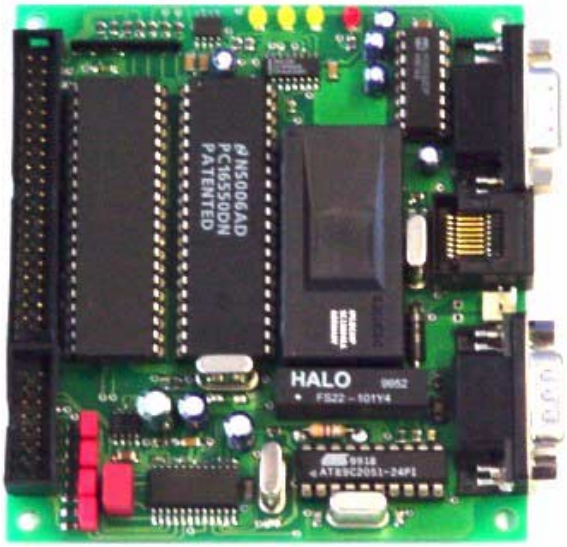
DeviLAN & CAN2Web

Universal WebInterface und Schnittstellenkoppler

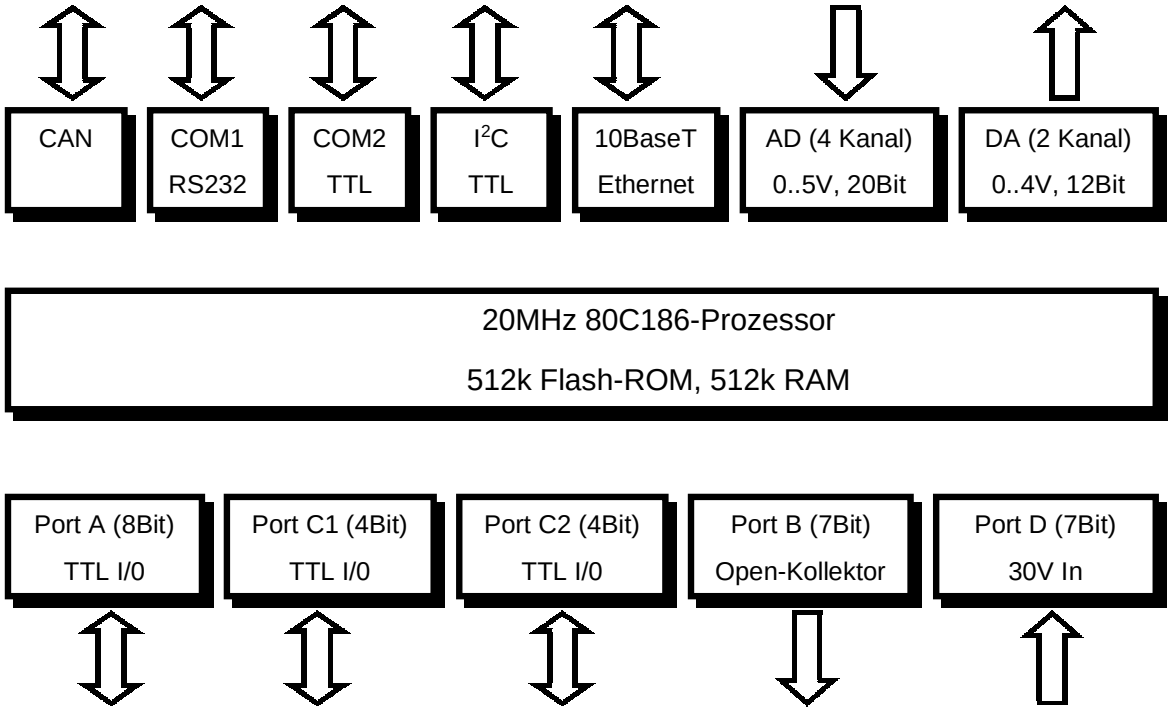
Handbuch

(Version vom 07.09.2004)

**für DeviLAN-Basic und DeviLAN-24 (SW-Version 1.06),
CAN2Web (SW-Version 1.02),
und Windows-Programm DeviLANControl (SW-Version 1.06)**



- **20MHz 80C186-Prozessor**
- **10BaseT Ethernet-Interface**
- **Embedded Web-Server, FTP, Telnet, CGI und WAP**
- **512k Flash-ROM, 512k RAM**
- **1 MBit CAN-Bus mit FIFO**
- **2 COM-Ports**
- **30 I/O-Ports (TTL, 24V und Open-Kollektor)**
- **4-Kanal 20 Bit ADC, 2-Kanal 12 Bit DAC**
- **inkl. Treiber für CAN, ADC, DAC und I/O**
- **PC-Programm DeviLANControl für Netzwerk-, RS232- und Modembetrieb**
- **Einfache Installation durch Plug&Play**



DeviLAN

Universal WebInterface

1 Wesentliche Merkmale

DeviLAN ist ein universelles Kommunikationsmodul, das dafür entwickelt wurde auf einfachste Art und Weise, bestehende Komponenten und Geräte an das Intra- und Internet anzubinden. Auf einer einzigen Einheit sind die weit verbreiteten Schnittstellen Ethernet, CAN-Bus und RS232 sowie diverse digitale und analoge Ports implementiert.

Das Gerät enthält einen TCP/IP-Stack mit integriertem Webserver. Über einen seriemäßigen CGI-Client können Web-Applikationen interaktiv auf Daten des DeviLAN-Moduls zugreifen. Hierbei sind alle Protokollvarianten wie HTTP, FTP, UDP, TCP und Telnet möglich. Eigene Anwendungen können problemlos mit gängigen DOS-Entwicklungstools erstellt werden. Umfangreiche Treiberbibliotheken erlauben einen einfachen und schnellen Hardwarezugriff.

Die max. Datenübertragungsrate des internen COM-Ports beträgt 430,4kBit, was eine bis zu 4-fach höhere Datenrate als die Standard-Schnittstelle eines PCs zulässt.

Die Implementation eines hochauflösenden 20 Bit A/D-Wandlers prädestiniert DeviLAN auch für anspruchsvolle Messaufgaben. Hierfür stehen 4 differentielle Eingänge zur Verfügung. Ein 2-Kanal 12 Bit D/A-Wandler ermöglicht im Gegenzug die Ansteuerung von analogen Sensoren und Umsetzern. Aufrund der hohen Messwertrate von max. 1000 Abtastwerten pro Sekunde und der Performance-optimierten Schnittstellenfunktionen eignet sich das Modul auch für Steuerungsaufgaben auf PC-Basis.

7 Leistungsausgänge erlauben die direkte Ansteuerung von induktiven Lasten wie Relais und Motoren bis 30V, ohne dass zusätzliche Treiberbausteine angeschlossen werden müssen. Für den Einsatz in Industrieautomationslösungen stehen 24V-Eingänge zur Verfügung. 16 TTL I/O-Ports und ein I2C-Interface erlauben umfangreiche Erweiterungsmöglichkeiten.

Einsatzgebiete sind Industrieautomation, Mess-, Steuerungs- und Regelungstechnik, Systemintegrationslösungen, Analytik Energietechnik und Gebäudemanagementsysteme. Anwendungsmöglichkeiten sind die Fernkonfiguration, -wartung und -überwachung sowie die standortübergreifende Vernetzung mehrerer Systeme durch globalen Datenaustausch über das Internet.

1.1 Anschlüsse

Durch On-Board-Integration aller notwendigen Anschlussverbinder kann das DeviLAN-Modul sofort ohne weitere Entwicklungsarbeit und Zusatzkomponenten in Betrieb genommen werden. Der Anschluss ans Ethernet erfolgt mit einem handelsüblichen Netzwerkkabel über eine RJ45-Buchse. Als Verbindung an ein CAN-Bus Netzwerk kommt ein 9-poliger SUB-D Stecker zum Einsatz. Die Terminierung des CAN-Busnetzes erfolgt über einen Jumper auf dem Board.

Um Verwechslungen zwischen CAN-Bus und RS232 Interface zu vermeiden, findet eine 9-polige Sub-D Buchse für die serielle Schnittstelle Verwendung. Der Anschuß ist als Datenterminal-Endgerät konfiguriert (DCE), so dass zur PC-Verbindung eine 1:1-Verlängerung zum Einsatz kommt.

Alle digitalen I/O-Ports, die I2C-Schnittstelle und der zweite COM-Port (TTL-Pegel) stehen an einer 40-poligen Stiftleiste zur Verfügung. Die analogen Ein- und Ausgänge sowie zwei Referenzspannungen belegen eine 14-polige Stiftleiste.

1.2 Software

Das DeviLAN-Modul wird mit Treibern für C/C++ ausgeliefert, die es ermöglichen eigene Softwareapplikationen für das Modul zu entwicklen. Als Entwicklungsumgebung reicht dafür ein einfacher Dos-C/C++ Compiler. Treiber stehen für die Zugriffe auf die seriellen Schnittstellen, 24V Ein-/Ausgänge, I/O-Ports, den CAN-Bus sowie für den A/D- und D/A-Wandler zur Verfügung. Als Betriebssystem kommt ein Echtzeit-RTOS zum Einsatz, das mit Multikasking-Fähigkeiten aufwartet.

Zusätzlich wird ein FTP-, Telnet- und Webserver mitgeliefert. Zur Datenauswertung über HTTP steht ein CGI-Client zur Verfügung. Standardmäßig kann dadurch jedes DeviLAN-Modul mittels eines Webbrowsers vollständig konfiguriert und bedient werden. Zusätzlich verfügt das DeviLAN-Modul auch über eine WAP-Schnittstelle, die eine Bedienung über ein Wap-fähiges Handy gestattet. Weiterhin verfügt es über mehrere Datenstrom-Schnittstellen, die es erlauben Daten kontinuierlich über Socket-Verbindungen an weitere Applikationen zu senden.

Als weitere Applikation steht das PC-Programm DeviLANControl zur Verfügung. Mit ihm läßt sich das DeviLAN-Modul sowohl über das Netzwerk als auch über eine serielle Schnittstelle des PCs bedienen. Durch die Verbindung mit einen externen Modem kann DeviLAN auch direkt über eine Telefonleitung angewählt werden.

2 Inbetriebnahme

2.1 Installation

Die Installation eines DeviLAN-Moduls erfolgt in drei Schritten:

1. Anbindung des Moduls an eine stabilisierte Gleichspannungsquelle

   DeviLAN-Basic:
   (5 V, mind. 280 mA). Die maximale Stromaufnahme ist von der angebunden Hardware abhängig.

   DeviLAN-24:
   (8 V...30V, Leistungsaufnahme typ. 1,4W). Die maximale Leistungsaufnahme ist von der angebundenen Hardware abhängig.

2. Anbindung des Moduls an die gewünschte Hardware über die bereitgestellten I/Os, COM- und CAN-Schnittstellen.

3. Anbindung an das Rechnernetzwerk (10 MBit) oder Verbinden mit einem lokalen PC über die serielle Schnittstelle (RS232).

Nach dem Einschalten ist das Modul nach ca. 8 s betriebsbereit und kann über einen Webbrowser oder mit dem PC-Programm DeviLANControl konfiguriert und bedient werden.

2.2 Sicherheitshinweise

Steckverbindungen:

Stecken oder ziehen Sie die Steckverbinder nie im laufenden Betrieb an/ab, da dies ggf. zur Zerstörung von Bauteilen führen kann!

Trennen Sie das Modul zunächst immer von der Spannungsversorgung und führen Sie dann die gewünschten Änderungen aus.

Berührschutz:

Schützen Sie das Modul vor Überspannungen durch elektrostatische Auf- und Entladungen.

3 Bedienung über Webbrowser

Ist die IP-Adresse bereits festgelegt, kann das Modul direkt in ein Rechnernetzwerk integriert werden. Nach der Inbetriebnahme erfolgt die Konfiguration eines Moduls folgendermaßen:

1. Start eines Webbrowsers z. B. Netscape Navigator oder Internet Explorer

2. Aufruf der DeviLAN-Startseite durch Eingabe der IP-Adresse in das Adressfeld des Browsers

3. Es erscheint die DeviLAN-Startseite über die die verschiedenen Webseiten für die Konfiguration und Datenabfrage angewählt werden kann (Abb. 1). Auf der linken Seite befindet sich das Hauptmemü über das die verschiedenen Untermenüs angewählt werden können.
   Die Webseiten enthalten für die Konfiguration und den Test der Hardware verschiedene Bedienelemente wie Textfelder, Radio-Button oder Listboxen. Über diese können die gewünschten Einstellungen vorgenommen und Daten an das DeviLAN-Modul gesendet bzw. vom ihm gelesen werden. Die Schnittstellen wurden bewusst einfach gehalten, wobei auf die Verwendung von JavaScript-Funktionen und Java-Applets verzichtet wurde.

4. Das Webserver/CGI-Interface erlaubt folgende Funktionen auszuführen:

   Festlegen der Ein- und Ausgänge der digitalen Ports, Konfiguration der COM und CAN-Schnittstelle(n), Aktivieren und Deaktivieren der einzelnen Schnittstellen, Schalten der digitalen Ausgänge, Abfragen (einzeln bzw. kontinuierlich) der digitalen Eingänge, AD-Wandlerkanäle, empfangenen CAN-Meldungen sowie der empfangenen Daten der beiden seriellen Schnittstellen. Senden von Daten an die beiden seriellen Schnittstellen

und den CAN-Bus.

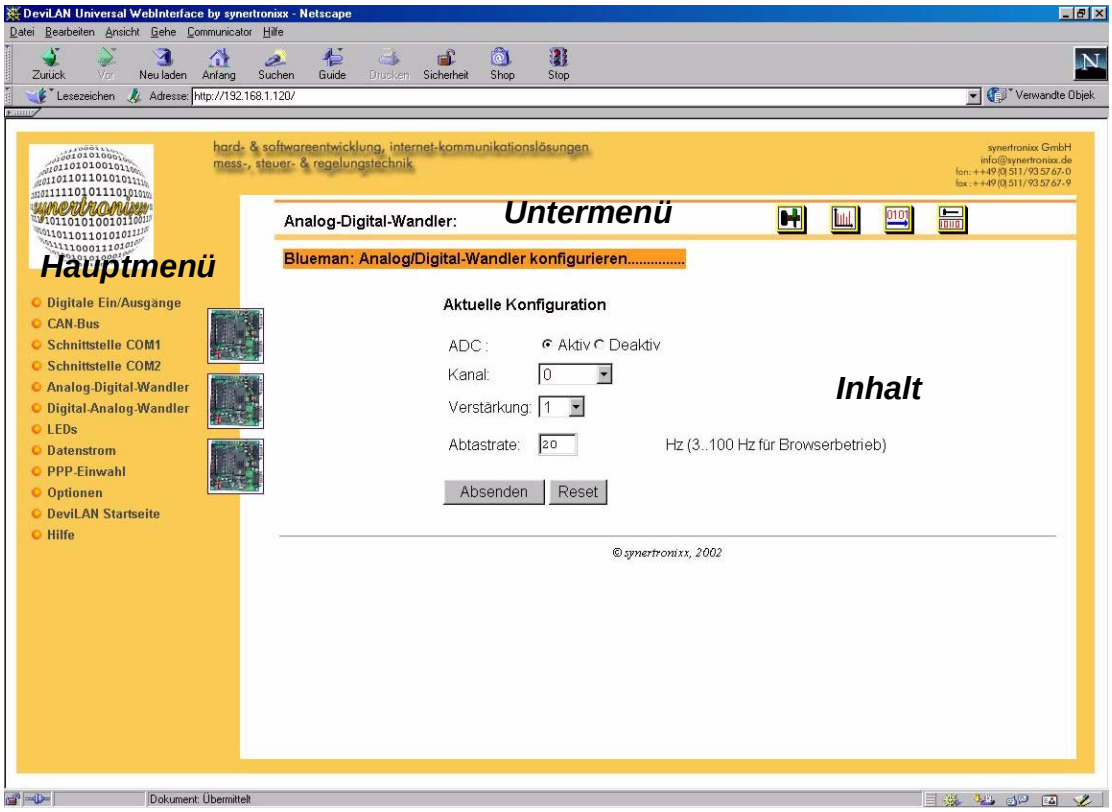


Abb. 1: DeviLAN Menüführung und Bedienung über Webbrowser

Weitere Informationen über Konfiguration, Bedienung und Inbetriebnahme mittels eines Webbrowsers befinden sich auch im Untermenü „Hilfe“.

Zum Konfigurieren und Testen der einzelnen Schnittstellen (CAN, COM1, etc.) wählen Sie auf der linken Seite den gewünschten Menüpunkt aus. Es erscheint dann das entsprechende Untermenü in Form einer Symbol-Leiste sowie die Konfigurationsseite für die angewählte Schnittstelle.

Die Konfiguration erfolgt in drei Schritten:

1. Konfiguration der angewählten Schnittstelle (CAN, COM1, etc.) über die bereitgestellten Oberflächenelemente (Textfelder, Radio-Button, Listboxen). Über den Button „Reset“ werden die ursprünglichen Einstellungen wieder hergestellt.

2. Drücken des Buttons „Absenden“. Die Einstellungen werden an das DeviLAN-Modul übertragen, überprüft und entsprechend geändert.

3. Es erscheint eine Webseite mit den geänderten Einstellungen.

Um die aktuelle Konfiguration zu speichern wählen sie im linken Menü „Weitere Optionen“. Drücken Sie anschließend auf der Webseite den Button „Speichern“. Alle aktuel-

len Einstellungen werden im EEPROM des Moduls gespeichert und bei einen Neustart (Power-On) automatisch geladen.

Der Zugriff auf die Schnittstelle erfolgt ebenfalls in drei Schritten:

1. Ändern der gewünschten Einstellungen der ausgewählten Schnittstelle (CAN, COM1, etc.) über die bereitgestellten Oberflächenelemente (Textfelder, Radio-Button) vor. Über den Button „Reset“ werden die ursprünglichen Einstellungen wieder hergestellt.

2. Drücken des Buttons „Absenden“. Die Daten werden an das DeviLAN-Modul übertragen und entsprechend geändert.

3. Es erscheint eine Webseite mit den geänderten Einstellungen/Werten.

4 Bedienung über ein WAP-fähiges Handy

Das DeviLAN-Modul verfügt neben einem Web-Server zusätzlich auch über einen WAP-Server mit CGI-Interface, dass eine Bedienung und Abfrage von Daten über eine WAP-fähiges Mobiltelefon gestattet. Da diese Telefone in der Regel nur mit einer sehr geringen Datenrate von 9600 Baud arbeiten und keine Anzeigemöglichkeit für einen kontinuierlichen Datenstrom besitzen, gestattet das WAP-Interface nur ein Konfigurieren, Schalten und Abfragen der digitalen Ein- und Ausgänge. Ein Zugriff auf die seriellen Schnittstellen sowie den CAN-bus ist in der momentanen Version und Ausbaustufe nicht realisiert.

Der Zugriff auf das WAP/CGI-Interface gestaltet sich folgendermaßen:

1. Auswahl der WAP-Startseite des Moduls in der Form http://<ip>/index.wml also z. B. http://130.76.65.253/index.wml
2. Es erscheint ein hierachisch aufgebautes Menü über das man die einzelnen Untermenüs für AD-Wandler, DA-Wandler sowie die digitalen Ein- uns Ausgänge zu erreichen sind. Innerhalb der Untermenüs können dann die Konfigurations- und Aktionsseiten aufgerufen werden.
3. Ändern der Einstellungen über die entsprechenden Auswahlboxen und Eingabefelder.
4. Absenden der Einstellungen/Daten über die Auswahl der Links „Schalten“, „Setzen“ etc.
5. Es erscheint die WAP-Seite mit den geänderten Einstellungen.

Wie der Zugriff auf WAP-Seiten, die Eingabe von Daten und Auswahl von Einstellungen erfolgt, ist dabei der Bedienungsanleitung des verwendeten Handys zu entnehmen.

5 Bedienung über DeviLANControl

Das Programm DeviLANControl dient als Kommunikationsprogramm mit einem DeviLAN-Modul. Die Anbindung erfolgt entweder direkt über eine serielle Schnittstelle, das LAN oder Telefon erfolgen. Für den letzten Fall muss der PC-über ein internes oder externes Modem und das DeviLAN-Modul über externes Modem, das an die serielle Schnittstelle COM1 des Moduls angebunden ist, erfolgen.

Die Konfiguration der Parameter IP-Adresse, Netmask und Gateway kann aus Sicherheitsgründen nur über die serielle Schnittstelle erfolgen.

5.1 Dialogelemente von DeviLANControl

Das Programm DeviLANControl verfügt über fünf Dialogboxen:

DeviLAN-Manager: gestattet ein schnelles Finden und Kontaktieren von Modulen im Netzwerk sowie die Auswahl des Verbindungstyps (LAN, RS232, Modem)

Konfiguration: dient zur Konfiguration der DeviLAN-Hardware und des Datenstroms

IO’s: zeigt die aktuellen Wert aller digitalen Ein- und Ausgänge sowie den eingehenden Daten über die seriellen Schnittstellen und den CAN-Bus des Moduls an. Weiterhin können Ausgänge geschaltet oder Meldungen an die verschiedenen Schnittstellen gesendet werden.

COM-PC: gestattet bei der Verbindung über eine serielle Schnittstelle eines PCs, die Schnittstelle zu konfigurieren und den Datenverkehr zu analysieren. Weiterhin kann eine Verbindung über ein Modem aufgebaut werden.

Socket-PC: erlaubt die Verbindung über eine Socket-Schnittstelle (TCP/IP) aufzubauen und den Datenverkehr zu analysieren.

Datei-PC: erlaubt Daten vom den verschiedenen DeviLAN-Schnittstellen in Dateien zu speichern.

Der Zugriff der Dialogboxen *Konfiguration* und *IO's* auf die verschiedenen Schnittstellen des PCs (Socket, RS232, Modem) wird über den *DeviLAN-Manager* gesteuert.

5.2 Die Dialogbox DeviLAN-Manager

Die Dialogbox DeviLAN-Manager beinhaltet die vier Bereiche:

Module im Netz: Baumstruktur zur Anzeige der verfügbaren Module

LAN: Suchen und Verbinden von Modulen im lokalen Netz (LAN)

Verbindung: Auswahl des Verbindungstyps TCP/IP, RS232 und Modem über den ein Modul angesprochen werden soll.

Konfiguration: Abfragen und Speichern der Konfiguration eines DeviLAN-Moduls sowie Einstellen der Netzwerkparameter (IP, Netmask, Gateway)

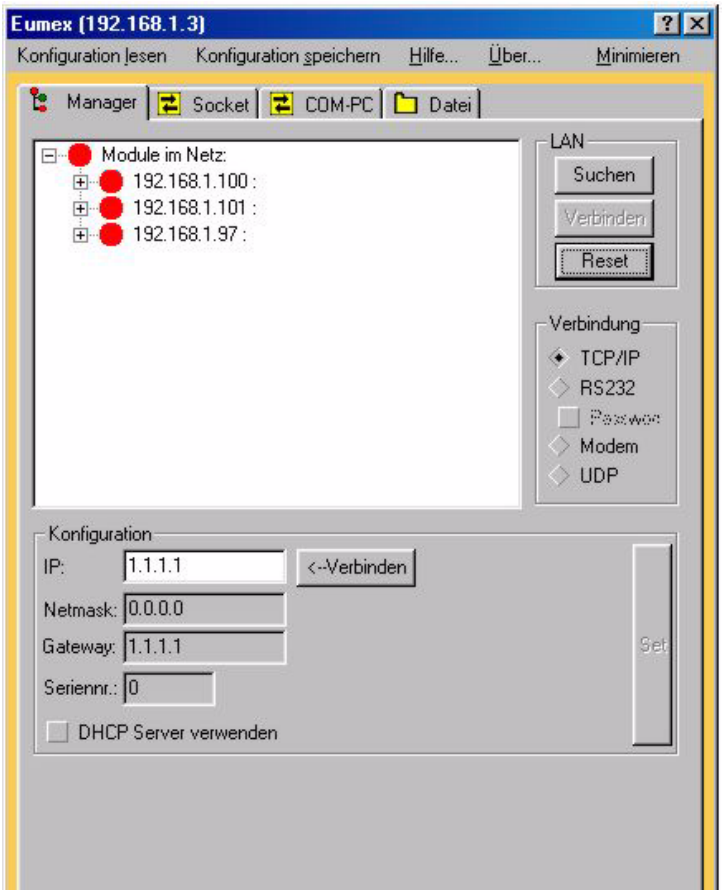


Abb. 2: Dialogbox DeviLAN-Manager

Die Vorgehensweise zum Aufbau einer Verbindung kann den nachfolgenden Kapiteln entnommen werden.

5.2.1 Betrieb über das LAN

Um Module im lokalen Netzwerk zu finden und zu verbinden, ist wie folgt vorzugehen:

1. Auswahl des Verbindungstyps „LAN“.
2. Drücken des Buttons „Suchen“. Es wird eine Broadcast-Meldung abgesetzt auf die alle Module im lokalen Netzwerk antworten. Die gefundenen Module werden mit IP sowie einigen Zusatzinformation in einer Baumstruktur angezeigt.
3. Auswahl einer IP-Adresse in der Baumstruktur durch Anklicken der Adresse mit der Maus.
4. Drücken des Buttons „Verbinden“. Es wird eine Verbindung zu dem Modul aufgebaut, wobei das Modul automatisch eine Modulkennung sendet. Gemäß der Modulkennung werden zusätzliche Dialogboxen (*PPP, Konfiguration, IO's, CAN&COM.....*) angezeigt. Die Bezeichnung des Buttons ändert sich nach erfolgreicher Verbindung in „Trennen“
5. Zum Beenden einer Verbindung den Button „Trennen“ drücken. Die Bezeichnung des Buttons ändert sich in „Verbinden“.

5.2.2 Konfiguration der Netzwerkparameter über UDP

Befindet sich das Modul innerhalb des lokalen Netzwerks, so können die Netzwerkparameter (IP, Netmask, Gateway..) auch über eine UDP-Verbindung eingestellt werden.

Für eine Änderung der Netzwerkeinstellungen ist wie folgt vorzugehen:

1. Unter „Verbindung“ den Eintrag „UDP“ wählen
2. Eintragen der gewünschten Einstellungen in den entsprechenden Eingabefelder
3. Ändern der Einstellungen durch Drücken des Buttons „Set“.

Die Einstellungen werden erst nach einem Neustart des Moduls wirksam (Trennen und erneutes Verbinden mit der Spannungsversorgung). Zur Kontrolle sollten die Netzwerkeinstellungen überprüft werden.

5.2.3 Betrieb über das Internet

Befindet sich das Modul nicht innerhalb des lokalen Netzwerks, so kann eine Internetverbindung wie folgt aufgebaut werden.

1. Eingabe der IP im entsprechenden Eingabefeld.
2. Drücken des Buttons „<-Verbinden“ rechts neben dem Eingabefeld. Es wird eine Verbindung zu dem Modul aufgebaut und die aktuelle Konfigura-

tion und Status angezeigt. Die Bezeichnung des Buttons ändert sich nach erfolgreicher Verbindung in „<-Trennen“.

3. Zum Beenden einer Verbindung den Button „<-Trennen“ drücken. Die Bezeichnung des Buttons ändert sich in „<-Verbinden“.

5.2.4 Betrieb über die serielle Schnittstelle

Nach dem Starten (Power-On) des DeviLAN-Moduls wartet das Modul einige Sekunden lang auf eintreffenden Daten über die serielle Schnittstelle COM1. Wird in dieser Zeit ein Passwort gesendet (z. B. durch DeviLANControl) wird die Schnittstelle COM1 als Kommadoschnittstelle konfiguriert und Kommandos zur Änderungen der IP-Adresse, Netmask und Gateway werden akzeptiert.

Bei Programmstart untersucht das Programm die verfügbaren seriellen Schnittstellen des PCs und stellt die Information dem Bediener in einer Auswahlbox in COM-Port zur Verfügung.

Zur Konfiguration des Moduls ist wie folgt vorzugehen:

1. Anbinden des DeviLAN-Moduls an eine freie serielle Schnittstelle (RS232) des PCs
2. Starten des Programms DeviLANControl

In der Dialogbox „DeviLAN-Manager“ folgende Schritte vornehmen (Abb. 3):

3. Auswählen von Verbinden über ... „RS232“
4. Selektieren von „Passwort“, hierdurch sendet DeviLANControl jede Sekunde das Passwort über die serielle Schnittstelle
5. Starten des DeviLAN-Modul (Einschalten der Spannungsversorgung). Die beiden gelben LEDs des Moduls blinken für einige Sekunden und zeigen an, dass das Modul auf die Passworteingabe wartet.

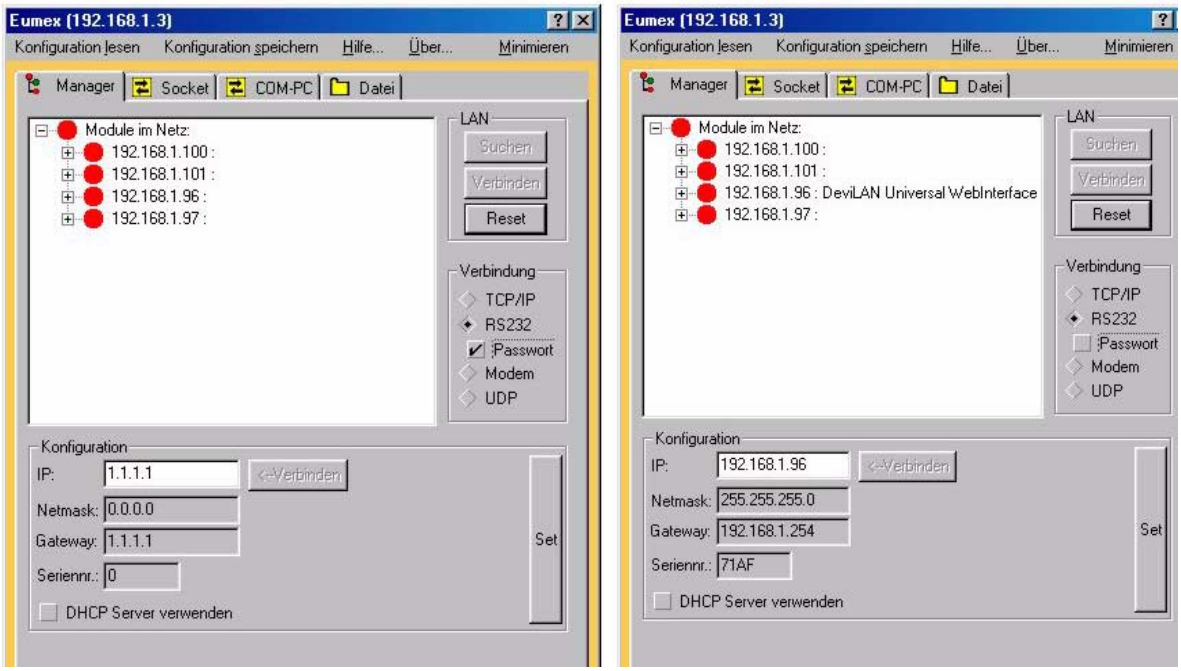


Abb. 3: Konfiguration über serielle Schnittstelle vor (links) und nach (rechts) Verbindungsaufbau

Nach erfolgreichen Verbindungsaufbau sendet das Modul alle Konfigurationseinstellungen an DeviLANControl. Diese werden in den Dialogboxen *IO's*, *Konfiguration, CAN&COM, PPP* und *Manager* angezeigt.

Für einen Änderung der Netzwerkeinstellungen ist wie folgt vorzugehen:

1. Eintragen der gewünschten Einstellungen in den entsprechenden Eingabefeldern
2. Ändern der Einstellungen durch Drücken des Buttons „Set".

Die Einstellungen werden erst nach einem Neustart des Moduls wirksam (Trennen und erneutes Verbinden mit der Spannungsversorgung). Zur Kontrolle sollte der Startvorgang wie oben erklärt noch einmal wiederholt und die Netzwerkeinstellungen überprüft werden.

5.2.5 Betrieb über ein Modem (Telefon)

Der Zugriff auf ein DeviLAN-Modul kann auch über eine analoge Telefonverbindung erfolgen. Um eine Modem-Verbindung zu einen DeviLAN-Modul aufzubauen, ist folgendermaßen vorzugehen:

1. Anbindung eines Modems an die Schnittstelle COM1 des Moduls
2. Konfiguration von COM1 (DeviLAN) als Kommando- und Modemschnittstelle (siehe auch Dialogbox *Konfiguration* Abb. 5)

3. Anbinden des Modems an einen analogen Telefonanschluss
4. Einschalten des DeviLAN-Moduls und des Modems

Zusätzlich muss auch der PC über externes oder internes Modem verfügen, das an eine Telefonleitung angeschlossen ist. Die verfügbaren Modems werden im Dialogelement *COM-PC* angezeigt. Um die Telefonverbindung aufzubauen, sind die folgenden Schritte notwendig.

1. Starten des Programms DeviLANControl
2. Auswahl des Verbindungstyps „Modem“ im Dialogelement *Manager*.
3. Auswahl eines Modems in der Auswahlbox im Dialogelement *COM-PC* (siehe auch Kap. 5.6)
4. Auswahl der Einstellungen für das Modem (Baudrate, Startbits, etc.)
5. Konfigurieren der Schnittstelle über den Button „Set“. Im Kontrollfenster erscheint eine Meldung, ob die Initialisierung erfolgreich war.
6. Eingabe der Telefonnummer im Bereich „Modem“ unter der das Modul erreichbar ist.
7. Starten des Verbindungsaufbaus über den Button „Wählen“. Die Bezeichnung des Buttons ändert sich in „Abbruch“. Der Aufbau erfolgt nun automatisch, der Status des Verbindungsaufbaus wird angezeigt. Sobald die Verbindung steht kann die Bedienung über die Dialogelemente *Konfiguration* und *IO's* erfolgen.
8. Das Beenden der Verbindung erfolgt durch Drücken des Buttons „Abbruch“.

5.2.6 Betrieb über ein Modem (PPP-Verbindung)

Die Vergabe von IP-Adressen innerhalb des Internets ist streng geregelt. Zwar können IP-Adressen innerhalb eines (Firmen-)Netzwerks (LAN) normalerweise frei gewählt und eingestellt werden, jedoch sind die entsprechenden Rechner bzw. DeviLAN-Module nicht über diese Adressen im Internet zugänglich. In vielen Fällen hat der Anwender nicht die Möglichkeit, feste IP-Adressen zu vergeben.

Um ein DeviLAN-Modul im Internet „sichtbar“ zu machen, kann sich das Modul bei einem Internet-Provider einwählen und bekommt eine dynamische IP-Adresse zugewiesen. Über diese IP kann das Modul dann über das Internet (Webbrowser, Telnet, FTP, DeviLANControl, etc.) angesprochen werden. Da sich die IP bei jeden Einwahlvorgang ändert und der Bediener diese nicht kennt, wurde zusätzlich die Möglichkeit vorgese-

hen, dass das Modul die zugewiesene IP-Adresse in einer Email an eine einstellbare Emailadresse weiterleitet.

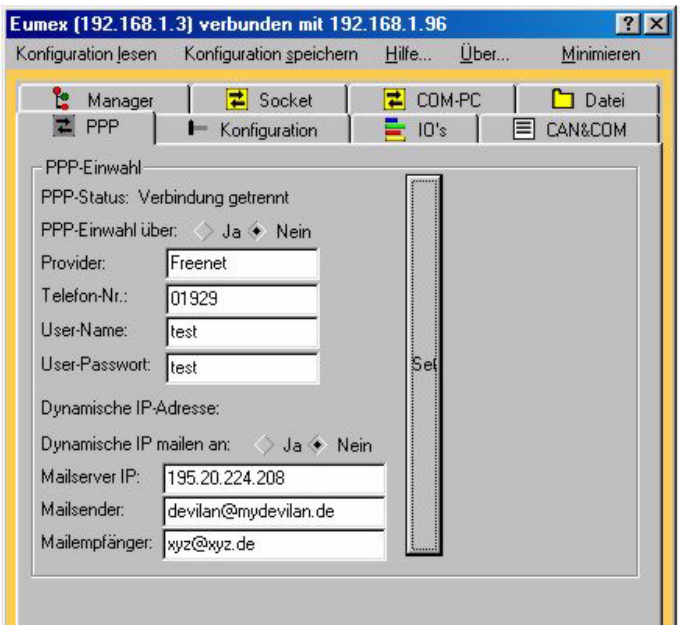


Abb. 4: Dialogbox PPP

Damit ein DeviLAN-Modul eine PPP-Verbindung aufbaut, ist folgendermaßen vorzugehen:

1. Anbindung eines Modems an die Schnittstelle COM1 des Moduls
2. Anbinden des Modems an einen analogen Telefonanschluss
3. Einschalten des DeviLAN-Moduls und des Modems
4. Eintragen der Paramater „Provider“, „Einwahlnummer“, „User-Name“, „User-Passwort“ (siehe auch Tabelle 1) in der Dialogbox *PPP* (Abb. 4)
   Der Eintrag „Provider“ kann beliebig gewählt werden und dient nur der Übersichtlichkeit.
5. „Einwahl über Provider“ auf „Ja“ setzen

Falls gewünscht kann eine Benachrichtung der dynamsich zugewiesenen IP-Adresse per Email erfolgen. Dafür ist wie folgt vorzugehen:

6. „Dynamische IP mailen an“ auf „Ja“ setzen
7. „Mailsender“ kann beliebig gewählt werden und muss der syntaktisch korrekten Form einer Email-Adressse entsprechen, also z. B. meinname@meinerdomain.de.
8. „Mailempfänger“ ist die Mailadresse, an die die dynamische IP des DeviLAN-Moduls übermittelt wird.

9. Damit die Mail verschickt werden kann, muss über den Texteintrag „Mail-server-IP“ die IP eines Mailservers angegeben werden (Tabelle 1). Diese kann frei gewählt werden und muss nicht die Mailserver-IP des Internet-providers sein, über den die Einwahl erfolgt.

10. Durch Drücken des Buttons „Set“-Button wird die Einwahl eingeleitet.

Nach der erfolgten Einwahl wird die zugewiesene dynamische IP angezeigt und die Statusanzeige wechselt auf „Verbindung aufgebaut“. Wurde eine Weiterleitung per Email eingestellt wird eine Mail der Form

Subject: DeviLAN-IP
Content: IP-Adresse ist 213.7.145.54

an den angegebene Mailempfänger verschickt.

Das Beenden der Verbindung erfolgt über „Einwahl über Provider“ auf „Nein“ und Betätigen des „Set“-Buttons.

In Tabelle 1 sind einige der getesteten Einwahlnummern und IP-Adressen von verschiedenen Mailservern angegeben.

Tabelle 1 : Auswahl an Internet-Providern

Provider	Tel.-Nr.	**User-Name**	Passwort	**SMTP**
Acor	0192070	arcor	internet	151.189.8.100 mail.nexgo.de
CompuServe	019160	„Benutzername@compuserve.de“ Anmerkung: Zunächst muss eine Anmeldung beim CompuServe erfolgen	„Benutzer-passwort“	62.52.27.101 smtp.compuserve.de
Freenet	0101901929	beliebig z. B.: test	beliebig z. B.: test	-
MSN	0192658	MSN	MSN	-
Puretec	-	-	-	195.20.224.220 smtp.puretec.de
Planet Interkom	0191799	anonymer	surfer	195.182.114.82 mail.planet-interkom.de

Anmerkungen:

a. Durch die Einwahl bei einem Provider entstehen i.d.R. zusätzliche Kosten.

b. Der Einwahl/Mail-Vorgang kann einige Minuten dauern. Sollte keine Ver-

bindung aufgebaut werden, so ist es möglich sich über Telnet auf das Modul einzuloggen und die Fehlermeldungen bei der Einwahl anzusehen.

c. Während der Einwahl erfolgt die Datenverarbeitung der übrigen Schnittstellen nur eingeschränkt.

5.3 Die Dialogbox Konfiguration

Die Einstellmöglichkeiten für die verschiedenen Schnittstellen sind in Gruppen zusammengefasst. Die Einstellungen werden ebenfalls in Gruppen durch Betätigen des zugehörigen „Set“-Buttons an das Moduls übertragen und übernommen.

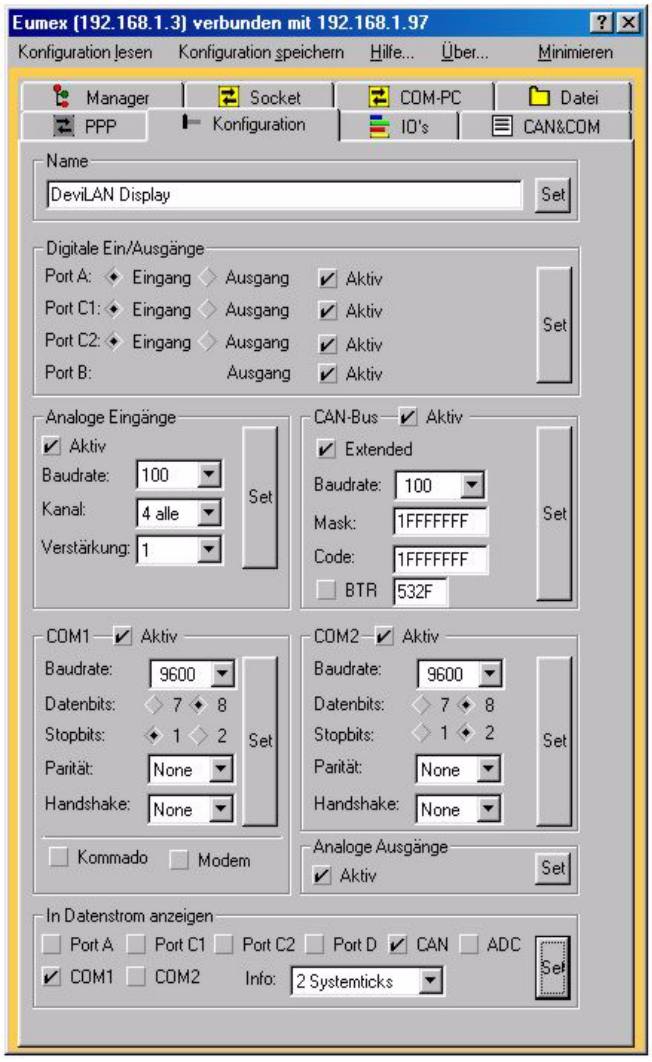


Abb. 5: Dialogbox Konfiguration

Die Parameter entsprechen den üblichen Bezeichnungen der Schnittstellen und werden an dieser Stelle nicht weiter erläutert.

Um die Netzwerk und Rechenbelastung des Moduls gering zu halten, kann eine Auswahl der anzuzeigenden Daten über den Bereich „In Datenstrom anzeigen“ erfolgen. Grundsätzlich sollten nur Schnittstellen aktiviert und deren Daten angezeigt werden, die benötigt werden.

Anmerkung:

Die auftretende Datenmenge kann die maximal übertragbare Datenmenge über eine Socket-, RS232- oder Modemverbindung überschreiten. Dies stellt keinen Fehler der DeviLAN-Software/Hardware dar, sondern ergibt sich aus den Spezifikation der unterschiedlichen Schnittstellen.

5.4 Die Dialogbox IO’s

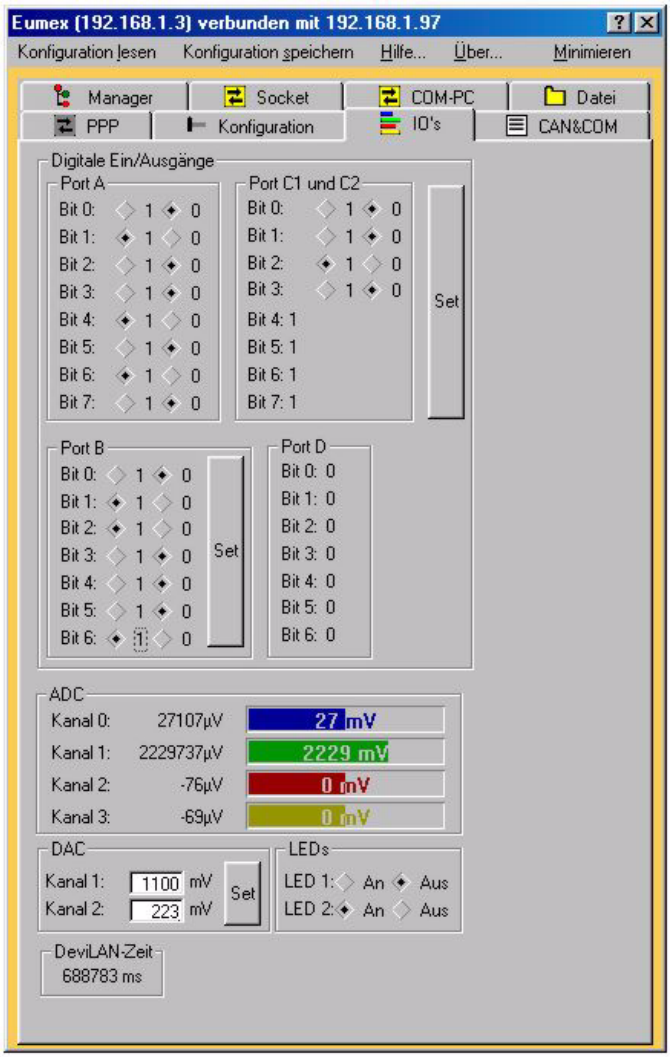


Abb. 6: Dialogbox IO’s

Die Dialogbox IO’s (Abb. 6) dient der Anzeige und dem Zugriff auf alle digitalen und analogen Ports. Die Anzeige für die verschiedenen Schnittstellen sind in Gruppen zusammengefasst. Die Werte werden ebenfalls in Gruppen durch Betätigen des zugehörigen „Set“ bzw. „Send“-Buttons an das Moduls übertragen und übernommen.

5.5 Die Dialogbox CAN&COM

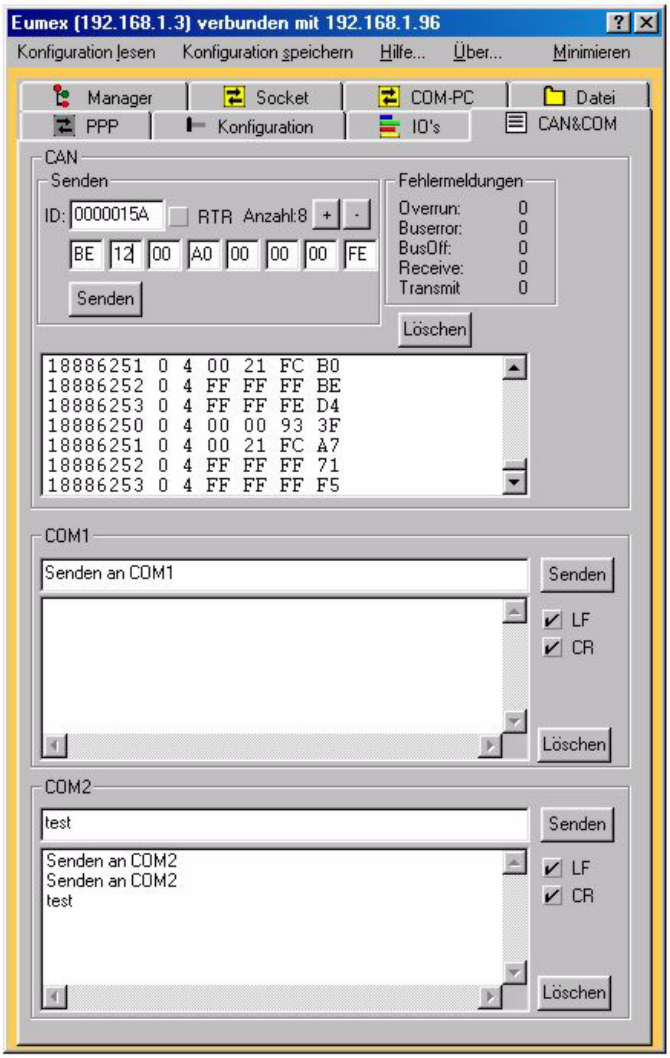


Abb. 7: Dialogbox CAN&COM

Die Dialogbox CAN&COM dient der Anzeige und dem Zugriff auf den CAN-Bus und die seriellen Schnittstellen. Die Anzeige für die verschiedenen Schnittstellen sind in Gruppen zusammengefasst.

CAN-Telegramme und Texte, die an die seriellen Schnittstellen übertragen werden sollen, können über die zugehörigen Eingabefelder editiert und über „Senden“ an das Modul übertragen werden.

Die Fensterelemente dienen zur Anzeige der eingehenden Daten des CAN-Bus sowie der Schnittstellen COM1 und COM2.

5.6 Die Dialogbox COM-PC

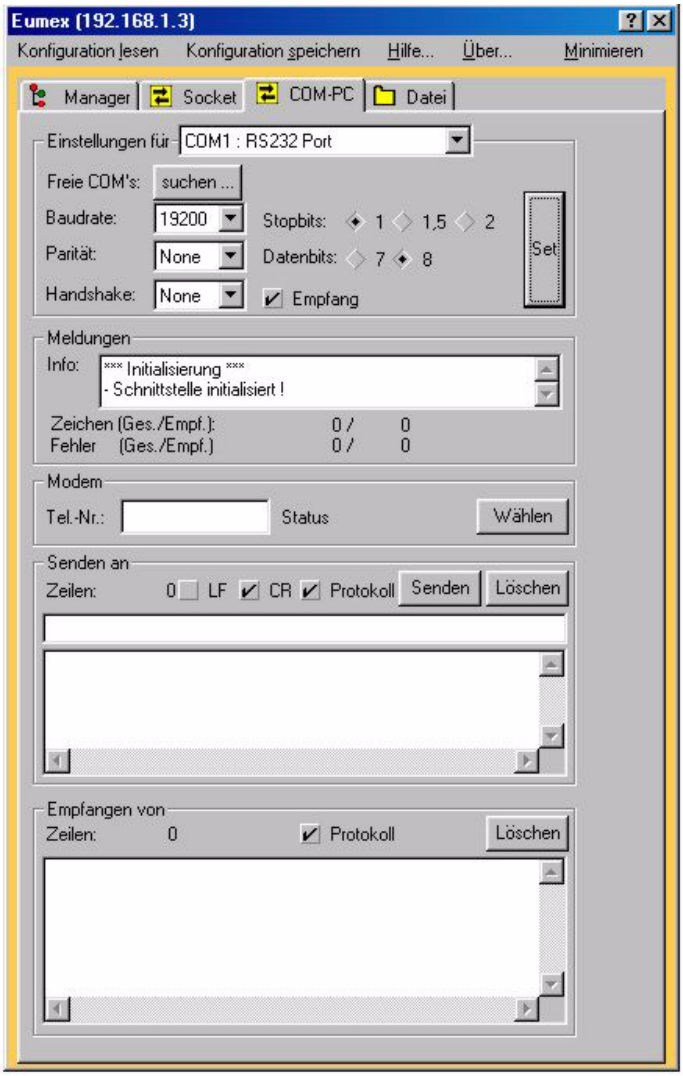


Abb. 8: Dialogbox COM-PC

Über die Dialogbox *COM-PC* (Abb. 8) kann der Datenverkehr zwischen dem PC und dem DeviLAN-Modul bei Verbindung über eine serielle Schnittstelle (siehe Kap. 5.2.4) überwacht werden. Zusätzlich besteht die Möglichkeit eine Modem-Verbindung (siehe Kap. 5.2.5) aufzubauen.

Für den Betrieb über die serielle Schnittstelle ist es dabei nicht notwendig, dass der Bediener Einstellungen in dieser Dialogbox vornimmt, da dies automatisch über den *DeviLAN-Manager* erfolgt.

Die verfügbaren Schnittstellen werden beim Start der Applikation DeviLANControl bzw. über den Button „suchen...“ermittelt und können über die Liste ausgewählt werden. Nachdem die Einstellungen (Baudrate, Parität, etc.) gewählt wurden, kann die Schnittstelle über den Button „Set“ initialisiert werden. Zustands- und Fehlermeldungen werden im Bereich „Meldungen“ ausgegegen.

Zum Absenden eines Kommandos, ist dieses zunächst in die Eingabezeile einzutragen. Das Absenden erfolgt dann entweder durch betätigen der „Enter“-Taste oder durch Drücken des Buttons „Senden“. Die gesendeten und empfangenen Zeilen erscheinen in den beiden entsprechenden Fenster. Dabei werden max. die letzten 200 Zeilen zwischengespeichert.

5.7 Die Dialogbox Socket

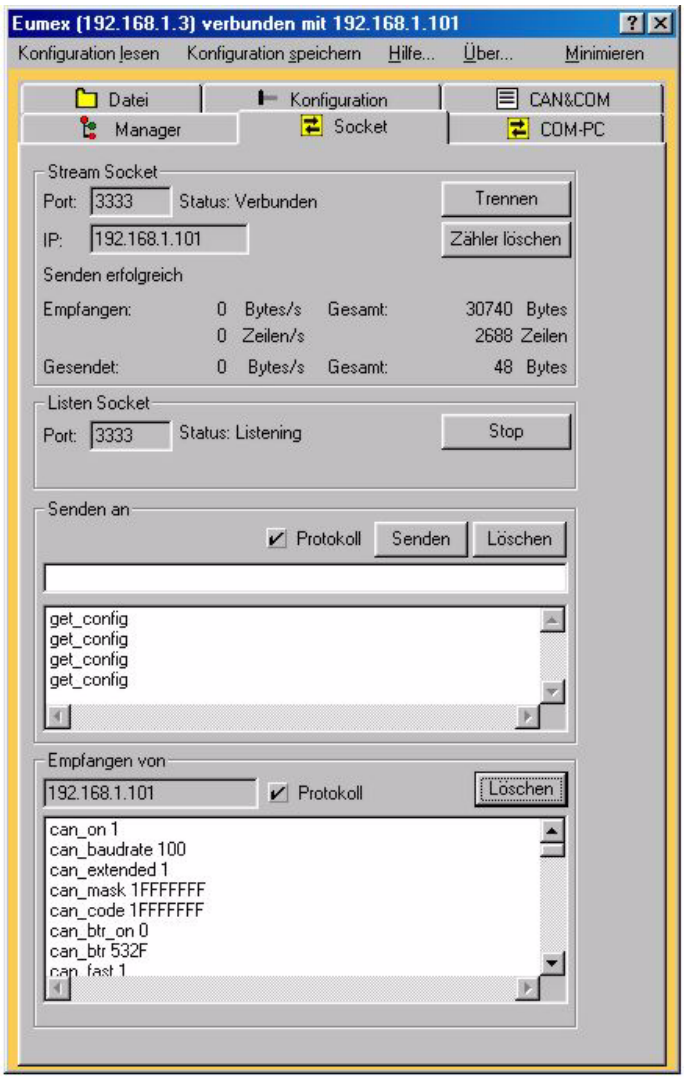


Abb. 9: Dialogbox Socket

Die Dialogbox *Socket* (Abb. 9) dient zum Aufbau einer Netzwerk/ Internetverbindung (TCP/IP) zu einem DeviLAN-Modul.

Weiterhin kann der Datenverkehr über zwischen PC-Programm und Modul überwacht werden. Für den normalen Betrieb ist es dabei nicht notwendig, dass der Bediener Einstellungen in dieser Dialogbox vornimmt, da dies automatisch über den *DeviLAN-Manager* erfolgt.

Um eine Verbindung manuell aufzubauen muss zunächst die IP und die Port-Nummer in den entsprechenden Eingabefeldern eingetragen und anschließend der Button „Verbinden“ gedrückt werden.

Alternativ ist es auch möglich, dass sich andere Prozesse auf die Dialogbox (Port 4444) aufschalten.

Zum Absenden eines Kommandos ist dieses zunächst in die Eingabezeile einzutragen. Das Absenden erfolgt dann entweder durch betätigen der „Enter“-Taste oder durch Drücken des Buttons „Senden“. Die gesendeten und empfangenen Zeilen erscheinen in den beiden entsprechenden Fenster. Dabei werden max. die letzten 200 Zeilen zwischengespeichert.

5.8 Die Dialogbox Datei-PC

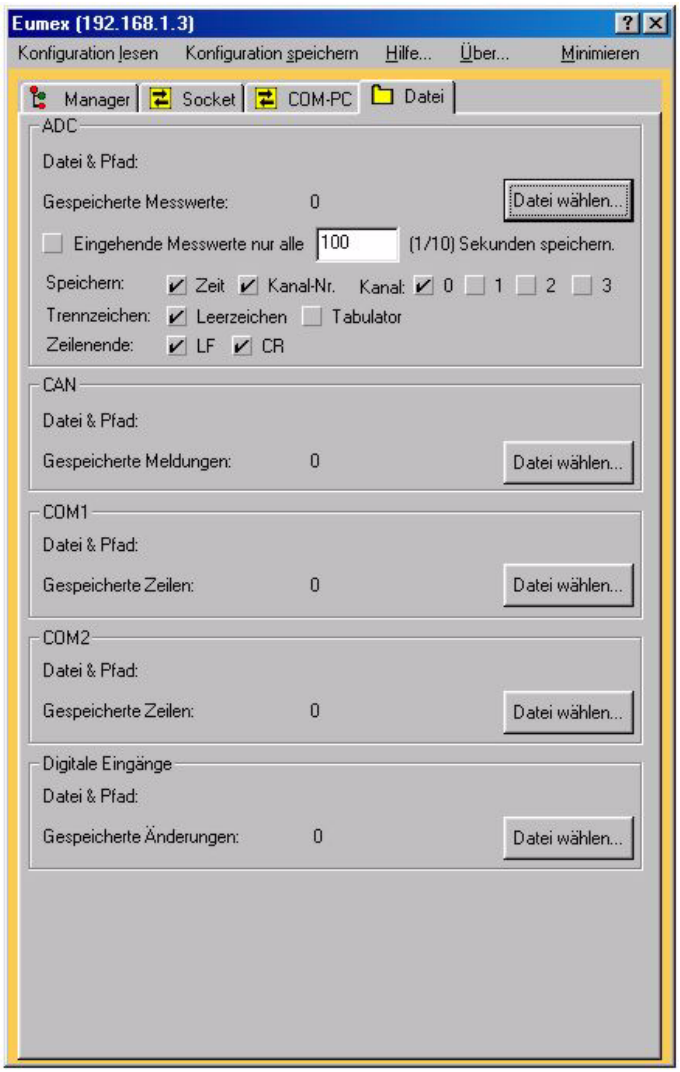


Abb. 10: Dialogbox Datei-PC

Die Dialogbox *Datei-PC* (Abb. 10) erlaubt Daten vom den verschiedenen DeviLAN-Schnittstellen in Dateien zu speichern.

Über die Button „Datei wählen“ können verschiedene Dateien für die Schnittstellen ADC, CAN, COM1, COM2 sowie die digitalen Eingänge ausgewählt werden. Wurde eine Datei ausgewählt, wird der Dateipfad und -name angezeigt und die zugehörige Button-Beschriftung ändert sich in „Schließen“. Durch die erneute Betätigung des Buttons kann der Speichervorgang beendet werden.

Die Anzahl der gespeicherten Messwerte, Meldungen bzw. Zeilen wird über einen Zähler angezeigt.

Die Daten werden als ASCII-Text in die Datei geschrieben und können mit jedem Textverarbeitungsprogramm oder auch z. B. mit MS Excel angezeigt/ausgewertet werden.

6 Kommandointerface

6.1 Hardwareübersicht

Das Kommandointerface gestattet einen vollständigen Zugriff auf alle Konfigurationseinstellungen und erlaubt, die Schnittstellen abzufragen. Die Hardware besteht aus den Funktionsgruppen, die in Abb. 11 wiedergegeben werden.

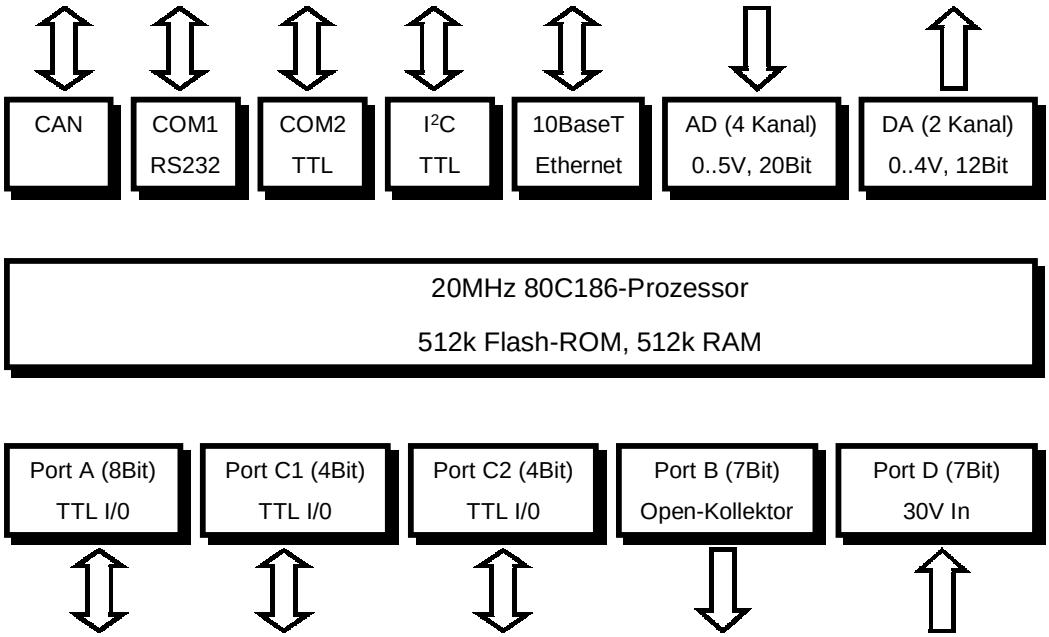


Abb. 11: Hardwareübersicht

Im folgenden wird nur dann auf den detaillierteren Aufbau der Hardware eingegangen, wenn sie DeviLAN-typisch sind. Alle übrigen Spezifikation der Schnittstellen entsprechen gängigen Standards und üblichen Bezeichnungen und können entsprechender Fachliteratur entnommen werden.

6.2 Kommunikationsschnittstellen

Das DeviLAN-Modul verfügt über zwei verschiedene Kommunikationsschnittstellen mit denen das Modul konfiguriert und abgefragt werden kann. Diese sind

- ein kombiniertes Webbrowser/CGI-Interface, das einen Zugriff über einen Webbrowser gestattet sowie
- ein Kommando/Datenstrom-Interface, das die Anbindung von Applikationen über ein Netzwerk, eine serielle Schnittstelle oder ein Modem gestattet.

Die Möglichkeiten der Anbindung sind in Abb. 12 wiedergegeben.

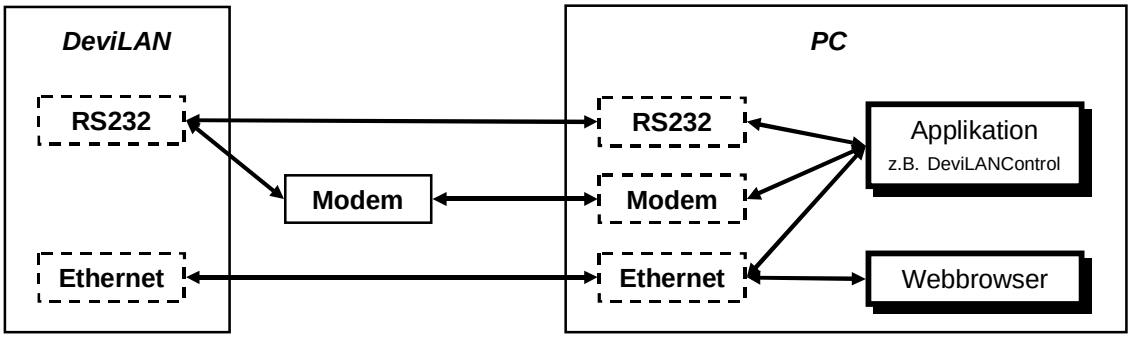


Abb. 12: DeviLAN Kommunikationsschnittstellen

Die Funktionsweise lässt sich vereinfacht mittels Abb. 13 darstellen. Bei einem Zugriff über einen Webbrowser werden die in den HTML-Seiten übertragenen Daten und Parameter vom CGI-Interface extrahiert und dem Kommandointerpreter zugeführt. Dieser führt die Kommandos aus und steuert gleichzeitig Zugriff auf die Hardware. Die über die Hardware eingehenden Daten werden der Datenaufbereitung zugeführt und stehen dann dem HTML-Seitengenerator und dem Datenstrom-Interface zur Verfügung.

Für die Anbindung eigener Applikationen steht dem Entwickler das Datenstrominterface zur Verfügung. Es gestattet, wie bei einen Zugriff über den Webbrowser, die vollständige Konfiguration und Abfragen aller Hardwarekomponenten.

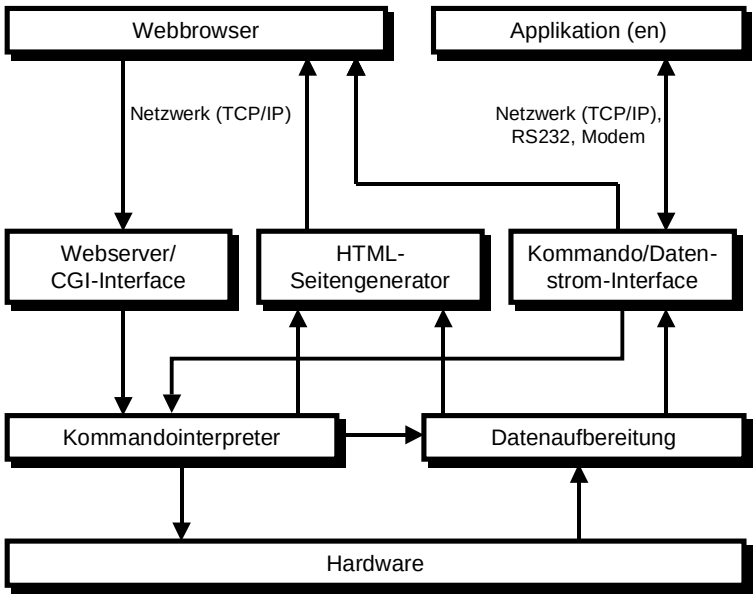


Abb. 13: Kommunikationszugriff (vereinfacht)

6.3 Allgemeine Informationen

Ein Kommandozeile startet mit einen Parameter (Kommandoname) gefolgt von einen ggf. auch mehreren Parameterwerten die jeweils durch ein Leerzeichen getrennt sind. In Einzelfällen kann der Parameterwert auch entfallen.

Syntaxangabe 1 :Aufbau einer Kommandozeile

```
<Parametername><SP>[<Parameterwert>]....[<SP>][<Parameterwert>]<CR>
```

Variablennamen und Steuerzeichen stehen werden in ’<’ und ’>’ angegeben, optionale Angaben stehen in eckigen Klammer ’[’ ’]’. Die verwendeten Datentypen können Tabelle 2, die Steuerzeichen Tabelle 3 entnommen werden.

Tabelle 2 : Datentypen

Zeichenfolge	Erklärung
INT	16 Bit Integerzahl
UINT	16 Bit vorzeichenlose Integerzahl
LONG INT	32 Bit Integerzahl
LONG UINT	32 Bit vorzeichenlose Integerzahl
CHAR	einzelnes Zeichen
STRING [x]	Zeichenkette mit max. x Zeichen
XX	Zweistellige hexadezimale Angabe (Byte)
hh:mm:ss	Zeitangabe in Stunden, Minuten, Sekunden seit Systemstart
IP	IP-Angabe in der Form aaa.bbb.ccc.ddd
MAIL [x]	gültige Mailadresse mit max. x Zeichen z. B. der Form name@domainname.de

Tabelle 3 : Steuerzeichen

Zeichenfolge	Hex. Wert	Beschreibung
<LF>	0x0A	Line Feed, Zeilenvorschub
<CR>	0x0D	Carriage Return, Zurück zum Zeilenanfang
<SP>	0x20	Space, Leerzeichen

Der Kommandointerpreter arbeitet **zeilenorientiert** im Gegensatz zu Terminalprogrammen, die zeichenorientiert arbeiten, also nach Erhalt jedes Zeichen auswerten.

Der Interpreter fügt alle eintreffenden Zeichen bis zum <CR> in seinen Empfangspuffer ein und wertet diese erst dann aus. Ein Kommando sollte daher keine Steuerzeichen wie ’Backspace’ oder ’Delete’ etc. enthalten, da diese keinen steuernden Charakter haben, sondern wie jedes andere Zeichen auch zunächst im Puffer abgelegt werden.

Für die Anbindung von Applikationen sollte dies kein Problem darstellen, da hier ein Vertippen, wie bei manueller Eingabe, durch ein Absenden einer vollständigen Kommandozeile vermieden werden kann.

Nach jeder gesendeten Kommandozeile wertet das DeviLAN-Modul diese aus und führt die entsprechende Anweisung aus. Bei Konfigurationseinstellungen wird der geänderte Wert als Quittung zurückgegeben. Bei Datenanfragen werden die angeforderten Daten zurückgegen.

Beispiel 1 : Port als Ausgang konfigurieren

```
porta 1; An DeviLAN: Port A als Ausgang konfigurieren
porta 1; Von DeviLAN:Port A wurde als Ausgang konfiguriert
```

Erkennt der Kommandointerpreter einen Fehler wird eine Fehlermeldung zurückgegeben.

Syntaxangabe 2 :Fehlermeldung des Kommandointerpreters

```
Error:<SP><Fehlermeldung im Klartext><LF><CR>
```

Beispiel 2 : Fehlermeldungen des Kommandointerpreters

```
Error: Unkown command ; (Parametername/Kommando unbekannt)
Error: Cannot convert ... ; (Parameterwert nicht bestimmbar, ausserhalb
des gültigen Bereichs oder im unerwarteten Format)
```

6.4 Konfiguration der Hardware/Protokolle

6.4.1 Digitale Ein- und Ausgänge

DeviLAN-Modul verfügt über 16 I/Os, die wahlweise als TTL-Eingänge oder TTL-Ausgänge geschaltet werden können. Die Konfiguration erfolgt in Gruppen zu 8 Bits (Port A), 4 Bits (Port C1) und 4 Bits (Port C2) Bit. Die Konfiguration erfolgt über die Befehle in Tabelle 4.

Tabelle 4 : Konfiguration der digitalen Ein- und Ausgänge (Port A und C)

Befehl	Wert/Typ	Default	Funktionsbeschreibung
porta	0,1	0	0=Port A Eingang, 1=Port A Ausgang (8 Bit)
portc1	0,1	0	0=Port C1 Eingang, 1=Port C2 Ausgang (4 Bit)
portc2	0,1	0	0=Port C2 Eingang, 1=Port C2 Ausgang (4 Bit)
porta_on	0,1	0	Aktiviert/Deaktiviert den Zugriff Port A; nur relevant wenn Port A als Ausgang konfiguriert ist 0=kein Zugriff auf Port A, 1=Zugriff auf Port A

Befehl	Wert/Typ	Default	Funktionsbeschreibung
portc1_on	0,1	0	Aktiviert/Deaktiviert den Zugriff Port C1; nur relevant wenn Port C1 als Ausgang konfiguriert ist 0=kein Zugriff auf Port C1, 1=Zugriff auf Port C1
portc2_on	0,1	0	Aktiviert/Deaktiviert den Zugriff Port C2; nur relevant wenn Port C2 als Ausgang konfiguriert ist 0=kein Zugriff auf Port C2, 1=Zugriff auf Port C2
portd_on	0,1	0	Aktiviert/Deaktiviert den Zugriff Port D; nur relevant wenn Port D als Ausgang konfiguriert ist 0=kein Zugriff auf Port D, 1=Zugriff auf Port D

Das Schalten der als Ausgänge konfigurierten Bits erfolgt über die Befehle in Tabelle 5 und Tabelle 6. Sind die entsprechenden Bits als Eingänge konfiguriert, werden die Befehle zum Schalten der zugehörigen Ausgänge ignoriert.

Tabelle 5 : Schalten von Port A

Befehl	Wert/Typ	Default	Funktionsbeschreibung
a0 a1 ..... a7	0,1	0	Bitweiser Zugriff auf Port A 0=Bit wird gelöscht 1=Bit wird gesetzt
a0to7	0x00..0xFF	0x00	Byteweiser Zugriff auf Port A, hexadezimale Angabe Beispiel: a0to7 A9

Tabelle 6 : Schalten der Ausgänge von Port C (C1 und C2)

Befehl	Wert/Typ	Default	Funktionsbeschreibung
c0 c1 c2 c3	0,1	0	Bitweiser Zugriff auf Port C1 0=Bit wird gelöscht 1=Bit wird gesetzt
c4 c5 c6 c7	0,1	0	Bitweiser Zugriff auf Port C2 0=Bit wird gelöscht 1=Bit wird gesetzt
c0to7	0x00..0xFF	0x00	Byteweiser Zugriff auf Port C, hexadezimale Angabe Beispiel: c0to7 7F Anmerkung: Ist Port C1 bzw. C2 als Eingang konfiguriert, werden die entsprechenden Bits ignoriert.

Als weiterer Ausgangsport steht Port B (Open-Kollektor) mit 7 Bits zur Verfügung. Der Zugriff erfolgt über die Befehle in Tabelle 7.

Tabelle 7 : Schalten der Ausgänge von Port B

Befehl	Wert/Typ	Default	Funktionsbeschreibung
b0 a1 ..... b6	0,1	0	Bitweiser Zugriff auf Port B 0=Bit wird gelöscht 1=Bit wird gesetzt
b0to6	0x00..0x7F	0x00	Byteweiser Zugriff auf Port B (7-Bit), hexadezimale Angabe Beispiel: b0to6 7A

6.4.2 Digitale Eingänge

Die digitalen TTL-Eingänge (Port A und C) sowie die 24V-Eingänge (Port D) werden zur Zeit nur über den Datenstrom-Interface (siehe auch Kap. 6.4.8 und Kap. 6.4.11) abgefragt. Ändert sich der Zustand eines Bits wird die Information an die angebundene Applikation übertragen.

Anmerkung: Das Abfragen der Ports erfolgt zyklisch. Schnelle Signaländerungen werden daher ggf. nicht gefasst.

6.4.3 Serielle Schnittstelle COM1

Die serielle Schnittstelle COM1 wird über die Befehle in Tabelle 8 konfiguriert.

Tabelle 8 : Konfiguration der seriellen Schnittstelle COM1

Befehl	Wert/Typ	Default	Funktionsbeschreibung
com1_on	0,1	1	Aktiviert/Deaktiviert den Zugriff auf die seriellen Schnittstelle COM1. 0=COM1 deaktiv, 1=COM1 aktiv
com1_baudrate	300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 56000	19200	Stellt die Baudrate für COM1 ein. Der Zahlenwert versteht sich als Angabe in kBit/s.
com1_stopbits	1, 2	1	Setzt die Anzahl an Stopbits.
com1_databits	7, 8	8	Setzt die Anzahl an Datenbits.

Befehl	Wert/Typ	Default	Funktionsbeschreibung
com1_parity	Even, Odd, Mark, None, Space	None	Stellt die Parität ein. Anmerkung: Es ist nur der erste Buchstabe des Wertes relevant, d. h. das Kommando com1_parity Even ist gleichbedeutend wie com1_parity E
com1_flowctrl	None, RTS/CTS, XON/XOFF	None	Stellt den Hardware-Handshake/ die Datenflus-skontrolle ein. Anmerkung: Es ist nur der erste Buchstabe des Wertes relevant, d. h. das Kommando com1_flowctrl RTS/CTS ist gleichbedeutend wie com1_flowctrl R
com1_command	0,1	1	0=Daten von COM1 werden weitergeleitet 1=die serielle Schnittstelle COM1 wird als Kom-mandoschnittstelle verwendet, d.h. eingehende Daten werden nicht weitergeleitet sondern als Kommandos interpretiert. Ein DeviLAN-Modul kann dadurch z. B. über eine RS232-Schnittstelle eines PCs angesteuert werden.
com1_modem	0,1	0	0=kein Modembetrieb 1=Modembetrieb Durch die Verbindung von COM1 mit einem exter-nen Modem kann ein DeviLAN-Modul auch über eine Telefonleitung angesteuert werden. Zusätz-lich muss dazu die Kommandofunktion von COM1 aktiviert sein.

Um Daten über die Schnittstelle zu senden, können die Befehle in Tabelle 9 verwendet werden. Die Kommunikation ist zeilenorientiert ausgelegt.

Tabelle 9 : Sendebefehle und Optionen der seriellen Schnittstelle COM1

Befehl	Wert/Typ	Default	Funktionsbeschreibung
com1_message	STRING[50]	--	Sendet eine Zeichenfolge über COM1. **Anmerkung:** Die Zeichenkette darf keine Steuerzeichen wie CR, LF, TAB, etc. enthalten.

Befehl	Wert/Typ	Default	Funktionsbeschreibung
com1_lf	0,1	1	Sendeoption: 0=Zeichenkette wird unverändert gesendet 1=Beim Senden wird an die Zeichenfolge ein Line Feed (<LF>, Hexadezimal: 0x0A) Zeilenvorschub angehängt.
com1_cr	0,1	1	Sendeoption: 0=Zeichenkette wird unverändert gesendet 1=Beim Senden wird an die Zeichenfolge ein Carriage Return (<CR>, Hexadezimal: 0x0D) angehängt.

Beispiel 3 : Senden von Daten über COM1

```
com1_lf 1 ; LF anhängen
com1_cr 1 ; CR anhängen
com1_message Diese Zeile wird gesendet
```

Über COM1 wird: „Diese Zeile wird gesendet <LF><CR>“ ausgegeben.

6.4.4 Serielle Schnittstelle COM2

Die serielle Schnittstelle COM2 wird über die Befehle in Tabelle 10 konfiguriert.

Tabelle 10 :Konfiguration der seriellen Schnittstelle COM2

Befehl	Werte	Default	Funktionsbeschreibung
com2_on	0,1	1	Aktiviert/Deaktiviert den Zugriff auf die seriellen Schnittstelle COM2. 0=COM2 deaktiv, 1=COM2 aktiv
com2_baudrate	300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 38400, 56000	19200	Stellt die Baudrate für COM2 ein. Der Zahlenwert versteht sich als Angabe in kBit/s.
com2_stopbits	1, 2	1	Setzt die Anzahl an Stopbits.
com2_databits	7, 8	8	Setzt die Anzahl an Datenbits.

Befehl	Werte	Default	Funktionsbeschreibung
com2_parity	Even, Odd, Mark, None, Space	None	Stellt die Parität ein. Anmerkung: Es ist nur der erste Buchstabe des Wertes relevant, d. h. das Kommando com3_parity Even ist gleichbedeutend wie com3_parity E
com2_flowctrl	None, RTS/CTS, XON/XOFF	None	Stellt den Hardware-Handshake/ die Datenflusskontrolle ein. Anmerkung: Es ist nur der erste Buchstabe des Wertes relevant, d. h. das Kommando com3_flowctrl RTS/CTS ist gleichbedeutend wie com3_flowctrl R

Um Daten über die Schnittstelle zu senden, können die Befehle in Tabelle 11 verwendet werden. Die Kommunikation ist zeilenorientiert ausgelegt.

Tabelle 11 :Sendebefehle und Optionen der seriellen Schnittstelle COM2

Befehl	Werte	Default	Funktionsbeschreibung
com2_message	Zeichenfolge, max. 50 Zeichen		Sendet eine Zeichenfolge über COM2. **Anmerkung:** Die Zeichenkette darf keine Steuerzeichen wie CR, LF, TAB, etc. enthalten.
com2_lf	0,1	1	Sendeoption: 0=Zeichenkette wird unverändert gesendet 1=Beim Senden wird an die Zeichenfolge ein Line Feed (<LF>, Hexadezimal: 0x0A) Zeilenvorschub angehängt.
com2_cr	0,1	1	Sendeoption: 0=Zeichenkette wird unverändert gesendet 1=Beim Senden wird an die Zeichenfolge ein Carriage Return (<CR>, Hexadezimal: 0x0D) angehängt.

6.4.5 CAN-Bus

Die Konfiguration des CAN-Bus erfolgt über die Befehle in Tabelle 12.

Tabelle 12 :Konfiguration des CAN-Bus

Befehl	Werte	Default	Funktionsbeschreibung
can_on	0,1	0	Aktiviert/Deaktiviert den Zugriff auf den CAN-Bus. 0=CAN-Bus deaktiv, 1=CAN-Bus aktiv
can_baudrate	5, 10, 20, 50, 100, 125, 250, 500, 1000	20	Stellt die Baudrate für den CAN-Bus ein. Der Zahlenwert versteht sich als Angabe in kBit/s.
can_mask	0x000... 0x7FF für (CAN2.0A) 0x00000000... 0x1FFFFFFF für (CAN2.0B)	0x7FF für (CAN2.0A) bzw. 0x1FFFFFFF für (CAN2.0B)	Beschreibt das Acceptance-Mask-Register des CAN-Controllers.
can_code	0x000... 0x7FF für (CAN2.0A) 0x00000000... 0x1FFFFFFF für (CAN2.0B)	0x7FF für (CAN2.0A) bzw. 0x1FFFFFFF für (CAN2.0B)	Beschreibt das Acceptance-Code-Register des CAN-Controllers.
can_btr_on	0,1	0	Aktiviert/Deaktiviert den Zugriff auf das BTR-Register des CAN-Controllers SJA1000 0=kein direkter Zugriff auf das BTR-Register; der über *can_btr* eingestellte wird ignoriert 1=Zugriff auf das BTR-Register; der über *can_baudrate* eingestellte Wert wird ignoriert, statt dessen wird die Baudrate direkt über *can_btr*
can_btr	0x0000... 0xFFFF	0x532F (20kBit/s)	Beschreibt das BTR-Register des CAN-Controllers SJA1000, wenn *can_btr_on*=1. Damit ist es möglich Baudraten einzustellen, die nicht den Standardvorgaben entsprechen.

Ein CAN-Datensatz setzt sich aus dem Identifier (ID), dem RTR-Bit, der Anzahl der Datenbytes sowie 0 bis 8 Datenbytes zusammen. Um ein CAN-Telegramm zusammen zusetzen und zu senden, stehen die Befehle in Tabelle 13 bereit.

Tabelle 13 :Senden über CAN-Bus

Befehl	Werte	Default	Funktionsbeschreibung
can_extended	0,1	0	Legt länge des Identifiers fest: 0: 11-Bit Identifier 1: 29-Bit Identifier
can_id	0x000... 0x7FF für (CAN2.0A) 0x00000000... 0x1FFFFFFF für (CAN2.0B)	0x00	Setzt den CAN-Identifier und legt die über die Befehle can_rtr, can_len und can_b0...can_b7 zusammengesetzte Meldung auf den Bus.
can_rtr	0,1	0	Ändert das RTR-Bit: 1=setzt das RTR-Bit 0=löscht das RTR-Bit
can_len	0,1,...,8	0	Setzt die Anzahl der Datenbytes
can_b0 can_b1 ..... can_b7	0x00..0xFF	0x00	Setzt die Datenbytes.
can_msg <id> <rtr> <len> <b0> <b1> <b2> <b3> <b4> <b5> <b6> <b7>	id=0x000... 0x7FF für (CAN2.0A) 0x00000000... 0x1FFFFFFF für (CAN2.0B) rtr= 0,1 len= 0..8 b0..b7= 0x00..0xFF	-	Sendet die eine komplette CAN-Meldung bestehend aus den übergebenen Werten (can_id, can_rtr, can_len, can_bo....can_b7) auf den CAN-Bus. Anmerkung: Die einzelnen Parameter müssen durch genau ein Leerzeichen <SP> voneinander getrennt werden.

Die Werte werden einzeln gesetzt und über den *can_id*-Befehl über den CAN-Bus gesendet. Die Werte bleiben nach dem Senden gespeichert und können ohne erneute Übertragung wieder gesendet werden. Ebenso können so einzelne Werte des CAN-Telegramms verändert werden. Der Befehl *can_msg* gestattet eine vollständige CAN-Message zu übertragen und zu senden.

6.4.6 Analog-Digital-Wandler (ADC)

Das AD-Wandlersystem verfügt über 4 getrennte, differentiell ausgelegt Kanäle. Die Kanalauswahl wird über einen Multiplexer gesteuert. Der gewählte Kanal wird zusätzlich vorverstärkt, um besonders bei kleinen Signalen eine bessere Auflösung zu erhalten. Die Verstärkung (V=1,2,4,8,16) kann fest vorgegeben werden. Der AD-Wandler arbeitet mit einer maximalen Abtastrate von 1kHz. Bei einer Kanalumschaltung treten bedingte Verzögerungen durch Umschalt- und Ausgleichvorgänge auf, sodass die Abtastrate bei kontinuierlicher Wandlung aller Kanäle auf etwa 1/4 der eingestellten Frequenz sinkt.

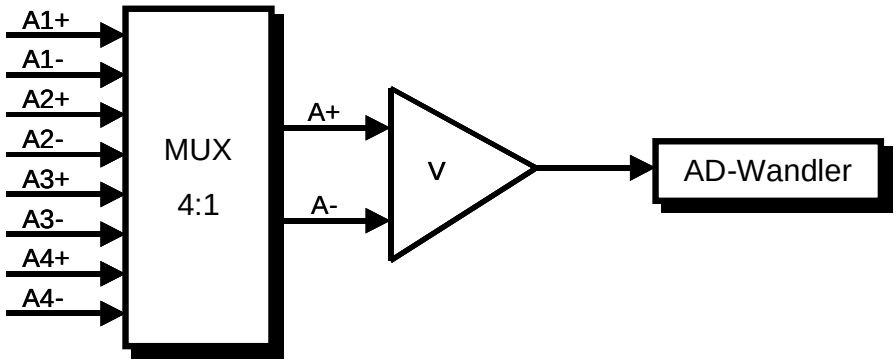


Abb. 14: Analog-Digital-Wandler mit Multiplexer und Verstärker

Tabelle 14 :Konfiguration des Analog-Digital-Wandlers

Befehl	Werte	Default	Funktionsbeschreibung
adc_on	0,1	0	Aktiviert/Deaktiviert den Zugriff auf den Analog-Digital-Wandler. 0=ADC deaktiv, 1=ADC aktiv
adc_baudrate	3...1000		Stellt die Baudrate für den ADC ein. Der Zahlenwert versteht sich als Angabe in Hz.
adc_gain	1,2,4,8,16		Setzt die Vorverstärkung der Analogsignale
adc_channel	0,1,2,3,4		Wählt den Kanal 0,1,2 oder 3 aus; 4 bedeutet die Kanäle 0..3 werden kontinuierlich hintereinander gewandelt.

6.4.7 Digital-Analog-Wandler (DAC)

Über die Befehle in Tabelle 15 und Tabelle 16 erfolgt der Zugriff auf den Digital-Analog-Wandler.

Tabelle 15 :Konfiguration des Digital-Analog-Wandlers

Befehl	Werte	Default	Funktionsbeschreibung
dac_on	0,1	0	Aktiviert/Deaktiviert den Zugriff auf den DA-Wandler. 0=DAC deaktiv, 1=DAC aktiv

Tabelle 16 :Einstellung der Digital-Analog-Wandlerwerte

Befehl	Werte	Default	Funktionsbeschreibung
dac_1	0...4095	0	Stellt den Analogwert für Kanal 1 des DA-Wandlers ein. Der Wert versteht sich als Angabe in mV.
dac_2	0...4095	0	Stellt den Analogwert für Kanal 2 des DA-Wandlers ein. Der Wert versteht sich als Angabe in mV.

6.4.8 Datenstrom

Tabelle 17 :Konfiguration des Datenstroms

Befehl	Werte	Default	Funktionsbeschreibung
stream_info	0,1,2	0	Wählt die Zusatzinformation aus, die neben den Daten im Datenstrom zusätzlich mit ausgegeben werden soll. 0= keine Zusatzinformation, 1= Systemzeit in ms seit Systemstart, 2= Systemzeit in Stunden-, Minuten- und Sekunden seit Systemstart
porta_stream	0,1	0	Zeigt kontinuierlich Port A an, wenn Port als Eingang konfiguriert ist
portc1_stream	0,1	0	Zeigt kontinuierlich Port C1 an, wenn Port als Eingang konfiguriert ist
portc2_stream	0,1	0	Zeigt kontinuierlich Port C2 an, wenn Port als Eingang konfiguriert ist
portd_stream	0,1	0	Zeigt kontinuierlich Port D an
adc_stream	0,1	0	Zeigt kontinuierlich die Messwerte des AD-Wandlers an
can_stream	0,1	0	Zeigt kontinuierlich die über den CAN-Bus eingehenden Meldungen an
com1_stream	0,1	0	Zeigt kontinuierlich die über die serielle Schnittstelle COM1 eingehenden Daten an
com2_stream	0,1	0	Zeigt kontinuierlich die über die serielle Schnittstelle COM2 eingehenden Daten an

6.4.9 Datenformate für ADC im Datenstrom

Ist der AD-Wandler aktiv, werden die Werte im folgenden Datenformat gesendet:

Syntaxangabe 3 :Datenformat des AD-Wandlers

```
[Zeit:]ADC<SP><Kanalnummer><SP><Wert><SP>µV<LF><CR>
```

Tabelle 18 : Erklärung zu Syntax 3

Befehl	**Werte**	Beschreibung
<Kanalnummer>	0..3	Angabe derv Kanalnummer
<Wert>	-5000000... 5000000	Messwert, Angabe in µV

Beispiel 4 : Datenausgabe der Kanäle 0..3

```
ADC 0   7201 µV
ADC 1 -26706 µV
ADC 2 -63519 µV
ADC 3 -16813 µV
ADC 0  51509 µV
ADC 1  17363 µV
ADC 2 -47382 µV
ADC 3    623 µV
```

Beispiel 5 : Datenausgabe von Kanal 2, zusätzlich Systemzeit, Baudrate 3 Hz (Steuerzeichen werden nicht dargestellt)

```
00:18:56: ADC 2 -338 µV
00:18:57: ADC 2 -318 µV
00:18:57: ADC 2 -138 µV
00:18:57: ADC 2 -166 µV
00:18:58: ADC 2 -336 µV
00:18:58: ADC 2 -321 µV
00:18:58: ADC 2 -145 µV
00:18:59: ADC 2 -159 µV
```

6.4.10 Datenformate für CAN im Datenstrom

Ist der CAN-Bus aktiv, werden die Werte im folgenden Datenformat gesendet:

Syntaxangabe 4 :Datenformat für den CAN-Bus

```
[Zeit:]<SP><can_message><id><SP><rtr><SP><len>[<SP><b0>][<SP><b1>]
[<SP><b2>][<SP><b3>][<SP><b4>][<SP><b5>][<SP><b6>][<SP><b7>]<LF><CR>
```

Tabelle 19 : Erklärung zu Syntax 4

Befehl	**Werte**	Beschreibung
<id>	id=0x00000000... 0x000007FF für (CAN2.0A) 0x00000000... 0x1FFFFFFF für (CAN2.0B)	CAN-Identifier
<rtr>	0,1	RTR-Bit
<len>	0..7	Anzahl der Datenbytes
<b0>..<b7>	0x00..0xFF	0..7 Datenbyte(s)

Beispiel 6 : Datenausgabe CAN-Bus
(Steuerzeichen werden nicht dargestellt)

```
can_message 00001FA0 1 8 00 11 22 33 44 55 66 77 88
can_message 00001FA3 1 8 88 11 22 33 44 55 66 77 88
can_message 00001FA3 1 4 11 22 33 44
can_message 002F1000 1 6 11 22 33 44 55 66
can_message 00F01FA3 1 0
```

Beispiel 7 : Datenausgabe CAN-Bus, zusätzlich Systemzeit
(Steuerzeichen werden nicht dargestellt)

```
00:18:56: can_message 00001FA0 1 8 00 11 22 33 44 55 66 77 88
00:18:57: can_message 00001FA3 1 8 88 11 22 33 44 55 66 77 88
00:18:57: can_message 00001FA3 1 4 11 22 33 44
00:18:57: can_message 002F1000 1 6 11 22 33 44 55 66
00:18:58: can_message 00F01FA3 1 0
```

Sollte ein Fehler vorliegen erfolgt folgende Meldung:

Syntaxangabe 5 :CAN-Fehlermeldung

```
[Zeit:]can_error<SP><errorbyte><LF><CR>
```

Tabelle 20 : Erklärung zu Syntax 5

Befehl	Werte	Beschreibung
<errorbyte>	XX	Bitweise Fehlerangabe mit folgenden Bits (siehe auch Datei can.h) CAN_ERR_OK=0x00, wenn kein Fehler vorliegt sonst wird Fehlerstatus mit folgenden Fehlerbits angegeben: CAN_ERR_XMTFULL=0x01: Sendepuffer des Controllers ist voll; diese Meldung kann ignoriert werden, da sie direkt vom Sendemechanismus verarbeitet wird. CAN_ERR_OVERRUN=0x02: Empfangspufferüberlauf CAN_ERR_BUSERROR=0x04: Fehlerzähler erreichte Limit CAN_ERR_BUSOFF=0x08: Busfehler, Controller ging 'Bus-Off' CAN_ERR_RECEIVEBUF_OVERFLOW=0x10: Überlauf des Software-Empfangspuffers, CAN-Meldungen gingen verloren, da Puffer nicht rechtzeitig ausgelesen wurde. CAN_ERR_TRANSMITBUF_OVERFLOW=0x20: Bei einem Sendeversuch ist der Software-Sendepuffer übergelaufen. Die CAN-Meldung wurde nicht im Puffer abgelegt; ggf. Sendeversuch wiederholen.

Beispiel 8 : Fehlermeldung CAN-Bus, zusätzlich Systemzeit (Steuerzeichen werden nicht dargestellt)

```
00:13:22: can_error 20
00:13:22: can_error 28
```

6.4.11 Datenformate für Port A, Port C und Port D im Datenstrom

Die digitalen Eingänge werden in den nachfolgenden Datenformaten übertragen. Dabei erfolgt optional vor dem Eingangswerten eine Zeitangabe. Die Ausgabe erfolgt Bitweise, wobei die Bits die Werte „0“ und „1“ aufweisen.

Syntaxangabe 6 :Datenformate der digitalen Ports

```
[Zeit:]PortA<SP><a7><a6><a5><a4><a3><a2><a1><a0><LF><CR>
[Zeit:]PortD<SP><d6><d5><d4><d3><d2><d1><d0><LF><CR>
[Zeit:]PortC1<SP><c3><c2><c1><c0><LF><CR>
[Zeit:]PortC2<SP><c7><c6><c5><c4><LF><CR>
```

Tabelle 21 : Erklärung zu Syntax 6

Befehl	Werte	Beschreibung
<d6>...<d0>	0,1	Bitweise Angabe für Port D
<c3>...<c0>	0,1	Bitweise Angabe für Port C1

Befehl	**Werte**	Beschreibung
<c7>...<c0>	0,1	Bitweise Angabe für Port C2
<a7>..<a0	0,1	Bitweise Angabe für Port A

Beispiel 9 : Digitale Eingänge mit Systemzeit (Format hh:mm:ss; Steuerzeichen werden nicht dargestellt)

```
00:31:56: PortD 0000000
00:31:56: PortA 11111111
00:31:56: PortC1 1111
00:31:56: PortC2 1111
```

6.4.12 Datenformate der seriellen Schnittstellen COM1/COM2 im Datenstrom

Die Kommunikation der seriellen Schnittstellen ist zeilenorientiert ausgelegt, d.h. die eingehenden Daten/Zeichen, die über COM1 und COM2 eingehen, werden in einen Zeilenpuffer geschrieben bis ein Zeilenumbruch <CR> die Zeile abschließt. Danach wird die Zeile weitergeleitet. Es dürfen maximal 50 Zeichen pro Zeile auftreten.

Ist die entsprechende Schnittstelle aktiv, werden die Werte im folgenden Datenformat gesendet, wenn diese Übertragung im Bereich Datenstrom aktiviert wurde:

Syntaxangabe 7 :Datenformat für COM1

```
com1<SP><Zeile><LF><CR>
```

Syntaxangabe 8 :Datenformat für COM2

```
com2<SP><Zeile><LF><CR>
```

Tabelle 22 : Erklärung zu Syntax 7 und Syntax 8

Befehl	**Werte**	Beschreibung
<Zeile>	STRING[50]	Zeichenkette

6.4.13 Leuchtdioden

Auf der DeviLAN-Platine befinden sich 4 Leuchtdioden von denen zwei direkt über die nachfolgenden Befehle (Tabelle 23) angesprochen werden können. Die beiden anderen dienen der Betriebsanzeige für den Netzwerkbetrieb (rote LED) und des AD-Wandlers (gelbe LED).

Tabelle 23 :Ein- und Ausschalten der LEDs

Befehl	Werte	Default	Funktionsbeschreibung
led1	0,1	0	0=LED1 ein, 1=LED1 aus
led2	0,1	0	0=LED2 ein, 1=LED2 aus

6.4.14 Provider-Einwahl / PPP-Verbindung

Die Konfiguration des CAN-Bus erfolgt über die Befehle in Tabelle 12 (siehe auch Kap. 5.2.6).

Tabelle 24 :Konfiguration für PPP/Email

Befehl	Wert/Typ	Default	Funktionsbeschreibung
provider_name	STRING [19]	Freenet	Name des Internetprovider
provider_number	STRING [19]	01929	Telefonnummer für Interneteinwahl
provider_user	STRING [29]	test	User-Name
provider_password	STRING [29]	test	User-Password
provider_connect	0,1	0	Legt fest, ob eine Einwahl über ein Modem bei einem Internetprovider erfolgen soll 0: keine Einwahl 1: Einwahl erfolgt
mail_serverip	IP	195.20.224.208	IP-Adresse des Mailservers, über den eine Mail verschickt werden soll, die die dynamische IP-Adresse des DeviLAN-Moduls enthält.
mail_send	MAIL[39]	devilan@mydevilan.de	Mailadresse des Absenders (beliebig einstellbar)
mail_receive	MAIL [39]	xyz@xyz.de	Mailadresse des Empfänger an den die Mail mit der dynamisch zugewiesenen IP des Moduls verschickt werden soll
mail_pppip	0,1	0	Legt fest, ob dynamische IP per Email verschickt werden soll. 0: es wird keine Email verschickt 1: es wird eine Email verschickt

6.4.15 Weitere Einstellungen und Abfragebefehle

Über die Befehle in Tabelle 25 können weitere Einstellungen getätigt und Information abgefragt werden.

Tabelle 25 :Konfiguration der weiteren Einstellungen

Befehl	Wert/Typ	Default	Funktionsbeschreibung
modul_name	STRING [39]	DeviLAN Universal WebInter-face	Setzt den Namen für das DeviLAN-Modul. Über den Namen ist eine bessere Unterscheidung der Module als nur über die IP-Adresse möglich.
get_config	-	-	Fordert das DeviLAN-Modul auf die gesamte Konfiguration sowie die aktuellen Werte der digitalen Ports zu senden
save_config	-	-	Speichert die aktuelle Konfiguration und die voreingestellten Werte für die digitalen und analogen Ausgänge sowie die Meldungen für COM1, COM2 und CAN des DeviLAN-Moduls

6.4.16 Weitere Meldungen von DeviLAN

Nach dem erfolgreichen Aufbau einer Verbindung zum Modul überträgt dieses automatisch eine Startmeldung bzw. bei bestehender Verbindung auf Anfrage zusätzliche Informationen (Tabelle 26).

Tabelle 26 : Zusatzinformationen

Befehl	Typ	Beschreibung
modul_ip	IP	Gibt die IP-Adresse des DeviLAN-Modul an.
modul_mac	XX XX XX XX XX XX	Gibt die MAC-Adresse des DeviLAN-Modul an
modul_port	UINT	Gibt die Portnummer an über die die aktuelle Socketverbindung erfolgt.
modul_copyright	STRING	Gibt den Ersteller der Software an und das Erstellungsjahr an.

Die aktuelle Konfiguration kann über den *get_config*-Befehl angefordert werden.

6.5 Kommandointerface des Betriebssystems

Das Betriebssystem der auf dem DeviLAN-Modul befindlichen Prozessoreinheit (SC12) verfügt über ein Kommandointerface (DOS-ähnlich), mit dem elementare Einstellungen an der Systemkonfiguration vorgenommen, Statusinformationen abgefragt sowie ein Zugriff auf das Dateiverzeichnis erfolgen kann. Der Kommandointerpreter führt dabei Befehle aus, die entweder über eine Konsole (z. B Telnet) eingegeben oder aus einer Steuerdatei/Batchdatei (autoexec.bat) eingelesen werden. Es sind die nachfolgenden Kommandos und Funktionen (Tabelle 27) implementiert.

Tabelle 27 : Kommandos des Betriebssystems

Befehl	Beschreibung
DEL filename Löschen einer Datei	Löscht eine Datei bzw. alle Dateien, die mit den eingegeben Wildcards übereinstimmen. Beispiel: del *.dat ; Alle Dateien mit der Endung 'dat ' löschen
DIR filename Inhaltsverzeichnis auflisten	Gibt das Inhaltverzeichnis oder alle Einträge die mit den eingegeben Wildcards übereinstimmen aus. Beispiele: dir ; Ausgabe des gesamten Verzeichnisses dir *.exe ; Alle Dateien mit Endung 'exe' auflisten
TYPE file Dateiinhalt anzeigen	Gibt den Inhalt einer Datei über die Konsole aus. Beispiel: type autoexec.bat ; Text der Datei autoexec.bat wird angezeigt
COPY file1 file2 Kopieren einer Datei	Kopiert die Datei file1 nach file2. file1 und file2 müssen vollständige Dateinamen sein. Wildcards wie '*' oder '?' sind nicht erlaubt. Beispiel: copy autoexec.bat test.txt
REN file1 file2 Umbenennen einer Datei	Umbenennen der Dateien file1 in file 2. file1 und file2 müssen vollständige Dateinamen sein. Wildcards wie '*' oder '?' sind nicht erlaubt. Beide Dateien müssen im selben Verzeichnis liegen. Beispiel: ren autoexec.bat test.txt
MD dir Inhaltsverzeichnis erzeugen	Erzeugen eines neuen Inhaltsverzeichnisses. Beispiel: md temp ; Verzeichnis temp wird erzeugt
XTRANS Dateitransfer mit XModem	Senden oder Empfangen einer Datei mit Xmodem/CRC-Protokoll über COM1. Beispiele: XTRANS COM R chip.ini ; Empfangen der Datei chip.ini über COM1 XTRANS COM S test.txt ; Senden der Datei test.txt über COM1 **Anmerkung:** Dieses Kommando arbeitet nur, wenn das DeviLAN-Steuerprogramm (devilan.exe) nicht läuft!

Befehl	Beschreibung
MEMOPT 0/1	Freigeben oder Sperren der Speicheroptimierung beim Laden einer 'exe'-Datei. Als Voreinstellung ist die Speicheroptimierung nicht freigegeben. Eine 'exe'-Datei erhält in der Startphase normalerweise den gesamten verfügbaren Speicher. Während der Startsequenz stellt das Programm diesen Speicherbereich wieder her. Wenn die Option freigegeben ist, erhält das Programm nur den Speicher, der im Header der 'exe'-Datei als erforderlich angegeben ist. Dadurch steht anderen Programmen mehr Speicher zur Verfügung. Es können jedoch Fehler bei dem Versuch auftreten Speicher vom Heap zu reservieren. Anwender von Borland C/C++ brauchen diese Option in der Regel nicht. Bei der Verwendung von Borland Pascal ist es jedoch meist notwendig, da Pascal-Programme normalerweise den Speicherbereich in der Startphase nicht wiederherstellen. Anmerkung: Seit der SC12 BIOS Version 0.67, ist MEMOPT gesperrt. Frühere Versionen hatten MEMOPT freigegeben, was aber ggf. zu Fehlern bei der Verwendung von malloc() führte.
CD dir Arbeitsverzeichnis wechseln	Wechselt das aktuelle Arbeitsverzeichnis. Beispiel: cd temp ; Verzeichnis temp al s Arbeitsverzeichnis einstellen
RD dir Inhaltsverzeichnis löschen	Löschen eines Inhaltsverzeichnisses. Diese Kommando wird nur ausgeführt, wenn das Verzeichnis keine Dateien enthält. Beispiel: rd temp ; Verzechnis temp löschen
CON Konsole festlegen	Legt fest welche Schnittstelle als Konsole verwendet wird. Mögliche Schnittstellen für deviLAN-Module sind COM und Telnet. Es können beide Schnittstellen als Konsole angegeben werden. Die Einstellung ist nur bis zum nächsten Systemneustart gültig. Beispiele: con com ; COM1 ist Konsole con com telnet ; COM1 und Telnet sind Konsole **Anmerkung:** COM kann nur als Schnittstelle als Konsole verwendet werden, wenn das DeviLAN-Steuerprogramm (devilan.exe) nicht läuft! Werkseitig ist Telnet als Konsole eingestellt.
IW Wort einlesen	Liest ein Wort (2 Byte) von einer vorgegeben Bit-Adresse ein. Die Angabe der Adresse und die Ausgabe des gelesenen Wertes erfolgt hexadezimal. Beispiel: iw 600 ; Byte von Adresse 600 lesen
OW Byte ausgeben	Schreibt ein Wort (2 Byte) auf eine vorgegebene Adresse. Die Angabe der Adresse und des zu schreibenden Bytes erfolgt hexadezimal. Beispiel: ow 600 FFAA ; Wort FFAA auf Adresse 600 schreiben
IB Byte einlesen	Liest ein Byte von einer vorgegeben Adresse ein. Die Angabe der Adresse und die Ausgabe des gelesenen Wertes erfolgt hexadezimal. Beispiel: ib 600 ; Byte von Adresse 600 lesen
OB	Schreibt ein Byte auf eine vorgegebene Adresse. Die Angabe der Adresse und des zu schreibenden Bytes erfolgt hexadezimal. Beispiel: ob 600 FF; Byte FF auf Adresse 600 schreiben

Befehl	Beschreibung
PCS Chip Select freigeben	Freigabe einer Chip Select Line. Gültige Werte sind 0,1,2,3,4,5 und 6. Beispiel: PCS 6
ALE ALE Pin freigeben	Gibt den Pin ALE frei bzw. sperrt ihn. Gültige Werte sind 1 (freigeben) und 0 (sperren). Beispiel: ale 1 ; Freigabe des ALE-Pins
IP address IP setzen	Setzt die IP-Adresse des Moduls. Die neue Adresse wird in der Datei chip.ini gespeichert. Die DHCP-Option wird gleichzeitig abgeschaltet. Die neue IP-Adresse ist erst nach einem Systemneustart gültig. Zur Überprüfung der neuen Adresse kann das IPCFG-Kommando verwendet werden. Beispiel: ip 130.75.65.130
NETMASK mask Netmask setzen	Setzt die Subnet-Maske des Moduls. Die neue Adresse wird in der Datei chip.ini gespeichert. Die DHCP-Option wird gleichzeitig abgeschaltet. Die neue Subnet-Maske ist erst nach einem Systemneustart gültig. Zur Überprüfung der neuen Adresse kann das IPCFG-Kommando verwendet werden. Beispiel: netmask 255.255.255.0
GATEWAY address Gateway setzen	Setzt die Gateway-Adresse die das Modul verwenden soll. Die neue Adresse wird in der Datei chip.ini gespeichert. Die DHCP-Option wird gleichzeitig abgeschaltet. Die neue IP-Adresse ist erst nach einem Systemneustart gültig. Zur Überprüfung der neuen Adresse kann das IPCFG-Kommando verwendet werden. Beispiel: gateway 130.75.65.126
DHCP 0/1 DHCP Freigabe	Gibt die Verwendung von DHCP frei bzw. sperrt diese. DHCP steht für Dynamic Host Configuration Protocol, also die dynamische Vergabe von IP-Adressen innerhalb eines Netzwerks. Über einen DHCP-Server kann der Netzwerk Administrator IP-Adressen für die verschiedenen Geräte im Netzwerk definieren ohne jedes Gerät einzeln konfigurieren zu müssen. Beispiel: dhcp 1 ; DHCP verwenden
IPETH Neustart des ethernet-Interface	Neustart des Ethernet-interface z. B. nach der Änderung der IP-Adresse ohne das ein System-Neustart erfolgt. Anmerkung: Wenn eine Fehlermeldung ausgegeben wird sollten die IP Parameter überprüft werden. Häufig ist ein ungültige Gateway-Adresse die Ursache für einen Fehler beim Neustart des Ethernet-Interface.
TCPIPMEM Speicherbelegung des TCP/IP-Bereichs ausgeben	Gibt die TCP/IP-Speicherbelegung an. Ausgegen wird der max. reservierte Speicher des TCP/IP-Stack und der momentan belegte Speicher.
BATCHMODE Ausführungsmodus festlegen	Setzt den Modus für die Ausführung von batch-Dateien auf concurrent (gleichzeitig) oder sequential (hintereinander) Beispiele: BATCHMODE 1 ; Kommandos hintereinander ausführen BATCHMODE 0 ; gleichzeitig Ausführung der Kommandos
FTP 0/1 FTP Freigabe	Gibt den Start des FTP-Servers frei oder sperrt diesen. Der neue Wert wird in der Datei chip.ini gespeichert. Die neue Einstellung wird erst nach eine System-Neustart gültig. Beispiel: ftp 1 ; Freigabe des FTP-Server

Befehl	Beschreibung
IPCFG IP-Konfiguration ausgeben	Gibt die Konfiguration (IP, Gateway, Netmask, DCHP, Seriennummer) des Ethernet-Interface aus.
REBOOT System-Neustart	Führt einen Neustart des Systems aus. Zunächst wird das Dateisystem geschlossen dann wird der Watchdog konfiguriert, um einen Reset auszuführen. Anmerkung: Laufende Tasks werden nicht über diesen Neustart benachrichtigt.
WAIT secs Wartezeit	Stoppt die Ausführung des Kommandointerpreters für den angegeben Zeitraum (Zeitangabe in Sekunden). Beispiel: wait 5 ; 5 Sekunden warten
FORMAT A: [/C:n] [/E Dateiverzeichnis formatieren	Formatiert das Dateiverzeichnis der Flashdisk. **Alle Informationen gehen dabei verloren**. Die Angabe der Clustergröße ist optional. Der voreingestelle Wert ist 2 für Laufwerk A und 4 für LAufwerk B. Wenn der Parameter /E angegeben wird wird die Dateibereich mit Nullen gefüllt. **Anmerkung:** Es ist sicher zu stellen, dass keine Tasks während der Formatierung auf LAufwerk A zugreift. Beispiel: format a: /c:2 /e
VER BIOS Version ausgeben	Gibt die Seriennummer, die BIOS-Version und das Erstellungsdatum aus.
MEM Speicherbelegung ausgeben	Gibt die Speicherbelegung inklusive der Task-Namen, die den Speicher belegt haben, aus.
CGISTAT CGI-Statistik ausgeben	Gibt alle installierten CGI-Routinen an.
CLOSETELNET Telnet-Verbindung beenden	Dieses Kommando beendet die aktuelle Telnet-Verbindung.
WEBSTAT Web-Statistik ausgeben	Gibt die Einstellungen des Webservers wie Port-Nummer, Pfadangabe des für Hauptverzeichnis der Webseiten oder die Startseite aus.
PING Echo-Anforderung senden	Testet eine Netzwerkverbindung mit dem ICMP-Kommando 'ping'. Dieses Kommando sendet im Abstand von einer Sekunde 4 Echo-Anforderungen (64 Byte) an die angegebene Adresse und zeigt das Ergebnis an. Beispiel: PING 192.168.200.10

Befehl	Beschreibung
TASKS Anzeige der Task	Listet alle Tasks inklusive Task-Status, CPU-Last, etc auf. Jede ms wird die Zählernangabe der Task um eins erhöht. Nach 10 Sekunden werden die Zählerstände kopiert und auf Null zurückgesetzt. Anmerkungen: Beim ersten Aufruf von TASKS wird die Timer-Interrupt-Routine des RTOS durch eine Routine des Task-Monitors ersetzt. Erst nach 10 Sekunden gibt das TASKS-Kommando brauchbare Werte zurück. Das Kommando zeigt für DOS-Applikationen nur eine Task-Stack Größe von 128 Byte an, da das DOS-Programm zur Laufzeit auf seinen internen Stack umschaltet, der für das Betreibssystem nicht sichtbar ist. Es können maximal 35 TASKS beobachtet werden. Zum Beenden des Tasks-Monitors ist das UTASKS-Kommando zu verwenden. Der angezeigte Task-Status (16 Bit hex. Angabe) ist nur eine Momentaufnahme der TASK: Bit0 timer wait (used with other bits) Bit1 trigger wait (i.e. idle) Bit2 semaphore wait Bit3 event group wait Bit4 message exchange wait Bit5 message send wait Bit6 suspended (waiting for resume) Bit7 waiting for wake Bit7. Bit15 internal use only Laufende System-Tasks (falls diese nicht in der chip.ini abgeschaltet wurden) sind: Sehr hohe Priorität haben: AMXK prio = 00 Kernel Task ETH0 prio= 05 Ethernet Receiver task Normal Priorität haben: PPPS prio= 06 PPP-Server TCPT prio= 06 TCPIP Timer Task CFGS prio= 07 UDP-Config-Server TELN prio= 11 Telnet-Server MTSK prio= 12 Konsole (command shell) Niedrige Priorität haben: WEBS prio= 41 Web-Server FTPS prio= 41 FTP-Server
HELP Kommandos auflisten	Listet alle verfügbaren Kommandos der Konsole auf.

7 Anbindung eigener Applikationen

Die Anbindung des Modul kann über ein Netzwerk über eine Socketschnittstelle (TCP/IP), über eine direkte RS232-Verbindung zwischen PC und DeviLAN oder auch über Modem erfolgen.

Als Beispielapplikation steht das PC-Programm DeviLANControl zur Verfügung (siehe Kap. 5). Mit ihm läßt sich das DeviLAN-Modul sowohl über das Netzwerk als auch über eine serielle Schnittstelle des PCs bedienen. Durch die Verbindung mit einen externen Modem kann DeviLAN direkt über eine Telefonleitung angewählt werden.

Wie eine TCP/IP-Socket-Verbindung aufgebaut wird, ist in den Beispielprogrammen Ihrer Entwicklungsumgebung (z.B Borland C++, Borland Builder, Delphi, MS Visual C/C++, etc.) dokumentiert und wird hier nicht weiter behandelt. Ein ausführliches Beispielprojekt (WINSOCK.IDE) befindet sich in der Borland 5.02 Entwicklungsumgebung im Verzeichnis EXAMPLES unter den Pfad OWL/CLASSES/WINSOCK.

Grundsätzlich ist es nur notwendig die IP-Adresse des Moduls und die Standard Portnummer 3333 des Kommando/Datenstrom-Interface für den Verbindungsaufbau einzugeben.

Das Modul ist Multiuser-fähig, d.h. es können gleichzeitig bis zu drei Socketverbindungen mit verschiedenen Applikationen bestehen. Das Modul verteilt dabei alle eingehenden Kommandos an alle angeschlossenen Applikationen, so dass diese immer über die aktuelle Konfiguration verfügen.

7.1 Anbindung über Netzwerk

LAN/Internet

7.2 Funktionstest über Telnet

Auf den meisten PCs mit Windows ist ein Terminalprogramm mit der Bezeichnung Telnet vorhanden. Über dieses kann auch eine Verbindung zu eine Kommando/datenstromverbindung zu einem DeviLAN-Modul aufgebaut werden. Gehen Sie dazu wie folgt vor:

Starten Sie Telnet, z.B. über „Start“, „Ausführen“ „Telnet“; Es erscheint folgende Fenster:

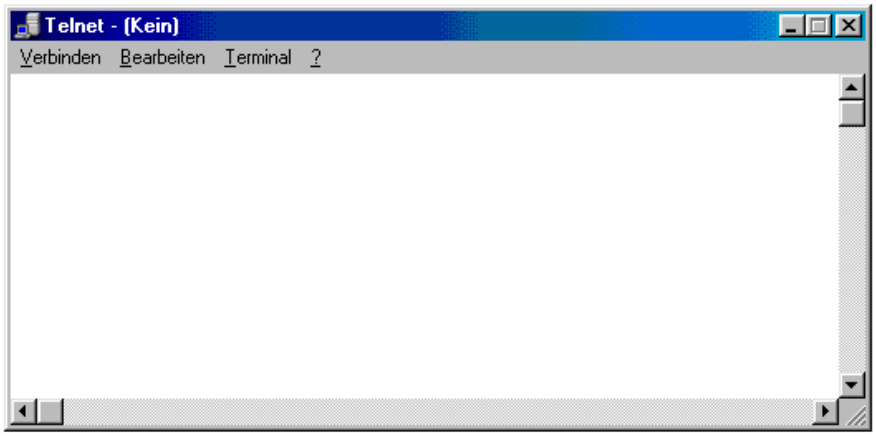


Abb. 15: Telnet-Startfenster

Im Menü auf „Verbinden“, „Netzwerksystem“ und unter „Hostname“ die IP-Adresse des Moduls eintragen also z.B. 130.75.65.130 (Abb. 16). Unter „Anschluss“ die Standard Portnummer 3333 des Kommando/Datenstrom-Interface angeben. Danach über „Verbinden“ die Verbindung aufbauen.

Abb. 16: Telnet-Startfenster

Ist der Verbindungsaufbau erfolgreich gibt das Modul automatisch einige Informationen aus (Abb. 17).

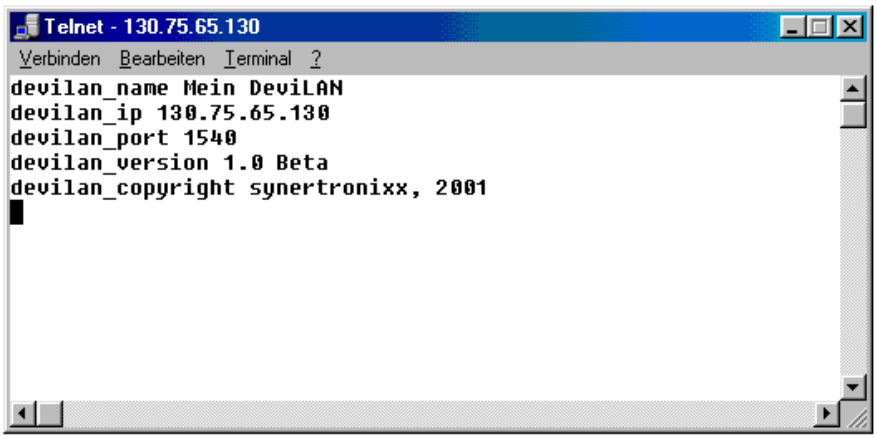


Abb. 17: Ausgabe bei erfolgreichem Verbindungsaufbau

Nun können manuell Kommandos an das Modul gesendet werden. Bitte beachten, dass das Kommandointerface zeilenorientiert arbeitet und kein Echo erzeugt, d.h. beim Eintippen sind zunächst keine Zeichen zu sehen. Erst nach einem Return (Zeilenabschluss) liefert das Modul Daten in der Form einer Quittung bzw. Fehlermeldung zurück.

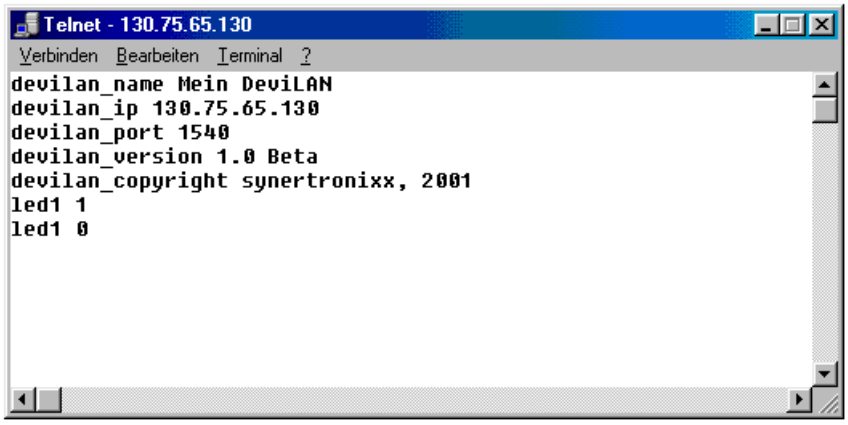


Abb. 18: Manuelle Kommandoeingabe

Sollten Sie sich Vertippen Drücken Sie einfach Return. Die Verwendung von Steuertasten wie „TAB“, „Backspace“, „Entfernen“, etc. kann zu Fehler in der Form führen, dass Kommandos nicht richtig erkannt werden.

8 Programmierinterface, C-Treiberroutinen

Das Programmierinterface bietet die Möglichkeit, mittels der bereit gestellten Objekt- und Headerdateien auf die DeviLAN Hardware zuzugreifen und eigene Applikation zu erstellen. Alle Treiber wurden für Borland C 5.02 entwickelt und getestet.

Nachfolgend werden nur die von synertronixx bereitgestellten Routinen zur Kommunikation und zur Ansteuerung der DeviLAN-Hardware beschrieben. Alle Dateien für die Betriebssystemroutinen sowie Beispielprogramme für die Programmierung des SC12-Moduls können unter

www.bcl-online.de bei der Fa. Beck-IPC

unter den angegebenen Nutzungsbedingungen herruntergeladen werden und werden hier nicht weiter erläutert.

Die DeviLAN-Treiberroutinen sind in der Bibliothek **devilan.lib** zusammengefasst und enthalten den ausführtbaren Code für die nachfolgenden Header-Dateien. Die Bibliothek muss zum Projekt dazu gelinkt werden.

Tabelle 28 :Header-Dateien

Header-Datei	Beschreibung
adc.h	Routinen für den Analog-Digital-Wandler sowie den Digital-Analog-Wandler
can.h	Routinen für den CAN-Bus
intrhndl.h	Routinen für die Verwendung der externen Interrupts sowie die Initialisierung des DeviLAN-Moduls
port.h	Routinen für die digitalen Ein- und Ausgangsports A, B, C und D
serial.h	Routinen für serielle Schnittstelle COM2

8.1 Allgemeine Routinen zur DeviLAN Initialisierung

Nachfolgend werden die Routinen zur Initialisierung des DeviLAN-Moduls beschrieben (Tabelle 33). Weitere Erläuterungen zur Verwendung der Routinen befinden sich auch in der Projektdatei **examples.ide**, die die Quelltexte **dio_test.c**, **can_test.c**, **addatest.c** und **can_test.c** enthält.

Tabelle 29 :Bibliotheksfunktionen für die Initialisierung

Funktion	Beschreibung
void **IPCEnableExternalInterrupts**(void)	Gibt die benötigten Interrupts für die externe DeviLAN-Hardware frei
void **IPCDisableExternalInterrupts**(void)	Sperrt die externen Interrupts.
void **DeviLANInit(**void)	Initialisiert die DeviLAN-Hardware.

8.2 Digitale Ein- und Ausgänge

Nachfolgend werden die Routinen zur Ansteuerung der digitalen Ein- und Ausgänge des DeviLAN-Moduls beschrieben (Tabelle 30). Weitere Erläuterungen zur Verwendung der Routinen befinden sich auch in der Projektdatei **examples.ide**, die den Quelltext **dio_test.c** enthält, um das Beispielprogramms **dio_test.exe** zu erzeugen.

Tabelle 30 :Bibliotheksfunktionen für die digitalen Ein- und Ausgänge

Funktion	Beschreibung
void **Port8255Init**(void)	Initialisiert den E/A-Baustein 82C55. Diese Funktion muss einmalig aufgerufen werden, um auf die Ein- und Ausgänge zugreifen zu können.
void **PortConfiguration**(int v_port_configuration)	Festlegen der Ein- und Ausgänge. Die Funktion muss für jeden Port (A,C1,C2) einzeln aufgerufen werden. **Parameter:** *v_port_configuration* wird über enum-Typ TPORTCONFIGURATION angegeben: PORTA_INPORT: Port A ist Eingang PORTA_OUTPORT: Port A ist Ausgang PORTC1_INPORT: Port C1 ist Eingang PORTC1_OUTPORT: Port C1 ist Ausgang PORTC2_INPORT: Port C2 ist Eingang PORTC2_OUTPORT: Port C2 ist Ausgang
void **PortAByteOut**(unsigned char byte)	Schreibt ein Byte auf Port A.
int **PortABitOut**(int v_nr, int v_pegel)	Bitweiser Zugriff auf Port A **Parameter:** *v_nr* gibt Nummer (0..7) des Ausgangsbits an; *v_pegel* wird über enum-Typ TPEGEL angegeben: HIGH: Ausgang wird gesetzt LOW: Ausgang wird rückgesetzt **Rückgabe:** 0, wenn *v_nr* , *v_pegel* im gültigen Bereich liegen, -1 sonst
unsigned char **PortAIn**(void)	Liest Port A (8Bit).
void **PortBByteOut**(unsigned char byte)	Schreibt ein Byte auf Port B (nur 7 Bits). Port B ist immer als Ausgang konfiguriert.

Funktion	Beschreibung
int **PortBBitOut**(int v_nr, int v_pegel)	Bitweiser Zugriff auf Port B (nur 7 Bits). Port B ist immer als Ausgang konfiguriert. **Parameter:** *v_nr* gibt Nummer (0..6) des Ausgangsbits an; *v_pegel* wird über enum-Typ TPEGEL angegeben: HIGH: Ausgang wird gesetzt LOW: Ausgang wird rückgesetzt **Rückgabe:** 0, wenn *v_nr* , *v_pegel* im gültigen Bereich liegen, -1 sonst
void **PortCByteOut**(unsigned char byte)	Schreibt ein Byte auf Port C.
int **PortCBitOut**(int v_nr, int v_pegel)	Bitweiser Zugriff auf Port C **Parameter:** *v_nr* gibt Nummer (0..7) des Ausgangsbits an; *v_pegel* wird über enum-Typ TPEGEL angegeben: HIGH: Ausgang wird gesetzt LOW: Ausgang wird rückgesetzt **Rückgabe:** 0, wenn *v_nr* und *v_pegel* im gültigen Bereich liegen, -1 sonst
unsigned char **PortCByteIn**(void)	Liest Port C (Port C1 und C2, 8 Bit). Port C1: Bits 0..3 Port C2: Bits 4..7
unsigned char **PortDByteIn(**void)	Liest Eingangsport D (nur 7 Bit); MSB ist immer 0. Port D ist immer als Eingang konfiguriert.

8.3 Analog-Digital- und Digital-Analog-Wandler

Nachfolgend werden die Routinen zur Ansteuerung der analogen Ein- und Ausgänge des DeviLAN-Moduls beschrieben (Tabelle 31). Weitere Erläuterungen zur Verwendung der Routinen befinden sich auch in der Projektdatei **examples.ide**, die den Quelltext **addatest.c** enthält, um das Beispielprogramms **addatest.exe** zu erzeugen.

Tabelle 31 :Bibliotheksfunktionen den AD- und DA-Wandler

Funktion	Beschreibung
void **ADCInit**(void)	Initialisiert den AD und DA-Wandler. Diese Funktion muss zu Beginn einmal aufgerufen werden.
unsigned char **ADCStartConvert**(void)	Startet den AD-Wandler. **Rückgabe:** ADC_NOERROR, wenn Kommando erfolgreich ausgeführt wurde, ADC_ACKNOWLEDGEERROR sonst

Funktion	Beschreibung
unsigned char **ADCStopConvert**(void)	Stoppt den AD-Wandler. **Rückgabe:** ADC_NOERROR, wenn Kommando erfolgreich ausgeführt wurde, ADC_ACKNOWLEDGEERROR sonst
unsigned char **ADCValidData**(void)	Überprüft ob neue Daten vom AD-Wandler vorliegen. Ist dies der Fall müssen diese mit der Funktion ADCRead ausgelesen werden. **Rückgabe:** 1, wenn neue Daten vorliegen; 0 sonst
unsigned char **ADCSet**(unsigned int adc_baudrate, unsigend char adc_channel, unsigned char adc_gain)	Stellt die Abtastrate, den Kanal und die Vorverstärkung der Signale ein und startet den AD-Wandler. **Parameter:** *adc_baudrate* gibt die Abtastrate im Bereich 3...1000 an. *adc_gain* Faktor für Vorverstärkung (1,2,4,8,16) an *adc_channel* gibt den Kanal (0..3) der abgetastet werden soll an. 4 bedeutet dass alle Kanäle hintereinander gewandelt werden. Auf Grund von Umschaltvorgängen sinkt die Abtastrate auf ca. 25% des eingestellten Wertes. **Rückgabe:** ADC_NOERROR bei erfolgreicher Ausführung, ADC_GAINOUTOFRANGE bei ungültiger Verstärkung ADC_DATARATEOUTOFRANGE bei ungültiger Abtastrate ADC_CHANNELOUTOFRANGE bei ungültiger Kanalangabe
void **ADCSelectChannel**(unsigned char channel)	Wählt den AD-Kanal für Einzelkanalwandlung aus, der abgetastet werden soll. Mit diesen Funktion kann im laufenden AD-Wandlerbetrieb eine Kanalumschaltung erfolgen. **Parameter:** *channel* gibt die Kanalnummer (nur Kanäle 0..3) an. Für kontinuierliche Wandlung aller Kanäle ist die Funktion ADCSet zu verwenden.

Funktion	Beschreibung
int **ADCRead**(unsigned char* channel, long int* value)	Liest Daten aus dem AD-Wandler **Parameter:** *channel* enthält die Nummer des Kanals (1..4) *value* enthält den Abtastwert in Mikrovolt. **Rückgabe:** 1, wenn gültige Daten ausgelesen wurden; -1 sonst
unsigned char **DACOut**(unsigned char channel, unsigned int value)	Stellt die Ausgabespannung für den DA-Wandler ein. Parameter: *channel* gibt den Ausgabekanal (1,2) an. *value* Angabe der Ausgabespannung in mV im Wertebereich 0...4095 an. Rückgabe: DAC_NOERROR wenn Parameter im gültigen Bereich DAC_CHANNELOUTOFRANGE bei ungültiger Kanalangabe DAC_VALUEOUTOFRANGE bei ungültiger Werteangabe

8.4 Serielle Schnittstellen COM1 und COM2 sowie LEDs

Das DeviLAN-Modul verfügt über drei serielle Schnittstellen:

Die Schnittstelle COM1 wird über den Steckverbinder CON1 (Kap. 10.4) nach aussen geführt und besitzt RS232 Pegel. Sie wird in der Dokumentation des SC12 auch als **externe** Schnittstelle bezeichnet (Nr.=0). Alle Routinen für die Ansteuerung können der Dokumentation des Betriebssystem entnommen werden und werden hier nicht weiter erläutert.

Eine zusätzliche serielle Schnittstelle ist COM2, die über einen weiteren Schnittstellen-Baustein bereitgestellt wird. Sie wird auch als interne Schnittstelle bezeichnet und steht auf dem 40-pol. Steckverbinder (Kap. 10.8) für die Anbindung von Geräten und Sensoren bereit.

Die Schnittstelle COM3 wird für die Kommunikation mit diverser DeviLAN-Hardware benötigt und steht daher nicht für Betriebssystemfunktionen wie z. B. eine PPP-Kommunikation oder auch für anwenderspezifische Funktionen bereit. COM3 wird in der Dokumentation des SC12 als **COM**-Schnittstelle (Nr.=1) bezeichnet.

Nachfolgend werden die Routinen zur Ansteuerung der analogen Ein- und Ausgänge des DeviLAN-Moduls beschrieben (Tabelle 32). Weitere Erläuterungen zur Verwendung der Routinen befinden sich auch in der Projektdatei **examples.ide**, die den Quelltext **com2test.c** enthält, um das Beispielprogramms **com2test.exe** zu erzeugen.

Tabelle 32 :Bibliotheksfunktionen für die serielle Schnittstelle COM2 und LEDs

Funktion	Beschreibung
unsigned char **COM2Init**(unsigned long int baudrate, int databits, int stopbits, int parity, int flowCtrl)	Initialisiert die serielle Schnittstelle COM2. **Parameter:** *baudrate* stellt die Übertragungsrate ein typische Werte sind 300...56000 Baud. *databits* gibt die Anzahl der Datenbits (5-8) über den enum-Typ TComPortDataBits an (db5BITS, db6BITS, db7BITS, db8BITS) an. *stopbits* gibt die Anzahl der Stopbits (1, 1.5, 2) über den enum-Typ TComPortStopBits (sb1BITS, sb1HALFBITS, sb2BITS) an. *parity* spezifiziert die Parität über den enum-Typ TComPort-Parity (ptNONE, ptODD, ptEVEN, ptMARK, ptSPACE). *flowCtrl* spezifiziert die Datenflusskontrolle über den enum-Typ TComPortHandshaking (hsNONE, hsRTSCTS, hsXONXOFF). **Rückgabe:** Fehlermeldung; Fehlerbits werden bei fehlerhafter Angabe gesetzt: INITSERIAL_NO_ERROR: kein Fehler INITSERIAL_BAUDRATE_ERROR: Baudrate INITSERIAL_HANDSHAKE_ERROR: Handshake INITSERIAL_DATABITS_ERROR: Datenbits INITSERIAL_STOPBIT_ERROR: Stoppbits INITSERIAL_PARITY_ERROR: Parität
void **COM2Release(**void)	Den ursprünglichen Interruptvektor wieder herstellen. Diese Funktion **muss** vor dem Programmende einmal aufgerufen werden, da es sonst zu Fehlern kommen kann.
TSerialError **COM2GetError**(void)	Gibt den Fehlerstatus der Schnittstelle zurück. SERIALERROR_NOERROR: kein Fehler SERIALERROR_OVERRUNERR: ? SERIALERROR_PARITYERR: Paritätsfehler SERIALERROR_FRAMEERR: ? SERIALERROR_BREAKERR: ? SERIALERROR_REVCBUFFERFULL: Empfangspuffer voll SERIALERROR_TRMBUFFERFULL: Sendepuffer voll
unsigned int **COM2TransmitBlock**(char* data_block, unsigned int length)	Sendet Daten über die serielle Schnittstelle COM2. **Parameter:** *data_block* ist Zeiger auf den Speicherbereich, der die Daten enthält; *length* gibt die Anzahl der Bytes an, die gesendet werden sollen. **Rückgabe:** Anzahl der gesendeten Bytes
unsigned int **COM2ReadBlock** (char* data_block, unsigned int buffersize)	Liest Daten aus der seriellen Schnittstelle COM2. **Parameter:** *data_block* ist Zeiger auf den Speicherbereich in den die Daten geschrieben werden; *buffersize* gibt die maximale Größe von *data_block* an. **Rückgabe:** Anzahl der gelesenen Bytes

Funktion	Beschreibung
void **LEDOn**(int LED)	Schaltet LEDs an. **Parameter:** LED gibt die Nummer der LED über den enum-Typ TLED an: LED1: LED1 wird eingeschaltet LED2: LED2 wird eingeschaltet **Anmerkung:** Damit die Funktion **LEDOn** arbeitet muss, zunächst die serielle Schnittstelle COM2 über die Funktion **COM2Init** initialisiert werden! Die Parameter (Baudrate, Datenbits, etc..) können dabei beliebig in den oben angegeben Grenzen variiert werden
void **LEDOff**(int LED)	Schaltet LEDs aus. **Parameter:** LED gibt die Nummer der LED über den enum-Typ TLED an: LED1: LED1 wird ausgeschaltet LED2 :LED2 wird ausgeschaltet **Anmerkung:** Damit die Funktion **LEDOn** arbeitet muss, zunächst die serielle Schnittstelle COM2 über die Funktion **COM2Init** initialisiert werden! Die Parameter (Baudrate, Datenbits, etc..) können dabei beliebig in den oben angegeben Grenzen variiert werden.

8.5 CAN-Bus

Nachfolgend werden die Routinen zur Ansteuerung für den CAN-Bus des DeviLAN-Moduls beschrieben (Tabelle 33). Weitere Erläuterungen zur Verwendung der Routinen befinden sich auch in der Projektdatei **examples.ide**, die den Quelltext **can_test.c** enthält, um das Beispielprogramms **can_test.exe** zu erzeugen.

Die Hardware unterstützt die beiden Varianten CAN 2.0A (11Bit Identifier) und CAN 2.0B (29 Bit Identifier), wobei die Daten in der Struktur TCANMsg gekapselt werden.

Die implementierte Treiberroutinen beinhalten CAN-Message-Puffer für Sende- und Empfangsbetrieb. Eingehende Meldungen werden durch den Treiber Interrupt-gesteuert in den Empfangspuffer eingelesen. Im Empfangspuffer können max. 64 Meldungen zwischengespeichert werden. Der Puffer muss durch die Anwendung daher rechtzeitig ausgelesen werden, damit kein Datenverlust auftritt.

Der Sendepuffer kann ebenfalls 64 CAN-Meldungen aufnehmen. Das Versenden der Meldungen erfolgt Interrupt-gesteuert durch den Treiber.

Tabelle 33 :Bibliotheksfunktionen für den CAN-Bus

Funktion	Beschreibung
unsigned char **CANInit(**unsigned int baudrate, int mode, unsigned long acceptance_mask unsigned long acceptance_code);	Initialisiert den CAN-Controller (SJA1000). Diese Funktion muss einmal aufgerufen werden, damit der Controller initialisiert wird und auf den CAN-Bus zugegriffen werden kann. **Parameter:** *can_bitrate* legt die Bitrate (Werte 5, 10, 20, 50, 100, 125, 250, 500 und 1000kBit/s) fest. *mode* legt den Modus über den enum-Typ TCANMode fest in dem der Controller betrieben werden soll: CAN_STANDARD = CAN2.0A-Protokoll, 11-Bit-Identifier CAN_EXTENDED = CAN2.0B-Protokoll, 29-Bit-Identifier *acceptance_mask* enthält den Wert, der in das Acceptance-Mask-Register geschrieben wird. Bit="0", Bit wird überprüft; Bit="1" (dont't care) Bit passiert den Filter immer *acceptance_code* enthält den Wert, der in das Acceptance-Code-Register geschrieben wird. Bit="0": Bit muss „0" sein, damit CAN-Message Filter passiert Bit="1", Bit muss „1" sein, damit CAN-Message Filter passiert **Ergebnis:** CAN_INIT_OK: alle Parameter im gütigen Bereich CAN_INIT_ERR_PARAMETER: einer oder meherere Parameter ungültig
unsigned char **CANInitBTR(**unsigned int btr1btr0, int mode, unsigned long acceptance_mask unsigned long acceptance_code);	siehe Funktion CANInit(), jedoch wird die Baudrate direkt über den Zugriff auf das BTR-Register des CAN-Controllers eingestellt. Dadurch sind auch Baudraten einstellbar, die nicht den vorgegeben Standardwerten entsprechen.
void **CANOn**(void);	Schaltet die Interrupt-Service-Routinen für den CAN-Bus ein, damit CAN-Daten gesendet und empfangen werden können.
void **CANOff**(void);	Schaltet die Interrupt-Service-Routinen für den CAN-Bus ab. CAN-Daten können weder gesendet noch empfangen werden.
unsigned char far **CANSendData**(TCANMsg msg);	Schreibt eine Meldung in den CAN-Bus-Softwaresendepuffer. Diese wird vom Sendemechanismus so schnell wie möglich auf den Bus gelegt. **Parameter:** *msg* enthält die CAN-Meldung in einer TCANMsg Datenstruktur. **Rückgabe:** 1, wenn Meldung in Sendepuffer geschrieben werden konnte; 0 sonst
int **CANTransmitData**(void);	Gibt die Anzahl der noch zu sendenden CAN-Meldungen im Softwaresendepuffer zurück

Funktion	Beschreibung
unsigned int **CANValidData**(void)	Überprüft ob Daten im Empfangspuffer verfügbar sind. Diese können dann über CANGetData ausgelesen werden. **Rückgabe:** Anzahl der CAN-Meldungen im Empfangspuffer
TCANMsg **CANGetData**(void)	Liest eine Meldung aus dem CAN-Bus-Empfangspuffer. **Rückgabe:** Die Funktionen liefert eine CAN-Meldung in einer TCANMsg-Datenstruktur.
unsigned char **CANGetStatus**(void)	Liest den Fehlerstatus des CAN-Bus. **Rückgabe:** CAN_ERR_OK, wenn kein Fehler vorliegt sonst wird Fehlerstatus mit folgenden Fehlerbits angegeben: CAN_ERR_XMTFULL: Sendepuffer des Controllers ist voll; diese Meldung kann ignoriert werden, da sie direkt vom Sendemechanismus verarbeitet wird. CAN_ERR_OVERRUN: Empfangspufferüberlauf CAN_ERR_BUSERROR: Fehlerzähler erreichte Limit CAN_ERR_BUSOFF: Busfehler, Controller ging 'Bus-Off' CAN_ERR_TRANSMITBUF_OVERFLOW: Bei einem Sendeversuch ist der Software-Sendepuffer übergelaufen. Die CAN-Meldung wurde nicht im Puffer abgelegt; ggf. Sendeversuch wiederholen. Die Fehlermeldung wird nach Aufruf von CANGetStatus wieder gelöscht. CAN_ERR_RECEIVEBUF_OVERFLOW: Überlauf des Software-Empfangspuffers, CAN-Meldungen gingen verloren, da Puffer nicht rechtzeitig ausgelesen wurde. Die Fehlermeldung wird nach Aufruf von CANGetStatus wieder gelöscht.

9 CAN2Web

9.1 Allgemeine Informationen

CAN2Web basiert auf unseren etablierten DeviLAN-Systemen mit einem leistungsstarken 16-Bit Prozessor und einen SJA1000-Controller von Philips.

CAN2Web verpackt die CAN-Messages in einen TCP/IP-Rahmen und überträgt diese Daten über das LAN. Hierzu muss nur die IP-Adresse der zweiten Moduls angegeben werden, um alles weitere kümmert sich CAN2Web automatisch. Durch den Einsatz von mehreren Modulen lassen sich bei Bedarf mehrere Netze miteinander koppeln (Abb. 19).

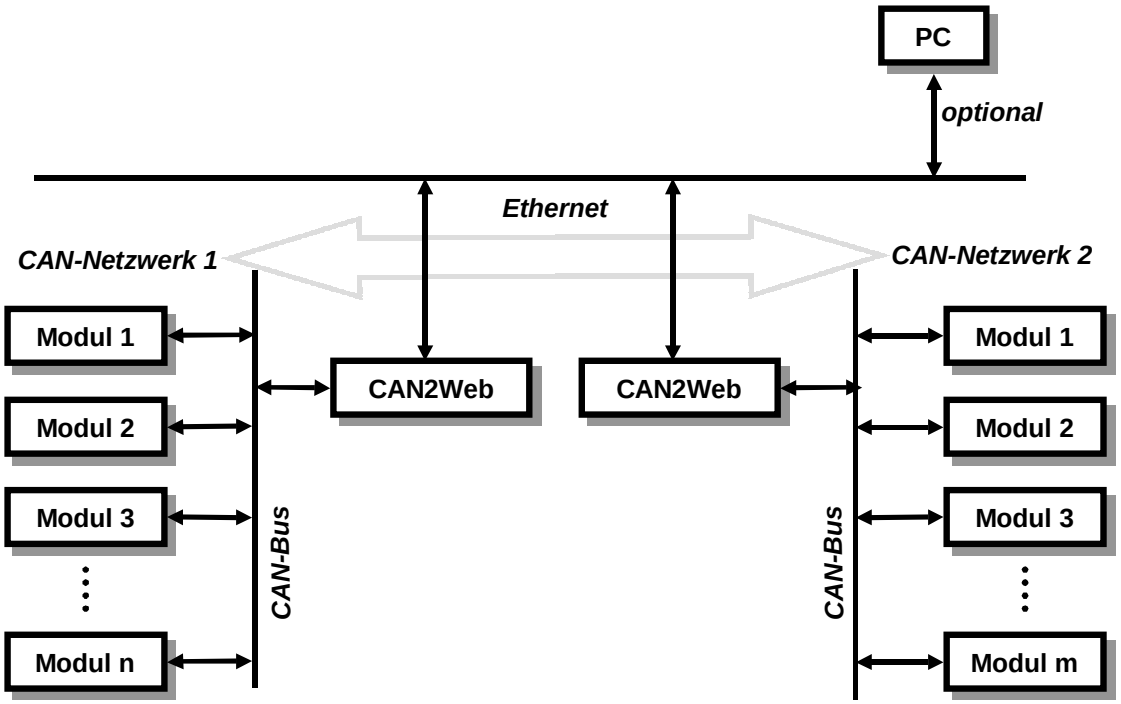


Abb. 19: Kopplung von CAN-Netzwerken

Die serielle Schnittstelle (RS232) ist wohl die weit verbreitetste Kommunikationsschnittstelle überhaupt. Die CAN2Web-Schnittstellenkoppler erlauben zusätzlich Geräte, Maschinen oder Sensoren mit serieller Schnittstelle an das Ethernet anzubinden und damit einen Zugriff auf Sie zu erhalten (Abb. 20).

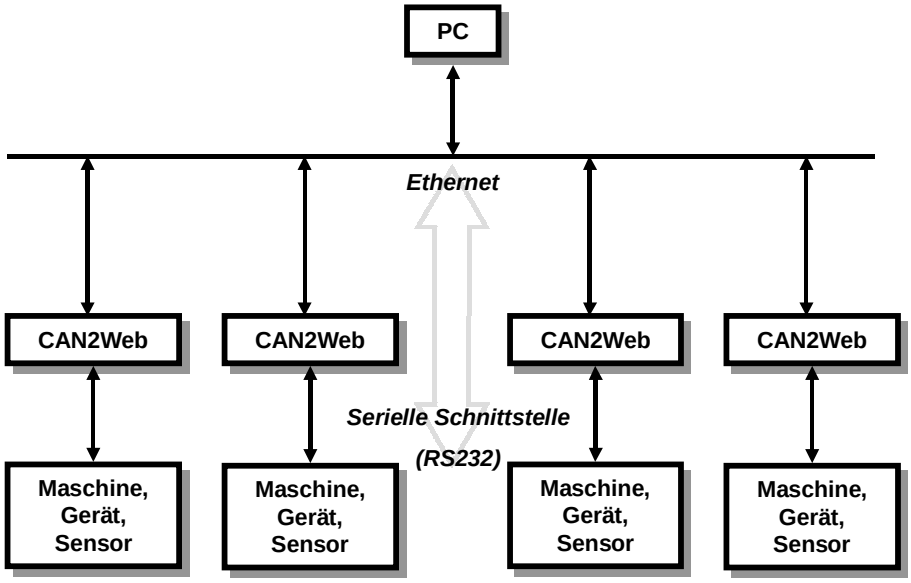


Abb. 20: PC-Anbindung von Maschinen, Geräten und Sensoren über Ethernet

9.2 Hardwareübersicht

Die Hardware des CAN2Web-Moduls besteht aus den Funktionsgruppen, die in Abb. 21 wiedergegeben werden. Die Steckerbelegung von CAN2Web stimmt mit denen von DeviLAN überein kann den folgenden Abschnitten entnommen werden:
- CAN-Bus (siehe Kap. 10.4), - COM1(siehe Kap. 10.5),
- RJ45 (siehe Kap. 10.6), - Spannungsversorgung (siehe Kap. 10.9)

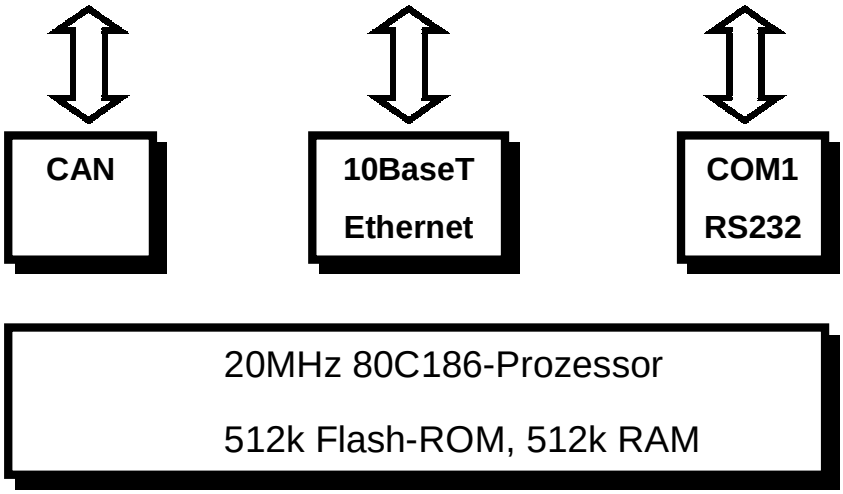


Abb. 21: Hardwareübersicht CAN2Web

Ein Kommandointerface gestattet einen vollständigen Zugriff auf alle Konfigurationseinstellungen und erlaubt, die Schnittstellen abzufragen.

9.3 Kommuniaktionsprotokoll

Die Befehle zur Konfiguration des CAN2Web-Moduls erfolgen analog zu DeviLAN-Modulen als ASCII-Kommandozeile. Der Aufbau von Kommandozeile ist in Kap. 6.3 beschrieben.

Das ASCII Protokoll ist für die Übertragung hoher Datenraten oft zu langsam. Daher wurde für CAN2Web-Module eine Kombination aus ASCII-Kommandozeile und binärer Kommunikation implementiert, wobei die über den CAN-Bus und die serielle Schnittstelle eingehenden Daten wahlweise als ASCII-Kommandozeile oder binär, und damit deutlich schneller, übertragen werden können.

Der Kommandointerpreter des Moduls arbeitet wie folgt:

Erkennt der Interpreter als erstes Zeichen ein ASCII-Zeichen, so geht er davon aus, dass es sich um eine ASCII-Kommandozeile handelt und verarbeitet sie wie in Kap. 6.3 beschrieben.

Wird als erstes Byte dagegen ein Zeichen >=0x80 empfangen so geht er davon, dass es sich um ein binäres Kommando handelt. Die Länge des Kommandos ergibt sich aus dem ersten Byte, das nachfolgend als Kennbyte bezeichnet wird, und ggf. in einer im Kommando integrierten Längenangabe.

Für die Anbindung von eigenen Applikationen ist es zwingend erforderlich die genauen Längenangaben für die binären Meldungen beim Senden und Empfangen einzuhalten.

9.4 Konfiguration der Software/Protokolle

In den nachfolgenden Abschnitten werden die Kommandos für CAN2Web beschrieben. Die Kommandos zur Konfiguration des CAN-Bus und der seriellen Schnittstelle sind analog zu DeviLAN-Modulen implementiert und können Kap. 6.4.5 und Kap. 6.4.3 entnommen werden.

9.4.1 Befehle für den Verbindungsmanager

Ein CAN2Web-Modul ist in der Lage bis zu drei bidirektionale Socketverbindungen parallel zu anderen Modulen bzw. Applikationen aufzubauen. Der Aufbau wird über die Befehle in Tabelle 34 gesteuert.

Tabelle 34 :Steuerung des Verbindungsmanager

Befehl	Wert/ Typ	Default	Funktionsbeschreibung
ip1_connect	0,1	0	Aktiviert/Deaktiviert den Verbindungsaufbau zu IP-Adresse 1 0: kein Verbindungsaufbau 1: Verbindungsaufbau erfolgt
ip1_str	IP	192.168.1.101	IP-Adresse 1 zu der eine Verbindung aufgebaut wird
ip1_port	INT	3333	Port 1 zum dem eine Verbindung wird
ip2_connect	0,1	0	Aktiviert/Deaktiviert den Verbindungsaufbau zu IP-Adresse 2 0: kein Verbindungsaufbau 1: Verbindungsaufbau erfolgt
ip2_str	IP	192.168.1.102	IP-Adresse 2 zu der eine Verbindung aufgebaut wird
ip2_port	INT	3333	Port 2 zum dem eine Verbindung wird
ip3_connect	0,1	0	Aktiviert/Deaktiviert den Verbindungsaufbau zu IP-Adresse 3 0: kein Verbindungsaufbau 1: Verbindungsaufbau erfolgt
ip3_str	IP	192.168.1.103	IP-Adresse 3 zu der eine Verbindung aufgebaut wird
ip3_port	INT	3333	Port 3 zum dem eine Verbindung wird

Ist die Verbindung aufgebaut werden CAN und COM-Daten an die verbundenen Module/Applikationen weitergeleitet.

Das Modul sendet zusätzlich zyklisch Informationen (Tabelle 35) zum Zustand/Status der Verbindung.

Tabelle 35 :Status der Socket-Verbindungen

Befehl	Wert/ Typ	Default	Funktionsbeschreibung
ipstate	0,1,2 0,1,2 0,1,2	0 , 0, 0	0: Verbindung abgebaut/keine Verbindung 1: Verbindung wird aufgebaut 2: Verbindung ist aufgebaut

Beispiel 10 : Socketzustandsmeldung

```
ip_state 0 2 1 ; Verbindung zu IP-Adresse 1 ist abgebaut
               ; Verbindung zu IP-Adresse 2 ist aufgebaut
               ; Verbindung zu IP-Adresse 3 wird aufgebaut
```

9.4.2 Umschaltung zwischen ASCII und binärer/schneller Kommunikation

Die Umschaltung zwischen ASCII-Kommandozeilen und binärer Kommunikation erfolgt über die Befehle in Tabelle 36.

Tabelle 36 :Umschaltbefehle auf schnelle (binäre) Socket-Kommuniaktion für CAN/COM

Befehl	Wert/Typ	Default	Funktionsbeschreibung
can_fast	0,1	0	Aktiviert/Deaktiviert die schnelle Übertragung von CAN-Daten über die Socketverbindung 0: ASCII-Kommandozeile 1: binäre, schnelle Kommunikation
com_fast	0,1	0	Aktiviert/Deaktiviert die schnelle Übertragung von COM-Daten über die Socketverbindung 0: ASCII-Kommandozeile 1: binäre, schnelle Kommunikation

9.5 Befehle für schnelle/binäre Socketkommunikation

Es sind folgende Kennbytes implementiert (Tabelle 37):

Tabelle 37 :Kennbytes für binäre Kommunikation

Kennbyte	Funktionsbeschreibung
0x81	CAN-Meldung inkl. Fehlermeldung
0x82	CAN-Fehlermeldung Anmerkung: Empfängt ein CAN2Web eine Meldung 0x82 so wird diese von ihm ignoriert.
0x91	Übertragung von Daten von der seriellen Schnittstelle COM1

Nachfolgend erfolgt eine Beschreibung der Meldungen.

9.5.1 Datenrahmen für binäre CAN-Meldung, Kennbyte 0x81

Empfängt ein CAN2Web-Modul ein Datenrahmen gemäß Tabelle 38 über eine Socketverbindung, so legt es die Meldung umgehend auf den CAN-Bus. Empfängt es eine Nachricht über den CAN-Bus sendet es diese ebenfalls gemäß Tabelle 38 über alle Socketverbindungen, wenn schnelle CAN-Kommunikation voreingestellt wurde.

Tabelle 38 :CAN-Meldung inkl. Status/Fehlerinformation

Byte	Wert/ Inhalt	Beschreibung
0	0x81	Kennbyte für schnelle CAN-Meldung inkl. Status/Fehlerinformation des CAN-Bus
1	<ERROR>	CAN-Status/Fehlerbyte
2	<CAN-ID3>	CAN-ID Byte 3
3	<CAN-ID2>	CAN-ID Byte 2
4	<CAN-ID1>	CAN-ID Byte 1
5	<CAN-ID0>	CAN-ID Byte 0

Byte	Wert/ Inhalt	Beschreibung
6	<RTR+LEN>	Angabe zu RTR-Bit und Anzahl der Datenbytes a) das MSB repräsentiert das RTR-Bit b) die unteren 4 Bit geben die Anzahl der Datenbyte der CAN-Meldung an
7..15	[>DATA 0...7>]	Datenbyte 0...7 der CAN-Meldung (optional)

Die Gesamtlänge der Meldung in Byte beträgt 7 + LEN.

Soll beispielsweise folgende Meldung auf den CAN-Bus gelegt werden:

ID : 0x03A2A3FE ; 29-Bit-Identifier
RTR : 0
8 Datenbyte: 0x01, 0x02, 0x03, 0x04, 0x80, 0x70, 0x60, 0x40

so sind über die Socketverbindung die 15 folgenden Bytes zu senden:

Nr. : Byte ; Beschreibung

0 : 0x81 ; Kennbyte für schnelle CAN-Meldung
1 : XX ; Fehlerbyte beliebig, da es von CAN2Web ignoriert wird
2 : 0x03 ; ID3
3 : 0xA2 ; ID2
4 : 0xA3 ; ID1
5 : 0xFE ; ID0
7 : 0x01 ; Datenbyte 0
8 : 0x02 ; Datenbyte 1
9 : 0x03 ; Datenbyte 2
10 : 0x04 ; Datenbyte 3
11 : 0x80 ; Datenbyte 4
12 : 0x70 ; Datenbyte 5
13 : 0x60 ; Datenbyte 6
14 : 0x40 ; Datenbyte 7

9.5.2 Datenrahmen für binäre CAN-Status/Fehlermeldung, Kennbyte 0x82

Tritt auf dem CAN-Bus eine Änderung des Fehlerzustandes auf, sendet es gemäß Tabelle 40 über alle Socketverbindungen eine Fehlermeldung, wenn schnelle CAN-Kommunikation voreingestellt wurde.

Tabelle 39 :CAN- Fehlermeldung

Byte	Wert/ Inhalt	Beschreibung
0	0x82	Kennbyte für schnelle CAN-Fehlermeldung
1	<ERROR>	CAN-Bus Fehlerstatus, bitweise maskiert CAN_ERR_OK, wenn kein Fehler vorliegt sonst wird Fehlerstatus mit folgenden Fehlerbits angegeben: CAN_ERR_XMTFULL: Sendepuffer des Controllers ist voll; diese Meldung kann ignoriert werden, da sie direkt vom Sendemechanismus verarbeitet wird. CAN_ERR_OVERRUN: Empfangspufferüberlauf CAN_ERR_BUSERROR: Fehlerzähler erreichte Limit CAN_ERR_BUSOFF: Busfehler, Controller ging 'Bus-Off' CAN_ERR_TRANSMITBUF_OVERFLOW: Bei einem Sendeversuch ist der Software-Sendepuffer übergelaufen. Die CAN-Meldung wurde nicht im Puffer abgelegt; ggf. Sendeversuch wiederholen. CAN_ERR_RECEIVEBUF_OVERFLOW: Überlauf des Software-Empfangspuffers, CAN-Meldungen gingen verloren, da Puffer nicht rechtzeitig ausgelesen wurde.

Die Meldung hat eine feste Gesamtlänge von 2 Byte. Empfängt ein CAN2Web eine Meldung 0x82 so wird diese von ihm ignoriert.

9.5.3 Datenrahmen für binäre COM-Meldung

Empfängt ein CAN2Web-Modul ein Datenrahmen gemäß Tabelle 40 über eine Socketverbindung, so legt es die Meldung umgehend auf den CAN-Bus. Empfängt es Daten über die serielle Schnittstelle sendet es diese ebenfalls gemäß Tabelle 40 über alle Sokketverbindungen, wenn schnelle COM-Kommunikation voreingestellt wurde.

Tabelle 40 :COM-Meldung

Byte	Wert/	Beschreibung
0	0x91	Kennbyte für schnelle COM-Meldung
1	<LEN>	Anzahl der nachfolgenden Datenbyte der seriellen Schnittstelle COM1, max. 255 Byte
2..256	[<DATA>]	Datenbyte, optional

Die Gesamtlänge der Meldung in Byte beträgt 2 + <LEN>.

Soll beispielsweise der Text „<STX>Hallo<ETX>“ über die serielle Schnittstelle ausgegeben werden, so sind über die Socketverbindung die 9 folgenden Bytes zu senden:

0 : 0x91 ; Kennbyte für schnelle COM-Meldung
1 : 0x05 ; Anzahl der Datenbyte
2 : 0x02 ; Start of Text, Textanfang
3 : 0x48 ; Buchstabe ’H’

4 : 0x61 ; Buchstabe ’a
5 : 0x6C ; Buchstabe ’l’
6 : 0x6C ; Buchstabe ’l’
7 : 0x6F ; Buchstabe ’o’
8 : 0x03 ; End of Text, Textende

9.6 Konfiguration über DeviLANControl

Für die Bedienung und Konfiguration von CAN2Web-Modulen stellt das Windows Programm DeviLANControl die beiden Dialogboxen „Konfiguration“ und „CAN&COM“ zur Verfügung. Diese erscheinen nach Verbindungsaufbau automatisch.

9.6.1 Die Dialogbox Konfiguration

Die Dokumentation ist momentan leider noch nicht fertig gestellt!

9.6.2 Die Dialogbox CAN&COM

Die Dokumentation ist momentan leider noch nicht fertig gestellt!

9.6.3 Anmerkungen zu DeviLANControl

a) In der jetzigen Version von DeviLANControl ist es testweise möglich binäre Meldungen an COM1 zu übertragen.

Zur Übertragung an die COM1-Schnittstelle müssen die Daten in hexadezimaler Form in das Texteingabefeld eingetragen werden (max. 30 Zeichen). Über „Senden binär“ werden die Daten binär an CAN2Web übertragen und auf die serielle Schnittstelle gelegt.

Soll beispielsweise der Text „<STX>Hallo<ETX>“ über die serielle Schnittstelle ausgegeben werden, so sind die 7 folgenden Bytes einzugeben.

Eingabe: 02 48 61 6C 6C 6F 03

0 : 0x02 ; Start of Text, Textanfang
1 : 0x48 ; Buchstabe ’H’
2 : 0x61 ; Buchstabe ’a
3 : 0x6C ; Buchstabe ’l’
4 : 0x6C ; Buchstabe ’l’
5 : 0x6F ; Buchstabe ’o’
6 : 0x03 ; End of Text, Textende

b) DeviLANControl kann binäre Meldungen verarbeiten und entsprechend in konvertierter Form als Zeichenkette ausgeben. Als Kennzeichnung, dass die ursprüngliche Mel-

dung in binärer Form vorlag setzt die Ausgabefunktion (z. B. im Dialog „Socket“) ein Ausrufezeichen vor die Meldung.

Beispiel 11 : Anzeige binärer Meldungen in Dialogbox „Socket“

```
!can_msg 32AE3 0 4 01 02 03 04 ; Ausgabe von binärem CAN-Telegramm mit 4
                               ; Datenbyte, RTR-Bit=0
!can_error 00                  ; Ausgabe von binärem CAN-Fehlermeldung
!com1 12 AE 36 06              ; hex. Ausgabe der binär empfangenen
                               ; Datenbytes der seiellen Schnittstelle
```

10 Technische Daten

10.1 Elektrische Daten

Symbol	Parameter	Limit Min	Limit Typ	Limit Max	Units	Conditions
V_{CC}	Supply Voltage **DeviLAN-Basic**	4,8		5,2	V	
I_{CC}	Supply Current **DeviLAN-Basic**		280		mA	
V_{IL}	Logical Zero Input Voltage			0,8	V	Port A, Port C, COM2
V_{CC}	Supply Voltage **DeviLAN-24**	8	24	30	V	
I_{CC}	Supply Current **DeviLAN-24**		70		mA	
V_{IH}	Logical One Input Voltage	2,2			V	Port A, Port C, COM2
V_{OL}	Logical Zero Output Voltage			0,4	V	I_{OL} = 2,5mA
V_{OH}	Logical One Output Voltage	2,4			V	I_{OH} = -2,5mA
V_{DSAT}	Driver Satuation Voltage			1,3	V	I_{OL} = 200mA, Port B
V_{HIL}	High Level Zero Input Voltage			5,2	V	Port D
V_{HIH}	High Level One Input Voltage	18,4			V	Port D
I_{BHL}	Bus Hold Low Current	50		400	µA	Port A, Port C
I_{BHH}	Bus Hold High Current	-50		-400	µA	Port A, Port C
I_{LK}	Input Leakage Current	-10		10	µA	COM int
R_{HI}	High Level Input Resistance	36	43	50	kΩ	Port D
V_{Ref}	Reference Output Voltage	2,4	2,5	2,6	V	
	Reference Drift		30		ppm/° C	
I_{Ref}	Reference Output Current			1	mA	
V_{Bias}	Bias Output Voltage	3,15	3,3	3,45	V	
	Bias Drift		50		ppm/° C	
I_{Bias}	Bias Load Current			8	mA	
V_{In}	Analog Voltage Range	0		5	V	must not exeed V_{CC}
	Analog Input Noise		8		ppm	
	ADC Effective Resolution		20		bits	f_{Data}=50Hz, no missing codes
	ADC Integral Linearity			± 0,0015	%	f_{Data} = 50Hz, Full scale Range
f_{Data}	ADC Sampling Rate	3		1000	Hz	
CMRR	ADC Common Mode Rejection	160			dB	f_{Data} = 50Hz
PSRR	ADC Power Supply Recetion	60			V	f_{Data} = 50Hz
$V_{DAC,max}$		4.079	4,096	4,111	V	
	DAC Relative Accuracy	-2	± 1/2	+2	LSB	
I_{DAC}	DAC Output Current	± 5	± 7		mA	V_{DAC} = 2,048V
C_{Load}	DAC Capacitive Load		500		pF	

10.2 Sonstige Daten

DeviLAN-Basic & DeviLAN-24:

Tabelle 41 :Umgebungsbedingungen für DeviLAN-Module

Parameter	Wertebereich/Anmerkung
Betriebstemperatur	0 °C..55 °C (Gehäusetemperatur des SC12-Prozessors!)
Lagertemperatur	-25 °C ... 70 ° C
Luftfeuchtigkeit	10..85%, nicht kondensierend

DeviLAN-Basic:

Tabelle 42 :Sonstige Angaben für DeviLAN-Basic

Parameter	Wert(e)
Abmessungen	ca. 96mm*100mm*20mm (Länge*Breite*Höhe)
Gewicht	ca. 120g

DeviLAN-24:

Tabelle 43 :Sonstige Angaben für DeviLAN-24

Parameter	Wert(e)
Abmessungen	ca. 102mm*100mm*20mm (Länge*Breite*Höhe)
Gewicht	ca. 120g

10.3 Anschlussbelegung

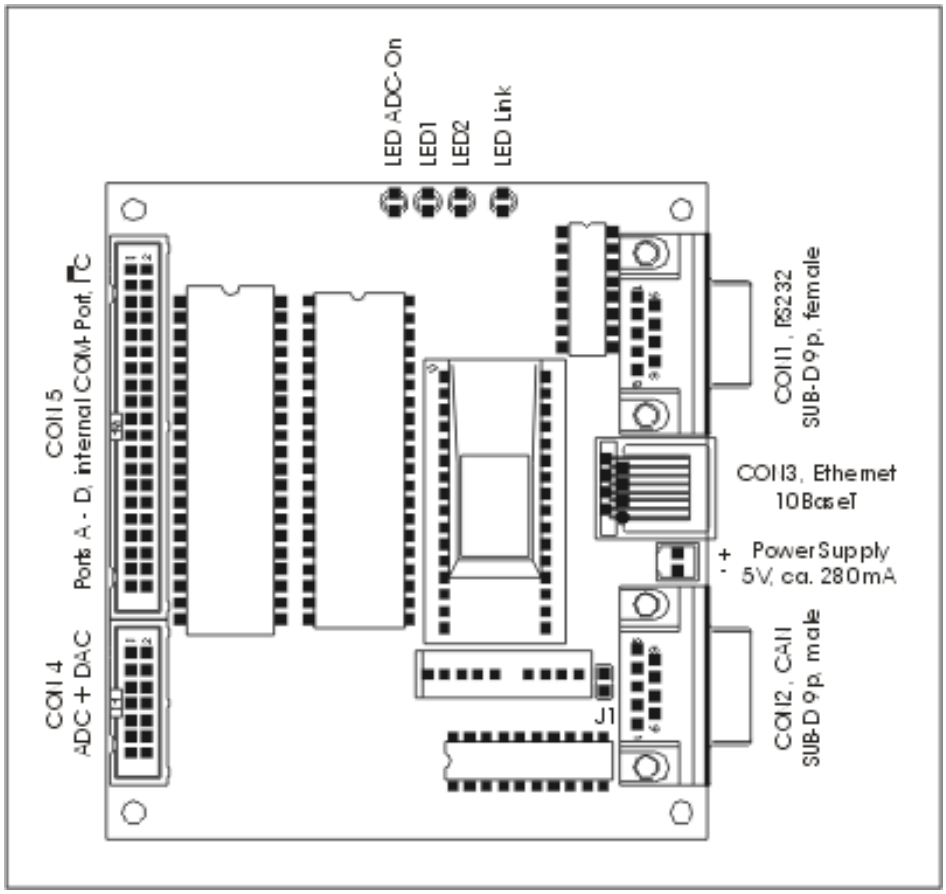


Abb. 22: Steckverbinder DeviLAN

10.4 Pinbelegung CON1 (RS232C, DCE, 9p female)

Pin	I/O	Name	Description
1	Out	--	always Low (+8V)
2	Out	RxD	max. 115 kBaud
3	In	TxD	max. 115 kBaud
4	In	DTR	
5		GND	
6	Out	DSR	always Low (+8V)
7	In	RTS	max. 115 kBaud
8	Out	CTS	max. 115 kBaud
9	Out	RI	always High (-8V)

10.5 Pinbelegung CON2 (CAN-Bus, 9p male)

Pin	I/O	Name	Description
1	--	NC	--
2	I/O	CANL	max. 1 MBit/s
3	--	GND	
4	--	NC	--
5	--	NC	--
6	--	NC	--
7	I/O	CANH	max. 1 MBit/s
8	--	NC	--
9	--	NC	--

10.6 Pinbelegung CON3 (Ethernet 10BaseT, 8p RJ45)

Pin	I/O	Name	Description
1	Out	Tx+	10BaseT Transmit, max. 100m
2	Out	Tx-	10BaseT Transmit, max. 100m
3	In	Rx+	10BaseT Receive, max. 100m
4	--	NC	--
5	--	NC	--
6	In	Rx-	10BaseT Receive, max. 100m
7	--	NC	--
8	--	NC	--

10.7 Pinbelegung CON4 (Analog I/O, 14p)

Pin	I/O	Name	Description
1	--	GND	--
2	--	GND	--
3	Out	DAC Channel 2	$F_{Data,Max.}$ = 200 Hz
4	Out	DAC Channel 1	$F_{Data,Max.}$ = 200 Hz
5	Out	Reference	$V_{Ref\ out}$ 2,5V typ.
6	Out	Bias	$V_{Bias\ out}$ 3,3V typ.
7	In	AIn3+	differential input, Channel 3, positive input
8	In	AIn3-	differential input, Channel 3, negative input
9	In	AIn2+	differential input, Channel 2, positive input
10	In	AIn2-	differential input, Channel 2, negative input
11	In	AIn1+	differential input, Channel 1, positive input
12	In	AIn1-	differential input, Channel 1, negative input
13	In	AIn0+	differential input, Channel 0, positive input
14	In	AIn0-	differential input, Channel 0, negative input

10.8 Pinbelegung CON5 (Digital I/O, 40p)

Pin	I/O	Name	Description
1	--	GND	--
2	--	GND	--
3	High Voltage In	D.6	$V_{In,max.}$ = 30V, Port D
4	High Voltage In	D.5	$V_{In,max.}$ = 30V, Port D
5	High Voltage In	D.4	$V_{In,max.}$ = 30V, Port D
6	High Voltage In	D.3	$V_{In,max.}$ = 30V, Port D
7	High Voltage In	D.2	$V_{In,max.}$ = 30V, Port D
8	High Voltage In	D.1	$V_{In,max.}$ = 30V, Port D
9	High Voltage In	D.0	$V_{In,max.}$ = 30V, Port D
10	I/O	SCK	I^2C-Interface, Clock, Master
11	I/O	SDA	I^2C-Interface, Data, Master
12	In	CTS	TTL, Internal COM-Port, High Speed
13	Out	TxD	TTL, Internal COM-Port, High Speed
14	In	RTS	TTL, Internal COM-Port, High Speed
15	Out	RxD	TTL, Internal COM-Port, High Speed
16	I/O	A.1	TTL I/O, Port A
17	I/O	A.0	TTL I/O, Port A
18	I/O	A.3	TTL I/O, Port A
19	I/O	A.2	TTL I/O, Port A
20	I/O	A.5	TTL I/O, Port A
21	I/O	A.4	TTL I/O, Port A
22	I/O	A.7	TTL I/O, Port A
23	I/O	A.6	TTL I/O, Port A
24	I/O	C.1	TTL I/O, Port C
25	I/O	C.0	TTL I/O, Port C
26	I/O	C.3	TTL I/O, Port C
27	I/O	C.2	TTL I/O, Port C
28	I/O	C.5	TTL I/O, Port C
29	I/O	C.4	TTL I/O, Port C
30	I/O	C.7	TTL I/O, Port C
31	I/O	C.6	TTL I/O, Port C
32	Test	Test	do not connect
33	Out	B.0	I_{max} = 200mA, Driver Port B, Open Collector
34	Out	B.1	I_{max} = 200mA, Driver Port B, Open Collector
35	Out	B.2	I_{max} = 200mA, Driver Port B, Open Collector

Pin	I/O	Name	Description
36	Out	B.3	I_{max} = 200mA, Driver Port B, Open Collector
37	Out	B.4	I_{max} = 200mA, Driver Port B, Open Collector
38	Out	B.5	I_{max} = 200mA, Driver Port B, Open Collector
39	Out	B.6	I_{max} = 200mA, Driver Port B, Open Collector
40	--	+5V	--

10.9 Pinbelegung CON6 (Power Supply, 2p)

Pin	I/O	Name	Description
1	In	Supply +	5V für DeviLAN-Basic 8V..30V für DeviLAN-24
2	In	Supply -	0V (GND)

(c) synertronixx GmbH, September 2004, Änderungen vorbehalten!