SYS TEC ELECTRONIC

CAN-Ethernet-Gateway

L-Nummer hier eingeben (wird nicht gedruckt): L-1032d_10

System Manual

Auflage Juli 2010

Systemhaus für verteilte Automatisierung

Status / Änderungen

Status: freigegeben

Datum/ Version	Abschnitt	Änderung	Editor
12.07.2010	alle	Handbuch auf neue Handbuchvorlage umgestellt	D. Glau
19.07.2010	4.5.14	Befehlsbeschreibung ipaccept	D.Glau

Im Buch verwendete Bezeichnungen für Erzeugnisse, die zugleich ein eingetragenes Warenzeichen darstellen, wurden nicht besonders gekennzeichnet. Das Fehlen der © Markierung ist demzufolge nicht gleichbedeutend mit der Tatsache, dass die Bezeichnung als freier Warenname gilt. Ebensowenig kann anhand der verwendeten Bezeichnung auf eventuell vorliegende Patente oder einen Gebrauchsmusterschutz geschlossen werden.

Die Informationen in diesem Handbuch wurden sorgfältig überprüft und können als zutreffend angenommen werden. Dennoch sei ausdrücklich darauf verwiesen, dass die Firma SYS TEC electronic GmbH weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgeschäden übernimmt, die auf den Gebrauch oder den Inhalt dieses Handbuches zurückzuführen sind. Die in diesem Handbuch enthaltenen Angaben können ohne vorherige Ankündigung geändert werden. Die Firma SYS TEC electronic GmbH geht damit keinerlei Verpflichtungen ein.

Ferner sei ausdrücklich darauf verwiesen, dass SYS TEC electronic GmbH weder eine Garantie noch die juristische Verantwortung oder irgendeine Haftung für Folgeschäden übernimmt, die auf falschen Gebrauch oder falschen Einsatz der Hard- bzw. Software zurückzuführen sind. Ebenso können ohne vorherige Ankündigung Layout oder Design der Hardware geändert werden. SYS TEC electronic GmbH geht damit keinerlei Verpflichtungen ein.

	EUROPA	NORDAMERIKA
Adresse:	SYS TEC electronic GmbH August-Bebel-Str. 29 D-07973 Greiz GERMANY	PHYTEC America LLC 203 Parfitt Way SW, Suite G100 Bainbridge Island, WA 98110 USA
Angebots-Hotline:	+49 (0) 36 61 / 62 79-0 info@systec-electronic.com	+1 (800) 278-9913 order@phytec.com
Technische Hotline:	+49 (0) 36 61 / 62 79-0 support@systec-electronic.com	+1 (800) 278-9913 support@phytec.com
Fax:	+49 (0) 36 61 / 6 79 99	+1 (206) 780-9135
Webseite:	http://www.systec-electronic.com	http://www.phytec.com

1. Auflage Juli 2010

1 Einleitung

1.1 Grundlagen

Internetkommunikation über TCP/IP verbreitet sich auch im industriellen Bereich immer weiter. SYS TEC electronic GmbH stellt mit dem CAN-Ethernet Gateway eine Lösung vor, die es ermöglicht, CAN-Netzwerke über Internet/Ethernet zu koppeln und über Fernzugriffe zu überwachen oder zu beeinflussen. Das CAN-Ethernet Gateway übernimmt die Kommunikation und stellt dem Nutzer eine transparent arbeitende CAN-basierte Applikationsschnittstelle zur Verfügung.
Es erfolgt eine transparente, protokollunabhängige Übertragung der CAN-Nachrichten. Dadurch eröffnet sich ein großes Anwendungsgebiet. So kann das CAN-Ethernet Gateway mit verschiedenen CAN-Protokollen (z.B. CANopen, SDS, J1939, DeviceNet oder firmenspezifische Protokolle usw.) eingesetzt werden.

Das CAN-Ethernet Gateway kann in CAN-Netzwerken mit einer Übertragungsrate von bis zu 1MBit/s entsprechend CAN-Spezifikation 2.0A (11-Bit CAN-Identifier) und 2.0B (29-Bit CAN-Identifier) eingesetzt werden. Für jede CAN-Nachricht kann ein Timestamp durch das CAN-Ethernet Gateway erzeugt und zusammen mit den Daten übertragen werden.

Das CAN-Ethernet Gateway lässt sich über eine asynchrone serielle Schnittstelle (UART nach RS232 incl. Hardwareflusskontrolle) oder eine Telnet-Verbindung frei konfigurieren. Der Anwender kann die Funktionen des CAN-Ethernet Gateway so an das spezielle Einsatzgebiet anpassen.

Für die Kommunikation zwischen den CAN-Ethernet Gateways kommt ein UDP/IP-basiertes Netzwerkprotokoll (BTP = Block Transfer Protocol) zum Einsatz. Damit werden die CAN-Nachrichten mit minimaler Zeitverzögerung im Ethernet weitergeleitet. Die Zeiten

des TCP/IP-Protokolls für den Auf- und Abbau von Netzwerkverbindungen entfallen.
Es besteht auch die Möglichkeit, die CAN-Nachrichten mittels TCP/IP-Netzwerkprotokoll zu übertragen.

Das Design der Gateway-Firmware ist auf hohen Datendurchsatz ausgerichtet. Die optimierte Pufferverwaltung arbeitet mit minimalem Aufwand für das Kopieren und die Zwischenspeicherung von Daten.
Übertragungsspitzen in den CAN-Netzen werden abgefangen. Bei sehr hohem Datenaufkommen werden mehrere CAN-Nachrichten zu einem UDP bzw. TCP Paket zusammengefasst und im Block übertragen.
Das CAN-Ethernet Gateway erkennt aufgetretene Fehler und sendet CAN-Nachrichten (Fehlernachrichten), die den Fehlergrund enthalten. Der zu verwendende CAN-Identifier der Fehlernachricht kann konfiguriert werden *(siehe Abschnitt 6.2)*.

1.2 Einsatzgebiete

1.2.1 Verbindung von zwei CAN-Netzen mittels Ethernet

Eine typische Anwendung ist die Verbindung von zwei CAN-Netzwerken mittels Ethernet über große Entfernungen. In jedem CAN-Netzwerk arbeitet ein CAN-Ethernet Gateway. CAN-Nachrichten werden transparent zwischen den CAN-Ethernet Gateways übertragen.
Die Firmware des CAN-Ethernet Gateway erlaubt die Filterung weiterzuleitender CAN-Nachrichten, so dass nur die relevanten Daten über das Ethernet übertragen werden.

Die prinzipiellen Möglichkeiten des Netzaufbaus mit CAN-Ethernet Gateways sind in den folgenden Abbildungen *Abbildung 1* und *Abbildung 2* dargestellt:

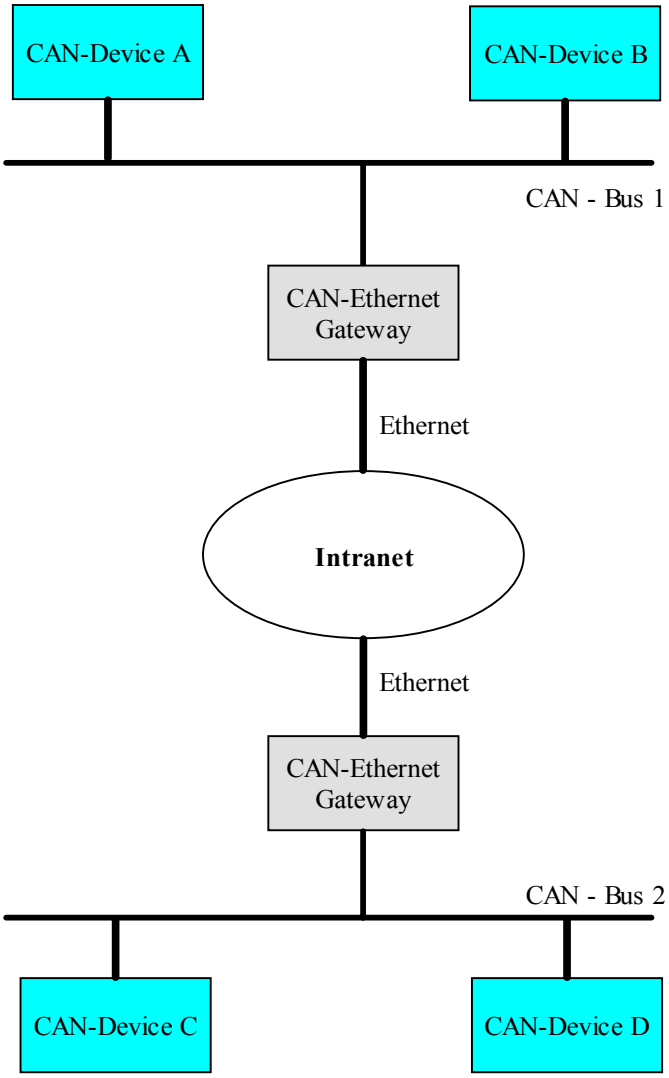

Abbildung 1: Transparente Verbindung zweier CAN

1.2.2 Ferndiagnose und Konfiguration von CAN-Netzen

Ein weiterer möglicher Anwendungsfall ist die Verbindung eines CAN-Netzwerks mit einem PC. Der Anwender benötigt ausschließlich die Netzwerkverbindung über Ethernet, um sich mit dem entfernten CAN-Netzwerk zu verbinden. Eine CAN-Hardware am PC ist nicht erforderlich.
Ein virtuelles CAN-Ethernet Gateway ist in Form einer PC-Software (DLL) unter MS-Windows verfügbar. Das Interface des virtuellen CAN-Ethernet Gateway entspricht einem CAN-Treiber.

Dadurch wird es möglich, CAN-Standardprogramme zu nutzen, die einen CAN-Treiber verwenden (z.B. CANopen Konfigurationstools wie ProCANopen™ oder CAN-Analysetools wie PCAN-Explorer™ oder PCANview™).

Das virtuelle CAN-Ethernet Gateway für den PC verlängert das CAN-Netz über das Ethernet/Intranet/Internet bis ins Büro und bietet damit neue Möglichkeiten der Konfiguration und Diagnose von CAN-Netzen in der Feldebene. Der PC in der Leitebene benötigt eine Ethernet-Verbindung zur Feldebene, bietet aber den Komfort gewohnter CAN und CANopen Tools.

Funktion und Parameter des Gateway selbst können über das Telnet-Protokoll fernbedient geändert werden.

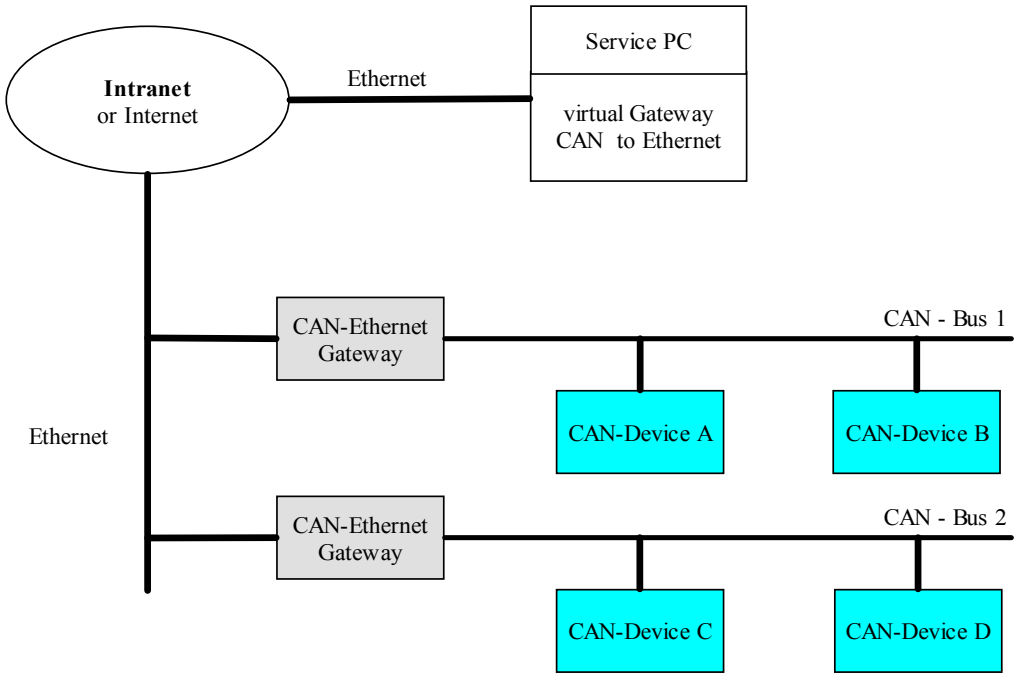

Abbildung 2: Ferndiagnose von CAN-Netzen mittels PC

1.3 Lieferumfang

Zum Lieferumfang des CAN-Ethernet Gateways gehören:

- GW-003
  CAN-Ethernet Gateway (1x CAN) im Gehäuse für Tragschienenmontage, incl. einem 2-pol. und einem 5-pol. abziehbaren Schraubklemmverbinder

- GW-003-20
  CAN-Ethernet Gateway (2x CAN) im Gehäuse für Tragschienenmontage, incl. einem 2-pol. und einem 5-pol. abziehbaren Schraubklemmverbinder

- L-1032
  Benutzerhandbuch *(dieses Handbuch, auf CD)*

- CD-ROM
  Installationsprogramm für Demo-Applikationen für TCP und UDP (SO-1027), Dokumentation, Beispiel-Konfigurationsfiles (SO-1027)

- WK041
  Nullmodemkabel zur Konfiguration des CAN-Ethernet Gateway via RS232

2 Technische Daten

Das CAN-Ethernet Gateway besitzt folgende technische Daten und Funktionalitäten:

- Überwachen und Steuern entfernter CAN-Netzwerke über das Internet
- Kopplung zweier CAN-Netzwerke
- Gateway konfigurierbar über Telnet (Fernwartung) oder RS232
- basiert auf internem Dateisystem für Konfigurationsdaten
- Fähigkeit, Scripte auszuführen (z.B. bei Start des Gateways)
- flexible Konfiguration durch Einsatz mehrerer Interfaces *(Abschnitt 4.2)*
- umfangreiche Filtermechanismen für CAN-Nachrichten mit der Möglichkeit zur Priorisierung
- Generierung eines Zeitstempels (Timestamp) für CAN-Nachrichten
- Anbindung zu Windows-Anwendungsprogrammen für CAN und CANopen
- 7 Leuchtdioden (LED) zur Visualisierung des Zustands des Gateways
- Generierung von CAN-Fehlernachrichten
- hoher Datendurchsatz
- 10Base-T Schnittstelle (10Mbit/s) mit RJ45-Buchse, galvanisch getrennt
- CAN-Schnittstelle nach CiA[1] DS102 bis 1MBit/s, Highspeed CAN nach ISO11898-1/2, galvanisch getrennt
- CAN-Bus-Anschluss: D-SUB-9 Stecker und 5-pol. abziehbarer Schraubklemmverbinder nach CiA DS102 bzw. DeviceNet-Standard
- Unterstützung der CAN-Spezifikation 2.0A (11-Bit CAN-Identifier) und 2.0B (29-Bit CAN-Identifier)
- RS232-Schnittstelle über D-SUB-9, Hardwareflusskontrolle
- Spannungsversorgung: 24VDC +20% -60%, verpolungssicher
- Stromaufnahme: ca. 90mA

[1] CiA, CAN in Automation, international users and manufacturers group

- Steckverbinder: 2-pol. abziehbarer Schraubklemmverbinder
- Maße ohne Steckverbinder: 70 x 100 x 61 (L x B x H) mm³ für DIN/EN-Tragschienenmontage
- Schutzgrad: IP20
- Einsatztemperaturbereich 0°C bis +70°C

Abbildung 3: Geräteansicht

3 Inbetriebnahme

3.1 Spannungsversorgung

Zum Betrieb des Gerätes ist eine Gleichspannung von 24V –60% bis +20% erforderlich. Die Stromaufnahme des Gerätes beträgt ca. 90mA. Der Anschluss erfolgt über einen 2-poligen, abziehbaren Schraubklemmverbinder. Der Anschluss von „+“ und „-„ ist am Gerät gekennzeichnet. Der korrekte Anschluss der Versorgungsspannung wird über die Spannungsanzeige „power“ signalisiert.

3.2 Netzwerkanschluss

3.2.1 CAN-Bus-Anschluss

Für das CAN-Netzwerk steht ein D-SUB-9 Stecker zur Verfügung. Alternativ ist der Anschluss mittels 5-poligem abziehbarem Schraubklemmverbinder möglich (passender Stecker PHOENIX CONTACT 1757048). Dieser Anschluss ist zum D-SUB-9 Stecker parallel geschaltet. Die Belegung entspricht dem DeviceNet bzw. CANopen-Standard.
Die Spannungsversorgung für den CAN-Bus (Pin 9 an D-SUB-9 bzw. Pin 5 am 5-pol. Steckverbinder) ist im Gateway nicht angeschlossen. Der Schirm CAN-Shield ist nur innerhalb der CAN-Anschlüsse verbunden. Der CAN-Bus ist galvanisch getrennt (optisch entkoppelt) vom Gateway.

D-SUB-9Male	5-pol.	Name	Beschreibung
1		n.c.	not connected
2	2	CL (CAN_L)	CAN_L bus line
3	1	V-(CAN_GND)	CAN Ground
4		n.c.	not connected
5	3	SH (CAN_SHLD)	CAN Shield
6		GND	CAN Ground (optional)
7	4	CH (CAN_H)	CAN_H bus line
8		n.c.	
9	5	V+ (CAN_V+)	not connected

Tabelle 1: Belegung CAN-Steckverbinder

3.2.2 Ethernet-Anschluss

Das Ethernet (10Base-T) wird mittels handelsüblichem CAT 3 oder CAT 5 Netzwerkkabel mit RJ45-Stecker angeschlossen. Für die direkte Verbindung (ohne Hub oder Switch) von einem CAN-Ethernet Gateway und einem PC ist ein Crosslink-Kabel erforderlich.
Der Ethernet-Anschluss ist galvanisch vom CAN-Ethernet Gateway getrennt.

Pin	Name	Beschreibung
1	TX+	Transceive Data +
2	TX-	Transceive Data -
3	RX+	Receive Data +
4	n.c.	not connected
5	n.c.	not connected
6	RX-	Receive Data +
7	n.c.	not connected
8	n.c.	not connected

Tabelle 2: Belegung Ethernet-Steckverbinder

3.2.3 RS232-Schnittstelle

Das CAN-Ethernet Gateway besitzt eine RS232-Schnittstelle mit Hardwareflusskontrolle. Sie wird über einen D-SUB-9 Stecker angeschlossen. Diese Schnittstelle erlaubt die Konfiguration des CAN-Ethernet Gateways. Insbesondere ist dieser Anschluss zur Erstkonfiguration vorgesehen *(siehe Abschnitt 3.5)*. Die RS232-Schnittstelle ist nicht galvanisch entkoppelt.

D-SUB-9Male	Name	Beschreibung
1	CD	Carrier Detect
2	RXD	Receive Data
3	TXD	Transmit Data
4	DTR	Data Terminal Ready
5	GND	System Ground
6	DSR	Data Set Ready
7	RTS	Request to Send
8	CTS	Clear to Send
9	RIN	Ring Indicator

Tabelle 3: Belegung RS232-Schnittstelle

Der Anschluss an den PC erfolgt mittels Nullmodemkabel .

3.3 Statusanzeige

Zur Anzeige des Betriebszustandes dienen insgesamt 7 LEDs *(siehe Tabelle 4)*. Die Anzeigen sind entsprechende ihrer Bedeutung zu den Netzwerken angeordnet (*Abbildung 3)*. Je eine rote und grüne LED zeigen den Zustand des CAN- bzw. Ethernet-Netzes an. Die detaillierte Beschreibung der „Error“-Anzeigen erfolgt im *Abschnitt 6.1*.

LED-Bezeichnung	Bedeutung
power	Spannungsversorgung OK
connect	Verbindung zu anderem Gateway über BTP oder TCP wurde aufgebaut
error (Ethernet)	Fehler bei der Datenübertragung auf Ethernet *(siehe Abschnitt 6.1)*
link	Verbindung zum Ethernet vorhanden, Verkabelung OK
active	Datenübertragung über Ethernet
traffic	Signalisierung von Datenverkehr auf dem CAN-Bus
error 0 (CAN0)	Fehler bei der Datenübertragung auf CAN 0 *(siehe Abschnitt 6.1)*
error 1 (CAN1)	Fehler bei der Datenübertragung auf CAN 1 *(siehe Abschnitt 6.1)*

Tabelle 4: Bedeutung der Anzeigeelemente

3.4 Schalter

Das CAN-Ethernet Gateway besitzt vier Schalter (DIP-Switch) *(siehe Abbildung 3)* mit folgender Bedeutung.

Schalter-Nr.	Bezeichnung	Bedeutung
1	TERM	Schaltet einen CAN-Abschlusswiderstand von 120Ohm ON → Abschlusswiderstand aktiviert OFF→ Abschlusswiderstand abgeschaltet
2	DEFT	Definiert die Anfangsinitialisierung des CAN-Ethernet Gateways *(siehe Abschnitt 3.5)* ON werksseitig eingestellte Default-Konfiguration wird geladen OFF→ Nutzerkonfiguration wird geladen
3	BOOT	Bei gesetztem Boot-Schalter und betätigten Reset-Schalter(ON/OFF) wird das Modul in den Bootstrab-Mode gesetzt. (Firmwareupdate) ON → Boot gesetzt OFF→ werkseitig eingestellte Defaulteinstellung
4	RES	Kurzeitiges Setzen und Rücksetzen des Reset-Schalters setzt das Modul hard- und softwareseitig zurück. ON → Reset gesetzt OFF→ werkseitig eingestellte Defaulteinstellung

Tabelle 5: Bedeutung der DIP-Schalter

3.5 Erstinbetriebnahme

3.5.1 Standardkonfiguration

Werksseitig besitzt das CAN-Ethernet Gateway folgende Standardkonfiguration *(Erstellung des Konfigurations-Script siehe Abschnitt 5.1)*:

Ethernet/Internet-Einstellungen

IP-Adresse des CAN-Ethernet Gateway:	192.168.10.111
Subnet-Mask:	255.255.255.0
Standard-Gateway:	192.168.10.1
ein UDP[1]-Server	
ein TCP-Server	

CAN-Einstellungen

CAN-Bitrate:	1Mbit/s
CAN-Identifier für Fehlernachrichten:	0xFE

RS232-Schnittstelle

Baudrate:	9600 Baud
Datenbits:	8
Parität:	keine
Stoppbits:	1
Protokoll:	Hardware

3.5.2 Erstkonfiguration über RS232-Schnittstelle

Vor der Übertragung von CAN-Nachrichten sind das CAN-Ethernet Gateway entsprechenden der Anforderungen zu konfigurieren.
Dazu sind folgende Schritte erforderlich:

[1] BTP: Block Transfer Protokoll zur Übertragung von CAN-Telegrammen mittels UDP/IP

- Verbinden Sie das mitgelieferte Nullmodemkabel mit der RS232-Schnittstelle des CAN-Ethernet Gateway und einer freien seriellen Schnittstelle des PC (z.B. COM1).
- Starten Sie auf dem PC ein Terminal-Programm, (im Weiteren wird das Programm „Tera Term“ verwendet. Dieses Programm ist erhältlich unter http://ttssh2.sourceforge.jp/.
- Stellen Sie die Baudrate und das Protokoll auf der seriellen Schnittstelle (z.B. COM1) unter **Einstellungen\Serieller Port** im Tera Term Menü ein *(siehe Abbildung 4 und Abbildung 5)*.

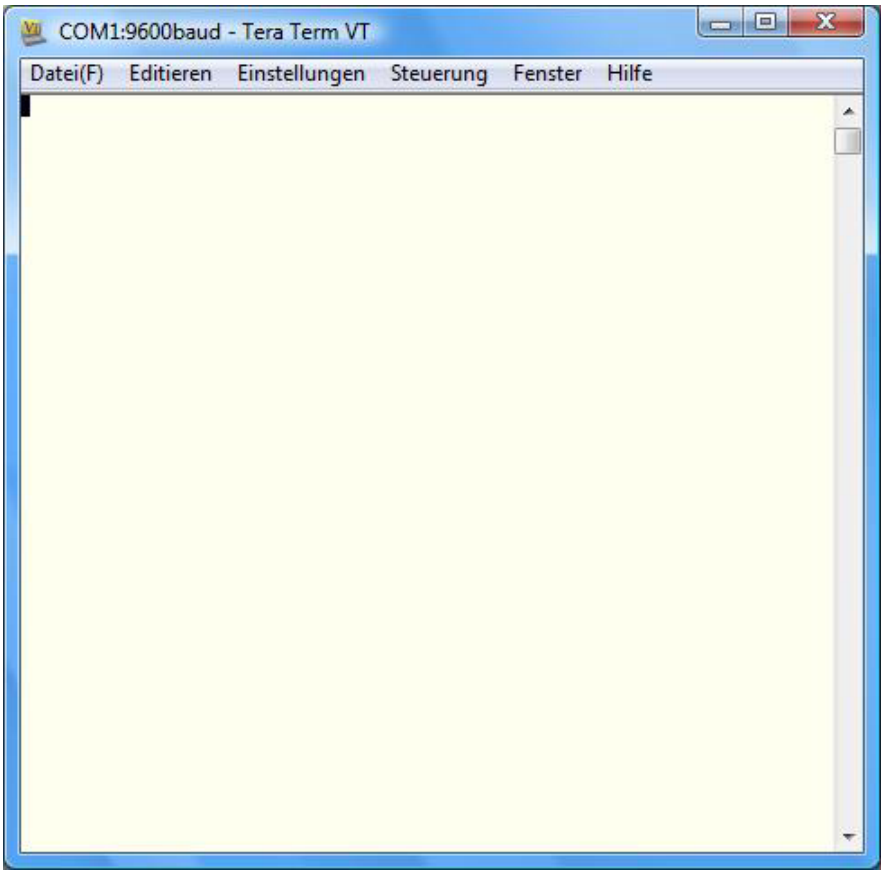

Abbildung 4: Konfiguration des Tera Term-Terminal (1)

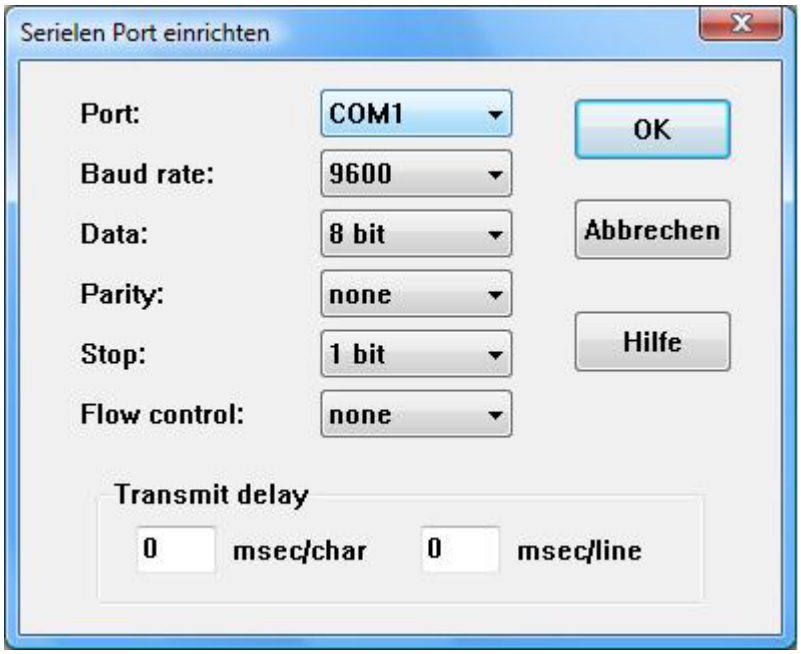

Abbildung 5: Konfiguration des Tera Term-Terminal (2)

- Stellen Sie sicher, dass Sie die Spannungsversorgung richtig angeschlossen haben und schalten Sie die Versorgungsspannung ein.

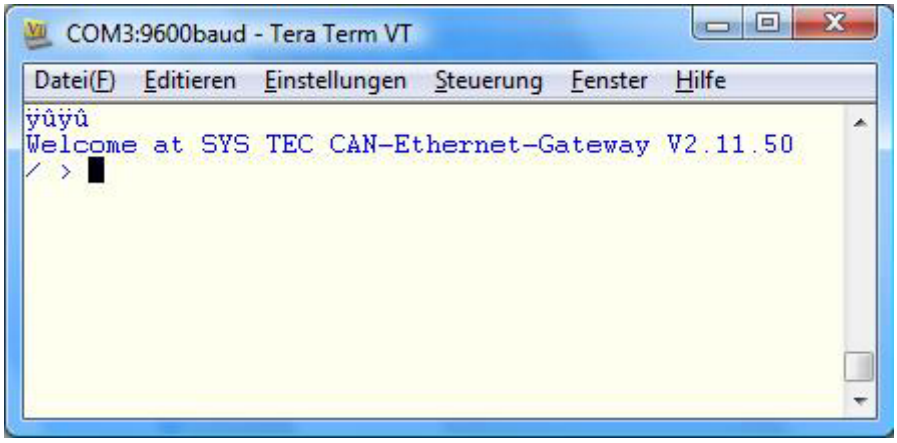

Abbildung 6: Startmeldung des CAN-Ethernet Gateway

Das Gerät meldet sich mit *(siehe Abbildung 6)*:

```
Welcome at SYS TEC CAN-Ethernet-Gateway V2.11.50
    / >
```

und fordert zur Eingabe von Kommandos auf. Die Ausgabe V2.11.50 stellt die aktuelle Versionsnummer dar und kann abweichen.

Da das CAN-Ethernet Gateway auf einem internen Dateisystem basiert, gibt es Kommandos zur Navigation. Dies sind *(siehe Abschnitt 4.5)*:

`ls`	um den Inhalt des aktuellen Verzeichnisses anzuzeigen,
`cd`	um das aktuelle Verzeichnis zu wechseln
`write`	um eine neue Datei anzulegen
`rm`	um ein Verzeichnis/Datei zu löschen
`sync`	um eine Datei in den nichtflüchtigen Speicher (EEPROM) zu speichern

Für die Erstkonfiguration werden vordefinierte Files mitgeliefert. Diese befinden nach der Installation der Software SO-1027 in das vordefinierte Standardverzeichnis unter:
`C:\Programme\SYSTEC-electronic\CAN-Ethernet-Gateway_Utility_Disk\Rc-Files\UDP`(für die UDP-Übertragung) bzw.
`C:\Programme\SYSTEC-electronic\CAN-Ethernet-Gateway_Utility_Disk\Rc-Files\TCP`(für die TCP-Übertragung).

Jedes Verzeichnis enthält ein File für eine Server-Konfiguration (`UDP_Server_1CAN.txt` oder `TCP_Server_1CAN.txt`) und ein File für eine Client-Konfiguration (`UDP_Client_1CAN.txt` oder `TCP_Client_1CAN.txt`).

- Bevor Sie eines dieser Files auf das CAN-Ethernet Gateway übertragen, müssen Sie diese Files ändern.
  Öffnen Sie dazu das File `TCP_Client_1CAN.txt` mit einem Editor.

- Ändern Sie die lokale IP-Adresse, die Subnet-Mask und die Standard-Gateway-Adresse auf die Werte, die Ihren Anforderungen entsprechen *(siehe Abschnitt 4.5.12)*.
  z.B.
  `ipcfg 192.168.10.179 255.255.255.0 192.168.10.1`
- Stellen Sie die CAN-Bitrate und den CAN-Identifier für Fehlernachrichten ein, *(siehe Abschnitt 4.2.4)*
  z.B. auf 1000kBit/s und CAN-Identifier 0xFD
  `/if/can0 bus:0 baud:0 canid:FD on`
- Speichern Sie das Konfigurationsfile unter einem anderen Namen
  z.B.
  `TCP_Client_1CAN_179.txt`

Das erstellte Konfigurationsfile wird wie folgt auf das CAN-Ethernet Gateway geschrieben:

1. Wechseln Sie in das Verzeichnis `/save`
   ```
   / >cd /save↵
   /save >
   ```

2. Löschen Sie das bestehende Konfigurationsfile `rc`
   ```
   /save >rm rc↵
   /save >sync↵
   ```

3. Schreiben Sie eine neues Konfigurationsfile `rc`
   ```
   /save >write rc↵
   ```
   Nach der Eingabe des Kommandos können Sie via Tera Term-Terminal eine Datei zum CAN-Ethernet Gateway senden. Die nachfolgenden Bilder zeigen dies.

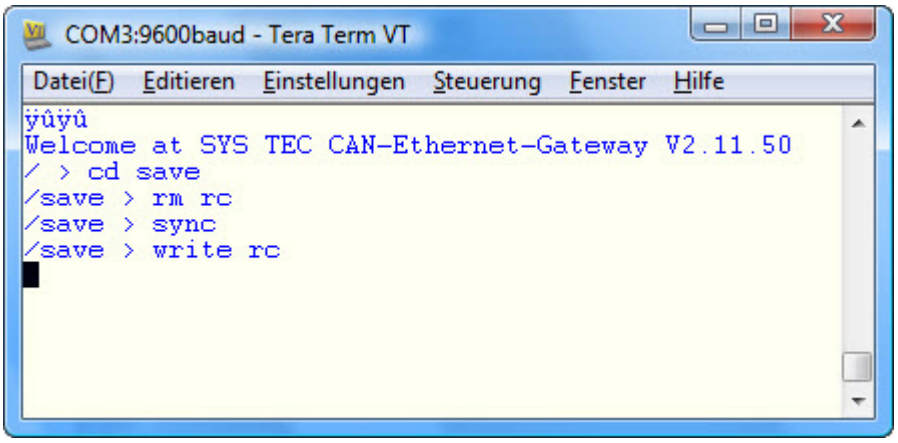

Abbildung 7: Senden des Konfigurationsfile mittels Tera Term

Wählen Sie aus dem Menü **Datei(F)\Sende Datei** aus worauf sich der folgende Dialog öffnet, in dem Sie das zu sendende Konfigurationsfile auswählen können.

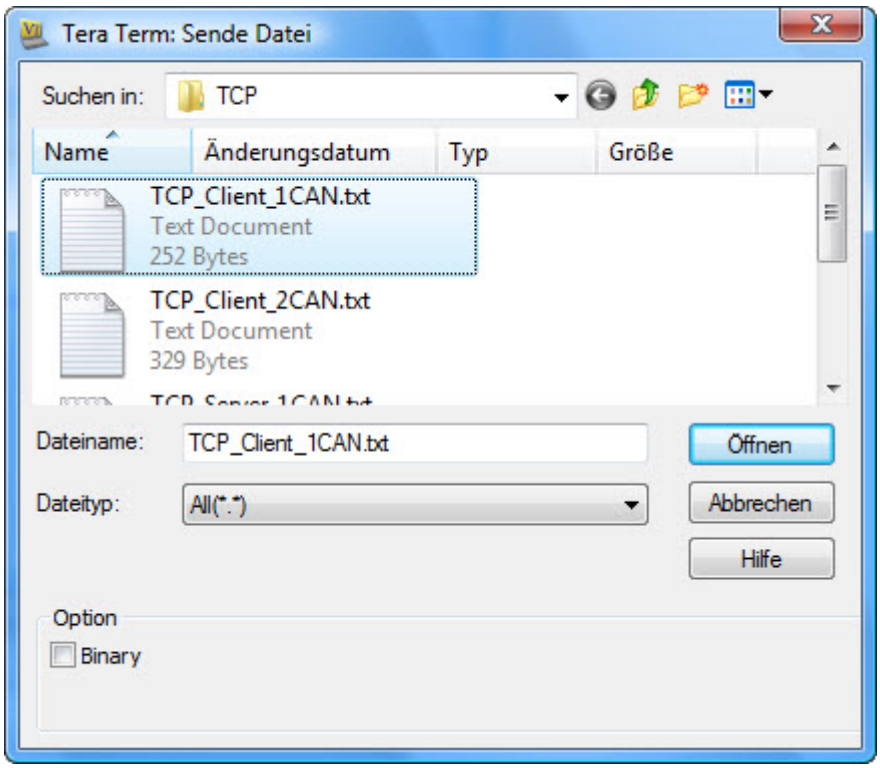

Abbildung 8: Konfigurationsfile auswählen

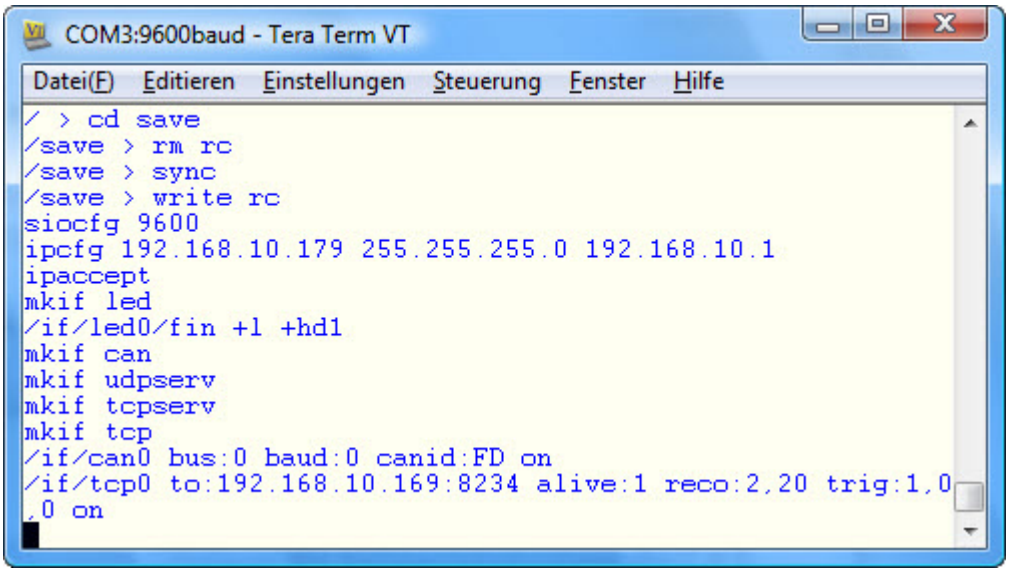

Abbildung 9: Übertragung des Konfigurationsfiles beendet

4. Zum Beenden der Übertragung drücken Sie die Tastenkombination `Strg+D`. Das Gateway zeigt die Ausführung durch den Prompt `/save >` an.

5. Speichern Sie das File `rc` mit dem Befehl `sync` im EEPROM.

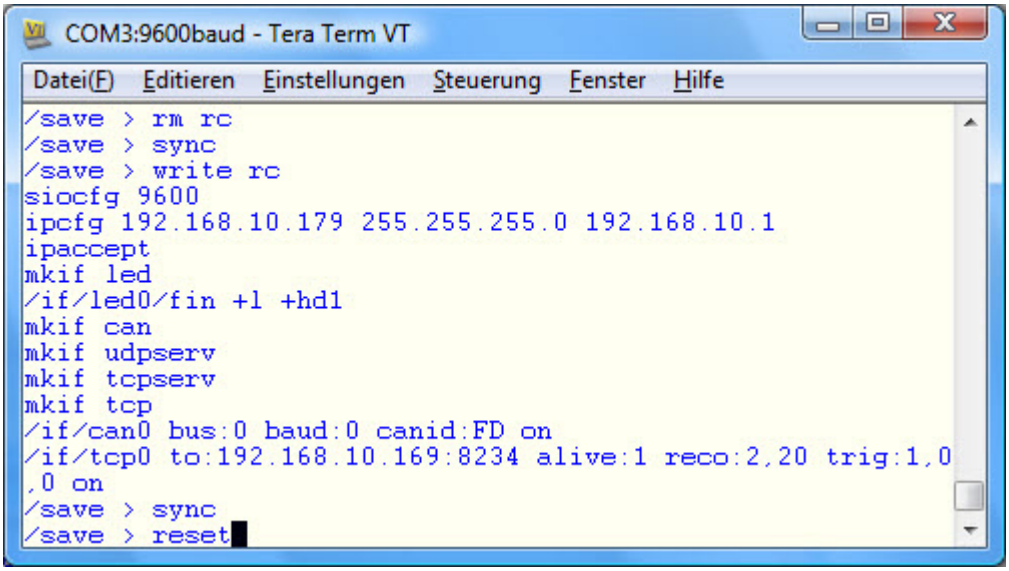

Abbildung 10: Abschluss der Konfiguration

6. Setzten Sie den Schalter 2 „DEFT“ auf OFF und starten Sie das CAN-Ethernet Gateway neu. Beim Neustart des CAN-Ethernet Gateway durch Power-On oder den Befehl `reset` wird das so gespeicherte Konfigurationsfile ausgeführt und das CAN-Ethernet Gateway konfiguriert.
   `/save> reset↵`

7. Überprüfen Sie die neue Konfiguration durch Eingabe des Befehls `ipcfg`. *(siehe Abbildung 11)*. Die im Konfigurationsfile gespeicherten Parameter IP-Adresse, Subnet-Mask und Standard-Gateway entsprechen Ihren Einstellungen. Jetzt ist das Gateway konfiguriert und kann via Ethernet (z.B. Telnet) bedient werden.

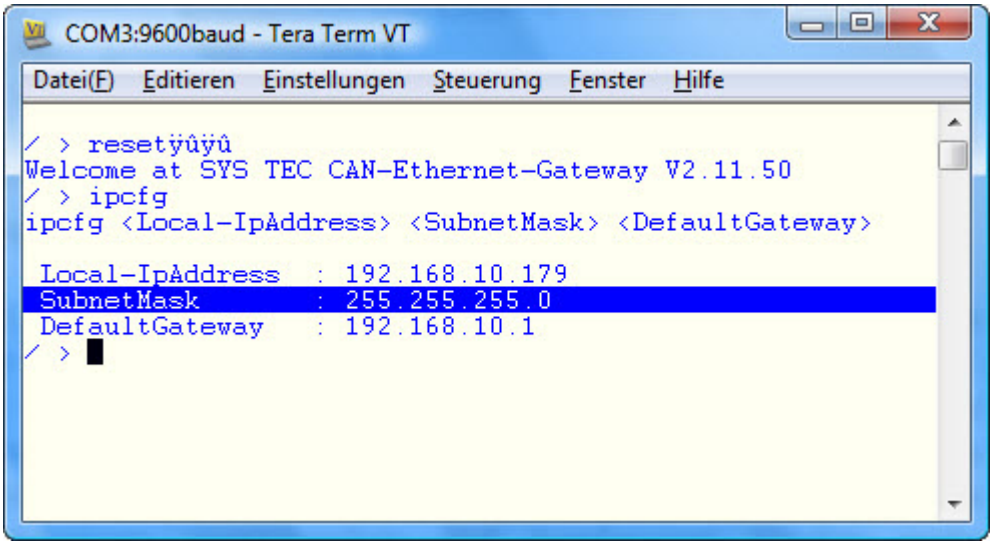

Abbildung 11: Überprüfung der eingestellten Konfiguration

Hiermit ist die Erstinbetriebnahme abgeschlossen.

Das CAN-Ethernet Gateway arbeitet mit verschiedenen Interfaces *(siehe Abschnitt 4.2)*, die den Datenaustausch über Ethernet bzw. CAN realisieren. Interfaces werden über das Kommando `mkif` angelegt und über `rm` gelöscht. Alle angelegten Interfaces erscheinen im Verzeichnis `/if`. Ein Interface kann durch direkten Aufruf des Verzeichnisses angesprochen und konfiguriert werden *(siehe Abschnitt 4.2.4)*.

3.5.3 Konfiguration und Bedienung über Telnet

Die Konfiguration im laufenden Betrieb des CAN-Ethernet Gateways kann auch über Telnet (TCP-Port 23) erfolgen. Der Funktionsumfang ist der Gleiche, wie der der RS232-Schnittstelle. Durch die Konfiguration über Telnet ist es möglich, auch entfernte CAN-Ethernet Gateways zu konfigurieren. Voraussetzung ist die einmalige Konfiguration der IP-Adresse mit dem Kommando `ipcfg` *(Abschnitt 4.5.12)*. Ohne Erstkonfiguration der IP-Adresse ist das CAN-Ethernet Gateway über seine Standard-Konfiguration ansprechbar *(siehe Abschnitt 3.5.1)*.
Im Lieferumfang von Windows ist bereits ein Telnet-Client enthalten. Dieser kann über `telnet <Adresse>` aufgerufen werden. Unter Linux können die Programme `telnet` oder `netcat` verwendet werden.

Als erstes öffnet man die Telnet-Shell mit der entsprechenden IP-Adresse des Gateways. Am Eingabe-Prompt wird in das Verzeichnis `<save>` gewechselt und das alte Resource-File gelöscht.

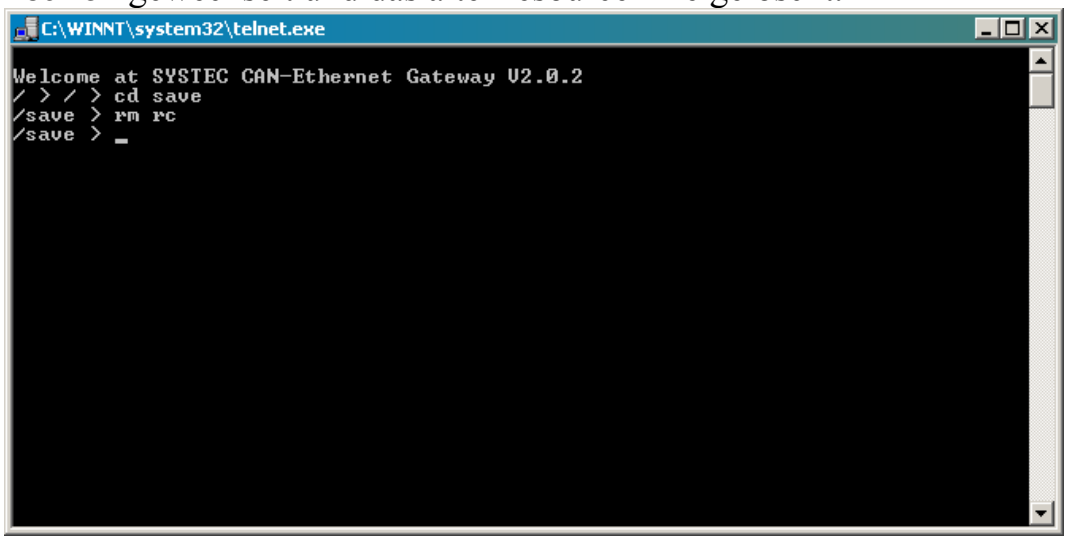

Abbildung 12: Löschen des Resource-Files via Telnet

Im Anschluss kann das neue Resource-File über Telnet auf das Gateway übertragen werden. Dazu ist der Inhalt des Resource-Files in die Zwischenablage zu kopieren. Durch Klicken der rechten Maustaste auf den Fensterrand der Shell *(siehe Abbildung 13)* öffnet sich ein Kontextmenü.

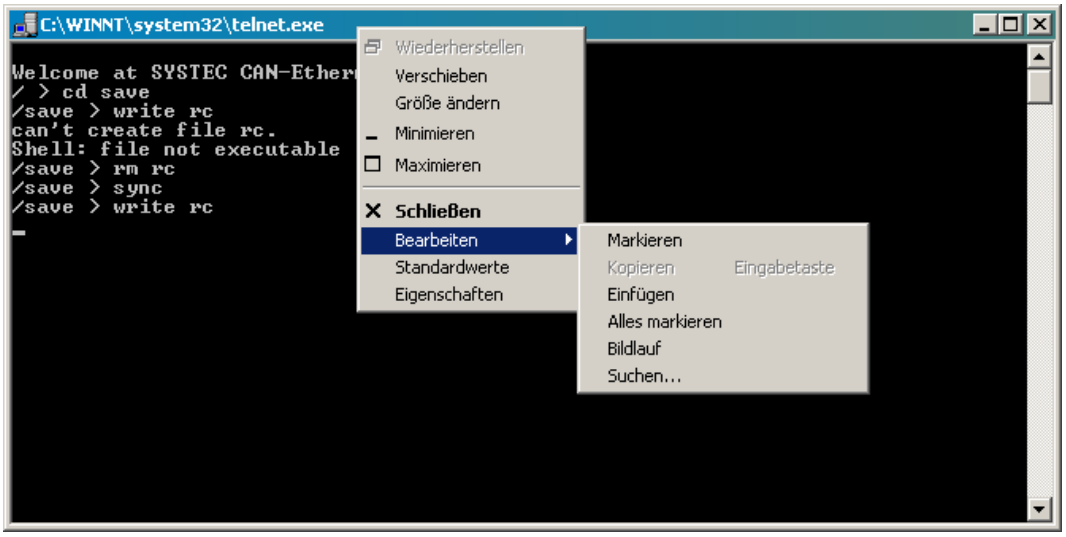

Abbildung 13: Download des Resource-Files via Telnet

Über den Menüeintrag ***Bearbeiten/Einfügen*** kann der Inhalt des Resource-Files aus der Zwischenablage eingefügt werden und wird mittels Telnet an das Gateway gesendet, wobei die übertragene Konfiguration innerhalb der Shell angezeigt wird.
Der Download ist mit dem Befehl `STRG + D` abzuschließen. Das Speichern der Konfiguration erfolgt mittels `sync`-Befehl.
Damit die neue Konfiguration wirksam wird, ist das Gateway mit dem Befehl `reset` zurückzusetzen. Dabei geht die Telnet-Verbindung zwischen PC und Gateway verloren. Dies gilt auch, wenn der Reset hardwareseitig über den ***RESET***-Schalter durchgeführt wird.

4 Gerätefunktion

4.1 Überblick

Das CAN-Ethernet Gateway enthält mehrere Interfaces für den Betrieb und die Steuerung. Der prinzipielle Aufbau sieht wie folgt aus:

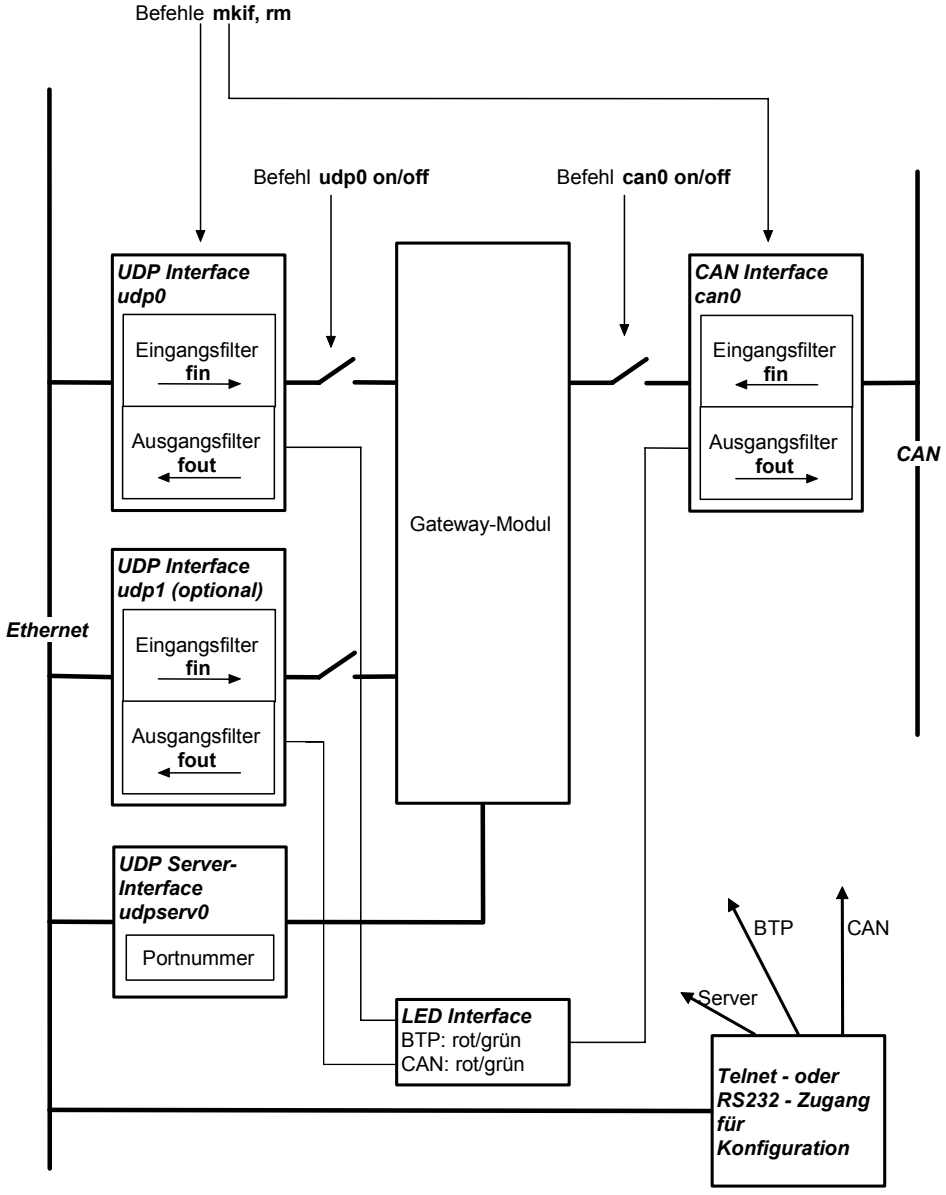

Abbildung 14: Prinzip CAN-Ethernet Gateway

4.2 Interfaces

4.2.1 Grundkonzept

Der Austausch von CAN-Nachrichten erfolgt über Interfaces. Ein Interface stellt die Verbindung zwischen einer CAN-Nachrichten-Ein-/Ausgabe und dem zentralen Verteiler (Datenpool im Gatewaymodul) im CAN-Ethernet Gateway her. Es können mehrere Interfaces aktiviert werden, sofern genügend Speicher auf dem Gateway frei ist.

Die zwei wichtigsten Interfaces sind `can` für das CAN-Interface und `udp` bzw. `tcp` für ein Ethernet-Interface nach dem Block-Transfer-Protokoll. Während das CAN-Interface CAN-Nachrichten über die Hardware auf ein CAN-Netz sendet und empfängt, ist das UDP-Interface für das Tunneln der Nachrichten über UDP/IP/Ethernet verantwortlich.

verfügbare Interfaces	Typ	maximale Anzahl	Bedarf
CAN	can	1	erforderlich
UDP-Client	udp	3	je nach Applikation erforderlich
UDP-Server	udpserv	1	erforderlich
TCP-Client	tcp	3	je nach Applikation erforderlich
TCP-Server	tcpserv	1	erforderlich
LED	led	1	erforderlich

Tabelle 6: Übersicht über Interfaces

Das UDP-Interface transportiert die CAN-Nachrichten auf Basis des UDP-Protokolls während das TCP-Interface stattdessen TCP als Transportprotokoll verwendet. Beide Interface-Typen sind funktionell identisch aber unterscheiden sich hinsichtlich der Übertragungsgeschwindigkeit.

Interfaces werden über das Kommando `mkif` angelegt und über das Kommando `rm` gelöscht. Alle angelegten Interfaces erscheinen im Verzeichnis `/if`. Innerhalb des Interfaceverzeichnisses befinden sich

3 Dateien. Die Dateien `fin` und `fout` sind für die CAN-Nachrichtenfilterung verantwortlich *(siehe Abschnitt 4.3)*. Die Datei `conf` ist für zukünftige Erweiterungen vorgesehen und derzeit ungenutzt.

Ein Interface kann durch Starten des Interfaceverzeichnisses angesprochen werden.

```
/if/<if-name> {<option>}
```

z.B.

```
/if/can0↵
```

oder auch

```
cd /if/<if-name>
. {<option>}
```

z.B.

```
/if/can0 canid:123↵
```

erfolgen.

Nach dem Erstellen eines Interface können durch das Kommando Interfacename + -nummer die aktuellen Einstellungen abgefragt werden.

z.B.

```
/if/can0↵
Usage:
/if/can0 [bus:<num>] [baud:<num>] [userbaud:<hex>]
          [canid:<hex>] [on] [off]

Current settings:
bus:0 baud:0 (1MBit/s) userbaud:0x0 canid:0xF6 on
state:0x0
```

4.2.2 UDP/TCP-Server Interface

Das UDP-Server-Interface (`udpserv`) wartet auf UDP-Verbindungsanfragen von einem anderen Gateway und erstellt dann eine neue UDP-Verbindung, die für den CAN-Nachrichtenaustausch benutzt wird. Der TCP-Server (`tcpserv`) wartet entsprechend auf TCP-Verbindungsanfragen und legt gegebenenfalls ein TCP-Interface zur Kommunikation an.

Die so erzeugten BTP/Typ-Interfaces werden bei Verbindungswende bzw. Abbruch durch den UDP/TCP-Server wieder entfernt. Um sowohl auf TCP- als auch auf UDP-basierende Anfragen zu reagieren, müssen ein TCP- und ein UDP-Server-Interface auf dem Gateway existieren.

Ein UDP-Server Interface wird durch `mkif udpserv` mit der nächsten freien Interfacenummer erzeugt. Der Verzeichnisname wird automatisch gewählt und ist `/if/udpserv0` für das erste Interface.

Ein TCP-Server Interface wird mit `mkif tcpserv` unter `/if/tcpserv0` angelegt.

Mögliche Optionen:

```
<option> ::= {port:<num>} |
             {trig:<cnt>[,<time>][,inhibit]}
```

`<num>` enthält den Port, auf dem das Interface auf Anfragen wartet.

Wertebereich:	1 <= `port` <= 65535 (UDP)
	1<= `port` <= 65535 und `port` != 23 (TCP) (reserviert für Telnet-Protokoll)
Default:	8234

```
<trig> ::= trig:<cnt>[,<time>][,inhibit]
```

Diese Parameter können für die Steuerung der Übertragung von CAN-Telegrammen in TCP/UPD-Paketen verwendet werden. Es wird damit eine Triggerschwelle eingestellt, die es ermöglicht, eine bestimmte Mindestanzahl von CAN-Telegrammen in einem UDP/TCP-Paket zu übertragen. Um sicherzustellen, dass die Pakete nach einer maximalen Zeit übertragen werden, auch wenn die Mindestanzahl der CAN-Telegramme nicht erreicht wurde, kann ein zusätzliches Timerevent die Sendung der UDP/TCP-Pakete auslösen. Diese Parameter gelten für Verbindungen, die über diesen UDP/TCP Server automatisch angelegt werden.

`cnt` Anzahl der CAN-Telegramme die mindestens im internen Puffer für die Übertragung mittels Ethernet stehen müssen, bevor ein Ethernet-Paket gesendet wird
Wertebereich: 1 - 127
Default: 1

`time` Zeit in Millisekunden [ms] die seit der letzten Übertragung maximal vergehen ***kann***, bis das nächste Ethernet-Paket übertragen wird.
Wertebereich: 0 - 4294967295 [ms]
Default: 0

`inhibit` Zeit in Millisekunden [ms] die seit der letzten Übertragung vergehen ***muss***, bis das nächste Ethernet-Paket übertragen wird.
Wertebereich: 0 - 100 [ms]
Default: 0

Es gilt folgende Bedingung für das Senden eines Ethernet-Telegramms:

Anzahl empfangene CAN-Telegramme >= `cnt`
ODER
Zeitdifferenz zwischen der letzten Sendezeit und der aktuellen Zeit > `time`
ODER
Zeitdifferenz zwischen der letzten Sendezeit und der aktuellen Zeit > `inhibit`

Ausgabe der aktuellen Konfiguration:

```
/if/udpserv0↵

Usage:
udpserv0 port:<num> [trig:<cnt>[,<time>][,<inhibit]]
Current port: 8234 trig:1,0,0
```

4.2.3 UDP/TCP-Client Interface

Das UDP-Client Interface stellt einen Tunnel über UDP/IP/Ethernet zur Verfügung, um CAN-Nachrichten zu versenden beziehungsweise zu empfangen. Beim TCP-Client Interface wird der Tunnel über TCP/IP/Ethernet betrieben. Das UDP/TCP-Client Interface ist instanziierbar, so dass auch mehrere, jedoch maximal 3 Verbindungen gleichzeitig möglich sind.

Ein UDP-Client Interface wird durch `mkif udp 0` mit der Interfacenummer 0 oder der nächsten freien Nummer erzeugt. Es dient dazu, aktiv eine Verbindung zu einem anderen Gateway aufzubauen. Der Verzeichnisname wird automatisch gewählt und ist `/if/udp0` für das erste Interface.

Ein TCP-Client Interface wird durch `mkif tcp 0` erzeugt.

Mögliche Optionen:

```
<option> ::={<addr>|<switch>|<alive>|<reco>}|<trig>
```

```
<addr> ::= to:<ipaddr>[:<port>]
```

Einstellung der Ziel-IP-Adresse und des Ports des UDP-Servers, mit dem kommuniziert werden soll (default: IP-Adresse 192.168.10.111, Port 8234) bei aktivem Verbindungsaufbau durch das Gateway

```
<switch> ::= {on | auto | off}
```

- `on` Startet den aktiven Verbindungsaufbau
- `auto` Startet den Verbindungsaufbau, sobald CAN-Nachrichten zu senden sind.
- `off` Beendet die Verbindung, nicht übertragene Datenrahmen werden verworfen (default)

`<alive> ::= alive:<num>`

`0` Schaltet die zyklische Überprüfung der Kommunikationsstrecke ab. Ein Fehlen des Kommunikationspartners wird nicht erkannt, das betreffende Interface muss dann mit Hand entfernt werden.

`1` Schaltet die zyklische Überprüfung der Kommunikationsstrecke ein (default bei UDP). Nicht verfügbar für TCP-Interface

`<reco> ::= reco:<num>,[<time>]`

`0` schaltet die automatische Wiederverbindungsfunktion (reconnection) aus

`1` schaltet die automatische Wiederverbindungsfunktion (reconnection) bei CAN-Nachrichtenempfang ein
Beim Reconnect-Type **1** wird eine Verbindung aufgebaut bzw. wiederhergetsellt, wenn CAN-Nachrichten empfangen und im Sendepuffer eingetragen sind. Wurde die Verbindung erfolgreich neu aufgebaut, dann bleibt die Verbindung im fehlerfreien Betrieb bestehen, auch wenn keine neuen CAN-Nachrichten empfangen werden.

`2` schaltet die automatische Wiederverbindungsfunktion (reconnection) ein (Default)
Beim Reconnect-Type **2** wird eine Verbindung aufgbaut bzw. wiederhergestellt, ohne dass der Verbindungsaufbau an bestimmte Bedingungen geknüpft ist. Der Aufbau der Verbindung erfolgt sofort nach PowerOn oder bei Verbindungsabbruch nach Ablauf des parametrierten Reconnect-Timeouts.

`time` Sekunden die gewartet werden, bis ein erneuter Verbindungsaufbau gestartet wird (default: 120s)

```
<trig> ::= trig:<cnt>[,<time>][,<inhibit]
```

Diese Parameter können für die Steuerung der Übertragung von CAN-Telegrammen in UDP/TCP-Paketen verwendet werden. Es wird damit eine Triggerschwelle eingestellt, die es ermöglicht, eine bestimmte Mindestanzahl von CAN-Telegrammen in einem UDP/TCP-Paket zu übertragen. Um sicherzustellen, dass die Pakete nach einer maximalen Zeit übertragen werden, auch wenn die Mindestanzahl der CAN-Telegramme nicht erreicht wurde, kann ein zusätzliches Timerevent die Sendung der UDP/TCP-Pakete auslösen.

`cnt` Anzahl der CAN-Telegramme die mindestens im internen Puffer für die Übertragung mittels Ethernet stehen müssen, bevor ein Ethernet-Paket gesendet wird
Wertebereich: 1 - 127
Default: 1

`time` Zeit in Millisekunden [ms] die seit der letzten Übertragung maximal vergehen darf, bis das nächste Ethernet-Paket übertragen wird.
Wertebereich: 0 - 4294967295 [ms]
Default: 0

`inhibit` Zeit in Millisekunden [ms] die seit der letzten Übertragung vergehen muss, bis das nächste Ethernet-Paket übertragen wird.
Wertebereich: 0 - 100 [ms]
Default: 0

Es gelten folgende Bedingung für das Senden eines UDP/TCP-Paketes:

Anzahl empfangene CAN-Telegramme >= `cnt`
ODER
Zeitdifferenz zwischen der letzten Sendezeit und der aktuellen Zeit > `time`
ODER
Zeitdifferenz zwischen der letzten Sendezeit und der aktuellen Zeit > `inhibit`

Ausgabe der aktuellen Konfiguration:
`/if/udp0↵`

```
Usage:udp0 [to:<ipaddr>[:<port>]] [on|auto|off]
      [alive:<num>] [reco:<num>[,<time>]]
      [trig:<cnt>[,<time>][,inhibit]]

0 frames sent, 0 frames recv'd, state: Connection
establishment in process

Current settings:
to:192.168.10.118:8234 alive:1 reco:1,120 trig:1,0,0
```

4.2.4 CAN-Interface

Ein CAN-Interface wird durch `mkif can 0` mit der Interfacenummer 0 erzeugt. Der Verzeichnisname wird automatisch gewählt und ist `/if/can0` für das erste Interface. Es darf maximal ein CAN-Interface aktiv sein.

Mögliche Optionen:

```
<option> ::=
    {<busid>|<bauidx>|<userbaud>|<canid>|<switch>}

<switch> ::= {on | off}
```

`on`	Initialisiert den CAN-Treiber
`off`	Schaltet den CAN-Treiber aus (default)

```
<baudidx> ::= baud:<num>
```

setzt Übertragungsrate entsprechend Baudratenindex laut CiA DSP305[1]

`0`	1000	kBit/s	(default)
`1`	800	kBit/s	
`2`	500	kBit/s	
`3`	250	kBit/s	
`4`	125	kBit/s	
`5`	100	kBit/s	
`6`	50	kBit/s	
`7`	20	kBit/s	
`8`	10	kBit/s	
`FF`	Verwendung der userspezifischen Baudrate		

[1] CiA DSP305: CAN in Automation Draft Standard Proposal CANopen Layer Setting Services and Protocol

`<userbaud> ::= userbaud:<hex>`

`hex` setzt den userspzifischen Wert `<hex>` für die CAN-Baudraten, welche keine Standard-Baudraten nach CiA DSP305 darstellen. Dieser hexadezimale Wert stellt den Initialiserunsgwert des Bittiming-Registers des CAN-Controllers dar und muss individuell berechnet werden. Bei Fragen zur Berechnung bzw. zur möglichen praktischen Umsetzung der gewünschten Baudrate wenden Sie sich bitte an den Support unter *support@systec-electronic.com* oder besuchen Sie unsere Homepage *http://www.systec-electronic.com*.

Default: 0x0000

Wertebereich: 0<=`hex` <= 0xFFFF

ACHTUNG!:
Die userspezifische CAN-Baudrate muss hexadezimal ohne Sonderzeichen übergeben werden. z.B. `2F41` entspricht einer resultierenden Wer von `0x2F41`(hex.)

`<canid> ::= canid:<hex>`

`hex` setzt den CAN-Identifier `<hex>`, der beim Versand von Fehlernachrichten verwendet wird

Default: 0xFE.

Wertebereich: 0<= `hex` <= 7FF

ACHTUNG!:
Der CAN-Identifier muss hexadezimal ohne Sonderzeichen übergeben werden. z.B. `123` entspricht einem resultierenden CAN-Identifier von 0x123hex bzw. 291dez. Es werden nur 11-Bit CAN-Identifier unterstützt.

Der Aufbau der Fehlernachricht ist im *Abschnitt 6.2* beschrieben.

`0` schaltet das Senden von Fehlernachrichten im CAN-Ethernet Gateway aus

`1`
bis
`7FF` setzt den CAN-Identifier auf den übergebenen Wert.

`<busid> ::= bus:<num>`

`num` bestimmt den CAN-Bus-Anschluss, der für das Interface benutzt werden soll. (default:0)
GW-003 unterstützt einen CAN-Bus: `num` = 0

GW-003-2 CAN-Ethernet Gateway mit zwei CAN-Bussen, Wertebereich: 0 <= `num` <=1

Ausgabe der aktuellen Konfiguration:

```
/if/can0↵
Usage:
can0 [bus:<num>] [baud:<num>] [userbaud:<hex>]
[canid:<hex>] [on] [off]

Current settings:
bus:0 baud:0 (1MBit/s) userbaud:0x0 canid:0xF6 on
state:0x0
```

4.2.5 LED-Interface für Anzeige

Das LED-Interface dient der Signalisierung von Betriebszuständen über die LED‘s für CAN und Ethernet. Es erfolgt die Auswahl der an den Status-LEDs angezeigten UDP bzw. TCP-Interfaces. Für CAN ist keine Auswahl möglich, aber das Interface muss konfiguriert sein.

Ein LED-Interface wird durch `mkif led` erzeugt. Der Verzeichnisname wird automatisch gewählt und ist `/if/led0` für das Interface.

Über die einstellbaren Filterregeln *(siehe Abschnitt 4.3)* kann ausgewählt werden, welche LEDs zu welchem Interface gehören. Beispiel:

Durch die Einstellung `/if/led0/fin +hd1 +ld23` werden die Zustände des Interface 1 den LEDs für CAN-Error und CAN-Traffic (gekennzeichnet durch `hd`) zugewiesen und die Zustände der Interfaces 2 und 3 über die BTP-Error und BTP-Connection-LEDs (gekennzeichnet durch `ld`) angezeigt. Zustände von anderen Interfaces werden nicht angezeigt. Die Interface-Id‘s können mit dem Kommando `ls` ermittelt werden *(siehe Abschnitt 4.5.2)*.

Default Einstellung (das CAN-Interface hat die Interfacenummer 1):

```
/if/led0/fin +hd1 +l
```

Mögliche Optionen:

```
<option> ::={<reset>}
```

`reset` Initialisiert die LEDs neu

4.3 Filterung

4.3.1 Filterkonzept:

Die Filterung basiert auf den CAN-Identifiern. Das CAN-Ethernet Gateway verarbeitet die Daten von Ethernet bzw. CAN in einem Datenpool. Mittels der Filterregeln wird bestimmt, welche Nachrichten aus diesem Datenpool weitergeleitet werden. Damit kann der Datenverkehr reduziert werden, wenn z.B. nur noch Nachrichten einer bestimmten Gruppe von CAN-Identifiern (CAN-IDs) weitergeleitet werden. Weiterhin können gezielt einzelne CAN-Identifier ausgewählt werden.
Die Filterregeln können durch Aufruf der Filterdateien eines Interfaces bestimmt werden. Sind keine Filterregeln vorhanden, so werden alle Nachrichten weitergeleitet.

4.3.2 Eingangsfilter:

Der Eingangsfilter legt fest, welche Nachrichten vom Interface-Eingang akzeptiert werden (`fin`). Beispielsweise gelangen beim CAN-Interface nur die Nachrichten in das CAN-Ethernet Gateway, deren CAN-IDs in der fin-Funktion festgelegt ist.

4.3.3 Ausgangsfilter:

Der Ausgangsfilter legt fest, welche Nachrichten aus dem Datenpool des Gateways weitergeleitet werden (`fout`).

4.3.4 Filterbeschreibung (Syntax)

Format: `fin <default> {<rule>}*`
`<default> ::= {+ | -}[{l | h}]`
`<rule> ::= {+ | -}[{l | h}][r][d]{0-9}* [<idl> [<idh>]]`

Bedeutung dieser Filterregel:

`+`	die Nachricht wird angenommen
`-`	die Nachricht wird verworfen
`h`	die Nachricht wird hoch priorisiert behandelt, sofern vom Interface unterstützt (LED-Interface: Fehler werden mittels CAN-LEDs signalisiert)
`l`	die Nachricht wird niedrig priorisiert behandelt (LED-Interface: Fehler werden mittels Ethernet-LEDs signalisiert)

Bestimmung des Geltungsbereichs der Filterregel:

`<idl>`	erster CAN-Identifier (Zahlenangaben hexadezimal) für den diese Filterregel gilt. Werden keine CAN-Identifier angegeben, gilt die Filterregel für beliebige CAN-Identifier.
`<idh>`	letzter CAN-Identifier (Zahlenangaben hexadezimal) für den diese Filterregel gilt. Wird nur ein CAN-Identifier angegeben, gilt die Filterregel nur für diesen CAN-Identifier.
`r`	diese Regel gilt für CAN-RTR[1]-Frames
`d`	diese Regel gilt für CAN-Daten-Frames
`0-9`	diese Regel gilt für Nachrichten, die von Interface mit der Interface-Id x kommen *(Ermittlung der Interface-Id mit dem Kommando `ls` siehe Abschnitt 4.5.2)*. Ist keine Nummer angegeben werden alle Interfaces akzeptiert.

Wenn einzelne CAN-Identifier gefiltert werden kann `<idh>` entfallen.

[1] RTR: Remote Transmission Request Frame

Beispiel 1:

```
cd /if/can0
/if/can0 > fout +l +hrd0123 10A 200 –r23 300
```

Sie bewirkt, dass vom CAN-Ethernet Gateway empfangene RTR-Frames und Daten-Frames von den Interfacen 0 bis 3 kommend und einem CAN-Identifier zwischen 0x10A und 0x200 (inklusive) hochpriorisiert gesendet werden. Weiterhin werden alle RTR-Frames von den Interfacen 2 und 3 mit Identifier 0x300 verworfen. Alle restlichen vom CAN-Ethernet Gateway kommenden Nachrichten werden niedrig priorisiert gesendet.

Beispiel 2:

```
cd /if/can0
/if/can0 > fin - +ld 400 400 +ld 2c0 2c0
```

Es werden nur die CAN-Nachrichten mit den CAN-Identifiern 0x400 und 0x2C0 vom CAN-Interface an die anderen Interfaces weitergeleitet.

Beispiel 3:
Filterregel können kombiniert werden.

```
/if/can0 > fin + -ld 180 57F
/if/can0 > fin -ld 700 700
```

Es werden alle CAN-Nachrichten vom CAN-Interface an die anderen Interfaces weitergeleitet, mit Ausnahme der CAN-Identifier zwischen 0x180 und 0x57F und mit dem CAN-Identifier 0x700

Durch Löschen einer Filterdatei mit dem Kommando `rm` werden die Filterregeln gelöscht. Die Datei `fin` bleibt erhalten.

Die Ausgabe der eingestellten Filterregeln erfolgt durch Aufruf der Filterdatei.

Beispielausgabe für die Konfiguration des Beispiels 3:

```
/if/can0 > fin
Usage: fin
    if-id: 1
    Current filter:
        -lds 180 57f
        -lde 180 57f
        -lds 700 700
        -lde 700 700
        default: +l
/if/can0 >
```

Hinweis:
Die Angabe der CAN-Identifier erfolgt hexadezimal, ohne Angabe eines vorangestellten 0x oder eines folgenden h.

4.4 Dateisystem

4.4.1 Aufbau

Im CAN-Ethernet Gateway ist ein Dateisystem integriert. Dieses ermöglicht es zu Laufzeit Konfigurationsänderungen des Gateways durchzuführen und über Script-Dateien die automatische Konfiguration zu steuern. Weiterhin besteht die Möglichkeit, Dateien in einem EEPROM nichtflüchtig zu hinterlegen.
Die Struktur des Dateisystems ist in dem *Abbildung 15* dargestellt und hat folgende festgelegte Struktur.

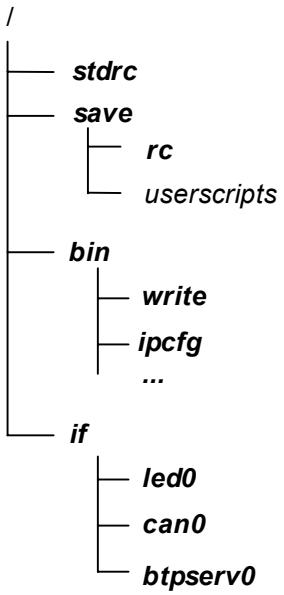

Abbildung 15: Aufbau Dateisystem

Das Dateisystem befindet sich im RAM. Nach einem Spannungsausfall oder Reset wird das Dateisystem mit der Default-Konfiguration initialisiert *(siehe Abschnitt 5)*.
Eine Besonderheit stellt das Verzeichnis `/save` dar. Alle darin enthaltenen Files können im EEPROM gespeichert werden.
Zur Navigation innerhalb des Dateisystems stehen Kommandos zur Verfügung *(siehe Abschnitt 4.5)*.

4.4.2 Datenablage im EEPROM

Zur Speicherung von Konfigurationsdaten steht ein EEPROM zur Verfügung. Alle Dateien, die nichtflüchtig gespeichert werden sollen, müssen sich im Verzeichnis `/save` befinden. Das Speichern in den EEPROM muss durch den Anwender aktiv durch Aufruf des Kommandos `sync` *(siehe Abschnitt 4.5.8)* gestartet werden.
Durch den Anwender sind die Dateien im Verzeichnis `/save` veränderbar. Die wichtigste Datei in diesem Verzeichnis ist das Konfigurations-Script `rc`. Diese wird beim Start der Firmware ausgeführt (vorausgesetzt Schalter „DEFT“ = OFF) und enthält alle Konfigurationsdaten des CAN-Ethernet Gateways *(siehe Abschnitt 5.3)*.

4.5 Beschreibung des Befehlssatzes

Für allgemeine Funktionen wie Verzeichniswechsel oder Anzeigen von Dateien werden Unix-Kommandos benutzt. Im folgenden Kapitel werden die verfügbaren Kommandos beschrieben. Optionen (z.B. `ls -l`) sind jedoch nicht verfügbar.

4.5.1 cd

Format: `cd <dir>`
`cd ..` wechselt zum übergeordneten Verzeichnis

Bedeutung: Das Kommando `cd` dient zum Wechsel in ein angegebenes Verzeichnis `<dir>` bzw. wechselt zum übergeordneten Verzeichnis.

4.5.2 ls

Format: `ls [<dir>]`

Bedeutung: Das Kommando `ls` zeigt die Dateien des angegebenen bzw. des aktuellen Verzeichnisses an. Die Interface-Id, die für die Angabe von Filterregeln benötigt wird, wird bei dem Kommando mit ausgegeben. siehe Beispiel, das Interface `can0` hat die Interface-Id 1 (`id:1`), dem Interface `udp0` wurde die Interface-Id 4 (`id:4`) zugeordnet

Beispiel:

`/ >ls /if↵` zeigt das Verzeichnis `/if`

```
 ..                typ:0xc2
 led0           use:0 typ:0xa0 ch:4 id:0
 can0           use:0 typ:0xa0 ch:4 id:1
 udpserv0       use:0 typ:0xa0 ch:4 id:2
 tcpserv0       use:0 typ:0xa0 ch:4 id:3
 udp0           use:0 typ:0xa0 ch:4 id:4
```

4.5.3 mkif

Format: `mkif <type> <if-num> [<if-name>]`

Bedeutung: Der Befehl `mkif` erstellt ein neues Interface vom Typ `<type>` *(siehe Tabelle 1)*und gliedert es in die Verarbeitungskette des Gateways ein. Das Interface erscheint unter dem angegebenen Name `<if-name>` bzw. unter dem Typname mit angehängter Nummer innerhalb des Verzeichnisses `/if`. Die Nummer `<if-num>` wird fortlaufend vergeben. Die verfügbaren Interfaces sind im *Abschnitt 4.2* beschrieben.
Entfernt werden kann das Interface durch den Befehl `rm` (siehe Abschnitt 4.5.5).

Es werden folgende Dateien erzeugt:

`/if/<if-name>/..`	Link auf Elternverzeichnis
`/if/<if-name>/conf`	Statische Konfiguration des Interfaces (Format ist Interface spezifisch). Diese Datei ist für zukünftige Erweiterungen vorgesehen und bis jetzt ungenutzt.
`/if/<if-name>/fin`	Konfiguration der Eingangsfilter, d.h. für Nachrichten von der Schnittstelle zum Gateway
`/if/<if-name>/fout`	Konfiguration der Ausgangsfilter, d.h. für Nachrichten vom Gateway zur Schnittstelle

Beispiel:

`/ >mkif tcp 0 tcpclient` erzeugt ein Interface mit dem Namen `tcpclient`

`/ >mkif can` erzeugt ein CAN-Interface mit dem Namen `can0`

4.5.4 mem

Format: `mem`

Bedeutung: Der Befehl `mem` gibt die freien Speicherressourcen aus. Bei der Anlage neuer Interfaces ist darauf zu achten, dass ausreichend Speicher zur Verfügung steht. Dies erfolgt mit dem Kommando `mem`.

Die Speicherbereiche haben folgende Bedeutung:

`Eeprm`	Daten im EEPROM zur nichtflüchtigen Ablage von Files
`lwIP`	interner Speicher des TCP/IP Stack
`SysSt`	Systemstack
`UsrSt`	Userstack
`Gcm`	Gatewayapplikation

Beispiel:

```
/ > mem↵

Free file handles: 40
Memory  total   free   max  blocks
Eeprm:   2026   1668   1442  2
lwIP :   8192   8090   8090  1
SysSt:   2410   2148   1908  2
UsrSt:   9000   7058   6658  2
Gcm  : 150000 144792  65528  5
```

4.5.5 rm

Format: `rm <fname>`

Bedeutung: Mit dem Kommando `rm` können Dateien, Verzeichnisse und Interfaces gelöscht werden. `<fname>` entspricht dem Namen der zu löschenden Datei, des zu löschenden Verzeichnisses oder Interfaces. Zum Löschen von Files im EEPROM ist nach dem Aufruf von `rm` im Verzeichnis `/save` das Kommando `sync` *(siehe Abschnitt 4.5.8)* zu rufen.

Beispiel:

`/ >rm /save/rc↵`	löscht das Konfigurationsfile `rc` im Verzeichnis `/save`
`/ >sync`	schreibt Verzeichnis `/save` in den EEPROM

4.5.6 write

Format: `write <fname>`

Bedeutung: Mit dem Befehl `write` wird eine ASCII-Datei mit dem Namen `<fname>` angelegt.
Soll eine vorhandene Datei geändert werden, so ist diese über `cat` auszugeben, mit `rm` zu löschen und über die Zwischenablage entsprechend geändert mit `write` neu zu schreiben. Das Kommando wird mit zwei Leerzeilen und der Tastenkombination `Strg+D` abgeschlossen. Die Verwendung des Kommandos ist im *Abschnitt 3.5* beschrieben.

4.5.7 cat

Format: `cat <fname>`

Bedeutung: Das Kommando `cat` gibt den Inhalt der Datei `<fname>` aus.

Beispiel: gibt den Inhalt des Standard-Konfigurationsfiles aus`/ > cat stdrc`↵

```
siocfg 9600
ipcfg 192.168.10.111 255.255.255.0 192.168.10.1
mkif led
/if/led0/fin +l +hd1
mkif can
mkif udpserv
mkif tcpserv
/if/can0 bus:0 baud:0 userbaud:0 canid:fe on
/ >
```

4.5.8 sync

Format: `sync`

Bedeutung: Das Kommando `sync` schreibt nicht gespeicherte Dateien aus dem Verzeichnis `/save` ins EEPROM. Damit stehen die Dateien nach einem PowerOn oder Reset wieder zur Verfügung *(siehe Abschnitt 4.4.2)*.

4.5.9 version

Format: `version`

Bedeutung: Das Kommando `version` gibt die Version der CAN-Ethernet Gateway-Firmware und die MAC[1]-Adresse aus.

Beispiel:

```
/ > version↵
 SYSTEC CAN-Ethernet Gateway V2.11.50
 MAC: 0x00:0x40:0xDC:0x00:0x0E:0xD3
/ >
```

4.5.10 exit

Format: `exit`

Bedeutung: Das Kommando `exit` beendet die Telnet-Sitzung. `Strg+D` erzielt die gleiche Wirkung.

4.5.11 reset

Format: `reset`

Bedeutung: Das Kommando `reset` startet das CAN-Ethernet Gateway neu (es wird ein Softwarereset ausgelöst). Entsprechend der Schalterstellung „DEFT“ wird das entsprechende Konfigurations-Script geladen.

[1] MAC: Media Access Control: Ethernetadresse, die für jedes CAN-Ethernet Gateway einmalig existiert.

4.5.12 ipcfg

Format: `ipcfg [<local-ip> <mask><gw>]`

Bedeutung: Das Kommando `ipcfg` konfiguriert die IP-Adresse, die Subnet-Mask und das Standard-Gateway für die Ethernet-Verbindung. Der TCP/IP Stack wird mit den übergebenen Werten initialisiert. Das Kommando `ipcfg` ist unbedingt vor der Benutzung des Telnet-Zugangs und des BTP-Interfaces aufzurufen, da sonst keine Kommunikation über die Ethernet-Schnittstelle möglich ist. Er sollte deshalb als erstes Kommando im Konfigurations-Script `/save/rc` stehen und dort eine eindeutige IP-Adresse im Netz enthalten. Die Anzeige der aktuellen Konfiguration erfolgt durch Eingabe des Kommandos ohne Parameter.

Beispiel:

Einstellung der Parameter

```
/>ipcfg 192.168.10.117 255.255.255.0 192.168.10.1↵
```

Ausgabe der aktuellen Konfiguration

```
/>ipcfg ↵

ipcfg ipcfg <local-ip> <mask> <gw>
local-ip: 192.168.10.117
mask:     255.255.255.10
gw:       192.168.10.1
```

4.5.13 siocfg

Format: `siocfg <Baudrate>`

Bedeutung: Das Kommando `siocfg` schaltet die RS232-Schnittstelle für Kommandoeingabe ein und konfiguriert die benutzte Baudrate `<Baudrate>`. Es werden folgende Baudraten unterstützt:

- 4800Baud
- 9600Baud
- 19200Baud
- 38400Baud
- 57600Baud
- 115000Baud

Fest eingestellt sind Hardwareflusskontrolle, 8 Datenbits, ein Stoppbit und keine Parität.

Beispiel:

```
/ >siocfg 57600↵
```

4.5.14 ipaccept

Format: `ipaccept -<param> [<IpAddress>]`

Bedeutung: Das Kommando `ipaccept` ermöglicht die Konfiguration eines Filters für IP-Adressen. Ist dieser Filter leer, kann das Gateway von allen IP-Adressen eingehende Verbindungen annehmen. Wurden IP-Adressen im Filter hinterlegt, erfolgt die Verbindungsannahme nur von den im Filter eingetragenen IP-Adressen. Es können maximal 10 verschiedene IP-Adressen im Filter angelegt werden.

Der Status des Filters kann mit Hilfe des Kommandos `ipaccept –all` abgefragt werden. Daraufhin erfolgt die Ausgabe aller im IP-Adressfilter eingetragenen IP-Adressen auf der Konsole.

Das Kommando `ipaccept –a <IpAddress>` fügt eine neue IP-Adresse zum Filter hinzu. Ist der Filter bereits mit 10 Einträgen gefüllt, erfolgt die Ausgabe der Fehlermeldung ***"No free table entry!"*** auf der Konsole.

Das Kommando `ipaccept –d <IpAddress>` löscht die angegebene IP-Adresse aus dem Filter. Beim Löschen einer nicht im Filter enthaltenen IP-Adresse erfolgt die Ausgabe der Fehlermedlung ***"IpAddress xxx.xxx.xxx.xxx does not exist!"***

Beispiel:

Hinzufügen der IP-Adressse 192.168.10.111 zum Filter

```
/ > ipaccept –a 1921.163.10.111
```

Löschen der IP-Adressse 192.168.10.179 aus dem Filter

```
/ > ipaccept –d 192.168.10.179
```

Ausgabe aller konfigurierten IP-Adressen

```
/ > ipaccept –all

IpAddress Table

192.168.10.111
192.168.10.156
…
…
```

Achtung:
Beim Konfigurieren des IP-Adressfilters ist darauf zu achten, dass stets eine IP-Adresse eines Service-PC's einzutragen ist, wenn man das Gateway weiterhin über eine Telnet-Verbindung erreichen und konfigurieren möchte!

5 Konfiguration des Gateway

5.1 Grundlagen

Die Konfiguration des CAN-Ethernet Gateway kann über zwei Wege erfolgen; die RS232-Schnittstelle mit einem Terminal-Programm *(siehe Abschnitt 3.5.2)* oder eine Telnet-Verbindung über Ethernet *(siehe Abschnitt 3.5.3)*.

Die Konfiguration des CAN-Ethernet Gateway erfolgt über Konfigurations-Scripte. Es gibt die folgenden zwei Konfigurations-Scripte:

`/stdrc` Standardkonfiguration,
ist fest in der Firmware;
wird ausgeführt, wenn der Schalter „DEFT“ auf ON steht
oder kein `/save/rc` existiert
oder ein Fehler des EEPROM erkannt wurde
oder ein Firmwareupdate eingespielt wurde
enthält folgende Einträge:

```
siocfg 9600
ipcfg 192.168.10.111 255.255.255.0
192.168.10.1
mkif led
/if/led0/fin +l +hd1
mkif can
mkif udpserv
mkif tcpserv
/if/can0 bus:0 baud:0 userbaud:0 canid:fe on
```

Das CAN-Ethernet Gateway befindet sich nach dem Start mit Standardkonfiguration im passiven Zustand und wartet auf Verbindungsanfragen.

`/save/rc` kundenspezifische Konfiguration;
kann mit den Kommandos `write` und `sync` erstellt und im EEPROM gespeichert werden
wird ausgeführt wenn der Schalter „DEFT“ auf OFF steht
wird vom Kunden entsprechend seiner Anforderungen erstellt

es gibt Beispiel-Konfigurations-Scripte für Client und Serveranwendungen *(siehe Abschnitt 3.5.2)*
Das CAN-Ethernet Gateway befindet sich nach dem Start mit kundenspezifischer Konfiguration in dem eingestellten Zustand und kann Verbindungen aufbauen oder auf einen Verbindungsaufbau warten.

Entsprechend der oben genannten Bedingungen wird bei einem PowerOn oder Softwarereset das entsprechende Konfigurations-Script ausgewählt und ausgeführt.

5.2 Beispiel für ein kundenspezifischen Konfigurations-Script

In der folgenden Übersicht wird am Beispiel der Datei `rc` die Reihenfolge der Kommandos und Parameter gezeigt.

`siocfg 9600`	Einschalten der RS232-Schnittstelle und Einstellung der Baudrate
`ipcfg 192.168.10.117 255.255.255.0 192.168.10.1`	Einstellung der IP-Adresse, Subnet-Mask und Standard Gateway
`mkif led`	Anlegen des LED-Interface für Statusanzeigen
`/if/led0/fin +l +hd1`	Konfiguration des LED-Interface
`mkif can`	Anlegen eines CAN-Interfaces
`mkif udp`	Anlegen eines UDP-Client-Interface für aktiven Verbindungsaufbau zu einem anderen Gateway
`mkif udpserv`	Anlegen des UDP-Server-Interface
`/if/can0 baud:0 canid:fe on`	Konfiguration des CAN-Interface für 1MBit, mit CAN-Fehlernachricht, CAN-Interface wird eingeschaltet
`/if/udp0 to:192.168.10.115 on`	Konfiguration des UDP-Client-Interface mit Angabe der Ziel-IP-Adresse, das Interface wird eingeschaltet und startet die Verbindung

5.3 Erstellung eines Konfigurations-Scripts

Für die Erstellung eines Konfigurations-Scripts sind folgende Schritte erforderlich.

Ausgabe der bisherigen Konfigurationsdatei:

```
cd /save
cat rc
```

Die Ausgabe kann markiert und in einem Editor eingefügt werden. Als Editor eignet sich beispielsweise „Notepad“. Die Konfiguration kann dann im Editor bearbeitet und an die Erfordernisse angepasst werden.
Vor dem Zurückspielen der Konfiguration muss die Datei `/save/rc` mittels `rm rc` gelöscht werden.Mit `write rc` wird die Eingabe der Konfiguration gestartet. Durch Markieren im Editor und Kopieren in das Terminalfenster erfolgt die Übertragung zum Gateway. Die Eingabe des Kommandos `write` wird mit `Strg+D` abgeschlossen.
Über `cat rc` kann die Konfiguration noch einmal überprüft werden und abschießend sind die Daten mit dem Befehl `sync` in dem EEPROM-Speicher zu sichern. Ein Neustart des Gateways aktiviert die neue Konfiguration (Schalter „DEFT“ muss auf OFF gestellt werden).

5.4 Rücksetzen in die Standardkonfiguration

Bei Wartungsarbeiten vor Ort, einem unbekannten Zustand des CAN-Ethernet Gateways oder einer fehlerhaften Konfiguration ist es notwendig, das CAN-Ethernet Gateway auf einer festen IP-Adresse ansprechen zu können. Dazu dient die IP-Adresse 192.168.10.111 der Standardkonfiguration.

Um das Gerät auf der Default-IP-Adresse 192.168.10.111 einzustellen, muss der Schalter „DEFT“ auf ON gestellt werden. Danach ist die Spannungsversorgung kurz zu unterbrechen (Auslösen eines PowerOn-Reset).

Hinweis:
Bereits gespeicherte Konfigurationen (`/save/rc`) werden nicht gelöscht.

Alternativ ist der Zugang über die RS232-Schnittstelle möglich. Dabei erfolgt der Zugriff auf das CAN-Ethernet Gateway unabhängig von der IP-Adresse.

5.5 Passwortvergabe

Es besteht die Möglichkeit ein Passwort für den Zugriff via Telnet und RS232 zu hinterlegen. Dazu wird im Verzeichnis `/save` die Datei `passwd` verwendet. Der Inhalt der Datei entspricht dem Passwort. Die Anlage der Datei erfolgt mit den Kommandos `write` und `sync`.

Beispiel:

`/>cd save`	Wechsel in das Verzeichnis `/save`
`/save >write passwd`	Anlegen der Datei passwd
`xyz0815`	Festlegen des Passwort Abschluss der Eingabe mit `Strg+D`
`/save >sync`	Speicherung des Passwort im EEPROM
`/save >reset`	Softwarereset
`Bye. Enter password:`	Aufforderung zur Eingabe des Passwort bei der Anmeldung einer neuen Sitzung über Telnet oder RS232

Die Eingabe eines falschen Passwortes wird mit `Access denied` quittiert.

6 Fehlerbehandlung

6.1 Fehlersignale des CAN-Ethernet Gateway

Folgende Fehlerzustände werden signalisiert:

- Verbindungsaufbau über BTP-Interface nicht möglich
  mögliche Ursachen:
  keine Verbindung zum konfigurierten Server möglich
  oder Verbindung abgelehnt
  Gateway: Ethernet-Error-LED leuchtet

- CAN-RxBuffer-Überlauf, CAN-Nachrichtenverlust
  mögliche Ursachen:
  zu hohe CAN-Buslast
  Empfangspuffer zu klein gewählt *(Einstellung siehe Abschnitt 4.2.4)*
  Gateway: CAN-Error-LED blinkt kurz

- CAN-TxBuffer-Überlauf
  CAN-Nachrichten werden zu schnell in CAN-Sendepuffer eingereiht (bspw. durch hohe Buslast und niedrige Priorität der zu sendenden CAN-ID)
  allgemein: Fehlernachricht einreihen ist möglich, wenn CAN-interner hochpriorisierter Puffer nicht übergelaufen ist
  Einreihung der verursachenden CAN-Nachricht wird weiterhin versucht.
  Gateway: CAN-Error-LED blinkt kurz

- BTP-TxBuffer-Überlauf
  keine CAN to BTP-Puffer verfügbar -> Nachrichtenverlust
  Gateway: Ethernet-Error-LED leuchtet

- BTP-RxBuffer-Überlauf
  keine BTPv to CAN-Puffer verfügbar -> Nachrichtenverlust
  Gateway: Ethernet-Error-LED leuchtet

- Fehler bei Empfang oder Versand per BTP
  z.B. wenn Empfänger keinen freien Puffer hat oder bei Timeout
  allgemein: CAN to TCP/IP-Sendepuffer wird verworfen -> Nachrichtenverlust
  Gateway: Ethernet-Error-LED leuchtet

- CAN-Busoff Fehler
  mögliche Ursachen:
  CAN-Bus-Verkabelung,
  falsche CAN-Bitrate,
  Hardwarefehler
  Gateway: CAN-Error LED blinkt mit dem Tastverhältnis 50:50 wird solange angezeigt, bis die Sendung bzw. der Empfang eines CAN-Telegramme erfolgreich war

- CAN-ACK Fehler
  mögliche Ursachen:
  kein weiteres CAN-Gerät am CAN-Bus angeschlossen,
  CAN-Bus-Verkabelung,
  Hardwarefehler,
  Abschlusswiderstände fehlen
  Gateway: CAN-Error LED blinkt mit dem Tastverhältnis 25:75 wird solange angezeigt, bis die Sendung bzw. der Empfang eines CAN-Telegramme erfolgreich war

Ethernet Error-LED	CAN Error-LED	Fehlerbeschreibung
leuchtet		kein Verbindungsaufbau über BTP-Interface oder Verbindungsunterbrechung
leuchtet		BTP-Sendepuffer-Überlauf (Nachrichtenverlust)
leuchtet		BTP-Empfangspuffer-Überlauf (Nachrichtenverlust)
leuchtet		Fehler bei Empfang oder Versand über BTP (Nachrichtenverlust)
	blinkt kurz	Empfangspuffer-Überlauf (Nachrichtenverlust)
	blinkt im Tastverhältnis[1] 50:50	CAN-Busoff erkannt,
	blinkt im Tastverhältnis[1] 25:75	CAN-ACK Fehler
	blinkt kurz	Sendepuffer-Überlauf

Tabelle 7: Übersicht Fehleranzeige

6.2 Fehlernachrichten über CAN

Um vom CAN-Netz den Zustand des Gateways beurteilen zu können, ist das Versenden von Fehlernachrichten nach dem CANopen-Standard (Emergency-Nachrichten) möglich.
Über die Option `canid:<id>` beim Anlegen des CAN-Interfacess kann der Versand von Fehlernachrichten mit dem anzugebenden Identifier im CAN-Ethernet Gateway eingeschaltet werden.
In der Standardkonfiguration ist der Versand auf der Adresse 0xFE eingeschaltet.

[1] Tastverhältnis: ON-Zeit : OFF-Zeit

Das Format der Fehlernachricht ist wie folgt spezifiziert:

Byte	0	1	2	3	4	5	6	7
Inhalt	Emergency-Code		Error-Register	Interface nummer	Fehler-Code		reser-viert	reser-viert

Tabelle 8: Aufbau der Emergency-Nachricht

Der Emergency-Code kann folgende Werte annehmen:

0x1000: wenn ein neuer allgemeiner Fehler aufgetreten
0x8140: Knoten kommt aus dem Zustand CAN-Busoff
0x0000: Gerät ist fehlerfrei

Das Error-Register ist 0x80, wenn noch Fehler vorhanden sind und 0x00, wenn alle Fehler behoben sind. Danach steht die Nummer des Interfaces, bei dem der Fehler aufgetreten ist, gefolgt von einer Bitmaske, die anzeigt, welche Fehler vorliegen. Bei den 2 Byte langen Codes (Byte 0,1 und 4,5) wird der höherwertige Teil zuletzt gesendet, z.B. 00 10 für 0x1000.

Folgende Bits des Fehlercodes sind spezifiziert:

0x0000	Fehlerfrei
0x0001	Pufferüberlauf beim Empfang
0x0002	Pufferüberlauf beim Senden
0x0004	Pufferüberlauf im CAN-Controller
0x0008	CAN-ACK-Fehler (Acknowledge error)
0x0010	CAN-Warning-Limit erreicht
0x0020	CAN-Passive mode erreicht
0x0040	Gerät befindet sich im CAN-Busoff Zustand
0x0080	Fehler beim Senden der Nachricht
0x0100	Fehler beim Empfang der Nachricht verbunden
0x0400	Die Verbindung des Interfaces ist abgebrochen
0x2000	Sammelfehler des CAN-Controllers, interne Hardwarefehler des Infineon TwinCAN (Stuff-Error, Form-Error, CRC-Error)

Diese Fehlerzustände des Gerätes können auch via Telnet oder RS232 abgefragt werden. Der Zustand (`state`) befinde sich im CAN-Interface. Neben den möglichen Fehlerzuständen geben folgende zusätzliche Bit's Auskunft über bestehende BTP-Verbindungen.

0x0200	Das Interface ist erfolgreich verbunden
0x0800	Signalisierung für Datentransfer

Beispiel:

```
/if > can0↵
Usage: can0 [bus:<num>] [baud:<num>] [userbaud:<hex>]
       [canid:<hex>] [on] [off]

Current settings: bus:0 baud:0 (1MBit/s) userbaud:0x0
                  canid:0xF5 on state:0x850
```

`state = 0x850` zeigt CAN-Busoff und CAN-Warning-Limit sowie Datentransfer via BTP

7 Softwareunterstützung

7.1 Anbindung des CAN-Ethernet Gateways an den PC

Für die Anbindung des CAN-Ethernet Gateway an den PC steht eine WIN32-DLL zur Verfügung, die eine Anzahl an Export-Funktionen zur Verfügung stellt. Mit Hilfe dieser DLL ist es möglich, eigene Anwendungen unter Windows zu entwickeln. Das CAN-Ethernet Gateway kann über diese Treiber-DLL direkt vom PC aus über Ethernet angesprochen werden.

7.2 Treiberinstallation unter Windows

Im Vorfeld ist die Installation der Treiber-DLL für Windows notwendig. Das zugehörige Setup-Programm finden Sie auf der beigelegten CD im Verzeichnis So-1027.

Starten Sie das darin enthaltene Setup-Programm und folgen Sie den Anweisungen auf dem Bildschirm. Die Installation erfolgt dabei standardmäßig in das folgende Verzeichnis:

C:\Programme\SYSTEC-electronic\CAN-Ethernet Gateway_Utility_Disk

Der Installationspfad kann jedoch beliebig verändert werden kann.

Hinweis:
Stellen Sie sicher, dass Sie für die Installation unter den Betriebssystemen Windows 2000, XP, Vista und Windows 7 Administratorrechte besitzen!

Während der Installation wird die Treiber-DLL (EthCan.Dll) je nach Betriebssystem in das entsprechende Windows-System-Verzeichnis kopiert. Des Weiteren erzeugt das Setup-Programm im Installationsverzeichnis, ausgehend vom Default-Installationspfad, folgende Verzeichnisstruktur :

Unterverzeichnis	Inhalt
Demo.Prj	„C“-Demo im Source für MSVC 5.0 bzw. 6.0
Doku	System-Manual CAN-Ethernet Gateway
Include	„C“-Header-Datei für die EthCan.Dll
Lib	EthCan.Lib und EthCan.Dll

Tabelle 9: Verzeichnisstruktur CAN-Ethernet Gateway_Utility_Disk

Das Verzeichnis ***„LIB”*** beinhaltet die Library sowie die zugehörige DLL. Im Verzeichnis ***„Include“*** finden Sie die Header-Datei zur ***„EthCan.Dll“*** , die alle Prototypen der PUBLIC-Funktionen der DLL sowie aller verwendeten Datenstrukturen und Datentypen beinhaltet. Diese Header-Datei ist bei der Entwicklung eigener Applikationen aufsetzend auf der DLL mit in das Entwicklungsprojekt einzubinden. Das Verzeichnis „***Doku“*** enthält das System-Manual des CAN-Ethernet Gateways in Form einer PDF-Datei.
Im Verzeichnis ***„Demo.Prj“*** finden Sie ein Demo-Projekt in Form eines Visual-Studio-Projektes. Es beinhaltet ein „C“-Source-File sowie zugehöriges Header-File, welches die Anwendung der DLL-Funktionen in Form eins Demo-Programms zeigt.

7.3 Die Dynamic Linked Library *EthCan.Dll*

Die Dynamic Linked Library (EthCan.Dll) ist eine Funktionslibrary für Anwendungsprogramme. Sie dient als Schnittstelle zwischen der Windows-Socket und einem Anwendungsprogramm. Sie sorgt für die Verwaltung der angeschlossenen CAN-Ethernet Gateways und für die Übersetzung der CAN-Nachrichten in IP-Pakete und umgekehrt.
Zur Einbindung der DLL in ein eigenes Projekt, muss die EthCan.Lib mit zum Projekt hinzugefügt werden. Dabei wird die DLL automatisch geladen, wenn das Anwendungsprogramm gestartet wird. Wird die LIB nicht zum Projekt hinzugelinkt, so muss die DLL mit der Windows-Funktion *LoadLibrary ()*geladen und die Libraryfunktionen mit der Funktion *GetProcAdress ()* hinzugefügt werden.
Die STDCALL-Direktive der Aufruffunktionen der DLL sorgt für eine standardisiere Aufrufschnittstelle zum Anwender. So ist sichergestellt, dass auch Anwender einer anderen Programmiersprache (Pascal, ...) diese Funktionen nutzen können.

7.3.1 Das Konzept der *EthCan.Dll*

Mit der ***EthCan.Dll*** können bis zu 5 CAN-Ethernet Gateways innerhalb einer Applikation gleichzeitig angesprochen werden. Des Weiteren kann von einer weiteren Applikation aus wiederum auf bis zu 5 CAN-Ethernet Gateways zugegriffen werden, die zudem die gleichen Remote-Adressen haben, wie in Applikation 1, da auf dem CAN-Ethernet Gateway gleichzeitig mehrere Interface angelegt werden können, so dass eine Mehrfachverbindung aus unterschiedlichen Applikationen heraus möglich ist. Es ist jedoch **nicht** möglich von einer Applikation aus mehrere Verbindungen zu ein und demselben CAN-Ethernet Gateway aufzubauen.

Bei der Verwendung dieser DLL entstehen für jedes CAN-Ethernet Gateway zwei Zustände für die Software. Nachdem das Anwendungsprogramm gestartet und die DLL geladen wurde, befindet sich die Software im Zustand DLL_INIT. Dabei wurden alle notwendigen Ressourcen für die DLL angelegt. Wird die Library-Funktion ***EthCanInitHardware ()*** gerufen, so wechselt die Software in den Zustand HW_INIT. Hier sind alle Ressourcen angelegt, die für die Kommunikation mit dem CAN-Ethernet Gateway notwendig sind.

Mit der Library-Funktion ***EthCanDeinitHardware ()*** gelangt man vom Zustand HW_INIT in den Zustand DLL_INIT zurück. Erst jetzt darf das Anwendungsprogramm beendet werden.

Zustand	Funktionsumfang
DLL_INIT	*EthCanGetVersion ()* *EthCanInitHardware ()*
HW_INIT	*EthCanGetVersion ()* *EthCanGetStatus ()* *EthCanDeinitHardware ()* *EthCanRreadCanMsg()* *EthCanWriteCanMsg()* *EthCanResetCan()* *EthCanGetConnectionState()*

Tabelle 10: Funktionsumfang der Softwarezustände

Werden mehrere CAN-Ethernet Gateways innerhalb einer Anwendung verwendet, so sind diese Zustände für jedes CAN-Ethernet Gateway zu verstehen. Während das erste CAN-Ethernet Gateway bereits im Zustand DLL_INIT ist, kann sich das zweite noch im Zustand HW_INIT befinden.

7.3.2 Das Funktionsinterface der *EthCan.Dll*

Dieses Kapitel beschreibt die Interface-Funktionen der CAN-Ethernet-DLL in ihrer Aufgabe, Anwendung und ihren Rückgabewerten. Die Anwendung der Funktionen ist durch Code-Beispiele veranschaulicht. Alle Übergabeparameter der Funktionen sind so gewählt, dass die DLL auch mit Programmiersprachen wie Pascal oder Visual Basic eingesetzt werden kann.

7.3.2.1 EthCanGetVersion

Syntax:

DWORD STDCALL EthCanGetVersion (void);

Verwendbarkeit: DLL_INIT, HW_INIT

Bedeutung:

Die Funktion liefert die Softwareversionsnummer der ***EthCan.Dll*** zurück.

Parameter: keine

Rückgabewert:

Der Rückgabewert ist die Softwareversionsnummer im *DWORD*-Format. Sie ist wie folgt aufgebaut:

Bit 0 bis 7: höherwertige Versionsnummer im Binärformat
Bit 8 bis 15: niederwertige Versionsnummer im Binärformat
Bit 16 bis 31: Release-Versionsnummer im Binärformat

Anwendungsbeispiel:

```
DWORD dwVersion;
char szVersion[16];
...
// Versionsnummer holen
dwVersion = EthCanGetVersion ();

// in einen String umwandeln
wsprintf( szVersion, „V%d.%02d.r%d“, (dwVersion&0xff),
         (dwVersion&0xff00)>>8, dwVersion>>16);
```

7.3.2.2 EthCanInitHardware

Syntax:

DWORD STDCALL EthCanInitHardware(
tEthCanHandle pEthCanHandle_p,*
tEthCanHwParam pEthCanHwParam_p,*
tEthCanCbConnectFct fpEthCanCbConnectFct_p
LPARAM pArg_p);

Verwendbarkeit: DLL_INIT

Bedeutung:

Diese Funktion initialisiert alle notwendigen Datenstrukturen und stellt im Anschluss eine Verbindung zum adressierten CAN-Ethernet Gateway her. Die dafür notwendigen Parameter wie IP-Adresse, Port-Nummer usw. werden in Form einer Adresse auf eine Hardware-Parameterstruktur (Parameter 2) übergeben.
Bei der Anwendung dieser Funktion wird generell zwischen zwei Aufrufmodi unterschieden:

1. Die Funktion arbeitet im so genannten ***„Blocked Mode“***, wenn als Pointer für die Callback-Funktion (Parameter 3) ein NULL-Pointer übergeben wird. Sie kehrt erst dann zurück, wenn eine erfolgreiche Verbindung zum CAN-Ethernet Gateway aufgebaut werden konnte oder ein Fehler, z.B. in Form eines Timeouts, aufgetreten ist.
2. Die Funktion arbeitet im so genannten ***„Nonblocked Mode“***, wenn eine gültige Adresse auf eine Callback-Funktion übergeben wurde. Dabei initialisiert die Funktion alle notwendigen Datenstrukturen und initiiert den Verbindungsaufbau, ohne auf den erfolgreichen Abschluss zu warten. Der Status des Verbindungsaufbaus erfolgt dann über die Callback-Funktion, die in der Applikationsschicht angelegt werden muss. Sie liefert den aktuellen Verbindungsstatus und wird immer dann aus der DLL heraus gerufen, wenn sich der

Verbindungsstatus ändert. Somit kann auf eventuelle Verbindungsabbrüche in der Applikation entsprechend reagiert werden.

Parameter:

***pEthCanHandle_p**:* Adresse des Instanz-Handle des CAN-Ethernet Gateways

Diese Variable ist ein Zeiger vom Typ ***tEthCanHandle***. Bei erfolgreicher Initialisierung enthält diese Adresse ein gültiges Hardware-Handle, welches als Instanz-Handle dient. Dieses Instanz-Handle ist zu sichern und beim Aufruf aller weiteren Funktionen dieser Instanz als Übergabeparameter anzugeben.

***pEthCanHwParam_p**:* Adresse auf die Hardwareparameter-Struktur

Diese Variable ist eine Adresse auf eine Hardware-Parameterstruktur vom Typ ***tEthCanHwParam***. Sie besitzt folgenden Aufbau:

```
typedef struct
{
  DWORD         m_dwIpAddress;         //IP-Adresse
  WORD          m_wPort;               //Port-Nummer
  tUsedProtocol m_UsedProtocol;        //Protokoll (UDP oder TCP)
  DWORD         m_dwReconnectTimeout; //Timeout für "Reconnect"
  DWORD         m_dwConnectTimeout;    //Timeout für "Connect"
  DWORD         m_dwDisconnectTimeout;//Timeout für "Disconnect"

}tEthCanHwParam;
```

Abbildung 16: Aufbau der Hardwareparameterstruktur

Diese Struktur ist vor Übergabe an die Funktion entsprechend auszufüllen. Die IP-Adresse und die Port-Nummer entsprechen der Remote-Adresse (IP-Adresse des CAN-Ethernet Gateways), zu der eine Verbindung aufgebaut werden soll. Sie sind in folgendem Format anzugeben:

```
#define IP_ADDR_DEFAULT ((192 << 0)+(168 << 8)+ (10 << 16)+(111 << 24))
#define IP_PORT_DEFAULT (8234)
```

Für das zu verwendende Übertragungsprotokoll stehen UDP und TCP zur Auswahl.

```
typedef enum
{
  kUseTCP = 0x00, // TCP Protokoll
  kUseUDP = 0x01  // UDP Protokoll

}tUsedProtocol;
```

Abbildung 17: Übertragungsprotokolle CAN-Ethernet Gateway

Die Member-Variable ***m_dwReconTime*** beschreibt die Zeitdauer, die nach einem eingetretenen Verbindungsabbruch gewartet werden soll, bis ein erneuter automatischer Verbindungsaufbau gestartet wird. Ist diese Zeit **0**, erfolgt **kein** erneuter Verbindungsaufbau.
Die Member-Variable ***m_dwConnectTimeout*** hat nur dann Bedeutung, wenn die Init-Funktion im ***„Blocked Mode“*** aufgerufen wird. Sie beschreibt die Zeit, nach der die Init-Funktion zurückkehrt, wenn kein erfolgreicher Verbindungsaufbau initiiert werden konnte. Ist diese Zeit **0**, wird ein Default-Timeout von **5s** eingestellt.
Das Gleiche gilt für die Member-Variable ***m_dwDisconnectTimeout***, die den Timeout für die Deinit-Funktion festlegt, nach der spätestens eine aktive Verbindung geschlossen sein muss.

fpEthCanCbConnectFct_p*:*

Adresse auf die Callback-Funktion für den Verbindungsstatus des CAN-Ethernet Gateways.

Dieser Wert kann bei der Übergabe an die Init-Funktion NULL sein, dass heißt, es ist keine Callback-Funktion vorgesehen.
Soll über eine Callback-Funktion auf die Änderung des Verbindungsstatus reagiert werden, so ist diese wie folgt zu deklarieren:

void PUBLIC EthCanConnectControlFct(
tEthCanHandle EthCanHandle_p,
DWORD dwConnectionState_p,
LPARAM pArg_p);

Dabei kann ein und die dieselbe Callback-Funktion bei der Initialisierung mehrerer Instanzen angegeben werden. Die Auswertung, bei welcher Instanz sich der Verbindungsstatus geändert hat, erfolgt über das der Callback-Funktion übergebene Instanz-Handle (***EthCanHandle_p***). Es besteht jedoch die Möglichkeit, pro initialisierte Instanz eine eigene Callback-Funktion anzugeben.

Hinweis:
Wird für jede Instanz eine eigene Callback-Funktion angelegt, so ist darauf zu achten, dass unterschiedliche Funktionsnamen vergeben werden, um Compiler- und Linkerfehler zu vermeiden!

Der Parameter ***dwConnectionState_p*** beschreibt den aktuellen Verbindungsstatus und kann folgende Werte annehmen, die durch den Typ ***tConnectionState*** beschrieben sind:

```
typedef enum
{
  kConnecting     = 0,  // Verbindungsaufbau läuft
  kEstablished    = 1,  // Verbindung hergestellt
  kClosing        = 2,  // Verbindungsabbau läuft
  kClosed         = 3,  // Verbindung geschlossen
}tConnectionState;
```

Abbildung 18: Verbindungsstatus CAN-Ethernet Gateway

pArg_p: Adresse auf Argument für die Callback-Funktion

An dieser Stelle kann ein Argument übergeben werden, welches beim Aufruf der Callback-Funktion aus der DLL heraus wieder zurückgeliefert wird.
Beispielsweise ist hier die Übergabe der Adresse auf eine Instanz des CAN-Ethernet Gateways möglich, wenn von einer Applikation mehrere Gateways angesprochen werden sollen, die innerhalb einer Instanztabelle verwaltete werden. Wurde für mehrere Instanzen nur eine Callback-Funktion deklariert, so kann mittels des Argumentenpointers und dessen Zugriff auf die Elemente der Instanztabelle eine Unterscheidung dahingehend getroffen werden, für welche Instanz die Callback-Funktion gerufen wurde.

Da der Übergabeparameter ***pArg_p*** vom Typ ***LAPARAM*** ist, können an dieser Stelle Parameter jeglicher Art übergeben. Dies hängt vor allem von der Applikation ab.

Rückgabewerte: (*siehe Kapitel 7.3.3)*

ETHCAN_SUCCESSFULL
ETHCAN_ERR_RESOURCE
ETHCAN_ERR_ILLHANDLE
ETHCAN_ERR_ILLPARAM
ETHCAN_ERR_HWINUSE
ETHCAN_ERR_HWCONNECT_FAILD
ETHCAN_ERR_MAXMODULES
ETHCAN_ERR_SAL
ETHCAN_ERR_IFBTP

Anwendungsbeispiel:

```
#define IP_ADDR_DEFAULT ((192 << 0)+(168 << 8)+ (10 << 16)+(111 << 24))
#define IP_PORT_DEFAULT (8234)

DWORD          dwRetcode;
tEthCanHandle  EthCanHandle;
tEthCanHwParam EthCanHwParam;

EthCanHwParam.m_dwReconnectTimeout  = 120000;//120s
EthCanHwParam.m_dwIpAddress         = IP_ADDR_DEFAULT;
EthCanHwParam.m_wPort               = IP_PORT_DEFAULT;
EthCanHwParam.m_dwConnectTimeout    = 5000;//5s
EthCanHwParam.m_dwDisConnectTimeout = 5000;//5s
```

ohne Callback-Funktion:

```
// ein CAN-Ethernet Gateway ohne Callbackfunktion initialisieren
dwRetcode = EthCanInitHardware (&EthCanHandle,&EthCanHwParam,NULL,NULL);
```

mit Callback-Funktion:

```
void PUBLIC EthCanConnectControlFct (tEthCanHandle EthCanHandle_p,
                                     DWORD dwConnectionState_p,
                                     LPARAM pArg_p)
{
      switch(dwConnectionState_p)
      {
            //Verbindung wird aufgebaut
```

```
            case kConnecting:……
                  break;

            //Verbindung aufgebaut
            case kEstablished:……
                  break;

            //Verbindung wird abgebaut
            case kClosing:……
                  break;

            //Verbindung abgebaut
            case kClosed:……
                  break;
      }
}

//Ein CAN-Ethernet Gateway mit Callbackfunktion initialisieren
dwRetcode = EthCanInitHardware (&EthCanHandle, &EthCanHwParam,
                                EthCanConnectControlFct,NULL);
```

7.3.2.3 EthCanDeinitHardware

Syntax:

DWORD STDCALL EthCanDeinitHardware (
tEthCanHandle EthCanHandle_p);

Verwendbarkeit: HW_INIT

Bedeutung:

Diese Funktion ist das Komplement zur Initialisierungsfunktion ***EthCanInitHardware()***. Das heißt, diese Funktion arbeitet sowohl im ***„Blocked Mode“*** als auch im ***„Nonblocked Mode“***. Der Aufrufmodus wird durch den Aufrufmodus der Initialisierungsfunktion bestimmt, so dass stets ein Äquivalent besteht.
Die Aufgabe der Funktion ist es, eine bestehende Verbindung kontrolliert abzubauen und eine Deinitialsierung der Datenstrukturen der Instanz vorzunehmen. Der Übergabeparameter ***EthCanHandle_p*** beschreibt die Instanz, deren Verbindung abgebaut werden soll.

1. Im ***„Blocked Mode“*** wird der Verbindungsabbau gestartet und auf den Abschluss des Verbindungsabbaus gewartet. Die Funktion kehrt erst dann zurück, wenn die Verbindung

geschlossen wurde beziehungsweise ein Fehler oder Timeout aufgetreten ist.

2. Im ***„Nonblocked Mode“*** wird lediglich der Verbindungsabbau gestartet ohne auf den Abschluss zu warten. Die Funktion kehrt sofort zurück. Ändert sich der Verbindungsstatus, so wird die Callback-Funktion aus der DLL heraus gerufen, die den aktuellen Verbindungsstatus liefert.

Parameter:

***EthCanHandle_p**:* Instanz-Handle des CAN-Ethernet Gateways

Rückgabewerte: (*siehe Kapitel 7.3.3*)

ETHCAN_SUCCESSFUL
ETHCAN_ERR_ILLHANDLE
ETHCAN_ERR_ILLPARAM
ETHCAN_ERR_HWNOINIT
ETHCAN_ERR_HWDISCONNECT_FAILD
ETHCAN_ERR_SAL
ETHCAN_ERR_IFBTP
ETHCAN_ERR_RESOURCE

Hinweis:
Die Funktion ***EthCanDeinitHardware()*** ist sooft zu rufen, wie ein **fehlerfreier** Aufruf der Funktion ***EthCanInitHardware()*** erfolgt ist. Wurden die Funktionen im *„Nonblocked Mode“* aufgerufen, ist zusätzlich zu beachten, dass vor Beendigung der Applikation in **jedem** Fall sichergestellt werden muss, dass über die Callback-Funktion der Abbau der Verbindungen signalisiert wurde. Erst dann wird der Prozess-Thread in der DLL beendet!

Anwendungsbeispiel:

Die beiden Anwendungsbeispiele zeigen die Verwendung der Funktion mit blockierendem und nicht blockierendem Aufruf.

blockierender Aufruf

```
#define IP_ADDR ((192 << 0)+(168 << 8)+ (10 << 16)+(111 << 24))
#define IP_PORT (8234)

void main (void)
{
  DWORD dwRetcode;
  tEthCanHandle  EthCanHandle;
  tEthCanHwParam EthCanHwParam;

  EthCanHwParam.m_IpAdress = IP_ADDR;
  EthCanHwParam.m_wPort = IP_PORT;
  EthCanHwParam.m_UsedProtocol = kUseTCP;

  dwRetcode=EthCanInitHardware ( &EthCanHandle,
                                 &EthCanHwParam,
                                 NULL,
                                 NULL);

  if(dwRetcode == ETHCAN_SUCCESSFUL)
  {
      printf("\n*** Successfully initialized! ***\n");
  }
  else
  {
    goto Exit;
  }
                        .
                        .
  dwRetcode = EthCanDeinitHardware(EthCanHandle);
  if(dwRetcode == ETHCAN_SUCCESSFUL)
  {
      printf("\n*** Successfully closed! ***\n",
  }

Exit:
  return(dwRetcode);
}
```

nichtblockierender Aufruf

```
//Callback-Funktion für Verbindungsstatus
void PUBLIC EthCanConnectControlFct (tEthCanHandle EthCanHandle_p,
                                     DWORD dwConnectionState_p,
                                     void* pArg_p)
{
      switch(dwConnectionState_p)
      {
            case kEstablished:
                  EthCanInst_g[EthCanHandle_p].fConnected=TRUE;
                  break;

            case kConnecting:
            case kClosing:
            case kClosed:
                  EthCanInst_g[EthCanHandle_p].m_fConnected=FALSE;
                  break;
      }
}

void main (void)
{
  DWORD dwRetcode;
  tEthCanHandle EthCanHandle;

  //Ein CAN-Ethernet Gateway mit Callbackfunktion initialisieren
  dwRetcode = EthCanInitHardware ( &EthCanHandle,
                                   &EthCanHwParam,
                                   EthCanConnectControlFct,
                                   NULL);

  if(dwRetcode == ETHCAN_SUCCESSFUL)
  {
     printf("\n*** Successfully initialised! ***\n",
  }
                        .
                        .
  dwRetcode = EthCanDeinitHardware(EthCanHandle);
  if(dwRetcode == ETHCAN_SUCCESSFUL)
  {
     printf("\n*** Successfully closed! ***\n");
  }

  //Verbindungsabb. abwarten, signalisiert durch Callback-Funktion
  do
  {
    Sleep(10);

  }while(EthCanInst_g[EthCanHandle].m_fConnected);

  //Applikation beenden
  return(dwRetcode);
}
```

7.3.2.4 EthCanReadCanMsg

Syntax:

BYTE STDCALL EthCanReadCanMsg(
tEthCanHandle EthCanHandle_p,
tCANMsg pRcvCanMsg_p*
tCANTimestamp pRcvTime_p);*

Verwendbarkeit: HW_INIT

Bedeutung:

Diese Funktion liest eine CAN-Nachricht aus dem Empfangspuffer der DLL aus. Die CAN-Nachricht wird dabei aus dem Empfangspuffer gelöscht. Ist keine CAN-Nachricht im Empfangspuffer, liefert die Funktion den Rückgabewert ***ETHCAN_CANERR_QRCVEMPTY***.

Parameter:

EthCanHandle_p: Instanz-Handle des CAN-Ethernet Gateways

pRcvCanMsg_p: Adresse auf eine CAN-Nachrichtenstruktur. Diese Adresse darf nicht NULL sein!

Die Struktur der CAN-Nachricht besitzt folgenden Aufbau:

```
typedef struct
{
  DWORD m_dwID;           // CAN-Identifier
  BYTE  m_bMsgType;       // CAN-Frame-Format
  BYTE  m_bLen;           // CAN-Datenlänge
  BYTE  m_bData[8];       // CAN-Daten (max. 8 Byte)

}tCANMsg;
```

Abbildung 19: Aufbau CAN-Nachrichten-Struktur

Beim Nachrichtenformat der CAN-Nachricht wird zwischen verschieden Typen unterschieden. Zum Einen werden CAN-Nachrichten mit 11-Bit Identifier (Standard-CAN-Frame und CAN-

Nachrichten mit 29-Bit Identifier (Extended CAN-Frame) unterstützt. Dies gilt auch für sogenannte Remote-Frames (RTR-Frames) im Standard- bzw. Extended-CAN-Frameformat. Das Nachrichtenformat der CAN-Nachricht entspricht einer Bitkombination, die wie folgt definiert ist:

```
//Standard CAN-Frame
#define ETHCAN_MSGTYPE_STANDARD  0x00

//Remote Frame (11 Bit und 29 Bit CAN-ID)
#define ETHCAN_MSGTYPE_RTR       0x01

//Extended CAN Frame 2.0 B Frame (29 Bit CAN-ID)
#define ETHCAN_MSGTYPE_EXTENDED  0x02
```

Für eine CAN-Nachricht als RTR-Frame im Extended-Format sind die Werte entsprechend zu kombinieren und als Nachrichtenformat in der CAN-Nachrichtenstruktur einzutragen.

pRcvTime_p: Adresse auf eine TimeStamp-Struktur einer CAN-Nachricht. Diese Adresse darf nicht NULL sein.

Die TimeStamp-Struktur ist wie folgt definiert:

```
typedef struct
{
  DWORD  m_dwMilliSec;        //Millisekunden
  WORD   m_wMilliSec_Overflow;     //Millisekunden-Überlauf
  WORD   m_wMicroSec;         //Mikrosekunden

}tCANTimestamp;
```

Abbildung 20: Aufbau der CAN-TimeStamp-Struktur

Die Member-Variable ***m_dwMilliSec*** enthält die Anzahl der Millisekunden, die seit dem Systemstart der Hardware des CAN-Ethernet Gateways vergangen sind. Der Abstand zwischen zwei CAN-Nachrichten ergibt sich aus der Differenz der beiden Millisekundenwerte.

Hinweis:
Die Member-Variablen ***m_wMilliSec_Overflow*** und ***m_wMicroSec*** der Zeitstempel-Struktur werden zurzeit nicht verwendet und enthalten stets den Wert **0**.

Rückgabewert: (*siehe Kapitel 7.3.3 und 7.3.4*)

ETHCAN_SUCCESSFUL
ETHCAN_ERR_ILLPARAM
ETHCAN_ERR_ILLHANDLE
ETHCAN_ERR_HWNOINIT
ETHCAN_ERR_HWNOTCONNECTED
ETHCAN_CANERR_QRCVEMPTY
ETHCAN_CANERR_QOVERRUN

Anwendungsbeispiel:

```
tEthCanHandle EthCanHandle;
tCANMsg       RecvCanMsg;
tCANTimestamp RecvTime;
DWORD         dwRetcode;

//CAN-Nachricht lesen
dwRetcode = EthCanReadCanMsg(EthCanHandle,&RecvCanMsg,&RecvTime);
if(dwRetcode == ETHCAN_SUCCESSFUL)
{
      //CAN-Nachricht empfangen
}
else
if(dwRetcode & ETHCAN_CANERR_QRCVEMPTY)
{
      //keine CAN-Nachricht im Nachrichtenpuffer
}
else
{
      //Fehler beim Empfang der CAN-Nachricht
}
```

7.3.2.5 EthCanWriteCanMsg

Syntax:

DWORD STDCALL EthCanWriteCanMsg(
tEthCanHandle EthCanHandle_p,
tCANMsg pSendCanMsg_p,*
tCANTimestamp pSendTime_p);*

Verwendbarkeit: HW_INIT

Bedeutung:

Diese Funktion schreibt eine CAN-Nachricht in den Sendepuffer, der innerhalb der ***EthCan.Dll*** angelegt wird. Konnte die CAN-Nachricht nicht im Sendepuffer hinterlegt werden (z.B. Pufferüberlauf), kehrt die Funktion mit dem Fehlercode ***ETHCAN_CANERR_QXMTFULL*** zurück, das heißt, es hat ein Pufferüberlauf stattgefunden.

Parameter:

EthCanHandle_p: Instanz-Handle des CAN-Ethernet Gateways

pSendCanMsg_p: Adresse auf eine CAN-Nachrichtenstruktur. Diese Adresse darf nicht NULL sein!

pSendTime_p: Adresse auf eine Zeitstempelstruktur Diese Adresse darf nicht Null sein!

Der Übergabeparameter ***pSendCanMsg_p*** entspricht einer Adresse auf die Struktur einer CAN-Nachricht, wie sie bereits bei der Funktion ***EthCanReadCanMsg()*** *(siehe Kapitel 7.3.2.4)* erläutert wurde. Je nachdem welche CAN-Nachricht (29-Bit,11-Bit, RTR) gesendet werden soll, so ist das Nachrichtenformat *(siehe Kapitel 7.3.2.4)* entsprechend einzustellen.
Der Übergabeparameter ***pSendTime_p*** ist eine Adresse auf eine Zeitstempelstruktur. Für die Funktion besitzt dieser Parameter derzeit

keine Bedeutung, trotzdem darf die übergebene Adresse nicht NULL sein. Die Strukturelemente sind mit **0** zu initialisieren.

Rückgabewert: (*siehe Kapitel 7.3.3 und 7.3.4*)

ETHCAN_SUCCESSFUL
ETHCAN_ERR_ILLPARAM
ETHCAN_ERR_ILLHANDLE
ETHCAN_ERR_HWNOINIT
ETHCAN_ERR_HWNOTCONNECTED
ETHCAN_CANERR_QXMTFULL

Anwendungsbeispiel:

```
tEthCanHandle EthCanHandle;
tCANMsg       CanMsg;
tCANTimestamp SendTime;
DWORD         dwRetcode;

//Initialisierung eines Standard-RTR-Frames
CanMsg.m_dwId     = 0x180; //CAN-ID
CanMsg.m_bMsgType = ETHCAN_MSGTYPE_STANDARD & ETHCAN_MSGTYPE_RTR;
CanMsg.m_bLen     = 8;//8 Byte angeforderte Datenlänge

SendTime.m_dwMillis = 0;

//CAN-Nachricht senden
dwRetcode = EthCanWriteCanMsg(EthCanHandle,&CanMsg,&SendTime);
if(dwRetcode == ETHCAN_SUCCESSFUL)
{
      //CAN-Nachricht gesendet
}
else
if(dwRetcode & ETHCAN_CANERR_ QXMTFULL)
{
      //Überlauf des Sendepuffers
}
else
{
      //Fehler beim Senden der CAN-Nachricht
}
```

7.3.2.6 EthCanGetStatus

Syntax:

DWORD STDCALL EthCanGetStatus(
tEthCanHandle EthCanHandle_p,
tStatus pStatus_p);*

Verwendbarkeit: HW_INIT

Bedeutung:

Die Funktion liefert den Fehlerstatus des CAN-Treibers sowie den Verbindungsstatus der Ethernet-Verbindung des CAN-Ethernet-Gateways zurück. Tritt auf dem CAN-Ethernet Gateway ein CAN-Fehler auf (z.B. Sende- oder Empfangspufferüberlauf), so wird dieser Status über Ethernet übertragen und kann mit Hilfe dieser Funktion abgerufen werden. Zusätzlich zum CAN-Status wird auch der aktuelle Verbindungsstatus der Ethernet-Verbindung zwischen PC und dem CAN-Ethernet Gateway zurückgeliefert.
Diese Funktion muss in gewissen Zeitabständen gerufen werden, um auf eventuelle CAN-Fehler reagieren zu können.

Parameter:

***EthCanHandle_p**:* Instanz-Handle des CAN-Ethernet Gateways

***pStatus_p**:* Adresse auf eine Status-Struktur
Diese Adresse darf nicht NULL sein.

Diese Statusstruktur ist wie folgt definiert:

```
typedef struct
{
  WORD  m_wCanStatus;         // aktueller CAN-Status
  WORD  m_wConnectionStatus;  // aktueller Verbindungsstatus

} tStatus;
```

Abbildung 21: Aufbau der CAN-Status-Struktur

Rückgabewerte: (*siehe Kapitel 7.3.3*)

ETHCAN_SUCCESSFUL
ETHCAN_ERR_ILLHANDLE
ETHCAN_ERR_ILLPARAM
ETHCAN_ERR_HWNOINIT
ETHCAN_ERR_HWNOTCONNECTED

Anwendungsbeispiel:

```
tEthCanHandle EthCanHandle;
tStatus       Status;
DWORD         dwRetcode;

//CAN-Status lesen
dwRetcode = EthCanGetStatus(EthCanHandle,&Status);
if(dwRetcode == ETHCAN_SUCCESSFUL)
{
      if(Status.m_wCanStatus & ETHCAN_CANERR_OVERRUN)
      {
            //Overrun aufgetreten
      }
}
else
{
      //Fehler beim Auslesen des CAN-Status
}
```

7.3.2.7 EthCanGetConnectionState

Syntax:

DWORD PUBLIC EthCanGetConnectionState(
tEthCanHandle EthCanHandle_p,
tConnectionState pState_p);*

Verwendbarkeit: HW_INIT

Parameter:

***EthCanHandle_p**:* Instanz-Handle des CAN-Ethernet Gateways

pState_p: Adresse auf Verbindungsstatus-Variable
Diese Adresse darf nicht NULL sein!

Bedeutung:

Diese Funktion liefert den aktuellen Verbindungsstatus des CAN-Ethernet Gateway. Wurde bei der Initialisierung des Gateways keine Callback-Funktion angegeben, die bei Änderung des Verbindungsstatus gerufen wird, so kann mittels dieser Funktion im Polling-Verfahren der Verbindungsstatus abgerufen werden. Sinnvoll ist die Verwendung dieser Funktion, wenn die Initialisierungs- und Deinitialisierungsroutine im sogenannten ***„Blocked Mode“*** aufgerufen wurden.
Der Parameter ***pState_p*** liefert nach dem Aufruf der Funktion den aktuellen Verbindungsstatus und kann die Werte annehmen, wie sie bereits in dem *Abbildung 18* erläutert wurden.

Rückgabewerte: (*siehe Kapitel 7.3.3*)

ETHCAN_SUCCESSFUL
ETHCAN_ERR_ILLHANDLE
ETHCAN_ERR_ILLPARAM
ETHCAN_ERR_HWNOINIT

Anwendungsbeispiel:

```
tEthCanHandle    EthCanHandle;
tConnectionState ConnectionState;
DWORD            dwRetcode;

//Verbindungsstatus lesen
dwRetcode = EthCanGetStatus(EthCanHandle,&ConnectionState);
if(dwRetcode == ETHCAN_SUCCESSFUL)
{
      if(ConnectionState == kConnecting)
      {
            //Auszuführender Code
      }
      if(ConnectionState == kEstablished)
      {
            //Auszuführender Code
      }
                  .
                  .
                  .
}
else
{
      //Fehler beim Lesen des Verbindungsstatus
}
```

7.3.2.8 EthCanResetCan

Syntax:

DWORD PUBLIC EthCanResetCan(
tEthCanHandle,
dwResetCode_p)

Verwendbarkeit: HW_INIT

Parameter:

***EthCanHandle_p**:* Instanz-Handle des CAN-Ethernet Gateways

dwResetCode_p: Reset-Code für CAN

Bedeutung:

Diese Funktion dient zum gezielten Zurücksetzen der CAN-Kommunikation des CAN-Ethernet Gateways und der DLL bei Auftreten von CAN-Fehler infolge von Pufferüberläufen oder Störungen des CAN-Busses. Über dem Parameter ***dwResetCode_p*** wird festgelegt, ob nur die Sende- und Empfangspuffer in der DLL oder auch die Sende- und Empfangspuffer des CAN-Ethernet-Gateways und dessen CAN-Interface (CAN-Controller) zurückgesetzt werden sollen.
Folgende Parameter-Werte können für den Reset-Code übergeben werden:

//Löschen des Sendbuffers für CAN-Nachrichten in der DLL
#define RESET_TRANSMIT_QUEUE 0x00

//Löschen des Empfangspuffers für CAN-Nachrichten in der DLL
#define RESET_RECEIVE_QUEUE 0x01

//Löschen des Sende- und Empfangspuffers für CAN-Nachrichten in der DLL
#define RESET_ALL_QUEUES 0x02

```
//Löschen des Sende- und Empfangspuffers für CAN-Nachrichten auf
//dem Gateway und Zurücksetzen des CAN-Controllers
#define RESET_CAN_CONTROLLER          0x04
```

Diese Konstanten können bitweise kombiniert werden, so dass je nach Anwendungsfall ein Zurücksetzen bestimmter oder aller CAN-Komponenten möglich ist.

Hinweis:
Während des Zurücksetzens des CAN-Interfaces beziehungsweise dem Löschen der Puffer können keine CAN-Nachrichten gesendet und empfangen werden!

Rückgabewerte: (*siehe Kapitel 7.3.3*)

ETHCAN_SUCCESSFUL
ETHCAN_ERR_ILLHANDLE
ETHCAN_ERR_ILLPARAM
ETHCAN_ERR_HWNOINIT
ETHCAN_ERR_HWNOTCONNECTED

Anwendungsbeispiel:

```
tEthCanHandle EthCanHandle;
DWORD         dwResetCode;
DWORD         dwRetcode;

//Reset aller Buffer und Reset des CAN-Interfaces
//des CAN-Ethernet Gateways
dwResetCode = (RESET_ALL_QUEUES & RESET_CAN_CONTROLLER);

//CAN-Interface zurücksetzen
dwRetcode = EthCanResetCan(EthCanHandle,dwResetCode);
if(dwRetcode == ETHCAN_SUCCESSFUL)
{
      //CAN-Interface wurde erfolgreich zurückgesetzt
}
else
{
      //Fehler beim Zurücksetzen des CAN-Interfaces
}
```

7.3.3 Beschreibung der Fehlercodes

Die Funktionen der EthCan.Dll liefern einen Fehlercode in Form eines *DWORD* zurück. Jeder Rückgabewert entspricht genau einem Fehler. In der folgenden Tabelle sind alle Fehlercodes und deren numerischer Wert abgebildet.

Fehlercodes	Numerischer Wert
ETHCAN_SUCCESSFUL	0x0
ETHCAN_ERR_ILLPARAM	0x1
ETHCAN_ERR_ILLPARAMVAL	0x2
ETHCAN_ERR_ILLHANDLE	0x3
ETHCAN_ERR_HWNOINIT	0x4
ETHCAN_ERR_HWINUSE	0x5
ETHCAN_ERR_HWNOTCONNECTED	0x6
ETHCAN_ERR_HWCONNECT_FAILED	0x7
ETHCAN_ERR_HWDISCONNECT_FAILED	0x8
ETHCAN_ERR_MAXMODULES	0x9
ETHCAN_ERR_SAL	0xA
ETHCAN_ERR_IFBTP	0xB
ETHCAN_ERR_RESOURCE	0xC

Tabelle 11: Fehlercodes Interfacefunktionen EthCan.Dll

ETHCAN_SUCCESSFUL

Die Funktion wurde fehlerfrei ausgeführt.

ETHCAN_ERR_ILLPARAM

Der aufgerufenen Funktion wurde ein illegaler Parameter übergeben. Häufigste Ursache ist zum Beispiel die Übergabe eines NULL-Pointers.

ETHCAN_ERR_ILLPARAMVAL

Der aufgerufenen Funktion wurde ein ungültiger Parameter-Wert übergeben.

ETHCAN_ERR_ILLHANDLE

Der aufgerufenen Funktion wurde ein ungültiges Instanz-Handle übergeben. Eine Ursache ist die Übergabe eines Instanz-Handle mit einer ungültigen Instanznummer, zum Beispiel **0**. Zudem unterstützt der Treiber nur maximal 5 Instanzen des CAN-Ethernet Gateways, so dass die Übergabe eines Instanz-Handle größer **5** ebenfalls zu diesem Fehler führt.

ETHCAN_ERR_HWNOINIT

Die Funktion wurde mit einem Instanz-Handle aufgerufen, für das die zugehörige Initialisierungsfunktion der Hardware noch nicht gerufen wurde. Es ist deshalb die Funktion ***EthCanInitHardware()*** aufzurufen, die nach erfolgreichem Anschluss ein gültiges Instanz-Handle zurückliefert.

ETHCAN_ERR_HWINUSE

Die Funktion ***EthCanInitHardware()*** wurde mit einem Instanz-Handle aufgerufen, für das die Initialisierungsroutine bereits erfolgreich gerufen wurde. Um eine erneute Initialisierung mit eventuell geänderten Parameterwerten auszuführen, ist in jedem Fall die Deinitialisierungsfunktion ***EthCanDeinitHardware()*** zu rufen, bevor eine erneute Initialisierung erfolgen kann.

ETHCAN_ERR_HWNOTCONNECTED

Die Funktion wurde aufgerufen, bei der zum Zeitpunkt des Aufrufs keine Verbindung des CAN-Ethernet Gateways mit dem PC bestand. Eine mögliche Ursache kann ein Verbindungsabbruch infolge des Verlustes der physischen Netzwerkverbindung(Ethernet-Kabel wurde getrennt) sein.

Wurde bei der Initialisierung eine Callback-Funktion angegeben, so kann an dieser Stelle auf das Schließen der Verbindung reagiert werden. Zum Beispiel sind die Sende- und Empfangsfunktionen für CAN-Nachrichten erst wieder zu rufen, wenn sich der Verbindungsstatus wieder im Zustand ***kEstablished*** befindet.
Ist keine Callback-Funktion definiert, so kann über die Funktion ***EthCanGetConnectionState()*** der Verbindungsstatus gepollt und nach einem erfolgreichen ***„Reconnect“*** der Aufruf der Funktion wiederholt werden.

ETHCAN_ERR_HWCONNECT_FAILED

Dieser Fehlercode wird nur von der Funktion ***EthCanInitHardware()*** zurückgegeben, wenn diese im blockierenden Modus das heißt ohne Angabe einer Callback-Funktion, aufgerufen wird. Die Ursache dafür ist, dass innerhalb der Timeout-Zeit, die der Parameter-Struktur bei der Initialisierung übergeben wurde, keine Verbindung zur angegebenen Remote-Adresse aufgebaut werden konnte. Der Default-Timeout ist im Header-File ***EthCan32.h*** auf **5s** gesetzt. Sollte bei der Initialisierung ein eigener Timeout-Wert übergeben worden sein, so ist zu prüfen, ob der Timeout für einen erfolgreichen Verbindungsaufbau ausreichend ist (abhängig von Entfernung und Netztopologie).
Des Weiteren ist zu prüfen, ob die angegebenen Parameter wie IP-Adresse und Port-Nummer richtig sind und die Remote-Adresse über die Ethernet-Verbindung erreichbar ist.

ETHCAN_ERR_HWDISCONNECT_FAILED

Dieser Fehlercode wird nur von der Funktion ***EthCanDeinitHardware*** zurückgegeben, wenn diese im blockierenden Modus aufgerufen wird. Die Ursache dafür ist, dass innerhalb der Timeout-Zeit, die der Parameter-Struktur bei der Initialisierung übergeben wurde, die Verbindung zur angegebenen Remote-Adresse nicht abgebaut werden konnte. Der Default-Timeout ist im Header-File ***EthCan32.h*** auf **5s** gesetzt. Sollte bei der Initialisierung ein eigener Timeout-Wert übergeben worden sein, so ist zu prüfen, ob der Timeout für einen

erfolgreichen Verbindungsabbau ausreichend ist (abhängig von Entfernung und Netztopologie).

ETHCAN_ERR_MAXMODULES

Die Anzahl der maximal durch die DLL unterstützten CAN-Ethernet Gateways ist erreicht. Ein Initialisieren einer weiteren Instanz ist nicht möglich. Deinitialisieren Sie eventuell nicht mehr benötigte Instanzen und rufen Sie dann erneut die Init-Funktion auf.

ETHCAN_ERR_SAL

Während der Initialisierung bzw. Deinitialisierung der SAL-Schicht (Stack-Abstraction-Layer) für TCP beziehungsweise UDP ist ein Fehler aufgetreten. Mögliche Fehlerursachen können sein:

- Das Anlegen beziehungsweise Schließen der Windows-Socket-Schnittstelle konnte auf Grund fehlender Ressourcen oder einer nicht unterstützten Version der Windows-Socket nicht durchgeführt werden.
- Der Aufruf der verwendeten WIN32-Funktionen für die Windows-Socket wie Connect(), Bind(), Accept(), Send() und Recv() haben auf Grund ungültiger Parameter einen Fehler zurückgeliefert.

ETHCAN_ERR_IFBTP

Während der Initialisierung beziehungsweise Deinitialisierung des BTP-Interfaces (Block Transfer Protocol) für UDP beziehungsweise TCP ist ein Fehler aufgetreten.

ETHCAN_ERR_RESOURCE

Eine Ressource konnte nicht erzeugt werden. Unter dem Begriff Ressourcen sind Speicheranforderungen, Handles, oder Threads zusammengefasst, die von Windows vergeben werden.

7.3.4 Beschreibung der CAN-Fehlercodes

Der CAN-Fehlercode, der durch die Funktionen ***EthCanReadCanMsg()***, ***EthCanWriteCanMsg()*** und ***EthCanGetStatus()*** zurückgeliefert wird, entspricht einer Bitkombination aus den in der folgenden Tabelle dargestellten Fehlercodes. Dabei können gleichzeitig mehrere Fehlerzustände angezeigt sein.

CAN-Error-Code	Numerischer Wert
ETHCAN_CANERR_OK	0x0000
ETHCAN_CANERR_XMTFULL	0x0001
ETHCAN_CANERR_OVERRUN	0x0002
ETHCAN_CANERR_BUSLIGHT	0x0004
ETHCAN_CANERR_BUSHEAVY	0x0008
ETHCAN_CANERR_BUSOFF	0x0010
ETHCAN_CANERR_QRCVEMPTY	0x0020
ETHCAN_CANERR_QOVERRUN	0x0040
ETHCAN_CANERR_QXMTFULL	0x0080
ETHCAN_CANERR_REGTEST	0x0100
ETHCAN_CANERR_MEMTEST	0x0200

Tabelle 12: CAN-Fehlercodes

ETHCAN_CANERR_OK

Kein CAN-Fehler aufgetreten

ETHCAN_CANERR_XMTFULL

Der Sendepuffer im CAN-Controller des CAN-Ethernet Gateways hat die maximale Anzahl an CAN-Nachrichten erreicht.

ETHCAN_CANERR_OVERRUN

Der Empfangspuffer im CAN-Controller des CAN-Ethernet Gateways hat die maximale Anzahl an CAN-Nachrichten erreicht.

ETHCAN_CANERR_BUSLIGHT

Der Fehlerzähler im CAN-Controller hat das Warning-Limit 1 erreicht. Siehe dazu das Manual zum CAN-Controller SJA 1000.

ETHCAN_CANERR_BUSHEAVY

Der Fehlerzähler im CAN-Controller hat das Warning-Limit 2 erreicht. Siehe dazu das Manual zum CAN-Controller SJA 1000.

ETHCAN_CANERR_BUSOFF

Der CAN-Controller ist auf Grund der Fehlerzähler in den Zustand BUS_OFF gegangen, um eine weitere Störung des CAN-Busses zu vermeiden.

ETHCAN_CANERR_QRCVEMPTY

Die Empfangsqueue für CAN-Nachrichten innerhalb der DLL enthält keine CAN-Nachrichten, das heißt, alle CAN-Nachrichten wurden ausgelesen beziehungsweise es wurden keine neuen CAN-Nachrichten empfangen.

ETHCAN_CANERR_QOVERRUN

Die Empfangsqueue für CAN-Nachrichten ist übergelaufen. Dabei können CAN-Nachrichten verloren gegangen sein.

ETHCAN_CANERR_QXMTFULL

Die Sendequeue für CAN-Nachrichten in der DLL ist übergelaufen. Dabei können CAN-Nachrichten verloren gegangen sein.

ETHCAN_CANERR_REGTEST

Der Registertest des SJA1000 ist fehlgeschlagen. Sehen Sie dazu das Manual zum CAN-Controller SJA 1000.

ETHCAN_CANERR_MEMTEST

Der Speichertest des SJA1000 ist fehlgeschlagen. Sehen Sie dazu das Manual zum CAN-Controller SJA 1000.

7.3.5 Anwendung der DLL-Funktionen

7.3.5.1 Demo-Projekt

Wie im *Kapitel 7.2* beschrieben, wird bei der Installation ein Demo-Projekt im Installationspfad angelegt. Dieses Projekt beinhaltet eine „C“-Quellcodedatei ***Demo.c*** sowie die zugehörige Header-Datei *Demo.h* und zeigt die Anwendung der DLL-Interface-Funktionen.
Dabei wird die ***EthCan.Dll*** zur Laufzeit über die Funktion ***LoadLibrary()*** dynamisch geladen und die Funktionspointer mit der Funktion ***GetProcAddress()*** ermittelt.

Das Demo-Programm basiert auf einer WIN32-Konsolenapplikation und ist auf die Unterstützung einer Instanz des CAN-Ethernet Gateways beschränkt. Nach dem Starten des Programms und dem Laden der ***EthCan.Dll*** wird eine Verbindung zu einem durch die IP-Adresse und Port-Nummer adressierten CAN-Ethernet Gateway aufgebaut. Konnte die Verbindung erfolgreich hergestellt werden, so wird zyklisch eine CAN-Nachricht mit der CAN-ID 0x180 gesendet.

Für die Überprüfung des Empfangs der CAN-Nachricht über Ethernet und dem sich anschließenden Senden der CAN-Nachricht auf den CAN-Bus des Gateways ist ein Analysetool (z.B. PCAN Explorer™ der Fa. Peak) zu nutzen, dass am CAN-Bus des CAN-Ethernet Gateways angeschlossen ist.

Das Senden von CAN-Nachrichten auf dem CAN-Bus des CAN-Ethernet Gateways kann ebenfalls von einem Analysetool oder einer weiteren angeschlossenen CAN-Hardware erfolgen. Der Empfang der CAN-Nachricht vom CAN-Ethernet Gateway auf dem PC wird im Konsolen-Fenster der Demo-Applikation mit Ausgabe der CAN-ID, der CAN-Nachrichtenlänge sowie der enthaltenen Daten quittiert.

7.3.5.2 Starten des Demo-Programms

Das Demo-Programm benötigt für den Aufruf verschiedene Kommandozeilenparameter, wie die IP-Adresse und die Port-Nummer des Gateways, sowie das zu verwendende Übertragungsprotokoll (TCP oder UDP). Für das Starten des Demo-Programms stehen zwei verschiedene Möglichkeiten zur Verfügung:

1. Öffnen Sie eine Kommando-Shell und wechseln Sie in das Verzeichnis der ausführbaren Datei ***EthCanDemo.exe*****,** das Sie bei der Installation angegeben haben.
   Zum Starten des Demo-Programms geben Sie am Eingabeprompt den Namen des ausführbaren Programms, die IP-Adresse, die Port-Nummer und das gewünschte Übertragungsprotokoll an. Im Folgenden sind zwei Beispiele für den Aufruf dargestellt:

   Beispiel 1:

   ...\Release\EthCanDemo.exe 192.168.010.111 8234 TCP

   Beispiel 2:

   ...\Release\EthCanDemo.exe 192.168.010.111 8234 UDP

   Die einzelnen Parameter sind durch Leerzeichen voneinander zu trennen. Durch Bestätigung mit der Eingabetaste wird die Demoapplikation gestartet.

2. Die zweite Möglichkeit besteht im Anlegen einer Verknüpfung auf dem Desktop, wie in der dargestellt ist. Gehen sie dazu auf den Desktop und legen eine neue Verknüpfung an. Anschließend richten Sie den Zielpfad auf das Arbeitsverzeichnis aus, in dem das ausführbare Programm liegt.

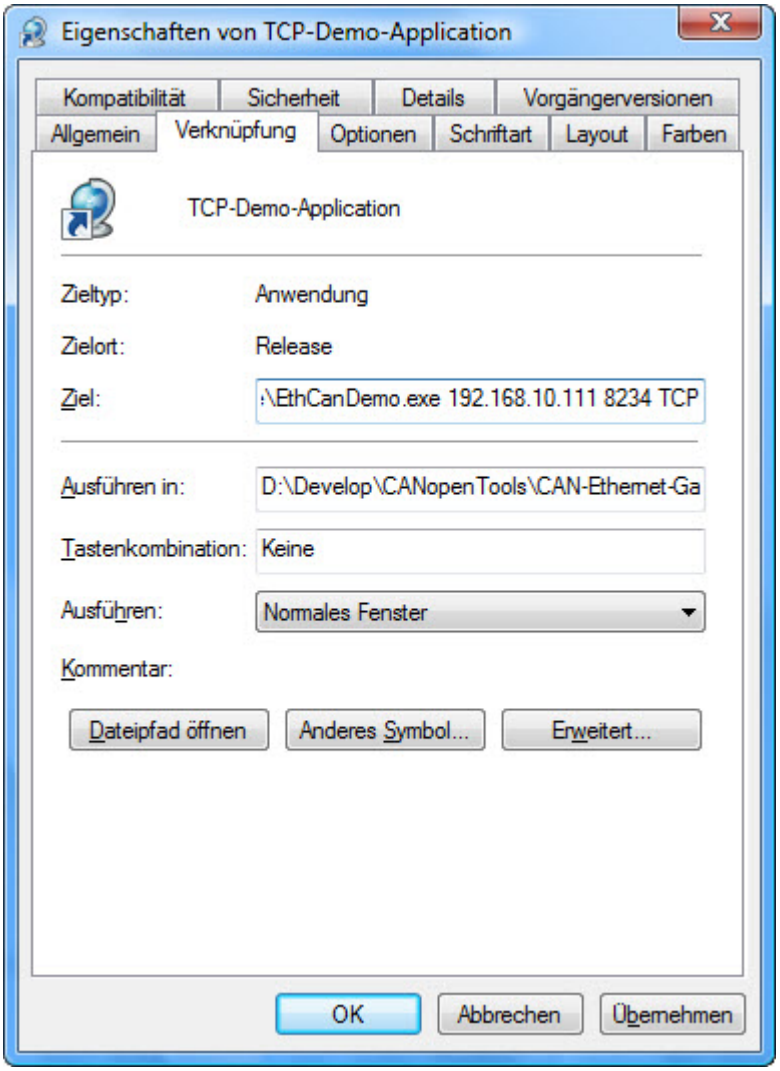

Abbildung 22: Desktop-Verknüpfung für Demo-Programm

Im Anschluss an den Zielpfad müssen durch Leerzeichen getrennt die IP-Adresse, die Port-Nummer und das Übertragungsprotokoll angegeben werden.

8 Beschreibung des Firmwareupdates

Die Modemschnittstelle des CAN-Ethernet Gateways dient gleichzeitig als Schnittstelle für die Durchführung eins Updates der Gerätefirmware.
Die Gerätefirmware des Gateways wird stets in den internen Programmspeicher (Flash) des verwendeten Mikrocontrollers geladen. Dafür steht das ***MemTool*** der Firma Infineon zur Verfügung, welches auf der beigelegten CD verfügbar ist beziehungsweise kostenlos von der Homepage ***www.infineon.com*** heruntergeladen werden kann.

Dieses Tool ist auf einem PC mit einem Windows-Betriebssystem zu installieren. Des Weiteren wird für die Verbindung zwischen der seriellen Schnittstelle des PCs und der Modemschnittstelle des CAN-Ethernet Gateways ein Nullmodemkabel benötigt.

8.1 Vorbereitungen

Folgende Schritte sind zur Vorbereitung des Firmwareupdates durchzuführen:

1. Schließen Sie die Spannungsversorgung an das CAN-Ethernet Gateway an.

2. Verbinden Sie die Modem-Schnittstelle des Gateways mit Hilfe eines Nullmodemkabels mit der seriellen Schnittstelle des PCs, auf dem das Tool zum Firmwaredownload installiert wurde.

3. Setzen Sie das Modul in den Bootstrap-Mode . Dazu ist der ***Boot***-Schalter des Moduls in die Stellung ***„On“*** zu schalten. Betätigen Sie anschließend den ***Reset***-Schalter kurz in die Stellung ***„On“*** und wieder zurück in die Stellung ***„Off“***

8.2 Firmwaredownload

Für den Firmwaredownload ist das MemTool zu starten. In der *Abbildung 23* ist der Startdialog dargestellt.

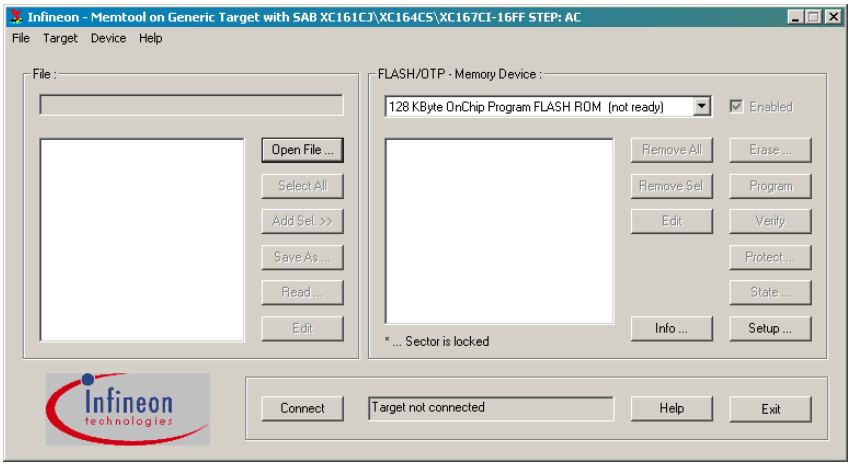

Abbildung 23: Startdialog MemTool

Über den Menüpunkt ***„Target/Change“*** ist der entsprechende Controllertyp auszuwählen. Wählen Sie hier den Controller mit der folgenden Bezeichnung aus:

XC16Board with XC161CJ/164CS/167CI-16F (AC step or newer)

Im Anschluss ist der Kommunikationsport (serielle Schnittstelle) zu konfigurieren. Durch Auswahl von ***„Target/Setup Communication Port...“*** öffnet sich der Konfigurationsdialog, in dem Sie die Einstellungen für die serielle Schnittstelle vornehmen können.

Nachdem die Konfiguration abgeschlossen wurde, ist die Verbindung zum Modul durch Auswahl des ***„Connect“***-Buttons herzustellen. Nach erfolgreicher Verbindungsaufnahme werden im rechten Fenster des Tools die Speichersektoren wie *Abbildung 24* angezeigt.

Diese Speichersektoren sind mittels des ***„Erase“***-Buttons zu löschen. Wählen Sie anschließend über den Button **„Open File...“** das Hex-File für den Download aus. Es trägt die Bezeichnung ***E4050.H86*** für

das CAN-Ethernet-Gateway mit 1 CAN-Kanal und ***E4079.H86*** für das CAN-Ethernet-Gateway mit 2 CAN-Kanälen.

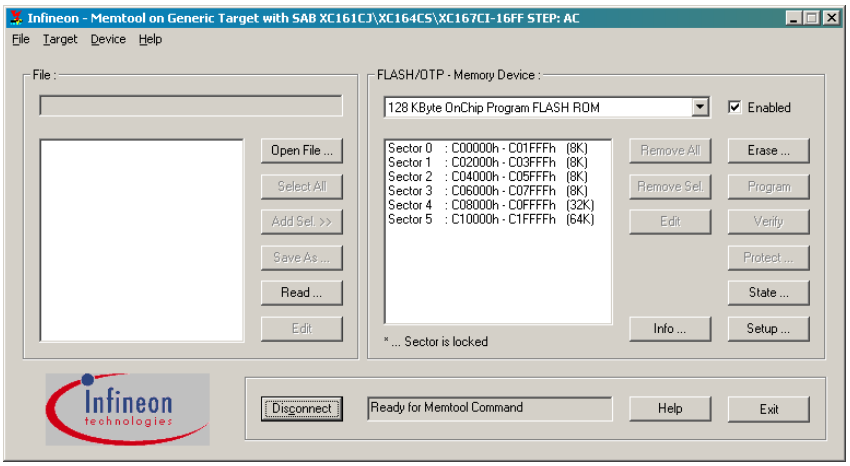

Abbildung 24: Ansicht der Speichersektoren

Selektieren Sie alle Speicherbereiche im linken Fenster des Tools mit Hilfe von ***„Select All“*** und ordnen diese durch ***„Add Sel >>“*** zu den Sektoren im rechten Fenster, wie in *Abbildung 25* dargestellt ist, zu.

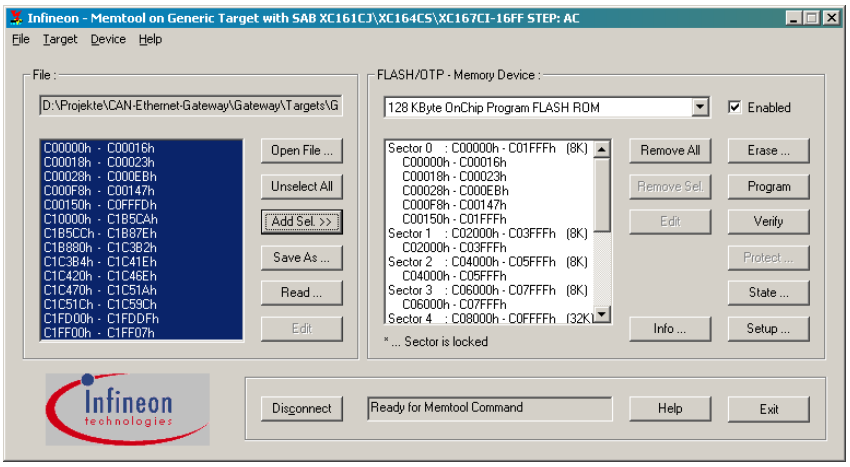

Abbildung 25: Speicherbereiche und Sektorenzuordnung

Über den Button ***„Program“*** wird die Firmware auf das Gateway heruntergeladen.

Nach Abschluss des Firmware-Downloads ist die Verbindung zwischen PC und Gateway durch den Button ***„Disconnect“*** zu trennen und ein Hardware-Reset des Gateways durchzuführen.

Der Boot-Schalter des Gateways ist nach erfolgtem Firmware-Download in die Stellung ***„Off“*** zu setzen. Anschließend muss das Gateway über den Reset-Schalter beziehungsweise durch Power-On-Reset zurückgesetzt werden.

Beachte:
Beim Download einer neuen Firmwareversion mit geänderter Versionsnummer werden die Einstellungen des Resource-Files, welches im EEPROM abgelegt ist, verworfen. Beim Neustart der Firmware ist demzufolge die IP-Adresse auf 192.168.10.111 des Standard-Resource-Files gültig.
Wurden die Einstellungen des Standard-Resource-Files durch den Download eines eigens erstellten Resource-Files geändert, so muss dieses nach einem Firmware-Download erneut auf das Gateway heruntergeladen werden.

Index

Dokument:	CAN-Ethernet-Gateway
Dokumentnummer:	L-1032d_10, Auflage Juli 2010

Wie würden Sie dieses Handbuch verbessern?

Haben Sie in diesem Handbuch Fehler entdeckt? Seite

Eingesandt von:

Kundennummer: _____

Name: _____

Firma: _____

Adresse: _____

Einsenden an:

SYS TEC electronic GmbH
August-Bebel-Str. 29
D-07973 Greiz
GERMANY
Fax : +49 (0) 36 61 / 62 79 99

Veröffentlicht von

SYS TEC ELECTRONIC

Best.-Nr. L-1032d_10
Printed in Germany