrollers dient (ebenso wie die SPI1 des Kommunikationscontrollers) zur hauseigenen Programmierung der Controller-Chip-Firmware. Eine Verwendungsmöglichkeit für den Endanwender besteht nicht.

Anwendungscontroller (Application Controller)

Ein zweiter Mikrocontroller (µC2) - der sogenannte Anwendungscontroller - verfügt über drei I/O-Port B, C und D mit jeweils sechs I/O-Anschlüssen, über die das Modul die ihm zugedachte Mess- und Steuerungsanwendung ausführt.

Die Datenkommunikation mit dem Kommunikationscontroller erfolgt intern über die seriellen Schnittstellenleitungen der beiden Chips mit 57600 Baud. Die interne serielle Verbindung ist über die Jumper Jx und Jy zugänglich. Als Anwendungscontroller kommt ebenfalls ein ATMEGA168-20 (hier in gesockelter Ausführung) zum Einsatz. Beide Controller verwenden eine Taktfrequenz von 18.432 MHz.

Die Verwendung von zwei Controllern ermöglicht es die aufwändige Netzwerkkommunikation vollständig von der (nicht minder aufwendigen) Steuerungsaufgabe weitestgehend zu trennen. Der Anwendungscontroller tauscht mit dem Kommunikationscontroller nur noch die relevanten Nutzdaten (intern per serieller Schnittstelle) aus und ist von der Datenaufbereitung der Netzwerkdatenpakete und der Netzwerkprotokolle völlig entlastet. Der Anwendungscontroller auf dem netPIO-Board verwendet exakt die gleiche Firmware, die auch auf unseren bewährten USB-µPIO-Bords zum Einsatz kommt. Die Verwendung der Ports der Port ist ebenfalls somit ebenfalls identisch und der Beschreibung Ihrer Firmware zu entnehmen.

I/O-Ports

Folgende I/O-Leitungen des Anwendungscontrollers stehen zur Verfügung:

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

Über die Funktion der Signalleitungen entscheidet die Firmware des Anwendungscontrollers mit der Sie das netPIO-Bord bezogen haben. Alle Portleitungen arbeiten mit TTL-Pegeln, d.h. die Spannungen an den Portleitungen dürfen die Betriebspannung (+5V) nicht übersteigen und das Massepotential (0V; GND) keinesfalls unterschreiten.

Alle genannten Portleitungen stehen direkt an der PIO-Anschluss-Reihe zur Verfügung,
die es dem Anwender ermöglicht wahlweise Stift- oder Buchsenleisten, Drähte oder andere Komponenten einzulöten. Port D steht zusätzlich DIREKT an den entsprechenden Schraubklemmen zu Verfügung. Port B und Port C verfügen darüber hinaus jeweils noch über eine Stiftleistenreihe (Jumper Field).

Die Verwendung der Stiftleiten erleichtert in vielen Fällen die externe Beschaltung. Die Stiftleisten sind beginnend an der Schraubklemme in dieser Reihenfolge belegt:

1.) S =Schraube (Screw) ist mit der zugehörigen Schraubklemme verbunden.
2.) P - Port ist direkt mit der zugehörigen I/O-Portleitung verbunden.
3.) GND - Masse (Ground) ist ein Masseanschluss.
4.) +5V ist mit der 5V-Versorgungsspannung des Bords verbunden.
5.) P - Port ist wiederum die zugehörige I/O-Portleitung des Anwendungscontrollers.
6.) GND - Masse (Ground) Ist wiederum ein Masseanschluss.

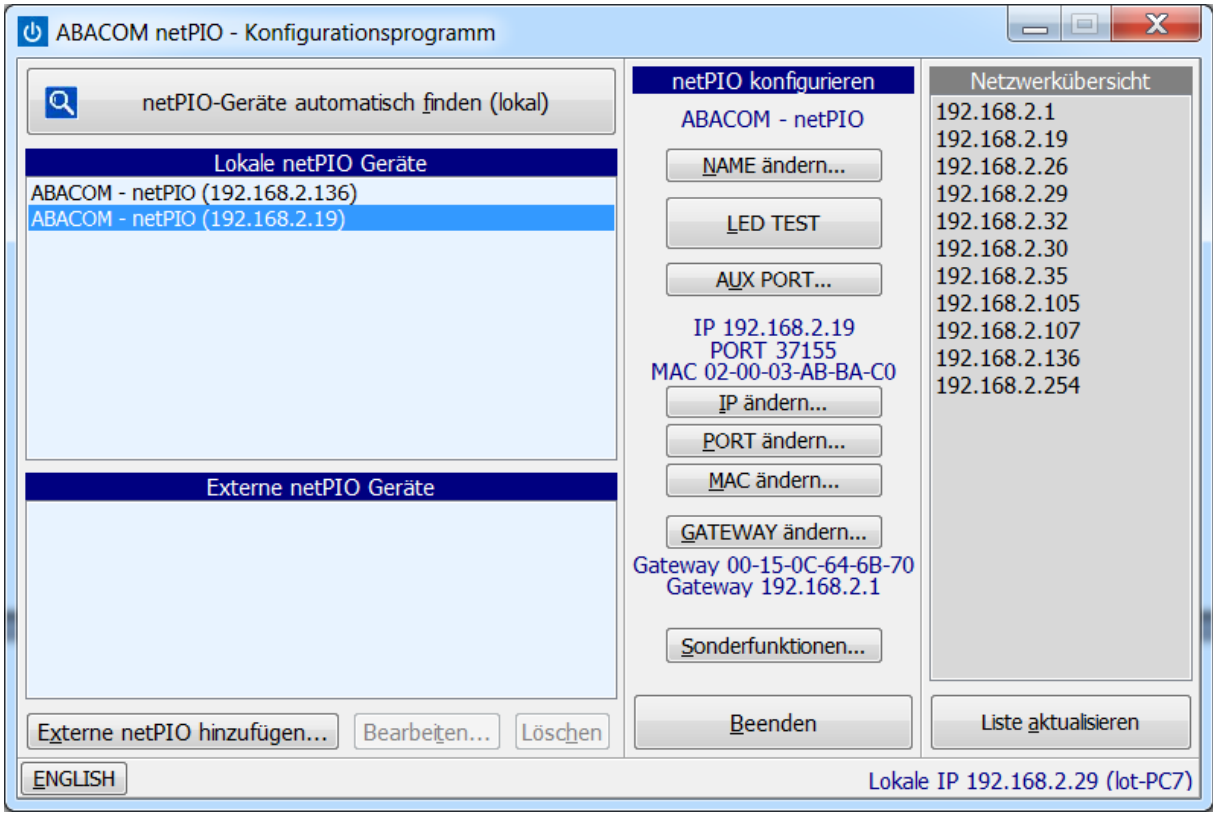
Achten Sie beim Anschluss externer Komponenten immer darauf, das Sie die Belastungsgrenzen einhalten:
- max. 10 mA pro Port
- in der Summe nicht mehr als 200mA externe Last
Externe Beschaltungen mit grösserem Strombedarf müssen extern selbst versorgt werden!

Die Anwendungen für die Stiftleisten sind vielfältig:
Beispiele 1: Jumper von S nach P
S und P sind mit einem Jumper verbunden (Werkseinstellung). Die Schraubklemme ist nun mit der Portleitung verbunden.
Beispiel 2: Jumper von P nach GND
Eine Eingangs(!)leitung wird definiert auf Masse-Potential (low) gelegt.
Beispiel 3: Jumper von +5V nach P
Eine Eingangs(!)leitung wird definiert auf Plus-Potential (high) gelegt.
Beispiel 4: Ein steckbarer Sensor mit der Anschlussreihenfolge 5V - Data - GND kann einfach auf die Pins 4,5 und 6 der Stiftleite gesteckt werden.
Beispiel 5: Ein eigene Eingangsbeschaltung (z.B. Überspannungsschutz, Signalverstärker, Abschwächer, o.ä.) kann zwischen Schraubklemme (Pin 1) und Port (Pin 2) eingeschleift werden, ohne den Schraubanschluss lösen zu müssen.
Beispiel 6: Eine eigene, selbstentworfene Huckepackplatine könnte aufgesteckt werden.

Netzwerkkonfiguration

Verbinden Sie den LAN-Anschluss mit Ihrem Router und versogen Sie das Bord mit der Betriebspannung. Der Router muss die Datenrate 10MBit unterstützen, was bei fast allen marktgängigen Geräten der Fall ist. Die POWER-LED (grün) und die CONNECTION-LED (grün) am LAN-Anschluss leuchten auf, wenn der Anschluss korrekt erfolgt ist. Die gelbe LED am LAN-Anschluss zeigt sofern vorhanden Datenpakete vom Netzwerk an.

Schliessen Sie zunächst immer nur EIN neues netPIO Gerät an Ihr Netzwerk. Weitere dürfen erst folgen wenn die Netzwerkkonfiguration für das neue Gerät erfolgreich abgeschlossen wurde.



Die Netzwerkkonfiguration erfolgt für alle ABACOM netPIO-Produkte in gleicher Weise, mit Hilfe des Konfigurationsprogramms (netPIO_config.exe). Das Konfigurationsprogramm muss auf einem Windows-PC ausgeführt werden, welcher sich im gleichen Netzwerk wie das neue netPIO-Gerät befindet.

Das Konfigurationsprogramm erkennt und verwaltet die Netzwerkkonfiguration aller im lokalen Netzwerk vorhandenen und neuen netPIO-Geräte. Die Erkennung erfolgt automatisch bei Programmstart oder einen Klick auf die entsprechende Schaltfläche.

Neue Geräte die sich im Auslieferungszustand befinden werden mit einer freien IP in Ihrem Netzwerk angemeldet. Das neue Geräte erscheint daraufhin in der Liste der lokalen netPIO-Geräte. Die verwendete Standard-Portadresse ist 37155 (UDP-Port).

Das Konfigurationsprogramm erfüllt folgende Aufgaben:

- Einstellung der Netzwerkparameter (IP, Port, Gateway, ggf. Ändern der MAC-Adresse)
- Vergabe benutzerfreundlicher Gerätenamen
- Status-Anzeige des AUX-Ports
- Master-Slave-Konfiguration (Traps)
- Aufruf weiterer Sonderfunktionen

Mit Ausnahme der Netzwerkparameter können auch Einstellungen an entfernt stehenden Geräten z.B. per Internet vorgenommen werden, wenn für diese ein entsprechender Zugang zuvor eingerichtet wurde.

Sofern erforderlich oder gewünscht können Sie nun die IP-Adresse und die Portnummer nach Bedarf ändern. Markieren Sie dazu das Gerät in der Liste der lokalen netPIO-Geräte, und betätigen Sie dann die gewünschte Konfigurationsschaltfläche (z.B. "IP ändern").

Die Schaltfläche LED TEST schaltet die TEST-LED auf dem Modul ein und aus. So können Sie leicht die Netzwerkfunktion und Zuordnung überprüfen.

netPIO-Geräte verwenden eine feste IP-Adresse (kein DHCP). Um Sie bei der Vergabe einer neuen IP zu unterstützen, zeigt das Konfigurationsprogramm eine Liste IP-Geräten die z.Zt. im Netzwerk aktiv sind.

Sobald Sie ein neues Gerät erfolgreich mit einer IP im Netzwerk angemeldet haben, können Sie ggf. nun weitere Geräte - eins nach dem andern - anschliessen und wie zuvor beschrieben konfigurieren.

Die Vergabe eines Gerätenamens dient der Benutzerfreundlichkeit und hat keinen Einfluss auf die Funktionalität der Geräte. Die Schaltfläche "NAMEN ändern" erlaubt die Vergabe eigener Gerätenamen, wie z.B. "Büro #1", "Garage", "Heizung", etc.

MAC-Adresse und Port müssen nur in (seltenen) Fällen verändert werden, wenn Netzwerkkonflikte bestehen.

Die Gateway-Adresse ist die lokale IP-Adresse Ihres Routers (z.B. 192.168.1.1).
Diese Adresse kann bei Bedarf umkonfiguriert werden.

Alle Konfigurationseinstellungen werden im EEPROM-Speicher des Kommunikationskontrollers abgelegt und dauerhaft gespeichert. Sollte es in Ausnahmefällen passieren das ein Gerät nicht mehr reagiert oder gefunden wird, betätigen Sie den RESET-Taster am Gerät. Danach sollte das Gerät von der Konfigurationssoftware wieder erkannt werden. Die Konfigurationsdaten bleiben dabei erhalten.

In Fällen in denen diese Vorgehensweise nicht zum Erfolg führt und andere Fehlerquellen auszuschliessen, wurde vermutlich eine falsche Netzwerkkonfiguration vorgenommen.
In diesem Fall muss das Gerät in den Auslieferungszustand zurück gesetzt werden und die Konfiguration wie bei einem Neugerät erneut vorgenommen werden.

Um den Auslieferungszustand wieder herzustellen...

- entfernen Sie den Jumper J3 (/Restore; zwischen AUX-Port und SPI1)

- Betätigen Sie nun die RESET-Taste am Gerät (Spannungsversorgung muss vorhanden sein)
- anschliessend stecken Sie den Jumper J3 wieder auf
- abschliessend betätigen nochmals den RESET-Taster

Das Gerät befindet sich nun wieder im Auslieferungszustand. Führen Sie nun die Netzwerk-Konfiguration wie zuvor für ein Neugerät beschrieben erneut durch.

Liste externer netPIO-Geräte
In dieser Liste können netPIO-Geräte manuell mit Hostnamen (oder IP) und Port angemeldet werden, die sich ausserhalb Ihres lokalen Netzwerks befinden, also z.B. Geräte für die Sie einen Fernzugriff per Internet konfiguriert haben. Die Netzwerkkonfiguration dieser Geräte muss zuvor lokal vorgenommen werden.

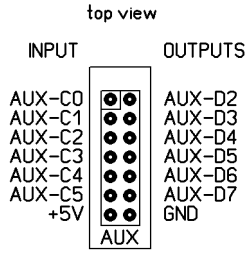
Für externe Geräte sind einige Funktionen der Konfigurationssoftware gesperrt, um z.B. das Verändern der IP-Adresse oder der Porteinstelleung per Fernzugriff zu verhindern. (Nach einer solchen Änderung wäre das Gerät sonst nicht mehr per Fernzugriff erreichbar.)

AUX-Port

Am Anschluss AUX aller netPIO-Geräte stehen Ihnen zusätzlich sechs digitale Eingänge und sechs digitale Ausgänge zur Verfügung. Diese sind unmittelbar mit dem (fest verbauten) Kommunikationskontroller verbunden. Ausserdem steht hier die 5V-Betriebsspannung und ein Masseanschluss zur Verfügung. Bei der Verwendung dieser Anschlüsse ist daher besondere Sorgfalt geboten. Fehlerhafte Beschaltungen, sowie Über- und Unterspannungen können den Controller beschädigen, der nicht ausgetauscht werden kann.

Alle Portleitungen arbeiten auch hier mit TTL-Pegeln, d.h. die Spannungen an den Portleitungen dürfen die Betriebspannung (+5V) nicht übersteigen und das Massepotential (0V; GND) keinesfalls unterschreiten. Bitte verwechseln Sie die AUX-Kanäle nicht mit den Ports des Anwendungscontrollers (z.B. "AUX-D2" ist NICHT GLEICH "D2")

Die Pin-Belegung des AUX-Port



- Ansicht von der Bauteilseite
- Links die Eingänge
- Rechts die Ausgänge

Offene Eingänge des AUX-Port sind intern über Pullup-Widerstände auf HIGH-Potential gelegt, und können extern z.B. über Taster oder Open-Collector-Ausgänge bedient (d.h. nach Masse gezogen) werden. Ausgänge des AUX-Port können in der Konfigurationssoftware ein-, aus- und umgeschaltet werden. Die Signalzustände der Ein- und Ausgange werden in der Konfigurationssoftware angezeigt.

Somit bietet der AUX-Port auf den ersten Blick lediglich weitere (Hilfs-) I/O-Leitungen an, die unabhängig von der Firmware des Anwendungscontrollers zur Verfügung stehen.

Doch der AUX-Port hält noch eine weiteres Leistungsmerkmal der netPIO bereit...

Signalgesteuerte Botschaften (Traps)

Der Netzwerkkontroller erkennt Änderungen am Signalzustand der AUX-Signale und der TEST-LED und sendet ggf. per Netzwerk eine Benachrichtigungsbotschaft an ein anderes netPIO-Gerät oder auch sich selbst. Der Empfänger dieser Botschaft führt daraufhin einen Schaltvorgang an einem seiner Signalausgänge oder der TEST-LED aus.

Das Gerät welches die Nachricht aussendet wird als MASTER bezeichnet. Das Gerät das die Botschaft empfängt und ausführt wird als SLAVE bezeichnet. Derartige Benachrichtigungsbotschaften, die ohne vorherige Anforderung ins Netzwerk geschickt werden, bezeichnet man als Traps (Fallen).

Jedes netPIO-Gerät erlaubt es maximal sechs verschiedene Traps (Botschaften) zu definieren und zu speichern.

Das Senden einer Trap (durch den Master) wird durch einen Zustandswechsel (steigende UND fallende Flanke) eines Triggersignals ausgelöst. Als Triggersignal können alle Signale des AUX-Ports und die TEST-LED verwendet werden. D.h. eine Triggerung kann z.B. durch ein Eingangssignal (z.B. einen Taster) ausgelöst werden. Ebenso kann aber auch das Umschalten eines Ausgangssignal eine Nachricht auslösen.

Die Trap-Nachricht die versendet wird ist immer an einen bestimmten Empfänger (Slave) gerichtet. Der Slave der die Nachricht erhalten soll (also irgendein netPIO-Gerät das per Netzwerk erreichbar ist) wird durch seine Netzwerkadresse (IP oder Hostname und Portadresse) adressiert. Die Adressierung lokal direkt über die Geräte-IP erfolgen. Es können aber auch Geräte per Hostnamen (z.B. per DynDNS-Hostnamen) angesprochen werden.

Die Nachricht enthält ausserdem noch die Anweisung für den Slave, die dieser bei Erhalte der Nachricht ausführen soll. Mögliche Anweisungen an den Slave sind das EIN-, AUS- oder Umschalten eines seiner AUX-Ausgänge und die Anweisung dem Zustand des Triggersignals zu FOLGEN.

Beispiel 1:
Gerät #1 erkennt den Zustandswechsel am AUX-Eingang C0 (z.B. ein Taster).
Gerät #1 sendet eine Nachricht an Gerät #2 mit der Anweisung "Schalte deinen AUX-AUSGANG D0 UM!"
Gerät #2 empfängt die Anweisung und führt diese aus (z.B. Umschalten eines Relais).

Beispiel 2:
Gerät #1 erkennt das Einschalten seiner TEST-LED.
Gerät #1 sendet eine Nachricht an Gerät #2 mit der Anweisung "FOLGE MIR MIT DEINER TEST-LED!"
Gerät #2 empfängt die Anweisung und schaltet seinerseits seine TEST-LED ein.

Dem aufmerksamen Leser werden die umfangreichen Möglichkeiten dieser Technik nicht entgangen sein, welche es ermöglicht Signale direkt von einem netPIO-Modul auf eine anderes zu übermitteln, ohne das dazu ein PC als Vermittler bemüht werden muss.

Allerdings besteht auch die mögliche Gefahr, dass durch fehlerhafte Konfiguration Nachrichten-

Schleifen (Rückkopplungen) entstehen, die dann dazu führen das das Netzwerk mit Nachrichten überflutet wird, die sich gegenseitig immer wieder neu auslösen.
Planen Sie Ihre Traps also sorgfältig!

Ein Beispiel was man BESSER NICHT machen sollte:

Gerät #1 erkennt den Zustandswechsel seiner TEST-LED
Gerät #1 sendet an Gerät #2 die Nachricht "SCHALTE DEINE TEST-LED UM!"
Gerät #2 erkennt den Zustandswechsel seiner TEST-LED
Gerät #2 sendet an Gerät #1 die Nachricht "SCHALTE DEINE TEST-LED UM!"
Gerät #1 erkennt den Zustandswechsel seiner TEST-LED
usw.

Einmal angestossen benachrichtigen sich hier zwei Geräte immer aufs Neue, bis der Netzwerkstecker gezogen wird!

Sonderfunktionen

Abschliessend ermöglicht die Konfigurationssoftware noch eine der folgenden Sonderfunktionen durchzuführen:

- RESET des Moduls per Software (entspricht dem Drücken der RESET-Taste am Gerät)
- RESET des Anwendungskontrollers
- WERKSEINSTELLUNG WIEDERHERSTELLEN, löscht alle Konfigurationsdaten aus dem EEPROM des Gerätes und versetzt das Gerät zurück in den Auslieferungszustand.

Firmware der Anwendungscontrollers

Das Board wird fertig programmiert in verschiedenen Firmware-Verianten angeboten,
welche auch auf unseren USB-Modulen (ABACOM USB-µPIO) zum Einsatz kommt.
Eine detaillierte Beschreibung der Firmware und die dazugehörige Funktion und die Belegung der I/O-Port finden Sie daher im Datenblatt der USB-Module.
Die Firmware und die Ports der netPIO und der USB-µPIO sind **identisch**.

Programmierung

Kommunikation mit dem Anwendungscontroller

Bei der Programmierung kommen die gleichen Protokolle zum Einsatz, die im Kapitel der jeweiligen Firmware beschrieben sind.

Die netPIO **kaspselt** die seriellen Protokolle der USB-Variante, und überträgt diese per UDP-Netzwerktprotokoll (User Datagram Protokoll). UDP ist ein einfaches (verbindungsloses) Netzwwerkprotokoll mit dem kleine Datenpakete (Datagramme) zwischen Netzwerkteilnehmern ausgetauscht werden können.

Das Prinzip der Frage-Antwort-Übertragung (Request-Response) der USB-Module bleibt dabei erhalten.

Programme die für ein netPIO entwickelt werden sollen, müssen die also die Requests in einem UDP-Datenpaket ablegen und übertragen. Das Gerät sendet daraufhin ein UDP-Antwortpaket zurück, welches die Response-Daten enthält.

Request per UDP

Bei der Kapselung eines Requests (der USB-Variante) wird dieser mit einigen, wenigen Zusatzdaten im Datagramm des UDP-Pakets abgelegt:

- Dem ursprünglichen Request wird die ASCII-Zeichenfolge "TXDATA" als Steuerkommando vorangestellt, welche dem Kommunikationskontroller mitteilt, den eigentlichen Request (intern per UART) an den Anwendungscontroller weiterzureichen.

- Zusätzlich werden noch zwei weitere Bytes an das Datenpaket angehängt. Ein Byte für die Länge der erwarteten Response und ein SYNC-Byte.

Das Datagramm das im UDP-Datenpaket übertragen werden muss ist also immer um acht Bytes länger als der ursprüngliche, gekapselte Request, nämlich 6 Bytes für die Zeichenfolge "TXDATA" und zwei Byte für die Antwortlänge und das SYNC-Byte.

<"TXDATA"(6 Bytes)> + <Request(n Bytes)> + <Antwortlänge(1 Byte)> + <Sync(1 Byte) >

Response per UDP

Nachdem das UDP-Paket verschickt wurde antwortet der Controller mit einem UDP-Datenpaket, welches die Response enthält. Das Antwort-Datagramm hat den gleichen Aufbau wie die Response der USB-Variante. Bei der Antwort ist keine Kapselung notwendig.

Hinweise zur Netzwerkprogrammierung

Zum Versenden muss das UDP-Datenpaket noch mit der Empfänger-IP und dem verwendeten UDP-Port adressiert werden. Je nach Programmiersprache wird man dabei auf fertige UDP-Komponenten zurückgreifen, die den Versand und Empfang der UDP-Pakete übernehmen. Die bekanntesten sind vermutlich die Indy-Netwerkkomponenten.

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

Bitte haben Sie Verständnis, dass wir keine Fragen zu UDP-Komponenten (wie. z.B. Indy) oder zu den vielfältigen Programmiersprachen, Versionen und Dialekten beantworten können.

Fragen wie z.B. "Wie verschicke ich ein UDP-Paket mit Visual Basic 6.0?" oder "Wie baue ich ein Request-Datenpaket in Java?" werden auch wir Ihnen nicht beantworten können.
Hobby-Programmierer finden Antworten auf derartige Fragen bitte in den einschlägigen Foren und Tutorials, die es zu jeder Programmiersprache zahlreich im Internet gibt.
Programmierbeispiele stehen aus o.g. Gründen ebenfalls nicht zur Verfügung.

Weitere Funktionen des Kommunikationscontrollers

Der Kommunikationskontroller kann weitere Steuerkommandos verarbeiten, die ebenfalls per UPD-Datagramm verschickt werden. Das Kommando "TXDATA" für die Datenübertragung zum Anwendungscontroller haben zuvor bereits kennengelernt.

Weitere Steuerkommandos für den Kommunikationscontroller sind:

"**LEDTOGGLE**"
Schaltet die TEST-LED um (ein/aus).

"**LEDON**"
Schaltet die TEST-LED ein.

"**LEDOFF**"
Schaltet die TEST-LED aus.

"**netPIO?**"
Gerät antwortet mit "netPIO!"

"**SLFRST**"
Das netPIO führt einen Neustart durch (RESET).

"**RSTCTRL**"
Der Anwendungscontroller führt einen Neustart aus.

"**RSTFAC**"
Löscht alle Benutzereinstellungen im Gerät und stellt die Werkseinstellung wieder her.

"**CHANGEIP**" **+ 4 Bytes** mit der neuen IP-Adresse
Ändert die IP-Adresse des Gerätes.

"**CHANGEPORT**" **+ 2 Bytes** mit der neuen Port-Nummer
Ändert den UDP-Port des Gerätes.

"**CHANGEMAC**" **+ 6 Bytes** mit der neuen MAC-Adresse
Ändert die MAC-Adresse des Gerätes.

"**GATEWAY**" **+ 4 Bytes** mit der neuen Gateway-IP
Ändert die Gateway-Adresse (=Router-IP) für den Internetzugang.

"**AUXD**x**ON**"
Schaltet einen AUX-Port-Ausgang auf high. ("x" = "**2**" ... "**7**")

"**AUXD**x**OFF**"
Schaltet einen AUX-Port-Ausgang auf low. ("x" = "**2**" ... "**7**")

"**AUXD**x**TOGGLE**"
Schaltet einen AUX-Port-Ausgang auf um. ("x" = "**2**" ... "**7**")

"**GETAUX**"
Fragt den Status des AUX-Ports und der TEST-LED ab. Die Antwort ist 4 Bytes lang. Die vier Bytes bilden einen lesbaren ASCII-String (z.B. "F3FC"). Der Wert des oberen Bytes (hier "F3") repräsentiert den Eingangsstatus, der Wert des unteren Bytes (hier "FC") den Ausgangsstatus des AUX-Ports:

Low-Byte : AUX-C(Bit 0..5) Eingangsstatus
High-Byte: AUX-D(Bit 2..7) Ausgangsstatus und TEST-LED (Bit 0)

"**TXDATA**" **+ n-Bytes** Request **+ 1 Byte** ResponseLen **+ 1 Byte** Sync
Überträgt den Request (n-Bytes) intern per UART (57600 Baud) an den Anwendungscontroller (d.h. ohne "TXDATA", ohne ResponseLen und ohne SYNC-Byte).

ResponseLen<>0 (= 1 ... 254):
Erwartet eine Antwort vom Anwendungscontroller mit der Länge ResponseLen.
Sobald die Antwort vom Anwendungscontroller vorliegt wird diese unverändert als UDP Antwort zurückgesendet.

ResponseLen=255:
Verwendet den Wert des SYNC-Bytes als Daten-Ende-Zeichen. Nimmt eine Antwort vom Anwendungscontroller entgegen, bis dieser abschliessend das SYNC-Byte sendet. Sobald die Antwort vom Anwendungscontroller vorliegt wird diese unverändert als UDP-Antwort zurückgesendet.

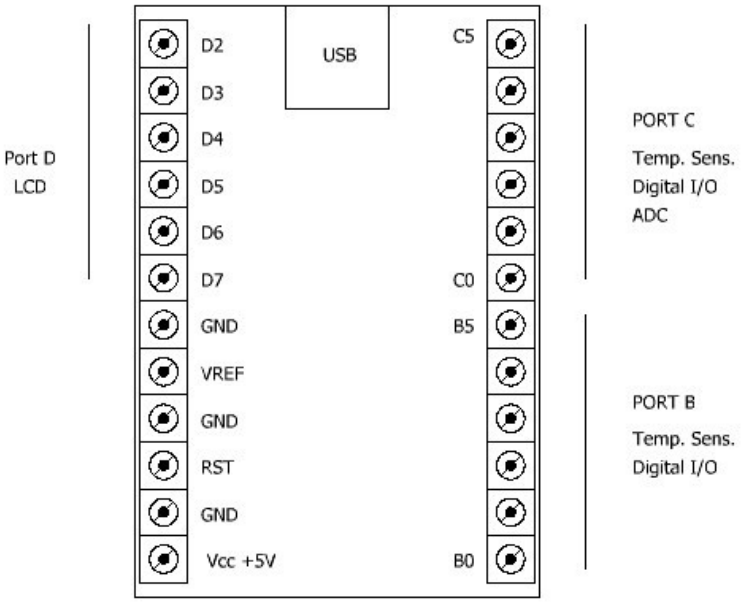
ResponseLen=0 ist nicht erlaubt.

Firmware /TEMP12

Die /TEMP12-Firmware ist für den Betrieb von bis zu zwölf digitalen Temperatursensoren vom Typ Dallas DS18B20 ausgelegt. Die Werte der zwölf Kanäle werden per USB zum PC übertragen. Der Anschluss der Sensoren erfolgt über die Anschlussklemmen B0…B5 und C0…C5. An jeden dieser Kanäle kann EIN Temperatursensor angeschlossen werden.

Freie Kanäle, die nicht für Temperatursensoren verwendet werden, können für alternative Aufgaben konfiguriert werden:

- DS18B20 Temperatursensor
- Digitaleingang
- Digitalausgang
- Alarmausgang
- Sensoreingang (ADC) (nur C0…C5)



Dieses Kapitel zeigt Anschlussbilder anhand des USB-µPIO-Moduls. Die Anschlussklemmen sind bei gleicher Anordnung und Funktion auch auf dem netPIO-Netzwerkmodul vorhanden.

Die Konfiguration des Gerätes erfolgt mit Hilfe der mitgelieferten Software. Diese ermöglicht die Einstellung der Kanäle. Die Kanaleinstellungen werden mit Hilfe der Konfigurationssoftware zum Gerät übertragen und dort temporär oder dauerhaft abgelegt. Somit ist das Board als PC-Interface geeignet, kann aber auch für den Betrieb ohne PC (Stand-Alone) verwendet werden.

Für jeden analogen Kanal (Temperatur- und Sensoreingänge) können zwei Grenzwerte definiert werden, die den Eingangswert als „HOCH“ und „TIEF“ einstufen. Diese Einstufung dient zur Ansteuerung von Kanälen, die als Alarmausgang konfiguriert sind. So kann ein Alarmausgang z.B. ein Steuersignal ausgeben, welches etwa einen Lüfter einschaltet, sobald eine Temperatur den Grenzwert „HOCH“ überschreitet. Ausserdem erscheint die Anzeige „HOCH“ bzw. „TIEF“ ggf. auch auf einem optionalen LCD-Display (siehe LCD-Option).

/TEMP12 Anschlussbelegung

DS18B20

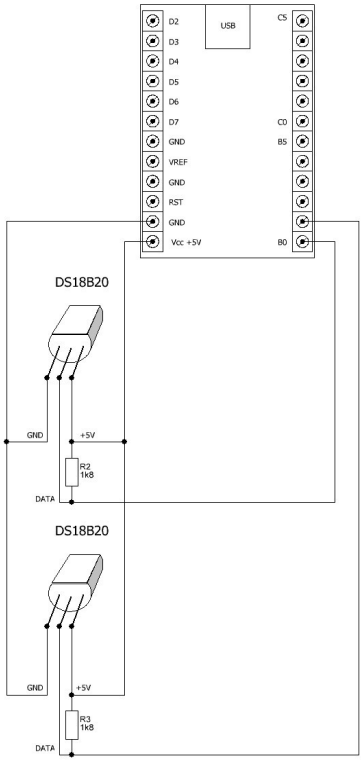
Der Anschluss von bis zu 12 Temperatursensoren ist an den Klemmen B0…B5 und C0…C5 möglich. Mit der Konfigurationssoftware muss der Kanal entsprechend konfiguriert werden.
Die Temperatursensoren sind für Temperaturen von -55 °C und 125° C geeignet. Sofern eine lineare Umrechnung in andere Temperatureinheiten gewünscht ist, (z.B. °F statt °C) so gibt man den Umrechnungsfaktor und Offset in die Felder für die Sensoranpassung ein.

Beispiel – Umrechnung °C in °F:
Einheit = „°F“
Faktor = 1,8
Offset = 32

Eine solche Skalierung des Messwerts verändert lediglich die Darstellung des Temperaturwertes. Dieses verändert jedoch NICHT den physikalischen Messbereich (-55 °C bis 125° C) des Sensors.

Die Eingabe der Grenzwerte für „HOCH“ und „TIEF“ erfolgt in der Einheit, in der Sie den Sensor skaliert haben.

Anschlussbild DS18B20
an Kanal B0 und B1



Dieses Kapitel zeigt Anschlussbilder anhand des USB-µPIO-Moduls. Die Anschlussklemmen sind bei gleicher Anordnung und Funktion auch auf dem netPIO-Netzwerkmodul vorhanden.

ADC Sensoreingang

Diese Konfiguration ist nur für die Kanäle C0…C5 möglich. Ein Analog-Digital-Wandler (ADC) erfasst eine Eingangsspannung, die z.B. ein Sensor liefert. Dazu vergleicht er intern die Eingangsspannung mit einer Referenzspannung VREF. Als Referenzspannungsquelle dient dabei entweder die (USB-) Betriebsspannung (Vcc; 5V), eine interne Referenz (1,1V) oder eine externe Referenzspannung, die über den Anschluss REF eingespeist wird. Die Referenzspannungsquelle ist für jeden Sensoreingang einzeln konfigurierbar, d.h. umschaltbar.

Die maximal zulässige Referenzspannung ist die Betriebsspannung Vcc! Die Eingangsspannung muss zwischen 0V und VREF liegen, damit sie vom AD-Wandler erfasst werden kann. Spannungen<0 Volt und Spannungen>Vcc sind an allen Eingängen unzulässig!

Bei korrekter Konfiguration der Referenzspannung liefert der ADC Messwerte in der Einheit Volt. Die Sensoranpassung erlaubt es, wie schon bei den Temperaturkanälen, die gemessene Eingangsspannung in eine andere Einheit linear umzurechnen.

Beispiel – Umrechnung Spannung (mit VREF = 5V) in Prozent

Einheit = „%“
Faktor = 20
Offset = 0

So ergibt sich bei einer Eingangsspannung von 5V eine interne Darstellung von 5(V) * 20 = 100%. Die Umskalierung verändert NICHT den physikalischen Bereich der Eingangsspannung von 0 V bis VREF, sondern nur die interne Darstellung. Die Anpassung des Messbereichs an die Ausgangsspannung des Sensors muss durch eine geeignete Beschaltung des Eingangs erfolgen (Spannungsteiler; Verstärker; o.ä.)

Alarmausgang

In dieser Konfiguration arbeitet der Kanal als digitaler (Schalt-) Ausgang. Im eingeschalteten Zustand „EIN“ liefert der Ausgang einen Highpegel (5V), im ausgeschalteten Zustand „AUS“ liefert der Ausgang einen Lowpegel (0V). Der Ausgang ist mit max. 20mA belastbar.
Ein Kanal der als Alarmausgang konfiguriert ist, wechselt seinen Schaltzustand EIN / AUS selbsttätig. Ist eine bestimmte BEDINGUNG erfüllt so schaltet der Ausgang EIN, andernfalls schaltet er AUS. Als Bedingungen dienen die Zustände „HOCH“ und „TIEF“ anderer Messkanäle (Temperatur oder ADC). Um ein „Flackern“ der Alarmausgänge zu vermeiden, wenn der Messwert leicht um den Grenzwert schwankt, ist eine Hysterese von ca. +/- 2 LSB fest vorgesehen.

Digitalausgang

In dieser Konfiguration arbeitet der Kanal als digitaler (Schalt-) Ausgang. Im eingeschalteten Zustand „EIN“ liefert der Ausgang einen Highpegel (5V), im ausgeschalteten Zustand „AUS“ liefert der Ausgang einen Lowpegel (0V). Der Ausgang ist mit max. 20mA belastbar. Das Umschalten erfolgt PC gesteuert.

Digitaleingang

In dieser Konfiguration arbeitet der Kanal als digitaler Eingang. Der eingeschaltete Zustand „EIN" wird durch einen Highpegel (5V) signalisiert, der ausgeschaltete Zustand „AUS" durch einen Lowpegel (0V). Offene (unbeschaltete) Eingänge sind mit einem internen Pullup-Widerstand auf Vcc Potential gelegt. So lässt sich z.B. leicht der Zustand eines Schalters erfassen, der die Eingangsklemme im geschlossenen Zustand mit Masse (GND) verbindet.

Bei Verwendung eines optionalen LCD-Displays werden die Zustände der digitalen Kanäle (Digitaleingang, Digitalausgang, Alarmausgang) als „EIN" bzw. „AUS" angezeigt.

Messen von Differenzen zwischen Sensoren

Da in der Praxis häufig Differenzen zwischen Sensoren ausgewertet werden müssen (z.B. Aussentemperatur – Innentemperatur) haben wir die /TEMP12-Firmware mit vier zusätzlichen (virtuellen) Messkanälen ausgestattet, die per Berechnung die Differenz aus zwei echten Messkanälen bilden. Folgende Differenzen können gebildet werden:

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

Damit die Differenzbildung sinnvoll ist, müssen die beiden Messkanäle, die zur Differenzbildung dienen sollen identisch konfiguriert werden. Ist dieses der Fall, so wird die zugehörige Differenz automatisch gebildet und die dabei Konfiguration der echten Messkanäle verwendet. Lediglich die Grenzwerte HOCH und TIEF sind für Differenzkanäle konfigurierbar und können ebenso wie die Grenzwerte echter Kanäle als Bedingung für Alarmausgänge verwendet werden.

Übertragen der Konfigurationseinstellungen zum Gerät

Vor dem eigentlichen Betrieb des Gerätes muss die Konfigurationseinstellungen zum Gerät übertragen werden. Dies geschieht mit Hilfe der Konfigurationssoftware. Die Konfiguration kann wahlweise in den flüchtigen RAM-Speicher oder in den nicht-flüchtigen EEPROM-Speicher des Gerätes übertragen werden. Bei einem Reset des Gerätes liest das Gerät die Konfiguration aus dem nicht-flüchtigen EEPROM-Speicher. Eine Konfiguration, die in den flüchtigen RAM-Speicher übertragen wurde geht dabei verloren. Ein Reset erfolgt beim Einschalten der Versorgungsspannung oder durch einen LOW-Impuls am Reset Anschluss (/RST). Ein Reset erfolgt auch, wenn der PC eine Datenverbindung mit dem Gerät aufbaut. Dieses kann durch das Entfernen des Jumpers J1 SOFT-RESET verhindert werden.
Beim Übertragen einer Konfiguration wird eine geräteeigene Uhr mit der Systemzeit des PC synchronisiert. Diese dient der Anzeige von Datum und Uhrzeit auf dem optionalen LCD-Display. Da die Uhrzeit technisch bedingt bei einem Reset verloren geht, erfolgt nach einem Reset keine Uhrzeitanzeige mehr. Für die Uhranzeige ist eine ununterbrochene Spannungsversorgung erforderlich.

/TEMP12 Datenprotokoll

Schnittstellenparameter

Baud 57600
8 Datenbits
Keine Parität
1 Stopbit

Die Kommunikation erfolgt nach dem Frage-Antwort-Prizip (Request/Response). Dabei besteht ein Request immer aus genau 21 Datenbytes. Nachdem Empfang dieser 21 Datenbytes sendet die Firmware in jedem Fall eine Antwort an den PC zurück. Ein neuer Request darf erst gesendet werden, wenn die Antwort vollständig empfangen wurden.

Request: 21 Bytes

Der Request-Datenblock hat eine Länge von 21 Bytes

Byte 1:	Immer 66 dez = chr(„B")
Byte 2…Byte 13:	Digitaler Ausgangsstatus B0…B5, C0…C5
Byte 14…21:	Reserviert

Diese Bytes setzen den Ausgangsstatus der Kanäle, sofern diese als digitaler Ausgang konfiguriert worden sind. Kanäle die nicht als digitaler Ausgang konfiguriert sind, bleiben unbeeinflusst. Ein Bytewert (Zeichen) von CHR(48) = „0" schaltet den zugehörigen Ausgang AUS. Ein Bytewert (Zeichen) von CHR(49) = „1" schaltet den zugehörigen Ausgang EIN. Andere Bytewerte werden ignoriert. Jeder Request muss mit einem grossen „B" CHR(66) beginnen.

Programmierbeispiele (Ausgabe per PRINT-Befehl auf serielle Schnittstelle):

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

Setzte alle Ausgänge auf Highpegel (sofern diese als digitaler Ausgang konfiguriert sind).
Beachten Sie das Semikolon! Nach den 21 Zeichen des Requests darf KEIN CHR(13) CHR(10) folgen!

```
Print „B000000000000--------";      // Schaltet alle digitalen Ausgänge auf Lowpegel.
Print „B100000000000--------";      // B0 auf HIGH; Alle andern LOW
Print „Bxxxxx11xxxxx--------";      // B5 und C0 auf HIGH; andere bleiben unverändert
Print „Bxxxxx01xxxxx--------";      // B5 auf LOW;C0 auf HIGH; andere unverändert
Print „Bxxxxx01xxxxx--------";      // B5 auf LOW; C0 auf HIGH; andere unverändert
Print „Bxxxxxxxxxxxx--------";      // Alle Kanäle unverändert (nur Antwort anfordern)
```

Nach JEDEM dieser Print-Befehl sendet das Gerät eine Antwort (Response).

Response

Nachdem ein Request mit 21 Zeichen vom Gerät empfangen wurden, sendet es eine Antwort an den PC. Diese besteht aus einer lesbaren Zeichenkette (String). Der String besteht aus mehreren Zellen von jeweils 6 Zeichen. Nach jeder Zelle folgt ein Semikolon als Trennzeichen. Die erste Zelle enthält den Kanalnamen des ersten Kanals (B0). Die darauf folgende Zelle enthält den aktuellen Wert dieses Kanals. Pro Kanal werden zwei aufeinander folgende Zellen gesendet. Nacheinander werden so die Zellenpaare (Kanalname; Wert) für alle Kanale übertragen und zwar in der Reihenfolge B0…B5, C0…C5, DIFF1..4.

Beispielantwort:

„ 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)

Zum Abschluss der Antwortzeichenkette werden die Zeichen CHR(13) CHR(10) übertragen. Ingesamt besteht eine Antwort also aus 32 Zellen (á 6 Zeichen + Semikolon) und den zwei Abschlusszeichen. Das macht 32 * 7+2 = 226 Zeichen.

Da die Zeichenketten der Zellen auch für die Darstellung auf dem LCD-Display dienen, enthalten diese einige Sonderzeichen. Für die Darstellung am PC müssen diese ggf. per Software wieder in die lesbare Darstellung gewandelt werden. Dabei gelten folgende Entsprechungen:

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

/TEMP12 LCD-Option

Die /TEMP12 -Firmware erlaubt den Betrieb eines Standard-LCD-Textdisplays (HD44780) mit 4x20 Zeichen an den Anschlüsse D2...D7. Andere Displayformate werden NICHT unterstützt. Auf dem Display werden die Werte und Zustände der Datenkanäle automatisch angezeigt.

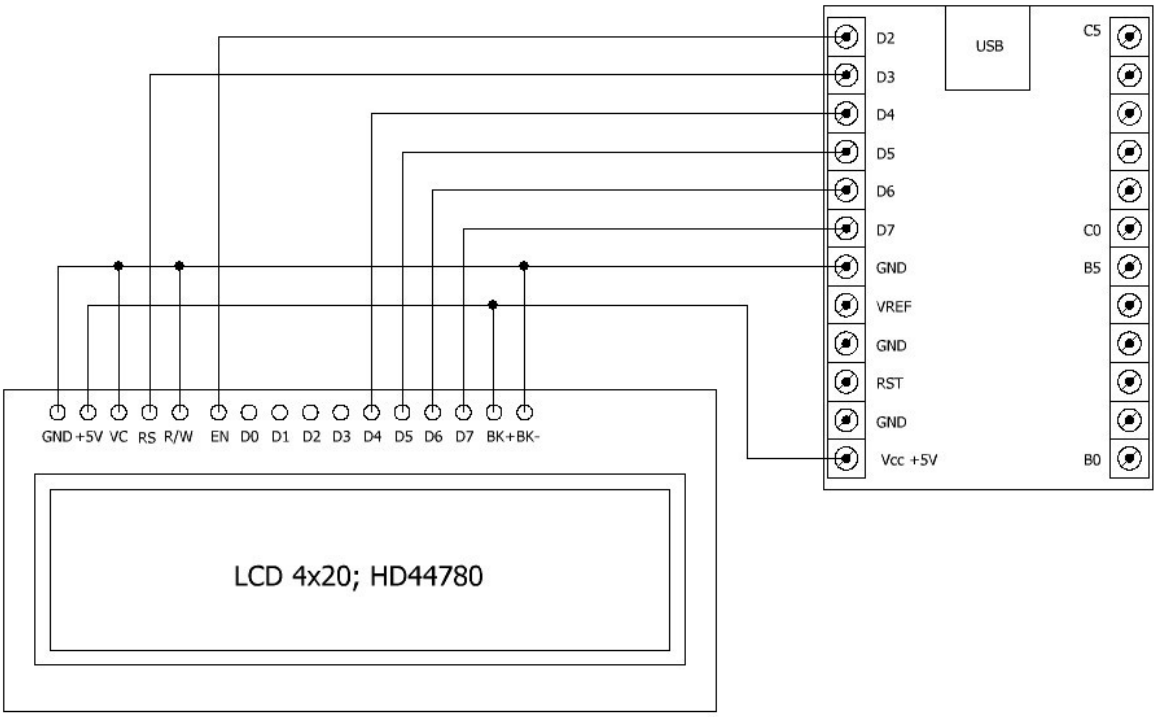
Der Anschluss des Displays erfolgt am Port D mit folgender Signalbelegung:

Board	LCD 4x20 Zeichen
PORT D	HD44780
D2	EN
D3	RS
D4	D4
D5	D5
D6	D6
D7	D7

Die übrigen Datenleitungen (D0...D3) **des LCD** bleiben unbeschaltet. Die Leitung R//W muss auf Masse-Potential (GND) geschaltet werden.

Anschlussbild LCD

Textdisplay 4x20 (HD44780)



Dieses Kapitel zeigt Anschlussbilder anhand des USB-µPIO-Moduls. Die Anschlussklemmen sind bei gleicher Anordnung und Funktion auch auf dem netPIO-Netzwerkmodul vorhanden.

Firmware /GPIO18

Die /GPIO18-Firmware arbeitet als PC-Interface mit 18 digitalen I/O-Kanälen deren Datenrichtung umschaltbar ist (18 GPIO).

/GPIO18 Anschlussbelegung

Die I/O-Leitungen sind zu drei Ports zusammengefasst (Port B, Port C, Port D). Jeder Port besitzt sechs I/O-Leitungen:

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

Die sechs Spannungen am Port C können darüber hinaus mit dem internen 10-Bit-A/D-Wandler (ADC) eingelesen werden. Die Referenzspannung für den ADC kann dabei aus drei umschaltbaren Quellen bezogen werden:

Externe Referenzspannung
Die Referenzspannung wird über den Anschluss VREF eingespeist.
0<Vref<Vcc ist dabei einzuhalten.

Interne Referenz VCC
Als Referenzspannung dient die positiver Versorgungsspannung (also ca. 5V).

Interne Referenz 1.1 V
Eine Referenzspannung von 1.1V wird intern erzeugt und dient als Referenz.

Für Kanäle die als Eingang arbeiten kann per Software ein interner Pullup-Widerstand aktiviert werden, der den jeweiligen Eingang auf das Potential der positiven Versorgungsspannung (Vcc) legt, um z.B. offenen Eingänge ein definiertes Potential zu geben.

/GPIO18 Datenprotokoll

Die Kommunikation mit dem PC erfolgt per USB. Der USB-Treiber installiert einen virtuellen COM-Port. Der Datenaustausch erfolgt somit wie über eine ‚echte' RS232-Schnittstelle.

Schnittstellenparameter

Baud 57600
8 Datenbits
Keine Parität
1 Stoppbit

Die Kommunikation erfolgt nach dem Frage-Antwort-Prizip (Request/Response). Dabei besteht ein Request immer aus genau sieben Datenbytes. Nachdem Empfang dieser sieben Datenbytes sendet die Firmware 15 Bytes als Antwort an den PC zurück.
Die erzielbare Abfragerate (sample rate) liegt bei etwa 150 Samples/Sek.

Request: 7 Bytes

Byte1: ADC-Referenzumschaltung (0=EXTERN; 1=INTERN VCC; 3=INTERN 1.1V)
Byte2: Richtung Port B
Byte3: Richtung Port C
Byte4: Richtung Port D
Byte5: Daten Port B
Byte6: Daten Port C
Byte7: Daten Port D

Die Bytes 2..4 geben die **Datenrichtung** an. Ein gesetztes Bit setzt die zugehörige Portleitung auf AUSGANG, ein gelöschtes Bit setzt die Portleitung auf EINGANG.

Die Bytes 5..7 setzen den Ausgangszustand der I/O-Leitungen deren Datenrichtung auf AUSGANG gesetzt ist. Ein gesetztes Bit schalten den Ausgang HIGH (5V), ein gelöschtes Bit schaltet den Ausgang LOW (0V). Für I/O-Leitungen die nicht als Ausgang sondern als EINGANG definiert wurden, wird mit den Datenbytes (Byte 5…Byte7) ein interner Pullup-Widerstand aktiviert . Ein gesetztes Bit aktiviert den Pullup für eine I/O-Leitung. Ein gelöschtes Bit deaktiviert den Pullup für die zugehörige I/O-Leitung.

Die Bits-Nummern der Datenbytes sind den Leitungsnummern der Portleitungen zugeordnet. Ein gesetztes Bit2 in Byte4 eines Requests setzt also z.B. die Datenrichtung der Portleitung D2 auf Ausgang.

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

Die Bytes 1…3 geben den Zustand der digitalen I/O-Leitungen wieder. Ist eine Leitung HIGH (5V) so ist das zugehörige Bit setzt, andernfalls ist es gelöscht. Dieses gilt unabhängig davon , ob die Leitung als Eingang oder als Ausgang arbeitet.

Die Bytes 4...15 liefern paarweise die 12-Bit Datenworte des A/D-Wandlers, der die Spannungen am Port C misst. Der Spannungswert ergibt mit Hilfe der verwendeten Referenzspannung zu:

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

Hinweise zur Verwendung des Protokolls

Zum Datenaustausch wird zunächst ein Request-Datenblock (Byte.array) mit einer Länge von genau sieben Bytes definiert, der mit ALLEN Informationen belegt sein muss, die zum Gerät geschickt werden sollen. Dann überträgt man diesen Block über die Schnittstelle. Anschliessend müssen die 15 Antwort-Bytes von der Schnittstelle eingelesen und nach Bedarf ausgewertet werden. Ein neuer Reqest-Block sollte erst dann ausgegeben werden, wenn alle 15 Bytes als Antwort auf den vorhergehenden Request empfangen wurden.

Firmware /FREQ

Die /FREQ-Firmware arbeitet als komfortabler **9-MHz-Frequenzzähler** (Counter) mit zahlreichen Zusatzfunktionen, wie Digital-I/O, PWM-Ausgang, Puls-/Taktgenerator.und sechs A/D-Eingängen.

Digital-I/O

Für alle Anschlüsse ist eine feste Datenrichtung vorgesehen, die nicht verändert werden kann. Alle Anschlüsse arbeiten mit 5V-TTL-Pegel und sind entsprechend der vorgegebenen Datenrichtung entweder als digital Eingang oder als Digitalausgang nutzbar. Je nach Verwendung und Konfiguration übernehmen die Leitungen aber andere, alternative Funktionen. Die Datenrichtung bleibt aber in jedem Fall erhalten.

9-MHz-Frequenzzähler

Die Einspeisung des Zählersignals erfolgt GLEICHZEITIG über die Anschlüsse D2 und D5. Je nach Eingangsfrequenz erfolgt die Frequenzmessung entweder durch eine Impulszählung mit Torzeit (D5; F>250 Hz) oder durch Flankenmessung (D2; F<250 Hz).

Als Torzeiten (Gate) für Impulszählung stehen folgende Zeiten zur Verfügung:
2000ms / 1000 ms / 500ms / 100 ms / 50 ms / 20 ms / 10ms. Bei Bedarf kann das Torzeitsignal (Gate) über den Digitalausgang D7 ausgegeben werden.

Bei niedrigen Signalfrequenzen (< 250 Hz) wird zusätzlich die vergangene Zeit ermittelt und ausgegeben, die seit dem letztmaligen Auftreten einer fallenden Flanke am Eingang D2 vergangen ist (ellapsed time). Bei niedrigen Frequenzen können somit Frequenzmessung, Zählfunktion und Zeitmessung parallel verwendet werden. Bei höheren Frequenzen sind nur noch die Frequenzmessung und die Zählerfunktion alternativ nutzbar.

32-Bit-Counter

Ohne Torzeitgenerator arbeitet der Frequenzzähler als Counter, der fallende Flanken an D5 aufwärts zählt. Digitaleingang B0 kann per Software als externer RESET-Eingang für den Zähler konfiguriert werden. Digitaleingang B4 kann per Software als externer ENABLE-Eingang (Freigabe) für den Zähler konfiguriert werden.

Generator 1

Dieser Generator erzeugt eine periodisches Ausgangstaktsignal am Ausgang D6.
Dieses Ausgangssignal wird aus einem Eingangstaktsignal abgeleitet wird.

Das Eingangstaktsignal wird dazu einem internen 8-Bit-Zähler als Taktsignal zugeführt.
Durch einen internen Vergleich des 8-Bit-Zählerstandes mit einem Vergleichswert, wird schliesslich das Ausgangssignal D6 gebildet. Je nach Konfiguration kann dieses ein Ausgangstaktsignal mit einstellbarer Frequenz oder mit einstellbarer Pulsbreite (PWM) sein.

Das Eingangstaktsignal kann entweder intern oder extern eingespeist werden.:

Bei interner Taktquelle wird der Systemtakt (18.432000 MHz) mit einem einstellbaren Vorteiler durch die Faktoren 1, 8, 64, 256 oder 1024 heruntergeteilt. Eingang D4 kann als per Software als externer Freigabeeingang für den internen Takt konfiguriert werden.

Bei externer Taktquelle dient Digitaleingang D4 kann als externer Takteingang, der per Software freigegeben oder gesperrt werden kann.

Generator 2

Diese Generator ist ein Pulsgenerator, der sein Ausgangssignal an D3 ausgibt.
Die Zeitdauer für die High- und Low-Phase des Ausgangssignal ist einstellbar. Die Zeiten stellen ganzzahlige Vielfache von vier möglichen Grundzeiteinheiten (Timebase) dar, die um den Faktor 1…65535 verlängert werden können. Die Grundzeiteinheiten sind 125µs, 500µs, 5ms und 500 ms. Während die Grundzeit für High- und Low-Phase dieselbe ist, können die Faktoren für die High-/Low-Dauer getrennt eingestellt werden.

Der Ausgangspuls von Generator 2 kann fortlaufend (continious) oder einmalig (singleshot) ausgegeben werden. Ein Einzelimpuls kann per Software ausgelöst werden. Alternativ kann Digitaleingang B1 per Software als Triggereingang konfiguriert werden. Bei fortlaufender Pulsausgabe wird das Ausgangssignal mit dem Triggersignal (B1 oder Software) synchronisiert.

Generator 3

Dieser Generator erzeugt am Digitalausgang B3 eine feste Ausgangsfrequenz von 72.000 KHz und einstellbarer Pulsbreite (8-Bit PWM).

Die Erzeugung der PWM erfolgt wie bei Generator 1 durch den Vergleich eines internen Zählerstandes mit einem Vergleichswert.

/FREQ Anschlussbelegung

Die Anschlüssen sind wie folgt belegt:

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 Diigtal Input (Frequenz-Eingang 0-250Hz)
D3 Digital Output (Generator 2)
D4 Diigtal Input (EXT CLK ENABLE)
D5 Diigtal Input (Frequenz-Eingang 250 Hz - 8MHz)
D6 Digital Output (Generator 1)
D7 Digital Output (Gate Clock Output)

C0…C5 Digital Inputs / ADC 0…5

Alle Eingänge von Port B und Port D sind mit Pullup-Widerständen auf Vcc-Potenzial gezogen. Port C kann als Digitaleingang (ohne Pullups) verwendet werden.

Die sechs Spannungen am Port C können alternativ mit dem internen 10-Bit-A/D-Wandler (ADC) eingelesen werden. Die Referenzspannung für den ADC kann dabei aus drei umschaltbaren Quellen bezogen werden:

Externe Referenzspannung:
Die Referenzspannung wird ggf. über den Anschluss VREF eingespeist.
0<Vref<Vcc ist dabei einzuhalten.

Interne Referenz VCC
Als Referenzspannung dient die positiver Versorgungsspannung (also ca. 5V).

Interne Referenz 1.1 V
Eine Referenzspannung von 1.1V wird intern erzeugt und dient als Referenz.

/FREQ Datenprotokoll

Schnittstellenparameter

Baud 57600
8 Datenbits
Keine Parität
1 Stoppbit

Die Kommunikation erfolgt nach dem Frage-Antwort-Prizip (Request/Response). Dabei besteht ein Request immer aus genau 10 Datenbytes. Nachdem Empfang dieser 10 Datenbytes sendet die Firmware in jedem Fall 33 Bytes als Antwort an den PC zurück. Ein neuer Request darf erst gesendet werden, wenn die 33 Antwortbytes empfangen wurden.

Request: 10 Bytes

Der Request-Datenblock hat eine Länge von 10 Bytes und überträgt sämtliche Daten zum Gerät, die für die Steuerung relevant sind:

Byte 1 Torzeit und Digital Out

Die Bit 0..2 selektieren die gewünschte Torzeit für die Frequenzmessung:

Bit 210	Hex	Dez	Torzeit
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	AUS / COUNTER

Die Bits 3…7 sind der Reihe nach den digitalen Ausgängen B2, B3, B5, D3 und D6 zugeordnet und setzen deren Ausgangsstatus bzw. aktivieren die alternative Funktion des jeweiligen Anschluss, wie z.B. Generator 2 am Ausgang D3.

Byte 2 Vergleichswert für Generator 1

Dieses Byte stellt den 8-Bit-Vergleichswert für Generator 1 ein und variiert je nach Modus die Ausgangsfrequenz bzw. die Pulsbreite des Ausgangssignal (D6).

Byte 3 Vergleichswert für Generator 3

Dieses Byte stellt den Vergleichswert für Generator 3 ein und variiert die Pulsbreite des Ausgangssignal (B3).

Byte 4 Modus

In Byte 4 sind folgende Funktionen bitweise kodiert:

- Modus und Taktquelle für Generator 1
- Zeitbasis für Generator 2
- Counter Reset per Software

Der Wert für Byte 4 ergibt sich aus der Addition der nachstehenden HEX oder Dezimalwerte, je nach gewünschter Funktion.

Die **Bits 0…2** selektieren die Taktquelle **für Generator 1**.

Bit [210]	Hex	Dez	Taktquelle	
000	00h	0	KEINE	
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	EXTERNER CLOCK (D4), fallende Flanke	
111	07h	7	EXTERNER CLOCK (D4), steigende Flanke	

Bit 3 setzt den Zählerstand des (Frequenz-)zählers auf Null zurück (**Counter Software Reset**), wenn das Bit gesetzt ist.

Bit	Hex	Dez	
[3]	08h	8	Counter Reset

Bit 4 und Bit 5 wählen den **Modus von Generator 1**

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

Modus: Normal
Der Generator arbeitet als Frequenzteiler, der die Frequenz des Eingangstaktsignals zunächst durch 2 und anschliessend durch 256 teilt.

Fout = (Fin / 2) / 256
Feste Ausgangsfrequenzen bei internem Takt:

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

Modus: PWM
Der Generator arbeitet als Frequenzteiler, der die Frequenz des Eingangstaktsignals zunächst durch 2 und anschliessend durch 255 teilt. Die Pulsbreite wird durch den Vergleichswert bestimmt.

Fout = (Fin / 2) / 255

PWM-Frequenzen bei internem Takt:

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

Modus: CTC
Der Generator arbeitet als einstellbarer Frequenzteiler, der die Frequenz des Eingangstaktsignals zunächst durch 2 und anschliessend durch den Vergleichswert+1 teilt.

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

Frequenzbereiche bei internem Takt:

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
Der Generator arbeitet als Frequenzteiler, der das Eingangstaktsignal durch 256 teilt.
Die Pulsbreite wird durch den Vergleichswert bestimmt

Fout = Fin / 256

PWM-Frequenzen bei internem Takt:

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

Bit 6 und Bit 7 stellen **die Zeitbasis von Generator 2** ein.

Bit [76]	Hex	Dez	Zeitbasis 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 ; Pulsdauer Dauer HIGH (INT16 / WORD)
Byte 7…8 TL; Generator 2; Pulsdauer Dauer LOW (INT16 / WORD)

Vier Bytes bilden jeweils einen ganzzahligen 16-Bit Wert, welche die High / Low-Dauer und damit das Tastverhältnis und die Frequenz des Ausgangssignals von Generator 2 (D3) bestimmen. Der Zeitwert ergibt sich aus der Multiplikation mit der gewählten Zeitbasis von Generator 2.

T high [s] = TH * Zeitbasis
T low [s] = TL * Zeitbasis

Mit TH = TL = 1 und einer Zeitbasis von 125µs ergibt sich
die maximale Ausgangsfrequenz von 4 kHz. f = 1/ ((TL + TH) * Zeitbasis)

Mit TH = TL = FFFFh und einer Zeitbasis von 500ms ergibt sich
Ausgangssignal mit einer maximalen Periodendauer von 65535 Sekunden, also mehr als 18 Stunden.
T = (TL + TH) * Zeitbasis

Byte 9 Konfiguration und Digital Out (Flags)

Die Bits in diesem Byte dienen als Flags für verschiedene Konfigurationen und Funktionen.

Bit 0 Ist Bit 0 gesetzt, arbeitet B0 als RESET-Eingang für den (Frequenz-)Zähler (Counter)
Bit 1 Ist Bit 1 gesetzt, arbeitet B1 SYNC/TRIGGER-Eingang für Generator 2
Bit 2 Ist Bit 2 gesetzt, arbeitet B4 als ENABLE-Eingang für den (Frequenz-)Zähler (Counter)
Bit 3 Bit 3 bewirkt die Umschaltung des Modus von Generator 2: Continuous / Single-shot
Bit 4 Bit 4 dient als SYNC/TRIGGER für Generator 2 (Software)
Bit 5 Bei **externer** Taktquelle für Generator 2:
Bit 5 gesetzt: Taktsignal von Eingang D4 freigegeben.
Bit 5 gelöscht Taktsignal von Eingang D4 gesperrt.
Bei **interner** Taktquelle für Generator 2:
Bit 5 gesetzt: Eingang D4 dient als Freigabesignal für den internen Takt.
Bit 5 gelöscht Interner Takt immer freigegeben (unabhängig von Eingang D4)
Bit 6 Bit 6 aktiviert die Ausgabe des Torzeit-Taktsignals am Ausgang D7.
Bit 7 Bit 7 schaltet den digitalen Ausgang D7 ein/aus.

Byte 10 ADC Referenzumschaltung (VREF SELECT)

Bit 0..1 wählen die **Referenzspannung** für die ADC-Spannungsmessung (C0…C5)

Bit 10	Hex	Dez	Modus Generator 1
00	00h	0	Externe Referenzspannung an VREF
01	01h	1	Interne Referenzspannung (Vcc)
10	02h	2	Nicht definiert
11	03h	3	Interne Referenzspannung 1.1 Volt
Bit 2..7	Reserviert		Auf Null setzen

Response: 33 Bytes

Nach dem Empfang von 10 Request-Bytes antwortet das Gerät mit 33 Response-Bytes:

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

Vier Bytes bilden jeweils einen ganzzahligen 32-Bit Wert (LSB first).Die drei Datenworte T1, T2 und T3 liefern das Ergebnis der Flankenmessung. Bei einer fallenden Flanke am Eingang D2 wird der Zählerstand eines freilaufenden 32-Zählers abgelesen, der intern mit 72 kHz (T = 13,8µs) getaktet ist.
Das Ergebnis dieser Ablesung befindet sich im Datenwort T2. Der Zählerstand der vorhergehenden Flanke wird zuvor in das Datenwort T1 verschoben. **T1 und T2 enthalten somit die Zählerstände von zwei aufeinander folgenden, fallenden Flanken.**

Per Software lässt sich daraus leicht die Periodendauer T und Frequenz F wie folgt ermitteln:

T = (T2 – T1) * 13,8 µs
F = 1 / T

Da der interne Zähler bei einem Wert von $FFFFFFFF überläuft und seine Zählung dann bei Null beginnt, sollte vor der Berechnung die Bedingung T2>=T1 geprüft werden.

Das Datenwort T3 enthält zusätzlich den aktuellen Zählerstand und ist unabhängig von der Flankenerkennung.

Die Differenz (T3 – T2) liefert somit ein Mass für die verstrichene Zeit, seit dem letzen Auftraten einer fallenden Flanke an D2. (Ellapsed Time).

T ellapsed = (T3 – T2) * 13,8 µs

Auch hier sollte T3>=T2 geprüft werden.

Byte 13…16 Frequenzzähler / Counter CNT (INT32 / DWORD)

Vier Bytes bilden einen ganzzahligen 32-Bit Wert (LSB first). Das Datenwort enthält den Zählerstand eines 32-Bit-Zählers, der durch den Frequenzeingang D5 getaktet aufwärts zählt. Sofern eine Torzeit aktiv ist, enthält das Datenwort den Zählerstand bei Ablauf der Torzeit. Mit Ablauf der Torzeit wir der Zähler auf Null zurückgesetzt. (Berechnung der Frequenz: siehe Byte 20)
Bei deaktivierter Torzeit enthält das Datenwort den aktuellen Zählerstand (Counter).

Byte 17 Digitalstatus Port B (Byte)

Die Bits 0..5 zeigen den Digitalstatus der Anschlüsse B0…B5 an.

Byte 18 Digitalstatus Port C (Byte)

Die Bits 0..5 zeigen den Digitalstatus der Anschlüsse C0…C5 an.

Byte 19 Digitalstatus Port D (Byte)

Die Bits 2..7 zeigen den Digitalstatus der Anschlüsse D2…D7 an. Die Erfassung des Digitalstatus ist unabhängig von der Datenrichtung.

Byte 20 Torzeit TZ [1/100 s]

Das Byte liefert die Torzeit (in 1/100 s), die dem Zählerstand CNT zu Grunde liegt. Per Software lässt sich aus der Torzeit TZ und dem Zählerstand CNT leicht die Frequenz am Eingang D5 ermitteln:

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

Ist keine Torzeit aktiv (Counter) ist TZ=0 und somit keine Frequenz zu ermitteln.

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)

Jeweils zwei Byte liefern ein 16 Bit Datenwort (LSB first). Das Datenwort enthält das Ergebnis der A/D-Konvertierung (10 Bit ADC; 000h….3FFh). Ein Spannungswert kann mit Hilfe der Referenzspannung Vref leicht errechnet werden:

U0 [Volt] = ADC0 / 3FFh * Vref

Byte 33 Reserviert

Das Byte 33 ist reserviert und liefert derzeit den Wert 13.

Firmware / INCR3

Das /INCR3 Firmware erlaubt den Anschluss von drei Inkrementalgebern mit TTL(5V)-Pegeln. Die typischen Signale der Inkrementalgeber (A; B) werden mit 32-Bit (Positions-) Zählern erfasst. Jede fallende Flanke am A-Eingang dient als Taktsignal für den zugehörigen Positionszähler. Die Zählrichtung wird durch das Richtungssignal am B-Eingang bestimmt
(high = aufwärts / low = abwärts). Die drei Zählerkanäle verfügen über Freigabe-Eingänge (EN; Enable), die den jeweiligen Zähler freigeben oder sperren.

Die maximale Zählfrequenz liegt bei 20 KHz (Tastverhältnis 1:1).

Sofern vorhanden kann ein Index-Signal (Z) verwendet werden, welches bei jeder vollen Umdrehung des Gebers einen kurzen Puls abgibt. Dabei wird der Positionszähler durch eine Flanke am Z-Eingang auf Null zurückgesetzt, so dass die Positionszählung dann synchron und relativ zur letzten Z-Flanke erfolgt. Ausserdem taktet das Z-Signal einen 16-Bit (Umdrehungs-)Zähler Z , dessen Zählrichtung durch die Zählrichtung des Positionszählers bestimmt wird.

/INCR3 Anschlussbelegung

Die Zählersignale sind den Anschlüssen von Port B und Port C wie folgt zugeordnet:

Zähler 1

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

Zähler 2

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

Zähler 3

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

Alle Zähler-Eingänge verwenden TTL 5V-Pegel und verfügen über interne Pullup-Widerstände, die offene Eingänge auf das Potential der positiven Betriebsspannung Vcc ziehen. Dies ermöglicht die z.B. die einfache Beschaltung mit Schaltkontakten gegen Masse.

Port D arbeitet als digitaler I/O-Port. Die Datenrichtung ist für jede Datenleitung einzeln umschaltbar. Port D6 gibt ein Taktsignal aus, sofern dieser auf Ausgang geschaltet ist. Die max. Grundfrequenz von 9 kHz ist durch die Faktoren 1, 2 , 3..., 256 teilbar. Für Leitungen die als Eingang arbeiten ist ein Pullup-Widerstand aktivierbar. Die Programmierung erfolgt mit Hilfe von Kommandos, die im Request-Datenblock übertragen werden.

/INCR3 Datenprotokoll

Schnittstellenparameter

Baud 57600
8 Datenbits
Keine Parität
1 Stopbit

Die Kommunikation erfolgt nach dem Frage-Antwort-Prizip (Request/Response). Dabei besteht ein Request immer aus genau fünf Datenbytes. Nachdem Empfang dieser fünf Datenbytes sendet die Firmware in jedem Fall 21 Bytes als Antwort an den PC zurück.
Die erzielbare Abfragerate (sample rate) liegt bei etwa 150 Samples/Sek.

Request: 5 Bytes

Byte 1: Kommando
Byte 2…5: Parameter

Das Kommando (Byte1) enthält eine Funktionsnummer, mit deren Hilfe verschiedene Geräteaktionen veranlasst werden können. Nicht definierte Funktionsnummern rufen nur den Antwortdatenblock ab, führen aber keine zusätzlichen Aktionen aus. Der Parameter (Byte2…Byte4) muss in jedem Fall mit gesendet werden, auch wenn dieser nur ausgewertet wird, wenn eine Funktion zusätzliche Angaben benötigt. Dieser besteht aus vier Bytes die je nach Bedarf einen 32-Bit-, 16 Bit oder 8 Bit-Wert übertragen. Dabei ist Byte2 das niederwertigste Byte und Byte 5 das höchstwertigste Byte des Parameters.

Folgende Zähler-Kommandos können im Request übertragen werden:

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

Port D-Kommandos

88dez	58h	„X“	Frequenzteiler f=9 KHz / (N+1)	Byte
89dez	59h	„Y“	PORTD = Status / Pullup	Byte
90dez	5Ah	„Z“	DDRD = Datenrichtung	Byte

Beispiel-Request Byte1…Byte 5: <5A><FC><00><00><00>
<5A> = Kommando „Datenrichtung Port D“
<FC> = %1111 1100 = Bit2…Bit7 gesetzt = D2…D7 Datenrichtung auf Ausgang setzen

Nach dem Empfang von fünf Request-Bytes, antwortet das Gerät stets mit einem Response-Datenblock. Ein neuer Request darf erst nach dem vollständigen Empfang der Antwort gesendet werden.

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

Positionszähler:

Byte4:	Zähler1	Bit 0…7 (LSB)
Byte5:	Zähler1	Bit 8…15
Byte6:	Zähler1	Bit 16…23
Byte7:	Zähler1	Bit 24…32 (MSB)
Byte8:	Zähler2	Bit 0…7 (LSB)
Byte9:	Zähler2	Bit 8…15
Byte10:	Zähler2	Bit 16…23
Byte11:	Zähler2	Bit 24…32 (MSB)
Byte12:	Zähler3	Bit 0…7 (LSB)
Byte13:	Zähler3	Bit 8…15
Byte14:	Zähler3	Bit 16…23
Byte15:	Zähler3	Bit 24…32 (MSB)

Umdrehungszähler (Z):

Byte16: Zähler1 Bit 8…15 (LSB)
Byte17: Zähler1 Bit 0…7 (MSB)

Byte18: Zähler2 Bit 8…15 (LSB)
Byte19: Zähler2 Bit 0…7 (MSB)

Byte20: Zähler3 Bit 8…15 (LSB)
Byte21: Zähler3 Bit 0…7 (MSB)

Die Bytes 1…3 geben den Zustand der digitalen I/O-Leitungen wieder. Ist eine Leitung HIGH (5V) so ist das zugehörige Bit setzt, andernfalls ist es gelöscht. Dieses gilt unabhängig davon, ob die Leitung als Eingang oder als Ausgang arbeitet. Die Bytes 4..15 liefern die 32-Bit Zählerwerte der Positionszähler. Die Bytes 16…21 liefern die 16-Bit Zählerwerte der Umdrehungszähler.

Firmware /PWMIO18

Die /PWMIO18-Firmware arbeitet als PC-Interface mit 18 digitalen I/O-Kanälen deren Datenrichtung umschaltbar ist (18 GPIO). Alle I/O-Leitungen erlauben alternativ die Ausgabe eines 8-Bit / 50Hz-PWM-Signals, sowie eines Blink- oder Puls-Signals. Port D kann für die Ansteuerung von (Modellbau-)Servos verwendet werden und gibt bei Bedarf ein 50Hz-PWM-Signal mit einer Pulsdauer von 1..2 ms aus. Servo- und PWM-Ausgänge können per Software, per ADC-Eingangsspannung oder von einem internen Rampengenerator gesteuert werden. Rampen und Puls-Ausgänge lassen sich durch Triggersignale am Port B auslösen. Die I/O-Leitungen sind zu drei Ports zusammengefasst (Port B, Port C, Port D).

/PWMIO18 Anschlussbelegung

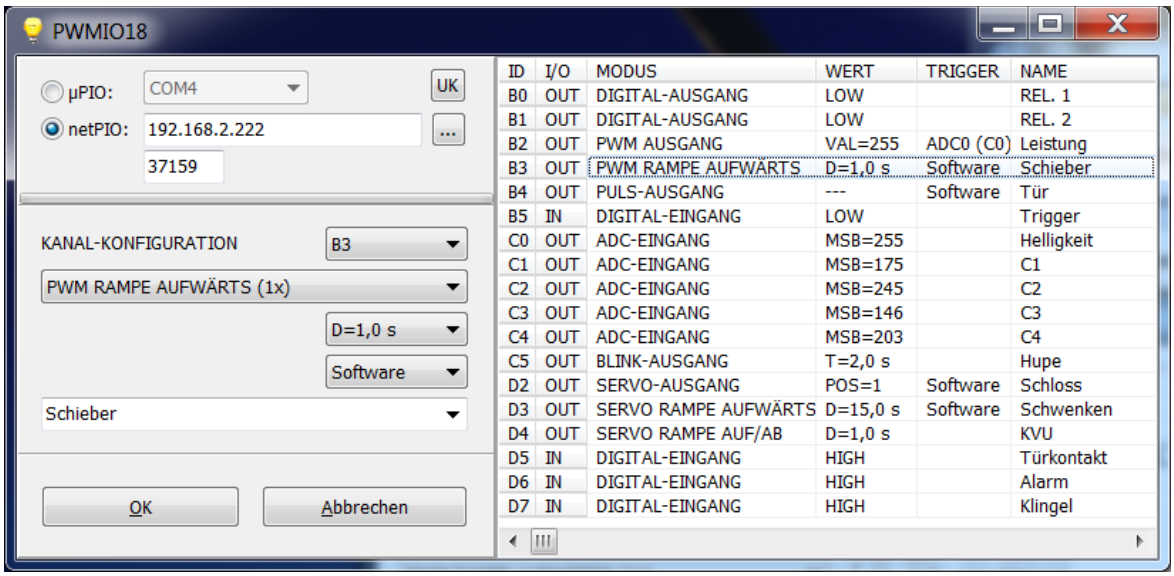
Jeder Port besitzt sechs I/O-Leitungen, die alternativ folgende Funktionen übernehmen können:

Port B0…Port B5	Digital-I/O, PWM, Blink/Puls-Ausgang, 6x Triggereingang
Port C0…Port C5	Digital-I/O, PWM, Blink/Puls-Ausgang, 6x ADC
Port D2…Port D7	Digital-I/O, PWM, Blink/Puls-Ausgang, 6x Servo

Die sechs Spannungen am Port C können darüber hinaus mit dem internen 10-Bit-A/D-Wandler (ADC) eingelesen werden. Die Referenzspannung für den ADC kann dabei aus drei umschaltbaren Quellen bezogen werden:

Externe Referenzspannung: Die Referenzspannung wird über den Anschluss VREF eingespeist. 0<Vref<Vcc ist dabei einzuhalten.
Interne Referenz VCC: Als Referenzspannung dient die +5V Versorgungsspannung.
Interne Referenz 1.1 V: Eine Referenzspannung von 1.1V wird intern erzeugt.

Für Kanäle die als Eingang arbeiten kann per Software ein interner Pullup-Widerstand aktiviert werden, der den jeweiligen Eingang auf das Potential der positiven Versorgungsspannung (Vcc) legt, um z.B. offenen Eingänge ein definiertes HIGH-Potential zu geben, welches von Schaltkontakten nach Masse gezogen werden kann. Mit dem Anlegen der Versorgungsspannung stellt die Firmware eine individuelle, auf dem Modul speicherbare Kanalkonfiguration selbstständig her, so dass das Modul auch ohne PC kleine Steuerungsaufgaben übernehmen kann. Zusammen mit der Konfiguration speichert das Modul für jeden Kanal einen benutzerdefinierbaren Kanalnamen (max. 17 Zeichen/pro Kanal).



Für die Kanalkonfiguration steht ein Software-Tool zur Verfügung, mit man die Grundkonfiguration des Moduls vornehmen kann.

/PWMIO18 Datenprotokoll

Schnittstellenparameter

Baud 57600
8 Datenbits
Keine Parität
1 Stoppbit

Die Kommunikation erfolgt nach dem Frage-Antwort-Prinzip (Request/Response). Dabei besteht ein Request immer aus genau 20 Datenbytes. Nachdem Empfang dieser 20 Datenbytes sendet die Firmware ebenfalls 20 Bytes als Antwort an den PC zurück.
Die erzielbare Abfragerate (sample rate) liegt bei etwa 10 Samples/Sek.

Request / Response: 20 Bytes

Ein Request besteht immer aus 20 Bytes. Das erste Byte dient als Kommando-Byte, das zweite Byte enthält die Kanalnummer. Es folgen 18 Parameter-Bytes.

/ Byte1 = Kommando // Byte 2 = Kanal // Byte 3..20 = Parameter /

Jedes Kommando wird mit einer Antwort von 20 Bytes vom Modul beantwortet. Die Antwort hat den gleichen Aufbau wie der Request.

Kommando (Byte 1):

Folgende Kommandos sind definiert:

Modus Lesen/Schreiben
    cmd_WriteChannelModes = 'D';
    cmd_ReadChannelModes = 'E';

Status Lesen/Schreiben
    cmd_WriteChannelStats = 'S';
    cmd_ReadChannelStats = 'T';

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

Namen Lesen/Schreiben
    cmd_WriteChannelname = 'N';
    cmd_ReadChannelname = 'O';

Referenzspannung unf ADC
    cmd_WriteVref = 'V';

cmd_ReadVref = 'W';
cmd_ReadADCs = 'A';

Speicherfunktionen
cmd_ConfigFromEeprom = 'X';
cmd_ConfigToEeprom = 'Y';

Kanalnummer (Byte 2):

Die Kanalnummer wird in **Byte 2** des Request abgelegt.
Es gilt folgende Kanalzuordnung (dezimal):

ch_ALL = 0; (Alle Kanäle ändern, sofern vom Kommando unterstützt)

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;

Parameter (Byte 3..20)

Die 18 Parameter-Bytes sind den Kanälen B0 bis D7 der Reihe nach zugeordnet.

Beispiele:
Das erste Parameter-Byte (P1 = Byte 3 des Requests) beeinflusst Kanal B0.
Das letzte Parameter-Byte (P18 = Byte 20 des Requests) beeinflusst Kanal D7.
Das achte Parameter-Byte (P8 = Byte 10 des Requests) beeinflusst Kanal 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'
Dieses Kommando stellt den gewünschten Kanal-Modus (Eingang, Ausgang, Blinken, etc) ein.

Wahlweise können die Modi aller 18 Kanäle auf einmal gesetzt, oder nur ein bestimmter Kanal verändert werden. (Siehe Abschnitt "Kanalnummer").

Die Bytewerte der 18 Parameter-Bytes (Byte 3..20 des Requests) bestimmen jeweils den Modus eines Kanals. Der Modus für Kanal B0 wird also z.B. im ersten Parameter-Byte (Byte 3 des Requests) abgelegt, der Modus für Kanal D7 im letzten Parameter-Byte (= Byte 20 des Requests). Wird ein bestimmter Kanal adressiert, werden die Parameter-Bytes aller anderen Kanäle ignoriert.

Nach dem Empfang des Kommandos sendet das Modul den Request unverändert als Response zurück.

Folgende Modi können durch die Parameterbytes eingestellt werden.

Pm_disabled = 0
Der Kanal wird deaktiviert und hochohmig geschaltet.

Pm_adc_input = 1
Kanal wird für die Verwendung des ADC (Spannungseingang) auf Eingang programmiert. Dieser Modus ist nur an Port C möglich!

Pm_open_input = 2
Kanal wird für die Verwendung als Digital-Eingang (5V-Logik) auf Eingang programmiert.
(Ohne externe Beschaltung ist der Zustand des hochohmigen Eingangs nicht definiert).

Pm_pullup_input = 3
Kanal wird für die Verwendung als Digital-Eingang auf Eingang programmiert und über einen interner Pullup-Widerstand auf HIGH-Potential gelegt, so dass dieser extern (z.B. durch einen Schaltkontakt) nach Masse gezogen werden kann.

Pm_output = 4
Kanal wird für die Verwendung als Digital-Ausgang (5V-Logik) auf Ausgang programmiert.

Pm_pwm = 5
Kanal wird für die Verwendung als PWM-Ausgang (5V-Logik) auf Ausgang programmiert.

Pm_servo = 6
Kanal wird für die Verwendung als Servo-Ausgang (5V-Logik) auf Ausgang programmiert.
Dieser Modus ist nur an Port D möglich!

Pm_blink100ms = 7
Kanal wird für die Verwendung als Blink-(Takt)-Ausgang programmiert. Dieser Modus ermöglicht ganzzahlige Intervalle von 1 * 100ms (0,1 Sekunde) bis 255 * 100 ms (25,5 Sekunden).

Pm_blink1s = 8
Kanal wird für die Verwendung als Blink-(Takt)-Ausgang programmiert. Dieser Modus ermöglicht ganzzahlige Intervalle von 1 * 1s (1 Sekunde) bis 255 * 1s (4Min,15Sekunden).

Pm_pulse100ms = 9
Kanal wird für die Verwendung als Puls-Ausgang programmiert. Dieser Modus ermöglicht ganzzahlige Pulsdauern von 1 * 100ms (0,1 Sekunde) bis 255 * 100 ms (25,5 Sekunden).
Dieser Modus ermöglicht die Triggerung (Auslösung) per Software oder durch eine fallende Flanke eines Kanals am Port B.

Pm_pulse1s = 10
Kanal wird für die Verwendung als Puls-Ausgang programmiert. Dieser Modus ermöglicht ganzzahlige Pulsdauern von 1 * 1s (1 Sekunde) bis 255 * 1s (4Min,15Sekunden). Dieser Modus ermöglicht die Triggerung (Auslösung) per Software oder durch eine fallende Flanke eines Kanals am Port B.

Pm_pwmRampUp = 11
Kanal wird für die Verwendung als PWM-Ausgang (5V-Logik) auf Ausgang programmiert. Die PWM-Modulation wird (einmalig) von einer ansteigenden Rampenfunktion des internen Rampengenerators moduliert (0...100%).
Dieser Modus ermöglicht die Triggerung (Auslösung) per Software oder durch eine fallende Flanke eines Kanals am Port B.

Pm_pwmSweepUp = 12
Kanal wird für die Verwendung als PWM-Ausgang (5V-Logik) auf Ausgang programmiert. Die PWM-Modulation wird (fortlaufend) von einer ansteigenden Rampenfunktion des internen Rampengenerators moduliert (0...100%).

Pm_ServoRampUp = 13
Kanal wird für die Verwendung als Servo-Ausgang (5V-Logik) auf Ausgang programmiert. Die Servo-Position wird (einmalig) von einer ansteigenden Rampenfunktion des internen Rampengenerators moduliert.
Dieser Modus ermöglicht die Triggerung (Auslösung) per Software oder durch eine fallende Flanke eines Kanals am Port B. Dieser Modus ist nur an Port D möglich!

Pm_ServoSweepUp = 14
Dieser Modus ist nur an Port D möglich!
Kanal wird für die Verwendung als Servo-Ausgang (5V-Logik) auf Ausgang programmiert. Die PWM-Modulation wird (fortlaufend) von einer ansteigenden Rampenfunktion des internen Rampengenerators moduliert (0...100%).

Pm_pwmRampDown = 15
Pm_pwmSweepDown = 16
Pm_servoRampDown = 17 Dieser Modus ist nur an Port D möglich!
Pm_servoSweepDown = 18 Dieser Modus ist nur an Port D möglich!
Diese Modi entsprechen den Modi 11,12,13,14, jedoch wird dabei die Rampenfunktion in abfallender Richtung durchlaufen.

Pm_pwmRampUpDown = 19

Pm_pwmSweepUpDown = 20
Pm_servoRampUpDown = 21 Dieser Modus ist nur an Port D möglich!
Pm_servoSweepUpDown= 22 Dieser Modus ist nur an Port D möglich!
Diese Modi entsprechen den Modi 11,12,13,14, jedoch wird dabei die Rampenfunktion abwechselnd in steigender und fallender Richtung durchlaufen.

cmd_ReadChannelModes = 'E'
Das Kommando ruft den eingestellten Kanal-Modus (Eingang, Ausgang, Blinken, etc) für alle Kanäle des Moduls gleichzeitig ab. Die Kanalnummer und Parameter-Bytes dieses Requests werden nicht ausgewertet (don´t care). Die Parameter-Byte der Antwort enthalten die Kanal-Modi, wie zuvor unter **cmd_WriteChannelModes** beschrieben**.**

cmd_WriteChannelStats = 'S'
Dieses Kommando überträgt Zustandsparameter an die Kanäle. Je nach eingestelltem Kanalmodus beeinflussen die Zustandsparameter das individuelle Kanalverhalten (wie z.B. Blinkintervall, Servoposition, etc.). Wahlweise können alle 18 Kanäle auf einmal gesetzt, oder nur ein bestimmter Kanal verändert werden. (Siehe Abschnitt "Kanalnummer").
Nach dem Empfang des Kommandos sendet das Modul die Zustandsparameter aller Kanäle als Response zurück. Die Aufgabe des Zustandsparameters ist vom eingestellten Modus des Kanals abhängig. Nachfolgende Modi werten das Schreiben von Zustandsparametern aus:
Pm_output:
Parameter-Byte = 0 => Ausgang geht auf LOW-Pegel
Parameter-Byte <> 0 => Ausgang geht auf HIGH-Pegel

Pm_blink100ms,
Pm_pulse100ms:
Parameter-Byte = 0 => Ausgang geht auf LOW-Pegel
Parameter-Byte = 1 ...255 => Stellt die Dauer ein (Parameter * 100ms)

Pm_blink1s,
Pm_pulse1s:
Parameter-Byte = 0 => Ausgang geht auf LOW-Pegel
Parameter-Byte = 1 ...255 => Stellt die Dauer ein (Parameter * 1s)

Pm_pwm:
Parameter-Byte = 0 ...255 => Stellt die PWM ein (0..100%)

Pm_servo:
Parameter-Byte = 0 ...255 => Stellt die Servoposition ein

Pm_xxRampxx,
Pm_xxSweepxx:
Parameter-Byte = 0 => Rampe aus
Parameter-Byte = 1 ...255 => Stellt die Rampendauer ein (1...255 s)

Die Puls bzw. Rampen-Auslösung per Software erfolgt durch das Schreiben der Dauer und nur dann wenn ein Kanal einzeln adressiert wurde.

cmd_ReadChannelStats = 'T'
Liest die Zustandsparameter aller Kanäle. Die Kanalnummer und Parameter-Bytes dieses Requests werden nicht ausgewertet (don´t care). Die Antwort enthält die Kanalzustände wie zuvor unter **cmd_WriteChannelStats** bechrieben. Darüber hinaus werden auch die Zustände von Eingangskanälen gelesen:

Pm_adc_input :
Parameter-Byte => Bytewert des ADC (MSB)

Pm_open_input ,
Pm_pullup_input = 3
Parameter-Byte = 0 => Eingang ist LOW
Parameter-Byte <> 0 => Eingang ist HIGH
cmd_WriteChannelSources = 'F'
Einige Modi können nicht nur per Software gesteuert werden, sondern erlauben alternativ eine Steuerung durch einen anderen, zuweisbaren Kanal (Source). Dieses Kommando wählt die gewünschte Steuerquelle für einen steuerbaren Kanal.

Pm_pwm, Pm_servo:
PWM- und Servokanäle können so konfiguriert werden, dass die PWM-Modulation bzw. Servoposition durch einen (an Port C verfügbaren) ADC-Kanal (d.h. spannungsabhängig) gesteuert wird. Dazu übergibt man in den Parameterbytes dieses Kommandos einen der nachstehenden Werte für den zu steuernden Kanal.

```
src_sw  = 0; // Steuerung per Software
src_adc0 = 1; // Steuerung durch einen ADC-Kanal
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,
Kanäle mit Puls- oder Rampenfunktion können so konfiguriert werden, dass die Auslösung (Triggerung) statt per Software, durch eine fallende Flanke an einem der Kanäle des Port B erfolgt. Dazu übergibt man in den Parameterbytes dieses Kommandos einen der nachstehenden Werte für den zu steuernden Kanal.

```
src_sw = 0;        // Auslösung per Software
src_B0 = 1;        // Auslösung durch fallende Flanke am Port B
src_B1 = 2;
src_B2 = 3;
src_B3 = 4;
src_B4 = 5;
src_B5 = 6;
```

cmd_ReadChannelSources = 'G'
Einige Modi können nicht nur per Software gesteuert werden, sondern erlauben alternativ eine Steuerung durch einen anderen, zuweisbaren Kanal (Source). Dieses Kommando liest die eingestellte Steuerquelle eines einen steuerbaren Kanals.
(siehe cmd_WriteChannelSource)

cmd_WriteVref = 'V'
Diese Kommando wählt eine von drei möglichen Optionen der Referenzspannung aus, die für die alle ADC-Kanäle des Port C gemeinsam verwendet wird.

```
VrefExt = 0; // Referenzsspannung extern eingespeisen
VrefVcc  = 1; // Referenzsspannung = 5V Betriebsspannung
Vref1V1  = 3; // Referenzsspannung = 1,1V (wird intern erzeugt)
```

Der Wert für die gewünschte Funktion, wird im Kanalbyte des Request übergeben, da dieses Kommando keine Kanalnummer benötigt.

cmd_ReadVref = 'W'
Dieses Kommando liest die gewählte Option für die Referenzspannung.
(siehe cmd_WriteVref)

cmd_ReadADCs = 'A'
Dieses Kommando liest die ADC-Werte (Port C) unabhängig davon welcher Modus für den Kanal konfiguriert ist. Das Kommando hat keinen Einfluss auf die Datenrichtung oder den Modus der Port C-Kanäle. Die ersten zwölf Parameter-Bytes der Antwort anhalten die RAW-Werte (Words) aller sechs ADC-Kanäle. Die Words sind "left-aligned", d.h. die oberen 10-Bit des Words (MSB) enthalten den 10-Bit-Wandler-Wert, während die unteren sechs Bit (LSB) undefiniert sind. Je zwei Parameter-Bytes bilden paarweise das Datenwort.

Das erste Parameterbyte enthält das LByte von ADC0.
Das zweite Parameterbyte enthält das HByte von ADC0.
Das dritte Parameterbyte enthält das LByte von ADC1.
usw.
...
Das zwölfte Parameterbyte enthält das HByte von ADC5.

cmd_WriteChannelName = 'N'
Dieses Kommando gibt einem Kanal einen benutzerdefinierten Namen.
Der zu übertragene ASCII-String (Kanalname) wird als NULL-terminierter String in den Parameter-Bytes (Byte 3..20) des Request abgelegt. Für jeden Kanal sind somit 17 ASCII-Zeichen nutzbar, die mit einem CHR(0) abgeschlossenen sein müssen. Nicht verwendete Zeichen am Ende sollten mit CHR(0) aufgefüllt werden.

Nach dem Empfang des Kommandos sendet das Modul den Request unverändert als Response zurück.

cmd_ReadChannelName = 'O'
Liest den Kanalnamen eines Kanals. Die Kanalzuordnung erfolgt wie zuvor beschrieben mit Byte 2. Die Parameterbytes (3..20) des Request werden nicht ausgewertet (don´t care bytes).

Die Antwort enthält den angeforderten Kanalname in den Parameter-Bytes (3..20) als null-terminierten ASCII-String.

cmd_ConfigToEeprom = 'Y'
Alle Konfigurationsdaten, wie Modus, Status und Name der Kanäle, usw., die mit den zuvor beschrieben Kommandos zum Modul übertragen werden, gelangen zunächst nur "vorläufig" in den flüchtigen RAM-Speicher des Moduls, und gehen somit beim Abschalten der Versorgungsspannung verloren.
Dieses Kommando übernimmt daher die aktuellen Einstellungen in den EEPROM-Speicher des Moduls und legt sie dort dauerhaft ab.
Nach dem Einschalten der Versorgungsspannung verwendet das Modul die Einstellungen, die mit diesem Kommando zuvor im EEPROM als Startkonfiguration abgelegt wurden.
Nach dem Empfang des Kommandos sendet das Modul den Request unverändert als Response zurück.

cmd_ConfigFromEeprom = 'X'
Stellt die Startkonfiguration wieder her, die im EEPROM-Speicher des Moduls abgelegt wurde (siehe ***cmd_ConfigToEeprom***). Nach dem Empfang des Kommandos sendet das Modul den Request unverändert als Response zurück.

/PWMIO18 Anhang

Kommmandos (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';
```

Kanaldressierung (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;

Protokollstruktur (USB VCP)

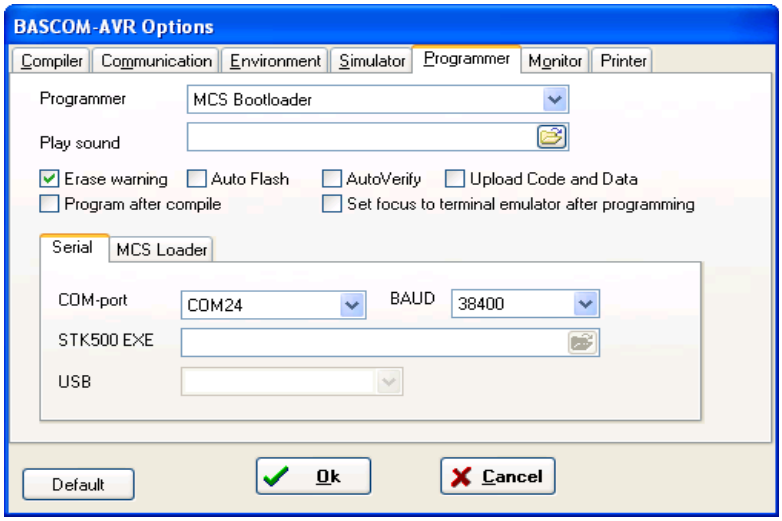
```
RequestSize = 20;
TRequest =  packed record
                Command: AnsiChar;
                ChannelNo: Byte;
                Params: array[0..17] of Byte;
            end;

ResponseSize = 20;
TResponse = TRequest;
```

Firmware /MCS

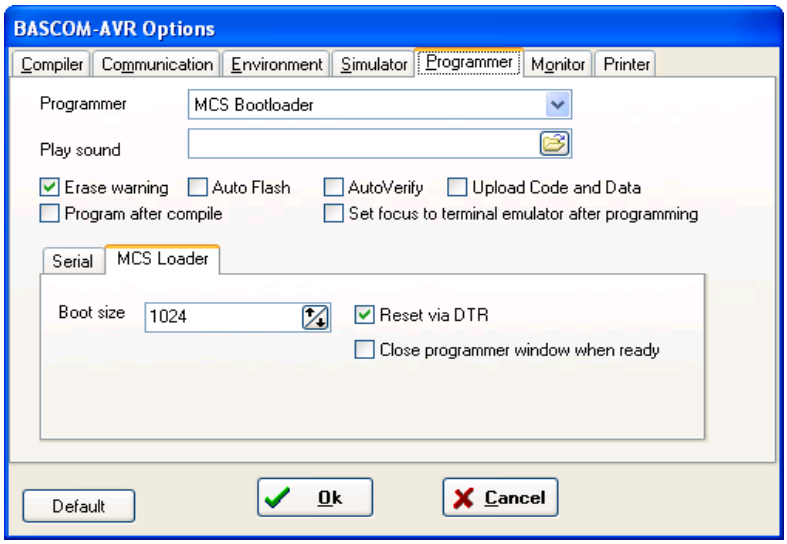
Mit der Firmwareoption /MCS ist das Board in Verbindung mit BASCOM AVR (MCS Electronics) sofort einsatzbereit. Programme die Sie selbst mit BASCOM AVR entwickelt haben, können bequem per USB auf das Board übertragen werden. Es ist also kein Hardware-Programmiergerät mehr erforderlich. Den MCS-Bootloader haben wir bereits für Sie installiert. Dieser benötigt etwa 2KByte des 16KByte grossen Flash-Speichers.

Bitte arbeiten Sie mit der neuesten Version von BASCOM AVR. Ein kostenloses Update und eine Demoversion gibt auf der Internetseite von MCS.

Zum Übertragen Ihrer Programme stellen Sie in BASCOM AVR unter OPTIONS->PROGRAMMER den MCS-Boolader ein.



Der COM-Port wird Ihnen vom USB-Gerätetreiber bereit gestellt. (Virtueller COM-Port; VCP). Schauen Sie im Gerätemanager nach welche COM-Port-Nummer vergeben wird.
Die Baudrate des Bootloaders ist 38400 Baud. Alle Software-Optionen, wie z.B. ‚Reset via DTR' sind damit nutzbar.



Weitere Informationen:
http://www.mcselec.com
AN #143 - MCS Bootloader

Firmware /CLEAR

Sie erhalten das Board bestückt mit einem unprogrammierten ATMega168-20.
Die Programmierung muss mit einem handelüblichen ISP-Programmer erfolgen. Dieser gehört NICHT zu Lieferumfang. Das Board besitzt einen 6-poligen ISP-Anschluss.

Für die Fusebits empfehlen wir folgende Grundeinstellung:

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

Programmer-Empfehlung:
Den „mySmartUSB light" von myAVR haben wir erfolgreich mit dem Board getestet und ist als Zubehör erhältlich. Ein passendes 6-poliges ISP-Kabel liegt dem Programmer bei.