

Das autonome Roboter Board RBNFRA 1.2



Ein universellen Roboter Board´s nach den Richtlinien, die von den RoboterNetz – jetzt unter [CC-Lizenz](#) freigegeben!

Dieses Mikrocontroller-Board eignet sich ideal um kleinere bis mittlere Roboter oder Modellfahrzeuge zu steuern. Die notwendigen Motoransteuerung ist bereits direkt auf dem Board verbaut, es können sowohl Getriebemotoren als auch Schrittmotoren angesteuert werden.

Die Board Version 1.2 ist noch etwas stärker gepackt und besitzt einige Bauteile als auch Ports mehr als deren Vorgängervariante.. Das Board ist nun auch in der Lage den Ruhestrom im Schrittmotorbetrieb als auch Getriebemodus durch softwaremäßige Ausschaltung der Motorendstufe erheblich zu reduzieren. Zudem gibt es nun einen zusätzlichen Sleep-Modus, welcher Hardware-Erweiterungen (welche dieses Unterstützen) auf einen Stromsparmmodus umschalten.

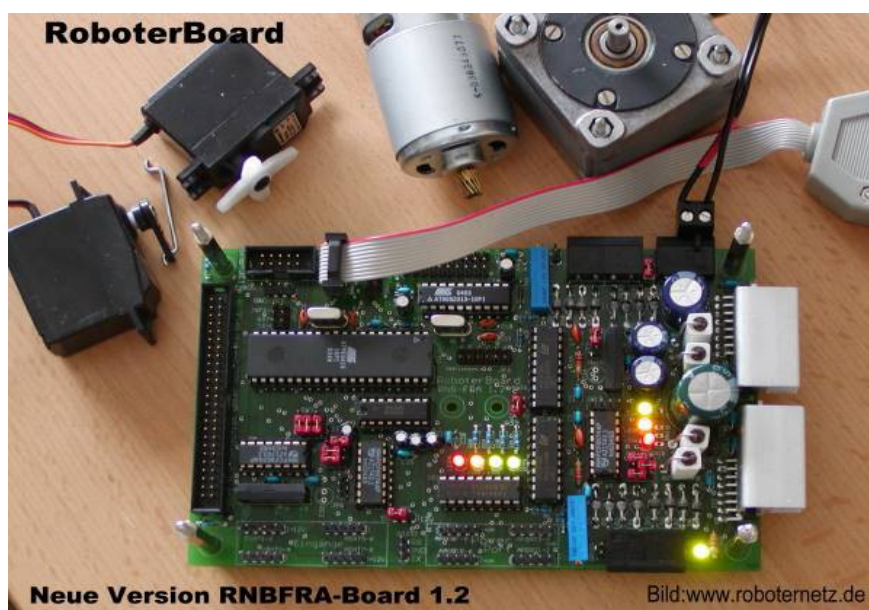
Getrennt davon ist das Board auch in der Lage per Software Sensoren und Aktoren auszuschalten. Durch alle diese Maßnahmen, die getrennt voneinander aktiviert werden können, erhöht die Akkuausdauer erheblich. So kann sich das Board nun z.B. mehrere Stunden schlafen legen und dann per Software erwachen um die Hausarbeit zu übernehmen.

Zudem stehen wie schon gesagt nun auch 5 weitere I/O Ports zur freien Verfügung. Weiterhin sind 3 LED´s hinzugekommen die nun unabhängig vom sogenannten Power-Port angesteuert werden können – diese eignen sich daher optimal als Status-Signale.

Da dieses Board nun alle wesentlichen Wünsche eines Roboter-Bastlers unterstützt – ist keine Revision mehr geplant. Da dieses Projekt jetzt unter der [CC-Lizenz](#) lizenziert wurde, kann das Board natürlich nach eigenen Wünschen modifiziert werden.

Die Anleitung, Bauteilelisten usw. und sogar die Eagle-Dateien können über die Seite <http://www.mikrocontroller-elektronik.de/> heruntergeladen werden.

Dieses Werk ist lizenziert unter einer [Creative Commons Namensnennung - Nicht-kommerziell - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz](#).



Lizenzhinweis

Dieses Projekt dient vornehmlich für Lehrzwecke und zum Experimentieren. Für den Aufbau sollten ausreichend Elektronik Grundkenntnisse und Kenntnisse bezüglich der Sicherheit (Experimentieren mit Strom und Handhabung gängiger Werkzeuge wie Lötkolben etc.) vorhanden sein. Auf der Seite www.Mikrocontroller-Elektronik.de empfehle ich diesbezüglich noch interessante Literatur mit der man sich dies erarbeiten kann. Weitere Literaturtipps sind auch unter RN-Wissen.de in der Rubrik [Buchvorstellungen](#) zu finden. Für Fragen bezüglich Elektronik und Mikrocontroller empfehle ich das Forum: Roboternetz.de

Um ihnen weitgehende Möglichkeiten zum Nutzen der Schaltung einzuräumen, wurde dieses Projekt jetzt unter die [CC-Lizenz](#) gestellt. Sie haben So die Möglichkeit die Schaltung beliebig zu verändern oder weiterzugeben. Lediglich die kommerzielle Weitergaben ist nur mit Genehmigung möglich! Genauere Hinweise finden Sie im Lizenztext.

Dieses Projekt (Schaltung und Projektdateien) steht unter der Creative-Commons-Lizenz Namensnennung - Nicht-kommerziell - Weitergabe unter gleichen Bedingungen 4.0 International. Um eine Kopie dieser Lizenz zu sehen, besuchen Sie <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Lizenziert wurde das Projekt von:

User Frank / www.Roboternetz.de & www.Mikrocontroller-Elektronik.de

Dieser Name und diese Webseiten sind bei der Weitergabe stets deutlich sichtbar zu nennen!

Achtung: Es kann keinerlei Garantie für die Fehlerfreiheit der Schaltung oder anderer Projektdateien übernommen werden! Der Nachbau und Betrieb geschieht auf eigene Gefahr! Jegliche Haftung für Schäden wird ausgeschlossen! Schadensersatzansprüche, gleich aus welchem Rechtsgrund, sind ausgeschlossen.



Dieses Projekt (Schaltung und Projektdateien) steht unter der Creative-Commons-Lizenz Namensnennung - Nicht-kommerziell - Weitergabe unter gleichen Bedingungen 4.0 International. Um eine Kopie dieser Lizenz zu sehen, besuchen Sie <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Lizenziert wurde das Projekt von: User Frank www.Roboternetz.de & www.Mikrocontroller-Elektronik.de
Dieser Name und diese Webseiten sind bei der Weitergabe stets deutlich sichtbar zu nennen!
Über diese Lizenz hinausgehende Erlaubnisse können Sie unter <http://www.mikrocontroller-elektronik.de/> erhalten.

Achtung: Der Nachbau und Betrieb geschieht auf eigene Gefahr! Jegliche Haftung für Schäden wird ausgeschlossen!
Es kann keinerlei Garantie für die Fehlerfreiheit der Schaltung oder anderer Projektdateien übernommen werden!

Wichtige Hinweise und Haftungsausschluss



Lesen Sie bitte diese Hinweise, bevor sie dieses Projekt nachbauen bzw. in Betrieb nehmen.

Bestimmungsgemäße Verwendung: Dieses Projekt ist nur Entwicklungsaufgaben, Forschung, Lehrzwecke und Unterricht und Prototypenbau konzipiert! Für die Einhaltung der technischen Vorschriften sind sie selbst verantwortlich.

Sicherheitshinweise

Beim Umgang mit Produkten, die mit elektrischer Spannung in Berührung kommen, müssen die gültigen VDE-Vorschriften beachtet werden, insbesondere VDE 0100, VDE 0550/0551, VDE 0700, VDE 0711 und VDE 0860.

Werkzeuge dürfen an Geräten, Bauteilen oder Baugruppen nur benutzt werden, wenn sichergestellt ist, dass die Geräte von der

Versorgungsspannung getrennt sind und elektrische Ladungen, die in den im Gerät befindlichen Bauteilen gespeichert sind, vorher entladen wurden.

Spannungsführende Kabel oder Leitungen, mit denen das Gerät, das Bauteil oder die Baugruppe verbunden ist, müssen stets auf Isolationsfehler oder Bruchstellen untersucht werden. Bei Feststellen eines Fehlers in der Zuleitung muss das Gerät unverzüglich aus dem Betrieb genommen werden, bis die defekte Leitung ausgewechselt worden ist. Bei Einsatz von Bauelementen oder Baugruppen muss stets auf die strikte Einhaltung der in der zugehörigen Beschreibung genannten Kenndaten für elektrische Größen hingewiesen werden. Wenn aus einer vorliegenden Beschreibung für den nicht gewerblichen Endverbraucher nicht eindeutig hervorgeht, welche elektrischen Kennwerte für ein Bauteil oder eine Baugruppe gelten, wie eine externe Beschaltung durchzuführen ist oder welche externen Bauteile oder Zusatzgeräte angeschlossen werden dürfen und welche Anschlusswerte diese externen Komponenten haben dürfen, so muss stets ein Fachmann um Auskunft ersucht werden. Es ist vor der Inbetriebnahme eines Gerätes generell zu prüfen, ob dieses Gerät oder Baugruppe grundsätzlich für den Anwendungsfall, für den es verwendet werden soll, geeignet ist!

Im Zweifelsfall sind unbedingt Rückfragen bei Fachleuten, Sachverständigen oder den Herstellern der verwendeten Baugruppen notwendig!

Bitte beachten Sie, dass Bedien- und Anschlussfehler außerhalb unseres Einflussbereiches liegen. Verständlicherweise können wir für Schäden, die daraus entstehen, keinerlei Haftung übernehmen. Bei Installationen und beim Umgang mit Netzspannung sind unbedingt die VDE-Vorschriften zu beachten. Geräte, die an einer Spannung über 35 V betrieben werden, dürfen nur vom Fachmann angeschlossen werden. In jedem Fall ist zu prüfen, ob der Bausatz oder die Platine für den jeweiligen Anwendungsfall und Einsatzort geeignet ist bzw. eingesetzt werden kann.

Derjenige, der einen Bausatz fertigstellt oder eine Baugruppe durch Erweiterung bzw. Gehäuseeinbau betriebsbereit macht, gilt nach DIN VDE 0869 als Hersteller und ist verpflichtet, bei der Weitergabe des Gerätes alle Begleitpapiere mitzuliefern und auch seinen Namen und Anschrift anzugeben. Geräte, die aus Bausätzen selbst zusammengestellt werden, sind sicherheitstechnisch wie ein industrielles Produkt zu betrachten.

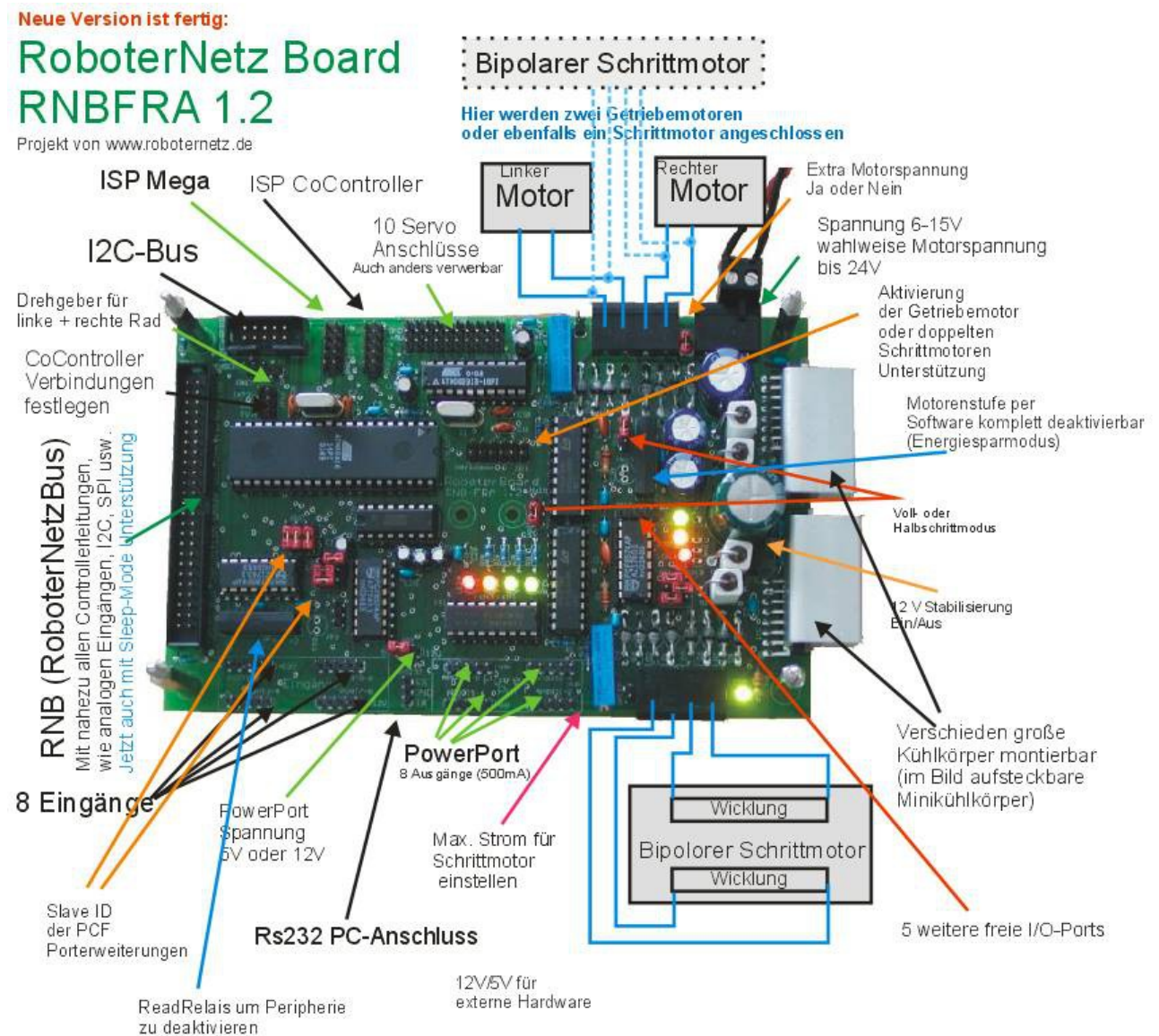
Für alle Personen- und Sachschäden, die aus nicht bestimmungsgemäßer Verwendung entstehen, ist nicht der Hersteller sondern der Betreiber verantwortlich. Bitte beachten Sie, dass Bedien- und/oder Anschlussfehler außerhalb unseres Einflußbereiches liegen. Verständlicherweise können wir für Schäden, die daraus entstehen, keinerlei Haftung übernehmen.

Das Board kann sowohl als eigenständiges Roboterboard als auch als Erweiterungsplatine für ein bestehendes Roboter Projekt genutzt werden! Da das Board in Verbindung mit unterschiedlichen Controllern und Programmiersprachen betrieben werden kann, eignet es sich sowohl für Einsteiger als auch für Roboter-Profis.

Hier die wichtigsten Features auf einen Blick:

- Das Board besitzt alle wichtigen Grundfunktionen die ein autonomer Roboter gewöhnlich benötigt. Sensoren und Motoren können direkt angeschlossen werden – Zusatzboards sind nicht notwendig, jedoch möglich. Denkbar wäre sogar die Kombination mehrerer RBNFRA-Boards – alle huckepack montiert und per I2C oder SPI kommunizierend.
- 2 Getriebemotoren + 1 Schrittmotor gleichzeitig anschließbar (auch größere bis max 2A Phasenstrom) oder nur 2 Schrittmotoren (max. 2A Phasenstrom) ohne Getriebemotoren (theoretisch noch weitere Motoren an den 8 Ausgängen (nennen wir Power-Port) anschließbar.
- Regelbare Schrittmotor Strombegrenzung (L297) , dadurch fast alle gängigen Schrittmotoren anschließbar (auch Schrittmotoren mit kleinen Nennspannungen wie z.B. 3V)
- Voll- und Halbschritt wählbar
- Bei voller Bestückung alles über Jumper oder Steckbrücken jederzeit um konfigurierbar (z.B. von Getriebemotoren auf Schrittmotoren und umgekehrt, ohne löten)
- 10 Servos können direkt angeschlossen werden (alternativ auch andere Dinge an den Servo Anschlüssen)
- 8 Eingangs- oder Ausgangsports über 4 Stiftleisten verfügbar (Sensoren etc.)
- 8 Power-Ausgangsports (max. 500 mA, wahlweise 5V oder 12V) über 4 Stiftleisten verfügbar (für Relais, Motoren oder Logik-Schaltungen)
- 8 Analog-Digitalwandler und weitere Ports über RNB-Stiftleiste verfügbar . Einer wird bereits für Messung der Batteriespannung genutzt
- Drehgeberanschluss für 2 Motoren (über AVR - Interrupt auswertbar)
- 5 LED's (4 zeigen den Zustand der ersten 4 Power Pin's an)
- Atmel AVR ATmega 16 on Board (optional Mega 32 oder Mega644)
- Programmierbarer AVR Co-Controller AT90S2313 on Board (entlastet Hauptcontroller) . In der Regel dient dieser zur Servo Steuerung – kann jedoch für andere Aufgaben umprogrammiert werden
- Betriebsspannung 6 bis 24 V (wahlweise getrennte Motorspannung)
- 5V und wahlweise 12 Stabilisierung!
- RNB-Bus (RoboterNetz-Bus zum Anschluss beliebiger Erweiterungen / Controller)
- Kompakter I2C-Bus(mit INT Leitung) zum Anschluss zahlreicher vorhandener Karten
- PC kompatible RS232 (MAX232 on Board) und wahlweise über Haupt- oder Co-Prozessor oder beide nutzbar
- Haupt- und Co-Controller sind über zahlreichen Programmiersprachen programmierbar. Wer nicht in Assembler programmieren möchte für den stehen im Internet kostenlose Open Source Entwicklungstools wie GCC (C-Compiler) oder nutzbare Demos wie [Bascom](#) (Basic Compiler) zur Verfügung.
- 6 genormte Bohrlöcher zur Befestigung und "Huckepack Montierung" weiterer Platinen
- Die Kommunikation zwischen Hauptcontroller und Co-Controller erfolgt über den I2C Bus. Über einen Jumper lässt sich ganz einfach zusätzlich auch eine RS232 und eine Interrupt-Verbindung zwischen Controller und Co-Controller einstellen.
- Alle Stecker sind nach der [Definition](#) vom Roboternetz ausgelegt
- Durch genaue Festlegung welcher Port für welche Funktion zuständig ist, lässt sich erstmals Software mit allen Boards austauschen die sich an die vereinbarten Definitionen halten
- Das Board verfügt über zwei Standard ISP-Anschlüsse sowohl für Hauptcontroller als auch CoController.
- Alle Bauteile sind über gängige Elektronikanbieter erhältlich. Bezugsquellen-Links findet man auf der Projektseite: <http://www.mikrocontroller-elektronik.de/>
- Eine professionelle Platine könnte man sich über die Eagle-Dateien, die auf der Projektseite zum Download bereitstehen, selber anfertigen lassen. Links zu entsprechenden Platinen-herstellern findet man auch unter <http://www.mikrocontroller-elektronik.de/>
- Neue Stromsparfunktionen – Ausgangsspannung auf Sensoranschlüssen und Aktoren kann per Software abgeschaltet werden
- Motorendstufe kann vollständig ausgeschaltet werden
- Sleep-Modus (Zusatzboards können in Schlafmodus (Stromsparmmodus) versetzt werden – dies muss jedoch vom Zusatzboard unterstützt werden)

- Ab Version 1.2 noch weitere 3 Status-LED's und 5 frei I/O Zusatzports
- Und das beste: Trotz aller Features ist dieses Board selbst bei voller Bestückung sehr preiswert realisierbar



Eagle-Dateien für Platinen Produktion oder eigene Veränderungen können auf der Projektseite heruntergeladen werden: <http://www.mikrocontroller-elektronik.de/>

Dort findet man auch Links zu Bezugsquellen der Bauteile!

Beschreibung der Anschlüsse

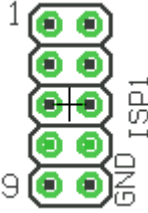
Bezeichnung auf der Platine	Funktion
<p>JP1</p>	<p>RNB-Bus (RoboterNetz-Bus) Diese Stiftleiste führt nahezu alle Controllerleitungen inkl. I2C,SPI,RX/TX und Spannungen. Hier können nahezu beliebige Erweiterungen über ein 50 poliges Flachkabel angeschlossen werden. An dem Flachkabel können mehrere Erweiterungsplatinen durchgeschleift werden.</p> <p>Wenn das ganze Board nur als Zusatzboard dienen soll, kann es über diesen Stecker mit einem anderen System verbunden werden.</p> <p>Die genaue Beschreibung des RNB-Bus finden Sie auf der nächsten Seite!</p>
<p>UREFR</p>	<p>Referenzspannung L297 messen Hier kann ein Multimeter angeschlossen werden wenn die Referenzspannung über den daneben liegenden Spindeltrimmer eingestellt werden soll. Die Referenzspannung bestimmt die Strombegrenzung beim rechten Schrittmotor.</p> <p>VRef berechnet sich nach folgender Formel:</p> <p style="text-align: center;">$V_{ref} = I_{motorstrom} * R_s$ (bei uns 0,51 Ohm)</p> <p>Angenommen Sie haben einen Motor der 0,5 A Strom verträgt, in diesem Fall würde Rechnung so aussehen:</p> <p style="text-align: center;">$V_{ref} = 0,5 * 0,51$ Ergebnis: $V_{ref} = 0,26 \text{ V}$</p> <p>Man muß also an dem Spindeltrimmer solange drehen, bis man an dieser Stiftleiste ca. 0,36V messen kann. In diesem Fall wird der Motor nie mit mehr als 0,5 A angesteuert, unabhängig welche Batteriespannung man verwendet. Somit lassen sich beispielsweise auch Schrittmotoren mit Nennspannung von 2 oder 3 V an Akkus mit 6 oder 12 Volt betreiben.</p>

JP5	<p>Stromversorgung für externe Erweiterungen An dieser 3 poligen Stiftleiste kann für beliebige Verbraucher die Spannung 5V/GND/12V abgegriffen werden. In der Mitte liegt GND (Masse). Immer aufpassen das der Stecker nicht falsch aufgesteckt wird!</p>
UREFL	<p>Referenzspannung L297 messen</p> <p>Hier kann ein Multimeter angeschlossen werden wenn die Referenzspannung über den daneben liegenden Spindeltrimmer eingestellt werden soll. Die Referenzspannung bestimmt die Strombegrenzung beim linken Schrittmotor.</p> <p>VRef berechnet sich nach folgender Formel:</p> <p style="text-align: center;">$V_{ref} = I_{motorstrom} * R_s$ (bei uns 0,51 Ohm)</p> <p>Angenommen Sie haben einen Motor der 0,5 A Strom verträgt, in diesem Fall würde Rechnung so aussehen:</p> <p style="text-align: center;">$V_{ref} = 0,5 * 0,51$ Ergebnis: $V_{ref} = 0,26 V$</p> <p>Man muß also an dem Spindeltrimmer solange drehen, bis man an dieser Stiftleiste ca. 0,26V messen kann. In diesem Fall wird der Motor nie mit mehr als 0,5 A angesteuert, unabhängig welche Batteriespannung man verwendet. Somit lassen sich beispielsweise auch Schrittmotoren mit Nennspannung von 2 oder 3 V an Akkus mit 6 oder 12 Volt betreiben. Hier schon einige wichtige Messwerte für die unterschiedlichen Stromstärken:</p>
JP19/Servos	<p>Servo Anschlüsse</p> <p>Diese beiden Stiftleisten sind so angeordnet das sich zehn 3 polige Servos direkt aufstecken lassen. Die Belegung entspricht den von Standard-Servos! Die Servos werden direkt vom Co-Prozessor gesteuert. Der Hauptprozessor muss dem Co-Prozessor nur die Positionsdaten der Servos übermitteln. Dies erleichtert die Programmierung zusätzlich. Bitte beachten das es auch einige Servos mit anderer Belegung gibt, hier wäre dann ein Adapter notwendig!</p>
Drehgeber	<p>Drehgeber zum messen der Wegstrecke</p> <p>An dieser Stiftleiste können zwei Drehgeber angeschlossen werden, gewöhnlich einer für das linke und einer für das rechte Rad. Üblich sind hier beispielsweise Gabellichtschranken oder Reflexlichtschranken die bei Drehbewegung des Rades 5V Impulse auslösen. Die Signale werden über diesen Stecker auf Port PD2(INT0) und PD3(INT1) geführt. Also zwei Interrupt fähige Ports des Hauptcontrollers. Der Hauptcontroller kann über Software somit die Wegstrecke messen als auch die unterschiedlichen Radgeschwindigkeiten ausgleichen damit eine recht genaue Geradeausfahrt möglich ist. Die Drehgebersignale werden auch auf dem RNB-Bus geführt, so das auch ein externer Controller diese Möglichkeit besitzt. Belegung des Steckers von Links: GND / 5V / PD2 (linker Drehgeber) / PD3 (rechter Drehgeber)</p>

<p>I2C-Bus</p>	<p>I2C-Bus Erweiterungskarten die nur den I2C-Bus benötigen können über diese 2x5 polige Stiftleiste angeschlossen werden. Die Belegung dieses Steckers entspricht weitgehend der Belegung, wie sie von einigen Elektor-Boards benutzt wird.</p> <p>Pin 1 SCL (Taktleitung) Pin 3 SDA (Datenleitung) Pin 5 +5V Pin 7 +5V Pin 9 Batteriespannung max. +12V Pin 2,4,6,8 GND Pin 10 INT Diese Leitung kann von allen I2C-Bus Erweiterungen genutzt werden um den Hauptcontroller darüber zu informieren das sich Daten (z.B. von Sensoren) verändert haben. In diesem Fall wird die Leitung solange auf Masse gelegt bis der rechtsprechende I2C-Baustein ausgelesen wird. Die Controller muss also immer alle I2C-Bausteine auslesen solange diese Leitung auf Masse liegt. Diese Leitung wird auf INT2 (PB2) des Hauptcontrollers geführt und steht auch am RBN-Bus zur Verfügung. Beim Elektro-Standard gibt es diese Leitung nicht, hier liegt dieses Signal immer auf Masse!</p>
<p>Aport1-2</p>	<p>Ausgangsport 1 + 2 Ausgangsport zum Anschluss beliebiger Aktoren (Relais, LED's usw.)</p> <p>Pin 1 Batteriespannung (max. 12 V) Pin 2 GND Pin 3 +5V (diese Spannung kann über PCF3 ausgeschaltet werden um Energie zu sparen) Pin 4 Port 1 des PCF1 Achtung je nach Stellung von Jumper JP7 liefert der Port 5 oder 12 V Ausgangsspannung. In beiden Fällen kann dieser mit maximal 500mA belastet werden. Relais etc. lassen sich direkt anschließen LED1 signalisiert den Zustand dieses Ports! Pin 5 Port 2 des PCF1 Achtung je nach Stellung von Jumper JP7 liefert der Port 5 oder 12 V Ausgangsspannung. In beiden Fällen kann dieser mit maximal 500mA belastet werden. Relais etc. lassen sich direkt anschließen LED3 signalisiert den Zustand dieses Ports!</p>
<p>Aport3-4</p>	<p>Ausgangsport 3 + 4 Ausgangsport zum Anschluss beliebiger Aktoren (Relais, LED's usw.)</p> <p>Pin 1 Batteriespannung (max. 12 V) Pin 2 GND Pin 3 +5V (diese Spannung kann über PCF3 ausgeschaltet werden um Energie zu sparen) Pin 4 Port 3 des PCF1 Achtung je nach Stellung von Jumper JP7 liefert der Port 5 oder 12 V Ausgangsspannung. In beiden Fällen kann dieser mit maximal 500mA belastet werden. Relais etc. lassen sich direkt anschließen LED2 signalisiert den Zustand dieses Ports! Pin 5 Port 4 des PCF1 Achtung je nach Stellung von Jumper JP7 liefert der Port 5 oder 12 V Ausgangsspannung. In beiden Fällen kann dieser mit maximal 500mA belastet werden. Relais etc. lassen sich direkt anschließen LED4 signalisiert den Zustand dieses Ports!</p>

<p>APort5-6</p>	<p>Ausgangsport 5 + 6 Ausgangsport zum Anschluss beliebiger Aktoren (Relais, LED's usw.)</p> <p>Pin 1 Batteriespannung (max. 12 V) Pin 2 GND Pin 3 +5V (diese Spannung kann über PCF3 ausgeschaltet werden um Energie zu sparen) Pin 4 Port 5 des PCF1 Achtung je nach Stellung von Jumper JP7 liefert der Port 5 oder 12 V Ausgangsspannung. In beiden Fällen kann dieser mit maximal 500mA belastet werden. Relais etc. lassen sich direkt anschließen Pin 5 Port 6 des PCF1 Achtung je nach Stellung von Jumper JP7 liefert der Port 5 oder 12 V Ausgangsspannung. In beiden Fällen kann dieser mit maximal 500mA belastet werden. Relais etc. lassen sich direkt anschließen</p>
<p>APort7-8</p>	<p>Ausgangsport 7 + 8 Ausgangsport zum Anschluss beliebiger Aktoren (Relais, LED's usw.)</p> <p>Pin 1 Batteriespannung (max. 12 V) Pin 2 GND Pin 3 +5V (diese Spannung kann über PCF3 ausgeschaltet werden um Energie zu sparen) Pin 4 Port 7 des PCF1 Achtung je nach Stellung von Jumper JP7 liefert der Port 5 oder 12 V Ausgangsspannung. In beiden Fällen kann dieser mit maximal 500mA belastet werden. Relais etc. lassen sich direkt anschließen Pin 5 Port 8 des PCF1 Achtung je nach Stellung von Jumper JP7 liefert der Port 5 oder 12 V Ausgangsspannung. In beiden Fällen kann dieser mit maximal 500mA belastet werden. Relais etc. lassen sich direkt anschließen</p>
<p>Port1-2</p>	<p>Eingangsport 1 + 2 Eingangsport zum Anschluss beliebiger Sensoren (Mikroschalter, IR-Sensoren usw..)</p> <p>Pin 1 Batteriespannung (max. 12 V) Pin 2 GND Pin 3 +5V (diese Spannung kann über PCF3 ausgeschaltet werden um Energie zu sparen) Pin 4 Port 1 des PCF U\$4 Empfohlen für Hinderniserkennung ganz Links Vorne Pin 5 Port 2 des PCF U\$2 Empfohlen für Hinderniserkennung Links Vorne</p>
<p>Port3-4</p>	<p>Eingangsport 3 + 4 Eingangsport zum Anschluss beliebiger Sensoren (Mikroschalter, IR-Sensoren usw..)</p> <p>Pin 1 Batteriespannung (max. 12 V) Pin 2 GND Pin 3 +5V (diese Spannung kann über PCF3 ausgeschaltet werden um Energie zu sparen) Pin 4 Port 3 des PCF U\$4 Empfohlen für Hinderniserkennung ganz Rechts Vorne Pin 5 Port 4 des PCF U\$2 Empfohlen für Hinderniserkennung Rechts Vorne</p>
<p>Port5-6</p>	<p>Eingangsport 5 + 6 Eingangsport zum Anschluss beliebiger Sensoren (Mikroschalter, IR-Sensoren usw..)</p> <p>Pin 1 Batteriespannung (max. 12 V) Pin 2 GND Pin 3 +5V (diese Spannung kann über PCF3 ausgeschaltet werden um Energie zu sparen) Pin 4 Port 5 des PCF U\$4 Pin 5 Port 6 des PCF U\$2</p>

Port7-8	<p>Eingangsport 7 + 8 Eingangsport zum Anschluss beliebiger Sensoren (Mikroschalter, IR-Sensoren usw..)</p> <p>Pin 1 Batteriespannung (max. 12 V) Pin 2 GND Pin 3 +5V (diese Spannung kann über PCF3 ausgeschaltet werden um Energie zu sparen) Pin 4 Port 7 des PCF U\$4 Pin 5 Port 8 des PCF U\$2</p>
MOTOR(EN)LINKS	<p>Linker Schrittmotor oder beide Getriebemotoren Je nach Stellung von Jumper JP3 (und dem Vorhandensein von IC 4) kann über diesen Anschluss ein Schrittmotor oder zwei Getriebemotoren angeschlossen werden.</p> <p>Im Schrittmotormodus sind die Klemmen wie folgt belegt: Klemme 1 und 2 = Wicklung 1 eines bipolaren Schrittmotors Klemme 3 und 4 = Wicklung 2 eines bipolaren Schrittmotors</p> <p>Im Getriebemotormodus sind die Klemmen wie folgt belegt: Klemme 1 und 2 = Motor 1 Klemme 3 und 4 = Motor 2</p> <p>Die Motorendstufe kann per Software deaktiviert werden um Energie zu sparen.</p>
Stepperrechts	<p>Rechter Schrittmotor Klemme 1 und 2 = Wicklung 1 eines bipolaren Schrittmotors Klemme 3 und 4 = Wicklung 2 eines bipolaren Schrittmotors</p>
Power	<p>Schraubklemmen zur Stromversorgung (Akku) des Boards</p> <p>Pin 3 GND Pin 2 Versorgungsspannung 6 bis ca. 15 Volt Pin 1 Motorspannung – Achtung: Dieser darf nur dann beschaltet werden wenn Jumper 18 entfernt wurde. In diesem Fall kann hier eine getrennte Motorspannung von 3 bis 24 Volt eingespeist werden. In den meisten Fällen kann man diesen Anschluss unbeschaltet lassen (JP 18 muss dann gesteckt sein).. In dem Fall würde die normale Batteriespannung auch für die Motoren genutzt.</p>
RS232	<p>PC kompatible RS232 Schnittstelle Über ein Adapterkabel kann die serielle Schnittstelle des PC direkt mit dem Controller und Co-Controller verbunden werden. Die Belegung ist kompatibel zum Conrad Roboter CCRP5:</p> <p>Pin 1 RX Pin 2 GND Pin 3 TX</p> <p>Ein geeignetes Anschlusskabel kann schnell selbst angefertigt werden .</p>

<p>ISP 1</p> 	<p>ISP – IN SYSTEM PROGRAMMING</p> <p>Über diesen Anschluss kann der Hauptcontroller mit einem Standard ISP-Kabel direkt an einen Parallelport des PC's angeschlossen und programmiert werden. Die Belegung des ISP-Anschlusses ist zu dem weit verbreitetet STK200 Programmier Dongle kompatibel. Ein entsprechender Dongle kann man sich entweder selber basteln (siehe Artikel „ARV Einstieg leicht gemacht“ unter RN-Wissen) . Bezugsquellen für moderne USB-Programmer finden Sie auch auf der Seite http://www.mikrocontroller-elektronik.de/</p> <p>Pin 1 MOSI Pin 2 VCC Pin 3 Nicht belegt Pin 4 GND Pin 5 RESET Pin 6 GND Pin 7 SCK Pin 8 GND Pin 9 MISO Pin 10 GND</p> <p>Achtung: Stecker nicht falsch herum einstecken.</p>
<p>ISP 2</p>	<p>ISP – IN SYSTEM PROGRAMMING</p> <p>Über diesen Anschluss kann der Co-Controller mit einem Standard ISP-Kabel direkt an einen Parallelport des PC's angeschlossen und programmiert werden. Die Belegung des ISP-Anschlusses ist zu dem weit verbreitetet STK200 Programmier Dongle kompatibel. Ein entsprechender Dongle kann man sich entweder selber basteln (siehe Artikel „ARV Einstieg leicht gemacht“ unter RN-Wissen) . Bezugsquellen für moderne USB-Programmer finden Sie auch auf der Seite http://www.mikrocontroller-elektronik.de/</p> <p>Achtung: Vor dem nutzen des Steckers müssen alle Servo-Stecker entfernt werden!</p> <p>Pin 1 MOSI Pin 2 VCC Pin 3 Nicht belegt Pin 4 GND Pin 5 RESET Pin 6 GND Pin 7 SCK Pin 8 GND Pin 9 MISO Pin 10 GND</p> <p>Achtung: Stecker nicht falsch herum einstecken.</p>
<p>JP12</p>	<p>Extension I/O-Port</p> <p>Diese fünf I/O-Ports sind im Board Version 1.2 hinzugekommen und stehen dem Anwender zur freien Verfügung.</p> <p>Die Belegung von Links nach rechts Pin 1 P3 von PCF3 Pin 2 P4 von PCF3 Pin 3 P5 von PCF3 - zudem mit Status LED verbunden Pin 4 P6 von PCF3 - zudem mit Status LED verbunden Pin 5 P7 von PCF3 - zudem mit Status LED verbunden</p>

RNB-Bus (RoboterNetz-Bus Belegung)

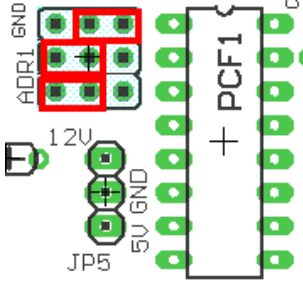
Pin	RoboterNetz-Bus	Funktion	AVR-Mega Pin des Grundboards	Empfehlung wenn Board über C-Control C-Control I Pin
1	5V	Logik Spannung	VCC, Pin 10	5V, A20
2	5V	Logik Spannung (doppelt wegen querschnitt)		
3	Reset	Controller Reset	Reset , Pin 9	Reset, B19
4	GND	Masse	GND, Pin 11	GND, A1
5	I2C Bus SCL	I2C Bus SCL	SCL, PC0, Pin 22	SCL, B12
6	I2C Bus SDA	I2C Bus SDA	SDA, PC1, Pin 23	SDA, B11
7	GND	Masse		
8	SPI Port SS	SPI Port	PB4, SS, Pin 5	Nicht vorhanden
9	SPI Port MOSI	SPI Port	PB5, MOSI, Pin 6	Nicht vorhanden
10	SPI Port MISO	SPI Port	PB6, MISO, Pin 7	Nicht vorhanden
11	SPI Port SCK	SPI Port	PB7, SCK, Pin 8	Nicht vorhanden
12	GND	Masse		
13	AD-Port 0 Entfernung Links	IR-Entfernungsmesser, Links vorne	PA0, ADC0, Pin 40	AD 1, A3
14	AD-Port 1 Entfernung Mitte	IR-Entfernungsmesser, Mitte vorne	PA1, ADC1, Pin 39	AD 2, A4
15	AD-Port 2 Entfernung Rechts	IR-Entfernungsmesser, Rechts vorne	PA2, ADC2, Pin 38	AD 3, A5
16	AD-Port 3 Entfernung Boden	IR-Entfernungsmesser, Bodenabstand vorne	PA3, ADC3, Pin 37	AD 4, A6
17	AD-Port 4 Batteriespannung	Batteriespannung Überwachung. Ein Spannungsteiler befindet sich bereits auf dem Board RBNFRA	PA4, ADC4, Pin 36	AD 5, A7
18	AD-Port 5 Tastatur	Reserviert für Tastatur mit Widerstandsnetzwerk	PA5, ADC5, Pin 35	AD 6, A8
19	AD-Port 6		PA6, ADC6, Pin 34	AD 7, A9
20	AD-Port 7		PA7, ADC7, Pin 33	AD 8, A10
21	GND	Masse		
22	Getriebemotor PWM Links	Motorgeschwindigkeit linker Motor	Port PD4, OC1B, Pin 18	DA-Ausgang 2, B17
23	GND	Masse		
24	Getriebemotor Links Kanal 1	Kanal 1 und Kanal 2 bestimmen Richtung	PC6, TOSC1, Pin28	Port 6, B7
25	Getriebemotor Links Kanal 2	Kanal 1 und Kanal 2 bestimmen Richtung	PC7, TOSC2, Pin29	Port 5, B6
26	GND	Masse		
27	Getriebemotor PWM Rechts	Motorgeschwindigkeit rechter Motor	Port PD5, OC1A, Pin 19	DA-Ausgang 1, B17
28	GND	Masse		
29	Getriebemotor Rechts Kanal 1	Kanal 1 und Kanal 2 bestimmen Richtung	PB0, XCK/T0, Pin 1	Port 1, B2
30	Getriebemotor Rechts Kanal 2	Kanal 1 und Kanal 2 bestimmen Richtung	PB1, T1, Pin 2	Port 2, B3
31	I2C-Eingang Interrupt	LOW wenn an I2C-Bus/Porterweiterung Signal wechselt	PB2, INT2, Pin 3	Port 3, B4
32	GND	Masse		
33	Drehgeber Links	Drehgeber für Raddrehung links	Port PD2, INT0, Pin 16	Port 9, A12 oder alternativ Zähler auf der Controller-Adapterplatine vorschalten
34	Drehgeber Rechts	Drehgeber für Raddrehung rechts	Port PD3, INT1, Pin17	Port 10, A13 oder alternativ Zähler auf der Controller-Adapterplatine vorschalten
35	Schrittmotoren Ein/Aus		Port PD6, ICP1, Pin 20	Port 11, A14
36	Schrittmotor links - Richtung	Die Drehrichtung des linken Schrittmotor	Port PC5, TDI, Pin27	Port 12, A15
37	Schrittmotor rechts - Richtung	Die Drehrichtung des rechten Schrittmotor	Port PC4, TDO, Pin26	Port 13, A16
38	GND	Masse		
39	Schrittmotor links Step	Ein Impuls bewegt Schrittmotor um 1 Schritt	Port PC3, TMS, Pin 25	Port 14, A17
40	Schrittmotor rechts Step	Ein Impuls bewegt Schrittmotor um 1 Schritt	Port PC2, TCK, Pin 24	Port 15, A18
41	GND	Masse		
42	RS232 TX	Serielle Schnittstelle TTL-Pegel	Port PD1, TXD, Pin 15	
43	RS232 RX	Serielle Schnittstelle TTL-Pegel	Port PD0, RXD, Pin 14	
44	RS485 A	Wird nicht auf RBNFRA1.1 genutzt	Nicht belegt	Nicht belegt
45	RS485 B	Wird nicht auf RBNFRA1.1 genutzt	Nicht belegt	Nicht belegt
46	RS485 Master	Externe RS485 TX/RX Umschaltung	PB3, OC0, Pin 4	Port 4, B5
47	Reserve		PD7, OC2, Pin21	Port 7, B8
48	Sleepmodus	Ist Low, wenn Strom gespart werden soll	PCF3	Port 8, B9
49	12 Volt	12 V für Sensoren etc.		
50	12 Volt	Nochmal 12 V um Leitungswiderstand zu verringern		

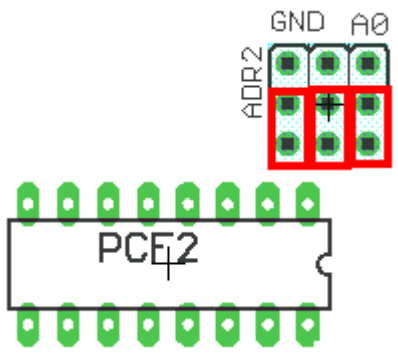
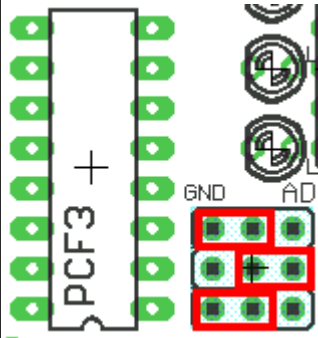
Die gelb markierten Leitungen werden bereits auf dem Grundboard RBNFRA1.2 genutzt oder über andere Stecker bereitgestellt

Beschreibung der Jumper / Kurzschlussbrücken

Wichtig: Alle Jumper (Kurzschlussbrücken) sollten nur umgesteckt werden wenn das Board nicht unter Spannung steht!

Bezeichnung auf Platine	Funktion
JP2	<p>12 Volt Spannungsregler überbrücken Wenn die Betriebsspannung des Boards kleiner als 14 V ist, dann macht es keinen Sinn einen 12 V Spannungsregler zu verwenden. In diesem Fall muss dieser Kurzschlussstecker gesetzt sein. In diesem Fall wird auf allen Ausgängen wo 12 V angegeben ist, die volle Batteriespannung anliegen.</p>
JP3	<p>Getriebemotoren-Modus aktivieren Wenn an das Board Getriebemotoren angeschlossen werden sollen , dann müssen hier alle 6 Kurzschlussbrücken eingesetzt werden. Wichtig: Das IC 4 (L297) muß vorher aus der Fassung entfernt werden, es wird im Getriebemotormodus nicht benötigt und muss entfernt werden. Die Getriebemotoren werden an den Schraubklemmen „Motore(en)Links“ an Pin 1+2 und 3+4 angeschlossen. An den Schraubklemmen „Stepperrechts“ kann in jedem Modus ein Schrittmotor angesteuert werden. Werden alle 6 Kurzschlussbrücken von JP3 entfernt, so wird an der Schraubklemme „Motore(en)Links“ auch ein Schrittmotor angeschlossen.</p>
JP6	<p>Eine Spannung für Elektronik und Motoren Gewöhnlich wird die eine Spannung für Motoren und Elektronik verwendet da man nur ein Akku anschließt. In diesem Fall muß diese Kurzschlussbrücke gesetzt werden. In diesem Fall wird die Volle Betriebsspannung für die Motoren genutzt. Der Pin 1 der „Power“-Schraubklemme darf dann nicht genutzt werden In bestimmten Fällen kann es Sinn machen die Motoren mit einer anderen Spannung zu betreiben. Zum Beispiel kann man mit höheren Spannungen mehr Leistung/Kraft aus Schrittmotoren herausholen. In diesem Fall muss diese Kurzschlussbrücke entfernt werden. An Pin 1 der „Power“-Schraubklemme kann dann bis zu 24 Volt zugeführt werden. Diese Spannung wird dann nur für die Motoren (Getriebe- als auch Schrittmotoren) genutzt.</p>
JP7	<p>Ausgangsspannung der Ausgangsports Über diesen Kurzschlussstecker wird festgelegt welche Spannung die Datenleitungen bei den Ausgangsports auf den Steckern APORTx-x führen sollen. Möchte man hier Logikbausteine anschließen, so muss hier ein Kurzschlussstecker zwischen mittleren Pin und dem linken 5V Pin eingesteckt werden. Dies ist die gewöhnliche Konfiguration! Möchte man dagegen auf diesen Datenleitungen 12V (bzw. volle Batteriespannung) ausgeben, dann muss hier ein Kurzschlussstecker den mittleren Pin mit dem rechten 12V Pin verbinden. Achtung, in diesem Fall darf man keine 5V TTL Logikbausteine ansteuern.. In beiden Betriebsarten können die Ausgangsports mit bis zu 500mA belastet werden. Bedenken sollte man jedoch das die Gesambelastung die Leistung des Spannungsreglers als auch 1,5A nicht überschreiten sollte.</p>
JP8	<p>Voll- oder Halbschrittmodus Rechts Verbindet man den mittleren Pin mit dem oberen Pin „HALB“ so wird der rechte Schrittmotor immer halbe Schritte bei jedem Impuls bewegt. Ein Motor mit 200 Schritten pro Umdrehung benötigt dann 400 Impulse. Verbindet man den mittleren Pin mit dem unteren Pin „VOLL“ so wird der rechte Schrittmotor immer volle Schritte bei jedem Impuls bewegt. Ein Motor mit 200 Schritten pro Umdrehung benötigt dann auch 200 Impulse. Dies ist die Standardkonfiguration.</p>
JP9	<p>Voll- oder Halbschrittmodus Links Verbindet man den mittleren Pin mit dem oberen Pin „HALB“ so wird der linke Schrittmotor immer halbe Schritte bei jedem Impuls bewegt. Ein Motor mit 200 Schritten pro Umdrehung benötigt dann 400 Impulse. Verbindet man den mittleren Pin mit dem unteren Pin „VOLL“ so wird der linke Schrittmotor immer volle Schritte bei jedem Impuls bewegt. Ein Motor mit 200 Schritten pro Umdrehung benötigt dann auch 200 Impulse. Dies ist die Standardkonfiguration</p>

<p>ADR1</p>	<p>Adresse des I2C Ausgabeportes Hier müssen beim Board RBNFRA 1.2 (anders beim Vorgänger) nur Jumper auf eine 3x3 Stiftleiste aufgesteckt werden. Die drei mal 3 Stiftleiste kann man sich übrigens durch drei einzelne 3er Stiftleisten bauen! Über die Jumper (Kurzschlussbrücken) kann man die Ansprechadresse des PCF-Ausgabeportes (wir nennen es Power-Port wegen der 500mA) frei wählen. Dadurch lassen sich mehrere Bords und PCF-Bausteine an einem I2C-Bus nutzen. Die Nachfolgende Skizze zeigt übliche Standardbelegung Hex 72!</p>  <p>Als Standard wird hier PCF-Adr. Hex 72 empfohlen (letzte Skizze)</p>
<p>JP16</p>	<p>Co-Controller Verbindung Der Co-Controller kommuniziert gewöhnlich per I2C mit dem Hauptcontroller. Und hat ansonsten keinerlei Verbindung zum Hauptcontroller. Durch 3 Kurzschlussstecker können weitere Verbindungen und Möglichkeiten geschaffen werden:</p> <p>Durch einen Kurzschlussstecker ganz oben (INT) kann PD0(RX) des Co-Controllers mit PB2(INT2) des Hauptcontrollers verbunden werden. PB2(INT2) ist interrupt fähig und dient zur Erfassung von Änderungen auf einem I2C-Baustein. Dies kann hilfreich sein wenn der Co-Controllers zur Sensor- oder Datenerfassung umprogrammiert wird.</p> <p>Durch einen Kurzschlussstecker in der Mitte (RX) kann PD0(RX) des Co-Controllers mit PD0(RX) Hauptcontrollers verbunden werden. Dadurch ist der Co Controller auch mit dem Baustein Max232 verbunden und kann über die serielle Schnittstelle kommunizieren. Hierbei ist darauf zu achten das natürlich nicht beide Controller gleichzeitig kommunizieren.</p> <p>Durch einen Kurzschlussstecker ganz unten (TX) kann PD1(TX) des Co Controllers mit PD1(TX) Hauptcontrollers verbunden werden. Dadurch ist der Co-Controller auch mit dem Baustein Max232 verbunden und kann über die serielle Schnittstelle kommunizieren. Hierbei ist darauf zu achten das natürlich nicht beide Controller gleichzeitig kommunizieren.</p> <p>Gewöhnlich sind auf JP16 keinerlei Kurzschlussbrücken gesetzt!</p>

<p>ADR2</p>	<p>Adresse des I2C Eingangsportes Hier müssen beim Board RBNFRA 1.2 (anders beim Vorgänger) nur Jumper auf eine 3x3 Stiftleiste aufgesteckt werden. Die drei mal 3 Stiftleiste kann man sich übrigens durch drei einzelne 3er Stiftleisten bauen! Über diese Brücken kann man die Ansprechadresse des PCF-Eingangsportes frei wählen. Dadurch lassen sich mehrere Bords und PCF-Bausteine an einem I2C-Bus nutzen. Die Nachfolgende Skizze zeigt die Standard Belegung</p>  <p style="text-align: center;">Als Standard wird hier PCF-Adr. Hex 7E empfohlen (letzte Skizze)</p>
<p>ADR3</p>	<p>Adresse des I2C Extensionports (auch EnergiePort genannt) Über diese Brücken kann man die Ansprechadresse des PCF-Extensionports frei wählen. Dadurch lassen sich mehrere Bords und PCF-Bausteine an einem I2C-Bus nutzen. Die Nachfolgende Skizze zeigt die Standard Belegung.</p> <p>Dieser Port ist für die Status-LED's als auch einige Energiesparfunktionen zuständig. So kann über diesen Port wahlweise die Logikspannung der Motorendstufe, oder die Ein- und Ausgabeport Logikspannung ein und ausgeschaltet werden. Zudem kann das Sleep-Bit am Roboternetz-Bus gesetzt oder gelöscht werden.</p>  <p style="text-align: center;">Als Standard wird hier PCF-Adr. Hex 74 empfohlen (letzte Skizze)</p>

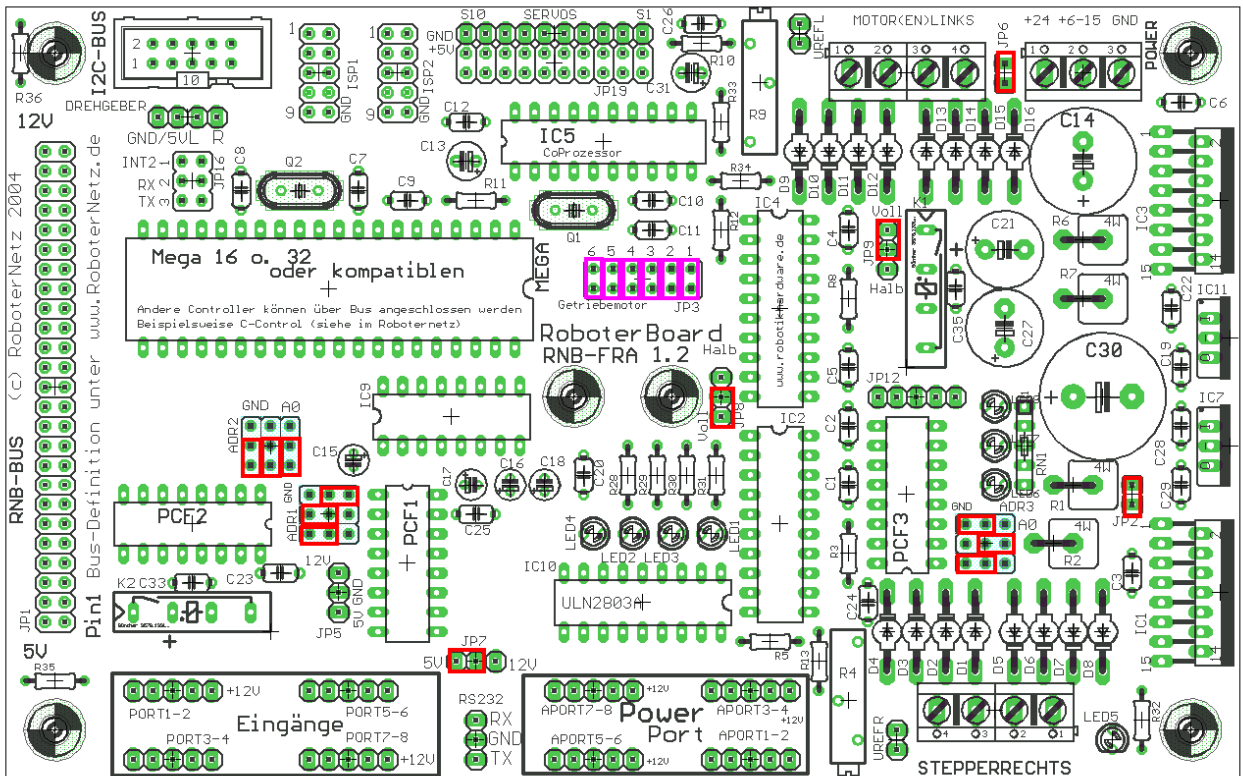
Bauteile – Bestückungsliste /Bestellliste

Die aktuelle Liste der Bauteile und von möglichen Bezugsquellen findet ihr stets unter folgendem [Link](#):

<http://www.mikrocontroller-elektronik.de/rnbfra-roboterboard-mit-motortreibern/>

Die Eagle-Dateien für Platine können über die [Projektseite](#) geladen werden.

Bestückungsliste (unbedingt beim bestücken ausdrucken)



Kurzschlussbrücken sind oben rötlich eingezeichnet (Getriebemodus)

Bezeichnung auf Platine	Wert / Beschreibung	Auch notwendig bei Verzicht auf Schrittmotoren?
C1	3,3n	NEIN
C2	100n	NEIN
C3	100n	NEIN
C4	3,3n	JA
C5	100n	JA
C6	100n	JA
C7	22pf	JA
C8	22pf	JA
C9	100n	JA
C10	22pf	JA
C11	22pf	JA
C12	100n	JA
C13	10uF	JA
C14	1000uF	JA
C15	1 uF oder 4,7uF	JA
C16	1 uF oder 4,7uF	JA
C17	1 uF oder 4,7uF	JA
C18	1 uF oder 4,7uF	JA
C19	100n	JA
C20	100n	JA
C21	220uF	JA
C22	100n	JA
C23	100n	JA
C25	100n	JA
C26	100n	JA
C27	220 uF	JA
C28	100n	JA
C29	100n	JA
C30	2200 uF	JA
C31	10 uF	JA

C32	Nicht bestücken!	
C33	100 nF	JA
C34	Nicht bestücken!	JA
C35	100 nF	JA
D1	BYV27/200	NEIN
D2	BYV27/200	NEIN
D3	BYV27/200	NEIN
D4	BYV27/200	NEIN
D5	BYV27/200	NEIN
D6	BYV27/200	NEIN
D7	BYV27/200	NEIN
D8	BYV27/200	NEIN
D9	BYV27/200	JA
D10	BYV27/200	JA
D11	BYV27/200	JA
D12	BYV27/200	JA
D13	BYV27/200	JA
D14	BYV27/200	JA
D15	BYV27/200	JA
D16	BYV27/200	JA
20 pol Fassung für IC2	20 pol Fassung für L297	Nein
20 pol Fassung für IC4	20 pol Fassung für L297	JA
20 pol Fassung für IC5	20 pol Fassung für AT 90S2313	JA
16 pol Fassung für IC9	16 pol Fassung für MAX232	JA
18 pol Fassung für IC10	18 pol Fassung für ULN2803A	JA
40 pol Fassung für Mega	40 pol Fassung für ATMEGA 16	JA
16 pol Fassung für PCF1	16 pol Fassung für PCF8574 AP	JA
16 pol Fassung für PCF2	16 pol Fassung für PCF8574 AP	JA
16 pol Fassung für PCF3	16 pol Fassung für PCF8574 AP	JA
IC1	L298	NEIN
IC2	L297	NEIN
IC3	L298	JA
IC4	L297	NEIN
IC5	Atmel AT 90S2313	JA (bei Verzicht auf Servo Anschlüsse kann dieser weggelassen werden)
IC7	7812 oder LM2940CT12	Ist nur notwendig, wenn die Betriebsspannung der Elektronik über 12 V liegt
IC9	MAX232	Ja (bei Verzicht auf RS232 kann dieser weggelassen werden)
IC10	ULN2803A	JA
IC11	78S05 (2A belastbar, Eingangsp. ab 8V) oder LM2940 C5 (1A, Eingangsp. ab 6V)	JA
MEGA	ATMEGA 16 oder Mega 32	JA (Wenn an den RNB-Bus (JP1) anderer Controller angeschlossen wird, kann dieser weggelassen oder als Coprozessor genutzt werden)
PCF1	PCF8574 P	JA
PCF2	PCF8574 P	JA
PCF3	PCF8574 P	JA
ISP1	Stiftleiste 2x5	JA
ISP2	Stiftleiste 2x5	Nur notwendig wenn der Coprozessor genutzt und noch nicht programmiert ist
JP1	Stiftleiste 2x25	JA
JP2	Jumper Stiftleiste 1x2	Ja (wenn IC7 genutzt wird, kann er weggelassen werden)
JP3	Jumper Stiftleiste 2x6	JA
UREFR	Stiftleiste 1x2	NEIN
JP5	Stiftleiste 1x3	Ja
JP6	Jumper Stiftleiste 1x2	Ja (wenn für Motoren

		getrennte Spannung angelegt wird, dann kann dieser weggelassen werden)
JP7	Jumper Stiftleiste 1x3	JA
JP8	Jumper Stiftleiste 1x3	NEIN
JP9	Jumper Stiftleiste 1x3	NEIN
JP12	Jumper Stiftleiste 1x5	NEIN
ADR1	Kontaktbuchse 3x3 für Brücken (kann man auch aus 3 einreihigen Stiftleisten oder einer einreihigen und einer zweireihigen zusammensetzen, siehe Bild unten) Stiftl. 1x3 und Stiftl. 2x3	JA
JP16	Jumper Stiftleiste 2x3	JA (bei Verzicht auf Coprozessor(Servos) kann dies weggelassen werden)
ADR2	Kontaktbuchse 3x3 für Brücken (kann man auch aus 3 einreihigen Stiftleisten oder einer einreihigen und einer zweireihigen zusammensetzen, siehe Bild unten) Stiftl. 1x3 und Stiftl. 2x3	JA
ADR3	Kontaktbuchse 3x3 für Brücken (kann man auch aus 3 einreihigen Stiftleisten oder einer einreihigen und einer zweireihigen zusammensetzen, siehe Bild unten) Stiftl. 1x3 und Stiftl. 2x3	JA
UREFL	Stiftleiste 1x2	JA
JP19	Stiftleiste 2 x 10	JA (bei Verzicht auf Coprozessor(Servos) kann dies weggelassen werden)
Drehgeber	Stiftleiste 1x4	JA
I2C-Bus	Wannenbuchse 2x5 gerade	JA
APort1-2	5 pol Stiftleiste	JA
APort2-4	5 pol Stiftleiste	JA
APort5-6	5 pol Stiftleiste	JA
APort7-8	5 pol Stiftleiste	JA
Port1-2	5 pol Stiftleiste	JA
Port2-4	5 pol Stiftleiste	JA
Port5-6	5 pol Stiftleiste	JA
Port7-8	5 pol Stiftleiste	JA
MOTOR(EN)LINKS	Schraubklemmen 4er (oder Reichelt AKL230-04)	JA
Stepperrechts	Schraubklemmen 4er (oder Reichelt AKL230-04)	Nein
Power	Schraubklemmen 3er (oder Reichelt AKL230-03)	JA
RS232	Stiftleiste 1x3	Ja (bei Verzicht auf RS232 kann dieser weggelassen werden)
SERVOS	Stiftleiste 1x10	
LED1	Leuchtdiode 3mm Grün	JA
LED2	Leuchtdiode 3mm Gelb	JA
LED3	Leuchtdiode 3mm Grün	JA
LED4	Leuchtdiode 3mm Rot	JA
LED5	Leuchtdiode 3mm Grün	JA
LED6	Leuchtdiode 3mm Rot	JA
LED7	Leuchtdiode 3mm Gelb	JA
LED8	Leuchtdiode 3mm Grün	JA
Q1	Quarz 4 Mhz	JA (bei Verzicht auf Coprozessor(Servos) kann dies weggelassen werden)
Q2	Quarz 8 Mhz oder 7,3728 Mhz	JA
R1	0,5 Ohm 4-5 Watt	NEIN
R2	0,5 Ohm 4-5 Watt	NEIN
R3	22k	JA
R4	10k Spindeltrimmer	NEIN
R5	6,2K	NEIN
R6	0,5 Ohm 4-5 Watt	Ja
R7	0,5 Ohm 4-5 Watt	Ja
R8	22k	JA
R9	10k Spindeltrimmer	Ja
R10	6,2K	JA

R11	10K	JA
R12	10K	JA
R13	10K	JA
R28	680	JA
R29	680	JA
R30	680	JA
R31	680	JA
R32	330	JA
R33	5,1k	JA
R34	22k (Alternativ 10 K, wenn andere Batteriespannungsberechnung als im Beispiel verwendet wird)	JA
R35	10K	JA
R36	10K	JA
RN1	Widerstandsnetzwerk 330 Ohm bis ca. 1 Kohm (auf Einbaurichtung achten, Punkt oben neben grünen LED)	JA
K1	Reed Relais (auf Polung achten)	JA
K2	Reed Relais (auf Polung achten)	JA

Aufbauanleitung

Durch die gedruckte Schaltung (Platine) ist der Aufbau des Boards nicht schwieriger als herkömmliche Bausätze. Für den Aufbau von Mikrocontroller Schaltungen sollte jedoch schon etwas Erfahrung im Umgang mit dem LötKolben vorhanden sein.

In jedem Fall ist ein LötKolben von 20 bis 30 Watt mit spitzer LötKolbenspitze empfehlenswert. Stärkere LötKolben können die Bauteile beschädigen. **Gelötet wird ausschließlich auf der Unterseite der Platine (die Seite die nicht bedruckt ist).**

Die Bauteile werden ausschließlich von der bedruckten Seite in das Board gesteckt. Hier ist in jedem Fall eine kleine Spitzzange und Seitenschneider notwendig. Die meisten Bauteile wie Kondensatoren, Schaltkreise etc. passen genau in die Bohrungen. Einige andere Bauteile wie Widerstände und Dioden müssen stets mit der Spitzzange gebogen werden. Da das Board wegen der zahlreichen Features recht kompakt bestückt werden musste, sind einige Bauteile wie die Widerstände sehr eng am Widerstandskörper abzubiegen (je nachdem über welchen Hersteller diese Teile bezogen wurden). Einsteiger sollten hier unbedingt darauf achten das dabei das Bauelement nicht zu stark belastet und so beschädigt wird.

Je nach Erfahrung kann der Aufbau schon einige Stunden in Anspruch nehmen wenn man sehr sorgfältig vorgeht.

Für den Aufbau sollte man sich möglichst dieses komplette Dokument ausdrucken. Sehr wichtig ist der Bestückungsplan und die Stückliste. Auf der Platine sind alle Bauteilbezeichnungen aufgedruckt, ein Blick in die Stückliste beschreibt das Bauteil genauer. In der Regel ist die Beschriftung der Platineseite in Verbindung mit der Stückliste, zum korrekten Aufbau ausreichend. Es gibt jedoch einige Stellen auf der Platine wo der Bestückungsdruck aufgrund von vielen Bohrlöchern und Durchkontaktierungen nicht eindeutig ist. Daher sollte man im Zweifel immer einmal einen Blick in den ausgedruckten Bestückungsplan werfen. Dort ist genau erkennbar wo welches Bauteil wie herum gehört.

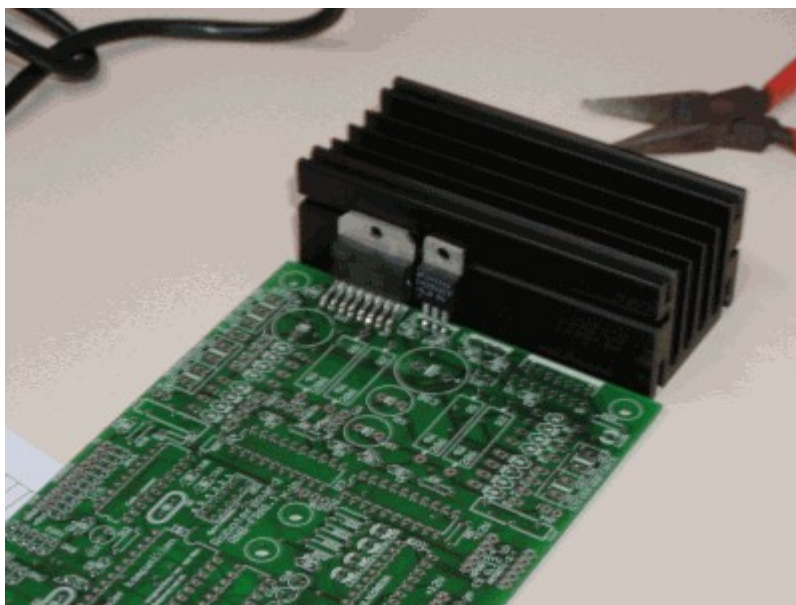
Besonders acht sollte man bei den Elkos C13, C14, C15, C16, C17, C18, C21, C27, C30,C31 geben, diese müssen richtig herum eingelötet werden, die Polarität ist auf der Platine, im Bestückungsplan und auf dem Bauteil erkennbar.

Wo fängt man an?

Dies ist im Grunde egal. Zuerst sollte man jedoch überdenken was man bestücken möchte. Am besten aber auch am teuersten ist es natürlich wenn man alles bestückt. Möchte man aber in keinem Fall Schrittmotoren benutzen, dann kann man einen Teil der Bauelemente weglassen und dadurch Kosten und Strom sparen. Welcher Teil in diesem Fall weggelassen werden kann, ist in der letzten Spalte der Bestückungsliste erkennbar. Ich persönlich empfehle jedoch möglichst alles zu bestücken, denn dadurch erhält man ein universelles Roboterboard mit dem man fast alles machen kann.

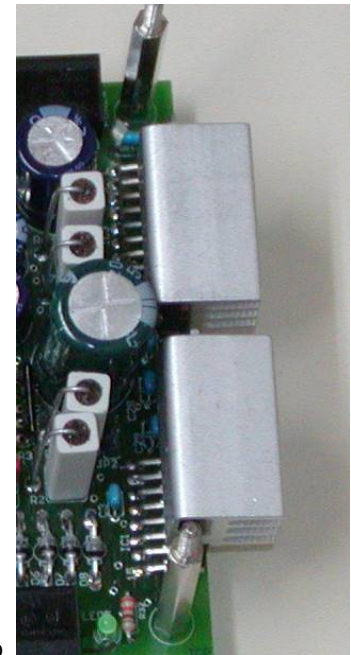
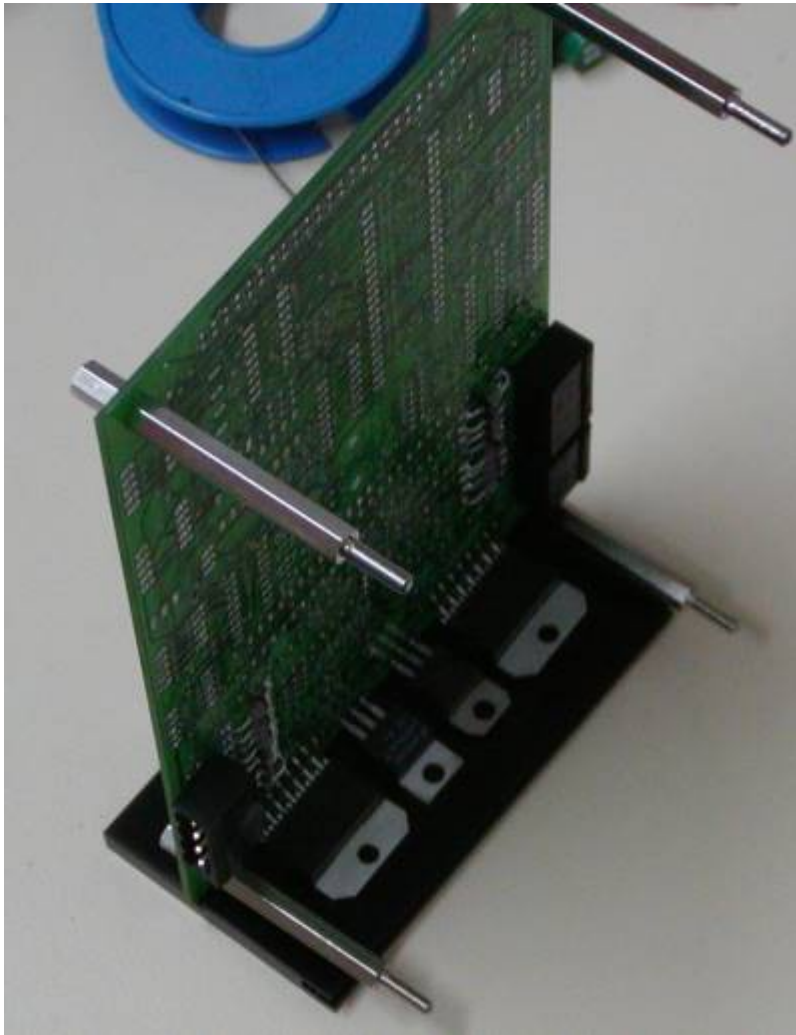
Hier nun ein paar Bilder wie der Aufbau verlaufen könnte:

1. Getriebemotortreiber L298 und 5V Spannungsregler einsetzen. Hier sollte man darauf achten das man die Bauteile so einsetzt, das alle Bauteile später gut an einem Kühlkörper anliegen und die Bohrlöcher zum Befestigen eines Kühlkörpers auf gleicher Höhe liegen.



Aus Platzgründen und um eine Huckepack-Montierung nicht zu behindern, wurde der Kühlkörper nämlich außerhalb der Platine platziert. Am besten nimmt man beim Einlöten dieser Bauteile einen Kühlkörper zur Hilfe (siehe vorheriges Bild).

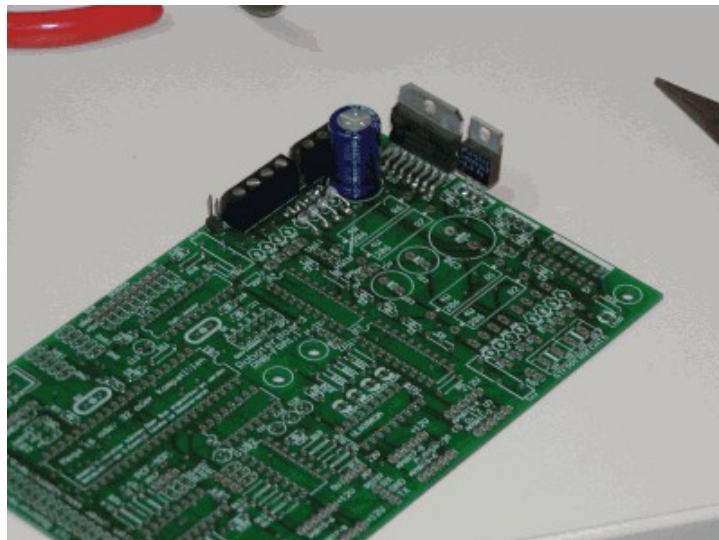
Oder wenn man einen flachen Kühlkörper aus Platzmangel bevorzugt, dann könnte es wie im nächsten Bild aussehen. Hier muß man später allerdings manuell die Löcher zur Befestigung der Bauteile am Kühlkörper selbst bohren.



oder so

Die Größe des Kühlkörpers ist abhängig von dem Strombedarf der Motoren und eventuellen Zusatzboards. Für den Testbetrieb mit kleinen Motoren die weniger als 0,6A benötigen kann man das Board auch schon mal eine Weile ohne Kühlkörper nutzen (insbesondere wenn die Eingangsspannung nicht höher als 9V ist). Für den Dauerbetrieb sollte man jedoch zumindest ein Blech oder kleineren Kühlkörper montieren und alle hinteren Schaltkreise verschrauben. Wenn man das Board in einem Alu-Gehäuse montiert, kann man die hinteren Bauteile auch einfach mit der Gehäusewand verschrauben.

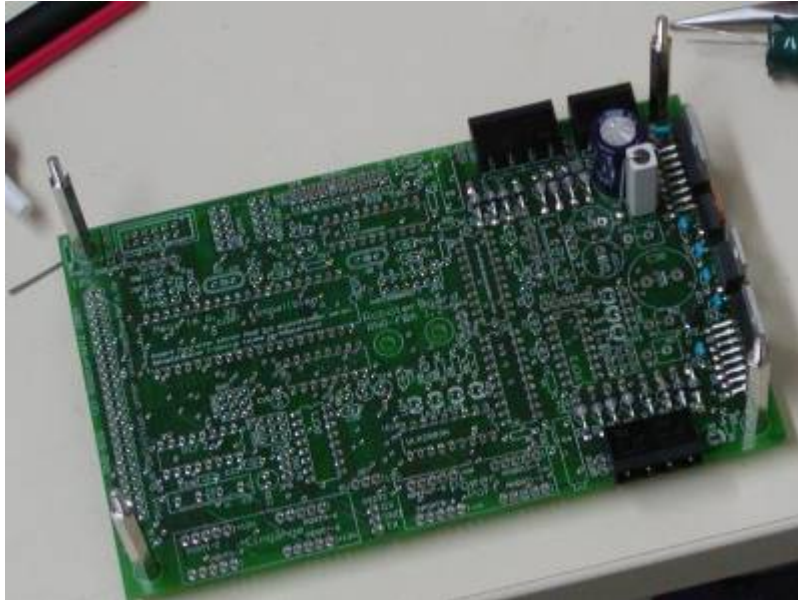
Wichtig ist nur, das die Bauteile nicht so stark mechanisch belastet werden, also eventuell den Kühlkörper mit Abstandsbolzen abstützen.



Will man später eventuell 2 Schrittmotoren ansteuern, dann sollte man gleich alle beiden L298 einlöten. Andernfalls würde einer reichen (siehe letztes Bild).
Danach sollte man mit den umliegenden Bauteilen beginnen.

Wichtig: Bauteile die auf der Unterseite eine leitende Fläche besitzen wie einige stehende Kondensatoren oder Quarze, sollten aus Sicherheitsgründen so eingelötet werden das noch etwa 1-2 mm Platz zwischen Bauteil und Platine besteht. Zwar sind die oberen Leiterbahnen durch Lötstoplack isoliert, aber ich empfehle hier immer auf Nummer sicher zu gehen!

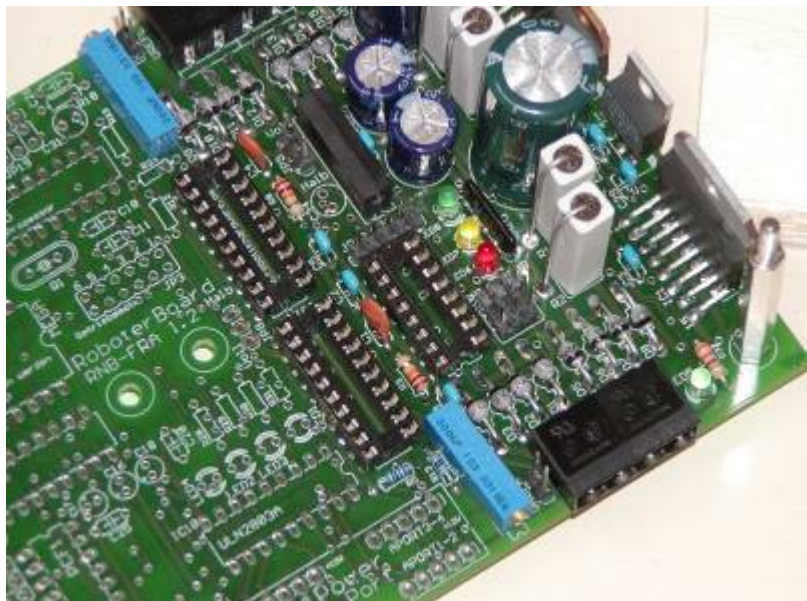
Ansonsten ist eigentlich nur darauf zu achten das man sich nicht selbst den Zugang zu kleineren Bauelementen erschwert.



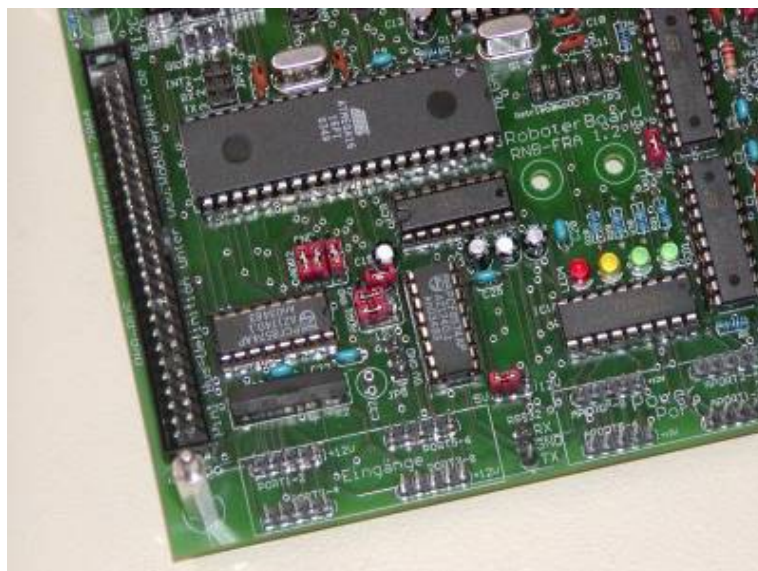
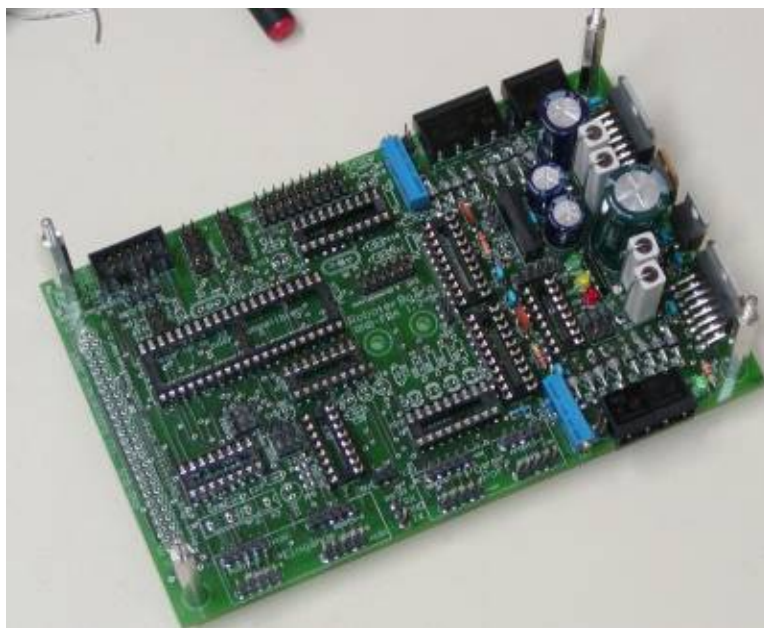
Mann kann danach Region für Region weiter mit Bauteilen bestücken. Sinnvoll ist es auch immer eine Gruppe von Bauelementen einzubauen. Zum Beispiel erst mal alle Widerstände eines Wertes, dann nächster Wert und dann alle Kondensatoren eines Wertes usw.

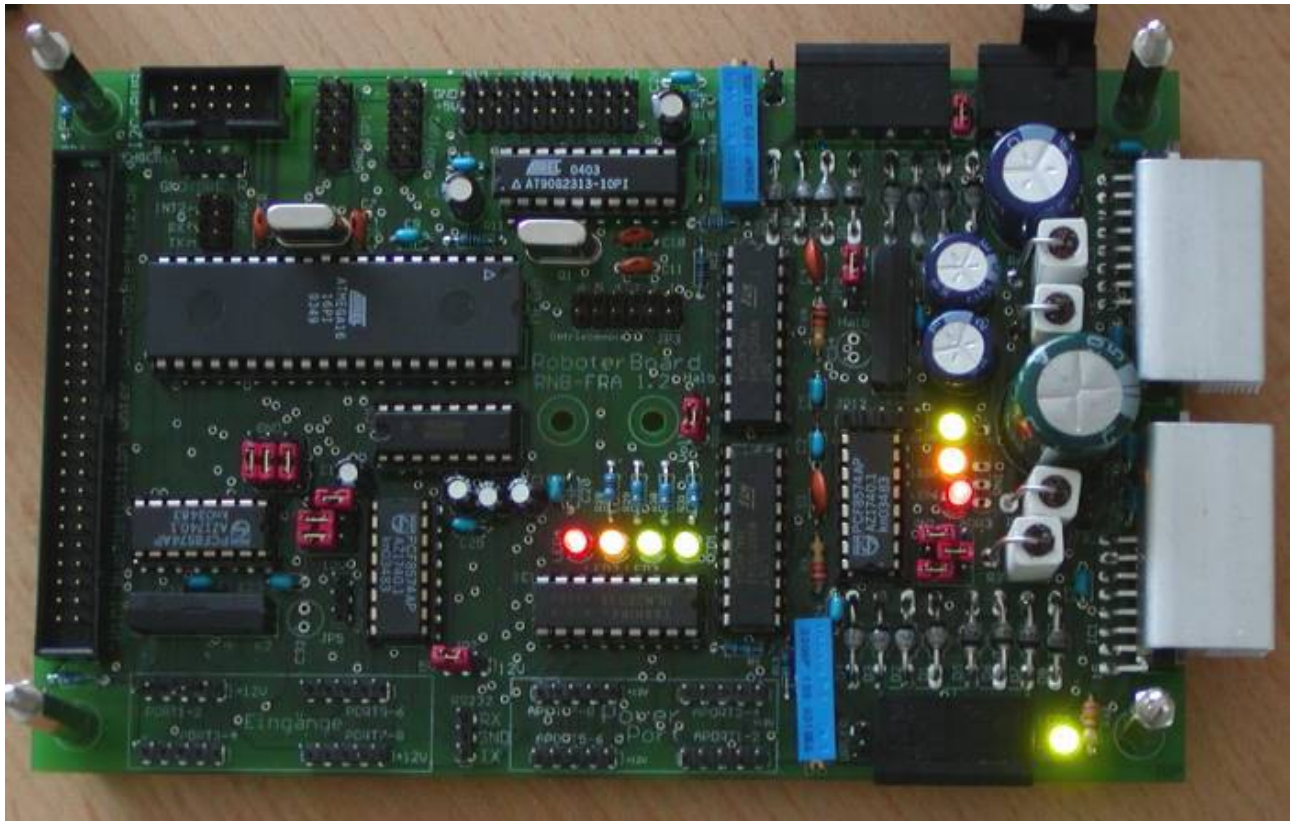
Tip: Um das bestücken zu vereinfachen kann man sich mit Abstandsbolzen behelfen. Wenn man diese schon vor der Bestückung oben und unten verschraubt, dann kann man die Platine sehr schön auf jede Seite stellen und wesentlich besser löten. Profis haben alternativ vielleicht auch einen Platinenhalter!

Es geht langsam aber sicher voran ;-)



Will man keine Brücken für die PCF-Adr. Einlöten, dann kann man Kontaktbuchsen einlöten und Drahtbrücke einstecken!



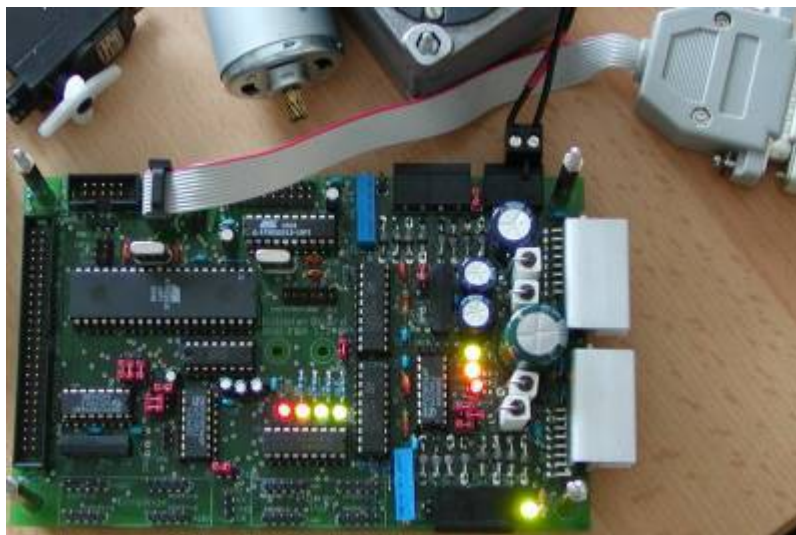


Hat man die Schaltung soweit wie auf dem oberen Bild, dann hat man es endlich geschafft!!!!
Das Board ist komplett aufgebaut und alle Jumper stecken korrekt (hier im Bild Schrittmotor-Betriebsart)

Achtung: Bevor Spannung angelegt wird sollte man nochmal genau alle Jumper (Steckbrücken) genau anschauen und überprüfen. Die genaue Bedeutung steht in dieser Anleitung weiter vorn. Eine falsche Steckbrücke kann unter ungünstigen Umständen Schaltkreise beschädigen!

Erst wenn alle Jumper stimmen und geprüft wurde das keine Kurzschlüsse beim Löteten verursacht wurden kann man Spannung an den beiden Schraubklemmen [+6-15] und [GND] anlegen (**nicht verpolen**). Natürlich tut sich dann noch nix, wie auch, der Controller ist ja noch nicht programmiert Er könnte jetzt aber mit einem kleinen Programmierdongle (siehe Abbildung unten) an die Druckerschnittstelle des PC angeschlossen und programmiert werden. Für erste Test's empfehle ich den Basic-Compiler Bascom da dafür schon einige Testprogramme im Download-Bereich des Roboternetzes stehen. Beim herunterladen immer darauf achten das die Testprogramme für Board Version 1.2 und nicht 1.1 geladen werden. Der Unterschied ist zwar gering, aber die Testprogramme von 1.1 würden zum Teil nicht auf Anhieb funktionieren.

Achten sollte man auch darauf das der Dongle immer richtig herum (wie untere Abbildung) aufgesteckt wird.



Wichtige Tipps

Was man noch beim Aufbau beachten sollte!

Inzwischen haben zahlreiche Roboter-Bastler dieses Board aufgebaut. Bei den meisten klappte dies völlig problemlos. Bei einigen ergaben sich jedoch einige Nachfragen. Um ihnen dies zu ersparen, möchte ich auf dieser Seite die kleinen Hürden über die einige gestolpert sind, schon im vornherein klären:

1. Wie herum muss die LED (Leuchtdiode eingelötet werden)?

Auf dem Bestückungsdruck der Platine ist das zum Teil wegen der Lötpad's (Löffflächen) nicht so sehr gut erkennbar. In diesem Fall empfehlen wir einen Blick auf den Bestückungsplan in der Anleitung. Hier ist deutlich erkennbar das auf einer Seite der LED eine flache Seite zu sehen ist. Wenn Sie ihre Leuchtdiode mal genau anschauen wird diese flache Seite ebenfalls sichtbar. Bei 3mm LED's muss man allerdings genau hinschauen. Sollten Sie es nicht auf Anhieb erkennen, dann kann man sich auch an der Länge der LED-Anschlussbeine orientieren. Auf der flachen Seite ist die Kathode, das ist zugleich das kurze Bein.

2. Wie herum müssen die ISP-Programmieradapter Stecker aufgesteckt werden

Hier sollte man darauf achten das Pin 1 des ISP-Kabels auch an Pin1 der Stiftleiste gesteckt wird. PIN1 ist auf der Platine gekennzeichnet. Allerdings wird diese Kennzeichnung eventuell durch Bauteile verdeckt, daher sollte man im Zweifel auch hier auf den Bestückungsdruck in der Anleitung schauen. Beim ISP-Kabel selbst ist die Leitung für Pin1 in der Regel farblich markiert. In unserem Fall muss also diese farbliche Markierung immer zur Außenkante der Platine hin zeigen. Somit ist dieses eindeutig! Hier sollten sie sich nicht von der Kabelrichtung auf Bildern verwirren lassen, diese kann unterschiedlich ausfallen. Einfach nur auf die farbliche Markierung achten.

3. Widerstandsnetzwerk einlöten

Nicht jeder Einsteiger kennt diese kleinen schwarzen Bauteile die sich Widerstandsnetzwerk nennen. Wichtig ist das diese richtig gepolt eingelötet werden. Dazu ist auf einer Seite des Bauteiles und auf der Platine ein Punkt zur Orientierung.

4. Relais richtig polen

Das Board besitzt zwei Miniaturrelais um den Energiebedarf softwaremäßig zu reduzieren. Auch diese müssen richtig gepolt eingelötet werden. Auch dazu ist auf der Platine und auf dem Bauteil ein Punkt zur Orientierung.

5. Welche LED's leuchten nach dem ersten einschalten

Wenn das Board das erste mal unter Spannung gesetzt wird, sollten die mittleren 4 LED's immer leuchten und die drei rechten aus sein. Wenn dies der Fall ist, dann ist dies schon mal ein gutes Zeichen das der Aufbau offensichtlich perfekt geklappt hat.

Der erste Test

Nachdem Board aufgebaut ist, können wir daran gehen und das Board testen. Da Schrittmotoren erst am Schluss getestet werden, sollte die Jumper so eingestellt werden das das Board zwei Getriebemotoren unterstützt. Für diesen Modus muß unbedingt das L297 IC4 aus der Fassung genommen werden. Zudem müssen bei der Stiftleiste JP3 alle 6 Kurzschlussbrücken gesteckt sein. Die PCF-Adresse des Ausgabeportes (ADR1) muß für die nachfolgenden Beispiele auf Hex 72 eingestellt werden (alternativ kann man natürlich auch die Beispiele ändern).

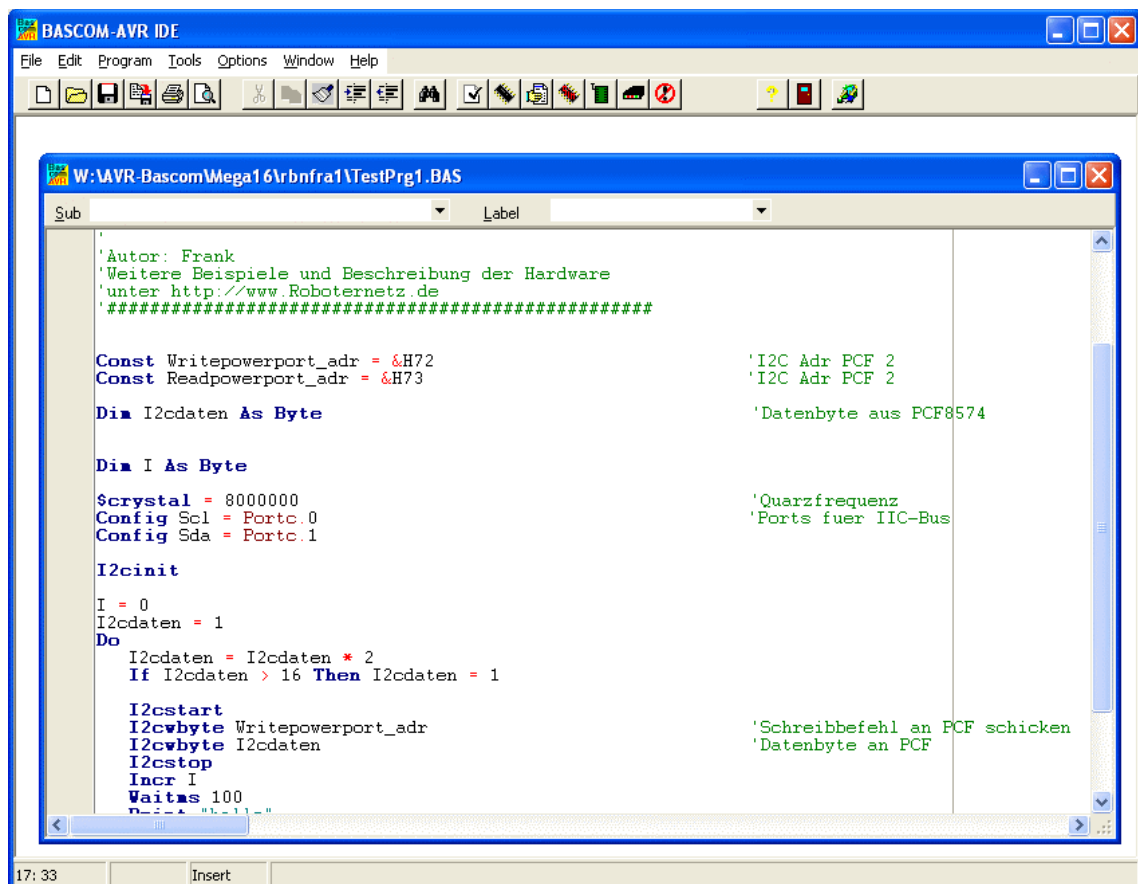
Danach kann mit einem üblichen ISP Dongle (am besten *STK200* / *STK300* kompatibel) das Board mit der Druckerschnittstelle ihres PC verbunden werden. Dabei ist darauf zu achten das der ISP-Stecker richtig herum aufgesteckt wird, wie auf dem letzten Bild zu ersehen.

Üblicher ISP-Dongle



Ist man kein AVR-Profi, so empfehle ich für den Test des Board die Entwicklungsumgebung und Programmiersprache Bascom. Ein schneller Basic-Compiler mit hervorragender Benutzeroberfläche. Eine Version die nur in der Programmlänge etwas beschränkt ist (max 2KB) gibt es beim Hersteller MCS Electronics kostenlos. Die genauen Links findet man am besten auf der Projektseite bzw. [hier](#).
Hilfreiche deutsche Foren für Bascom und AVR gibt es [hier](#).

Über diesen Link findet man 4 Zip Dateien die man in einem leeren Verzeichnis entpackt. Anschließend kann man das ganze einfach mit SETUP installieren. Anschließend legt man über File/New ein neue Datei an und gibt die nachfolgenden Beispielprogramme ein. Alternativ kann man die ganzen Beispielprogramme auch auf unserer Projektseite herunterladen: <http://www.mikrocontroller-elektronik.de/>



```
W:\AVR-Bascom\Mega16\rbnfra1\TestPrg1.BAS
Sub
Label
'
'Autor: Frank
>Weitere Beispiele und Beschreibung der Hardware
'unter http://www.Roboternetz.de
'#####
Const Writepowerport_adr = &H72      'I2C Adr PCF 2
Const Readpowerport_adr = &H73      'I2C Adr PCF 2
Dim I2cdaten As Byte                'Datenbyte aus PCF8574
Dim I As Byte
$crystal = 8000000                   'Quarzfrequenz
Config Scl = Portc.0                 'Ports fuer IIC-Bus
Config Sda = Portc.1
I2cinit
I = 0
I2cdaten = 1
Do
  I2cdaten = I2cdaten * 2
  If I2cdaten > 16 Then I2cdaten = 1
  I2cstart
  I2cwbyte Writepowerport_adr      'Schreibbefehl an PCF schicken
  I2cwbyte I2cdaten                 'Datenbyte an PCF
  I2cstop
  Incr I
  Waitms 100
Print "I2cdaten="
```

Danach muß man noch über *Options/Compiler/Chip* den Controllertyp anwählen. Bei der üblichen Bestückung müssten sie dort M16 (für Mega16) wählen. Dann muss in einem anderen Abschnitte dieses Dialoges „*Programmer*“ noch der ISP-Adapter gewählt werden. In den meisten Fällen dürfte das de *STK200 / STK300 Programmer* sein.

Damit dürfte das wichtigste passiert sein. Allerdings ist nun noch der Quarz deaktiviert, da der Mega16 generell immer den internen 1 Mhz Takt nutzt. Aber das muss uns für den ersten Test nicht stören. Im Beispiel sollte dann jedoch statt `$crystal = 8000000` die Anweisung `$crystal = 1000000` stehen. Jetzt kennen Sie auch schon die Anweisung die die Taktfrequenz angibt. Diese ist sehr wichtig damit Zeitabhängige Dinge wie Timerprogrammierung, RS232, Wait-Funktionen usw. korrekt funktionieren.

```
#####
'Testprogramm 1
'für
'RoboterNetz Standard-Roboter Board RBNFRA 1.2
'
'Aufgabe:
'Ausgabe über PowerPort per I2C testen indem
'die vier Led's im lauflichtartig leuchten
'
'Autor: Frank
>Weitere Beispiele und Beschreibung der Hardware
'unter http://www.Roboternetz.de & http://www.mikrocontroller-elektronik.de/
#####

Const Writepowerport_adr = &H72           'I2C Adr PCF 2
Const Readpowerport_adr = &H73           'I2C Adr PCF 2

Dim I2cdaten As Byte                     'Datenbyte aus PCF8574

Dim I As Byte

$crystal = 8000000                       'Quarzfrequenz
$baud = 9600
Config Scl = Portc.0                     'Ports fuer IIC-Bus
Config Sda = Portc.1

I2cinit

'***** Diese 4 Befehle sind nur ab RBNFRA Version 1.2 (nicht in V 1.1)
' notwendig und bzw. möglich (erweiterte Energiesparfunktion und LED's)
' Bei Board 1.1 bitte auskommentieren oder löschen
I2cstart
I2cwbyte &H74                           'Schreibbefehl an PCF3 schicken
' Led's ein ,Motorendstufen ein, Port-Peripherie ein, RBN-Bus Sleep Modus aus (also Peripherie
aktiv)
I2cwbyte &B00000010                       'Datenbyte an PCF3
I2cstop
'*****

I = 0
I2cdaten = 1
Do
    I2cdaten = I2cdaten * 2
    If I2cdaten > 16 Then I2cdaten = 1

    I2cstart
    I2cwbyte Writepowerport_adr           'Schreibbefehl an PCF schicken
    I2cwbyte I2cdaten                     'Datenbyte an PCF
    I2cstop
    Incr I
    Waitms 100
    Print "hallo"
Loop

End
```

Haben sie alles richtig gemacht und oberes Beispiel auch korrekt eingegeben, dann können sie das Programm über das Symbol „Schwarze IC in der Toolbar“ compilieren. Es darf keine Fehlermeldung im unteren Fenster erscheinen.

Erscheint eine Fehlermeldung dann klicken sie diese doppelt an um zur fehlerhaften Stelle zu gelangen, korrigieren sie die fehlerhafte Zeile und compilieren sie erneut.

Ist das Programm fehlerfrei compiliert, so klicken sie auf das grüne Symbol etwas weiter rechts (soll wohl eine IC-Wechselfassung sein). Anschließend wählen sie „Program“ in dem erscheinenden Menü. Dadurch gelangen sie in das eingebaute Übertragungsprogramm. Rechts oben müssen Sie nun nochmals M16 für Mega16 wählen (falls noch nicht vorgegeben). Danach klicken sie auch in diesem Dialog auf das gleiche grüne Symbol. Wenn sie alles in Bascom richtig eingestellt haben und die Platine in Ordnung ist, dann sollte das Programm in wenigen Sekunden übertragen worden und auch gestartet worden sein. Diese ganze Prozedur lässt sich auch über die rechte Maustaste statt über die Toolbar bewerkstelligen.

Es sollte nun ein kleines 4 LED Lauflicht erscheinen.

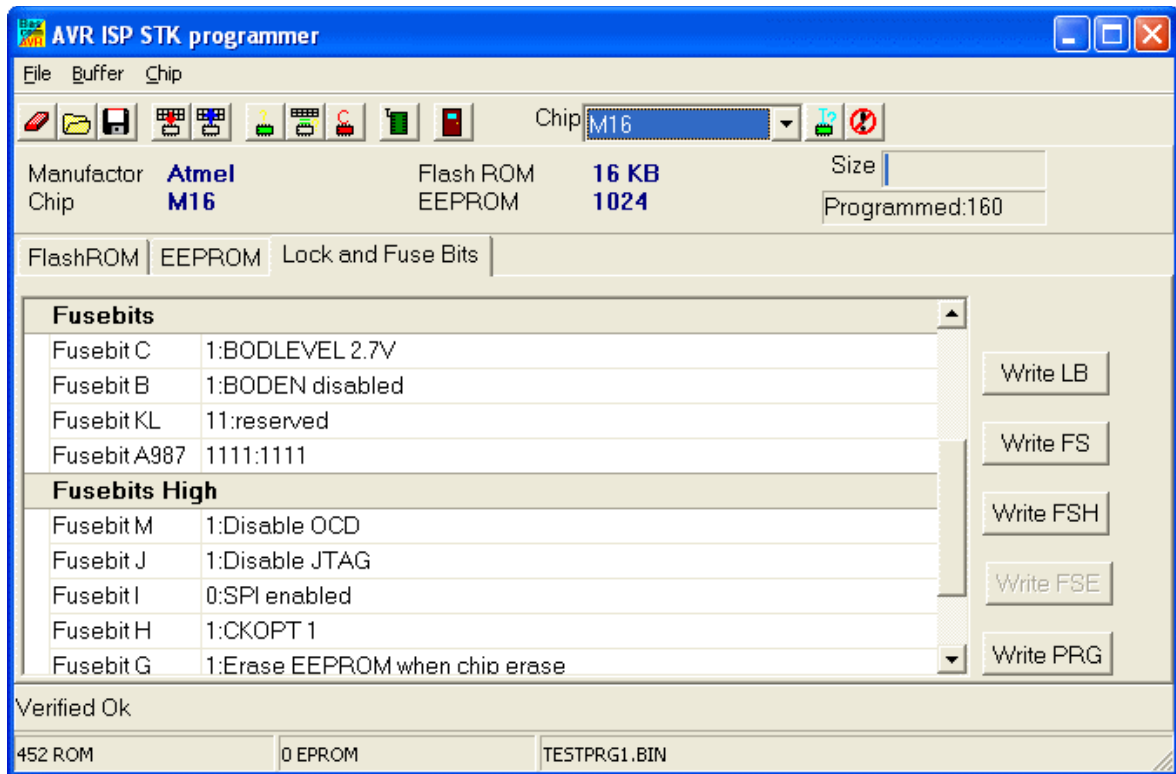
Ist dies nicht der Fall, so müssen sie forschen ob es an Bascom-Einstellungen oder am Board liegt. Es würde natürlich den Rahmen sprengen wenn in dieser Anleitung Bascom genau beschrieben würde. Aber im Roboternetz-Download-Bereich finden Sie unter Tutorials einige etwas ausführlichere Bascom Hilfen. Zudem kann man bei Fragen ja auch unser Bascom Forum nutzen.

Wenn alles funktioniert, dann haben wir schon eine ganze Reihe von Dingen auf dem Board getestet. Zum einen scheint der Mega 16 und die ISP korrekt zu funktionieren. Zum anderen funktioniert auch die I2C-Bus Ansteuerung des PCF sowie der PCF-Ausgangstreiber selbst. Gleichzeitig wird dadurch doch der PowerPort getestet, denn die LED's hängt parallel zu den ersten 4 Portleitungen dieses Powerports. Die Portleitungen können mit bis zu 500 mA belastet werden, wobei jedoch die Gesamtbelastung die Leistung des Spannungsreglers nicht übersteigen darf, versteht sich. Wenn Sie nun an den Ausgängen die Spannung der Ports prüfen wollen, so müssen Sie bedenken das dieser Powerport immer im High Zustand auf Masse schaltet und im Low Zustand offen ist. Wenn Sie also beispielsweise ein Relais anschließen wollen, dann müssen Sie ein Pin des Relais mit +5V oder +12V verbinden und das andere mit einer Portleitung des Powerports. Wenn die LED leuchtet, würde auch das Relais anziehen. Zu beachten ist bei dem Powerport noch, das wahlweise 5V oder 12V Verbraucher angeschlossen werden dürfen, jedoch muß dies eindeutig durch einen Jumper JP7 festgelegt werden. Wenn logische TTL kompatible Schaltkreise angeschlossen werden sollen, dann muß dieser Jumper natürlich immer auf 5V stehen. Zu beachten ist das die 12V natürlich nur da sein können, wenn die Batteriespannung auch mindestens so hoch ist, ansonsten würde hier bei der Einstellung 12V immer die volle Batteriespannung möglich sein. Ist die Batteriespannung höher als 12 V, dann ist es sinnvoll JP2 zu entfernen und den Spannungsstabilisator 7812 wirken zu lassen. Dadurch ist sichergestellt das hier niemals mehr als 12V geschaltet werden kann.

Das Beispiel hat auch schön demonstriert, wie einfach doch die I2C.Bus Ansteuerung unter Bascom ist.

Nun Fuse-Bits des Mega einstellen

Nachdem das Beispiel so gut funktioniert hat sollte man gleich die Gelegenheit nutzen und den Controller auf die richtige Quarz Taktfrequenz von 8 Mhz umschalten. Dies kann man auch in Bascom machen indem man wieder den Programmer über das grüne Symbol aufruft und unten die Seite „Lock and Fuse Bits“ anwählt (siehe Bild).



Das ganze muss bei angeschlossenen Mega16 erfolgen damit die momentan eingestellten werte erscheinen. Die Werte sollten so eingestellt werden, das sie der oberen Abbildung entsprechen. Hier sollten Sie unbedingt *Fusebit C,B,KL* und *A987* vergleichen und gegebenenfalls ändern. Gleichzeitig sollten Sie an dieser Stelle *Fusebit J* auf *Disable JTAG* einstellen, damit diese Ports für die spätere Schrittmotorsteuerung frei werden. Haben sie die Werte verändert, dann klicken Sie auf *Write FS* und *Write FSH*. Dadurch sollte der Controller auf 8 Mhz umgeschaltet werden.

Alternativ kann man auch das beliebte Pony-Program verwenden, dazu mehr in dem Artikel „AVR leicht gemacht“ im Roboternetz im unter Artikeln http://rn-wissen.de/wiki/index.php/AVR-Einstieg_leicht_gemacht

Jetzt muss noch die Quarzfrequenz im letzten Beispiel wieder mit der Anweisung `$crystal = 8000000` angegeben werden und schon können sie erneut compilieren und das Programm übertragen. Klappt alles genauso wie vorher, dann sollte alles perfekt funktioniert haben. Leuchten die LED's jedoch schneller oder langsamer, dann stimmt mit den Einstellungen irgend etwas nicht. Im Zweifel sollte man zu dem Programm Pony greifen um die Fusebits zu ändern. Zum Teil ist es da etwas übersichtlicher. Man muss diese Einstellung ja nur ein einziges mal bei neuen Controllern durchführen. **Achtung:** Eine falsche Einstellung der Fuse-Bits kann den Controller unbrauchbar machen!

Jetzt testen wir die RS232 und den MAX

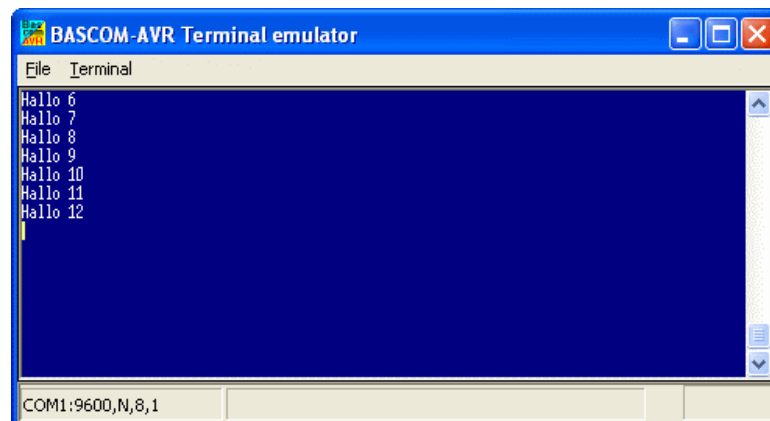
Nun laden oder tippen wir Testprogramm2 ein. Dieses Testprogramm wird die Datenübertragung zum PC testen. Dazu muss die RS232-Schnittstelle über ein 3 poliges Kabel mit der 3 poligen Stiftleiste RS232 auf dem Board verbunden werden. Wer zufällig einen Conrad CCRP5 Zuhause haben sollte, der kann dazu das mitgelieferte RS232 Kabel verwenden da unsere RS232 Stiftleiste pinkompatibel ist. Auf dem letzten Board Bild können Sie ersehen wie herum das Kabel aufgesteckt werden muss. Da in der Mitte GND und an den Außenseiten RX/TX liegt, schadet es in der Regel nicht wenn es mal falsch aufgesteckt werden sollte. Allerdings tut sich dann auch nix ;-)

Natürlich kann man sich auch selbst ein Kabel basteln, es muss ja lediglich RX / GND und TX an die D-Sub-Buchse des PC angeschlossen werden.

Ist dieses erfolgt, kann das nachfolgende Programm übertragen werden.

```
'#####  
'Testprogramm 2  
'für  
'RoboterNetz Standard-Roboter Board RBNFRA 1.2  
'  
'Aufgabe:  
'Testen vom MAX232 und RS232 Schnittstelle indem  
'das Wort Hallo und eine Zahl an die RS232 sendet!  
'Unter Windows kann man das mit dem Hyperterminal  
'anzeigen lassen (9600 Baud einstellen)  
'Noch einfacher kann man es anzeigen, wenn man  
'in der Bascom Entwicklungsumgebung unter Tools  
'TERMINAL EMULATOR aufruft (oder Modem Symbol klickt)  
'  
'Autor: Frank  
'Weitere Beispiele und Beschreibung der Hardware  
'unter http://www.RoboterNetz.de und http://www.mikrocontroller-elektronik.de/  
'#####  
  
Dim I As Byte  
  
$crystal = 8000000 'Quarzfrequenz  
$baud = 9600  
  
I = 0  
Do  
  Incr I  
  Wait 2  
  Print "Hallo " ; I  
Loop  
End
```

Danach kann man über Menü Tools von Bascom (oder Modem Symbol) einen Terminalemulator starten. Dort wird nun das Wort „Hallo x“ empfangen, wobei x jede Sekunde um 1 erhöht wird. Kommt garnix, dann ist eventuell das Kabel falschrum auf die Stiftleiste aufgesteckt worden. Kommen wirre Zeichen, dann ist vermutlich im Terminalprogramm noch keine 9600 Baud eingestellt.



Wenn es so aussieht, dann stimmt Quarzfrequenz und ihr Max-Pegelwandler funktioniert ebenfalls. Übrigens, nicht alle Baudraten gehen mit jeder Quarzfrequenz. In jedem fall geht bei 1 Mhz 1200 Baud und bei 8 Mhz 9600 Baud.

Funktioniert die Getriebemotoransteuerung mit L298?

Im nächsten Beispiel testen wir die Ansteuerung des linken Motortreibers L298 (der neben dem Spannungseingang) in Verbindung mit zwei Getriebemotoren. Ich hab dazu zwei Getriebemotoren vom Typ RG40 an die Klemme „Motor(en)Links“ angeschlossen. Ein Motor an Klemme 1+2 und ein Motor an 3 + 4. Es kann natürlich auch ein ganz anderer Motor verwendet werden. Solange aber noch kein Kühlkörper am L298 hängt, sollten die Motoren möglichst nicht mehr als 0,6 A benötigen. Zudem sollten diese die volle Batteriespannung vertragen.

Ich möchte nochmals daran erinnern, das IC4 (L297) aus der Fassung entfernt werden muss um Getriebemotoren ansteuern zu können. Zudem müssen alle Brücken an JP3 gesetzt sein. Da sie vermutlich nur eine Spannung anschließen, sollte auch JP6 gesetzt sein.

Ist das alles korrekt, dann laden, compilieren und übertragen wir Testprogramm3

```
#####
'Testprogramm 3
'für
'RoboterNetz Standard-Roboter Board RBNFRA 1.2
'
'Aufgabe:
'Testen der Getriebemotorensteuerung
'1. Linker Motor wird 5 Sekunden gedreht
'2. 5 Sekunden Pause
'3. Linker Motor wird 5 Sekunden in andere Richtung gedreht
'4. 5 Sekunden Pause
'5. Rechter Motor wird 5 Sekunden gedreht
'6. 5 Sekunden Pause
'7. Rechter Motor wird 5 Sekunden in andere Richtung gedreht
'8. 5 Sekunden Pause
'9. Das ganze wieder ab linken Motor wiederholen
'
'Autor: Frank
>Weitere Beispiele und Beschreibung der Hardware
'unter http://www.mikrocontroller-elektronik.de/
#####

Const Writepowerport_adr = &H72           'I2C Adr PCF 2
Const Readpowerport_adr = &H73           'I2C Adr PCF 2
Dim I2cdaten As Byte                    'Datenbyte aus PCF8574

Dim I As Byte

$crystal = 8000000                       'Quarzfrequenz

Config Scl = Portc.0                      'Ports fuer IIC-Bus
Config Sda = Portc.1

I2cinit
'***** Diese 4 Befehle sind nur ab RBNFRA Version 1.2 (nicht in V 1.1)
' notwendig und bzw. möglich (erweiterte Energiesparfunktion und LED's)
' Bei Board 1.1 bitte auskommentieren oder löschen
I2cstart
I2cwbyte &H74                            'Schreibbefehl an PCF3 schicken
' Led's ein ,Motorendstufen ein, Port-Peripherie ein, RBN-Bus Sleep Modus aus (also Peripherie
aktiv)
I2cwbyte &B00000010                       'Datenbyte an PCF3
I2cstop
'*****

Nochmal:

'Ports für linken Motor
Config Pinc.6 = Output                    'Linker Motor Kanal 1
Config Pinc.7 = Output                    'Linker Motor Kanal 2
Config Pind.4 = Output                    'Linker Motor PWM

'Linker Motor ein
Portc.6 = 1                              'bestimmt Richtung
Portc.7 = 0                              'bestimmt Richtung
Portd.4 = 1                              'Linker Motor EIN

Wait 5                                    'Warte 5 Sekunden
Portd.4 = 0                              'Linker Motor AUS
```



```

Wait 5                                     'Warte 5 Sekunden

'Linker Motor andere Richtung
Portc.6 = 0                                'bestimmt Richtung linker Motor
Portc.7 = 1                                'bestimmt Richtung linker Motor
Portd.4 = 1                                'linker Motor EIN

Wait 5                                     'Warte 5 Sekunden
Portd.4 = 0                                'Motor AUS
Wait 5                                     'Warte 5 Sekunden

'Ports für rechten Motor
Config Pinb.0 = Output                    'Rechter Motor Kanal 1
Config Pinb.1 = Output                    'Rechter Motor Kanal 2
Config Pind.5 = Output                    'Rechter Motor PWM

'Rechter Motor ein
Portb.0 = 1                                'bestimmt Richtung rechter Motor
Portb.1 = 0                                'bestimmt Richtung rechter Motor
Portd.5 = 1                                'rechter Motor EIN

Wait 5                                     'Warte 5 Sekunden
Portd.5 = 0                                'Rechter Motor AUS
Wait 5                                     'Warte 5 Sekunden

Portb.0 = 0                                'bestimmt Richtung rechter Motor
Portb.1 = 1                                'bestimmt Richtung rechter Motor
Portd.5 = 1                                'rechter Motor EIN

Wait 5                                     'Warte 5 Sekunden
Portd.5 = 0                                'Rechter Motor AUS
Wait 5                                     'Warte 5 Sekunden

Goto Nochmal

End

```

Wenn wir alles korrekt gemacht haben, dann sollten nun erst der eine Motor 5 Sekunden in die eine Richtung, dann 5 Sekunden in die andere Richtung und dann der andere Motor das gleiche machen. Danach wiederholt sich das ganze. Die Motoren werden in diesem Programm nicht in der Geschwindigkeit geregelt, also nicht mit PWM (Pulsweitenmodulation) angesteuert. Sie laufen daher immer auf voller Leistung!

Der I2C Bus ist für die Motorsteuerung nicht erforderlich, da dafür reservierte Controllerports (siehe Listing) benutzt werden. Diese Leitungen sind auch am RNB-Bus vorhanden, so das auch ein externer Controller diese Motoren steuern könnte. In diesem Fall sollte man die betreffenden MEGA-Ports auf Eingang schalten oder den Mega16 ganz weglassen.

Welche Ports für was zuständig sind ist ja sehr gut aus dem Listing zu entnehmen. Später soll natürlich noch ein etwas komfortableres Treiberprogramm ins RoboterNetz gestellt werden, bei dem die Motoren und auch andere Dinge bequem über Subroutinen (Basic-Funktionen) gesteuert werden. Eine Art Engine die einfach in eigene Programme integriert wird. Allerdings bin ich momentan noch etwas ausgelastet so das dies noch etwas dauern kann. Vielleicht hat ja zwischenzeitlich dann auch schon jemand anders eine solche Engine geschrieben

Sollte dieses Beispiel nicht funktionieren, sollte man die zuständigen Jumper, Ports und den L298 sowie die Leiterbahnverbindungen auf Lötfehler überprüfen.

Nun zur Geschwindigkeitsregelung der Motoren

Die Jumper-Einstellungen am Board lassen wir jetzt genauso wie zuvor. Für die Geschwindigkeitsregelung brauchen wir lediglich ein anderes Programm. Nun wird die oft im Roboternetz genannte Pulsweitenmodulation (PWM) angewendet.

Bei diesem Modus werden die gleichen Ports verwendet wie im letzten Listing, was die Ports für die Motorrichtung betrifft, so bleibt alles beim alten. Lediglich der Port der zum Einschalten der jeweiligen Motoren verwendet wird, dieser wird nun über einen internen Timer pulsweitenmoduliert. Die Pulsweitenmodulation bestimmt die jeweilige Dauer eines periodisch wechselnden Spannungspegels. Im Grunde wird der Motor mit einer festen Frequenz angesteuert, wobei jedoch die Länge des High-Pegels in jeder Periode verändert wird. Dadurch wird der Motor quasi in jeder Periode eine kurze Zeit ausgeschaltet. Da die Perioden so schnell Wechsel, ergibt sich so ein ähnliches Ergebnis als wenn der Motor mit einer niedrigeren Spannung angesteuert würde. Dieses Verfahren hat sogar noch eine ganze Reihe von Vorteilen, zum Beispiel entstehen kaum Verluste, da die Elektronik den Strom ja nicht in Wärme verheizen muß sondern einfach nur den Motor intelligent ein und ausschaltet. Dieses Verfahren wird in der Praxis auch bei großen Motoren oft angewendet, in ähnlicher Form ist es auch in den bekannten Licht-Dimmern der Fall (die Wechselspannung ist hier quasi die Grundfrequenz und die Halbwelle wird dort mit einem Triac angeschnitten – im Prinzip ist es das gleiche Ergebnis).

Hört sich alles etwas kompliziert an, aber ihr werden euch wundern, mit Bascom ist das in wenigen Programmzeilen erledigt, siehe Quelltext: von Testprogramm 4

```

#####
'Testprogramm 4
'für
'RoboterNetz Standard-Roboter Board RBNFRA 1.2
'
'Aufgabe:
'Testen der Geschwindigkeitsregelung der
'Getriebemotorensteuerung über PWM
'Regelt die Geschwindigkeit beider Motoren langsam
'hoch und wieder runter
'
'Autor: Frank
>Weitere Beispiele und Beschreibung der Hardware
'unter http://www.Roboternetz.de und http://www.mikrocontroller-elektronik.de/
#####

Dim I As Word

$crystal = 800000                                'Quarzfrequenz

Dim Geschwindigkeitlinks As Word
Dim Geschwindigkeitrechts As Word

'Ports für linken Motor
Config Pinc.6 = Output                            'Linker Motor Kanal 1
Config Pinc.7 = Output                            'Linker Motor Kanal 2
Config Pind.4 = Output                            'Linker Motor PWM

'Ports für rechten Motor
Config Pinb.0 = Output                            'Rechter Motor Kanal 1
Config Pinb.1 = Output                            'Rechter Motor Kanal 2
Config Pind.5 = Output                            'Rechter Motor PWM

Config Scl = Portc.0                              'Ports fuer IIC-Bus
Config Sda = Portc.1

I2cinit
'***** Diese 4 Befehle sind nur ab RBNFRA Version 1.2 (nicht in V 1.1)
' notwendig und bzw. möglich (erweiterte Energiesparfunktion und LED's)
' Bei Board 1.1 bitte auskommentieren oder löschen
I2cstart
I2cwbyte &H74                                    'Schreibbefehl an PCF3
schicken
' Led's ein ,Motorendstufen ein, Port-Peripherie ein, RBN-Bus Sleep Modus aus (also
Peripherie aktiv)
I2cwbyte &B00000010                              'Datenbyte an PCF3

```

```

I2cstop
'*****

Config Timer1 = Pwm , Pwm = 10 , Compare A Pwm = Clear Down , Compare B Pwm = Clear Down
Geschwindigkeitlinks = 50
Geschwindigkeitrechts = 2900
Pwm1a = Geschwindigkeitrechts
Pwm1b = Geschwindigkeitlinks
Tccr1b = Tccr1b Or &H02                                'Prescaler = 8

'Linker Motor ein
Portc.6 = 1                                            'bestimmt Richtung
Portc.7 = 0                                            'bestimmt Richtung
Portd.4 = 1                                            'Linker Motor EIN

'Rechter Motor ein
Portb.0 = 1                                            'bestimmt Richtung rechter
Motor
Portb.1 = 0                                            'bestimmt Richtung rechter
Motor
Portd.5 = 1                                            'rechter Motor EIN

I = 0
Do
    Pwm1a = I
    Pwm1b = I
    Waitms 40
    I = I + 5
Loop Until I > 1023

Wait 1
Do
    Pwm1a = I
    Pwm1b = I
    Waitms 40
    I = I - 5
Loop Until I < 1

Pwm1a = 0                                            'Linker Motor aus
Pwm1b = 0                                            'rechter Motor aus

End
    
```

Wenn ihr das Programm geladen, kompiliert und übertragen habt, sollten sich beide Motoren langsam von 0 bis auf volle Geschwindigkeit beschleunigen, danach 1 Sekunde pausieren und wieder langsam ihre Geschwindigkeit auf 0 reduzieren.

Für diese Aufgabe wird der Timer1 in einigen Zeilen konfiguriert:

```

Config Timer1 = Pwm , Pwm = 10 , Compare A Pwm = Clear Down , Compare B Pwm = Clear Down
Geschwindigkeitlinks = 50
Geschwindigkeitrechts = 2900
Pwm1a = Geschwindigkeitrechts
Pwm1b = Geschwindigkeitlinks
Tccr1b = Tccr1b Or &H02                                'Prescaler = 8
    
```

Danach ist das ganze äußerst einfach anzusteuern, ähnlich wie beim letzten Listing. Dem Befehl Pwm1a und Pwm1b für den anderen Motor, wird einfach die Geschwindigkeit (in Stufen von 0 bis 1023) zugewiesen. **Mehr nicht!**

Achten sollte man darauf das nun die Motoren anders ausgeschaltet werden müssen als im letzten Beispielprogramm. Nun muß man dazu einfach die Geschwindigkeit auf 0 setzen.

```

Pwm1a = 0                                            'Linker Motor aus
Pwm1b = 0                                            'rechter Motor aus
    
```

Wenn man mal von den Timer-Registern absieht, wurde durch dieses Testprogramm keine weitere Hardware des Boards getestet, da wir die selben Bausteine wie im letzten Listing genutzt haben.

Jetzt testen wir IC L297 und Schrittmotoren

Schrittmotoren gelten immer noch als etwas kompliziert in der Ansteuerung. Das dies ein Vorurteil ist, werden wir jetzt feststellen.

Wir wollen gleich zwei Schrittmotoren anschließen, daher ist eine Umkonfigurierung des Boards notwendig. Also Spannung weg nehmen und dann:

1. IC4 und IC2, beides L297, richtig herum in die IC-Fassung stecken
2. Alle Kurzschlussstecker aus JP3 entfernen
3. Motoren noch nicht anschließen
4. Ein Spannungsmessgerät im 5V Bereich an JP18 anschließen, möglichst mit einem kleinen Stecker (Kurzschlüsse sind hier zu vermeiden)
5. Nun anhand der Motordaten die Referenzspannung berechnen.
6. Die Referenzspannung bestimmt die Strombegrenzung beim **linken** Schrittmotor.

V_{Ref} berechnet sich nach folgender Formel:

$$V_{ref} = I_{motorstrom} * R_s \text{ (bei uns 0,51 Ohm)}$$

Angenommen Sie haben einen Motor der 0,5 A Strom verträgt, in diesem Fall würde Rechnung so aussehen:

$$V_{ref} = 0,5 * 0,51 \\ \text{Ergebnis: } V_{ref} = 0,26 \text{ V}$$

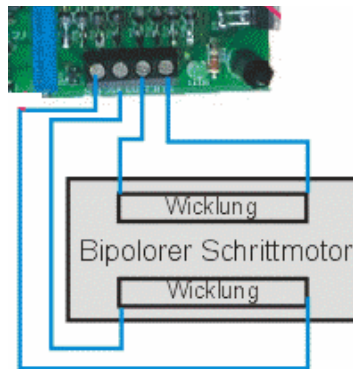
Für diese Berechnung muss man also nur den Nennstrom (Phasenstrom) eines Schrittmotors kennen. Dieser sollte bei jedem Motor angegeben sein.

7. Als nächstes schalten wir die Spannung des Boards ein
8. Jetzt überprüfen wir ob das Messgerät die berechnete Referenzspannung anzeigt. Ist dies nicht der Fall, so drehen wir den daneben liegenden Spindeltrimmer mit einem kleinen Schraubenzieher nach links oder nach rechts bis der gewünschte Wert gemessen wird. Damit ist die Strombegrenzung festgelegt.
9. Das gleiche machen wir nun auf der anderen Seite mit JP4 und dem dortigen Spindeltrimmer
10. Jetzt nimmt man wieder die Spannung weg und schließt die Motoren an. Wichtig ist das man hier die äußeren Spulenden der Wicklungen benutzt. Mit einem Messgerät kann man diese durch den ohmschen Widerstand gut in Erfahrung bringen. Die meisten Motoren haben 4 Anschlüsse und sind somit optimal geeignet. Es gibt auch Motoren welche die Mittelanzapfung der jeweiligen Wicklung herausgeführt haben. Diese haben dann logischerweise 6 Anschlüsse. Bei diesen Motoren ignoriert man einfach diesen Mittelanschluß, man sollte ihn allerdings gut isolieren, ein Kontakt mit einer anderen Leitung könnte den L298 zerstören.

Vorsicht bei Motoren mit 5 Anschlüssen, hier sind in der Regel die Mittelanschlüsse zusammengelegt! Diese Motoren dürfen auf keinen fall angeschlossen werden!

Bei Verwendung Kühlkörper können Motoren mit maximal 3 A Phasenstrom angeschlossen werden. Bei Dauerbelastung empfehle ich aber nicht unbedingt immer an diese Grenze zu gehen. Die Nennspannung des Schrittmotors ist durch die Stromregelung fast unwichtig geworden. Hier gilt generell das Schrittmotoren mit möglichst kleiner Nennspannung (durchaus auch 1 oder 2V) die beste Leistung bringen werden. Dies hat mit der Induktivität die dem Stromfluss entgegenwirkt zu tun (nähere Infos im Roboternetz). Wichtig ist nur, das die Nennspannung des Motors nicht die zur Verfügung stehende Batterie bzw. Motorspannung übersteigt, dadurch würde die Leistung stark reduziert und eventuell bei höheren Drehzahlen fast keine Leistung mehr vorhanden sein.

Beide Motoren also entsprechend dieser Skizze anschließen, jeweils einen an Anschluss STEPPERRECHTS und einen an MOTOR(EN)LINKS.



11. Nun ist es soweit, jetzt können wir wieder daran gehen das Testprogramm 5 zu laden, compilieren und zu übertragen

```

#####
'Testprogramm 5
'für
'RoboterNetz Standard-Roboter Board RBNFRA 1.2
'
'Aufgabe:
'Testen der Schrittmotortreiber
'Dreht erst linken Schrittmotor in beide Richtungen
'und dann rechten Schrittmotor in beide richtungen
'
'Autor: Frank
'Weitere Beispiele und Beschreibung der Hardware
'unter http://www.Roboternetz.de und http://www.mikrocontroller-elektronik.de/
'#####

Declare Sub Ledein
Declare Sub Ledaus

Const Writepowerport_adr = &H72           'I2C Adr PCF 2
Const Readpowerport_adr = &H73           'I2C Adr PCF 2

Dim I2cdaten As Byte                     'Datenbyte aus PCF8574

Dim I As Word

$crystal = 8000000                       'Quarzfrequenz
Config Scl = Portc.0                      'Ports fuer IIC-Bus
Config Sda = Portc.1
I2cinit

'***** Diese 4 Befehle sind nur ab RBNFRA Version 1.2 (nicht in V 1.1)
' notwendig und bzw. möglich (erweiterte Energiesparfunktion und LED's)
' Bei Board 1.1 bitte auskommentieren oder löschen
I2cstart
I2cwbyte &H74                             'Schreibbefehl an PCF3 schicken
' Led's ein ,Motorendstufen ein, Port-Peripherie ein, RBN-Bus Sleep Modus aus (also Peripherie
aktiv)
I2cwbyte &B00000010                       'Datenbyte an PCF3
I2cstop
'*****

Config Pind.6 = Output                    'Schrittmotoren Ein/Aus
Config Pinc.5 = Output                    'Schrittmotor Links Richtung
Config Pinc.3 = Output                    'Schrittmotor Links Step

Config Pinc.4 = Output                    'Schrittmotor Rechts Richtung
Config Pinc.2 = Output                    'Schrittmotor Rechts Step

Ledaus

```

```

Portd.6 = 0                                     'Schrittmotoren erst mal ausschalten

Wait 2

Ledein
Portd.6 = 1                                     'Beide Schrittmotoren einschalten

'Linker Schrittmotor
Portc.5 = 0                                     'Richtung
Portc.3 = 0
I = 400                                         'Anzahl der Schritte die Motor bewegt
werden soll
Do
  Ledaus
  Portc.3 = 0
  Waitms 5
  Portc.3 = 1
  Ledein
  Waitms 5
  I = I - 1
Loop Until I < 1

Portc.5 = 1                                     'Andere Richtung
I = 400                                         'Anzahl der Schritte die Motor bewegt
werden soll
Do
  Ledaus
  Portc.3 = 0
  Waitms 5
  Portc.3 = 1
  Ledein
  Waitms 5
  I = I - 1
Loop Until I < 1

Wait 2

'Rechter Schrittmotor
Portc.4 = 0                                     'Richtung
I = 400                                         'Anzahl der Schritte die Motor bewegt
werden soll
Do
  Ledaus
  Portc.2 = 0
  Waitms 5
  Portc.2 = 1
  Ledein
  Waitms 5
  I = I - 1
Loop Until I < 1

Portc.4 = 1                                     'Andere Richtung
I = 400                                         'Anzahl der Schritte die Motor bewegt
werden soll
Do
  Ledaus
  Portc.2 = 0
  Waitms 5
  Portc.2 = 1
  Ledein
  Waitms 5
  I = I - 1
Loop Until I < 1

Portd.6 = 0                                     ' Schrittmotoren erst mal ausschalten
Ledaus

End

Sub Ledein()
  I2cstart
  I2cwbyte Writepowerport_adr                 'Schreibbefehl an PCF schicken
  I2cwbyte 15                                 'Datenbyte an PCF
  I2cstop
End Sub

```

```

Sub Ledaus
  I2cstart
  I2cwbyte Writepowerport_adr           'Schreibbefehl an PCF schicken
  I2cwbyte 0                             'Datenbyte an PCF
  I2cstop
End Sub

```

Hat man alles richtig gemacht, dann sollte sich erst der eine Schrittmotor 400 Schritte in die eine und dann 400 Schritte in die andere Richtung drehen. Danach macht der andere Schrittmotor das gleiche.

Für die Ansteuerung der Schrittmotoren werden andere Ports als für die Getriebemotoren verwendet. Der I2C-Bus wird ebenfalls nicht benötigt (außer für Aktivierung der Entstufe.). Dennoch wurden in dem Beispiel einige I2C-Routinen genutzt um die LED's (am Powerport) mit einer Unteroutine ein und auszuschalten. Dadurch kann man besser demonstrieren wann die Schrittmotoren unter Spannung stehen und wann nicht. Zudem wird der einzelne Schritt durch ein kurze Flackern sichtbar gemacht.

Das Listing sieht dadurch vielleicht etwas komplizierter aus als es ist.

Die eigentlichen Routinen um den Motor eine genau definierte Anzahl von Schritten zu bewegen sind jedoch äußerst einfach:

```

'Rechter Schrittmotor
Portc.4 = 0           'Richtung
I = 400              'Anzahl der Schritte die Motor bewegt
werden soll
Do
  Ledaus
  Portc.2 = 0
  Waitms 5
  Portc.2 = 1
  Ledein
  Waitms 5
  I = I - 1
Loop Until I < 1

```

Da man die Schritte die sich der Motor bewegen soll genau definieren kann und man weiß wieviel Schritte der Motor pro Umdrehung benötigt, kann man in Verbindung mit dem Raddurchmesser genau berechnen wie weit der Roboter mit welcher Anzahl von Schritten bewegt wird.

Man kann das ganze also in eine Unterfunktion packen der man dann nur noch Fahrrichtung in Grad und Fahrtstecke in cm angibt. Die Unteroutine kann dann die jeweiligen Schritte berechnen und den Motor entsprechend ansteuern. So genau würde man das mit Getriebemotoren nie hin bekommen!

Vergessen darf man allerdings nicht, das Schrittmotoren auch im Stillstand den gleichen Strom (sogar etwas mehr) benötigen. Wenn der Roboter steht, sollte man deshalb die Motoren ausschalten.

```
Portd.6 = 0           ' Schrittmotoren erst mal ausschalten
```

Bei jedem Programm das im Schrittmotormodus arbeitet, sollte man schon bei Programmbeginn die Schrittmotoren auf diese Weise ausschalten, selbst wenn keine angeschlossen sein sollten. Es wird sonst unnötig Strom verbraucht.

Benötigt man keinesfalls Schrittmotoren dann sollte man die L297 aus der Fassung nehmen um noch einige mA zu sparen.

Nun wissen wir also auch das die beiden Schaltkreise L297 in Ordnung sind.

Achtung: Sollten sich die Schrittmotoren nicht drehen, so kann dies auch an den Fusebit-Einstellungen des AVR's liegen. Generell ist nämlich das JTAG-Interface aktiviert und dadurch kann der STEP-Impuls nicht an am Port ausgegeben werden. Daher muß unbedingt der JTAG-Modus ausgeschaltet (disable) werden. Wie dies geht, wurde ja schon bei dem ersten Beispielprogramm beschrieben.

Jetzt testen wir den Eingangsport und den letzten PCF

Jetzt gehen wir daran die zweite Porterweiterung (IC PCF2 PCF8574AP) zu testen. Dieser PCF-Baustein ist für die 8 freien Eingänge zuständig. Also hier können Sensoren (Mikroschalter, IR-Sensoren usw. angeschlossen werden).

Obwol der Port nur für Eingänge gedacht ist, können einzelne oder auch alle seine Portleitungen auch auf Ausgang geschaltet werden. Um möglichst Softwarekompatibel zu anderen Projekten mit diesem Board zu bleiben, sollte man dies allerdings nur tun wenn es wirklich notwendig ist.

Wir testen nun den Port indem wir einfach den gesamten Portwert als Byte über den I2C-Bus einlesen und den Wert an die serielle Schnittstelle schicken. Sie müssen also den PC mit der RS232 des Boards verbunden haben.

Ob das Board nun für den Getriebe- oder Schrittmotormodus konfiguriert ist spielt keinerlei Rolle für diesen Test.

Allerdings sollte man im Schrittmotormodus an den Anfang des Listings den schon beschriebenen Ausschaltbefehl für die Schrittmotoren setzen (spart Strom)

```
Portd.6 = 0                                     ' Schrittmotoren erst mal ausschalten
```

Ansonsten gleich wie immer: Testprogramm 6 laden, compilieren und übertragen.

```
#####
'Testprogramm 6
'für
'RoboterNetz Standard-Roboter Board RBNFRA 1.2
'
'Aufgabe:
'Testet Eingangsport indem Wert über RS232
'übermittelt wird (9600 Baud einstellen)
'
'Autor: Frank
'Weitere Beispiele und Beschreibung der Hardware
'unter http://www.Roboternetz.de und http://www.mikrocontroller-elektronik.de/
'#####

Const Writeeingabeport_adr = &H7E               'I2C Adr PCF 2
Const Readeingabeport_adr = &H7F               'I2C Adr PCF 2

Dim I2cdaten As Byte                           'Datenbyte aus PCF8574

Dim I As Byte

$baud = 9600
$crystal = 8000000                             'Quarzfrequenz
Config Scl = Portc.0                            'Ports fuer IIC-Bus
Config Sda = Portc.1

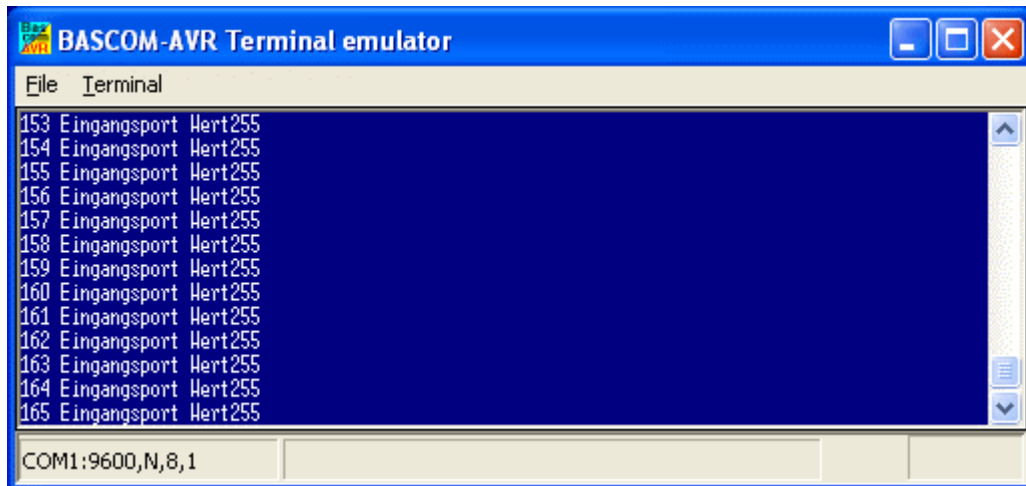
Portd.6 = 0                                     'Schrittmotoren erst mal ausschalten
I2cinit

'***** Diese 4 Befehle sind nur ab RBNFRA Version 1.2 (nicht in V 1.1)
' notwendig und bzw. möglich (erweiterte Energiesparfunktion und LED's)
' Bei Board 1.1 bitte auskommentieren oder löschen
I2cstart
I2cwbyte &H74                                   'Schreibbefehl an PCF3 schicken
' Led's ein ,Motorendstufen ein, Port-Peripherie ein, RBN-Bus Sleep Modus aus (also Peripherie
aktiv)
I2cwbyte &B00000010                             'Datenbyte an PCF3
I2cstop
'*****

I = 0
I2cdaten = 1
Do
    I2cstart
    I2cwbyte Readeingabeport_adr                 'Lesebefehl an PCF schicken
    I2crbyte I2cdaten , Nack                    'Datenbyte von PCF lesen
    I2cstop
    Waitms 50
    Print I ; " Eingangsport Wert" ; I2cdaten
    Incr I
Loop
```

End

Nachdem das Programm übertragen und damit auch gestartet wurde, muss man wieder das Terminalprogramm von Bascom aktivieren. Wenn nichts an den Eingangsports angeschlossen ist, sollten nun rechts lauter Werte in Höhe von 255 ausgegeben werden.



Geht man nun dazu über und schließt die Portleitungen mit einem Anschlusskabel an Masse an, dann verringert sich dieser Wert um das entsprechende Bit.
Beim Kurzschließen unbedingt darauf achten das nur die beiden linken Stifte der Anschlussstecker Portleitungen sind, also keinesfalls die Spannung-Pin's kurzschließen. Die genaue Steckerbelegung ist ja vorne in dieser Dokumentation beschrieben.

Sollte sich der Wert nicht verringern, dann haben Sie eventuell die falsche Slave-Adresse durch die Brücken ADR2 eingestellt oder die Slave Adresse im Programm falsch angegeben.

Batteriespannung messen

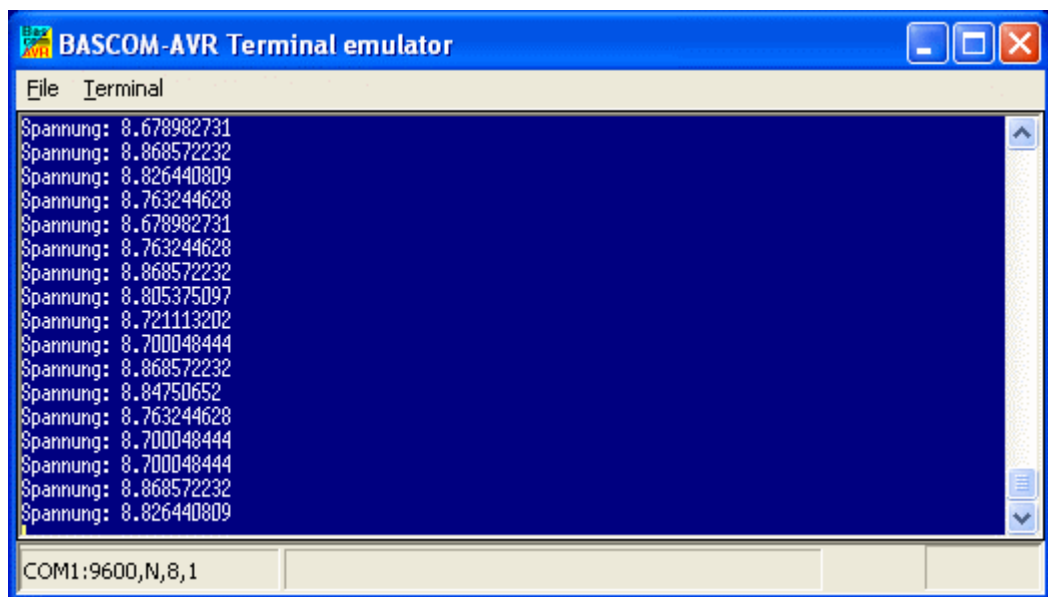
Das nächste Beispiel für den Hauptcontroller demonstriert wie man die Batteriespannung über einen analogen Port messen und über die RS232 anzeigen kann. Die erforderliche Hardware ist bereits auf dem Board integriert.

```
'#####  
'Testprogramm 7  
'für  
'RoboterNetz Standard-Roboter Board RBNFRA 1.2  
'  
' Aufgabe:  
' Dieses Testprogramm ermittelt die Batteriespannung  
' und gibt diese über die RS232 aus (9600 Baud einstellen)  
'  
'Autor: Frank  
'Weitere Beispiele und Beschreibung der Hardware  
'unter http://www.Roboternetz.de und http://www.mikrocontroller-elektronik.de/  
'#####  
  
Const Ref = 5 / 1023  
  
Dim I As Byte  
Dim W As Word  
Dim Volt As Single  
  
Config Adc = Single , Prescaler = Auto  
  
$baud = 9600  
$crystal = 8000000  
Portd.6 = 0  
  
Start Adc  
Do  
  W = Getadc(4)  
  Volt = W * Ref  
  Volt = Volt * 4.31  
  Print "Spannung: " ; Volt  
  
  Waitms 200  
Loop  
  
End
```

'Quarzfrequenz
'Schrittmotoren erst mal ausschalten

'end program

So sieht das Ergebnis aus:



Status LED's ansteuern

Ab Version 1.2 dieses Boards gibt es auch den neuen Energie-Port. Dieser unterstützt neben einigen interessanten Funktionen um Energie zu sparen auch 3 Status-LED's die frei zur Verfügung stehen und unabhängig vom Powerport benutzt werden können. Das nachfolgende Programm demonstriert wie man diese Status-LED's ansteuert indem ein Laufflicht ausgegeben wird.

```
#####
'Testprogramm 8
'für
'RoboterNetz Standard-Roboter Board RBNFRA 1.2
' nur für Board Version >>> 1.2 <<<
'
'Aufgabe:
'Hier wird demonstriert wie man die zusätzlichen
'Status LED's auf dem Board 1.2 gezielt ansteuert.
'Das Programm erzeugt ein kleines Laufflicht
'
'Autor: Frank
>Weitere Beispiele und Beschreibung der Hardware
'unter http://www.Roboternetz.de oder http://www.mikrocontroller-elektronik.de/
#####

Const Ergieport_adr = &H74                                'I2C Adr PCF 3

Dim I2cdaten As Byte                                    'Datenbyte aus PCF8574

Dim I As Byte

$crystal = 8000000                                       'Quarzfrequenz
Config Scl = Portc.0                                       'Ports fuer IIC-Bus
Config Sda = Portc.1

I2cinit

I = 0
I2cdaten = 1
Do
  'Die folgende Bitfolge bestimmt welche LED leuchtet
  'Die oberen 3 Bit's sind für die 3 LED's zuständig. Zu beachten ist
  'hier jedoch das eine 0 für eine leuchtende LED steht!!!
  'Hier wird die grüne LED aktiviert
  I2cdaten = &B11000000
  I2cstart
  I2cwbyte Ergieport_adr                                'Schreibbefehl an PCF schicken
  I2cwbyte I2cdaten                                     'Datenbyte an PCF
  I2cstop
  Waitms 300

  'Die folgende Bitfolge bestimmt welche LED leuchtet
  'Die oberen 3 Bit's sind für die 3 LED's zuständig. Zu beachten ist
  'hier jedoch das eine 0 für eine leuchtende LED steht!!!
  'Hier wird die gelbe LED aktiviert
  I2cdaten = &B10100000
  I2cstart
  I2cwbyte Ergieport_adr                                'Schreibbefehl an PCF schicken
  I2cwbyte I2cdaten                                     'Datenbyte an PCF
  I2cstop
  Waitms 300

  'Die folgende Bitfolge bestimmt welche LED leuchtet
  'Die oberen 3 Bit's sind für die 3 LED's zuständig. Zu beachten ist
  'hier jedoch das eine 0 für eine leuchtende LED steht!!!
  'Hier wird die rote LED aktiviert
  I2cdaten = &B01100000
  I2cstart
  I2cwbyte Ergieport_adr                                'Schreibbefehl an PCF schicken
  I2cwbyte I2cdaten                                     'Datenbyte an PCF
  I2cstop
  Waitms 300
Loop
End
```

Der neue Energie-Port

Das letzte Beispielprogramm demonstriert die Möglichkeiten des neuen Energieportes. Da nicht viel im sichtbaren Bereich geschieht, sollte man hier insbesondere auf die Kommentare achten. Im Grunde schaltet das Programm zuerst Motorentstufe (Logik-Spannung), Ein- und Ausgabeport (Logikspannung), Status-LED's und Peripherie am RNB-Bus ein und dann nach 5 Sekunden wieder aus. Das Programm macht also nicht viel, dennoch ist dieser Port sehr wichtig und vielseitig.

Jedes jedes Bit des PCF3 (Energieportes) hat spezielle Aufgaben. Die drei oberen Bits (7,6,5) 'xxx00000'sind mit den Status-Led's verbunden und 'werden auf JP12 bereitgestellt als I/O Ports bereitgestellt. Beachten muß man hier, 'das die LED's nur leuchten wenn das Bit auf 0 geschaltet wird

Die Bits 3+4 sind frei und werden auf JP12 als I/O Ports bereitgestellt.

Das Bit 2 schaltet die 5V Logikspannung, die auf den Steckern der Board Ein- und Ausgänge liegt aus. Damit kann Energie eingespart werden, falls 'Sensoren etc. angeschlossen sind. Ausgeschaltet wird die Spannung indem 'das Bit auf 1 gesetzt wird!

'Das Bit 1 ist das Sleep-Bit des 25 poligen RNBUS-Steckers. Gewöhnlich sollte 'dieses Bit auf 1 stehen. Wenn sie es auf 0 schalten, dann sollten andere Erweiterungen (Zusatzplatinen die über den RNB-Bus verbunden sind) in einen Energiesparmodus gehen. Dies muß jedoch vom jeweiligen Erweiterungsboard unterstützt werden

Das Bit 0 schaltet die Schrittmotor- und Getriebemotorentstufe aus. 'Werden keine Motoren benötigt oder soll der Roboter eine kleine Pause machen, so kann man hierdurch erheblich Energie sparen (insbesondere im Schrittmotormodus). Ausgeschaltet wird die Spannung indem das Bit auf 1 gesetzt wird!

```

#####
'Testprogramm 9
'für
'RoboterNetz Standard-Roboter Board RBNFRA 1.2
' nur für Board Version >>> 1.2 <<<
,
'Aufgabe:
'Hier werden die Energiesparmöglichkeiten
'erläutert. Dazu sollte der Quellcode
'und die Kommentarzeilen gelesen werden
'Das Programm schaltet zuerst alles ein und geht dann
'in einen Energiesparmode. Weitere Energie könnte gespart werden
'wenn man den Controller in den Sleep-Modus setzt
,
'Autor: Frank
'Weitere Beispiele und Beschreibung der Hardware
'unter http://www.Roboternetz.de und http://www.mikrocontroller-elektronik.de/
#####

Const Energieport_adr = &H74                                'I2C Adr PCF 3

Dim I2cdaten As Byte                                       'Datenbyte aus PCF8574

Dim I As Byte

$crystal = 8000000                                          'Quarzfrequenz
Config Scl = Portc.0                                       'Ports fuer IIC-Bus
Config Sda = Portc.1

I2cinit

***** Diese 4 Befehle sind nur ab RBNFRA Version 1.2 (nicht in V 1.1)
'Jedes jedes Bit des PCF3 (Energieport) hat spezielle Aufgaben
'Die drei oberen Bits (7,6,5) 'xxx00000'sind mit den Status-Led's verbunden und
'werden auf JP12 bereitgestellt als I/O Ports bereitgestellt. Beachten muss man hier,
'das die LED's nur leuchten wenn das Bit (Port) auf 0 geschaltet wird

'Die Bits 3+4 sind frei und werden auf JP12 als I/O Ports bereitgestellt.

'Das Bit 2 schaltet die 5V Logikspannung, die auf den Steckern der
'Board Ein- und Ausgänge liegt aus. Damit kann Energie eingespart werden, falls
'Sensoren etc. angeschlossen sind.Ausgeschaltet wird die Spannung indem
'das Bit auf 1 gesetzt wird!

```

```
'Das Bit 1 ist das Sleep-Bit des 25 poligen RBNBUS-Steckers. Gewöhnlich sollte
'dieses Bit auf 1 stehen. Wenn sie es auf 0 schalten, dann sollten andere
'Erweiterungen (Zusatzplatinen die über den RNB-Bus verbunden sind) in einen
'Energiesparmodus gehen. Dies muss jedoch vom jeweiligen Erweiterungsboard
'unterstützt werden

'Das Bit 0 schaltet die Schrittmotor- und Getriebemotorentstufe aus
'Werden keine Motoren benötigt oder soll der Roboter eine kleine Pause
'machen, so kann man hierdurch erheblich Energie sparen. Ausgeschaltet wird die Spannung indem
'das Bit auf 1 gesetzt wird!

' Dies ist die übliche Anweisung die in Programmen stehen sollte
' Mit dieser Codesequenz wird quasi alles aktiviert Eingabeports, Ausgabeports, Status-LED's,
' Motorentstufe, und SLEEPMODUS wird deaktiviert (also kein Energiesparmode)
I2cstart
I2cwrite Energieport_adr          'Schreibbefehl an PCF3 schicken
  ' Led's ein ,Motorentstufen ein, Port-Peripherie ein, RBN-Bus Sleep Modus aus (also Peripherie
aktiv)
I2cwrite &B00000010              'Datenbyte an PCF3
I2cstop
*****

Wait 5

'Hier erfolgt das Gegenteil. Alles was zum Energiesparen abgeschaltet werden kann, wird
abgeschaltet
I2cstart
I2cwrite Energieport_adr          'Schreibbefehl an PCF3 schicken
  ' Led's ein ,Motorentstufen ein, Port-Peripherie ein, RBN-Bus Sleep Modus aus (also Peripherie
aktiv)
I2cwrite &B11100101              'Datenbyte an PCF3
I2cstop

End
```

Der Co-Controller

Nachdem wir alle Grundfunktionen des Boards beschrieben haben, wenden wir uns nun an den Co-Controller. Denn neben dem Mega16 tut noch ein weiterer Atmel vom Typ AT90S2313P seinen Dienst. Auf dem Board sind also zwei völlig eigenständige Controller die auch getrennt programmiert werden können vorhanden. Beide haben ihren eigenen ISP-Anschluss und Quarz. Der Co-Controller arbeitet mit 4 Mhz etwas langsamer.

Beide Controller sind mit dem gleichen I2C-Bus verbunden. Somit könnte auch der Co-Controller Ausgabe und Eingangsports steuern. Er könnte sogar mit dem PC kommunizieren wenn er über Jumper JP16 mit dem Max verbunden wird.

Er könnte sich auch per I2C mit dem Hauptcontroller austauschen. Es dürfen allerdings niemals gleichzeitig bei Controller auf dem I2C-Port als Master aktiv sein. Möchte man dieses jedoch machen, so muss man sich entsprechende Befehle ausdenken um dem anderen Controller mitzuteilen das dieser aktiv werden kann. Dieser wiederum muß dann später wieder den anderen Controller aktivieren.

Unterstützen kann man diese I2C Kommunikation noch, indem man über Jumper JP16 eine Portleitung des Co-Controllers mit einer Interruptleitung des Hauptcontrollers verbindet. Diese Leitung wird auf Low gesetzt, wenn ein I2C-Baustein etwas von dem Hauptcontroller möchte. Da dies mehrere Bausteine am I2C-Bus machen können, muss der Hauptcontroller in einem solchen Fall alle I2C Komponenten überprüfen.

Diese Aufgaben sind aber mehr etwas für den erfahreneren Programmierer. Derzeit steht auch noch kein solches Testprogramm bereit, daher sollten auch an Jumper JP16 keine Kurzschlussstecker gesetzt werden.

Die Standard-Aufgabe des Co-Controllers ist eigentlich das steuern von bis zu 10 Servos. Dazu wurden auch alle notwendigen Portleitung auf Servo kompatiblen Stiftleisten herausgeführt. Servos können so direkt angesteckt werden.

Um die Servos steuern zu können, muss noch ein Servo-Steuerprogramm in den CoController geladen werden. Derzeit ist dieses noch nicht ganz fertig. Ein Mitglied aus dem Roboternetz Nicknamen Kjion arbeitet schon eine Weile daran. Er will in Kürze den Code zum Download frei geben. Danach kann man den CoProzessor kann normal wie ein anderen Schaltkreis per I2C Befehle zum Bewegen der Servos übermitteln. So lassen sich Servos sehr komfortabel steuern ohne den Hauptcontroller in irgend einer Weise zu belasten.

Wie schon gesagt, wer keine Servos braucht, könnte die 10 Ports des Co-Controllers, sowie dessen Rechenleistung, auch ganz anders nutzen. Hier kommt es in Zukunft einfach auf den Einfallsreichtum der Programmierer an.

An dieser Stelle will ich nur ein Testprogramm vorstellen damit auch festgestellt werden kann ob der Co-Controller korrekt funktioniert.

Dazu muss nun der ISP Stecker auf die daneben liegende Stiftleiste ISP2 gesteckt werden.

Nun kann das Testprogramm „Co-Controller Testprogramm1“ wie gewohnt geladen, kompiliert und übertragen werden. Vor dem Kompilieren und Übertragen muss aber diesmal ein anderer Atmel Chip ausgewählt werden, nämlich der 90S2313.

Ansonsten unterschieden sich die Programme quasi gar nicht oder kaum. Statt 8 Mhz muss hier halt `$crystal = 4000000` angegeben werden.

Aus diesem Grund verwenden wir zum Test nahezu das gleiche Testprogramm das wir vom Hauptcontroller schon kennen. Wir lassen diesmal das Lauflicht vom Co-Controller aus steuern.

Der einzige Unterschied ist, das wir dies nur einige male und nicht unendlich wie beim Co-Controller machen lassen. In diesem Beispiel arbeitet also der Co-Controller als I2C-Busmaster.

Bei solchen Experimenten muss man nur aufpassen das nicht im Haupt- und Co-Controller gleichzeitig ein Programm vorhanden ist, welches auf den I2C-Bus zugreift. In solchen Fällen könnten verwirrende Ergebnisse oder Abstürze die Folge sein. Also vorher immer vergewissern das der andere Controller den I2c-Bus in Ruhe lässt!. Am einfachsten ist es natürlich wenn man auch den Co-Controller als Slave programmiert. In dem Fall meldet er sich also nur zu Wort, wenn er was gefragt wird. So etwas ist ja generell besser ;-)

```
#####  
'CoController Testprogramm 1  
'für  
'RoboterNetz Standard-Roboter Board RBNFRA 1.2  
'  
'Aufgabe:  
'CoController übernimmt nun  
'Ausgabe über PowerPort per I2C testen indem  
'die vier Led's im lauflichtartig 50 mal leuchten läßt  
'Danch ist er deaktiv  
'  
'Autor: Frank  
'Weitere Beispiele und Beschreibung der Hardware
```



```
'unter http://www.Roboternetz.de und http://www.mikrocontroller-elektronik.de/
'#####

Const Writepowerport_adr = &H72           'I2C Adr PCF 2
Const Readpowerport_adr = &H73           'I2C Adr PCF 2

Dim I2cdaten As Byte                     'Datenbyte aus PCF8574

Dim I As Byte

$crystal = 4000000                       'Quarzfrequenz
Config Scl = Portd.3                     'Ports fuer IIC-Bus
Config Sda = Portd.2

I2cinit

I = 0
I2cdaten = 1
For I = 1 To 200
    I2cdaten = I2cdaten * 2
    If I2cdaten > 16 Then I2cdaten = 1

    I2cstart
    I2cwbyte Writepowerport_adr          'Schreibbefehl an PCF schicken
    I2cwbyte I2cdaten                    'Datenbyte an PCF
    I2cstop
    Waitms 100
Next

End
```

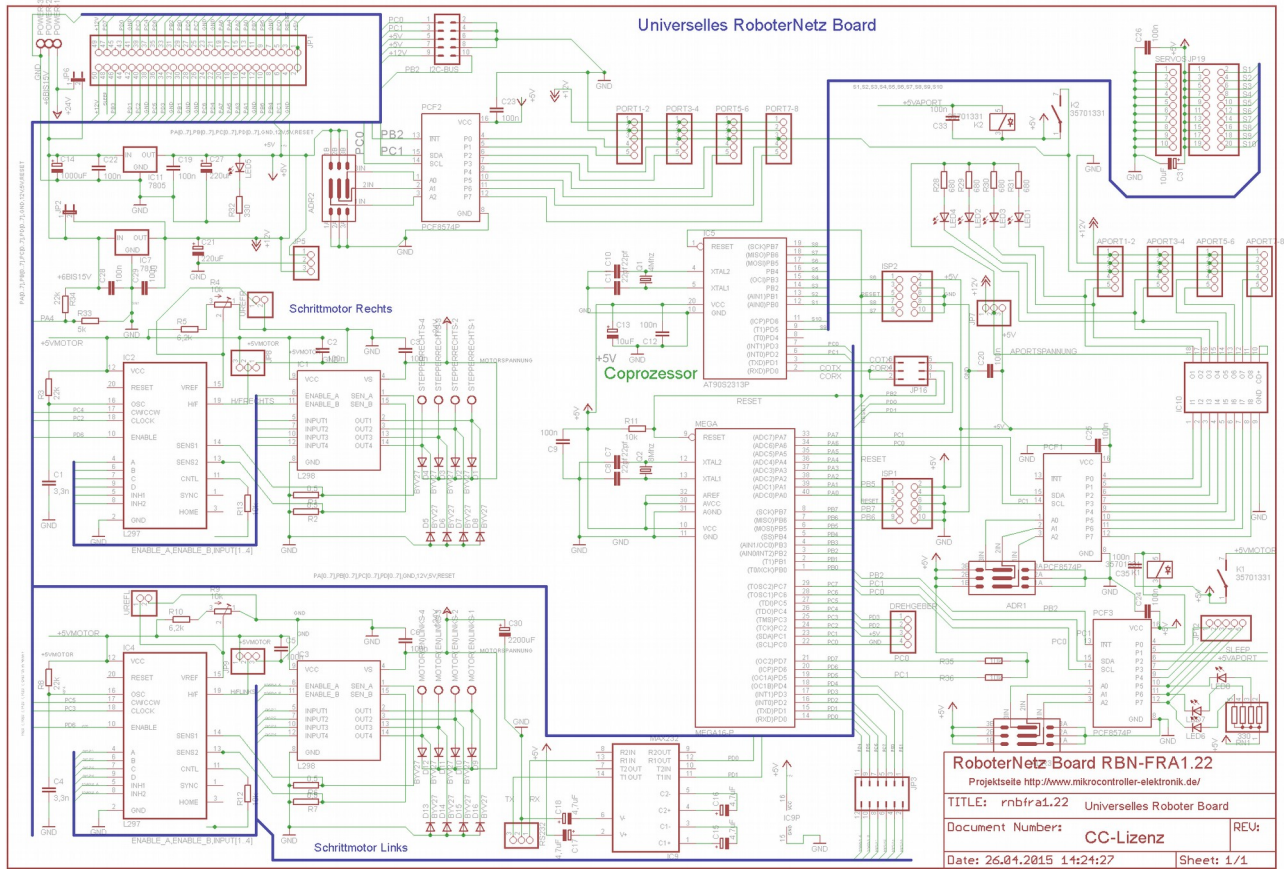
Zusammenfassung der wichtigsten Board-Adressen

Funktion	Port	Logische Bedeutung und Beispiel
Linker Getriebemotor Ein/ Aus	PD4	PD4 = 0=Aus 1=Ein Bascom-Beispiel: <i>Config Pind.4 = Output</i> <i>Portd.4 = x</i> Im PWM-Mode: <i>Pwm1a = 0 oder Geschwindigkeitl 0-1023</i>
Linker Getriebemotor Drehrichtung	PC6 PC7	Erste Richtung: PC6=0 PC7=1 Zweite Richtung PC6=1 PC7=0 Bascom-Beispiel: <i>Config Pinc.6 = Output</i> <i>Config Pinc.7 = Output</i> <i>Portc.6 = 1</i> <i>Portc.7 = 0</i>
Rechter Getriebemotor Ein/ Aus	PD5	PD5 = 0=Aus 1=Ein Bascom-Beispiel: <i>Config Pind.5 = Output</i> <i>Portd.5 = x</i> Im PWM-Mode: <i>Pwm1b = 0 oder Geschwindigkeitl 0-1023</i>
Rechter Getriebemotor Drehrichtung	PB0 PB1	Erste Richtung: PB=0 PB=1 Zweite Richtung PB=1 PB=0 Bascom-Beispiel: <i>Config Pinb.0 = Output</i> <i>Config Pinb.1 = Output</i> <i>Portb.0 = 1</i> <i>Portb.1 = 0</i>
Beide Schrittmotoren Ein/Aus	PD6	PD6 = 0=Aus 1=Ein Bascom-Beispiel: <i>Config Pind.6 = Output</i> <i>Portd.6 = 1</i>
Linker Schrittmotor Drehrichtung	PC5	PC5 = 0=Links 1=Rechts Bascom-Beispiel: <i>Config Pinc.5 = Output</i> <i>Portc.5 = 1</i>
Schrittipuls für linken Schrittmotor	PC3	PC3=Impuls Bascom-Beispiel: <i>Config Pinc.3 = Output</i> <i>Do</i> <i>Portc.3 = 0</i> <i>Waitms 5</i> <i>Portc.3 = 1</i> <i>Waitms 5</i> <i>Loop Until</i>
Rechter Schrittmotor Drehrichtung	PC4	PC4 = 0=Links 1=Rechts Bascom-Beispiel: <i>Config Pinc.4 = Output</i> <i>Portc.4 = 1</i>

Schrittimпуль für rechten Schrittmotor	PC2	PC2=Impuls Bascom-Beispiel: <i>Config Pinc.2 = Output</i> <i>Do</i> <i>Portc.2 = 0</i> <i>Waitms 5</i> <i>Portc.2 = 1</i> <i>Waitms 5</i> <i>Loop Until</i>
Drehgeber Impuls linkes Rad	PD2 (INT0)	Kann durch Interrupt ausgewertet werden. Beispiel folgt im Roboternetz
Drehgeber Impuls rechtes Rad	PD3 (INT1)	Kann durch Interrupt ausgewertet werden. Beispiel folgt im Roboternetz
RS232 TX	PD1	Bascom-Beispiel: <i>Print „Hallo“</i>
RS232 RX	PD0	Bascom-Beispiel: <i>Liest 3 Byte ein:</i> <i>Dim a as Byte, c as integer</i> <i>Inputbin a,c</i> Oder: <i>Input „warte auf zwei zeichen“,x,y</i>
AD Port0 IR-Entfernung Links vorne	PA0	Bascom-Beispiel: <i>Config Adc = Single , Prescaler = Auto</i> <i>Start Adc</i> <i>W = Getadc(0)</i>
AD Port1 IR-Entfernung Mitte vorne	PA1	Wie zuvor
AD Port2 IR-Entfernung Rechts vorne	PA2	**Wie zuvor
AD Port3 IR-Entfernung Boden	PA3	Wie zuvor
AD-Port4 Batteriespannung	PA4	Bascom-Beispiel: <i>Const Ref = 5 / 1023</i> <i>Config Adc = Single , Prescaler = Auto</i> <i>Start Adc</i> <i>W = Getadc(4)</i> <i>Volt = W * Ref</i> <i>Volt = Volt * 4.31</i>
AD-Port5 Tastatur	PA5	Wie zuvor
AD-Port6 Frei	PA6	Wie zuvor
AD-Port7 Frei	PA7	Wie zuvor
I2C BUS SCL	PC0	Bascom-Beispiel: <i>I2cstart</i> <i>I2cwbyte Writepowerport_adr</i> <i>I2cwbyte 15</i> <i>I2cstop</i>
I2C BUS SDA	PC1	Bascom-Beispiel: <i>I2cstart</i> <i>I2cwbyte Readpowerport_adr</i> <i>I2crbyte I2cdaten , Nack</i> <i>I2cstop</i>
Powerport	I2C Slave Empfohlen wird 72/73	Bascom-Beispiel: <i>Const Writepowerport_adr = &H72</i> <i>Const Readpowerport_adr = &H73</i> <i>I2cstart</i> <i>I2cwbyte Writepowerport_adr</i> <i>I2cwbyte 15</i> <i>I2cstop</i>
Eingangsport	I2C Slave Empfohlen wird 7E/7F	Bascom-Beispiel: <i>Const Writeeingabeport_adr = &H7E</i> <i>Const Readeingabeport_adr = &H7F</i> <i>Do</i> <i>I2cstart</i> <i>I2cwbyte Readeingabeport_adr</i> <i>I2crbyte I2cdaten , Nack</i> <i>I2cstop</i> <i>Waitms 400</i> <i>Print I ; " Eingangsport Wert" ; I2cdaten</i> <i>Incr I</i> <i>Loop</i>

<p>Energieport</p>	<p>I2C Slave Empfohlen wird 74/75</p>	<p>Bascom-Beispiel: <i>Dieses Beispiel schaltet alle Energiesparfunktionen aus und aktiviert Motorentstufe, Status-LED's, Ein- und Ausgabepotentialspannung und Peripherie am RNB-Bus</i> <i>I2cinit</i> <i>I2cstart</i> <i>I2cwbyte &H74</i> <i>I2cwbyte &B00000010</i> <i>I2cstop</i></p> <p>Hinweis: Jedes jedes Bit des PCF3 (Energieport) hat spezielle Aufgaben. Die drei oberen Bits (7,6,5) 'xxx00000'sind mit den Status-Led's verbunden und 'werden auf JP12 bereitgestellt als I/O Ports bereitgestellt. Beachten muß man hier, 'das die LED's nur leuchten wenn das Bit auf 0 geschaltet wird</p> <p>Die Bits 3+4 sind frei und werden auf JP12 als I/O Ports bereitgestellt.</p> <p>Das Bit 2 schaltet die 5V Logikspannung, die auf den Steckern der Board Ein- und Ausgänge liegt aus. Damit kann Energie eingespart werden, falls 'Sensoren etc. angeschlossen sind. Ausgeschaltet wird die Spannung indem 'das Bit auf 1 gesetzt wird!</p> <p>'Das Bit 1 ist das Sleep-Bit des 25 poligen RBNBUS-Steckers. Gewöhnlich sollte 'dieses Bit auf 1 stehen. Wenn sie es auf 0 schalten, dann sollten andere Erweiterungen (Zusatzplatinen die über den RNB-Bus verbunden sind) in einen Energiesparmodus gehen. Dies muß jedoch vom jeweiligen Erweiterungsboard unterstützt werden</p> <p>Das Bit 0 schaltet die Schrittmotor- und Getriebemotorentstufe aus. 'Werden keine Motoren benötigt oder soll der Roboter eine kleine Pause machen, so kann man hierdurch erheblich Energie sparen (insbesondere im Schrittmotormodus). Ausgeschaltet wird die Spannung indem das Bit auf 1 gesetzt wird!</p>
---------------------------	---	--

Schaltplan zum Board Version 1.2



Projektseite <http://www.mikrocontroller-elektronik.de/> Forum <http://www.roboternetz.de>

Projekt (Schaltung & Projektdateien) von Frank ist lizenziert unter einer Creative Commons Namensnennung – Nicht-kommerziell – Weitergabe unter gleichen Bedingungen 4.0 International Lizenz. Über diese Lizenz hinausgehende Erlaubnisse können Sie unter <http://www.mikrocontroller-elektronik.de/> erhalten.

Achtung: Es kann keinerlei Garantie für die Fehlerfreiheit der Schaltung oder anderer Projektdateien übernommen werden. Der Nachbau und Betrieb geschieht auf eigene Gefahr! Jegliche Haftung für Schäden wird ausgeschlossen. Schadensersatzansprüche, gleich aus welchem Rechtsgrund, sind ausgeschlossen.

Ein etwas größerer Schaltplan als auch die Eagle Dateien selbst findet ihr auf der Projektseite <http://www.mikrocontroller-elektronik.de>

Projekt-Seite zum herunterladen: <http://www.mikrocontroller-elektronik.de/>

Forum-Seite für Fragen: <http://www.roboternetz.de/community/forum.php>

Weiterentwicklungen können gerne ebenfalls auf der [Projektseite](#) unter CC-Lizenz veröffentlicht werden!



Dieses Projekt (Schaltung und Projektdateien) steht unter der Creative-Commons-Lizenz Namensnennung - Nicht-kommerziell - Weitergabe unter gleichen Bedingungen 4.0 International. Um eine Kopie dieser Lizenz zu sehen, besuchen Sie <http://creativecommons.org/licenses/by-nc-sa/4.0/>

Lizenziert wurde das Projekt von: User Frank www.Roboternetz.de & www.Mikrocontroller-Elektronik.de
Dieser Name und diese Webseiten sind bei der Weitergabe stets deutlich sichtbar zu nennen!
Über diese Lizenz hinausgehende Erlaubnisse können Sie unter <http://www.mikrocontroller-elektronik.de/> erhalten.

Achtung: Der Nachbau und Betrieb geschieht auf eigene Gefahr! Jegliche Haftung für Schäden wird ausgeschlossen!
Es kann keinerlei Garantie für die Fehlerfreiheit der Schaltung oder anderer Projektdateien übernommen werden!