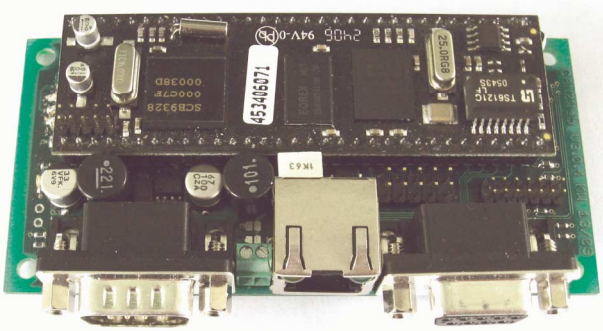
CAN2Web

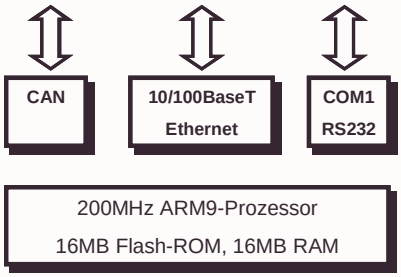
CAN-Ethernet-Gateway

Handbuch CAN2WEB-Serie

(Version vom 16.12.2011)

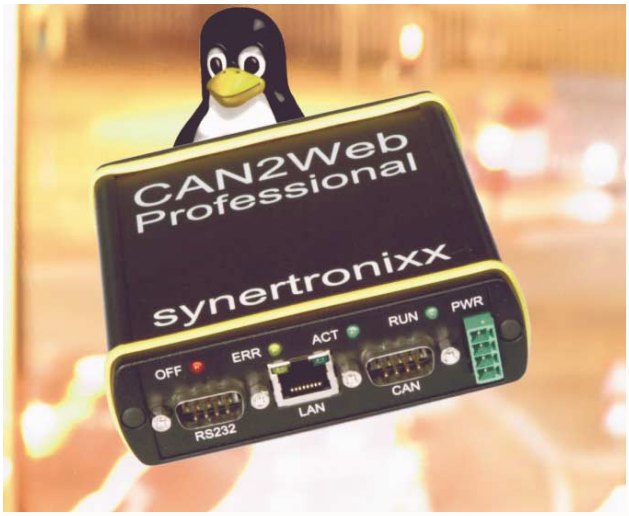
CAN2Web-Advanced

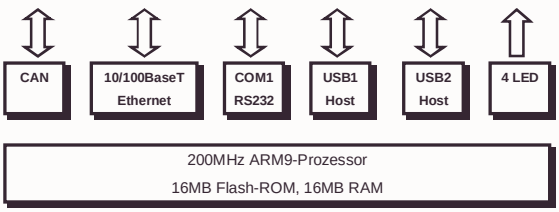




- **200MHz ARM9-Prozessor**
- **10/100BaseT Ethernet-Interface**
- **16MB Flash-ROM, 16MB RAM**
- **1 MBit CAN-Bus**
- **1 serielle RS232-Schnittstelle**
- **1 serielle RS232-Debug-Schnittstelle**
- **Beispielprogramme**
- **LINUX 2.6.31**
- **LINUX Kernel Treiber für CAN**
- **PC-Programm DeviLANControl für Netzwerkbetrieb (MS Windows)**
- **Einfache Installation durch Plug&Play**
- **WLAN und 512MB NAND Flash-Disk (optional)**

CAN2Web-Professional





- **200MHz ARM9-Prozessor**
- **10/100BaseT Ethernet-Interface**
- **16MB Flash-ROM, 16MB RAM**
- **1 MBit CAN-Bus (galvanisch getrennt)**
- **1 serielle RS232-Debug-Schnittstelle**
- **2 USB-Host-Schnittstellen**
- **4 Status/Debug-LEDs**
- **Beispielprogramme**
- **LINUX 2.6.31**
- **LINUX Kernel Treiber für CAN**
- **PC-Programm DeviLANControl für Netzwerkbetrieb (MS Windows)**
- **Einfache Installation durch Plug&Play**
- **WLAN und 512MB NAND Flash-Disk (optional)**

CAN2Web

CAN-Ethernet-Gateway

1 Wesentliche Merkmale

1.1 Einleitung

Die Module der CAN2Web-Serie sind CAN-Ethernet-Gateways, die dafür entwickelt wurden auf einfachste Art und Weise bestehende CAN-Komponenten und Geräte an das Intra- und Internet anzubinden. Die Einsatzgebiete liegen typischerweise im Bereich der Industrieautomation für mess-, steuerungs- und regelungstechnische Anwendungen.

Die CAN2Web-Module CAN2Web-Professional und CAN2Web-Advanced unterscheiden sich von der Funktionalität nur leicht und werden daher in einen gemeinsamen Handbuch beschrieben. CAN2Web-Module basieren auf unserem etablierten Socket-Computer der SCB932x-Familie mit einem leistungsstarken 32-Bit ARM9-Prozessor und einen SJA1000-CAN-Controller von Philips sowie Linux als Betriebssystem.

Weitere Informationen zum SCB932x finden Sie auch auf unseren Webseiten unter http://www.synertronixx.de/produkte/produkte_socketcomputer.htm.

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. 1).

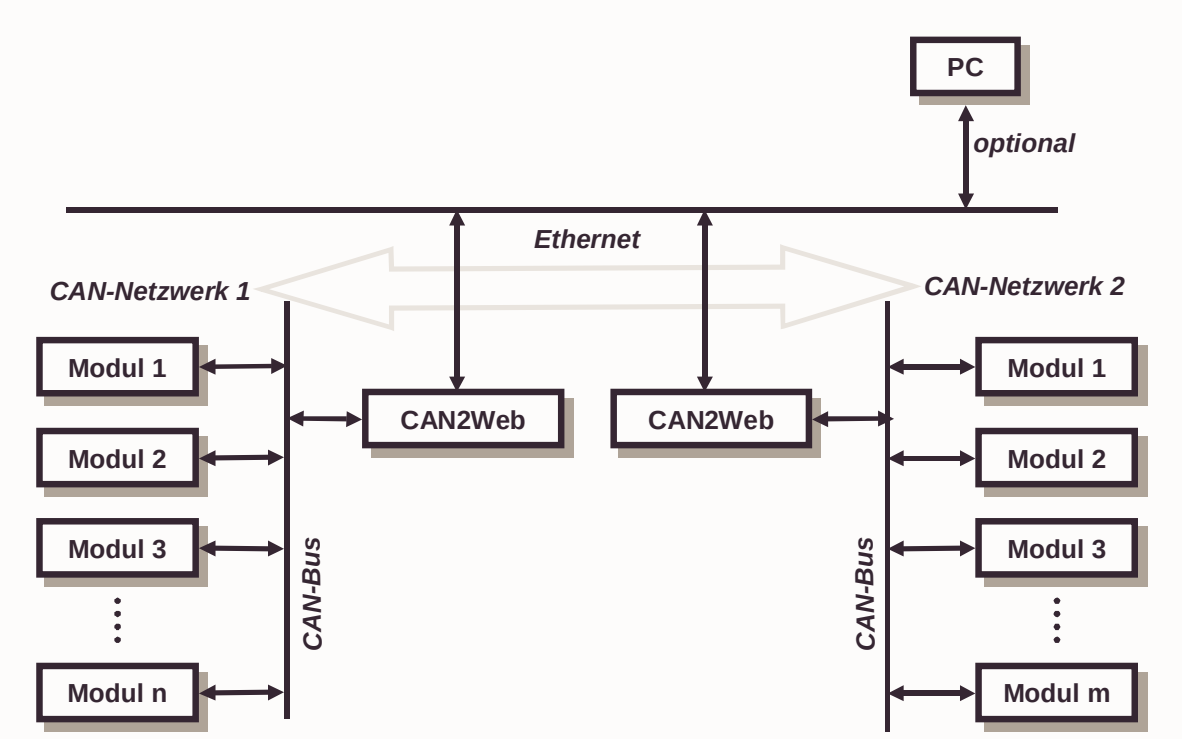


Abb. 1: Kopplung von CAN-Netzwerken

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. 2).

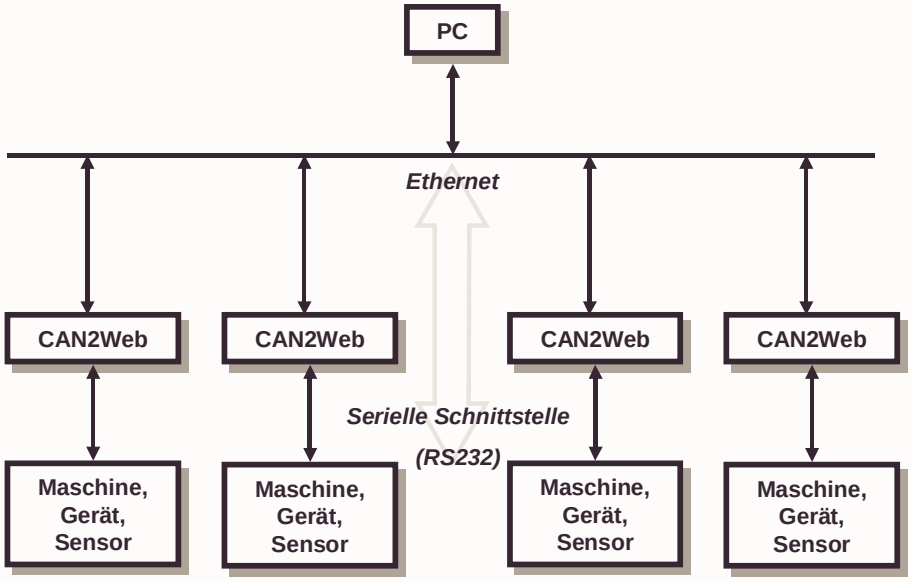


Abb. 2: PC-Anbindung von Maschinen, Geräten und Sensoren über Ethernet

1.2 Hardwareübersicht

1.2.1 Hardwareübersicht Advanced

Die Hardware des CAN2Web-Advanced-Moduls besteht aus den Funktionsgruppen, die in Abb. 3 wiedergegeben werden. Die Steckerbelegung des Moduls kann Kap. 9.2 entnommen werden.

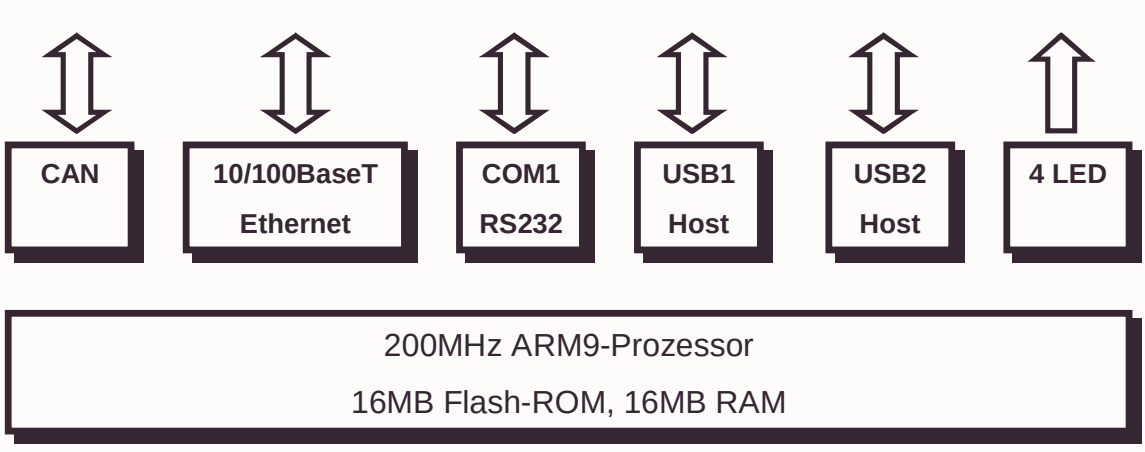


Abb. 3: Hardwareübersicht CAN2Web-Advanced

Durch On-Board-Integration aller notwendigen Anschlussverbinder kann das CAN2Web-Advanced-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.

1.2.2 Hardwareübersicht CAN2Web-Professional

Die Hardware des CAN2Web-Professional-Moduls besteht aus den Funktionsgruppen, die in Abb. 4 wiedergegeben werden. Die Steckerbelegung des Moduls kann Kap. 10.2 entnommen werden.

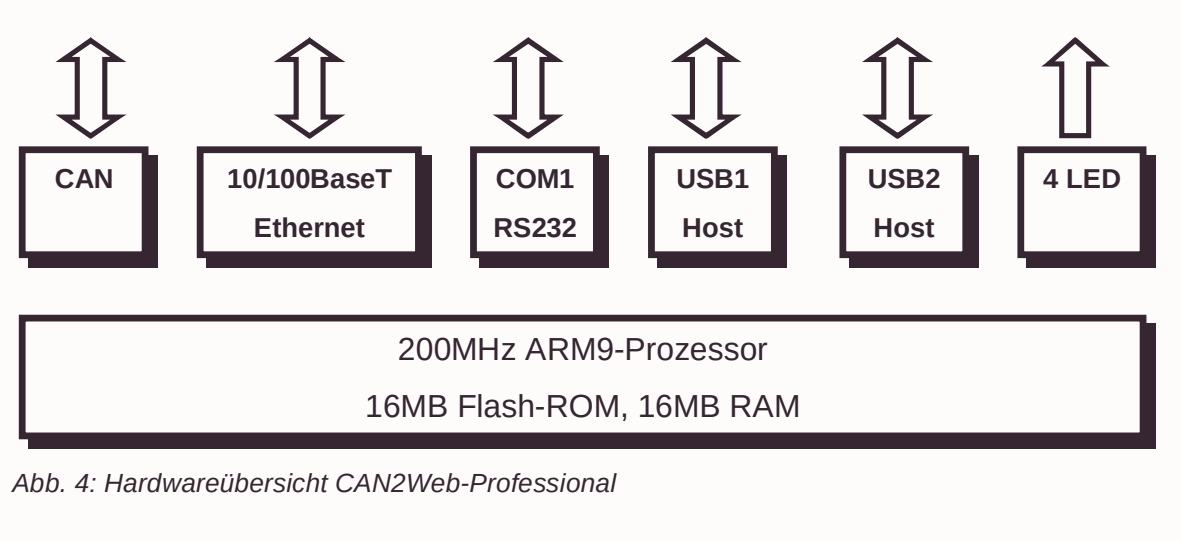


Abb. 4: Hardwareübersicht CAN2Web-Professional

Durch On-Board-Integration aller notwendigen Anschlussverbinder kann das CAN2Web-Professional-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. Der CAN-Bus ist galvanisch getrennt ausgeführt. Zwei USB-Stecker (Host-Schnittstellen) gestatten die Erweiterung des Moduls beispielsweise um USB-WLAN-Adapter oder USB-Memory-Sticks. Das Modul ist in einen robusten Metallgehäuse eingebaut.

1.3 Softwareübersicht

Als Betriebssystem kommt Linux in der Version 2.6.x zum Einsatz. Die mitgelieferte Beispiel- und Testsoftware (can2web.exe) erfüllt die Funktionalität eines CAN-Ethernet-Gateways. Standardmäßig kann jedes CAN2Web-Modul mittels eines Webbrowsers vollständig konfiguriert und bedient werden. Ebenfalls gestattet ein Kommandointerface einen vollständigen Zugriff auf alle Konfigurationseinstellungen und erlaubt es, die Schnittstellen abzufragen. Weiterhin verfügt es über mehrere Datenstrom-Schnittstellen, die es erlauben, Daten kontinuierlich über Socket-Verbindungen an weitere Applikationen zu senden.

Das CAN2Web-Modul wird mit Linux-Kernel-Treibern für den CAN-Bus ausgeliefert, die es ermöglichen, eigene Softwareapplikationen für das Modul zu entwickeln. Als Entwicklungsumgebung kann dabei z. B. der GNU C-Compiler zum Einsatz kommen. Ein Beispielprogramm, welches die Verwendung des Kernel-Treibers erläutert, ist beigefügt.

Als weitere Applikation steht das PC-Programm DeviLANControl für MS Windows zur Verfügung. Mit ihm lassen sich die CAN2Web-Module ebenfalls bedienen und konfigurieren.

2 Inbetriebnahme

2.1 Installation

Die Installation eines CAN2Web-Moduls erfolgt in drei Schritten:

1. Anbindung des Moduls an eine stabilisierte Gleichspannungsquelle (8 V...30V, Leistungsaufnahme typ. 3W). Die maximale Leistungsaufnahme ist von der angebundenen Hardware abhängig.

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

3. Anbindung an das Rechnernetzwerk (10/100 MBit) oder Verbinden mit einem lokalen PC über die serielle Schnittstelle (RS232).

Nach dem Einschalten ist das Modul nach ca. 10 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 vorsichtshalber nie im laufenden Betrieb an/ab.

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. Firefox oder Internet Explorer

2. Aufruf der CAN2Web-Startseite durch Eingabe der IP-Adresse in das Adressfeld des Browsers

3. Es erscheint die CAN2Web-Startseite über die die verschiedenen Webseiten für die Konfiguration und Datenabfrage angewählt werden können (Abb. 5). 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 CAN2Web-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:

   Konfiguration der COM und CAN-Schnittstelle(n), Aktivieren und Deaktivieren der einzelnen Schnittstellen, empfangenen von CAN-Meldungen sowie der empfangenen Daten der seriellen Schnittstelle, Senden von Daten an die serielle Schnittstelle und den CAN-Bus.

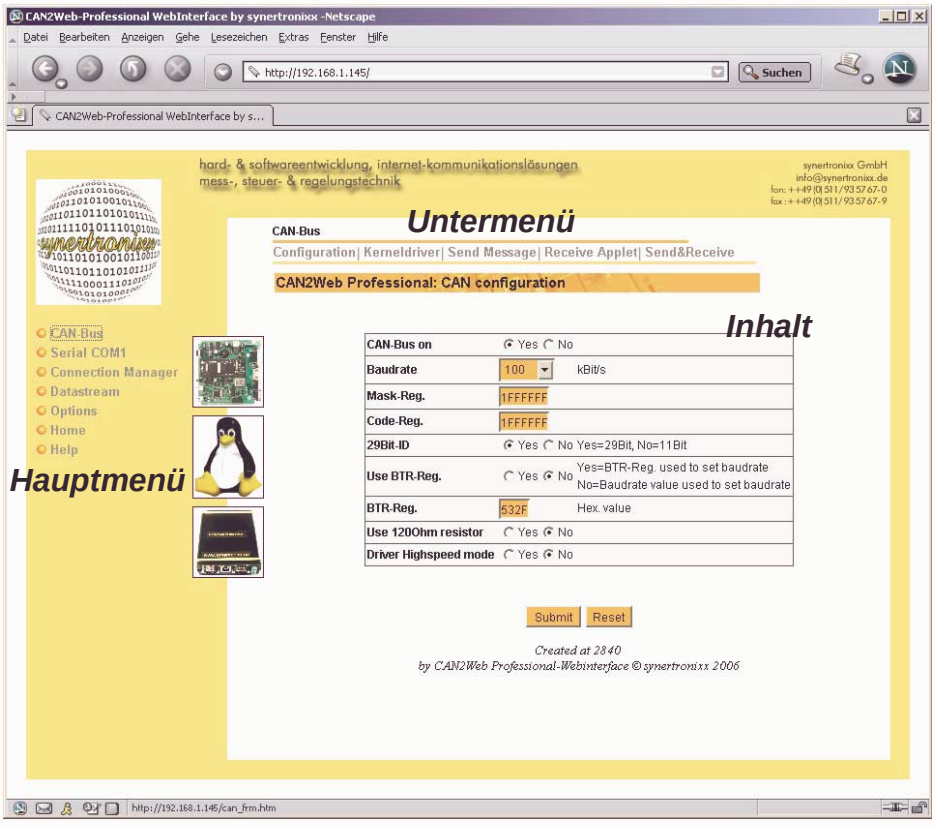


Abb. 5: CAN2Web Menüführung und Bedienung über Webbrowser

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 CAN2Web-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 aktuellen Einstellungen werden in einer Konfigurationsdatei des Moduls gespeichert und bei Start des Programms automatisch geladen.

Der Zugriff auf die Schnittstelle erfolgt ebenfalls in drei Schritten:

1. Ändern der gewünschten Einstellungen der ausgewählten Schnittstelle (CAN, COM1) 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 CAN2Web-Modul übertragen und entsprechend geändert.

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

3.1 Applet zur Anzeige von CAN-Meldungen

Zur Anzeige von CAN-Meldungen in einem Browser wurde zusätzlich ein Applet programmiert. Dieses baut nach dem Laden automatisch eine TCP/IP-Verbindung zum CAN2Web-Modul auf und zeigt die eingehenden CAN-Meldungen und Fehlerzustände an.

Anmerkung: Da das Applet unter Java läuft, ist die Ausführungsgeschwindigkeit relativ langsam. Bei hohen Datenvolumen und/oder bei PCs mit geringerer Rechenleistung kann es zu Problemen mit der Datenverarbeitung und Anzeige kommen. Das Applet ist hier nur zu Testzwecken zu verwenden.

4 Bedienung über DeviLANControl

Das Programm DeviLANControl dient als Kommunikations- und Konfigurationsprogramm für diverse Produkte von synertronixx, die über Netzwerkfunktionalität verfügen. Mit DeviLANControl können auch CAN2Web-Modul einfach und schnell konfiguriert und in Betrieb genommen werden.

4.1 Dialogelemente von DeviLANControl

Das Programm DeviLANControl verfügt über sechs Dialogboxen von denen zwei erst nach dem Verbindungsaufbau zum CAN2Web erscheinen:

DeviLAN-Manager: gestattet ein schnelles Finden und Kontaktieren von Modulen im Netzwerk (LAN)

COM-PC: gestattet bei der Verbindung über eine serielle Schnittstelle eines PCs, die Schnittstelle zu konfigurieren und den Datenverkehr zu analysieren.

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.

Konfiguration: dient zur Konfiguration der Hardware und des Datenstroms des CAN2Webs

CAN&COM: dient zur Darstellung von CAN/COM-Datenströmen vom CAN2Web

4.2 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 CAN2Web-Moduls sowie Einstellen der Netzwerkparameter (IP, Netmask, Gateway)

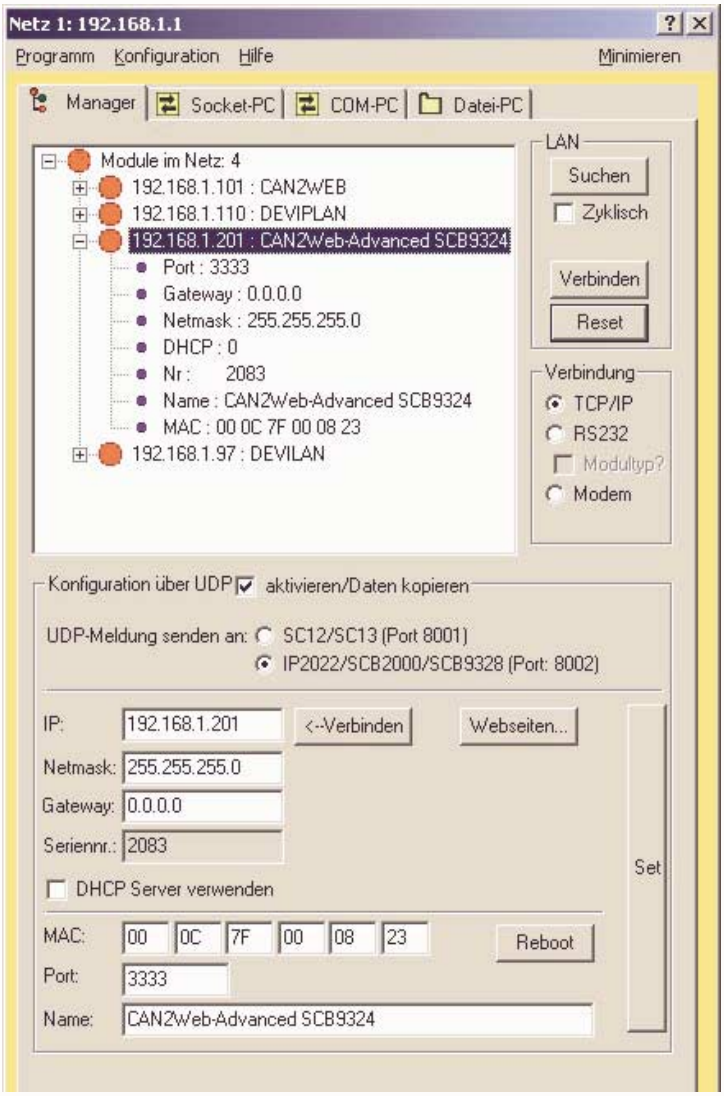


Abb. 6: Dialogbox DeviLAN-Manager

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

4.2.1 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 (*Konfiguration, CAN, COM, etc.*) 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“.

(*) Anmerkung: Damit der UPD-Identifikationsmechanismus arbeitet muss auf dem Modul das Programm „updconfig.exe“ laufen.

4.2.2 Konfiguration der Netzwerkparameter über UDP

Befindet sich das Modul innerhalb des lokalen Netzwerks, so können die Netzwerkparameter (IP, Netmask, Gateway, etc.) 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.

4.2.3 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 Konfiguration 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“.

4.2.4 Betrieb über die serielle Schnittstelle

Zur Anbindung des CAN2Webs an den PC ist wie folgt vorzugehen.

1. Die serielle Schnittstelle COM1 des CAN2Web-Moduls als Kommadoschnittstelle konfigurieren und die gewünschten Baudraten und Konfigurationseinstellungen vornehmen (siehe auch Kap. 4.3). Diese Einstellungen sind dabei zunächst per Webinterface oder mit DeviLAN-Control vorzunehmen.

2. Verbinden der seriellen Schnittstelle COM1 des CAN2Web-Moduls mit einer freien serielle Schnittstelle (RS232) des PCs

3. Starten des Programms DeviLANControl

4. Karteikarte „COM-PC“ wählen und mittels des Buttons „Suchen“ nach nicht belegten verfügbaren Schnittstellen des PCs suchen

5. Auswahl der gewünschten PC-Schnittstelle über die bereitgestellte Auswahlbox und Einstellen der Parameter (Baudrate, etc.), sodass diese mit den COM1-Einstellungen überein stimmen.

6. Auf Dialogbox „DeviLAN-Manager“ wechseln (Abb. 7) und Auswählen von „Verbinden über ... RS232“

7. Checkbox „Modultyp?“ aktivieren. Das Programm fragt jetzt zyklisch über die serielle Schnittstelle nach dem Modultyp.

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

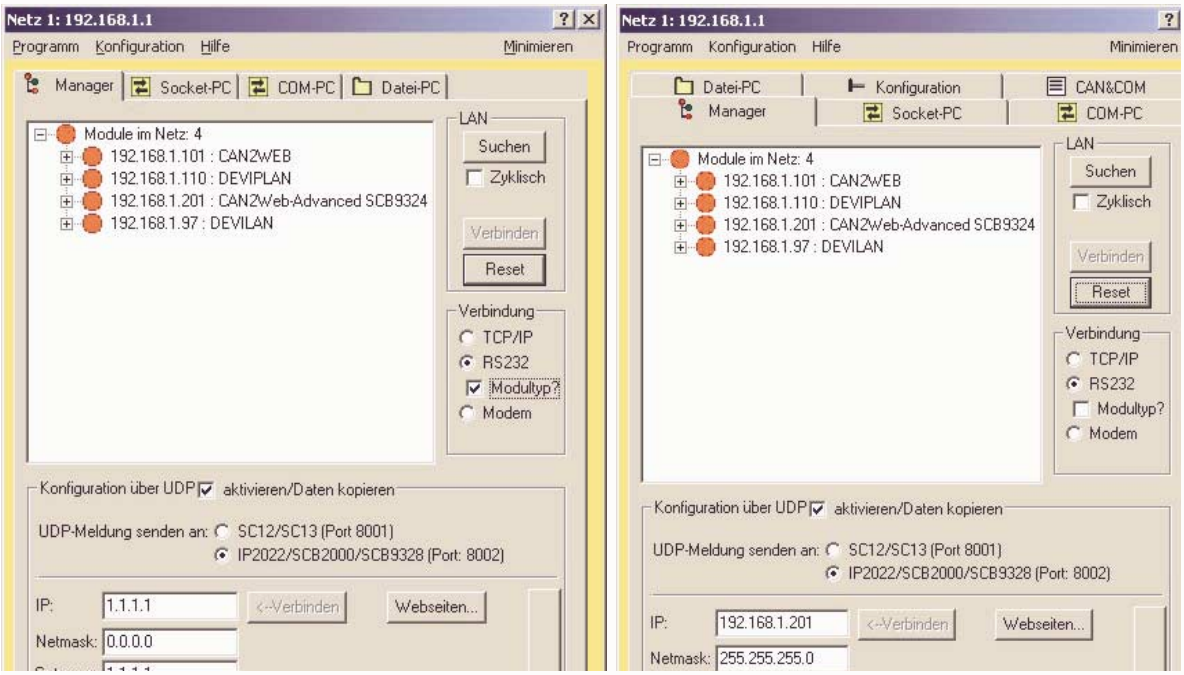


Abb. 7: Konfiguration über serielle Schnittstelle vor (links) und nach (rechts) Verbindungsaufbau

Das CAN2Web ist nun über die serielle Schnittstelle mit DeviLANControl verbunden und kann wie gewünscht konfiguriert und bedient werden.

4.2.4.1 Serielle Schnittstelle des CAN2Web-Professionals

Die serielle Schnittstelle des CAN2Web-Professional-Moduls wird nach dem Einschalten (Power-Up) zunächst vom Bootloader belegt (115200 Baud, 8 Datenbit, 1 Stoppbit, keine Parität, kein Handshake). Der Bootloader sendet über die Schnittstelle diverse Startmeldungen. Nach dem Starten von Linux dient sie als Linux-Konsole.

Es ist möglich die Konsolen-Funktionalität in den Einstellungen des Bootloaders zu deaktivieren, damit die Schnittstelle NACH dem Start von Linux auch für andere Aufgaben/Applikationen zur Verfügung steht.

In diesem Fall ist es dann auch möglich das CAN2Web-Professional-Modul über die serielle Schnittstelle zu konfigurieren.

4.3 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.

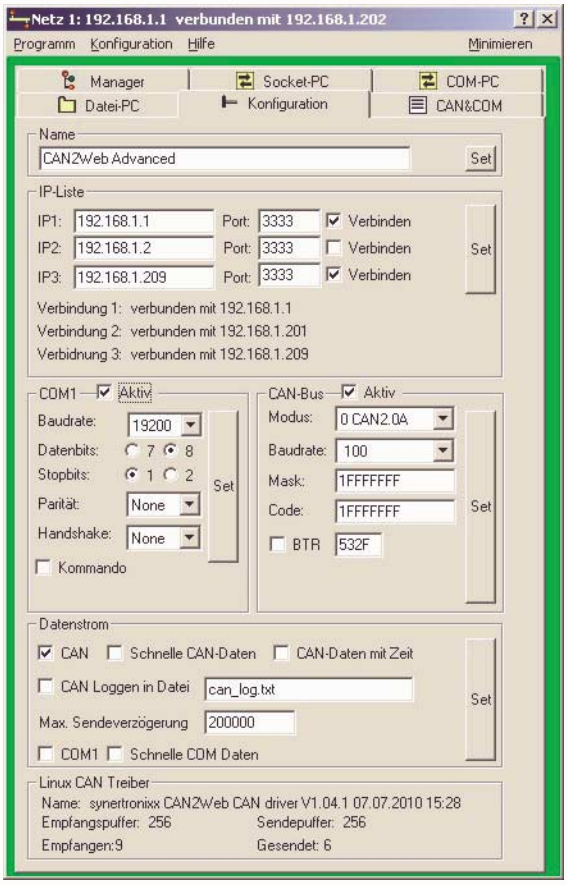


Abb. 8: Dialogbox Konfiguration

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 „Datenstrom“ erfolgen. Grundsätzlich sollten nur Schnittstellen aktiviert und deren Daten angezeigt werden sollen.

Anmerkung:

a) Die auftretende Datenmenge kann die maximal übertragbare Datenmenge über eine Socket- oder RS232-Verbindung überschreiten. Dies stellt keinen Fehler der CAN2Web-Software/Hardware dar, sondern ergibt sich aus den Spezifikationen der unterschiedlichen Schnittstellen.

b) Die Checkboxen „120 Ohm Widerstand“ und „High Speed Mode“ haben für das CAN2Web-Advanced-Modul keine Funktion.

4.4 Die Dialogbox CAN&COM

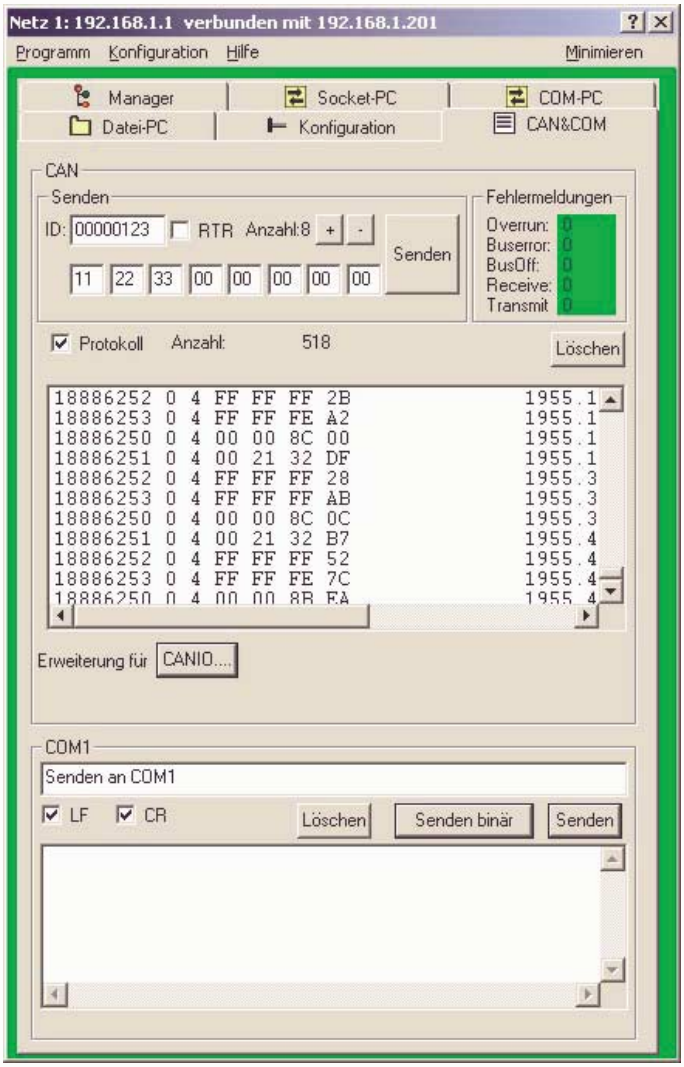


Abb. 9: Dialogbox CAN&COM

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 Schnittstelle COM1.

Weitere Funktionen:

a) Ab Version 1.07 von DeviLAN-Control 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 Meldung in binärer Form vorlag setzt die Ausgabefunktion (z. B. im Dialog „Socket“) ein Ausrufezeichen vor die Meldung.

Beispiel 1 : 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
```

4.5 Die Dialogbox COM-PC

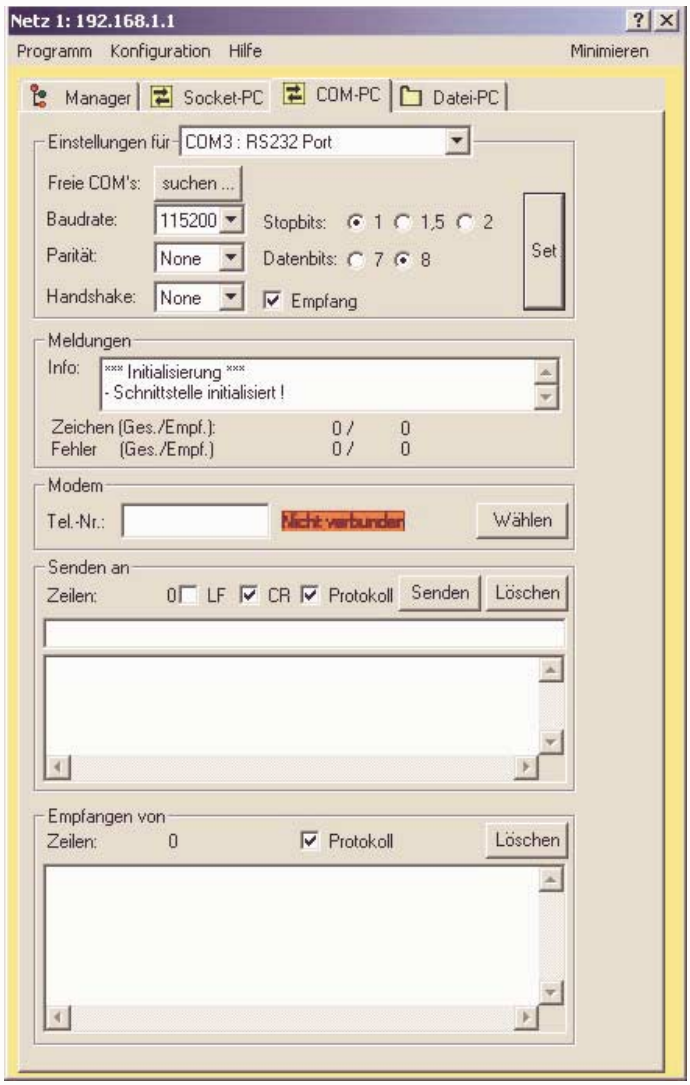


Abb. 10: Dialogbox COM-PC

Über die Dialogbox *COM-PC* (Abb. 10) kann der Datenverkehr zwischen dem PC und dem CAN2Web-Modul bei Verbindung über eine serielle Schnittstelle (siehe Kap. 4.2.4) überwacht werden.

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.

4.6 Die Dialogbox Socket

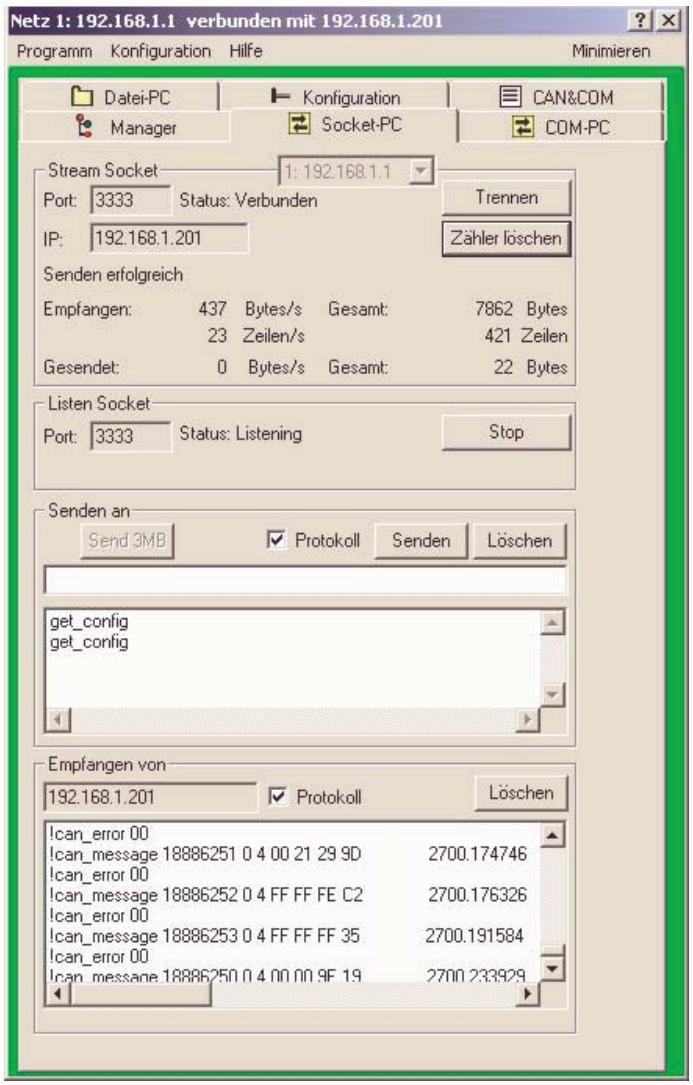


Abb. 11: Dialogbox Socket

Die Dialogbox *Socket* (Abb. 11) dient zum Aufbau einer Netzwerk/ Internetverbindung (TCP/IP) zu einem CAN2Web-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 3333) 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.

4.7 Die Dialogbox Datei-PC

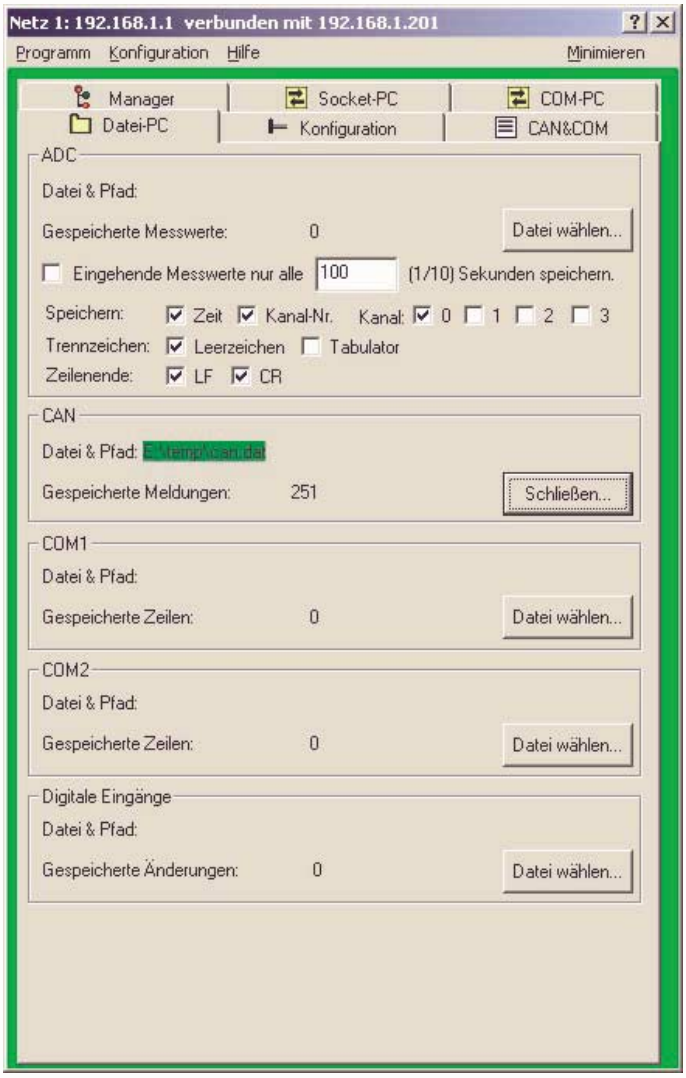


Abb. 12: Dialogbox Datei-PC

Die Dialogbox *Datei-PC* (Abb. 12) erlaubt Daten vom den verschiedenen Schnittstellen in Dateien zu speichern. Beim CAN2Web sind dies nur die CAN/COM-Daten.

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 bzw. ausgewertet werden.

5 Kommandointerface

5.1 Hardwareübersicht

Das Kommandointerface gestattet einen vollständigen Zugriff auf alle Konfigurationseinstellungen und erlaubt, die Schnittstellen abzufragen. Die Hardware besteht aus den Funktionsgruppen, die in Abb. 13 wiedergegeben werden.

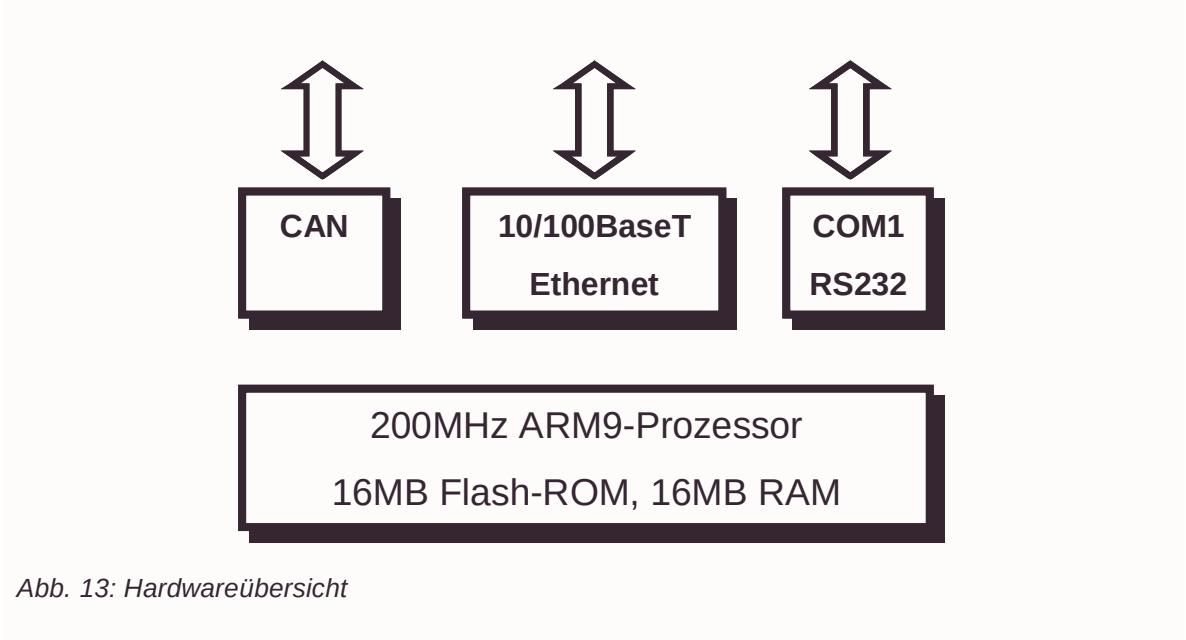


Abb. 13: Hardwareübersicht

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

5.2 Kommunikationsschnittstellen

Das CAN2Web-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 oder eine serielle Schnittstelle gestattet.

Die Möglichkeiten der Anbindung sind in Abb. 14 wiedergegeben.

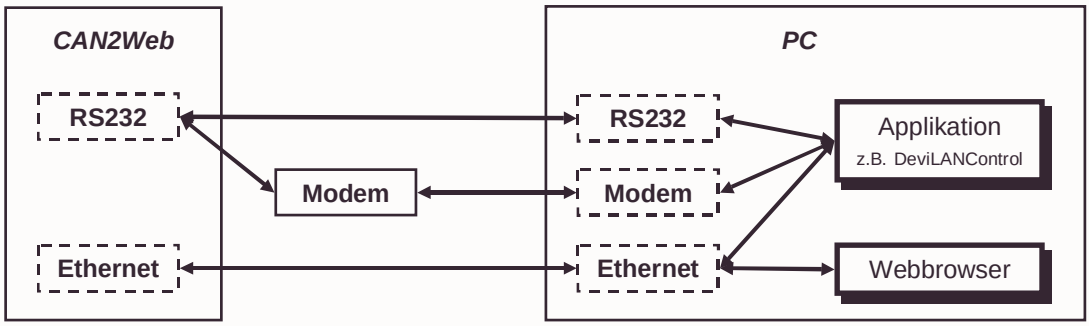


Abb. 14: CAN2Web Kommunikationsschnittstellen

Die Funktionsweise lässt sich vereinfacht mittels Abb. 15 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.

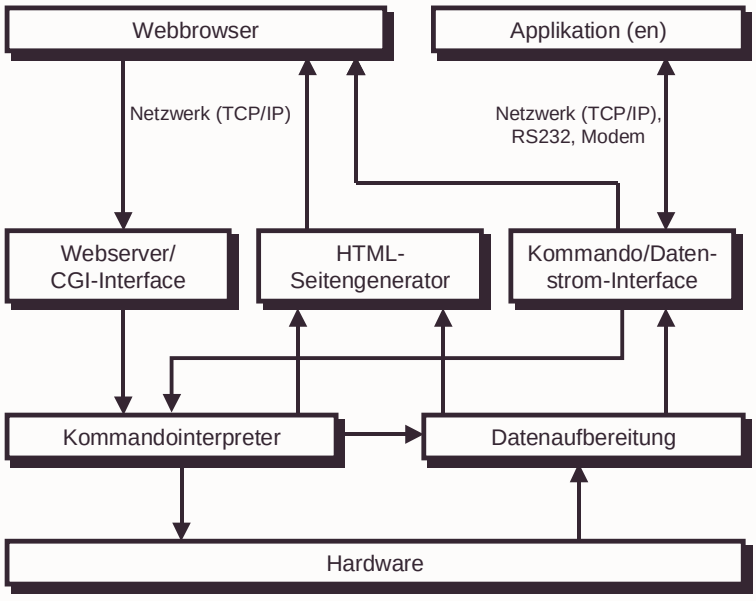


Abb. 15: Kommunikationszugriff (vereinfacht)

5.3 Allgemeine Informationen

5.4 Kommunikationsprotokoll

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 nachfolgend 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.

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.

Ein ASCII-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 1, die Steuerzeichen Tabelle 2 entnommen werden.

Tabelle 1 : 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 also z. B. 192.168.1.226

Tabelle 2 : Steuerzeichen

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

Der ASCII-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 CAN2Web-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 2 : Kommunikationsmodus konfigurieren

```
can_fast 1; An CAN2Web: verwende schnelle CAN-Kommunikation
can_fast 1; Von CAN2Web: schnelle CAN-Kommunikation eingestellt
```

5.4.1 Fehlermeldungen für ASCII-Kommunikation

Entdeckt der Kommanodinterpreter während der Verarbeitung einer ASCII-Kommandozeile einen Fehler, so sendet er eine Fehlermeldung (Tabelle 3) mit folgenden Aufbau.

Syntaxangabe 2 :Fehlermeldung des Kommandointerpreters

```
<„error“><SP><Fehler-Nr.><„:“><„Fehlermeldung im Klartext“><LF><CR>
```

Beispiel 3 : Fehlermeldungen des Kommandointerpreters

```
error 4: Unkown parameter name <LF><CR> ; (Parametername/Kommando unbekannt)
```

Es sind folgende weitere Fehlermeldungen definiert:

Tabelle 3 : Fehlermeldungen des Kommandointerpreters

Fehlernr.	Fehlerbeschreibung
ERROR_NO_ERROR	"No error!" Es liegt kein Fehler vor
ERROR_COMMAND_LINE_SIZE	"Command line to long!" Die Kommandozeile ist zu lang.

Fehlernr.	Fehlerbeschreibung
ERROR_PARAMETER_NAME_SIZE	"Parameter name to long" Der Parametername ist zu lang
ERROR_PARAMETER_VALUE_SIZE	"Parameter value to long or short!" Der Parameterwert ist zu lang, zu kurz/nicht vorhanden
ERROR_UNKOWN_KEYWORD	"Unkown parameter name" Der Parametername ist unbekannt.
ERROR_CONVERSION	"Unable to convert parameter" Die Konvertierung des Parameterwertes ist nicht möglich. Es wird zusätzlich eine Meldung mit dem aktuellen Wert des Parameters zurück gegeben.
ERROR_OUT_OF_RANGE	"Parameter out of range" Der Parameterwert liegt ausserhalb des spezifizierten Bereichs. Es wird zusätzlich eine Meldung mit dem aktuellen Wert des Parameters zurück gegeben.
unbekannt	"Unkown error id!" Unbekannte Fehlernummer, dieser Fehler sollte nicht auftreten.

5.5 Konfiguration der Hardware/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 CAN2Web-Modulen implementiert und können Kap. 5.5.2 und Kap. 5.5.1 entnommen werden.

5.5.1 Serielle Schnittstelle COM1

Die serielle Schnittstelle COM1 wird über die Befehle in Tabelle 4 konfiguriert.

Tabelle 4 : 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 CAN2Web-Modul kann dadurch z. B. über eine RS232-Schnittstelle eines PCs angesteuert werden.

Um Daten über die Schnittstelle zu senden, können die Befehle in Tabelle 5 verwendet werden. Die Kommunikation ist zeilenorientiert ausgelegt.

Tabelle 5 : 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.
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 Car-riage Return (<CR>, Hexadezimal: 0x0D) ange-hängt.

Beispiel 4 : 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.

Anmerkung für das CAN2Web-Professional:

Die serielle Schnittstelle COM1 wird beim Systemstart zunächst als Bootloader bzw. Linux-Debug-Schnittstelle genutzt. Ggf. muss diese Option im Bootloader abgeschaltet werden, damit nach dem Start von Linux die Schnittstelle uneingeschränkt zur Verfügung steht.

Beim CAN2Web-Advanced steht die serielle Schnittstelle COM1 uneingeschränkt zur Verfügung. Die Linux-Debug-Schnittstelle wird über einen zusätzlichen Steckverbinder zur Verfügung gestellt.

5.5.2 CAN-Bus

Die Konfiguration des CAN-Bus erfolgt über die Befehle in Tabelle 6.

Tabelle 6 : 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_extended	0,1,2	0	Legt den CAN-Modus und die Länge des CAN-Identifiers fest: 0: 11-Bit Identifier (CAN2.0A) 1: 29-Bit Identifier (CAN2.0B) 2: 11/29-Bit Identifier (CAN2.0A/CAN2.0B) gemischter Modus
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*

Befehl	Werte	Default	Funktionsbeschreibung
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.
can_resistor	0,1	0	Nur für CAN2Web-Professional implementiert: Schaltet den 120Ohm Abschlusswiderstand beim CAN2Web-Professional parallel zu den beiden CAN-Leitungen wenn *can_resistor*=1 ist.
can_highspeed	0,1	0	Nur für CAN2Web-Professional implementiert: Schaltet den CAN-Treiber beim CAN2Web-Professional in den High-Speed-Modus wenn *can_highspeed*=1 ist.

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 7 bereit.

Tabelle 7 : Senden über den CAN-Bus

Befehl	Werte	Default	Funktionsbeschreibung
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 (bzw. can_msa) (bzw. can_msb) <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: a) Die einzelnen Parameter müssen durch genau ein Leerzeichen <SP> voneinander getrennt werden. b) Wird der CAN-Bus im CAN2.0A/2.0B gemischten Modus (*can_extended*=2) betrieben, so ist der Befehl *can_msg* für eine CAN2.0A-Nachricht durch *can_msa* und für eine CAN2.0B-Nachricht durch *can_msb* zu ersetzen, damit die Meldung im richtigen CAN-Modus gesendet wird.

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* bzw. *can_msb* gestattet es eine vollständige CAN-Meldung zu übertragen und zu senden.

Mit den nachfolgenden Kommandos (Tabelle 8) können weitere Informationen zum Linux CAN-Treiber ausgelesen werden. Diese Parameter können nicht geschrieben werden.

Tabelle 8 : Linux CAN-Treiber

Befehl	Wert/Typ	Default	Funktionsbeschreibung
can_driver_name	String[127]		Name und Version/Datum des Linux CAN-Treibers
can_driver_rcv	LONG UINT	0	Anzahl der empfangenen CAN-Meldungen seit der letzten Initialisierung des Busses
can_driver_snd	LONG UINT	0	Anzahl der gesendeten CAN-Meldungen seit der letzten Initialisierung des Busses
can_driver_rcv_buf	LONG UINT	256	Größe des Empfangsbuffers für CAN-Meldungen
can_driver_rcv_snd	LONG UINT	256	Größe des Sendebuffers für CAN-Meldungen

Anmerkung:

Die obigen Parameter werden nur aktualisiert, wenn der CAN-Bus neu initialisiert wurde oder wenn die gesamte Konfiguration vom CAN2Web gelesen wird.

5.5.3 Konfiguration des Datenstroms

In Tabelle 9 sind die Befehle zur Konfiguration des Datenstrom aufgeführt. Diese enthalten auch Befehle zur Umschaltung zwischen ASCII-Kommandozeilen und binärer Kommunikation.

Tabelle 9 : Konfiguration des Datenstroms

Befehl	Werte	Default	Funktionsbeschreibung
can_stream	0,1	0	Zeigt kontinuierlich die über den CAN-Bus eingehenden Meldungen an
can_fast	0,1	0	Aktiviert/Deaktiviert die schnelle Übertragung von CAN-Daten über die Socketverbindung 0: ASCII-Kommandozeile 1: binäre, schnelle Kommunikation

Befehl	Werte	Default	Funktionsbeschreibung
can_time	0,1	0	Aktiviert/Deaktiviert die Empfangszeit 0: keine Zeitangabe 1: Zeitangabe (Systemzeit: Sekunden+µSekunden-wert wird zusätzlich übertragen)
can_log_on	0,1		Aktiviert/Deaktiviert die CAN-Log-Funktion 0: Loggen der CAN-Daten eingeschaltet 1: Loggen der CAN-Daten abgeschaltet
can_logfile	STRING [50]	can_log. txt	Logdatei: Pfad und Datei in dem Logdaten als ASCII-Text geschrieben werden. Zur Verwendung von USB-Memory-Sticks siehe auch die SXX-Dateien in etc/init.d/
com1_stream	0,1	0	Zeigt kontinuierlich die über die serielle Schnittstelle COM1 eingehenden Daten an
com_fast	0,1	0	Aktiviert/Deaktiviert die schnelle Übertragung von COM-Daten über die Socketverbindung 0: ASCII-Kommandozeile 1: binäre, schnelle Kommunikation **Wichtig**: Werden zwei CAN2Web-Module über TCP/IP miteinander verbunden, so **muss** die Kommunikation zwingend auf ***can_fast=1*** gesetzt werden, damit die CAN-Meldungen übertragen werden. Die Module reagieren nicht auf die eingehenden ASCII-Meldungen *can_message*, *can_messaga* und *can_messagb* !
tcp_max_delay	0...999.999	200.000	Max. Verzögerungszeit in µs nachdem die im Puffer stehenden CAN-Meldungen über TCP-IP gesendet werden. Die typische Durchlaufzeit der Hauptprogrammschleife beträgt 100µs, sodass Werte unter 100µs keinen Einfluss auf die Reaktionszeit haben.

5.5.4 Datenformate für CAN im Datenstrom

Ist der CAN-Bus aktiv, werden die Werte im folgenden Datenformat gesendet, wenn ASCII-Kommunikation (*can_fast=0*) gewählt wurde und der CAN-Bus für CAN2.0 (*can_extended=0*) oder CAN2.0B *(can_extended=1*) konfiguriert ist.

Syntaxangabe 3 :Datenformat für den CAN-Bus

```
<can_message><SP><id><SP><rtr><SP><len>[<SP><b0>][<SP><b1>]
[<SP><b2>][<SP><b3>][<SP><b4>][<SP><b5>][<SP><b6>][<SP><b7>]
[Systemzeit Sekundenangabe]<.>[Systemzeit µ-Sekundenangabe]<LF><CR>
```

Ist der CAN-Bus für den gemischten CAN2.0A/CAN2.0B-Betrieb (*can_extended=2*) konfiguriert, wird für eine CAN2.0A-Meldung der Schlüsselwort *can_messaga* und für eine CAN2.0B-Meldung der Schlüsselwort *can_messagb* gesendet.

Tabelle 10 : Erklärung zu Syntax 3

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 5 : 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 6 : Datenausgabe CAN-Bus, zusätzlich Systemzeit
(Steuerzeichen werden nicht dargestellt)

```
can_message 00001FA0 1 8 00 11 22 33 44 55 66 77 88 19256.334590
can_message 00001FA3 1 8 88 11 22 33 44 55 66 77 88 19256.456012
can_message 00001FA3 1 4 11 22 33 44  219256.9852232
```

Sollte ein Fehler vorliegen erfolgt folgende Meldung:

Syntaxangabe 4 :CAN-Fehlermeldung

```
can_error<SP><errorbyte><LF><CR>
```

Tabelle 11 : Erklärung zu Syntax 4

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

```
can_error 20
can_error 28
```

5.5.5 Datenformate der seriellen Schnittstelle COM1 im Datenstrom

Die Kommunikation der seriellen Schnittstellen ist zeilenorientiert ausgelegt, d.h. die eingehenden Daten/Zeichen, die über COM1 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 5 :Datenformat für COM1

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

Tabelle 12 : Erklärung zu Syntax 5

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

5.5.6 Weitere Einstellungen und Abfragebefehle

Über die Befehle in Tabelle 13 können weitere Einstellungen getätigt und Information abgefragt werden.

Tabelle 13 :Konfiguration der weiteren Einstellungen

Befehl	Wert/Typ	Default	Funktionsbeschreibung
modul_name	STRING [39]	CAN2Web	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 1	-	-	Speichert die aktuelle Konfiguration und die voreingestellten Werte des CAN2Web-Moduls

5.5.7 Weitere Meldungen von CAN2Web

Nach dem erfolgreichen Aufbau einer Verbindung zum Modul überträgt dieses automatisch eine Startmeldung bzw. bei bestehender Verbindung auf Anfrage zusätzliche Informationen (Tabelle 14).

Tabelle 14 : Zusatzinformationen

Befehl	Typ	Beschreibung
modul_ip	IP	Gibt die IP-Adresse des CAN2Web-Moduls an.
modul_mac	XX XX XX XX XX XX	Gibt die MAC-Adresse des CAN2Web-Moduls an
modul_netmask	IP	Gibt die Netmask des CAN2Web-Moduls an.
modul_gateway	IP	Gibt die Gateway-Einstellung des CAN2Web-Moduls an.

5.5.8 Informationen zum Linux CAN-Treiber

Mit den nachfolgenden Befehlen können Informationen zum Linux-CAN-Treiber abgefragt werden (Tabelle 15).

Tabelle 15 : CAN-Treiber-Informationen

Befehl	Typ	Beschreibung
can_driver_name	STRING[128]	Name des Linux CAN-Treibers
can_driver_rcv_buf	UINT	Größe des CAN-Empfangspuffers
can_driver_snd_buf	UINT	Größe des CAN-Sendepuffers

Befehl	Typ	Beschreibung
can_driver_rcv	LONG UINT	Anzahl der empfangenen CAN-Meldungen seit der letzten Konfiguration des CAN-Busses
can_driver_snd	LONG UINT	Anzahl der gesendeten CAN-Meldungen seit der letzten Konfiguration des CAN-Busses

Anmerkung:

Alle oben angegebenen Befehle dienen nur zum Auslesen der Werte. Sie können nicht geschrieben werden. Der im Befehl übergebene Parameter wird nicht übernommen. Statt dessen wird der Wert aus der Treiberinformation zurück gegeben.

5.5.9 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 16 gesteuert.

Tabelle 16 :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 17) zum Zustand/Status der Verbindung.

Tabelle 17 :Status der Socket-Verbindungen

Befehl	Wert/ Typ	Default	Funktionsbeschreibung
ipx_state (x= 1,2,3)	0,1,2	0	0: Verbindung abgebaut/keine Verbindung 1: Verbindung wird aufgebaut 2: Verbindung ist aufgebaut, es wird zusätzlich die IP übertragen mit der der Socket verbunden ist

Beispiel 8 : Socketzustandsmeldung

```
ip1_state 0              ; Verbindung zu IP-Adresse 1 ist abgebaut
ip2_state 1              ; Verbindung zu IP-Adresse 2 ist aufgebaut
ip3_state 3 192.168.1.1 ; Verbindung zu IP-Adresse 3 aufgebaut
```

5.6 Befehle für schnelle/binäre Socketkommunikation

Es sind folgende Kennbytes (Startkennungen) für die schnelle/binäre Kommunkation implementiert worden (Tabelle 18):

Tabelle 18 :Kennbytes für schnelle/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.
0x83	CAN-Meldung mit Fehlerstatus des Busses und Empfangszeit der CAN-Meldung
0x91	Übertragung von Daten von der seriellen Schnittstelle COM1

Nachfolgend erfolgt eine Beschreibung der Meldungen.

5.6.1 Datenrahmen für binäre CAN-Meldung, Kennbyte 0x81

Empfängt ein CAN2Web-Modul einen Datenrahmen gemäß Tabelle 19 über eine Sokketverbindung, so legt es die Meldung umgehend auf den CAN-Bus. Empfängt es eine Nachricht über den CAN-Bus sendet es diese ebenfalls gemäß Tabelle 19 über alle Socketverbindungen, wenn schnelle CAN-Kommunikation voreingestellt wurde.

Tabelle 19 :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

Byte	Wert/ Inhalt	Beschreibung
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
6	<RTR+Modus+ LEN>	Angabe zu RTR-Bit, CAN-Modus und Anzahl der Datenbytes Bit 7: RTR-Bit Bit 6: 0:CAN-Bus in CAN2.0A oder CAN2.0B Modus betrieben 1:CAN-Bus wird im CAN2.0A/B gemischten Modus betrieben Bit 5: wenn Bit 6=1: 0:CAN2.0A-Meldung, 1:CAN2.0B-Meldung wenn Bit 6=0: immer 0 Bit 4: reserviert, momentan immer 0 Bit 3...0: Anzahl der Datenbyte der CAN-Meldung an
7..15	[<DATA0...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 (Betrieb im CAN2.0B-Modus) 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
6 : 0x08 ; Länge
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

Wird der Bus im CAN2.0A/B gemischten Modus betrieben so ist Byte 6 = 0x68 zu setzen.

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

Tritt auf dem CAN-Bus eine Änderung des Fehlerzustandes auf, sendet es gemäß Tabelle 22 über alle Socketverbindungen eine Fehlermeldung.

Tabelle 20 :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.

5.6.3 Datenrahmen für binäre CAN-Meldung mit Zeitangabe, Kennbyte 0x83

Empfängt es eine Nachricht über den CAN-Bus sendet es diese gemäß Tabelle 21 über alle Socketverbindungen, wenn schnelle CAN-Kommunikation voreingestellt wurde. Empfängt ein CAN2Web-Modul einen Datenrahmen gemäß Tabelle 21 über eine Sokketverbindung, so legt es die Meldung umgehend auf den CAN-Bus. Da die Zeitangabe keine Funktions für das Senden der Meldung hat, ist es sinnvoller stattdessen einen Datenrahmen mit Kennbyte 0x81 gemäß Kap. 5.6.1 zu verwenden.

Tabelle 21 :CAN-Meldung inkl. Status/Fehlerinformation und Zeitangabe

Byte	Wert/ Inhalt	Beschreibung
0	0x83	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

Byte	Wert/ Inhalt	Beschreibung
5	<CAN-ID0>	CAN-ID Byte 0
6	<RTR+Modus + LEN>	Angabe zu RTR-Bit, CAN-Modus und Anzahl der Datenbytes Bit 7: RTR-Bit Bit 6: 0:CAN-Bus in CAN2.0A oder CAN2.0B Modus betrieben 1:CAN-Bus wird im CAN2.0A/B gemischten Modus betrieben Bit 5: wenn Bit 6=1: 0:CAN2.0A-Meldung, 1:CAN2.0B-Meldung wenn Bit 6=0: immer 0 Bit 4: reserviert (immer 0) Bit 3...0: Anzahl der Datenbyte der CAN-Meldung an
7 bis 7+<LEN>	[DATA 0...7]	Datenbyte 0...7 der CAN-Meldung (optional)
11 bis 11+<LEN>	<TIME SEC>	Empfangszeit des CAN-Telegramms; Systemzeit Sekundenwert als 4 Byte unsigned long int (MSB zuerst). Die Empfangszeit setzt sich aus <TIME SEC> und <TIME µSEC> zusammen.
15 bis 15+<LEN>	<TIME µSEC>	Empfangszeit des CAN-Telegramms, Systemzeit µ-Sekundenwert als 4 Byte unsigned long int (MSB zuerst), Wertebereich 0...999999 Die Empfangszeit setzt sich aus <TIME SEC> und <TIME µSEC> zusammen.

Die Gesamtlänge der Meldung in Byte beträgt 15 + LEN.

Wurde beispielsweise folgende Meldung auf dem CAN-Bus empfangen

ID : 0x03A2A3FE ; 29-Bit-Identifier
RTR : 0
8 Datenbyte : 0x01, 0x02, 0x03, 0x04, 0x80, 0x70, 0x60, 0x40
Time Sec : 0x00, 0x00, 0xFA, 0xC0
Time µSec : 0x00, 0x01, 0x80, 0x33

so betrug die Linux-Systemzeit von 64192 Sekunden und 98355 µ-Sekunden.

Anmerkung: Der CAN-Controller löst einen Interrupt aus, sobald ein CAN-Telegramm vollständig empfangen wurde. Innerhalb der Interrupt-Service-Routine (ISR) wird unmittelbar vor dem Auslesen des CAN-Controllers die Systemzeit bestimmt. Zeitangaben zwischen Empfang der CAN-Meldung durch den CAN-Controller und dem Aufruf der ISR liegen nicht vor.

5.6.4 Datenrahmen für binäre COM-Meldung

Empfängt ein CAN2Web-Modul ein Datenrahmen gemäß Tabelle 22 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 22 über alle Sokketverbindungen, wenn schnelle COM-Kommunikation voreingestellt wurde.

Tabelle 22 :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(s), 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

6 Anbindung eigener Applikationen

6.1 Anbindung über Netzwerk

Die Anbindung des Moduls kann über ein Netzwerk (LAN) über eine Socketschnittstelle (TCP/IP) zwischen PC und CAN2Web erfolgen.

Als Beispielapplikation steht das PC-Programm DeviLANControl zur Verfügung (siehe Kap. 4). Mit ihm läßt sich das CAN2Web-Modul über das Netzwerk bedienen und konfigurieren.

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.

6.2 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 CAN2Web-Modul aufgebaut werden. Gehen Sie dazu wie folgt vor:

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

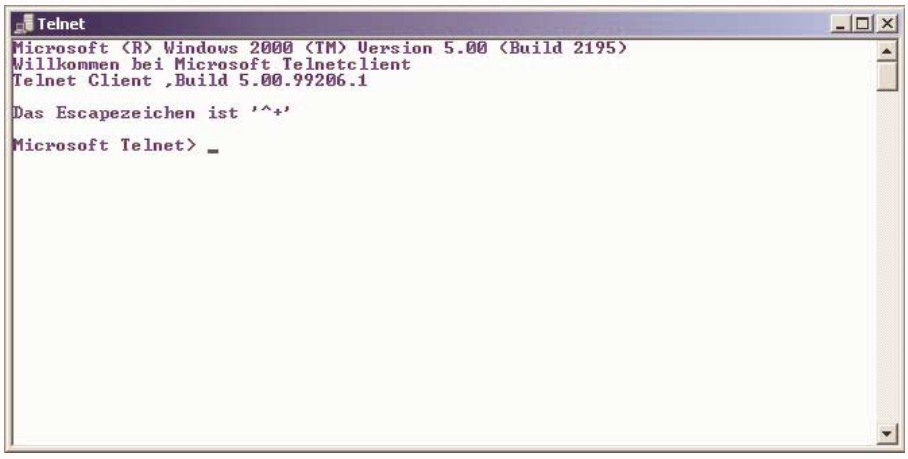


Abb. 16: Telnet-Startfenster

Im Menü auf „Verbinden“, „Netzwerksystem“ und unter „Hostname“ die IP-Adresse des Moduls eintragen also z.B. 130.75.65.130 (Abb. 17). Unter „Anschluss“ die Standard Portnummer 3333 des Kommando/Datenstrom-Interface angeben. Danach über „Verbinden“ die Verbindung aufbauen.

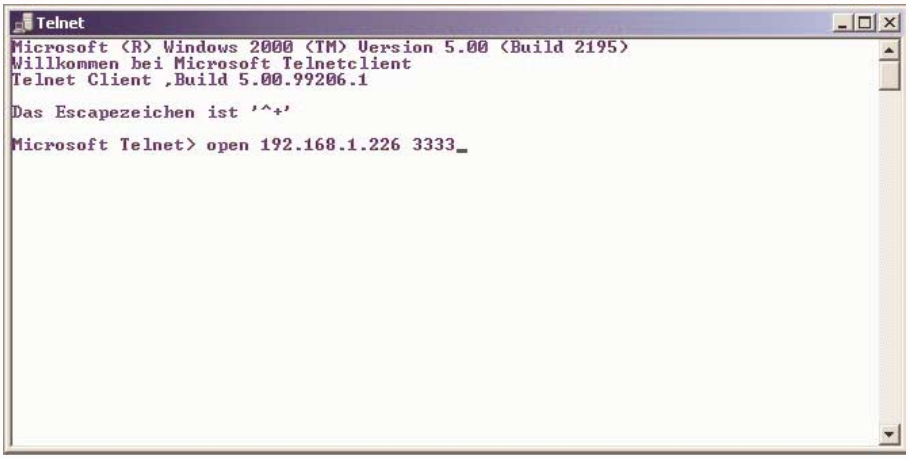


Abb. 17: Telnet-Startfenster

Ist der Verbindungsaufbau erfolgreich gibt das Modul automatisch einige Informationen aus (Abb. 18).

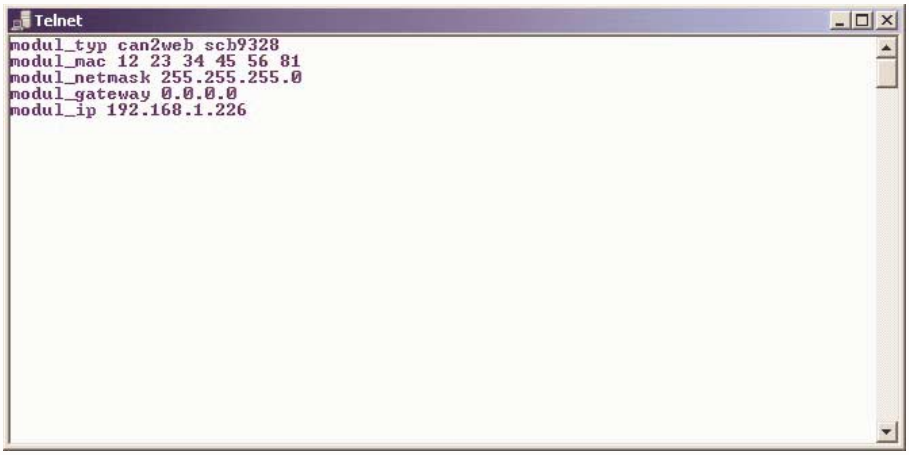


Abb. 18: Ausgabe bei erfolgreichem Verbindungsaufbau

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. 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.

7 Linux Kernel Treiber

7.1 Allgemein

Der bereitgestellte Linux-Treiber für den CAN-Bus bietet die Möglichkeit eigene Applikationen in C/C++ zu erstellen.

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 implementierten 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. 256 Meldungen zwischengespeichert werden. Der Puffer muss durch die Anwendung daher rechtzeitig ausgelesen werden, damit kein Datenverlust auftritt.

Der Sendepuffer kann ebenfalls 256 CAN-Meldungen aufnehmen. Das Versenden der Meldungen erfolgt ebenfalls Interrupt-gesteuert durch den Treiber.

Tabelle 23 :Header-Dateien

Header-Datei	Beschreibung
can_driver.h	Header Datei zur Erstellung eigener Applikationen
can_driver.ko	Kernel Treiber für den CAN-Bus

7.2 Verwendung des CAN-Treibers

Anhand der nachfolgenden Quelltexte wird exemplarisch die Verwendung des Treiber erklärt. Weitere Erläuterungen zur Verwendung der Routinen befinden sich auch im Quelltext von **catest.c**, mit dem das Beispielprogramms **catest.exe** erzeugt werden kann.

Der Zugriff auf den Treiber erfolgt dabei prinzipiell analog zum Zugriff auf Dateien. Dazu muss zunächst eine „Datei“ geöffnet werden, um danach Lese- und Schreibvorgänge durchführen zu können. Im Gegensatz zu Dateien gibt es zusätzlich Kontroll- und Konfigurationsbefehle. Wird die „Datei“ nicht mehr benötigt, sollte sie geschlossen werden.

7.2.1 Öffnen und Schließen (open, close)

Die Befehle open bzw. close öffnen eine Verbindung zum CAN-Treiber bzw. schließen diese wieder. open gibt einen gültigen file descriptor zurück, wenn der Vorgang erfolgreich war, anderenfalls sollte überprüft werden, ob der CAN-Treiber läuft.

Beispiel 9 : Öffnen/Schliessen einer Verbindung zum CAN-Treiber

```
int fd_can;                          // file descriptor
....
fd_can = open("/dev/can", O_RDWR);  // open CAN device for read/write
```


```
if (fd_can<0)
  printf("can't open CAN device\n";
....                                  // configure, read, write, ...
close(fd_can);                        // close CAN device
```

7.2.2 Lesen (read)

Mit dem read-Befehl werden CAN-Telegramme vom Empfangspuffer des CAN-Treiber gelesen. Der Befehl arbeitet meldungsorientiert, d.h. der Auslesezeiger wird mit Ausführung von read auf die nächste empfangene CAN-Meldung gesetzt. Je nach Anwendung können die CAN-Meldungen wahlweise mit oder ohne Empfangszeitangabe gelesen werden.

Beispiel 10 : Auslesen von empfangenen CAN-Meldungen

```
int fd_can;         // file descriptor for CAN kernel driver
struct TCANMsg  c1; // CAN-Message-stucture without time
struct TCANMsgT c2; // CAN-Message-stucture with time
.....
// Read CAN msg. without time information
if (read(fd_can,(void*)&c1, sizeof(c1)) == sizeof(c1) )
  // read msg. correctly: process data ....
....
// Read CAN msg. with time information
if (read(fd_can,(void*)&c2, sizeof(c2)) == sizeof(c2) )
  // read msg. correctly: process data ....
```

Vor dem Auslesen von CAN-Meldungen sollte per ioctl-Kommando getestet werden, ob CAN-Meldungen im Empfangspuffer verfügbar sind.

7.2.3 Schreiben (write)

Mit dem write-Befehl werden CAN-Telegramme zum Sendepuffer des CAN-Treibers geschreiben. Der Befehl arbeitet meldungsorientiert, d.h. es muss eine vollständige CAN-Meldung übergeben werden.

Beispiel 11 : Senden von CAN-Meldungen

```
int fd_can;         // file descriptor for CAN kernel driver
struct TCANMsg  c1; // CAN-Message-stucture without time
.....
// build CAN-message
c1.ID  = 0x123456;
c1.RTR = 0;
c1.LEN = 8;
for (i=0;i<=c1.LEN-1;i++)
  c1.DATA[i]=i;
// send messages
if (write(fd_can,(void*)&c1, sizeof(c1)) == -1 )
  printf("CAN transmit buffer full!\n");
```

7.2.4 Konfigurieren (ioctl)

Der CAN-Kernel-Treiber steuert die CAN2Web-Hardware an. Er wird über die nachfolgenden Konfiguration- und Kontroll-Kommandos (ioctl) vom User-Space aus eingestellt (siehe auch Datei can_driver.h).

Tabelle 24 :ioctl-Kommandos des CAN-Treibers

Kommando	Beschreibung
CAN_SET_SETTINGS	CAN-Bus Initialisierung: Die Einstellungen werden dem Treiber über die Struktur „CANSettings“ übergeben. Alle Sende/Empfangspuffer werden gelöscht und alle Zähler zurück gesetzt.
CAN_GET_SETTINGS	Lesen der CAN-Bus Einstellungen: Die Einstellungen werden vom Treiber über die Struktur „CANSettings“ übergeben.
CAN_GET_DRIVER_INFO	Lesen von weiteren Treiber-Informationen: Die Informationen wie Treiberversion, Größe der Sende/Empfangspuffer und Summe aller gesendeten und empfangenen CAN-Meldungen werden vom Treiber über die Struktur „DriverInfo“ übergeben.
CAN_SET_ON_OFF	Ein/Ausschalten der CAN-Kommunikation '1' = Einschalten und '0' =Ausschalten
CAN_GET_STATE	Lesen des CAN-(Fehler)-Status
CAN_GET_RECEIVE_MSG	Lesen der Anzahl der CAN-Meldungen im Empfangspuffer
CAN_GET_TRANSMIT_MSG	Lesen der Anzahl der CAN-Meldungen im Sendepuffer
CAN_SET_DEBUGMODE	Ein/Ausschalten der CAN-Treiber Debugmeldungen auf der Konsole '1' = Einschalten und '0' = Ausschalten

Beispiel 12 : ioctl: Anzahl der CAN-Meldungen im Empfangspuffer

```
int fd_can;   // file descriptor for CAN kernel driver
int nr;       // nr. of CAN messages
.....
if (ioctl(fd_can, CAN_GET_RECEIVE_MSG, &nr)<0)
  ....        // handle error
```

Die ioctl-Funktion des CAN-Treibers liefert einen negativen Wert zurück, falls das Kommando nicht ausgeführt werden konnte.

8 Linux-Tipps, -Befehle und -Konfiguration

8.1 Allgemeines

Die folgenden Abschnitte enthalten einige Tipps zum Umgang mit Linux und sollen den Einstieg erleichtern. Momentan sind dies hauptsächlich eine Zusammenstellung einiger nützlicher Linux-Befehle. Dieser Abschnitt befindet sich noch in der Bearbeitung und wird kontinuierlich erweitert.

Über die Konsole des Socketcomputers kann mit dem Befehl

```
<Befehlsname> --help
```

eine Hilfetext (soweit vorhanden) angefordert werden.

```
mv -- help  ; Hilfe zu mv (move) ausgeben

Ausgabe:
Usage: mv [OPTION]... SOURCE DEST
or: mv [OPTION]... SOURCE... DIRECTORY
Rename SOURCE to DEST, or move SOURCE(s) to DIRECTORY
Options:
        -f      Don't prompt before overwriting
        -i      Interactive, prompt before overwrite
```

Die Konsole (shell) verfügt über einen Mechanismus zur Autovervollständigung von Eingaben. Durch zweimaliges Drücken der Taste „Tab“ erhält man eine Liste der wichtigsten Befehle und Programme.

8.2 Passwörter

Um sich mit der Linux-Konsole zu verbinden, ist eine Authentifizierung notwendig. Der voreingestellte Benutzername ist „root“, das voreingestellte Passwort ist „pass“.

Dieser Passwörter werden auch für Telnet, FTP und SSH/SCP verwendet.

Um das Passwort zu ändern, kann der Befehl „passwd [nutzername]“ verwendet werden. Wird der Nutzername weg gelassen, wird das Passwort des aktuellen Nutzers geändert.

```
passwd root                         ; Passwort für root ändern
Ausgabe:
Changing password for root
New password:                       ; hier neues Passwort eingeben
Retype password:                    ; hier neues Passwort wiederholen
Password for root changed by root   : Info: Passwort geändert
```

Um weitere Nutzer anzulegen, wird der Befehl „adduser [OPTIONS] user_name“ verwendet.

Die angelegten Nutzer stehen in der Datei „/etc/passwd“, die angelegten Gruppen in „/etc/group“.

8.3 Editieren von Text-Dateien

Auf dem Socketcomputer ist das Editorprogramm „nano“ installiert. Mit diesem können z. B. Konfigurationdateien oder Scripte geändert werden, ohne das Dateien zunächst auf dem PC editiert und dann übertragen werden müssen.

```
Aufruf: nano [Pfad][Dateiname]
```

Soll z. B. die IP-Adresse des Moduls geändert werden, so ist die Datei „interfaces“ im Ordner „/etc/network/“ zu modifizieren (Abb. 19).

```
nano /etc/network/interfaces
```

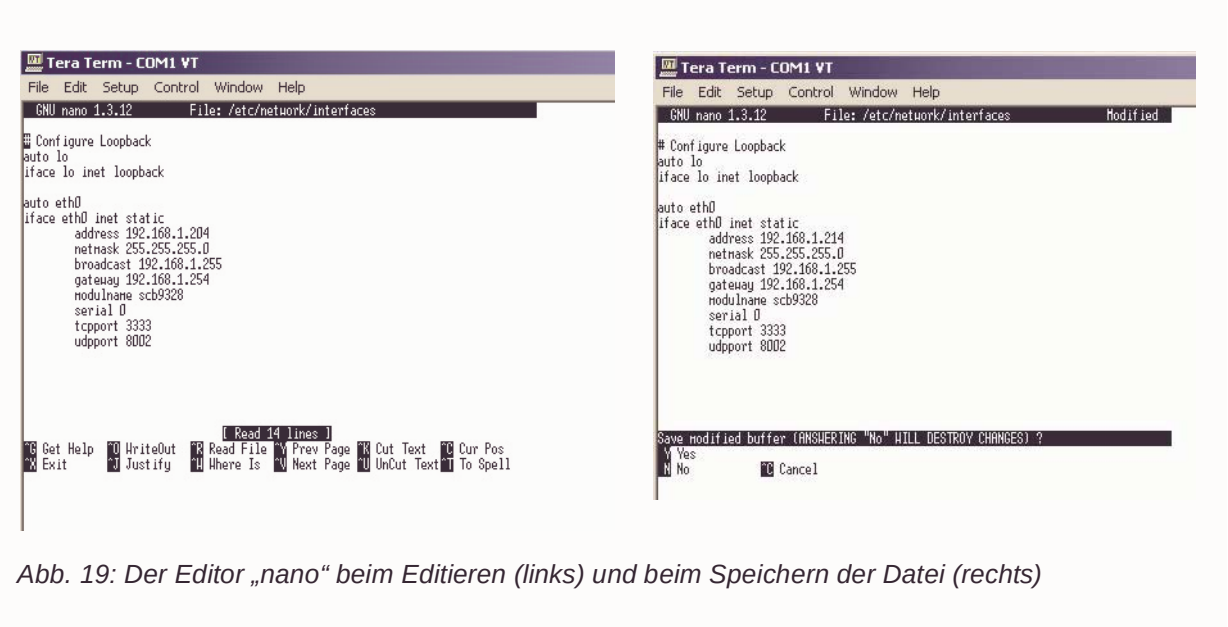


Abb. 19: Der Editor „nano“ beim Editieren (links) und beim Speichern der Datei (rechts)

Nach dem Aufruf erscheint im Terminal eine einfache Oberfläche ähnlich zu Abb. 19. Jetzt kann die gewünschte Änderung an der IP-Adresse durchführt und das Programm mit der Tastenkombination „Control-x“ beendet werden.

Die Frage „Save modified buffer ?“ mit „y“ bestätigen (Alternativ mit „c“ den Vorgang abbrechen oder mit „n“ die Datei unverändert lassen) und mit „Return“ die Datei unter demselben Namen speichern.

8.4 Softwareupdate per WinSCP

Um die Modulsoftware oder Konfigurationsdateien zu aktualisieren, kann unter MS Windows z. B. das Tool „WinSCP“ eingesetzt werden. WinSCP ist ein grafischer Open Source SFTP und FTP Client für Windows, der auch das SCP-Protokoll unterstützt. Er erlaubt einen geschützten Daten- und Dateitransfer zwischen verschiedenen Rechner.

Um einen Verbindung zum Modul aufzubauen, ist wie folgt vorzugehen:

Nach dem Starten des Programms eine neue Session mit „root@IP-Adresse“ anlegen, wobei die IP-Adresse die IP des Moduls ist (hier 192.168.1.204).

Als Nutzer „root“ und als Passwort „pass“ eintragen (Abb. 20).

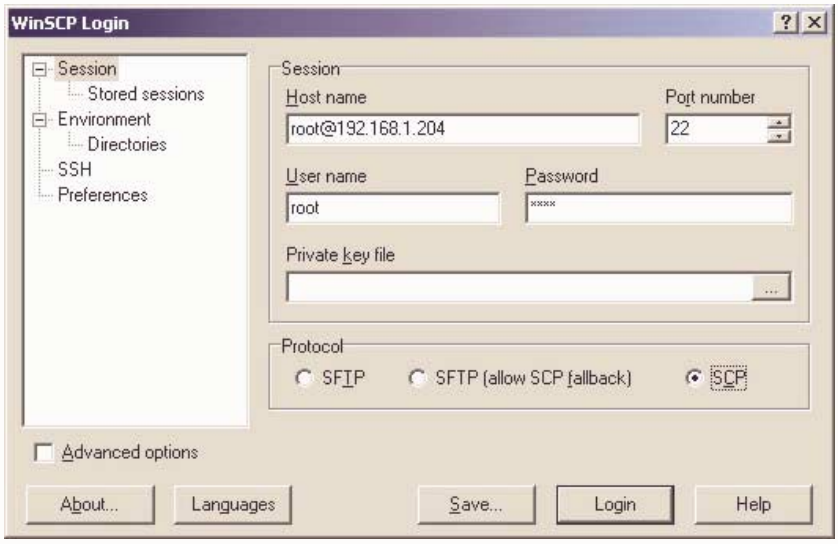


Abb. 20: Anlegen einer neuen Session unter „WinSCP“

Mit dem Button „Save“ werden die Einstellungen gespeichert. Durch Drücken des Button „Login“ wird die Verbindung zum Modul aufgebaut. Es erscheint eine Benutzeroberfläche ähnlich zu Abb. 21.

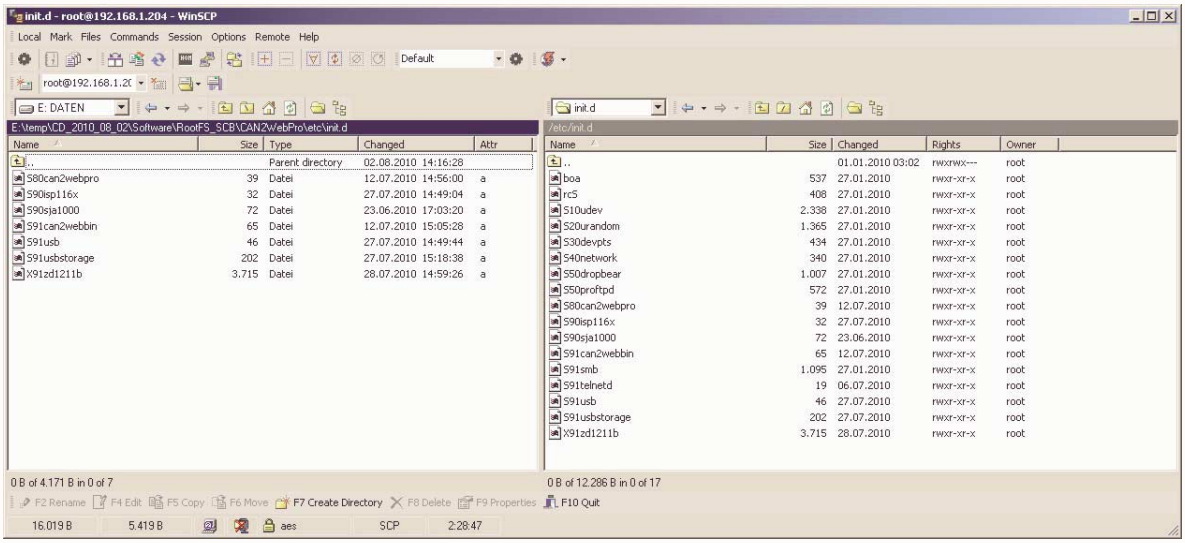


Abb. 21: Beispiel der Dateiansicht von „WinSCP“

Hier können Dateiverzeichnisse auf Seiten des PCs und Moduls dargestellt und Dateien oder auch ganze Verzeichnisse kopiert, aktualisiert oder auch gelöscht werden.

Weiterhin lassen sich auch einfach Dateien auf dem Socketcomputer editieren. Dazu einfach die gewünschte Datei mit der rechten Maustaste markieren und den Eintrag „Edit“ wählen. Es erscheint ein Editorfenster mit der ausgewählten Datei (Abb. 22).

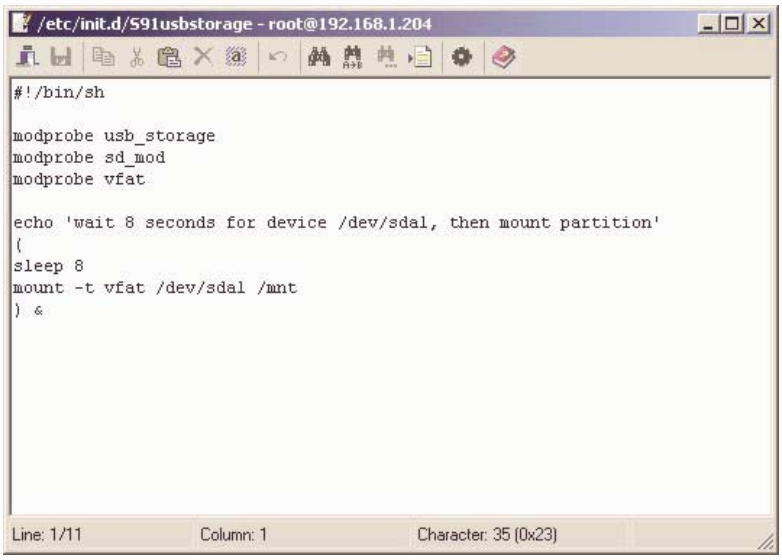


Abb. 22: Editieren einer Datei mit „WinSCP“

Nun können die gewünschten Änderungen vorgenommen werden und die Datei gespeichert werden.

Die Open Source Software „WinSCP“ kann unter den angegeben Nutzungsbedingungen unter http://winscp.net/eng/docs/lang:de herunter geladen werden. Dort finden sich auch umfassende Erklärungen der Funktionalität sowie Hilfen zu dem Programm.

8.5 Diverse Linux-Kommandos

8.5.1 Laden und Entladen von Linux-Kernel-Treiber

Die nachfolgenden Befehle beziehen sich auf die Linux-Kernel-Treiber.

Tabelle 25 :Auflistung einiger Linux-Treiber-Kommandos

Kommando	Beschreibung
lsmod	Auflistung aller laufenden Linux-Kernel-Treiber
insmod „Treibername“ z.B: „insmod can_driver.ko“	Installiere Treiber
rmmod „Treibername“ z.B: „insmod can_driver.ko“	Deinstalliere Treiber
Eintrag in Datei etc/init.d: insmod „Treibername“ z.B. „insmod /can_driver.ko“	Der Treiber wird automatisch nach dem Start von Linux gestartet, wenn er in einer der Sxx-Dateien in etc/init.d eingetragen wird.

Kommando	Beschreibung
mknod path c major minor z.B.: „mknod /dev/can c 250 0“	Anlegen eines Device-Files für den CAN-Treiber „c“ für Character-Device, 250 = Device Major Nr., 0=Device Minor Nr.
depmod	Abhängigkeiten für ladbare Kernelmodlule aktualiseren.

8.5.2 Prozesse anzeigen und beenden

Mit den nachfolgenden Kommandos können Prozesse angezeigt und beendet werden:

Tabelle 26 :Prozesse anzeigen und beenden

Kommando	Beschreibung
ps ax	Auflistung aller laufenden Prozesse
kill „Prozessnummer“ z.B.: kill 183	Beenden eines Prozesses mit der Nr. „Prozessnummer“ Beenden des Prozesses mit der Nr. 183
kill -9 „Prozessnummer“ z.B.: kill 183	Beenden eines Prozesses mit der Nr. „Prozessnummer“ Beenden des Prozesses mit der Nr. 183 Anmerkung: *kill -9* kann immer dann verwendet werden falls *kill* alleine den Prozess nicht beendet.

8.5.3 Befehle für das Netzwerkinterface

Mit den nachfolgenden Kommandos können die Netzwerkinterfaces aktiviert, deaktiviert und konfiguriert werden.

Tabelle 27 :Auflistung einiger Netzwerk Kommandos

Kommando	Beschreibung
ifconfig	Auflistung aller Netzwerkschnittstellen
ifconfig „Interface“ „IP“ z.B: ifconfig eth0 192.168.1.123	IP-Adresse des Interfaces „Interface“ setzen
ifup „Interfacename“ z.B.: „ifup eth0“ oder „ifup wlan0“	Interface installieren/starten z.B.: Ethernet Interface0 oder WLAN-Interface 0
ifdown „Interfacename“ z.B.:“ ifdown etho“ oder „ifdown wlan0“	Interface deinstallieren/stoppen z.B.: Ethernet Interface0 oder WLAN-Interface 0
route	Ausgabe der Routing Tabelle
....	

8.5.4 Befehle für das Dateisystem

Die nachfolgenden Befehle beziehen sich auf das Dateisystem.

Tabelle 28 :Auflistung einiger Datei-Kommandos

Kommando	Beschreibung
ls z. B. ls -la ls -laR	Auflistung der Dateien und Ordner im aktuellen Verzeichnis zusätzliche Infos ausgeben rekursiv alle Unterordner mit auflisten
pwd	Arbeistverzeichnis ausgeben
df	Infos zum Filesystem (Partitionen) anzeigen
mount z. B.: mount -t jffs2 /dev/mtdblock5 /mnt2	Dateisystem mounten Das CAN2Web-Advanced mit WLAN (SCB9324) verfügt über eine 512 MB-Flash Disk. Davon sind standarmäßig 64MB in /mnt gemountet. Dieser Befehl dient dazu, die zweite (448 MB) in /mnt2 zu mounten.
mkdir „Verzeichnis“ z. B.: mkdir /mnt2	Unterordner anlegen Verzeichnis „mnt2“ anlegen
.....	

8.5.5 Kommandos für den Bootloader

Die nachfolgende Liste enthält einige Befehle des Bootloader u-boot. Ein komplette Dokumentation zum Bootloader findet sich im Web unter http://u-boot.sourceforge.net/ auf der u-boot Hompage.

Tabelle 29 :Auflistung einiger Bootloader-Kommandos

Kommando	Beschreibung
printenv	Enviroment ausgeben
setenv „Parametername“= „Wert“ z.B.: setenv bootdelay=3	Parameter in Enviroment ändern z.B.: Bootzeit auf 3 Sekunden setzen
saveenv	Enviroment (im Flash) speichern
boot	Linux starten

8.5.6 Datei kopieren mit SCP

Tabelle 30 :Kopieren mit SCP

Kommando	Beschreibung
scp filename user@IP:path z.B.: scp mytext.txt root@192.168.1.123:/usr/local/bin	Kopieren von Dateien mit SCP Kopiert die Datei mytext.txt zum Rechner mit der IP 192.168.1.123 in das Verzeichnis /usr/local/bin
.....	

8.5.7 Loggen von Ereignissen

Die nachfolgenden Befehle gestatten es zusätzliche Debuginformationen zu erhalten.

Tabelle 31 :Loggen von Ereignissen

Kommando	Beschreibung
cat /proc/interrupts	Liste der aktiven Interrupts
klogd	Kernel-Log Dämon, schreibt als Hintergrund-prozess Kernel Logdaten in die Datei /var/log/messages
syslogd	System-Log Dämon, schreibt als Hintergrund-prozess Kernel Logdaten in die Datei /var/log/messages
tail -f „Datei“ z. B. tail -f /var/log/messages&	Ausgabe von neuen Eintägen in Datei „Datei“ Ausgabe von neuen Einträgen in die Logddatei /var/log/messages

8.6 Liste aller CAN2Web-Dateien und Verzeichnisse

Die nachfolgende Liste gibt einen Überblick aller CAN2Web-spezifischen Dateien und Verzeichnisse auf den CAN2Web-Modulen.

```
Linux: Version 2.6.31
Date:  16.12.2011

CAN2Web-Advanced:
-----------------
/etc/init.d                         // content: shell start scripts
- S80can2webadv                     // shell start script for board package for driver
                                    // scb9320-can2webadvanced.ko
- S90sja1000                        // shell start script for CAN driver
- S90uart16c550                     // shell start script for serial interface
- S91can2webbin                     // shell start script for can2web.exe and udpconfig.exe
- S91telnetd                        // shell start script for telnet
- S90bgw211                         // shell start script for WLAN with WPA supplicat
- S90adhoc                          // shell start script for WLAN in ad-hoc mode with iwconfig

CAN2Web-Professional:
---------------------
/etc/init.d                         // content: shell start scripts
- S80can2webpro                     // shell start script for board package for driver
```


```
                                    // scb9320-can2webpro.ko
- S90isp116x                        // shell start script for ISP116x driver
- S90sja1000                        // shell start script for CAN driver
- S91can2webbin                     // shell start script for can2web.exe and udpconfig.exe
- S91usb                            // shell start script for USB
- S91usbstorage                     // shell start script for USB mass storage device
- S91zd1211b                        // shell start script for USB-WLAN driver

CAN2Web-Advanced:
-----------------
/lib/modules/2.6.31/kernel
-  8250.ko                         // kernel driver for serial RS232 interface
-  scb9320-can2webadvanced.ko      // CAN2Web-Advanced hardware specific driver


CAN2Web-Professional:
--------------------
/lib/modules/2.6.31/kernel
- cfg80211.ko
- fat.ko
- firmware_class.ko
- isp116x-hcd.ko
- mac80211.ko
- scb9320-can2webpro.ko
- scb9320-io-dev.ko
- scsi_mod.ko
- scsi_wait_scan.ko
- sd_mod.ko
- usb-storage.ko
- usbcore.ko
- vfat.ko
- zd1211rw.ko

-------------------
All CAN2Web-Moduls:
-------------------

/lib/modules/2.6.31/kernel/extra
- can_driver.ko                     // CAN kernel driver for CAN2Web's

/usr/bin                            // content: binaries & configuration
- udpsocket.exe                     // UDP-Configuration & Identification SW
- can2web.exe                       // CAN/Ethernet-Gateway SW
- can2web.cfg                       // configuration file (ASCII-Text) for CAN/Ethernet-Gateway SW
                                    // when file is missing it will be generated automatically by
                                    // can2web.exe


/usr/bin/webpages                   // content: webpages & applet for CAN2Web webinterface
- CAN2WebVisu.html                  // webpage for applet
- CAN2WebVisu.class                 // class for visualization
- Network$CheckNetwork.class        // class for visualization
- Network$NetworkThread.class       // class for visualization
- Network.class                     // class for visualization
- CAN2Web_Handbuch.pdf              // manual for CAN2Web-Advanced & CAN2Web-Professioanl

- adresse.gif                       // several HTML pages, images, etc. .........
- bgbilder.gif
- bgheads.gif
- bgnavi.gif
- bilder__advanced.htm
- bilder__professional.htm
- bottom.gif
- can2web_professional_gehaeuse.jpg
- can2web_professional_leiterplatte_2a_klein.jpg
- can2web_schema.gif
- can2web_schema2.gif
- can2webtab.js                     // CAN2Web tabs lib (webpages using AJAX)
- can_frm.htm
- can_frm_2.htm
- can_menu.htm
- canfrm.htm
- com1_frm.htm
- com1_frm_2.htm
- com1menu.htm
- favicon.ico
- help_frm.htm
- help_frm_2.htm
- hlp_allg.htm
- hlp_cfg.htm
- hlp_i_allg.htm
- hlp_i_cfg.htm
- hlp_i_manual.htm
```


```
- hlp_i_pin.htm
- hlp_menu.htm
- hlp_pin__advanced.htm
- hlp_pin__professional.htm
- home__advanced.htm
- home__professional.htm
- index.htm
- ip_frm.htm
- ip_frm_2.htm
- ip_menu.htm
- leer.htm
- leer2.htm
- loadstrm.htm
- main_frm_2.htm
- menu.htm
- menu1.htm
- menu2.htm
- othr_frm.htm
- othr_frm_2.htm
- othrmenu.htm
- pipe.gif
- prototype.js                  // AJAX lib
- punkt1.gif
- spher128.gif
- strm_frm.htm
- strm_frm_2.htm
- strmmenu.htm
- synertronixx.css              // general stylesheet for synertronixx webpages
- tablebg.jpg
- tabs.css                      // stylesheet for CAN2Web tabs (webpages using AJAX)
- top.htm
- top.jpg
- tux.gif                       // the lovely Linux penguin TUX
```

8.7 WLAN-Konfiguration für CAN2Web (Internes Interface)

Dieser Abschnitt bezieht sich auf die folgenden Produkte mit dem Socketcomputer SCB9324 mit integriertem WLAN-Interface:

- CAN2Web-Advanced WLAN und

- CAN2Web-Professional WLAN II.

8.7.1 WLAN-Konfiguration über WPA Supplicant

Zur Konfiguration des WLAN wird u. a. das Programm „WPA supplicant“ verwendet. Dieses führt die notwendigen Verschlüsselungen und Anmeldungen durch.

Zum Starten des WLAN-Interfaces ist das Script „X90bgw211“ im Verzeichnis /etc/init.d/ auszuführen. Soll das Script beim Systemstart automatisch ausgeführt werden, so muss es in „S90bgw211“ umbenannt werden.

```
mv X90bgw211 S90bgw211
```

Das Script deaktiviert das Ethernet Interface „eth0“,

```
ifconfig eth0 down
```

startet den WLAN-Treiber "ga_linuxdrv_211_imx_sdio",

```
modprobe ga_linuxdrv_211_imx_sdio
```

legt die IP-Adresse des WLAN für eth1 fest

```
ifconfig eth1 192.168.1.205
```

(Um die WLAN-IP einzustellen, ist der Eintrag „IPADDRESS“ im Script entsprechend zu ändern.)

und führt danach „wpa_supplicant“ aus.

```
wpa_supplicant -Dphilips -ieth1 -c/etc/wpasupplicant.cfg -B
```

Dieser wird über die Datei „wpasupplicant.cfg“ im Verzeichnis /etc/ konfiguriert.

Die Datei sieht für die WPA-Verschlüsselung etwa folgendermaßen aus:

```
# wpa-psk / bgw211

ctrl_interface=/var/run/wpa_supplicant
ap_scan=2 # scan never finishes with ap_scan=1
fast_reauth=0

network=
{
  ssid="myssid"
  key_mgmt=WPA-PSK
  pairwise=CCMP
  group=CCMP
  psk="mypassword"
}
```

Um das CAN2Web-Advanced bei einem Access Point (AP) anzumelden, sind die Einträge „ssid“ und „psk“ entsprechend der AP-Einstellungen zu ändern. Sofern der AP über einen MAC-Filter verfügt, ist dort die WLAN-MAC-Adresse des SCB9324 einzutragen.

Die Socketcomputer vom Typ SCB9324 verfügen über zwei MAC-Adressen. Die Erste (siehe Aufkleber auf SCB9324) wird für das Ethernet-Interface, die Zweite für das WLAN-Interface verwendet. Die MACs sind fortlaufend numeriert also z. B.

```
00 0C 7F – 00 08 21 (Ethernet, siehe Aufkleber)
00 0C 7F – 00 08 22 (WLAN)
```

Weitere Informationen zum „wpa_supplicant“ und dessen Konfiguration finden sich z. B. auch im Internet unter

http://en.wikipedia.org/wiki/Wpa_supplicant

http://hostap.epitest.fi/wpa_supplicant/

http://hostap.epitest.fi/gitweb/gitweb.cgi?p=hostap.git;a=blob_plain;f=wpa_supplicant/wpa_supplicant.conf

8.7.2 WLAN-Konfiguration über iwconfig (für Ad-hoc Modus)

Es ist möglich zwei bzw. mehrere CAN2Web-Advanced-Module auch direkt über WLAN ohne einen Access Point im sog. „Ad-hoc“-Modus zu verbinden.

Zum Starten des WLAN-Interfaces im „Ad-hoc“-Modus ist das Script „X90adhoc“ im Verzeichnis /etc/init.d/ auszuführen. Soll das Script beim Systemstart automatisch ausgeführt werden, so muss es in „S90adhoc“ umbenannt werden.

```
mv X90adchoc S90adhoc
```

Das Script deaktiviert das Ethernet Interface „eth0“,

```
ifconfig eth0 down
```

startet den WLAN-Treiber "ga_linuxdrv_211_imx_sdio"

```
modprobe ga_linuxdrv_211_imx_sdio
```

und führt danach „iwconfig“ im „Ad-hoc“-Modus mit der SSID „can_net“ aus.

```
iwconfig eth1 mode ad-hoc essid can_net
```

Um die WLAN-IP einzustellen, ist der Eintrag „IPADDRESS“ entsprechend zu ändern. Die WLAN-IP-Adressen der CAN2Webs müssen unbedingt unterschiedlich sein.

Die Module sind jetzt im „Ad-hoc“-Modus verbunden. Zum Test kann z. B. ein „ping“ ausgeführt werden. Hat Modul 1 z. B. die IP-Adresse 192.168.1.205 und Modul 2 die Adresse 192.168.1.215 und pingt man von Modul 1 Modul 2 an, so erhält man z. B. folgende Ausgabe:

```
ping 192.168.1.215
Ausgabe:
PING 192.168.1.215 (192.168.1.215): 56 data bytes
64 bytes from 192.168.1.215: seq=0 ttl=64 time=8.6 ms
64 bytes from 192.168.1.215: seq=1 ttl=64 time=7.7 ms
64 bytes from 192.168.1.215: seq=2 ttl=64 time=6.0 ms

--- 192.168.1.215 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 6.0/7.4/8.6 ms
```

Anmerkung: Soll die Gateway-Funktionalität der CAN2Webs verwendet werden, so sind im Verbindungsmanager (siehe auch Kap. 4.3 und Kap. 5.5.9) eines der beiden Module die entsprechenden Einstellungen vorzunehmen.

Weitere Informationen zum Linux-Kommando „iwconfig“ und dessen Konfiguration finden sich z. B. auch im Internet unter

http://linuxcommand.org/man_pages/iwconfig8.html.

Warnung:

Bei der obigen Konfiguration von „iwconfig“ findet keine Verschlüsselung oder eine Passwortabfrage statt. Die CAN2Web-Module sind „offen“ und es können sich auch andere WLAN-Teilnehmer auf die Module aufschalten und diese ggf. beliebig manipulieren. Es ist daher genau zu prüfen, ob dieser Modus verwendet werden kann.

8.8 WLAN-Konfiguration für CAN2Web (externes Interface)

Dieser Abschnitt bezieht sich auf die folgenden Produkte CAN2Web-Professional unter Verwendung des externem USB-WLAN-Interface:

- CAN2Web-Professional WLAN (mit SCB9328) und

- CAN2Web-Professional WLAN II (mit SCB9324).

8.8.1 WLAN-Konfiguration über WPA Supplicant und iwconfig

Zur Konfiguration des WLAN wird u. a. das Programm „WPA supplicant“ verwendet. Dieses führt die notwendigen Verschlüsselungen und Anmeldungen (WPA/WPA2) durch.

Zum Starten des WLAN-Interfaces ist das Script „X91zd1211b“ im Verzeichnis /etc/init.d/ auszuführen. Soll das Script beim Systemstart automatisch ausgeführt werden, so muss es in „S91zd1211b“ umbenannt werden.

```
mv X91zd1211b S91zd1211b
```

Das Script deaktiviert die Interfaces „eth0“, „eth1“ und „wlan0“ und startet danach (wieder) alle Treiber für „wlan0“.

Um die WLAN-IP einzustellen, ist der Eintrag „IP“ im Script entsprechend zu ändern.

Danach führt es „wpa_supplicant“ aus.

```
wpa_supplicant -D wext -i ${IFACE} -c ${WPACONF} -B -dd
```

Dieser wird über die Datei „wpasupplicant.cfg“ im Verzeichnis /etc/ konfiguriert.

Die Datei sieht für die WPA-Verschlüsselung etwa folgendermaßen aus:

```
# wpa-psk / zd1211b

ctrl_interface=/var/run/wpa_supplicant
ap_scan=1
fast_reauth=0

network=
{
  ssid="ssid"
  key_mgmt=WPA-PSK
  pairwise=CCMP
  group=CCMP
  psk="password"
}
```

Um das CAN2Web-Professional bei einem Access Point (AP) anzumelden, sind die Einträge „ssid“ und „psk“ entsprechend der AP-Einstellungen zu ändern. Sofern der AP über einen MAC-Filter verfügt, ist dort die WLAN-MAC-Adresse des USB-WLAN-Sticks einzutragen.

Weitere Informationen zum „wpa_supplicant“ und dessen Konfiguration finden sich z. B. auch im Internet unter

http://en.wikipedia.org/wiki/Wpa_supplicant

http://hostap.epitest.fi/wpa_supplicant/

http://hostap.epitest.fi/gitweb/gitweb.cgi?p=hostap.git;a=blob_plain;f=wpa_supplicant/wpa_supplicant.conf

Anmerkung:

Wird die Variable „MODE“ von „wpasupplicant“ auf „iwconfig“ umgestellt, so läßt sich auch eine WEP-Verschlüsselung einstellen. Dieser Modus sollte jedoch nicht verwendet werden, da er nur einen sehr schwachen Schutz bietet und leicht überwunden werden kann. Um eine Anmeldung am AP zu ermöglichen, sind die Variablen „ESSID“ und „ENC“ entsprechend der Einstellungen des APs zu setzen.

8.9 Serielle Schnittstelle als Linux-Konsole (de)aktivieren

8.9.1 Konsole des CAN2Web-Advanced

Dieser Abschnitt bezieht sich nur auf das CAN2Web-Advanced. Wird die serielle Schnittstelle (CON4, 9-pol. SubD-Buchse) nicht für die Applikation benötigt, so kann sie auch als Linux-Debug-Schnittstelle (Linux-Konsole) konfiguriert werden. Dazu ist in der Datei /etc/inittab der Eintrag

```
ttyS0::respawn:/sbin/getty -L ttyS0 115200 vt100
```

vorzunehmen. Zum Deaktivieren ist der Eintrag wieder zu entfernen.

Wird die serielle Schnittstelle benötigt, kann eine Anbindung an die Linux-Konsole auch über CON5 (10-poliger Pfostenverbinder) erfolgen. Dazu bietet synertronixx einen speziellen RS232-Adapter (Pegelwandler) an.
Diese Schnittstelle ist immer als Bootloader/Linux-Konsole vorgesehen.

8.9.2 Konsole des CAN2Web-Professional

Dieser Abschnitt bezieht sich nur auf das CAN2Web-Professional. Die serielle Schnittstelle des CAN2Web-Professional-Moduls wird nach dem Einschalten (Power-Up) zunächst vom Bootloader uboot belegt (115200 Baud, 8 Datenbit, 1 Stoppbit, keine Parität, kein Handshake). Der Bootloader sendet über die Schnittstelle diverse Startmeldungen. Nach dem Starten von Linux dient sie als Linux-Konsole.

Diese Funktionalität lässt sich wie folgt deaktivieren:

Es müssen zunächst in der Datei "/etc/inittab" die beiden Zeilen nach dem Kommentar

```
# Put a getty on the serial port
```

in

```
# ttySMX0::respawn:/sbin/getty -L ttySMX0 115200 vt100
# ttymxc0::respawn:/sbin/getty -L ttymxc0 115200 vt100
```

geändert werden (mit „#“ wird die Zeile auskommentiert) und danach gespeichert werden. Damit ist die Konsolenfunktion abgeschaltet. Soll die Konsole wieder aktiviert werden, ist das Kommentarzeichen zu entfernen.

Als zweiten Schritt müssen noch die Debug-Meldungen deaktiviert werden. Dazu kann z. B. ein Terminalprogramm wie Teraterm oder Putty verwendet werden. Eine freie seri-

elle Schnittstelle des PCs ist dazu mit der RS232-Schnittstelle des CAN2Web-Moduls zu verbinden.

Dann ist zunächst ein Neustart des Moduls z. B. über den Befehl "reboot" oder durch Drücken des Reset-Tasters auf der Rückseite des Gehäuses auszuführen. Der Bootloader startet und wartet darauf, dass innerhalb von 3 Sekunden ein Zeichen über die Schnittstelle empfangen wird. Ist dies der Fall wird der Bootvorgang unterbrochen und die Konsole des Bootloaders wird aktiv.

Durch Eingabe des Befehls "printenv" werden die aktuellen Booteinstellungen für das Linux ausgegeben. Unter anderem steht dort eine Zeile:

```
...
bootargs=console=ttymxc0,115200n8 root=/dev/mtdblock3 rootfstype=jffs2
mtdparts=physmap-flash.0:128k(U-boot)ro,128k(U-boot_env),2m(kernel),-
(root)
....
```

Diese wird über den Befehl "setenv"

```
setenv bootargs console=null root=/dev/mtdblock3 rootfstype=jffs2 mtd-
parts=physmap-flash.0:128k(U-boot)ro,128k(U-boot_env),2m(kernel),-(root)
```

geändert. Zur Kontrolle kann nochmal mit "printenv" kontrolliert werden, ob der Eintrag richtig übernommen wurde. Dort sollte jetzt folgende Zeile stehen:

```
bootargs=console=null root=/dev/mtdblock3 rootfstype=jffs2 mtdparts=phys-
map-flash.0:128k(U-boot)ro,128k(U-boot_env),2m(kernel),-(root)
```

Die Einstellungen werden mit "saveenv" gespeichert. Die Linux-Debugausgaben sind damit dauerhaft deaktiviert. Der Bootloader selbst sendet jedoch beim Starten noch Meldungen!

Mit dem Befehl "reset" kann das System neu gestartet werden. NACH dem Starten von Linux werden keine Meldungen mehr auf der seriellen Konsole erscheinen. Die serielle Schnittstelle steht jetzt für beliebige Applikationen zur Verfügung.

Um die Debugausgaben wieder zu aktivieren, kann obiger Eintrag mit dem Befehl

```
setenv bootargs console=ttymxc0,115200n8 root=/dev/mtdblock3 rootfs-
type=jffs2 mtdparts=physmap-flash.0:128k(U-boot)ro,128k(U-
boot_env),2m(kernel),-(root)
```

geändert werden.

9 Technische Daten CAN2Web Advanced

9.1 Elektrische und weitere Daten

Tabelle 32 :Technische Daten für CAN2Web-Advanced-Module

Parameter	Wertebereich/Anmerkung
Versorgungsspannung	8...30V DC
Leistungsaufnahme	typisch 3 Watt
Betriebstemperatur	0 °C...55 °C
Lagertemperatur	-25 °C ... 70 ° C
Luftfeuchtigkeit	10...85%, nicht kondensierend
Abmessungen	Leiterplatte ca. 100mm*54mm*30mm (Länge*Breite*Höhe)
Gewicht	Leiterplatte ca. 70g

9.2 Anschluss- und Pinbelegung

9.2.1 Übersicht der Anschlüsse

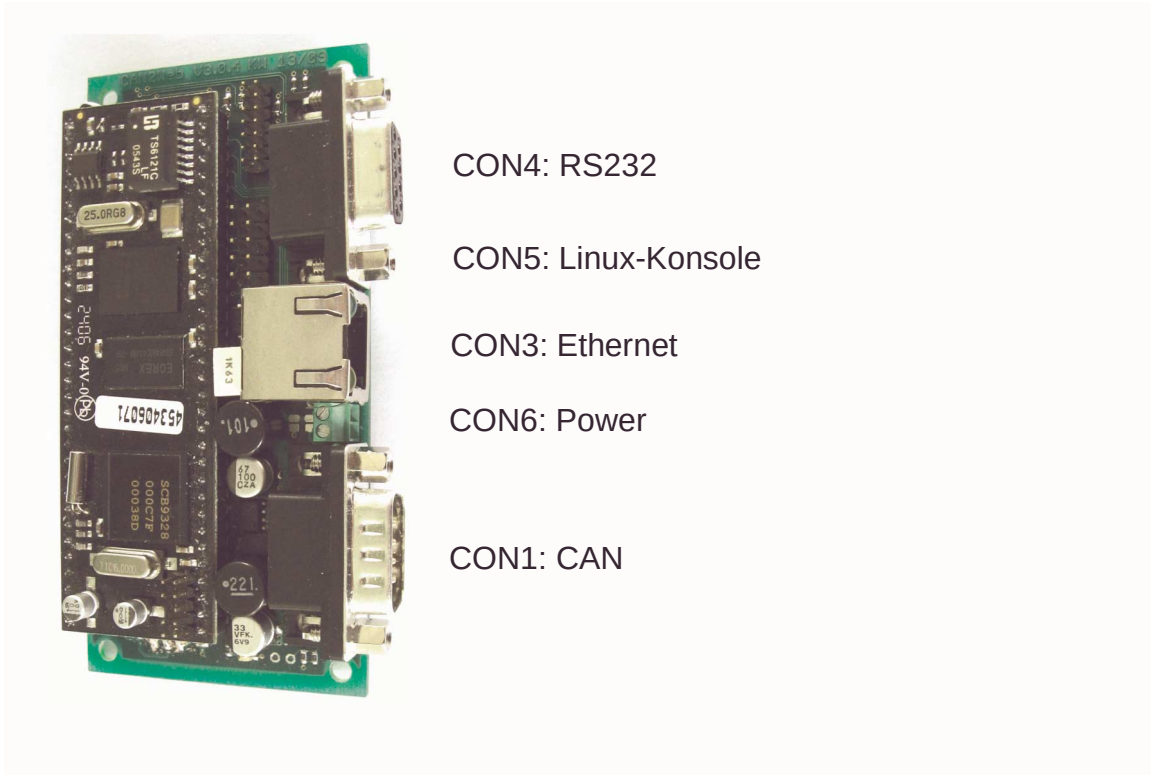


Abb. 23: Steckverbinder CAN2Web-Advanced

9.2.2 Pinbelegung CON1 (CAN-Bus, 9p male)

Pin	I/O	Name	Beschreibung
1	--	NC	--
2	I/O	CANL	max. 1 MBit/s
3	--	0V	Masse CAN-Bus
4	--	NC	--
5	--	NC	--
6	--	NC	--
7	I/O	CANH	max. 1 MBit/s
8	--	NC	--
9	--	NC	-nicht verbunden wenn Löt-pad SJ2 offen ist - verbunden mit V+ wenn Lötpad SJ2 geschlossen ist

9.2.3 Pinbelegung CON3 (Ethernet 10/100BaseT, 8p RJ45)

Pin	I/O	Name	Beschreibung
1	Out	Tx+	10/100MBit Transmit
2	Out	Tx-	10/100MBit Transmit
3	In	Rx+	10/100MBit Receive
4	--	NC	NC
5	--	NC	NC
6	In	Rx-	10/100MBit Receive
7	--	NC	NC
8	--	NC	NC

9.2.4 Pinbelegung CON4 (RS232C, 9p male)

Pin	I/O	Name	Beschreibung
1	--		NC
2	Out	RxD	max. 115200 Baud
3	In	TxD	max. 115200 Baud
4	--		NC
5	GND	GND	GND
6	--		NC
7	In	RTS	max. 115200 Baud
8	Out	CTS	max. 115200 Baud
9	--		NC

9.2.5 Pinbelegung CON5 (Linux-Konsole, 10p)

Dieser Steckverbinder dient ausschließlich zur Anbidnung des „RS232-Adapters V1.1.1“ (Pegelwandler) von synertronixx.

Pin	I/O	Name	Beschreibung
1	Out	3,3V	3,3V Versorgungsspannung für externen Adapter
2	--	NC	--
3	--	NC	--
4		RxD	max. 115200 Baud
5		TxD	max. 115200 Baud
6	--	NC	--
7		RTS	max. 115200 Baud
8		CTS	max. 115200 Baud
9	--	NC	--
10	Out	GND	GND

9.2.6 Pinbelegung CON6, (Power Supply, 2polig)

Pin	I/O	Name	Beschreibung
1	In	V+	8V...30V-DC
2	In	GND	GND

10 Technische Daten CAN2Web-Professional

10.1 Elektrische und weitere Daten

Tabelle 33 :Technische Daten für CAN2Web-Professional-Module

Parameter	Wertebereich/Anmerkung
Versorgungsspannung	8...30V DC
Leistungsaufnahme	typisch 3 Watt (ohne angschlossenes USB-Device)
Betriebstemperatur	0 °C...55 °C
Lagertemperatur	-25 °C ... 70 ° C
Luftfeuchtigkeit	10...85%, nicht kondensierend
Abmessungen	Leiterplatte ca. 115mm*100mm*30mm (Länge*Breite*Höhe) Alu-Gehäuse ca 117mm*109mm*35mm (Länge*Breite*Höhe)
Gewicht	Leiterplatte ca. 120g, Leiterplatte+Gehäuse ca. 306g

10.2 Anschluss- und Pinbelegung

10.2.1 Übersicht der Anschlüsse

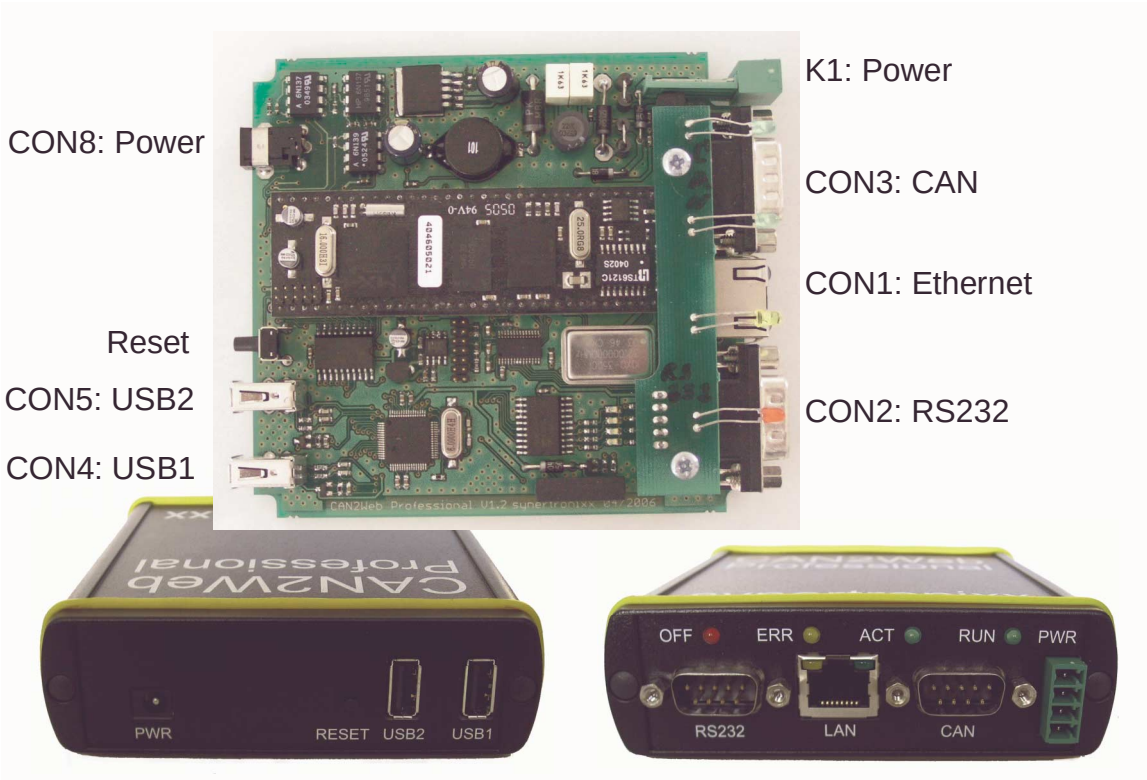


Abb. 24: Steckverbinder CAN2Web-Professional

10.2.2 Pinbelegung CON1 (Ethernet 10/100BaseT, 8p RJ45)

Pin	I/O	Name	Beschreibung
1	Out	Tx+	10/100MBit Transmit
2	Out	Tx-	10/100MBit Transmit
3	In	Rx+	10/100MBit Receive
4	--	NC	NC
5	--	NC	NC
6	In	Rx-	10/100MBit Receive
7	--	NC	NC
8	--	NC	NC

10.2.3 Pinbelegung CON2 (RS232C, 9p male)

Pin	I/O	Name	Beschreibung
1	--		NC
2	Out	RxD	max. 115200 Baud
3	In	TxD	max. 115200 Baud
4	--		NC
5	GND	GND	GND
6	--		NC
7	In	RTS	max. 115200 Baud
8	Out	CTS	max. 115200 Baud
9	--		NC

10.2.4 Pinbelegung CON3 (CAN-Bus, 9p male)

Pin	I/O	Name	Beschreibung
1	--	NC	--
2	I/O	CANL	max. 1 MBit/s
3	--	0V	Masse CAN-Bus, galvanisch getrennt
4	--	NC	--
5	--	NC	--
6	--	NC	--
7	I/O	CANH	max. 1 MBit/s
8	--	NC	--
9	--	NC	--

10.2.5 Pinbelegung CON4, CON5 (USB, 4polig)

Pin	I/O	Name	Beschreibung
1	Out	V+	5V DC
2	In/Out	USB D-	USB Host
3	In/Out	USB D+	USB Hostl
4	Out	GND	GND

10.2.6 Pinbelegung K1 (Power Supply, 4polig)

Pin	I/O	Name	Beschreibung
1	In	GND	GND (intern verbunden mit Pin 3)
2	In	Supply +	+8V...30V-DC (intern verbunden mit Pin 4)
3	In	GND	GND (intern verbunden mit Pin 1)
4	In	Supply +	8V...30V-DC (intern verbunden mit Pin 2)

Anmerkung: Orientierung von Pin1 zur Gehäuseoberseite gerichtet

10.2.7 Pinbelegung CON8, (Power Supply, 2polig für Steckernetzteil)

Pin	I/O	Name	Beschreibung
1	In	V+	8V...30V-DC
2	In	GND	GND

Anmerkung: Pin 1 innen liegend (Dorn)

11 Impressum

Dieses Dokument ist Teil des Online-Angebotes der synertronixx GmbH.

11.1 Kontakt

synertronixx GmbH
Lange Laube 22
30159 Hannover

Tel.: +49 (0) 511 / 262 999 - 0
Fax: +49 (0) 511 / 262 999 - 29
Email: info @ synertronixx.de
Internet: www.synertronixx.de

11.2 Verweise und Links

Bei direkten oder indirekten Verweisen auf fremde Internetseiten ("Hyperlinks"), die außerhalb des Verantwortungsbereiches des Autors liegen, würde eine Haftungsverpflichtung ausschließlich in dem Fall in Kraft treten, in dem der Autor von den Inhalten Kenntnis hat und es ihm technisch möglich und zumutbar wäre, die Nutzung im Falle rechtswidriger Inhalte zu verhindern. Der Autor erklärt hiermit ausdrücklich, dass zum Zeitpunkt der Linksetzung keine illegalen Inhalte auf den zu verlinkenden Seiten erkennbar waren. Auf die aktuelle und zukünftige Gestaltung, die Inhalte oder die Urheberschaft der gelinkten/verknüpften Seiten hat der Autor keinerlei Einfluss. Deshalb distanziert er sich hiermit ausdrücklich von allen Inhalten aller gelinkten/verknüpften Seiten, die nach der Linksetzung verändert wurden. Diese Feststellung gilt für alle gesetzten Links und Verweise innerhalb dieses Dokumentes. Für illegale, fehlerhafte oder unvollständige Inhalte und insbesondere für Schäden, die aus der Nutzung oder Nichtnutzung solcherart dargebotener Informationen entstehen, haftet allein der Anbieter der Seite, auf welche verwiesen wurde, nicht derjenige, der über Links auf die jeweilige Veröffentlichung lediglich verweist.