CAN-Ethernet-Gateway V2

System Manual

Auflage Dezember 2014

SYS TEC electronic GmbH
Am Windrad 2 · 08468 Heinsdorfergrund · Deutschland
Telefon: +49 3765 38600-0 · Fax: +49 3765 38600-4100
Web: http://www.systec-electronic.com · Mail: info@systec-electronic.com

Systemhaus für verteilte Automatisierung

Status / Changes

Status: freigegeben

Datum/ Version	Abschnitt	Änderung	Editor
16.05.2011		Erstellung	Stein
16.04.2012	1.2.2	Liste der CAN-Software angepasst	Stein
	3	Beschreibung angepasst	Stein
25.04.2012	alle	Formatierung des Dokumentes angepasst	Glau
09.07.2012	9	ASCII-Protokoll hinzugefügt	Stein
16.07.2012	6	Konfigurationsopiotnen ergänzt	Stein
08.05.2014	9	ASCII-Protokoll erweitert	Stein
03.06.2014	9	ASCII-Format für Firmware 1.2.2 beschrieben	Stein
	6.4.5	Hinweis für BTP/TCP und EthCan.dll hinzugefügt	Stein
12.06.2014	9	Link zu ASCII parser demo hinzugefügt	Stein
17.09.2014	6.5, 6.6	Hinweis zu INI-Format: Der Filtertrenner ist Kommentarzeichen, Beispiele angepasst	Stein
24.11.2014	6.4.1	Option UserBaud hinzugefügt	Stein
15.12.2014	6.4.7	Option Timestamp hinzugefügt	Stein
	9.2.1	Beschreibung Zeitstempel hinzugefügt	Stein

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.

Kontakt	
Adresse:	SYS TEC electronic GmbH Am Windrad 2 08468 Heinsdorfergrund GERMANY
Angebots-Hotline:	+49 3765 / 38600-2110 info@systec-electronic.com
Technische Hotline:	+49 3765 / 38600-2140 support@systec-electronic.com
Fax:	+49 3765 / 38600-4100
Website:	http://www.systec-electronic.com

1 Einleitung

1.1 Grundlagen

Das CAN-Ethernet-Gateway der SYS TEC electronic GmbH war bisher das Standardprodukt für die Verbindung von CAN-Netzen. Da es aber an seine Kapazitätsgrenzen stieß, wurde das CAN-Ethernet-Gateway V2 entwickelt, welches die bisherigen Begrenzungen nicht mehr besitzt.

Internetkommunikation über TCP/IP verbreitet sich auch im industriellen Bereich immer weiter. SYS TEC electronic GmbH stellt mit dem CAN-Ethernet-Gateway V2 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 V2 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 V2 mit verschiedenen CAN-Protokollen (z.B. CANopen, SDS, J1939, DeviceNet oder firmenspezifische Protokolle usw.) eingesetzt werden.

Das CAN-Ethernet-Gateway V2 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 V2 erzeugt und zusammen mit den Daten übertragen werden.

Das CAN-Ethernet-Gateway V2 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 V2 so an das spezielle Einsatzgebiet anpassen.

Für die Kommunikation zwischen den CAN-Ethernet-Gateways V2 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 V2 erkennt aufgetretene Fehler und sendet CAN-Nachrichten (Fehlernachrichten), die den Fehlergrund enthalten. Der zu verwendende CAN-Identifier der Fehlernachricht kann konfiguriert werden *(siehe Abschnitt 7.2)*.

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 V2. CAN-Nachrichten werden transparent zwischen den CAN-Ethernet-Gateways V2 übertragen.
Die Firmware des CAN-Ethernet-Gateways V2 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 V2 sind in den folgenden Abbildungen *Abbildung 1* und *Abbildung 2* dargestellt:

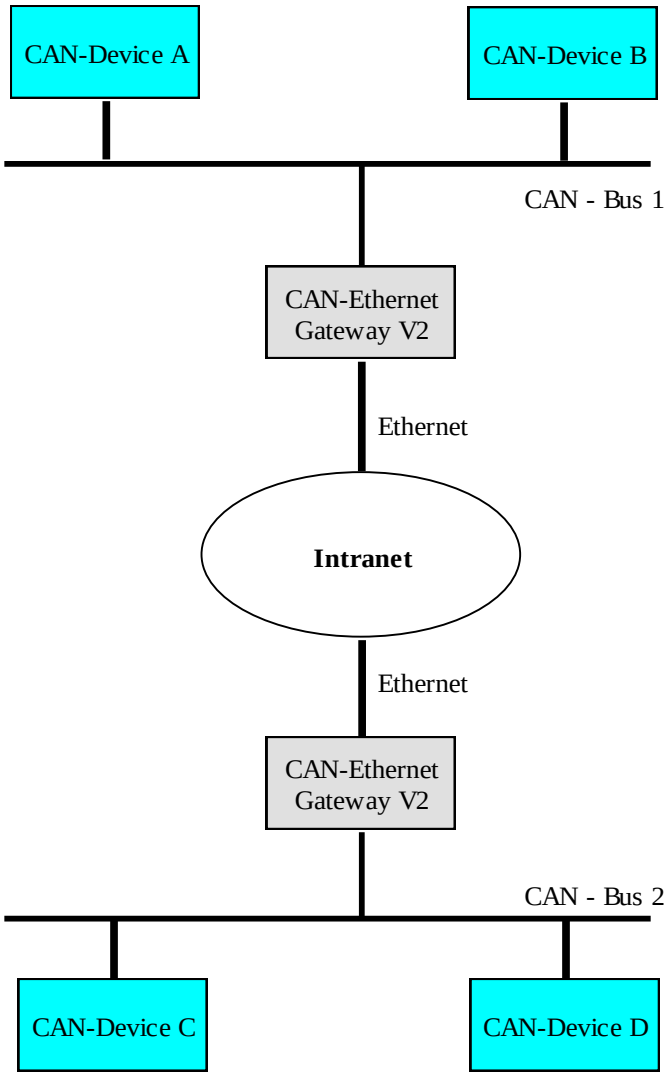

Abbildung 1: Transparente Verbindung zweier CAN

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 V2 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-Tools von Port, weitere produkte auf Anfrage).

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.

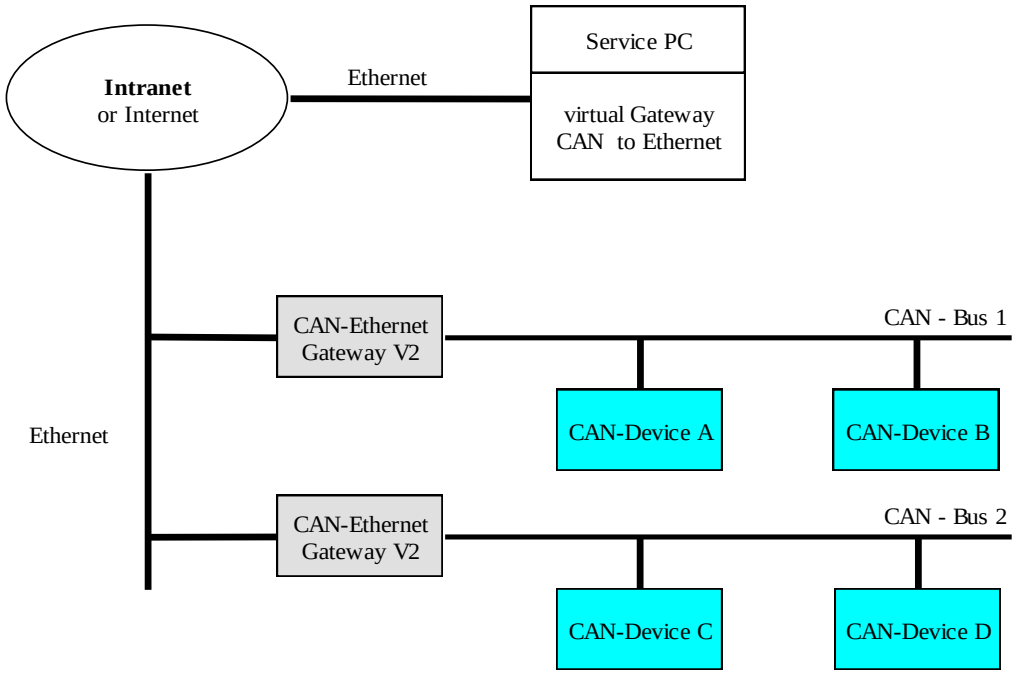

Abbildung 2: Ferndiagnose von CAN-Netzen mittels PC

2 Lieferumfang

Zum Lieferumfang des CAN-Ethernet-Gateways V2 gehören:

- 3004010 CAN-Ethernet-Gateway V2 1xCAN Grundausbau
  im Gehäuse für Tragschienenmontage, inkl. 2-pol. abziehbarem
  Schraubklemmverbinder und einem 2x5 pol. Zug-Federanschluss

oder

- 3004011 CAN-Ethernet-Gateway V2 2xCAN Grundausbau
  im Gehäuse für Tragschienenmontage, inkl. 2-pol. abziehbarem
  Schraubklemmverbindern und zwei 2x5 pol. Zug-Federanschluss

oder

- 3004013 CAN-Ethernet-Gateway V2 1xCAN Vollausbau
  im Gehäuse für Tragschienenmontage, inkl. 2-pol. abziehbarem
  Schraubklemmverbinder und einem 2x5 pol. Zug-Federanschluss

oder

- 3004014 CAN-Ethernet-Gateway V2 2xCAN Vollausbau
  im Gehäuse für Tragschienenmontage, inkl. inkl. 2-pol. abziehbarem
  Schraubklemmverbinder und zwei 2x5 pol. Zug-Federanschluss

- WK806 USB-Anschlusskabel Stecker Serie - Stecker Serie B 1,8m
- L-1314 Produktbeilageblatt

3 Technische Daten

Das CAN-Ethernet-Gateway V2 besitzt folgende technische Daten und Funktionalitäten:

- Betriebssystem: Linux
- Überwachen und Steuern entfernter CAN-Netzwerke über das Internet
- Kopplung zweier CAN-Netzwerke
- Gateway konfigurierbar über Telnet, FTP (Fernwartung) oder serielle Schnittstelle über USB-Buchse
- Dateisystem für Konfigurationsdaten
- flexible Konfiguration durch Einsatz mehrerer Interfaces (Abschnitt *5.2*)
- umfangreiche Filtermechanismen für CAN-Nachrichten
- Generierung eines Zeitstempels (Timestamp) für CAN-Nachrichten
- Anbindung zu Windows-Anwendungsprogrammen für CAN und CANopen
- Leuchtdioden (LED) zur Visualisierung des Zustands des Gateways, 8St. in Grundausbau, 12St. in Vollausbau
- Generierung von CAN-Fehlernachrichten
- hoher Datendurchsatz
- 10Base-T/100Base-TX Schnittstelle (10/100Mbit/s) mit RJ45-Buchse
- 2 CAN-Schnittstellen nach CiA[1] 102 bis 1MBit/s, Highspeed CAN nach ISO11898-1/2, galvanisch getrennt
- 2 CAN-Bus-Anschlüsse: Steckklemm-verbinder jeweils 2x5polig nach CiA 102 bzw. DeviceNet-Standard
- Unterstützung von 11-Bit CAN-Identifier und 29-Bit CAN-Identifier
- RS232-Schnittstelle je nach Variante über USB-Device-Buchse mit USB-seriell-Wandler oder direkt an USB-Device-Schnittstelle der CPU
- Spannungsversorgung: 24VDC +20% -60%, verpolungssicher
- Stromaufnahme: ca. 100mA
- Power-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

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

Abbildung 3: Geräteansicht

Ausbau-Varianten

Grundausbau:

- Spannungsversorgung
- CPU-Kern mit 32MiB SDRAM (32bit) und 4MiB Flash(16bit)
- Ethernet-Interface
- 1x oder 2x galv. entkoppeltes CAN-Interface
- USB-Anschluss ist mit microcontroller-interner USB-Device-Schnittstelle verbunden
- 6 LEDs neben USB-Schnittstelle platziert

Vollausbau:

- Bestandteile des Grundausbaus (außer LEDs)
- 10 LEDs auf Modulfrontplatte angeordnet
- Echtzeituhr mit Batteriepufferung
- 2kiB EEPROM für Daten aus Anwender-Applikationen
- USB über USB-Seriell-Wandler an DRxD/DTxD des Microcontroller

4 Inbetriebnahme

4.1 Spannungsversorgung

Zum Betrieb des Gerätes ist eine Gleichspannung von 24V –60% bis +20% erforderlich. Die Stromaufnahme des Gerätes beträgt ca. 100mA. Der Anschluss erfolgt über einen 2-poligen, abziehbaren Schraubklemmverbinder. Der Anschluss ist am Gerät gekennzeichnet ("+" = "L+" / "-" = "0G"). Der korrekte Anschluss der Versorgungsspannung wird über die Spannungsanzeige „power“ signalisiert.

4.2 Netzwerkanschluss

4.2.1 CAN-Bus-Anschluss

Für das CAN-Netzwerk stehen zwei 2x5-polige abziehbare Steckklemmverbinder zur Verfügung (passender Stecker Weidmüller Minimate B2L 3.5/10). Die Belegung jeder Reihe entspricht dem DeviceNet bzw. CANopen-Standard. Alle Pins der beiden Reihen an den Steckern sind miteinander verbunden. Damit ist es möglich den Bus am Modul durchzuschleifen.

Die Spannungsversorgung für den CAN-Bus (Pin 5A/5B am 2x5-pol. Steckverbinder) ist im Gateway nicht angeschlossen. Die Versorgungsspannung der beiden galvanisch entkoppelten Kanäle wird intern über 2 DC/DC-Wandler realisiert.
Der Schirm-Anschluss dient nur für den Schutz des jeweiligen CAN-Kanals. Es gibt keine Verbindung zwischen den Schirm-Anschlüssen von CAN0, CAN1, USB und Ethernet. Der Schirm ist daher zusätzlich nahe beim Modul mit PE zu verbinden.

2x5-pol.	Name	Beschreibung
1	V-(CAN_GND)	CAN Ground
2	CL (CAN_L)	CAN_L bus line
3	SH (CAN_SHLD)	CAN Shield
4	CH (CAN_H)	CAN_H bus line
5	V+ (CAN_V+)	not connected

Tabelle 1: Belegung CAN-Steckverbinder

4.2.2 Ethernet-Anschluss

Das Ethernet (10Base-T/100Base-TX) 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 V2 und einem PC ist ein Crosslink-Kabel erforderlich.
Der Ethernet-Anschluss ist galvanisch vom CAN-Ethernet-Gateway V2 getrennt.

Pin	Name	Beschreibung
1	TX+	Transmit Data +
2	TX-	Transmit 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

4.2.3 USB-Device-Schnittstelle

Das CAN-Ethernet-Gateway V2 besitzt eine USB-Device--Schnittstelle. Sie wird über einen USB-Stecker Typ B angeschlossen. Diese Schnittstelle erlaubt die Konfiguration des CAN-Ethernet-Gateways V2. Insbesondere ist dieser Anschluss zur Erstkonfiguration vorgesehen (siehe Abschnitt *4.5*). Die USB-Device-Schnittstelle ist nicht galvanisch entkoppelt.

4.3 Statusanzeige

Zur Anzeige des Betriebszustandes dienen insgesamt 10 bzw. 8 LEDs (siehe *Tabelle 3*). Die Anzeigen sind auf der LED-Platine entsprechend der CAN-Netze zusammengefasst und geben deren Status wieder. Zusätzlich ist eine Diagnose-LED der Gatewayapplikation vorhanden.

- Nur in vollständigen Ausbaustufe vorhanden
- In der vollständigen Ausbaustufe sind die beiden Ethernet-LEDs sowohl an der Buchse als auch auf der LED-Platine vorhanden

LED-Bezeichnung	Bedeutung
Power/24V	Spannungsversorgung OK
CAN state 0/1	CAN-Bus 0/1 wird gerade verwendet
CAN error 0/1	Fehler bei der Datenübertragung auf CAN-Bus 0/ 1 (siehe Abschnitt *7.1*)
diagnose	Diagnose-LED der Gatewayapplikation
Link	grün: Ethernetlink vorhanden, Verkabelung OK blinkt: Ethernettraffic aus: kein Ethernetlink
Speed	orange: 100MBit/s aus: 10MBit/s
CAN traffic 0/1	Signalisierung von Datenverkehr auf dem CAN-Bus 0/1

Tabelle 3: Bedeutung der Anzeigeelemente

4.4 Taster

Das CAN-Ethernet-Gateway V2 besitzt 2 Taster (siehe Tabelle 4) mit folgender Bedeutung.

Taster-Nr.:	Bezeichnung	Bedeutung
1	Reset	Kurzes betätigen setzt das Modul zurück und verursacht einen Neustart
2	Boot	Das Halten des Tasters während des Bootvorgangs setzt einige Konfigurationsoptionen zurück, damit im Notfall der Zugriff auf das Modul immer noch möglich ist. Zurückgesetzt werden: • Passwörter von root und gw werden gelöscht • Netzwerkkonfiguration • Gatewayconfig

Tabelle 4: Bedeutung der Taster

4.5 Erstinbetriebnahme

4.5.1 Standardkonfiguration

Werksseitig besitzt das CAN-Ethernet-Gateway V2 folgende Standardkonfiguration (Erstellung des Konfigurations-Script siehe Abschnitt *6.1*):

Ethernet/Internet-Einstellungen

IP-Adresse des CAN-Ethernet-Gateway V2:	192.168.10.49
Subnet-Mask:	255.255.255.0
Standard-Gateway:	192.168.10.1
DNS-Server:	192.168.10.5

CAN-Bus 0
ein UDP[1]-Server
ein TCP-Server
ein Datenlogger

CAN-Einstellungen

CAN-Bitrate:	1Mbit/s
CAN-Identifier für Fehlernachrichten:	0xFC, aber deaktiviert

serielle Schnittstelle über USB-Device-Anschluss

Baudrate:	115200 Baud
Datenbits:	8
Parität:	keine
Stoppbits:	1
Protokoll:	keine

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

4.5.2 Erstkonfiguration über USB-Device-Schnittstelle

Vor der Übertragung von CAN-Nachrichten sind das CAN-Ethernet-Gateway V2 entsprechenden der Anforderungen zu konfigurieren.
Dazu sind folgende Schritte erforderlich:

- Installieren Sie den entsprechenden Treiber für die USB-Schnittstelle: CP210x bei Komplettausbau oder CDC-ACM im anderen Fall. Für Linux verwenden Sie die Kerneltreiber cp210x bzw. cdc-acm. Für Windows sind beide Treiber auf der CD mitgeliefert. Beide Treiber stellen eine serielle Schnittstelle zur Verfügung, auf die mit jedem üblichen Terminal-Emulator zugegriffen werden kann.
- Verbinden Sie das mitgelieferte USB-Kabel mit der USB-Device-Schnittstelle des CAN-Ethernet-Gateway V2 und einer freien USB-Schnittstelle des PC.
- Starten Sie auf dem PC ein Terminal-Programm, (im Weiteren wird das Programm „Terra Term“ verwendet; bei Verwendung eines anderen Terminal-Programms sind die Einstellungen entsprechen vorzunehmen).
- Stellen Sie die Schnittstelle, Baudratenkonfiguration ein (siehe Abbildung 4 und Abbildung 5).

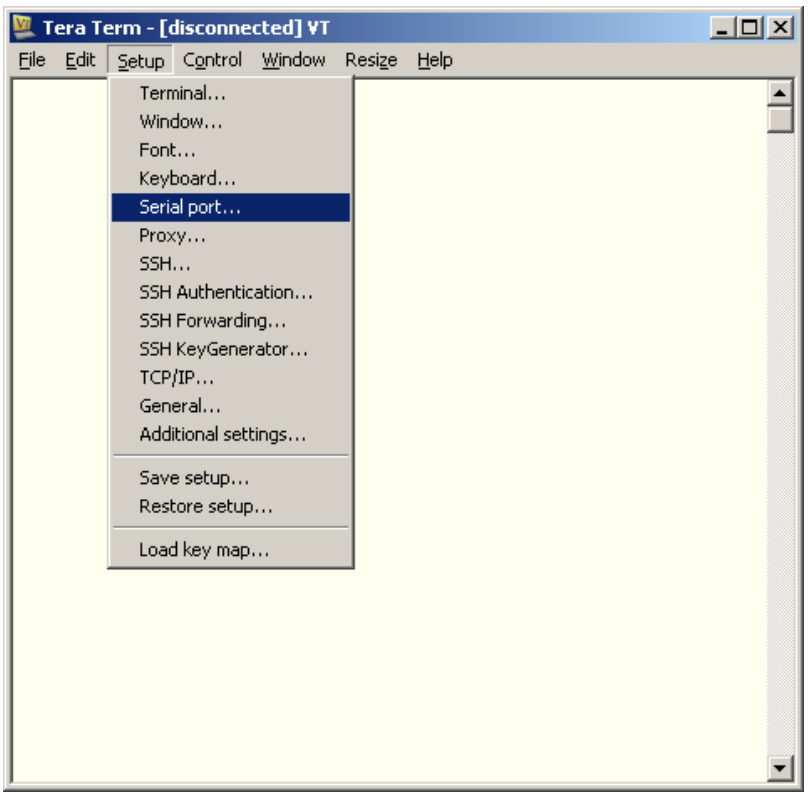

Abbildung 4: Konfiguration des TeraTerm (1)

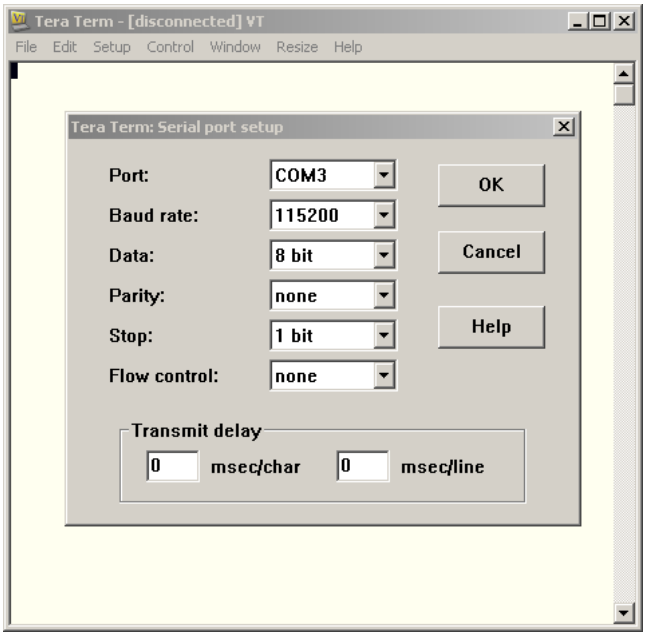

Abbildung 5: Konfiguration des TeraTerm (2)

- Stellen Sie sicher, dass Sie die Spannungsversorgung richtig angeschlossen haben und schalten Sie die Versorgungsspannung ein.
- Beachten Sie, dass nur im Komplettausbau die emulierte serielle Schnittstelle sofort mit dem Anlegen der Stromversorgung erstellt wird und sofort benutzt werden kann. Nur in dieser Variante sind auch die Bootmeldungen zu sehen. In der Variante mit CDC-ACM wird die Schnittstelle erst erstellt, nachdem Linux gebootet hat.

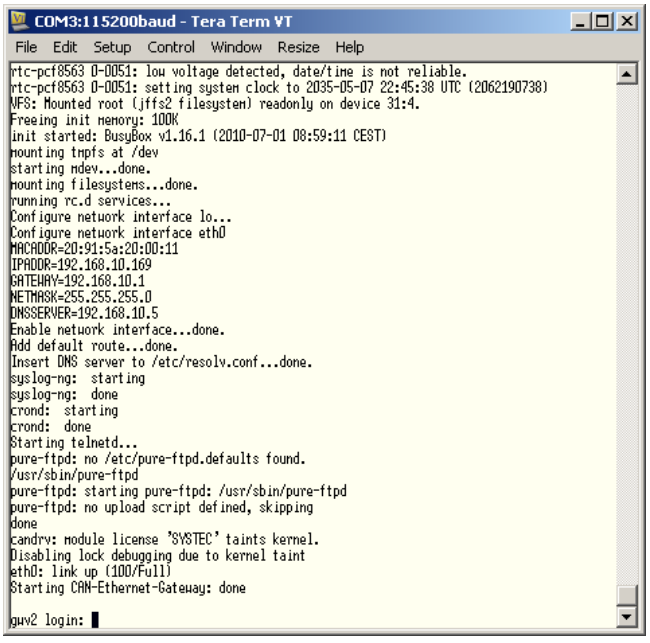

Abbildung 6: Startmeldung des CAN-Ethernet-Gateway V2

In beiden Fällen erscheint nach dem Bootvorgang der Loginprompt und erwartet Login und Passwort für den Systemzugang.

Das Login ist `gw` mit dem Standardpasswort: `gw`.
Für administrative Tätigkeiten, wie z.B. ein Firmwareupdate ist der Nutzer `root` mit dem Standardpasswort `root` vorgesehen. Mit diesem Nutzer ist der komplette Systemzugriff möglich.

Hinweis:
Beim Arbeiten mit dem Nutzer `root` sollte mit großer Vorsicht gearbeitet werden, da im Falle des Nichtbootens nur in der vollständigen Ausbaustufe eine Systemwiederherstellung ohne weitere Hilfe prinzipiell möglich ist.

Da das CAN-Ethernet-Gateway V2 auf Linux mit einem JFFS2-Dateisystem basiert, gibt es verschiedene Kommandos, um darauf zu arbeiten. Diese alle zu erklären, würde den Rahmen dieser Dokumentation sprengen, daher werden nur die notwendigsten kurz beschrieben. Es ist `busybox` mit einer typischen Auswahl an Programmen installiert, die u.A. zur Navigation dienen. Dies sind z.B.:

- `ls` um den Inhalt des aktuellen Verzeichnisses anzuzeigen,
- `pwd` um das aktuelle Verzeichnis anzuzeigen,
- `cd` um das aktuelle Verzeichnis zu wechseln,
- `rm` um ein Verzeichnis/Datei zu löschen,
- `sync` um sicherzustellen, dass alle Daten des JFFS2-Dateisystems in den nichtflüchtigen Speicher (NOR-Flash) gespeichert wurden

Hinweis: Das Programm `sync` ist normalerweise nicht notwendig, da die Dateien nach einer bestimmten Zeit und/oder Datenmenge automatisch in den Flashspeicher geschrieben werden. Jedoch kann durch diesen Befehl das Synchronisieren erzwungen werden.

Die vollständige Liste der Busybox-Programme ist durch das ausführen von `busybox` ohne Argumente möglich.

Für die Gatewayconfiguration steht das Programm `gatewayconfig` unter /usr/sbin/ zur Verfügung. Ohne Parameter aufgerufen listet es die unterstützten Optionen und die aktuelle Konfiguration auf, z.B.

```
gatewayconfig
Usage: gatewayconfig <option> <value>
available options:
  ipaddr       host IP address
  netmask      network mask
  dnsip        DNS server IP address
  gatewayip    gateway IP address
Current configuration:
ipaddr=192.168.10.49
netmask=255.255.255.0
dnsip=192.168.10.5
gatewayip=192.168.10.1
```

Hinweis:
`gatewayip` ist die IP-Adresse des Default-IP-Gateways, nicht die des CAN-Ethernetgateways. Dies ist `ipaddr`!

Dort sollten bei Inbetriebnahme die Einstellungen zu IP-Adresse, Netzmaske, Gateway und DNS-Server eingestellt werden. Die IP-Adresse 0.0.0.0 bedeutet automatische Netzkonfiguration mittels DHCP.

Die Einstellungen werden bei Neustart übernommen. Dazu kann der Befehl `reboot` verwendet werden.
Überprüfen Sie nach einem Neustart, mittels des Programmes `gatewayconfig` die aktuelle Konfiguration. Damit ist die Netzwerkkonfiguration abgeschlossen und das Gateway kann via Ethernet (z.B. Telnet) bedient werden.

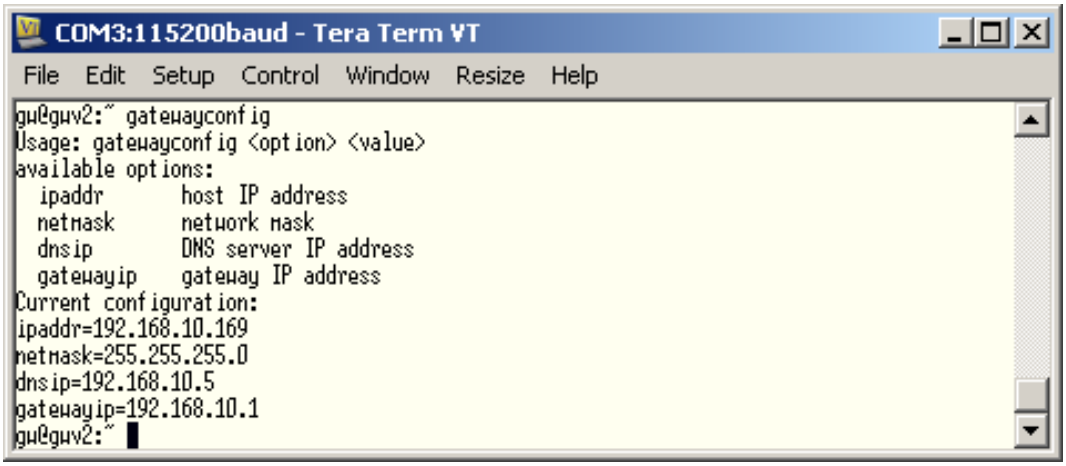

Abbildung 7: Überprüfung der eingestellten Konfiguration

4.5.3 Konfiguration und Bedienung über Telnet

Die Konfiguration im laufenden Betrieb des CAN-Ethernet-Gateways V2 kann auch über Telnet (TCP-Port 23) erfolgen. Der Funktionsumfang ist der Gleiche, wie mit der USB-Device-Schnittstelle. Durch den Zugang über Telnet ist es möglich, auch entfernte CAN-Ethernet-Gateways V2 zu konfigurieren. Voraussetzung ist die einmalige Konfiguration der IP-Adresse (siehe Abschnitt *4.5.3*). Ohne Erstkonfiguration der IP-Adresse ist das CAN-Ethernet-Gateway V2 über seine Standard-Konfiguration ansprechbar (siehe Abschnitt *4.5.1*).
Unter Linux kann das Programm `telnet` verwendet werden. Im Lieferumfang von Windows ist bereits ein Telnet-Client enthalten. Dieser kann über `telnet <Adresse>` aufgerufen werden. Alternativ kann auch das für die serielle Schnittstelle verwendbare PuTTY benutzt werden.
Nach dem Login ist die Vorgehensweise mit der über USB-Device identisch.

5 Gerätefunktion

5.1 Überblick

Das CAN-Ethernet-Gateway V2 enthält mehrere Interfaces für den Betrieb und die Steuerung. Der prinzipielle Aufbau sieht wie folgt aus:

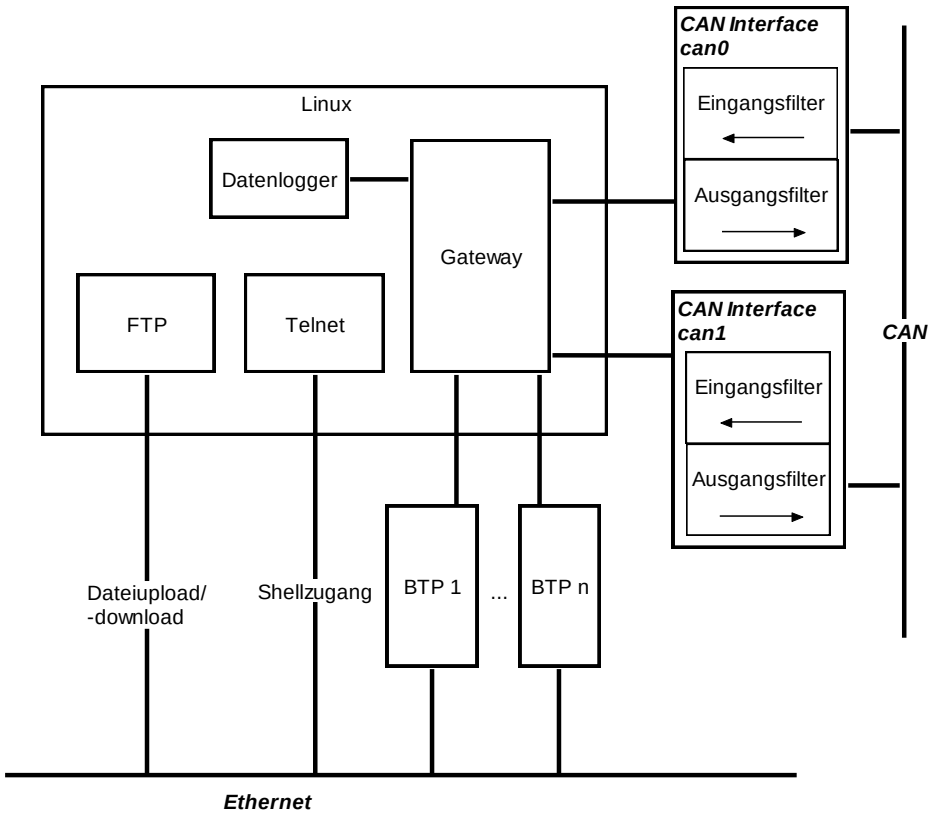

Abbildung 8: Prinzip CAN-Ethernet-Gateway V2

5.2 Interfaces

5.2.1 Grundkonzept

Der Austausch von CAN-Nachrichten erfolgt über Interfaces. Ein Interface stellt die Verbindung zwischen dem Gateway und einer entsprechenden Gegenstelle her, die unterschiedliche Funktionen hat. Es können mehrere Interfaces aktiviert werden.

Die zwei wichtigsten Interfaces sind `CAN` für das CAN-Interface und `BTP_UDP_CAN_SRV` bzw. `BTP_TCP_CAN_SRV` 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
CAN	CAN-Bus
UDP-Client	BTP_UDP_CAN_CLIENT
UDP-Server	BTP_UDP_CAN_SRV
TCP-Client	BTP_TCP_CAN_CLIENT
TCP-Server	BTP_TCP_CAN_SRV
Datenlogger	DLOG

Tabelle 5: 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.

Das Datenlogger-Interface speichert CAN-Nachrichten in eine zu konfigurierende Logdatei. Dabei können auch unerwünschte CAN-Nachrichten herausgefiltert werden.

Alle Interfaces sind instanziierbar und können deshalb, falls benötigt, mehrfach verwendet werden.

5.2.2 UDP/TCP-Server Interface

Das UDP-Server-Interface (`BTP_UDP_CAN_SRV`) 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 (`BTP_TCP_CAN_SRV`) wartet entsprechend auf TCP-Verbindungsanfragen.

Um sowohl auf TCP- als auch auf UDP-basierende Anfragen zu reagieren, müssen ein TCP- und ein UDP-Server-Interface auf dem Gateway existieren.

Das Erstellen der Interfaces erfolgt in der Konfigurationsdatei (siehe Kapitel 5 « Konfiguration des Gateway »).

5.2.3 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 Erstellen der Interfaces erfolgt in der Konfigurationsdatei (siehe Kapitel 6 „Konfiguration des Gateway“)

5.2.4 CAN-Interface

Ein CAN-Interface liefert CAN-Nachrichten an das Gateway weiter, wo sie über Ethernet, Datenlogger oder einem weiterem CAN-Bus weiterverarbeitet werden können.

Das Erstellen der Interfaces erfolgt in der Konfigurationsdatei (siehe Kapiel 6 „Konfiguration des Gateway“).

5.2.5 Datenlogger-Interface

Das Datenlogger-Interface erlaubt es Nachrichten in eine zu konfigurierende Datei zu speichern.
Je nach Speicherort im Dateisystem ist die Logdatei temporär bis zum nächsten Reboot oder persistent im Flash gespeichert.

Das Erstellen der Interfaces erfolgt in der Konfigurationsdatei (siehe Kapitel 6 „Konfiguration des Gateway“).

5.3 Filterung

Die Filterung basiert auf den CAN-Identifiern. Das CAN-Ethernet-Gateway V2 verarbeitet alle empfangenen und versandten Nachrichten anhand von Filterlisten innerhalb der Instanzen. Damit kann der Datenverkehr reduziert werden, wenn z.B. nur noch Nachrichten einer bestimmten Gruppe von CAN-Identifiern (CAN-IDs) weitergeleitet werden.
Je nach Instanztyp können getrennte Filter für den Empfang oder Versand konfiguriert werden. Auch werden 11Bit und 29 Bit CAN-IDs getrennt eingestellt.
Die Schlüsselnamen für die einzelnen Instanztypen werden im Abschnitt 6.4 beschrieben. Die generelle Syntax ist im Abschnitt 6.5 aufgeführt.

5.4 Dateisystem

5.4.1 Aufbau

Im CAN-Ethernet-Gateway V2 ist ein Dateisystem integriert. Dieses ermöglicht es bezüglich der Laufzeit, Konfigurationsänderungen am Gateway durchzuführen. Weiterhin besteht die Möglichkeit, Dateien in einem NOR-Flash nichtflüchtig zu hinterlegen.
Das Dateisystem weist folgende Struktur auf (Auszug):

```
.
|-- bin
|-- dev
|-- etc
|   `-- ceg
|-- home
|-- proc
|-- sbin
|-- sys
|-- tmp
|-- usr
|   |-- bin
|   `-- sbin
`-- var
    `-- log
```

Der Baum unterhalb von /home stellt den Nutzerbereich dar, wo unter anderem die Nutzerkonfiguration des Gateways gespeichert wird.
Die Verzeichnisse /bin und /usr/bin enthalten Programme, die dem Nutzer frei zur Verfügung stehen. Die Programme unter /sbin und /usr/sbin sind für administrative Tätigkeiten installiert.
Das Verzeichnis /tmp ist ein Dateisystem, welches im Arbeitsspeicher liegt und kann deshalb für temporäre Daten benutzt werden. Diese gehen nach einem Neustart verloren.
Zur Navigation innerhalb des Dateisystems stehen mehrere Kommandos zur Verfügung *(siehe Abschnitt 5.5).*

5.5 Beschreibung von wichtigen Befehlen

Auf dem CAN-Ethernet-Gateway V2 ist busybox mit einer typischen Auswahl von Unix/Linux-Befehl vorhanden. Meist bieten die Programme etwas weniger Funktionalität als ihr Pendant auf einem Desktop-Linux, aber die wichtigsten Funktionen sind vorhanden. Daher unterscheidet sich die Bedienung prinzipiell nicht.
Im Folgenden sollen einige wenige Befehle kurz näher erklärt werden, um einen schnellen Einstieg zu schaffen.

5.5.1 cd

Format: `cd [<dir>]`

Bedeutung: Das Kommando `cd` dient zum Wechsel in das angegebenes Verzeichnis `<dir>`. Dabei stellt `..` das übergeordnete Verzeichnis dar. Wird `cd` ohne Argument angegeben, wird in das Nutzerverzeichnis (`/home`) gewechselt.

5.5.2 ls

Format: `ls [<dir>]`

Bedeutung: Das Kommando `ls` zeigt die Dateien des angegebenen bzw. des aktuellen Verzeichnisses an. Mit der Option `-l` werden für jede Datei und jedes Verzeichnis weitere Information ausgegeben.

Beispiel:
```
ls -l /etc/ceg/
-rw-r--r-- 1 root root  578 Jun 10  2010 default.rc
```

5.5.3 rm

Format: `rm <fname>`

Bedeutung: Mit dem Kommando `rm` können Dateien, Verzeichnisse gelöscht werden. `<fname>` entspricht dem Namen der zu löschenden Datei. Für Verzeichnisse muss zusätzlich die Option `-r` (rekursiv) angegeben werden. Alternativ kann für das Löschen von Verzeichnissen `rmdir` verwendet werden.

Beispiel:
`rm /home/datei↵` löscht die Datei `datei` im Verzeichnis `/home`

5.5.4 cat

Format: `cat <fname>`

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

Beispiel:
`cat datei↵` gibt den Inhalt aus

5.5.5 version

Format: `version`

Bedeutung: Das Kommando `version` gibt die Version der CAN-Ethernet-Gateway V2-Firmware aus.

Beispiel:

```
version↵
V1.01
2010.07.0
CAN Ethernet Gateway V2 0.5.0-00042-g25ba83c
U-Boot version: 2010.03-00042-gacdb25b
PCB Version: 4248.1
```

5.5.6 exit

Format: `exit`

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

5.5.7 reboot

Format: `/sbin/reboot`

Bedeutung: Das Kommando `reboot` startet das CAN-Ethernet-Gateway V2 neu (es wird ein Softwarereset ausgelöst).
Der absolute Pfad ist notwendig, da für den eingeschränkten Nutzer das Verzeichnis `/sbin` nicht in PATH liegt.

5.5.8 gatewayconfig

Format: `/usr/sbin/gatewayconfig`

Bedeutung: Das Kommando `gatewayconfig` dient zur Netzwerkkonfiguration über den Bootloader. Ohne Argument werden die verfügbaren Optionen und die aktuelle Konfiguration angezeigt. Da dafür root-Rechte benötigt werden, ist dieses Programm mit dem SUID-Bit installiert und kann von jedem Nutzer benutzt werden.

6 Konfiguration des Gateway

6.1 Grundlagen

Die Konfiguration des CAN-Ethernet-Gateway V2 kann über mehrere Wege erfolgen: über die USB-Schnittstelle mit einem Terminal-Programm (siehe Abschnitt *4.5.2*), eine Telnet-Verbindung über Ethernet (siehe Abschnitt *4.5.3*) oder der Download der Konfigurationsdatei über FTP.

Die Konfiguration des CAN-Ethernet-Gateway V2 erfolgt über eine Konfigurationsdatei. Bis auf wenige Ausnahmen sind die Optionen optional und werden mit entsprechenden Standardwerten belegt, falls sie nicht angegeben sind. Das Format entspricht dem Windows-INI-Format. Die Konfigurationsdatei ist in mehrere Sektionen unterteilt, die unterschiedliche Teile konfigurieren.

6.2 Abschnitt [Interfaces]

In diesem Abschnitt werden die einzelnen Interfaces bzw. Instanzen definiert. Der Schlüssel ist eine laufende Nummer von 0 an aufsteigend. Der Wert dazu ist der Name der Instanz und durch Leerzeichen getrennt der Typ.
Eine Besonderheit ist der Typ CAN da dort der Name auch die CAN-Instanz des Treibers bestimmt. So bedeutet CAN0 CAN-Bus 0 und CAN1 verwenden den CAN-Bus 1.

Typbezeichner	Interfacetyp
CAN	Can-Schnittstelle
DLOG	Datenlogger
BTP_UDP_CAN_SRV	BTP-UDP-Server
BTP_UDP_CAN_CLIENT	BTP-UDP-Client
BTP_TCP_CAN_SRV	BTP-TCP-Server
BTP_TCP_CAN_CLIENT	BTP-TCP-Client
ASCII_TCP_SRV	Ascii-TCP-Server

Tabelle 6: Instanztypen

Beispiel:
```
[Interfaces]
0=CAN0 CAN
1=Btp BTP_UDP_CAN_SRV
2=Remote1 BTP_TCP_CAN_CLIENT
3=Dlog DLOG
```

Mit dieser Konfiguration werden eine CAN-Schnittstelle, ein BTP-UDP-Server, ein UDP-TCP-Client und ein Datenlogger angelegt.
Die Konfiguration zu den einzelnen Schnittstellen wird separat beschrieben.

6.3 Abschnitt [Connections]

In diesem Abschnitt werden die Instanzen miteinander verbunden. Dazu kann für jede Instanz angegeben werden, von welchen anderen Instanzen sie Nachrichten empfangen möchten.

Beispiel (Fortsetzung):

```
[Connections]
CAN0=Btp
DLog=CAN0 Btp
Btp=CAN0
Remote1=CAN0
```

Mit diesem Beispiel werden alle über den BTP-UDP-Server empfangen Nachrichten über CAN und umgekehrt versandt. Zusätzlich werden die Nachrichten vom CAN-Bus und vom BTP-UDP-Server von Datenlogger gespeichert.
Weiterhin werden die Nachrichten vom CAN-Bus über BTP-TCP verschickt.

6.4 Abschnitte [InstanceX]

In den Abschnitten [InstanceX], wobei X der Zahl aus [Interfaces] entspricht, werden die einzelnen Instanzen konfiguriert. Die konkreten Optionen sind vom Typ der Instanz abhängig und unterschiedlich.

6.4.1 Instanztyp CAN

Baudrate	CAN-Baudrate in kBaud. Kann durch UserBaud überschrieben werden. erlaubte Werte: 20, 50, 100, 125, 250, 500, 800, 1000 Standard: 125
UserBaud	Nutzerdefinierte Bitrate. Falls gesetzt, wird die Einstellung in Baudrate ignoriert. Standard: nicht definiert 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
Acr	Acceptance Code Register Standard: 0
Amr	Acceptance Mask Register Standard: 0xFFFFFFFF
LowbufRxMaxEntries	Größe des Empfangspuffers im CAN-Treiber Standard: 5000
LowbufTxMaxEntries	Größe des Sendepuffers im CAN-Treiber Standard: 5000
FilterStdRejectIn	Filterliste mit Standard-CAN-IDs, die nicht empfangen werden dürfen

	Standard: <leer>
FilterStdAllowIn	Filterliste mit Standard-CAN-IDs, die empfangen werden dürfen Standard: <leer>
FilterStdRejectOut	Filterliste mit Standard-CAN-IDs, die nicht versandt werden sollen Standard: <leer>
FilterStdAllowOut	Filterliste mit Standard-CAN-IDs, die versandt werden sollen Standard: <leer>
FilterExtRejectIn	Filterliste mit erweiterten CAN-IDs, die nicht empfangen werden dürfen Standard: <leer>
FilterExtAllowIn	Filterliste mit erweiterten CAN-IDs, die empfangen werden dürfen Standard: <leer>
FilterExtRejectOut	Filterliste mit erweiterten CAN-IDs, die nicht versandt werden sollen Standard: <leer>
FilterExtAllowOut	Filterliste mit erweiterten CAN-IDs, die versandt werden sollen Standard: <leer>
ListenOnly	Listen-only-Mode des CAN-Controllers aktivieren 0: deaktiviert 1: aktiviert Standard: 0
EnableErrorMsg	Verschicken von CAN-Fehler-Nachrichten aktivieren 0: deaktiviert 1: aktiviert Standard: 0
ErrorMsgId	CAN-ID der CAN-Fehlernachricht nur Standard-CAN-IDs unterstützt Standard: 0xFE

Tabelle 7: Optionen für den Instanztyp CAN

6.4.2 Instanztyp DLOG

LogFile	Dateiname der Logdatei muss angegeben werden!
FilterStdReject	Filterliste mit Standard-CAN-IDs, die nicht gespeichert werden sollen Standard: <leer>
FilterStdAllow	Filterliste mit Standard-CAN-IDs, die gespeichert werden sollen Standard: <leer>
FilterExtReject	Filterliste mit erweiterten CAN-IDs, die nicht gespeichert werden sollen Standard: <leer>
FilterExtAllow	Filterliste mit erweiterten CAN-IDs, die gespeichert werden sollen Standard: <leer>

Tabelle 8: Optionen für den Instanztyp DLOG

6.4.3 Instanztyp BTP_UDP_CAN_SRV

LocalPort	Port an den der BTP-UDP-Server gebunden wird Standard: 8234
TriggerTime	Nach wie vielen ms soll der BTP-Transfer spätestens beginnen Standard: 0
TriggerCount	Nach wie vielen CAN-Nachrichten soll spätestens der BTP-Transfer beginnen Standard: 1

Tabelle 9: Optionen für den Instanztyp BTP_UDP_CAN_SRV

6.4.4 Instanztyp BTP_UDP_CAN_CLIENT

RemoteIP	IP zu der sich der BTP-UDP-Client verbinden soll Standard: 0.0.0.0
RemotePort	Port zu den sich der BTP-UDP-Client verbinden soll Standard: 8234
ReconnectionType	Einstellungen zum automatischen Wiederverbinden bei Verbindungsverlust 0: nicht wiederverbinden 1: wiederverbinden, wenn Nachricht verschickt werden soll 2: sofort wiederverbinden (Standard)
TriggerTime	Nach wie vielen ms soll der BTP-Transfer spätestens beginnen (Standard: 0)
TriggerCount	Nach wie vielen CAN-Nachrichten soll spätestens der BTP-Transfer beginnen (Standard: 1)

Tabelle 10: Optionen für den Instanztyp BTP_UDP_CAN_CLIENT

6.4.5 Instanztyp BTP_TCP_CAN_SRV

LocalPort	Port an den der BTP-UDP-Server gebunden wird Standard: 8234
TriggerTime	Nach wie vielen ms soll der BTP-Transfer spätestens beginnen Standard: 0
TriggerCount	Nach wie vielen CAN-Nachrichten soll spätestens der BTP-Transfer beginnen Standard: 1

Tabelle 11: Optionen für den Instanztyp BTP_TCP_CAN_SRV

Hinweis: Bei einer TCP-BTP-Verbindung zwischen CAN-Ethernet-Gateway V2 und einem Windows PC, der die EthCan.dll verwendet, kann es vorkommen, dass das TCP Window sich erschöpft. Um dies zu vermeiden, wird eine Einstellungen für TriggerTime und TriggerCount auf jeweils mindestens 10 empfohlen.

6.4.6 Instanztyp BTP_TCP_CAN_CLIENT

RemoteIP	IP zu der sich der BTP-UDP-Client verbinden soll Standard: 0.0.0.0
RemotePort	Port zu den sich der BTP-UDP-Client verbinden soll Standard: 8234
ReconnectionType	Einstellungen zum automatischen Wiederverbinden bei Verbindungsverlust 0: nicht wiederverbinden 1: wiederverbinden, wenn Nachricht verschickt werden soll 2: sofort wiederverbinden Standard: 2
TriggerTime	Nach wie vielen ms soll der BTP-Transfer spätestens beginnen Standard: 0
TriggerCount	Nach wie vielen CAN-Nachrichten soll spätestens der BTP-Transfer beginnen

	Standard: 1

Tabelle 12: Optionen für den Instanztyp BTP_TCP_CAN_CLIENT

Hinweis: Bei einer TCP-BTP-Verbindung zwischen CAN-Ethernet-Gateway V2 und dem CAN-Ethernet-Gateway V1 kann es aufgrund der Performanceunterschiede dazu kommen, dass das Gateway V1 mit TCP überschüttet wird. Damit es nicht zu Nachrichtenverlust und Ausfällen kommen kann, wird empfohlen die Einstellungen für TriggerTime und TriggerCount auf jeweils mindestens 10 zu setzen. Je nach konkreten CAN-Bus-Nachrichtenaufkommen, sind diese Werte ggf. weiter zu erhöhen.

6.4.7 Instanztyp ASCII_TCP_SRV

LocalPort	Port an den der ASCII-TCP-Server gebunden wird Standard: 12000
Timestamp	Wenn ungleich 0, werden im ASCII-Protokoll Zeitstempel mit versendet. Standard: 0

Tabelle 13: Optionen für den Instanztyp ASCII_TCP_SRV

6.5 Filter

Die Filteroptionen funktionieren in den Instanzen gleich, wenn sie unterstützt werden. Es gibt getrennte Einstellungen für Standard- bzw. erweiterte CAN-IDs (11 oder 29 Bits). Dabei kann getrennt für den Versand und den Empfang eingestellt werden, welche Nachrichten empfangen bzw. verworfen werden sollen. Empfang und Versand ist aus Sicht der einzelnen Instanz zu sehen. So bedeutet z.B. Empfang für die CAN-Instanz das Weiterleiten von CAN-Nachrichten vom CAN-Bus zum Gateway. Versand bedeutet umgekehrt das Weiterleiten von CAN-Nachrichten vom Gateway auf den CAN-Bus.
So kann z.B. der Empfang einer Nachricht über die CAN-Schnittstelle erlaubt sein, aber nach der Weiterleitung an den Datenlogger dieser die Nachricht nicht speichert.
Bei einigen Instanzen wie z.B: dem Datenlogger gibt es nur Filtereinstellungen für den "Versand", da der Datenlogger selbst keine Nachrichten erzeugen kann.
Dabei gilt: Es werden nur die Nachrichten weitergeleitet, deren ID erlaubt wurde (Accept-Regel), aber nicht explizit verboten wurde (Reject-Regel). D.h. ohne eine Einstellung werden alle Nachrichten verworfen.
Für alle Filter gilt die folgende Syntax, die in der Konfigurationsdatei als Wert angegeben werden kann:

<FilterList> ::= [<FilterEntry>[;<FilterList>]]
<FilterEntry> ::= <CanIdLow>-<CanIdHigh>
<CanIdLow> untere CAN-ID für Filterbereich
<CanIdHigh> obere CAN-ID für Filterbereich

z.B.

```
FilterStdAllowIn=0x0-0x7ff
```

Erlaubt den Empfang aller 11Bit CAN-IDs

```
FilterStdAllowIn=0x0-0x7ff
FilterStdRejectIn="0x251-0x251;0x300-0x301"
```

Erlaubt den Empfang aller 11Bit CAN-IDs außer, 0x251 und von 0x300 bis 0x301

Um einzelne CAN-IDs zu filtern, ist für die obere und unteren CAN-ID der gleiche Wert zu setzen.
z.B.
"0x50-0x50;" um die CAN-ID 0x50 zur Filterliste hinzuzufügen.

Hinweis:
Da das ';' das Kommentarzeichen im INI-Dateiformat ist, muss die Filterangabe in "" gesetzt werden, wenn mehrere Einträge genutzt werden sollen.

6.6 Beispiel für ein kundenspezifisches Konfigurations-Script

In der folgenden wird eine Beispielkonfiguration mit Erläuterungen gezeigt.

```
[Interfaces]
0=CAN0 CAN
1=Btp BTP_UDP_CAN_SRV
2=Dlog DLOG
```

In diesem Bereich werden die einzelnen Schnittstellen angelegt. In diesem Fall eine CAN-Schnittstelle für den Bus 0 (siehe Abschnitt 6.2). Zusätzlich gibt es einen BTP-UDP-Server und einen Datenlogger

```
[Connections]
CAN0=Btp
DLog=CAN0 Btp
Btp=CAN0
```

In diesem Bereich werden die Instanzen miteinander verbunden. In diesem Beispiel wird CAN mit BTP in beide Richtungen verbunden und zusätzlich erhält der Datenlogger alle CAN-Nachrichten vom CAN-Bus und BTP.

```
[Instance0]
BaudRate=1000
FilterStdRejectIn="0x251-0x251;0x300-0x301"
FilterStdAllowIn=0x0-0x7ff
FilterStdAllowOut=0x0-0x7ff
```

Die Instanz 0 bezieht sich auf den Schlüssel 0 aus [Interfaces] und es handelt sich demzufolge um eine CAN-Schnittstelle. Es wird die Baudrate auf 1000kBaud gesetzt.
Die Filter erlauben den Versand und Empfang aller 11Bit CAN-IDs, wobei die IDs 0x251, 0x300 und 0x301 beim Empfang nicht akzeptiert werden.

```
[Instance1]
LocalPort=8234
```

Der BTP-Server verwendet den UDP-Port 8234 und wartet auf Verbindungen.

```
[Instance2]
FilterStdAllow="0x25A-0x25A;0x250-0x252"
LogFile=/tmp/logfile.txt
```

Der Datenlogger speichert die Daten in der Datei /tmp/logfile.txt und speichert nur CAN-Nachrichten mit den IDs 0x250, 0x251, 0x252 und 0x25a.

6.7 Erstellung eines Konfigurations-Scripts

Es besteht die Möglichkeit die Konfigurationsdatei direkt in der Linuxkonsole zu erstellen bzw. zu bearbeiten. Dazu ist ein Zugang über die serielle Konsole oder Telnet notwendig. Auf dem Modul sind die Texteditoren vi (eine reduzierte Variante aus der Busybox) und

nano installiert und können nach persönlicher Präferenz verwendet werden. Für die Verwendung sei auf die jeweiligen Webseiten verwiesen (http://www.vim.org/docs.php, http://www.nano-editor.org/docs.php), da diese den Umfang dieser Dokumentation überschreiten würden.
Alternativ kann die Konfigurationsdatei auch am PC erstellt bzw. bearbeitet werden und über FTP auf das Gateway kopiert werden.

Als guter Einstiegspunkt kann die vorhandene Standarddatei von /etc/dec/default.rc nach /home/ceg/default.rc kopiert und dort bearbeitet werden.

Hinweis:
In jedem Fall muss die Konfigurationsdatei das Unix-Zeilenende (nur Linefeed) haben.

7 Fehlerbehandlung

7.1 Fehlersignale des CAN-Ethernet-Gateway V2

Fehler werden einerseits durch das Aufleuchte von LEDs signalisiert bzw. wird ein Eintrag in das Systemlog geschrieben. Dieses kann unter `/var/log/messages` gefunden werden.
Folgende Fehlerzustände werden signalisiert:

- Verbindungsaufbau über BTP-Interface nicht möglich
  mögliche Ursachen:
  keine Verbindung zum konfigurierten Server möglich
  oder Verbindung abgelehnt
  Gateway: Eintrag im Systemlog

- CAN-RxBuffer-Überlauf, CAN-Nachrichtenverlust
  mögliche Ursachen:
  zu hohe CAN-Buslast
  Empfangspuffer zu klein gewählt (Einstellung siehe Abschnitt 5.2.4)
  Gateway: CAN-Error-LED leuchtet kurz auf

- CAN-TxBuffer-Überlauf
  CAN-Nachrichten werden zu schnell in CAN-Sendepuffer eingereiht (bspw. durch
  hohe Buslast und niedrige Priorität der zu sendenden CAN-ID)
  Betreffende CAN-Nachricht wird verworfen.
  Gateway: CAN-Error-LED leuchtet kurz auf

- BTP-TxBuffer-Überlauf
  keine CANtoBTP-Puffer verfügbar -> Nachrichtenverlust
  Gateway: Eintrag im Systemlog

- BTP-RxBuffer-Überlauf
  keine BTPtoCAN-Puffer verfügbar -> Nachrichtenverlust
  Gateway: Eintrag im Systemlog

- 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: Eintrag im Systemlog

- CAN-Busoff Fehler
  mögliche Ursachen:
  CAN-Bus-Verkabelung,
  falsche CAN-Bitrate,
  Hardwarefehler
  Gateway: CAN-Error LED blinkt solange, bis die Sendung bzw. der Empfang eines CAN-Telegramms erfolgreich war

7.2 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 Optionen `EnableErrorMsg ErrorMsgId` bei der Konfiguration des CAN-Interfacess kann der Versand von Fehlernachrichten mit dem entsprechenden Identifier im CAN-Ethernet-Gateway V2 eingeschaltet werden.
In der Standardkonfiguration ist der Versand ausgeschaltet. 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 14: Aufbau der Emergency-Nachricht

Der Emergency-Code kann folgende Werte annehmen:

0x1000: wenn ein neuer allgemeiner Fehler aufgetreten ist
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
0x0400	Sammelfehler des CAN-Controllers, interne Hardwarefehler des SJA-1000 (Stuff-Error, Form-Error, CRC-Error)

7.3 Statusübersicht

Für Supportanfragen gibt es das Script `support-info` unter /usr/sbin/, was einen Überblick über die Konfiguration und Fehlermeldungen erstellt und in der Datei `/tmp/support-info.txt` speichert. Diese kann bei Supportanfragen mitgeliefert werden. Zusätzlich werden die Informationen auch auf der Textkonsole ausgegeben.
Um das Script aufzurufen, muss sich der Nutzer über Telnet oder die serielle Konsole anmelden. Anschließend wird das Kommando `support-info` ausgeführt. Die

angezeigten Informationen können direkt aus dem Terminal- oder Telnetprogramm kopiert werden. Alternativ kann die Datei support-info.txt direkt über FTP vom Board geladen werden.

8 Softwareunterstützung

8.1 Anbindung des CAN-Ethernet-Gateways V2 an den PC

Für die Anbindung des CAN-Ethernet-Gateway V2 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 V2 kann über diese Treiber-DLL direkt vom PC aus über Ethernet angesprochen werden.

8.2 Treiberinstallation unter Windows

Im Vorfeld ist die Installation der Treiber-DLL für Windows notwendig. Das zugehörige Setup-Programm finden Sie auf der Webseite.

Starten Sie das heruntergeladene 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 V2l_Utility_Disk

Der Installationspfad kann jedoch beliebig verändert werden kann.

Hinweis:
Stellen Sie sicher, dass Sie für die Installation unter den Betriebssystemen Windows 2000 und XP 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 V2
Include	„C“-Header-Datei für die EthCan.Dll
Lib	EthCan.Lib und EthCan.Dll

Tabelle 15: Verzeichnisstruktur CAN-Ethernet-Gateway V2l_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 V2 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 eines Demo-Programms zeigt.

8.3 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-Gateway V2 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.

8.3.1 Das Konzept der *EthCan.Dll*

Mit der ***EthCan.Dll*** können bis zu 5 CAN-Ethernet-Gateway V2 innerhalb einer Applikation gleichzeitig angesprochen werden. Des Weiteren kann von einer weiteren Applikation aus wiederum auf bis zu 5 CAN-Ethernet-Gateway V2 zugegriffen werden, die zudem die gleichen Remote-Adressen haben, wie in Applikation 1, da auf dem CAN-Ethernet-Gateway V2 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 V2 aufzubauen.

Bei der Verwendung dieser DLL entstehen für jedes CAN-Ethernet-Gateway V2 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 V2 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 16: Funktionsumfang der Softwarezustände

Werden mehrere CAN-Ethernet-Gateway V2 innerhalb einer Anwendung verwendet, so sind diese Zustände für jedes CAN-Ethernet-Gateway V2 zu verstehen. Während das erste CAN-Ethernet-Gateway V2 bereits im Zustand DLL_INIT ist, kann sich das zweite noch im Zustand HW_INIT befinden.

8.3.2 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.

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);
```

8.3.2.1 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 V2 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 V2 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 V2

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 9: 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 V2), 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 10: Übertragungsprotokolle CAN-Ethernet-Gateway V2

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 V2.

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 11: Verbindungsstatus CAN-Ethernet-Gateway V2

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 V2 möglich, wenn von einer Applikation mehrere Gateways angesprochen werden sollen, die innerhalb einer Instanztabelle verwaltet 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 werden. Dies hängt vor allem von der Applikation ab.

Rückgabewerte: (siehe Kapitel *8.3.4*)

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 V2 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 V2 mit Callbackfunktion initialisieren
dwRetcode = EthCanInitHardware (&EthCanHandle, &EthCanHwParam,
                                EthCanConnectControlFct,NULL);
```


8.3.2.2 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 Deinitialisierung 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 V2

Rückgabewerte: (siehe Kapitel *8.3.4*)

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 V2 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");
  }

  //Auf Verbindungsabbau warten, signalisiert durch Callback-Funktion
  do
  {
    Sleep(10);

  }while(EthCanInst_g[EthCanHandle].m_fConnected);

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

8.3.3 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 V2

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 12: Aufbau CAN-Nachrichten-Struktur

Beim Nachrichtenformat der CAN-Nachricht wird zwischen verschiedenen 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 13: 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 V2 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 zur Zeit nicht verwendet und enthalten stets den Wert 0.

Rückgabewert: (siehe Kapitel *8.3.4* und *8.3.5*)

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
}
```

8.3.3.1 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, dass heißt, es hat ein Pufferüberlauf stattgefunden.

Parameter:

EthCanHandle_p: Instanz-Handle des CAN-Ethernet-Gateways V2

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 *0*) erläutert wurde. Je nachdem welche CAN-Nachricht (29-Bit,11-Bit, RTR) gesendet werden soll, so ist das Nachrichtenformat (siehe Kapitel *0*) 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 *8.3.4* und *8.3.5*)

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
}
```

8.3.3.2 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-Gatways zurück. Tritt auf dem CAN-Ethernet-Gateway V2 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 V2 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 V2

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 14: Aufbau der CAN-Status-Struktur

Rückgabewerte: (siehe Kapitel *8.3.4*)

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
}
```

8.3.3.3 EthCanGetConnectionState

Syntax:

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

Verwendbarkeit:
HW_INIT

Parameter:

EthCanHandle_p: Instanz-Handle des CAN-Ethernet-Gateways V2

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

Bedeutung:

Diese Funktion liefert den aktuellen Verbindungsstatus des CAN-Ethernet-Gateway V2. 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 11* erläutert wurden.

Rückgabewerte: (siehe Kapitel *8.3.4*)

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
}
```

8.3.3.4 EthCanResetCan

Syntax:

DWORD PUBLIC EthCanResetCan(
tEthCanHandle,
dwResetCode_p)

Verwendbarkeit:
HW_INIT

Parameter:

EthCanHandle_p: Instanz-Handle des CAN-Ethernet-Gateways V2

dwResetCode_p: Reset-Code für CAN

Bedeutung:

Diese Funktion dient zum gezielten Zurücksetzen der CAN-Kommunikation des CAN-Ethernet-Gateways V2 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-Gatways 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 Buffer können keine CAN-Nachrichten gesendet und empfangen werden!

Rückgabewerte: (siehe Kapitel *8.3.4*)

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 V2
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
}
```

8.3.4 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 17: 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 V2, 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 V2 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 dass 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 V2 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.

8.3.5 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 18: CAN-Fehlercodes

ETHCAN_CANERR_OK

Kein CAN-Fehler aufgetreten

ETHCAN_CANERR_XMTFULL

Der Sendepuffer im CAN-Controller des CAN-Ethernet-Gateways V2 hat die maximale Anzahl an CAN-Nachrichten erreicht.

ETHCAN_CANERR_OVERRUN

Der Empfangspuffer im CAN-Controller des CAN-Ethernet-Gateways V2 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.

8.3.6 Anwendung der DLL-Funktionen

8.3.6.1 Demo-Projekt

Wie im Kapitel *8.2* 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 V2 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 V2 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 V2 angeschlossen ist.

Das Senden von CAN-Nachrichten auf dem CAN-Bus des CAN-Ethernet-Gateways V2 kann ebenfalls von einem Analysetool oder einer weiteren angeschlossenen CAN-Hardware erfolgen. Der Empfang der CAN-Nachricht vom CAN-Ethernet-Gateway V2 auf dem PC wird im Konsolen-Fenster der Demo-Applikation mit Ausgabe der CAN-ID, der CAN-Nachrichtenlänge sowie der enthaltenen Daten quittiert.

8.3.6.2 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 Abbildung 15 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.

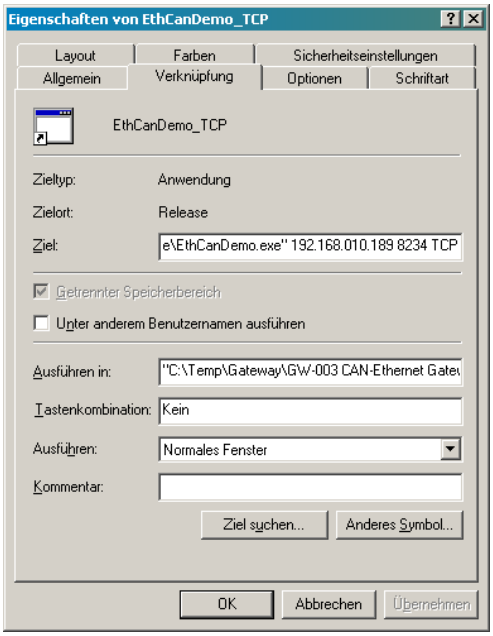

Abbildung 15: Desktop-Verknüpfung für Demo-Programm

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

9 ASCII-Protokoll

Um möglichst frei in der Wahl der Gegenstelle zum CAN-Ethernet-Gateway V2 wurde ein ASCII-Protokoll entwickelt. Das Format ist so entworfen, dass man auch Daten manuell interpretieren kann, wenn die transferierten Daten übertragen werden.

9.1 Verbindungsaufbau

Das ASCII-Protokoll setzt auf TCP auf und verwendet standardmäßig den Port 12000. Dies kann aber ggf. in der Konfiguration verändert werden. Bei mehreren ASCII-Instanzen gleichzeitig muss jeweils ein anderer TCP Port verwendet werden.
Sobald eine TCP-Verbindung aufgebaut wurde, können ASCII-nachrichten an das CAN-Ethernet-Gateway V2 versandt werden. Gleichzeitig beginnt das CAN-Ethernet-Gateway V2 auch mit dem Versand von Nachrichten, wenn welche an das Interface weitergereicht werden.

9.2 Übertragungsformat

Alle ASCII-Nachrichten starten mit $ und werden mit # abgeschlossen. Es können auch mehrere ASCII-Nachrichten mit einmal verschickt werden, da sie jeweils mit dem Anfangs und Endzeichen markiert sind.
Zur besseren manuellen Lesbarkeit ist zwischen dem Endzeichen und dem Anfangszeichen der nächsten Nachricht ein Carriage-Return-Zeichen (CR, \r oder 0xd) und/oder ein Newline-Zeichen (NL, \n oder 0xa) erlaubt, aber nicht zwingend notwendig. Das CAN-Ethernet-Gateway V2 wird diese Zeichen immer an eine ASCII-Nachricht anhängen.
Alle Elemente der ASCII-Nachricht werden durch ein ; abgetrennt und es dürfen keine Leerzeichen, Zeilenumbruch oder andere nicht erlaubte Zeichen innerhalb der ASCII-Nachricht auftreten.
Syntaktisch oder inhaltlich fehlerhafte Nachrichten werden stillschweigend verworfen.

Genereller Aufbau:

```
$<Typ>;<typspezifischer Inhalt>;#
```

9.2.1 CAN-Nachrichten

CAN-Telegramme werden in ASCII-Nachrichten verschickt, die den folgenden Aufbau besitzen:

```
$CAN;<Zeistempel><ID-Typ>;<Frame-Typ>;<CAN-
   ID>;<DLC>;<Data-Bytes>;#
```

Dabei sind für die einzelnen Elemente folgende Werte zulässig:

Element	Zulässiger Inhalt
Zeitstempel	Wird nur gesendet, wenn Timestamp für die Instanz aktiviert wurde (siehe 6.4.7) Format: "<Zeit>;"

	Zeitbasis ist 1ms
ID-Typ	"S": Standard CAN-ID "E": Extended CAN-ID
Frame-Typ	"D": Data Frame "R": RTR-Frame
CAN-ID	Zahl für die CAN-ID. Dabei werden die Prefix 0x für Hexadezimal und 0 für Oktal unterstützt. Eine Zahl ohne Prefix wird als Dezimalzahl interpretiert. Dabei ist zu beachten, dass abhängig von ID-Typ unterschiedliche Wertebereiche zulässig sind.
DLC	Zahl, die dem Data Length Code der CAN-nachricht entspricht. Gültige Werte: 0, 1, 2, 3, 4, 5, 6, 7, 8

`<Data-Bytes>` ist eine Liste von Zahlen (durch Semikolon getrennt), die den Datenbytes der CAN-nachricht entsprechen. Auch hier werden die Prefixe für Oktal und Hexadezimal unterstützt.
Die Reihenfolge der Liste beginnt mit dem Daten-Byte 0, gefolgt von Daten-Bytes 1 usw. Bis maximal Daten-Byte 8.
Bei den Datenbytes ist zu beachten, dass die Menge an Bytes der im DLC angegebenen Größe entsprechen muss. Außer bei RTR-Nachrichten, die zwar keine Datenbytes haben, aber einen DLC größer 0 besitzen können.
Hinweis:

Beginnend mit Firmwareversion 1.2.2 werden vom Gateway alle DLC und DatenBytes im Format 0x00 generiert. Auch die CAN-ID wird im Format 0x7ff bzw. 0x1fffffff erstellt. Dies ermöglich ein einfacheres Parsen, da die Länge der Elemente immer gleich lang ist.
Eine Beispielimplementierung für das Parsen dieses Formats ist unter http://www.systec-electronic.com/can-ethernet im Tab "Downloads" unter dem Eintrag "CAN-Ethernet Gateway V2 - ASCII parser example" zu finden.

Hinweis:

Beginnend mit Firmwareversion 1.2.4 is es möglich Zeitstempel zu versenden und empfangen. Diese Funktion ist standardmäßig deaktiviert, um mit bisherigen Implementierungen kompatibel zu bleiben. Unabhäng von der Konfiguration ist das Empfangen von ASCII-Frames mit Zeitstempel immer möglich.

10 Beschreibung des Firmwareupdates

Zum Firmwareupdate muss über FTP die neue Firmware auf das Gateway geladen werden. Dort wird dann über Telnet oder USB-Schnittstelle der Updatevorgang gestartet. Dabei wird die neue Firmware in den nichtflüchtigen Programmspeicher (Flash) geschrieben.
Für den Vorgang wird eine bestehende Ethernetverbindung zwischen dem Gateway und den PC vorausgesetzt. Auf dem PC werden ein FTP-Client und ein Terminalprogramm benötigt.

10.1 Vorbereitungen

Folgende Schritte sind zur Vorbereitung des Firmwareupdates durchzuführen:

1. Schließen Sie die Spannungsversorgung an das CAN-Ethernet-Gateway V2 an.

2. Verbinden Sie sich nach dem Bootvorgang über FTP mit dem Gateway.

3. Stellen Sie über das Terminalprogramm oder einen Telnetclient eine Verbindung zur Konsole her. Melden sie sich als `root` an.

10.2 Firmwaredownload

Kopieren Sie die neue Firmware über FTP nach /tmp auf das Gateway. Danach wird die FTP-Verbindung nicht mehr benötigt und kann getrennt werden. Anschließend starten Sie das Updateprogramm `gatewayupdate`. Dazu ist die neue Firmware als Argument anzugeben, z.B. `gatewayupdate /tmp/V1.01_update.tar.gz`.
Nach Abschluss des Firmware-Updates muss das Gateway neu gestartet werden. Verwenden Sie dazu den Befehl `reboot`.

Index

Dokument:	CAN-Ethernet-Gateway V2
Dokumentnummer:	L-1294d_10, Auflage Dezember 2014

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
Am Windrad 2
D-08468 Heinsdorfergrund
GERMANY
Tel: +49 3765 38600-0
Fax +49 3765 38600-4100

Veröffentlicht von

SYS TEC ELECTRONIC

Best.-Nr. L-1294d_10
Printed in Germany