esd gmbh

CAN-API

mit Software-Tools
und Installationshinweisen

Software-Handbuch

Handbuch-Datei:	I:\texte\Doku\MANUALS\PROGRAM\CAN\Schicht2\DEUTSCH\Univers\UNIAPI24.ma9
Handbuch-Bestellnummer:	C.2001.20
Datum des Ausdrucks:	27.02.2002

Beschriebene Software	**Revision**	**Bestellnr.**
Windows 95/98/ME VxD-Driver	V1.0x	C.20xx.10
Windows NT Device-Driver	V1.6.x	C.20xx.11
Windows 2000/XP Device-Driver	ab V.2.x.x	C.20xx.26
CANscope	V1.21	-
CANbatch	V1.1	-
cantest	V1.0	-
Linux-Driver	V2.4.x	C.20xx.19
Linux-RTAI-Driver	V2.4.x-RTAI	C.20xx.33
Linux-RT-Driver	V2.4.x-RTLINUX	C.20xx.34
LynxOS-Driver	V1.2.x	C.20xx.31
PowerMAX OS-Driver	V1.1.1	P.140x.64
Solaris-Driver	V1.2.x	-
SGI-IRIX6.5-Driver	V2.1.x	C.20xx.27
AIX-Driver	V1.3.0	C.20xx.28
VxWorks-Driver	V1.4.x	C.20xx.55
QNX4-Driver	V1.0.0	C.20xx.32
RTOS-UH-Driver	V1.0.x	-

Implementiert auf folgenden CAN-Modulen	**Bestellnummer**
CAN-Bluetooth	C.2065.xx
CAN-ISA/200	C.2011.xx
CAN-ISA/331	C.2010.xx
CAN-PC104/200 (nur SJA1000)	C.2013.xx
CAN-PC104/331	C.2012.xx
CAN-PCI/200	C.2021.xx
CAN-PCI/331	C.2020.xx
CAN-PCI/360	C.2022.xx
PMC-CAN/331	C.2025.xx
CPCI-CAN/331	C.2027.xx
CPCI-CAN/360	C.2026.xx
CAN-PCC	C.2422.xx
CAN-USB-Mini	C.2464.xx
VME-CAN2	V.1405.xx
VME-CAN4	V.1408.xx

Änderungen in der Software und/oder der Dokumentation

Änderung in diesem Handbuch gegenüber der Vorversion	Änderung in der Software	Änderung in der Dokumentation
Neues Kapitel ‘3.3 Objektmodus’	x	x
Neues Kapitel ‘3.4 Senden und Empfangen von CAN 2.0B (29-Bit) Nachrichten’ mit Übersichtstabelle zu 29-Bit-Unterstützung	x	x
Neues Bit ‘No Data’ in CMSG-Datenstruktur (Kap. 4.1)	x	x
Parameter *mode* von *canOpen* beschrieben (Kap. 4.1)	x	x
canTake- und *canRead*-Beschreibung überarbeitet (Kap. 4.2)	-	x
Datenstruktur der CAN-Status-Message erweitert (Kap. 4.4)	x	x
Ausführliche Beschreibung der Rückgabewerte eingefügt (Kap. 5.)	x	x
Installationshinweise zu neu unterstützten Betriebssystemen (Windows 2000/ME/XP, Linux-RTAI, Linux-RT, PowerMAX OS) eingefügt **(Eine Übersicht finden Sie auf den Seiten A-4 und A-5.)**	x	x
Installationshinweise zu neuen CAN-Modulen (CAN-USB-Mini, CAN-Bluetooth, VME-CAN4) eingefügt **(Eine Übersicht finden Sie auf den Seiten A-4 und A-5.)**	x	x

Der Inhalt dieses Handbuches wurde mit größter Sorgfalt erarbeitet und geprüft. **esd** übernimmt jedoch keine Verantwortung für Schäden, die aus Fehlern in der Dokumentation resultieren könnten. Insbesondere Beschreibungen und technische Daten sind keine zugesicherten Eigenschaften im rechtlichen Sinne.

esd hat das Recht, Änderungen am beschriebenen Produkt oder an der Dokumentation ohne vorherige Ankündigung vorzunehmen, wenn sie aus Gründen der Zuverlässigkeit oder Qualitätssicherung vorgenommen werden oder dem technischen Fortschritt dienen.

esd electronic system design gmbh
Vahrenwalder Str. 207
30165 Hannover

Tel.: 0511/372 98-0
FAX : 0511/372 98-68
E-Mail: info@esd.electronics.com
Internet: www.esd-electronics.com

Inhalt

Seite

Diese Seite ist bewußt unbedruckt.

1. Einleitung

Dieses Handbuch beschreibt die Installation und Funktionsweise der Gerätetreiber für esd-CAN-Module sowie den Einsatz der Standard-Programmierschnittstelle NTCAN-API in eigenen Applikationen. Zusätzlich werden die Hilfsprogramme, die zum Test des Gerätetreibers und zur Überwachung des CAN-Busses dienen, erläutert.

Das Handbuch besitzt folgenden Aufbau:

Kapitel 1: *Einleitung* - gibt einen Überblick über dieses Handbuch.

Kapitel 2: *Gerätetreiber und NTCAN-API* - gibt einen Überblick über die Möglichkeiten des Gerätetreibers und der NTCAN-API.

Kapitel 3: *Die NTCAN-API* - beschreibt, wie die NTCAN-API in eigenen Applikationen zur Realisierung einer Kommunikation auf dem CAN-Bus genutzt werden kann.

Kapitel 4: *Programmierschnittstelle* - ist die Referenz der NTCAN-API.

Kapitel 5: *Rückgabewerte* - beschreibt die Rückgabewerte der NTCAN-API Funktionsaufrufe im Fehlerfall.

Kapitel 6: *Beispielprogramme* - enthält zwei komplette kleine Beispiel-Applikationen, die das Senden und das Empfangen von CAN-Nachrichten veranschaulichen.

Kapitel 7: *Testprogramm cantest* - beschreibt die Beispielapplikation `cantest`, die jedem Gerätetreiber im Quelltext beigefügt ist.

Kapitel 8: *CANscope* - beschreibt das CAN-Bus Monitorprogramm CANscope, das Bestandteil des CAN-SDK für Windows 9x/NT ist.

Kapitel 9: *CANbatch* - beschreibt, wie das Hilfsprogramm CANbatch, das Bestandteil des CAN-SDK für Windows 9x/NT ist, zur Automatisierung von z.B. Initialisierungen auf dem CAN-Bus genutzt werden kann.

Anhang: *Installation und Implementation* - beschreibt detailliert, wie die Gerätetreiber unter den verschiedenen Betriebssystemen installiert werden und ein Firmware-Update durchgeführt werden kann.
Am Anfang dieses Kapitels finden Sie einen Wegweiser, der Sie zu den Abschnitten führt, die für Ihr CAN-Modul und für das eingesetzte Betriebssystem wichtig sind.

Grundlegende und weiterführende Informationen zum CAN-Bus können Sie z.B. den folgenden Veröffentlichungen entnehmen:

ISO 11898	Road vehicles -- Interchange of digital information -- Controller area network (CAN) for high-speed communication
CAN Specification 2.0	Robert Bosch GmbH, 1991

2. Gerätetreiber und NTCAN-API

2.1 Übersicht

Dieses Kapitel soll einen Überblick über die Möglichkeiten der esd-CAN-Gerätetreiber und des darauf basierenden NTCAN Application Programming Layers (API) geben. Die NTCAN-API ist unabhängig von der verwendeten Hardware und stellt die gleiche Funktionalität auf den nachfolgenden esd-CAN-Modulen zur Verfügung:

- CAN-ISA/200, CAN-PC104/200, CAN-PCI/200
- CAN-ISA/331, CAN-PC104/331, CAN-PCI/331, CPCI-CAN/331, PMC-CAN/331
- CAN-PCI/360, CPCI-CAN/360
- CAN-PCC
- CAN-USB-Mini
- CAN-Bluetooth

Als Kommunikationsschicht zwischen Applikation und Gerätetreiber entsprechend der folgenden Abbildung ist die NTCAN-API auf den folgenden Desktop-, Embedded-, RealTime- und UNIX-Betriebssystemen implementiert.

- Windows NT, Windows 2000, Windows XP
- Windows 95, Windows 98, Windows ME
- Linux, Linux-RTAI, LynxOS, PowerMAX OS, Solaris, SGI-Unix
- QNX/Neutrino, VxWorks, AIX, RTOS-UH

Hinweis: Die C-Schnittstelle ist für verschiedene Betriebssysteme auf verschiedenen esd-CAN-Modulen implementiert. Sollten Funktionen der Schnittstelle nur eingeschränkt nutzbar sein, so ist dies an entsprechender Stelle vermerkt.

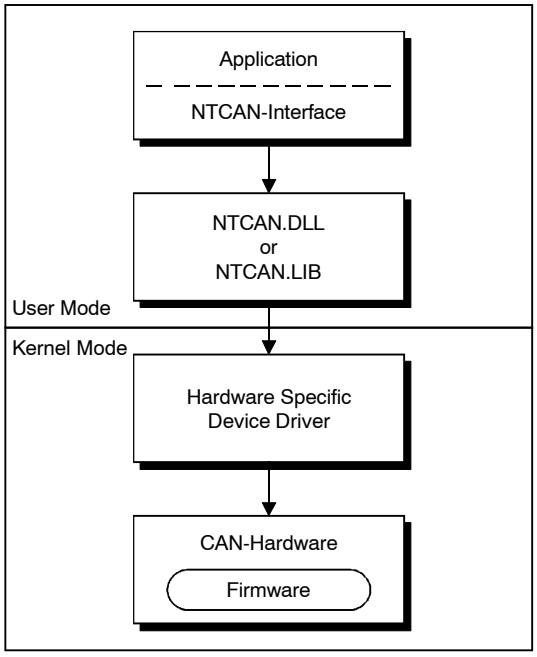


Abb. 2.1.1: Einbindung der NTCAN-API in die Systemstruktur

2.2 Eigenschaften von Gerätetreibern und der NTCAN-API

2.2.1 Betriebssystem-Integration

Die Gerätetreiber nutzen stets die vom Betriebssystem zur Verfügung gestellte Schnittstelle zur Integration neuer Hardware. Dadurch wird erreicht, daß der CAN-Treiber wie die Treiber anderer Geräte im Betriebssystem integriert ist. Außerdem bleiben so die Mechanismen zur Verhinderung von Zugriffsverletzungen, die bei vielen Betriebssystemen die Kern- von der Applikationsebene und die Applikationen untereinander trennen, erhalten.
Die NTCAN-API nutzt stets die vom Betriebssystem zur Verfügung gestellten Mechanismen zur Kommunikation mit dem Geräte-Treiber entsprechend der vorangegangenen Abbildung, so daß auch Zwischenschichten mit einer anderen Funktionalität und API parallel zur NTCAN-API genutzt werden können.

2.2.2 Plug&Play Unterstützung

Zur Vereinfachung der Treiber-Installation bei PCI-Karten werden entweder die Mechanismen der Plug&Play fähigen Betriebssysteme genutzt oder das Plug&Play vom Treiber nachgebildet, sofern zuvor z.B. das BIOS eine Plug&Play Konfiguration durchgeführt hat.

2.2.3 Multitasking/Multithreading Unterstützung

Alle Treiber sind in der Lage die CAN-Hardware zu virtualisieren, so daß mehrere Applikationen (Prozesse, Tasks) und/oder mehrere Threads einer Task die CAN-Hardware gleichzeitig nutzen können.

2.2.4 Interaction

Wird eine CAN-Nachricht auf einem bestimmten CAN-Identifier auf einem bestimmten CAN-Bus von einem Prozeß gesendet, so werden diese Nachrichten auch von anderen Prozessen empfangen, die auf CAN-Nachrichten mit diesem CAN-Identifier auf dem gleichen physikalischen CAN-Bus warten. Dieses Verhalten ist in der folgenden Abbildung dargestellt und wird als *Interaction* bezeichnet und ermöglicht z.B. einen CANopen Master und einen CANopen-Slave auf einem Rechner mit nur einer CAN-Schnittstelle zu betreiben.

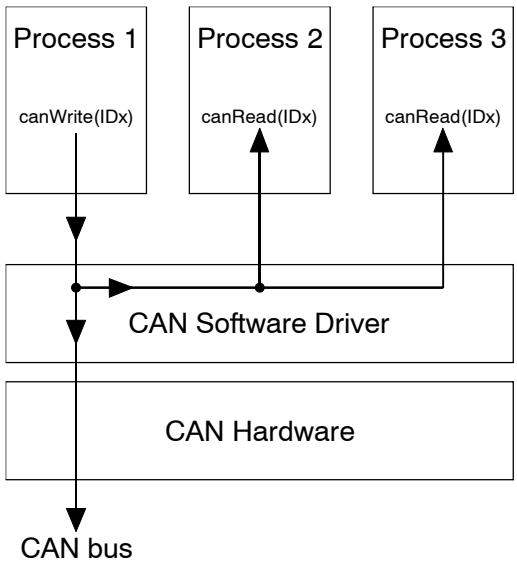


Abb. 2.2.1: Interaction-Beispiel

Die Weiterleitung der Nachrichten erfolgt dabei nicht über Interprozess-Mechanismen des jeweiligen Betriebssystems, sondern wird vom Treiber mit dem erfolgreichen Senden der Nachricht auf dem CAN-Bus verknüpft. Dies hat den Vorteil, daß lokale Empfänger die Daten zum gleichen Zeitpunkt empfangen wie andere Teilnehmer am CAN-Bus, bedeutet aber auch das das *Interaction* jedoch nur funktioniert, wenn ein anderer CAN-Knoten den Empfang der Nachricht bestätigt.

2.2.5 Multiprozessor Unterstützung

Treiber für Betriebssysteme, die mehr als einen Prozessor unterstützen, wurden mit dem Ziel entwickelt, daß sie auch in einer Mehrprozessorumgebung funktionieren.

2.2.6 Bus-Off-Behandlung im Hintergrund

Werden von einem CAN-Controller zu viele Error-Frames empfangen, wechselt er in den Zustand 'Bus-Off' und nimmt nicht mehr an der Kommunikation über den CAN-Bus teil. Der Gerätetreiber bei passiven bzw. die Firmware bei aktiven CAN-Modulen versucht automatisch nach einer gewissen Zeit den CAN Controller zu re-initialisieren und wieder an der Kommunikation über den Bus teilzunehmen ohne das die Applikation hierfür besondere Maßnahmen ergreifen muß.

2.2.7 Firmware-Update

Alle Treiber für aktive CAN-Module mit eigenem Mikroprozessor besitzen eine eigene Firmware. Es ist mit den meisten Gerätetreibern möglich, diese Firmware zu aktualisieren ohne die CAN-Hardware unter einem anderen Betriebssystem zu betreiben.

2.2.8 Hardware-Unabhängigkeit

Ein Gerätetreiber unterstützt den gleichzeitigen Betrieb von bis zu 5 CAN-Modulen, wobei je nach eingesetzter CAN-Hardware ein einzelnes Modul bis zu 4 physikalische Netze zur Verfügung stellt. Mit 5 CAN-PCI/360 Karten könnte ein einzelner PC daher gleichzeitig auf 20 CAN-Netze zugreifen.

Darüber hinaus ist bei vielen Betriebssystemen der gleichzeitige Betrieb von Gerätetreibern für unterschiedliche esd-CAN-Module möglich. Beim Start des Gerätetreibers werden den physikalischen Netzen logische Netznummern zugeordnet, mit deren Hilfe aus der Applikation über die NTCAN-API auf die einzelnen Netze zugegriffen werden kann. Mit Hilfe der Netznummern kann hardware-unabhängig auf den CAN-Bus zugegriffen werden, so daß ein Wechsel von einem esd-CAN-Modul-Typ auf einen anderen problemlos möglich ist, ohne die Applikation zu beeinflussen.

2.2.9 Betriebssystem-Unabhängigkeit

Aufgrund der weitgehend Betriebssystem-unabhängigen Definition der NTCAN-API sind auf dieser Basis entwickelte Applikationen bezüglich der CAN-Kommunikation weitgehend portabel zwischen verschiedenen Betriebssystemen.

3. CAN-Kommunikation mit der NTCAN-API

Dieses Kapitel soll einen Überblick über Funktionsweise der CAN-Kommunikation mit der NTCAN-API verschaffen, bevor die einzelnen Aufrufe in der API-Referenz des nachfolgenden Kapitels detaillierter beschrieben werden.

3.1 Grundlagen der CAN-Kommunikation

Eine Applikation, die auf den CAN-Bus zugreifen möchte, muß eine logische Verbindung zum Gerätetreiber öffnen, die durch ein Handle repräsentiert wird. Dieses Handle muß bei allen nachfolgenden Funktionsaufrufen übergeben werden. Es ist möglich, daß gleichzeitig verschiedene Prozesse logische Verbindungen zu dem Gerätetreiber nutzen oder ein Prozeß mehrere logische Verbindungen besitzt.

Jedes Handle ist über die beim Öffnen übergebene logische Netznummer eindeutig einer physikalischen Schnittstelle und dem daran angeschlossenen CAN-Bus zugeordnet. Jedes Handle besitzt individuell konfigurierbare Größen für Empfangs- und Sende-Buffer sowie für Empfangs- und Sende-Timeout. Zur applikations-spezifischen Selektion der empfangenen Nachrichten auf Basis der CAN-Identifier kann für jedes Handle ein individueller Filter konfiguriert werden. Die CAN-Identifier der zu sendenden Nachrichten müssen dem Gerätetreiber zuvor nicht explizit bekannt gemacht werden.

Bevor Daten gesendet oder empfangen werden können, muß die Baudrate der CAN-Schnittstelle einmalig initialisiert werden. Diese Initialisierung ist dann systemweit für diese logischen Netznummer und somit für die zugeordnete physikalische Schnittstelle gültig. Um zu verhindern, daß zwei Applikationen die gleiche Schnittstelle mit unterschiedlichen Baudraten betreiben wollen, kann zuvor abgefragt werden, ob die Baudrate bereits auf einen anderen Wert initialisiert wurde und entsprechend verfahren werden.

Zum Senden von Nachrichten stellt die API blockierende und nicht-blockierende Aufrufe zur Verfügung. Dies ermöglicht es dem aufrufenden Prozeß oder Thread entweder zu *blockieren* bis die Nachricht erfolgreich auf dem CAN-Bus übertragen bzw. die dem Handle zugeordnete Timeout-Zeit überschritten wurde oder sofort *zurückzukehren* und den Gerätetreiber den Nachrichtenversand asynchron zum Aufrufer ausführen zu lassen

Zum Empfangen von Nachrichten stell die API blockierende und nicht-blockierende Aufrufe zur Verfügung. Dies ermöglicht dem Aufrufer entweder bei Bedarf zu überprüfen, ob neue Daten in dem Empfangsbuffer verfügbar sind (*Polling*), oder zu *blockieren*, bis eine oder mehrere Nachrichten, die den Filterkriterien dieses Handles entsprechen, empfangen werden bzw. die dem Handle zugeordnete Timeout-Zeit überschritten wurde.

Jeder API-Aufruf liefert im Falle eines Fehlers einen entsprechenden Fehlercode zurück. Die einzelnen Fehlercodes werden ab Seite 37 näher beschrieben.

3.2 Senden und Empfangen

Die CAN-Kommunikation mit der NTCAN-API basiert auf sogenannten Nachrichten-Warteschlangen (Message Queues), die vielfach auch als FIFO (First-In-First-Out) Speicher bezeichnet werden, in denen sich Sequenzen von CAN-Nachrichten befinden.

Jedem Handle ist eine Empfangs- und eine Sende-FIFO zugeordnet, deren Größe beim Öffnen des Handles festgelegt werden kann. In der Empfangs-FIFO werden CAN-Nachrichten in der zeitlichen Reihenfolge des Eintreffens abgelegt. Durch Aufruf einer Leseoperation (canRead() oder canTake()) werden eine oder mehrere CAN-Nachrichten aus der Handle-FIFO in den Adreßraum der Applikation kopiert. Durch Aufruf einer Schreiboperation (canWrite() oder canSend()) werden eine oder mehrere CAN-Nachrichten aus dem Adreßraum der Applikation in die Sende-FIFO kopiert und die CAN-Nachrichten werden unter Einhaltung der Reihenfolge auf dem CAN-Bus übertragen.

Ein canRead()-Aufruf kehrt so lange sofort mit neuen Daten zurück, solange die Empfangs-FIFO noch nicht vollständig leer ist. Ist die FIFO leer, so blockiert der aufrufende Thread entweder bis neue Daten verfügbar sind oder bis innerhalb des Empfangs-Timeout keine neuen CAN-Nachrichten für dieses Handle eingetroffen sind. Ein canTake()-Aufruf kehrt immer sofort zurück, bei einer leeren FIFO jedoch ohne Daten in den Adreßraum der Applikation zu kopieren. Läuft die Empfangs-FIFO über, weil die Applikation die CAN-Nachrichten nicht schnell genug abholen kann, werden die ältesten Daten überschrieben und der Verlust von Daten angezeigt.

Ein canWrite()-Aufruf blockiert der aufrufende Thread so lange, bis alle CAN-Nachrichten erfolgreich übertragen worden sind oder innerhalb des Sende-Timeouts eine Nachricht nicht auf den CAN-Bus übertragen werden konnte. Reicht der Platz in der Sende-FIFO nicht aus, um alle Daten der Applikation aufzunehmen, erfolgt die Aufteilung in Blöcke, die der FIFO-Größe entsprechen, automatisch. Ein canSend()-Aufruf kehrt immer sofort zurück und die CAN-Nachrichten werden asynchron zum aufrufenden Thread übertragen. Ist der Platz für weitere CAN-Nachrichten in der Sende-FIFO erschöpft, teilt der canSend()-Aufruf die Daten nicht automatisch in Blöcke auf, sondern liefert eine entsprechende Fehlermeldung zurück.

Soll auf die Message Queues verzichtet werden, so kann deren Länge auf ‘1’ gesetzt werden. Für den Empfang bedeutet dies, daß sich in der Empfangs-FIFO stets die zuletzt empfangene CAN-Nachricht befindet.

3.3 Objektmodus (Empfangen)

Neben dem zuvor beschriebenen Empfang der Daten über eine FIFO, in der jede CAN-Nachricht in der Reihenfolge des Eintreffens abgelegt und von der Applikation ausgewertet wird (Monitor-Modus), ist es in einigen Anwendungen lediglich notwendig, daß die Applikation bei Bedarf die zuletzt empfangenen und somit aktuellen Daten einer CAN-Nachricht auswertet. Zu diesem Zweck unterstützen einige Treiber einen Objektmodus, der für jedes Handle individuell beim Öffnen statt des Default-Verhaltens (Monitor-Modus) aktiviert werden kann. Das Senden von CAN-Nachrichten mit diesem Handle über eine FIFO und die Möglichkeiten der Konfiguration eines Empfangsfilters werden von der Wahl des Empfangs-Modus nicht beeinflußt.

Ist für ein Handle der Objektmodus aktiviert, so ist für den Empfang von Daten lediglich ein Aufruf des nicht-blockierenden canTake() möglich. Ein Aufruf von canRead() kehrt mit einem Fehler zurück.

Im Gegensatz zu einem Aufruf von canTake() im FIFO-Modus müssen in der übergebenen Datenstruktur die CAN-Identifier in der entsprechenden Datenstruktur schon beim Aufruf initialisiert sein, da der Gerätetreiber anhand dieser Information feststellt, an welchen CAN-Nachrichten die Applikation interessiert ist. Die Auswahl und Reihenfolge der Nachrichten kann bei jedem Aufruf von canTake() von der Applikation neu festgelegt werden, muß jedoch mit der Konfiguration des Nachrichtenfilters übereinstimmen.

Ob ein bestimmter Treiber den Objektmodus unterstützt, kann mit Hilfe des canStatus() Aufrufes festgestellt werden.

3.4 Senden und Empfang von CAN 2.0B (29-Bit) Nachrichten

Die meisten esd CAN-Module unterstützen auch das Senden und Empfangen von CAN-Nachrichten mit 29-Bit Identifiern gemäß dem CAN 2.0B-Standard. Folgende Einschränkungen sind dabei zu beachten: Der zuvor beschriebenen Objektmodus kann nur für 11-Bit Identifier genutzt werden. Im FIFO-Modus ist die Nachrichtenfilterung basierend auf CAN-Identifier nur für 11-Bit Identifier möglich. Wird für ein Handle im FIFO-Modus der erste 29-Bit Identifier aktiviert, so werden ab diesem Zeitpunkt alle empfangenen CAN-Nachrichten mit 29-Bit Identifiern in der Empfangs-FIFO dieses Handles abgelegt.

Zur Zeit unterstützen die folgenden esd-CAN-Module 29-Bit-Identifier (Stand 02/2002):

CAN-Modul	Bestell-nummer	Windows 95	Windows 98/ME	Windows NT	Windows 2000/XP	Linux	Linux-RTAI	Linux-RT	LynxOS	PowerMAX OS	Solaris	SGI-IRIX6.5	AIX	VxWorks	QNX4	RTOS-UH
CAN-Bluetooth	C.2065.xx	-	☻	-	☻	☻	☻	☻	-	-	-	-	-	-	-	-
CAN-USB-Mini	C.2422.xx	-	☻	-	☻	☻	-	-	-	-	-	-	-	-	-	-
CAN-PCC	C.2422.xx	☹	☹	☹	☹x	-	-	-	-	-	-	-	-	-	-	-
CAN-ISA/200 *)	C.2011.xx	☻	☻	☹	☹x	☻	☻	☻	-	-	-	-	-	☻	☹	-
CAN-ISA/331	C.2010.xx	☻	☻	☻	☻x	☻	☻	☻	☻	-	☹	-	-	☻	☹	-
CAN-PC104/200 (nur SJA1000) *)	C.2013.xx	☻	☻	☹	☹x	☻	☻	☻	☻	-	-	-	-	☻	☹	-
CAN-PC104/331	C.2012.xx	☻	☻	☻	☻x	☻	☻	☻	☻	-	☹	-	-	☻	☹	-
CAN-PCI/200	C.2021.xx	☻	☻	☻	☻	☻	☻	☻	-	-	-	-	-	-	-	-
CAN-PCI/331	C.2020.xx	☻	☻	☻	☻	☻	☻	☻	☻	-	☹	☹	☹	☻	☹	-
CAN-PCI/360	C.2022.xx	☻	☻	☻	☻	☻	☻	☻	-	-	-	-	-	-	-	-
PMC-CAN/331	C.2025.xx	☻	☻	☻	☻	☻	☻	☻	☻	-	☹	☹	☹	☻	☹	☻
CPCI-CAN/331	C.2027.xx	☻	☻	☻	☻	☻	☻	☻	☻	-	☹	☹	☹	☻	☹	-
CPCI-CAN/360	C.2026.xx	☻	☻	☻	☻	☻	☻	☻	-	-	-	-	-	-	-	-
VME-CAN2	V.1405.xx	-	-	-	-	-	-	-	☻	☻	-	-	-	-	-	☻
VME-CAN4	V.1408.xx	-	-	-	-	-	-	-	☻	☻	-	-	-	-	-	☻

☻ ... 11-Bit- und 29-Bit-Identifier werden unterstützt
☹ ... nur 11-Bit-Identifier werden unterstützt
☻x, ☺x ... diese Module werden unter Windows 2000/XP mit dem Windows NT Treiber betrieben
- ... keine Unterstützung durch dieses Betriebssystem

*) Für die Module CAN-ISA/200 und CAN-PC104/200 (SJA1000) steht für Windows 9x/ME ein Treiber für 11-Bit-IDs und ein Treiber für 29-Bit-IDs zur Verfügung. Während der Installation kann über den Hardware-Wizard der gewünschte Treiber ausgewählt werden. Für Windows 2000/XP ist für diese Module kein 29-Bit-ID-Treiber verfügbar. Module, die noch mit dem CAN-Controller 82C200 bestückt sind (bis ca. 12/1999 möglich) unterstützen generell keine 29-Bit-Identifier.

Tabelle 3.4.1: 29-Bit-Identifier-Unterstützung bei CAN-Modulen

3.5 Event-Schnittstelle

Neben den Rückgabewerten, die der Applikation Fehler bei einem Funktionsaufruf signalisieren, kann der Gerätetreiber über Events signalisieren, daß ein globaler Fehler wie z.B. Bus-Off aufgetreten ist.

4. Programmierschnittstelle

Im nachfolgenden Kapitel wird die C-Programmierschnittstelle zum CAN-Controller beschrieben. Die Bedeutung der Rückgabewerte im Fehlerfall ist in einem eigenen Kapitel ab Seite 37 beschrieben.

4.1 Parametrierung

Datenstruktur der CAN-Messages

```
typedef struct
{
    long           id;        /* 11-, bzw. 29-Bit-CAN-Identifier          [in, out]   */
    unsigned char  len;       /* Bit 0-3 = Anzahl Daten-Bytes 0...8       [in, out]   */
                              /* Bit 4   = RTR                            [in, out]   */
                              /* Bit 5   = No_Data (Monitor-Modus)        [in     ]   */
                              /* Bit 6-7 = reserviert                                 */
    unsigned char  msg_lost;  /* Zähler für verlorene Rx-Nachrichten      [out]       */
    unsigned char  reserved[2]; /* reserved                                           */
    unsigned char  data[8];   /* 8 Daten-Bytes                            [in, out]   */
}   CMSG;
```

Wert von *len* [binär] bit7... ...bit0	Anzahl der Daten-Bytes [Bytes]
xxxx 0000	0
xxxx 0001	1
xxxx 0010	2
xxxx 0011	3
xxxx 0100	4
xxxx 0101	5
xxxx 0110	6
xxxx 0111	7
xxxx 1000	8

Tabelle 4.1.1: Kodierung der Länge

Um eine Nachricht mit einem 29-Bit-Identifier zu senden, muß das Bit 29 des CAN-Identifier-Parameters *id* in der Struktur CMSG zusätzlich zum CAN-Identifier gesetzt werden. Dies kann mit einer logischen ODER-Verknüpfung mit NTCAN_20B_BASE (definiert im Header 'ntcan.h'), erreicht werden.

Wert des Bits '*RTR*' [binär]	Funktion
0	kein RTR senden
1	RTR-bit wird gesendet

Tabelle 4.1.2: Funktion des Bits '*RTR*'

Wert des Bits '*No_Data*' [binär]	Funktion (nur Monitor-Modus)
0	Empfangene Daten gültig
1	Empfangene Daten ungültig

Tabelle 4.1.3: Funktion des Bits '*No_Data*'

Die Bits 6...7 sind für zukünftige Anwendungen reserviert. Bei Leseoperationen ist der Zustand dieser Bits undefiniert. Bei Schreiboperationen sollten diese Bits und auch Bit 5 immer auf '0' gesetzt werden.

Ist die Receive-FIFO des Handles voll und neue Nachrichten treffen ein, so werden alte Nachrichten überschrieben und der *msg_lost*-Zähler wird hochgezählt.

Der Anwender kann mit Hilfe des *msg_lost*-Zählers einen Datenüberlauf erkennen. Ein ansteigender Zählerwert signalisiert, daß das Anwendungsprogramm den CAN-Datenstrom langsamer liest als ihn der Sender auf dem CAN-Bus sendet.

Wert des Message-Lost-Zählers	Bedeutung
$msg\_lost = 0$	no lost messages
$0 < msg\_lost < 255$	number of lost frames = value of msg-lost
$msg\_lost = 255$	number of lost frames ≥ 255

Tabelle 4.1.4: Wertebereich von *msg_lost*

canOpen()

Name: **canOpen()** - Erzeugen eines Handles für Lese- und Schreiboperationen

Aufruf:

```
DWORD  canOpen
       (
       int              net,          /* Netz-Nummer */
       unsigned long    mode,
       long             txqueuesize,  /* Anzahl der Entries in der Message-Queue */
       long             rxqueuesize,  /* Anzahl der Entries in der Message-Queue */
       long             txtimeout,    /* Tx-Timeout in ms */
       long             rxtimeout,    /* Rx-Timeout in ms */
       HANDLE           *handle       /* Out: Handle */
       )
```

Beschreibung: Diese Funktion gibt ein Handle für nachfolgende I/O-Aufrufe zurück. Die Anzahl der möglichen Handles ist durch den Treiber zur Zeit auf 1024 limitiert. Bitte beachten Sie, daß die maximale Anzahl der Handles in Abhängigkeit vom eingesetzten Betriebssystem außerdem durch Prozeßgrenzen und systemglobale Grenzen eingeschränkt sein kann!

canOpen ist als **erster** Funktionsaufruf auszuführen, da alle anderen Funktionen das *Handle* benötigen !

Mit *net* wird die logische Netznummer übergeben, der dieses Handle zugeordnet werden soll

Der Parameter *mode* legt spezielle Eigenschaften des Handles fest. Folgende Flags sind z.Z. im Header *ntcan.h* definiert.

Flag	Beschreibung
`NTCAN_MODE_OVERLAPPED`	**Nur für Windows-Betriebssysteme:** Dieses Flag öffnet das Handle für Overlapped-I/O-Operationen. Falls dieses Flag gesetzt ist, sind mit diesem Handle nur noch Overlapped-I/O-Operationen möglich! D.h. die Overlapped-Parameter bei canRead() und canWrite() müssen versorgt werden.
`NTCAN_MODE_OBJECT`	Dieses Flag öffnet das Handle für den Empfang im Objektmodus anstatt des FIFO-Modus. Für den Empfang ist dann der Aufruf von canRead() nicht mehr möglich.

Tabelle 4.1.5: Flags für spezielle Handle-Eigenschaften im Header *ntcan.h*

txqueuesize setzt die Anzahl der CAN-Tx-Message-Strukturen für dieses Handle. Der absolute Maximalwert für *txqueuesize* ist in der Header-Datei *ntcan.h* über NTCAN_MAX_TX_QUEUESIZE definiert (zur Zeit 16383).

rxqueuesize setzt die Anzahl der CAN-Rx-Message-Strukturen für dieses Handle. Der absolute Maximalwert für *txqueuesize* ist in der Header-Datei *ntcan.h* über NTCAN_MAX_RX_QUEUESIZE definiert (zur Zeit 16383).

txtimeout definiert das Timeout-Intervall in ms für Schreibzugriffe mit Timeout. Bei canWrite()-Aufrufen muß die Tx-Nachricht innerhalb der Zeit *txtimeout* gesendet werden.

txtimeout = 0 (kein Timeout, Funktion wartet endlos)
1...65535 (0xFFFF hex) (Timeout in ms)

rxtimeout definiert das Timeout-Intervall in ms für Lesezugriffe mit Timeout. Die Funktion canRead() wird mit einem Timeout-Fehler beendet, wenn nicht mindestens eine Rx-Nachricht innerhalb der über *rxtimeout* definierten Zeit empfangen wird.

rxtimeout = 0 (kein Timeout, Funktion wartet endlos)
1...65535 (0xFFFF hex) (Timeout in ms)

Gibt die Funktion NTCAN_SUCCESS zurück, so wird in *handle* das CAN-Handle geschrieben. In Abhängigkeit vom eingesetzten CAN-Controller können bis zu 1024 Handles geöffnet werden.

Rückgabe: Nach erfolgreicher Durchführung wird NTCAN_SUCCESS zurückgegeben, ansonsten einer der ab Seite 37 aufgeführten Fehler-Codes.

Header: ntcan.h

---

canClose()

Name: **canClose()** - Schließen eines Handles für Lese- und Schreibzugriffe

Aufruf:

```
DWORD  canClose
       (
       HANDLE      handle          /* CAN-Handle */
       )
```

Beschreibung: Diese Funktion gibt ein CAN-Handle und alle belegten Ressourcen frei.

Rückgabe: Nach erfolgreicher Durchführung wird NTCAN_SUCCESS zurückgegeben, ansonsten einer der ab Seite 37 aufgeführten Fehler-Codes.

Header: ntcan.h

canSetBaudrate()

Name: **canSetBaudrate()** - Initialisierung des CAN-Interfaces

Aufruf:

```
DWORD   canSetBaudrate
        (
        HANDLE      handle,     /* CAN-Handle           */
        DWORD       baud        /* Baudraten-Konstante *//
        )
```

Beschreibung: Diese Funktion initialisiert das CAN-Interface und setzt die Baudrate *baud*. Ein Modul verhält sich passiv auf dem CAN-Bus, solange die Baudrate nicht wenigstens einmal gesetzt worden ist.

Werden für *baud* Werte von 0x00000000 (hex) bis 0x0000000D (hex) übergeben, so erfolgt die Einstellung der Baudrate entsprechend der folgenden Tabelle:

baud [hex]	Baudrate [kbit/s]
0	**1000**
1	666.6
2	**500**
3	333.3
4	**250**
5	166
6	**125**
7	100
8	66.6
9	**50**
A	33.3
B	**20**
C	12.5
D	**10**

Tabelle 4.1.6: Einstellung der Baudrate in 14 Stufen

Wird das hochwertigste Bit (Bit 31) des Langwortes des Parameters *baud* auf '1' gesetzt, so wird der übergebene Wert auf andere Weise ausgewertet. In diesem Fall wird bei Modulen mit den CAN-Controllern 82C200, SJA1000, 82527 (und allen anderen Controllern mit dieser Baudratenregisterstruktur) direkt der Registerwert für die Bit-Timing-Register BTR0 und BTR1 übergeben. Der Parameter *baud* wird dann gemäß der folgenden Tabelle ausgewertet.

Belegung der 32 Bits des Parameters *baud*, wenn Bit 31='1'			
31	30... ...16	15... ...8	7... ...0
1	keine Auswertung	BTR0	BTR1

Tabelle 4.1.7: Setzen der Register BTR0 und BTR1 über den Parameter *baud*

Die Berechnung des Bit-Timings und der Baudrate nach den Registerwerten kann den Handbüchern der Controller SJA1000 (Philips), 82C200 (Philips), oder 82527 (Intel) entnommen werden. Das Handbuch des SJA1000 kann z.B. aus dem Internet von der Philips Homepage unter folgender Adresse heruntergeladen werden: http://www-us7.semiconductors.philips.com/pip/SJA1000

Anmerkung:
Bitte beachten Sie, daß für die Funktionsfähigkeit eines CAN-Netzes nicht nur alle CAN-Teilnehmer die selbe Bitrate besitzen müssen, sondern auch die anderen Timing-Parameter identisch sein sollten! Die folgende Tabelle zeigt die von der CiA empfohlenen Werte:

Bit rate Bus length	nominal bit time t_B	Number of time quanta per bit	Length of time quantum t_q	Location of sample point	BTR 0 Setting at 16 MHz, i.e. SJA1000 [HEX]	BTR 1 Setting at 16 MHz, i.e. SJA1000 [HEX]
1 Mbit/s 25 m	1 µs	8	125 ns	6 tq (750 ns)	00	14
800 kbit/s 50 m	1,25 µs	10	125 ns	8 tq (1 µs)	00	16
500 kbit/s 100 m	2 µs	16	125 ns	14 tq (1,75 µs)	00	1C
250 kbit/s 250 m	4 µs	16	250 ns	14 tq (3,5 µs)	01	1C
125 kbit/s 500 m	8 µs	16	500 ns	14 tq (7 µs)	03	1C
100 kbit/s 650 m	10 µs	16	625 ns	14 tq (8,75 µs)	04	1C
50 kbit/s 1 km	20 µs	16	1,25 µs	14 tq (17,5 µs)	09	1C
20 kbit/s 2,5 km	50 µs	16	3,125 µs	14 tq (43,75 µs)	18	1C
10 kbit/s 5 km	100 µs	16	6,25 µs	14 tq (87,5 µs)	31	1C

Tabelle 4.1.8: Bit-Timing-Werte gemäß CiA-Empfehlung

Rückgabe: Nach erfolgreicher Durchführung wird NTCAN_SUCCESS zurückgegeben, ansonsten einer der ab Seite 37 aufgeführten Fehler-Codes.

Header: ntcan.h

canGetBaudrate()

Name: **canGetBaudrate()** - Ermittlung der eingestellten Baudrate

Aufruf:

```
DWORD  canGetBaudrate
       (
       HANDLE    handle,     /* CAN-Handle              */
       DWORD     *baud       /* zurückgegebene Baudrate */
       )
```

Beschreibung: Mit dieser Funktion kann die aktuelle Baudrate gelesen werden. Es gelten die gleichen Baudratenwerte und Formate wie bei *canSetBaudrate*. Der gelesene Wert kann entweder einem der 14 Werte aus der Baudratentabelle oder dem Inhalt der Bit-Timing-Register BTR0 und BTR1 entsprechen.
Außerdem läßt sich feststellen, ob die Karte bisher noch nicht initialisiert wurde: Wird der Wert 0x7FFFFFFF (hex) gelesen, so bedeutet dies, daß für diesen CAN-Knoten bisher keine Baudrateneinstellung erfolgt ist.

Rückgabe: Nach erfolgreicher Durchführung wird NTCAN_SUCCESS zurückgegeben, ansonsten einer der ab Seite 37 aufgeführten Fehler-Codes.

Header: ntcan.h

canIdAdd()

Name: **canIdAdd()** - Auswahl eines Rx-Identifiers oder eines Event-IDs für den Empfang von Daten

Aufruf:

```
DWORD  canIdAdd
       (
       HANDLE    handle,     /* Read-Handle */
       long      id          /* Rx-CAN-Identifier oder Event-ID */
       )
```

Beschreibung: Diese Funktion selektiert einen Rx-Identifier für ein geöffnetes Handle. Wird kein Rx-Identifier ausgewählt, so werden auch keine Daten empfangen!
Der selbe ID kann für maximal 15 verschiedene Handles pro physikalischem CAN-Knoten selektiert werden.

Ein id-Wert wird als 29-Bit-Identifier ausgewertet, wenn das Bit 29 auf '1' gesetzt ist (Identifier-Wert in Bit 0...28). Das CAN-Modul kann alle eintreffenden Nachrichten mit 29-Bit-Rx-Identifier empfangen, sobald ein einziger 29-Bit-Rx-Identifier definiert ist.

id = 0...07FF (hex) (11-Bit-Rx-Identifiers)
0...2047 (dez)

id = 0x20000000...0x3FFF.FFFF (hex) (29-Bit-Rx-Identifiers)

id = *event-id* (im Falle eines Event-IDs)

event-id = 0x40000000 (hex) ... 0x400000FF (hex)
NTCAN_EV_BASE ... NTCAN_EV_LAST

Hinweis:
Eine Übersicht der Kombinationen von CAN-Modulen und Betriebssystemen, die 29-Bit-CAN-Identifier unterstützen, finden Sie auf Seite 13.

Rückgabe: Nach erfolgreicher Durchführung wird NTCAN_SUCCESS zurückgegeben, ansonsten einer der ab Seite 37 aufgeführten Fehler-Codes.

Header: ntcan.h

canIdDelete()

Name: **canIdDelete()** - Löschen eines Rx-Identifiers oder eines Event-IDs

Aufruf:

```
DWORD   canIdDelete
        (
        HANDLE      handle,     /* Read-Handle */
        long        id          /* Rx-Identifier oder Event-ID */
        )
```

Beschreibung: Diese Funktion löscht den selektierten Rx-Identifier für dieses Handle.

id = 0...07FF (hex) (11-Bit-Rx-Identifier)
0...2047 (dez)

id = 0x20000000...0x3FFF.FFFF (hex) (29-Bit-Rx-Identifier)

id = *event-id* (im Falle eines Event-IDs)

event-id = 0x40000000 (hex) ... 0x400000FF (hex)
NTCAN_EV_BASE ... NTCAN_EV_LAST

Rückgabe: Nach erfolgreicher Durchführung wird NTCAN_SUCCESS zurückgegeben, ansonsten einer der ab Seite 37 aufgeführten Fehler-Codes.

Header: ntcan.h

4.2 Lesen und Schreiben von Daten

canTake()

Name: **canTake()** - Nicht-blockierendes Lesen der aktuellen Rx-Daten

Aufruf:

```
DWORD  canTake
       (
       HANDLE   handle,   /* Handle */
       CMSG     *cmsg,    /* Pointer auf Daten-Buffer */
       long     *len      /* bei Aufruf:      Out: Größe des CMSG-Buffers */
                          /* nach Rückkehr:   In:  Anzahl der gelesenen Nachrichten */
       )
```

Beschreibung: Diese Funktion gibt die für dieses Handle neu empfangenen Nachrichten zurück ohne zu blockieren.

FIFO-Modus
Wurde das Handle mit canOpen() im FIFO-Modus geöffnet (Default) werden die für dieses Handle bezüglich der aktiven Konfiguration des Nachrichtenfilters empfangenen CAN-Nachrichten in der Reihenfolge des zeitlichen Eintreffens in den durch *cmsg* definierten Adreßraum der Applikation kopiert.

Die Größe des CMSG-Buffers muß beim Aufruf von canTake() in *len* übergeben werden. Die Größe muß in Einheiten von Strukturen (eine Struktur enthält 16 Bytes, siehe Seite 14) angegeben werden. Nach dem Transfer wird die Anzahl der kopierten CAN-Nachrichtenstrukturen in *len* abgelegt. Wurden keine Daten empfangen, ist der zurückgegebene Wert für *len* '0'.

> **Achtung:** Bitte beachten Sie die Hinweise zum Lesen von CAN-Daten mit 29 Bit-Identifier, die bei der Funktion *canRead()* aufgeführt sind.

Objekt-Modus
Wurde das Handle mit canOpen() im Objekt-Modus geöffnet, müssen die id-Felder der CMSG-Strukturen zuvor mit den (11-Bit) Identifiern der gewünschten CAN-Nachrichten initialisiert werden. Die Größe des CMSG-Buffers muß beim Aufruf von canTake() in *len* übergeben werden.

Nach Rückkehr der Funktion enthalten die CMSG-Strukturen die Daten der zuletzt empfangenen CAN-Nachricht dieses Identifiers. Ist im *len*-Feld der einzelnen CMSG-Struktur das Bit 5 gesetzt, so wurden bisher noch keine Daten für diesen Identifier empfangen.

Rückgabe: Nach erfolgreicher Durchführung wird NTCAN_SUCCESS zurückgegeben, ansonsten einer der ab Seite 37 aufgeführten Fehler-Codes.

Header: ntcan.h

canRead()

Name: **canRead()** - Blockierendes Lesen der aktuellen Rx-Daten

Aufruf:

```
DWORD  canRead
       (
       HANDLE      handle,   /* Handle */
       CMSG        *cmsg,    /* Pointer auf Daten-Buffer */
       long        *len      /* bei Aufruf:     Out: Größe des CMSG-Buffers */
                             /* nach Rückkehr:  In:  Anzahl der gelesenen Nachrichten */
       OVERLAPPED  *ovrlppd  /* NULL oder Overlapped-Struktur */
       )
```

Beschreibung: Diese Funktion gibt die für dieses Handle neu empfangenen Nachrichten zurück. Wurden seit dem letzten Aufruf keine neuen Nachrichten empfangen, blockiert der Aufruf bis neue Nachrichten eintreffen oder das beim Öffnen des Handles mit canOpen() vereinbarte Rx-Timeout überschritten ist.

Wurde das Handle mit canOpen() im FIFO-Modus geöffnet (Default) werden die für dieses Handle bezüglich der aktiven Konfiguration des Nachrichtenfilters empfangenen CAN-Nachrichten in der Reihenfolge des zeitlichen Eintreffens in den durch *cmsg* definierten Adreßraum der Applikation kopiert.

Die maximale Größe des CMSG-Buffers muß beim Aufruf von canRead() in *len* definiert werden. Die Größe muß in Einheiten von Strukturen (eine Struktur enthält 16 Bytes, siehe Seite 14) angegeben werden. Nach dem Transfer wird die Anzahl der kopierten CAN-Nachrichtenstrukturen in *len* abgelegt. Wurden keine Daten empfangen, ist der zurückgegebene Wert für *len* '0'.

Wurde das Handle im Objekt-Modus geöffnet, kehrt der Aufruf sofort mit einem Fehler zurück.

Nur bei Windows-Betriebssystemen:
Werden keine Overlapped-Funktionen ausgeführt, und wird mindestens eine Rx-Nachricht empfangen, kehrt die Funktion sofort zurück.
Andernfalls kehrt die Funktion erst zurück, wenn das beim Aufruf der Funktion spezifizierte Timeout-Intervall überschritten wurde oder neue Daten innerhalb diese Intervalls empfangen worden sind.

Bei Overlapped-Operationen kann die Funktion canRead() wie eine asynchrone Windows-Funktion angesprochen werden. Hierbei kehrt die Funktion unabhängig davon, ob Daten empfangen wurden oder nicht, sofort zurück. Der Status der Übertragung (z.B. Timeout) kann zu einem späteren Zeitpunkt unter Verwendung von canGetOverlappedResult() (siehe Seite 29) ermittelt werden.
Für weitere Informationen über Windows-Overlapped-Strukturen wird auf das entsprechende Windows-Handbuch verwiesen.

Für alle anderen Betriebssysteme und das CAN-Bluetooth-Modul gilt:
Hier muß für *ovrlppd* immer NULL eingetragen werden.

29-Bit-Identifier
Um eine Nachricht mit einem 29-Bit-Identifier empfangen zu können, müssen Sie einen 2.0B-Identifier aktivieren. Dies erreichen Sie z.B. für den Identifier '0' durch Aufruf der Funktion *canIdAdd(handle, 0 | NTCAN_20B_BASE)*
Die Aktivierung eines einzelnen CAN 2.0B-Identifiers führt dazu, daß auf dem zugehörigen Handle alle CAN 2.0B-Identifier empfangen werden können.
Da auf den esd-CAN-PC-Modulen keine Rx-Filter zur Verfügung stehen, wie z.B. auf der VME-CAN2, muß zur Unterscheidung zwischen CAN 2.0A- und CAN 2.0B-Identifier die folgende Operation ausgeführt werden:

```
if(cmsg.id & NTCAN_20B_BASE)
    {
    /* 29-Bit ID received */
    id = cmsg.id & (NTCAN_20B_BASE-1);
    }
else
    {
    /* 11-Bit ID received */
    id = cmsg.id;
    }
```

Rückgabe: Nach erfolgreicher Durchführung wird NTCAN_SUCCESS zurückgegeben, ansonsten einer der ab Seite 37 aufgeführten Fehler-Codes.

Header: ntcan.h

Beispiel: Beispiel für Lesezugriffe (11 Bit-Identifier):

```
#include "ntcan.h"
  .
  .
  .
{
  CMSG cmsg[100];
  DWORD status;
  long len, i;
  HANDLE handle;
  .
  .
  .
  len=100;
  status =canRead( handle, cmsg, &len, NULL);
  if(status == NTCAN_SUCCESS)
    {
      for(i=0; i< len; i++)
      {
        printf("id=%03x len=%x data: ", cmsg[i].id,
                                        cmsg[i].len);
        for(j=0; j< cmsg[i].len; j++ )
          printf("%02x ", cmsg[i].data[j] );
        printf("\n");
      }
    }
  else
    {
      printf("canRead returned %x\n", status);
    }
  .
  .
  .
}
```

canWrite()

Name: **canWrite()** -Blockierendes Senden einer Nachricht

Aufruf:

```
DWORD  canWrite
       (
       HANDLE       handle,    /* Handle */
       CMSG         *cmsg,     /* Pointer auf Daten-Buffer */
       long         *len,      /* Out:  Größe des CMSG-Buffers  */
                               /* In:   Anzahl der gesendeten Nachrichten */
       OVERLAPPED   *ovrlppd   /* NULL oder Overlapped-Struktur */
       )
```

Beschreibung: Diese Funktion startet die Übertragung von **len*-Nachrichten aus der über *cmsg* definierten Speicherstruktur.
Die Größe des CMSG-Buffers muß beim Aufruf von canWrite() als Eingabeparameter in *len* definiert werden. Die Größe muß in Einheiten von Strukturen (eine Struktur enthält 16 Bytes, siehe Seite 14) angegeben werden.

Im Falle von 'NTCAN_SUCCESS' gibt **len* die Anzahl der erfolgreich gesendeten CAN-Frames zurück.

Werden keine Overlapped-Funktionen ausgeführt, kehrt die Funktion erst zurück, wenn der Sendeauftrag erfolgreich beendet oder das beim Aufruf der Funktion spezifizierte Timeout-Intervall überschritten wurde.

Die Tx-Timeout-Zeit wird für jede CAN-Nachricht, nicht für das komplette Nachrichtenpaket gezählt. Daher kann im Extremfall die Funktion das Programm für die Dauer der Nachrichtensendung multipliziert mit der Timeout-Zeit blockieren!

Nur bei Windows-Betriebssystemen:
Bei Overlapped-Operationen kann die Funktion canWrite() wie eine asynchrone Windows-Funktion angesprochen werden. Hierbei kehrt die Funktion unabhängig davon, ob die Sendung erfolgreich war oder nicht, sofort zurück. Der Status der Übertragung (z.B. Timeout) kann zu einem späteren Zeitpunkt unter Verwendung von canGetOverlappedResult() (siehe Seite 29) ermittelt werden.
Für weitere Informationen über Windows-Overlapped-Strukturen wird auf das entsprechende Windows-Handbuch verwiesen.

Für alle anderen Betriebssysteme und das CAN-Bluetooth-Modul gilt:
Hier muß für *ovrlppd* immer ‘0’ eingetragen werden.

Rückgabe: Nach erfolgreicher Durchführung wird NTCAN_SUCCESS zurückgegeben, ansonsten einer der ab Seite 37 aufgeführten Fehler-Codes.

Header: ntcan.h

canSend()

Name: **canSend()** -Nicht-blockierendes Senden einer Nachricht

Aufruf:

```
DWORD  canSend
       (
       HANDLE     handle,    /* Handle */
       CMSG       *cmsg,     /* Pointer auf Daten-Buffer */
       long       *len       /* bei Aufruf und Rückkehr: */
                             /* Out:  Anzahl der gesendeten Messages */
                             /* In:   Anzahl der zu sendenden Messages */
       )
```

Beschreibung: Diese Funktion startet die Übertragung von **len*-Nachrichten aus der über *cmsg* definierten Speicherstruktur.
Die Größe des CMSG-Buffers muß als Eingabeparameter in *len* definiert werden. Die Größe muß in Einheiten von Strukturen (eine Struktur enthält 16 Bytes, siehe Seite 14) angegeben werden.

Die Funktion kehrt sofort zurück. Im Falle von 'NTCAN_SUCCESS' gibt **len* die Anzahl der erfolgreich in die Tx-Queue kopierten CAN-Frames zurück.

Rückgabe: Nach erfolgreicher Durchführung wird NTCAN_SUCCESS zurückgegeben, ansonsten einer der ab Seite 37 aufgeführten Fehler-Codes.

Header: ntcan.h

canGetOverlappedResult()

Name: **canGetOverlappedResult()** - Ermitteln des Ergebnisses von Overlapped-Funktionen

Aufruf:

```
DWORD   canGetOverlappedResult
        (
        HANDLE      handle,     /* Handle */
        OVERLAPPED  *ovrlppd,   /* Pointer auf Overlapped-Struktur*/
        long        *len,       /* Größe in CMSG- oder Event-Messages */
        BOOL        wait        /* Auswahl des Rückgabe-Modes*/
        )
```

Beschreibung: Diese Funktion ist **nur bei Windows-Betriebssystemen** implementiert. Sie muß aufgerufen werden, um das Ergebnis von Overlapped- Operationen (d.h. canRead() mit *ovrlppd* != NULL) auszuwerten, wenn die zugehörige Funktion mit dem Fehler-Code NTCAN_IO_PENDING beendet wurde.

Der Pointer zur gewünschten Overlapped-Struktur muß in **ovrlppd* definiert werden.

**len* gibt die Anzahl der CMSG-Messages oder Event-Messages zurück.

Der Parameter *wait* selektiert den Rückgabe-Mode:

wait = TRUE	canGetOverlappedResult() kehrt zurück, wenn der entsprechende I/O-Job beendet ist.
wait = FALSE	canGetOverlappedResult() kehrt sofort zurück, auch wenn der I/O-Job noch nicht beendet ist. Die Funktion gibt NTCAN_IO_INCOMPLETE zurück.

Rückgabe: Nach erfolgreicher Durchführung wird NTCAN_SUCCESS zurückgegeben, ansonsten einer der ab Seite 37 aufgeführten Fehler-Codes.

Header: ntcan.h

4.3 Events für verschiedene Funktionen

Zur Zeit ist nur ein Event zum *Lesen* des Fehler-Status definiert (`NTCAN_EV_CAN_ERROR`). Prinzipiell sind auch Events möglich, auf die schreibend zugegriffen werden kann, daher wird der Funktionsaufruf hier ebenfalls beschrieben.

Datenstruktur von Event Messages

```
typedef struct
{
    long            evid;          /* Event-ID: 0x40000000 hex ... 0x400000FF  hex          */
    unsigned char   len;           /* Anzahl der Event-Daten-Bytes:  Bit 0-3 = Länge 0-8    */
                                   /*                                Bit 4-7 = reserviert   */
    unsigned char   reserved[3];   /* reserviert                                            */
    union
    {   unsigned char     c[8];
        unsigned short    s[4];
        unsigned long     l[2]
    }   evdata;
} EVMSG;
```

Event-IDs:
NTCAN_EV_BASE = 0x40000000 hex
NTCAN_EV_USER = 0x40000080 hex
NTCAN_EV_LAST = 0x400000FF hex

Wert von *len* [binär] bit7... ...bit0	Anzahl der Daten-Bytes [Bytes]
xxxx 0000	0
xxxx 0001	1
xxxx 0010	2
xxxx 0011	3
xxxx 0100	4
xxxx 0101	5
xxxx 0110	6
xxxx 0111	7
xxxx 1000	8

Tabelle 4.3.1: Kodierung der Länge

Die Bits 7...4 sind für zukünftige Anwendungen reserviert. Bei Lesezugriffen ist der Wert dieser Bits undefiniert. Bei Schreibzugriffen sollten diese Bits auf ‘0’ gesetzt werden.

canReadEvent()

Name: **canReadEvent()** - Lesen von Eingangsvariablen

Aufruf:

```
HANDLE  canReadEvent
        (
        HANDLE      handle,     /* Handle */
        EVMSG       *evmsg,     /* Pointer auf Event-Message-Buffer */
        OVERLAPPED  *ovrlppd    /* NULL oder Overlapped-Struktur */
        )
```

Beschreibung: Mit dieser Funktion kann z.B. der Fehler-Status über den Event `NTCAN_EV_CAN_ERROR` gelesen werden.

Werden keine Overlapped-Funktionen ausgeführt, kehrt die Funktion zurück, wenn ein Event eintritt oder das beim Aufruf der Funktion spezifizierte Timeout-Intervall überschritten wurde.

Nur bei Windows-Betriebssystemen:
Bei Overlapped-Operationen kann die Funktion canReadEvent() wie eine asynchrone Windows-Funktion angesprochen werden. Hierbei kehrt die Funktion unabhängig davon, ob ein Event eintritt oder nicht, sofort zurück. Die Auswertung des Events kann zu einem späteren Zeitpunkt unter Verwendung von canGetOverlappedResult() (siehe Seite 29) erfolgen.
Für weitere Informationen über Windows-Overlapped-Strukturen wird auf das entsprechende Windows-Handbuch verwiesen.

Für alle anderen Betriebssysteme und das CAN-Bluetooth-Modul gilt:
Hier muß für *ovrlppd* immer ‘0’ eingetragen werden.

Rückgabe: Nach erfolgreicher Durchführung wird NTCAN_SUCCESS zurückgegeben, ansonsten einer der ab Seite 37 aufgeführten Fehler-Codes.

Header: ntcan.h

canSendEvent()

Name: **canSendEvent()** - Setzen von Ausgangsvariablen

Aufruf:

```
HANDLE  canSendEvent
        (
        HANDLE      handle,     /* Handle */
        EVMSG       *evmsg      /* Pointer auf Event-Message-Buffer */
        )
```

Beschreibung: Diese Funktion dient dem Setzen von Ausgangsvariablen.
Die Funktion kann auch verwendet werden, um spezielle Read-Events zu triggern, damit sie ihre Nachrichten senden.

Diese Funktion findet zur Zeit noch keine Anwendung!

Die Funktion kehrt sofort zurück.

Rückgabe: Nach erfolgreicher Durchführung wird NTCAN_SUCCESS zurückgegeben, ansonsten einer der ab Seite 37 aufgeführten Fehler-Codes.

Header: ntcan.h

CAN-Events

NTCAN_EV_CAN_ERROR

Event-ID: NTCAN_EV_CAN_ERROR
Event-Nr.: NTCAN_EV_BASE + 0x00 (hex)
Lesen des Events: canReadEvent()
Senden des Events: nicht möglich
Daten-Länge: 6 Bytes
Daten-Format: evdata.s[0]

Offset-Adresse:	+0	+1	+2	+3	+4	+5
Belegung:	reserved	*Error*	reserved	*LOST_ MESSAGE*	reserved	*LOST_FIFO*

Tabelle 4.3.2: Belegung des Events NTCAN_EV_CAN_ERROR

Die Funktion kehrt zurück, wenn sich mindestens ein Fehler-Parameter ändert, also beim Eintreten und beim Rückgang einer Fehlerbedingung und bei Änderung der Zählerstände.

Parameter-Wertebereiche:

ERROR ... 0x00 hex - kein Fehler
(Fehlermeldung wurde zurückgenommen)
0xC0 hex - CAN-Controller-Status ist 'BUS OFF'
(Rückmeldung nur bei Sende-Aufträgen)
0x40 hex - CAN-Controller-Status ist 'WARN' (error passive)
(Rückmeldung nur bei Sende-Aufträgen)

LOST_MESSAGE ... Zähler für verlorengegangene Nachrichten des CAN-Controllers
Dieser Zähler wird von einem Fehler-Ausgang des CAN-Controllers gesetzt. Er zeigt die Anzahl der verlorengegangenen CAN-Frames an (Empfangs- oder Sende-Nachrichten).

LOST_FIFO ... Zähler für verlorengegangene Nachrichten des FIFOs
Dieser Zähler wird herauf gezählt, wenn Nachrichten aufgrund eines FIFO-Überlaufs verloren gehen (FIFO full).

4.4 Status-Message

Datenstruktur der Status-Message

```
typedef struct
{
    unsigned short  hardware;       /* Hardware-Revisionsnummer des Boards          */
    unsigned short  firmware;       /* Revisionsnummer der verwendeten Firmware     */
    unsigned short  driver;         /* Revisionsnr. des verwendeten Software-Treibers */
    unsigned short  dll;            /* Revisionsnummer der verwendeten dll          */
    unsigned long   board_status;   /* Fehler-Status des Boards                     */
    unsigned char   board_id [14];  /* Board-Id-Name (zero terminated)              */
    unsigned short  features;       /* Eigenschaften von Hardware und Treiber       */
} CAN_IF_STATUS;
```

Bedeutung der Parameter:

hardware, firmware, driver

Die aktuelle Versionsnummer der Hardware, Firmware und des Software-Treibers werden als vierstellige Hexadezimal-Zahlen zurückgegeben. Das höherwertige Byte entspricht dabei einer je 4 Bit umfassenden Major- und Minor-Nummer und das niederwertige Byte einer 8 Bit umfassenden Revisionsnummer. Der zurückgegebene Wert 0x120C (hex) ist also als Version 1.2.12 zu interpretieren.

board_status

Der Status des Boards ist wie folgt kodiert:
0 ... OK
< > ... Error

board_id

Der ID-Name der CAN-Hardware als nullterminierter ASCII-String.

features

Der Parameter enthält Flags, die eine bestimmte Konfiguration bzw. bestimmte Eigenschaften von Hardware und/oder Gerätetreiber anzeigen. Folgende Flags sind definiert:

Flag	Bedeutung
NTCAN_FEATURE_FULL_CAN	Ist das Flag gesetzt, wird auf dem CAN-Module ein FullCAN Controller (z.B. Intel 82527) eingesetzt, ansonsten ein BasicCAN Controller (z.B. Philips SJA 1000).
NTCAN_FEATURE_CAN_20B	Ist das Flag gesetzt, können CAN-Nachrichten mit 29-Bit-IDs (CAN 2.0B Mode) gesendet und empfangen werden. Ist das Flag nicht gesetzt, können nur Nachrichten mit 11-Bit-IDs (CAN 2.0A Mode) gesendet und empfangen werden.
NTCAN_FEATURE_DEVICE_NET	Ist das Flag gesetzt, unterstützt die Firmware das CAN-Protokoll *DeviceNet*.
NTCAN_FEATURE_CYCLIC_TX	Ist das Flag gesetzt, besitzt die Firmware eine kundenspezifische Erweiterung zum zyklischen Senden von Tx-Nachrichten (nur CAN-PCI/360).
NTCAN_FEATURE_RX_OBJECT_MODE	Ist das Flag gesetzt, ist eine Unterstützung für den *Rx-Object-Mode* vorhanden.

Tabelle 4.4.1: Flags des Parameters *features*

canStatus()

Name: **canStatus()** - Gibt Informationen über Hardware und Software zurück

Aufruf:

```
DWORD  canStatus
       (
       HANDLE          handle,     /* Status-Handle */
       CAN_IF_STATUS   *cstat      /* Pointer auf Status-Struktur */
       )
```

Beschreibung: Diese Funktion liefert Informationen über Hardware und Software der diesem CAN-Handle zugeordneten CAN-Schnittstelle. Die Zuordnung des Handles zu einer CAN-Schnittstelle geschieht beim Öffnen des Handles über die logische Netznummer beim Aufruf von canOpen().

Rückgabe: Nach erfolgreicher Durchführung wird NTCAN_SUCCESS zurückgegeben und die Status-Informationen werden in den Speicherbereich geschrieben auf den der Zeiger *cstat* verweist. Im Fehlerfall wird einer der ab Seite 37 aufgeführten Fehler-Codes zurückgegeben.

Header: ntcan.h

5. Rückgabewerte

Alle API-Aufrufe geben einen Rückgabewert zurück, der mit dem Präfix `'NTCAN_'` beginnt und in der Applikation stets abgefragt werden sollte. Kehrt der Aufruf mit einem Fehlercode zurück, so sind die Inhalte aller durch Zeiger referenzierten Rückgabewerte undefiniert und dürfen von der Applikation nicht ausgewertet werden.

Die Konstanten für die Rückgabewerte sind in der Header-Datei ntcan.h definiert. In Applikationen sollten aus Gründen der Portabilität stets diese Konstanten genutzt werden, da jedes Betriebssystem einen eigenen 'Nummernraum' für die numerischen Werte von Fehlern festlegt und daher den Konstanten in der Header-Datei für verschiedene Betriebssysteme unterschiedliche numerische Werte zugeordnet werden müssen. Ferner werden zur Erhöhung der Portabilität einige Konstanten für Fehler, die nicht CAN-spezifisch sind und typischerweise vom Betriebssystem autonom erzeugt werden (z.B. `NTCAN_INVALID_HANDLE`), auf bereits vorhandene Fehlerkonstanten des Betriebssystems abgebildet.

Nachfolgend werden alle Rückgabewerte in Tabellenform aufgelistet. Die Werte werden in die Fehlerklassen *Erfolgreich*, *Warnung* und *Fehler* eingeteilt. Ferner wird eine Beschreibung für die Fehlerursache und die Lösungsmöglichkeiten sowie eine Liste der API-Aufrufe, bei denen dieses Problem auftreten kann, angeführt.

Fehlerkonstanten, die in der Header-Datei vorhanden, hier aber nicht beschrieben sind, werden vom Treiber nicht mehr erzeugt und sind nur noch aus Gründen der Rückwärtskompatibilität von Quelltexten nicht entfernt.

NTCAN_SUCCESS

Erfolgreiche Ausführung des Aufrufs.

Klasse	Erfolgreich
Ursache	Der Aufruf wurde ohne Fehler beendet. Die Inhalte aller durch Zeiger referenzierten Rückgabewerte sind gültig und dürfen von der Applikation ausgewertet werden.
Funktion	Alle Funktionen

NTCAN_CONTR_BUSY

Die Kapazität der internen Sende-FIFO wurde überschritten.

Klasse	Fehler/Warnung
Ursache	Die Kapazität der internen Sende-FIFO reicht nicht aus, um zusätzliche Nachrichten aufzunehmen.
Problem-lösung	Wiederholung des fehlerhaften Aufrufs nach kurzer Wartezeit.
Funktion	canSend(), canSendEvent(), canIdAdd()

NTCAN_CONTR_OFF_BUS

Fehler beim Senden.

Klasse	Fehler
Ursache	Der CAN-Controller ist während einer blockierenden Sende-Operation in den Zustand *OFF BUS* gewechselt, da zu viele CAN-Errorframes empfangen wurden.
Problem-lösung	Wiederholung des Aufrufs nach kurzer Wartezeit, da der Treiber automatisch versucht, die Fehlersituation zu beheben. Tritt der Fehler weiterhin auf, sollte geprüft werden, ob die Verkabelung des CAN-Busses korrekt ist und alle Busteilnehmer mit der gleichen Baudrate senden.
Funktion	canWrite()

NTCAN_CONTR_WARN

Fehler beim Senden.

Klasse	Fehler
Ursache	Der CAN-Controller ist während einer blockierenden Sende-Operation in den Zustand *Error Passive* gewechselt, da zu viele CAN- Errorframes empfangen wurden.
Problem-lösung	Wiederholung des Aufrufs nach kurzer Wartezeit, da der Treiber automatisch versucht, die Fehlersituation zu beheben. Tritt der Fehler weiterhin auf, sollte geprüft werden, ob die Verkabelung des CAN-Busses korrekt ist und alle Busteilnehmer mit der gleichen Baudrate senden.
Funktion	canWrite()

NTCAN_ID_ALREADY_ENABLED

Der CAN-ID wurde für dieses Handle bereits aktiviert.

Klasse	Warnung
Ursache	Der CAN-Identifier wurde für dieses Handle bereits aktiviert.
Problem-lösung	Jeden CAN-ID pro Handle nur einmal aktivieren. Die Aktivierung einer CAN 2.0B ID führt dazu, daß alle anderen CAN 2.0B Ids ebenfalls als aktiviert gelten.
Funktion	canIdAdd()

NTCAN_INSUFFICIENT_RESOURCES

Nicht genügend interne Resourcen verfügbar.

Klasse	Fehler
Ursache	Aufgrund eines Mangels an internen Resourcen konnte der Aufruf nicht ausgeführt werden.
Problem-lösung	• Tritt der Fehler beim Aufruf von canOpen() auf, sollte die Handle-Queuesize verkleinert werden. • Tritt der Fehler beim Aufruf von canIdAdd() auf, ist dieser CAN-ID bereits für zu viele andere Handles aktiviert worden.
Funktion	canOpen(), canIdAdd()

NTCAN_INVALID_DRIVER

Treiber und NTCAN-Bibliothek passen nicht zusammen.

Klasse	Fehler
Ursache	Die Version der NTCAN-Bibliothek setzt eine neuere Treiberversion voraus.
Problem-lösung	Neuere Treiberversion verwenden.
Funktion	canOpen()

NTCAN_INVALID_FIRMWARE

Treiber und Firmware passen nicht zusammen.

Klasse	Fehler
Ursache	Die Version des Gerätetreibers setzt eine neuere Firmwareversion voraus.
Problem-lösung	Firmware der aktiven CAN-Karte aktualisieren.
Funktion	canOpen()

NTCAN_INVALID_HANDLE

Ungültiger CAN-Handle.

Klasse	Fehler
Ursache	Einem Funktionsaufruf wurde ein ungültiges Handle übergeben.
Problem-lösung	• Überprüfen, ob das Handle mit canOpen() korrekt geöffnet wurde. • Überprüfen, ob das Handle nicht zuvor mit canClose() geschlossen wurde.
Funktion	Alle Funktionen außer canOpen()

NTCAN_INVALID_HARDWARE

Treiber und Hardware passen nicht zusammen.

Klasse	Fehler
Ursache	Die Version des Gerätetreibers funktioniert mit dieser Hardware nicht.
Problem-lösung	Andere Treiberversion verwenden.
Funktion	canOpen()

NTCAN_INVALID_PARAMETER

Ungültiger Parameter.

Klasse	Fehler
Ursache	Einem Aufruf wurde ein ungültiger Parameter übergeben.
Problem-lösung	• Überprüfung aller Argument für diese Aufruf auf Gültigkeit. • Aufruf von canRead() mit einem Handle, das für den Objekt-Modus geöffnet wurde.
Funktion	Alle Funktionen

NTCAN_IO_INCOMPLETE

Operation noch nicht beendet (Nur Win32).

Klasse	Fehler/Warnung
Ursache	Die Funktion canGetOverlappedResult() wurde mit FALSE für den Parameter *bWait* aufgerufen und die asynchrone Operation ist nicht abgeschlossen. Siehe Win32 Plattform-SDK Hilfe zu canGetOverlappedResult() für weitere Details.
Problem-lösung	Siehe Win32 Plattform-SDK Hilfe zu canGetOverlappedResult().
Funktion	canGetOverlappedResult()

NTCAN_IO_PENDING

Operation nicht beendet (Nur Win32).

Klasse	Warnung
Usache	Eine asynchroner Lese-/oder Schreiboperation mit gültiger Overlapped-Struktur wurde nicht abgeschlossen. Siehe Win32 Plattform-SDK zum Thema ‘Asynchronous Input and Output’ für weitere Details.
Problem-lösung	Siehe Win32 Plattform-SDK zum Thema ‘Asynchronous Input and Output’ für weitere Details.
Funktion	canRead(), canWrite(), canReadEvent()

NTCAN_NET_NOT_FOUND

Ungültige logische Netznummer.

Klasse	Fehler
Usache	Die beim Aufruf von canOpen() angegebene logische Netznummer existiert nicht.
Problem-lösung	• Überprüfen, ob die Netznummer vergeben wurde. • Überprüfen, ob der Treiber, dem diese Netznummer zugeordnet sein soll, korrekt gestartet ist.
Funktion	canOpen()

NTCAN_NO_ID_ENABLED

Der CAN-ID ist für dieses Handle nicht aktiviert.

Klasse	Warnung
Usache	Der CAN-ID ist für dieses Handle nicht aktiviert.
Problem-lösung	Jeden CAN-ID pro Handle nur einmal deaktivieren. Die Deaktivierung einer CAN 2.0B ID führt dazu, daß alle anderen CAN 2.0B IDs ebenfalls als deaktiviert gelten.
Funktion	canIdDelete()

NTCAN_ABORTED

Abbruch einer blockierenden Sende-/Empfangsoperation.

Klasse	Warnung/Fehler
Usache	Eine blockierende Sende-/oder Empfangsoperation wurde abgebrochen, weil ein anderer Thread für dieses Handle canClose() aufgerufen hat oder der Treiber terminiert wurde.
Problem-lösung	Wenn das Verhalten von der Applikation nicht gewollt ist, überprüfen warum Handle geschlossen werden.
Funktion	canRead(), canWrite(), canReadEvent()

NTCAN_PENDING_READ

Empfangsoperation konnte nicht durchgeführt werden.

Klasse	Warnung/Fehler
Usache	Eine Empfangsoperation wurde nicht initiiert, da das Handle bereits von einem anderen Thread für eine Empfangsoperation genutzt wird.
Problem-lösung	• Durch Synchronisationsmaßnahmen verhindern, daß ein anderer Thread das Handle ebenfalls gleichzeitig zum Empfangen verwendet. • Threads mit unterschiedlichen Handlen nutzen.
Funktion	canRead(), canTake(), canReadEvent()

NTCAN_PENDING_WRITE

Sendeoperation konnte nicht durchgeführt werden.

Klasse	Warnung/Fehler
Usache	Eine Sendeoperation wurde nicht initiiert, da das Handle bereits von einem anderen Thread für eine Sendeoperation genutzt wird.
Problem-lösung	• Durch Synchronisationsmaßnahmen verhindern, daß ein anderer Thread das Handle ebenfalls gleichzeitig zum Senden verwendet. • Threads mit unterschiedlichen Handlen nutzen.
Funktion	canWrite(), canSend(), canSendEvent()

NTCAN_RX_TIMEOUT

Timeout-Ereignis für blockierende Empfangsoperation.

Klasse	Warnung
Usache	Innerhalb der bei canOpen() vereinbarten Rx-Timeouts wurden keine Daten empfangen.
Problem-lösung	• Rx-Timeout bei canOpen() vergrößern. • Sicherstellen, daß Daten auf den gewünschten CAN-IDs von anderen CAN-Teilnehmer gesendet werden. • Verkabelung überprüfen.
Funktion	canRead(), canReadEvent()

NTCAN_TX_ERROR

Timeout-Ereignis für blockierende Sendeoperation.

Klasse	Fehler
Usache	Innerhalb einer internen Timeout-Zeit (typischerweise 1000 ms) konnte die Nachricht nicht gesendet werden. Dieser Fehler wird von NTCAN_TX_TIMEOUT ersetzt, wenn bei canOpen() für das Tx-Timeout eine kleinere Zeit gewählt wurde.
Problem-lösung	• Verkabelung überprüfen. • Baudrate überprüfen.
Funktion	canWrite()

NTCAN_TX_TIMEOUT

Timeout-Ereignis für blockierende Sendeoperation.

Klasse	Warnung/Fehler
Usache	Innerhalb der bei canOpen() vereinbarten Tx-Timeouts konnten die Daten nicht gesendet werden.
Problem-lösung	• Tx-Timeout bei canOpen() vergrößern. • Verkabelung überprüfen. • Baudrate überprüfen.
Funktion	canWrite()

6. Beispielprogramme

6.1 Beispielprogramm: Empfangen von Nachrichten

```
#include <stdio.h>
#include <ntcan.h>

/*
 * This example demonstartes how the NTCAN-API can be used to open a handle,
 * set a baudrate and wait for reception of a CAN frame with an
 * identifier that has been previously enabled for this handle.
 * Finally all proper cleanup operations are performed
 */
int main(void)
{
  int net=0;               /* logical netnumber used for example */
  unsigned long mode=0;    /* mode for canOpen */
  long txqueuesize=8;      /* maximum number of messages to transmit */
  long rxqueuesize=8;      /* maximum number of messages to receive */
  long txtimeout=100;      /* timeout for transmit */
  long rxtimeout=10000;    /* timeout for receiving data */
  HANDLE rxhandle;         /* can handle for used functions */
  DWORD retvalue;          /* returnvalue for used functions */
  DWORD baud=2;            /* used baudrate (here: 500 kBit/sec.) */
  CMSG cmsg[8];            /* buffer for can messages */
  int rtr=0;               /* rtr bit */
  int i;                   /* loop counter */
  int len;                 /* Bufsize in number of messages for canRead() */

  /* ################################################################ */

  retvalue = canOpen(   net,
                        mode,
                        txqueuesize,
                        rxqueuesize,
                        txtimeout,
                        rxtimeout,
                        &rxhandle);

  if (retvalue != NTCAN_SUCCESS)
    {
      printf("canOpen() failed with error %d!\n", retvalue);
      return(-1);
    }

  printf("function canOpen() returned OK !\n");

  /* ################################################################ */

  retvalue = canSetBaudrate(  rxhandle,
                              baud);

  if (retvalue != 0)
    {
      printf("canSetBaudrate() failed with error %d!\n", retvalue);
      canClose(rxhandle);
      return(-1);
    }

  printf("function canSetBaudrate() returned OK !\n");
```

```
/* ################################################################ */

retvalue = canIdAdd(rxhandle,
                    0);

if (retvalue != NTCAN_SUCCESS)
  {
    printf("canIdAdd() failed with error %d!\n", retvalue);
    canClose(rxhandle);
    return(-1);
  }

printf("function canIdAdd() returned OK !\n");

/* ################################################################ */

do {
  /*
   * Set max numbers of messages that can be returned with
   * one canRead() call according to buffer size
   */
  len = 8;

  retvalue = canRead( rxhandle,
                      &cmsg[0],
                      &len,
                      NULL);

  if (retvalue == NTCAN_RX_TIMEOUT)
    {
      printf("canRead() returned timeout\n");
      continue;
    }
  else if(retvalue != NTCAN_SUCCESS)
    {
      printf("canRead() failed with error %d!\n", retvalue);
    }
  else
    {
      printf("function canRead() returned OK !\n");
      printf("Id of received message :%x!\n", cmsg[0].id);
      printf("Len of received message :%x!\n", (cmsg[0].len & 0x0f));
      printf("Rtr of received message :%x!\n", ((cmsg[0].len & 0x10)>>4));
      for (i=0;i<(cmsg[0].len & 0x0f);i++)
        printf("Byte %d of received message :%x!\n", i, cmsg[0].data[i]);
    }

  break;
} while(1);

/* ################################################################ */

retvalue = canIdDelete( rxhandle,
                        0);

if (retvalue != NTCAN_SUCCESS)
   printf("canIdDelete() failed with error %d!\n", retvalue);

printf("function canIdDelete() returned OK !\n");
```

```
  /* ################################################################ */

  retvalue = canClose (rxhandle);

  if (retvalue != NTCAN_SUCCESS)
    printf("canClose() failed with error %d!\n", retvalue);

  printf("function canClose returned OK !\n");

  /* ################################################################ */

  return(0);
}
```

6.2 Beispielprogramm: Senden von Nachrichten

```
/*
 * This example demonstartes how the NTCAN-API can be used to open a handle,
 * set a baudrate and transmitting a CAN frame.
 * Finally all proper cleanup operations are performed
 */
int main(void)
{
  int net=0;                  /* Net number used in this example */
  unsigned long mode=0;       /* mode used for canOpen */
  long txqueuesize=8;         /* size of transmit queue */
  long rxqueuesize=8;         /* size of receive queue */
  long txtimeout=100;         /* timeout for transmit operations */
  long rxtimeout=1000;        /* timeout for receive operations */
  HANDLE txhandle;            /* can handle used for several functions */
  DWORD retvalue;             /* returnvalue for can functions */
  DWORD baud=2;               /* baudrate of this example. (here: 500 kBit/s.)*/
  CMSG cmsg[8];               /* can message buffer */
  int rtr=0;                  /* rtr bit */
  int i;                      /* loop counter */
  int len;                    /* no of messages of the message buffer (cmsg) */
                              /* that should be transmitted                  */

  /* ############################################################### */

  retvalue = canOpen(net,
                     mode,
                     txqueuesize,
                     rxqueuesize,
                     txtimeout,
                     rxtimeout,
                     &txhandle);

  if (retvalue != NTCAN_SUCCESS)
    {
      printf("canOpen() failed with error %d!\n", retvalue);
      return(-1);
    }

  printf("function canOpen() returned OK !\n");

  /* ############################################################### */

  retvalue = canSetBaudrate(  txhandle,
                              baud);

  if (retvalue != 0)
    {
      printf("canSetBaudrate() failed with error %d!\n", retvalue);
      canClose(txhandle);
      return(-1);
    }

  printf("function canSetBaudrate() returned OK !\n");

  /* ############################################################### */

  /*
   * Initialize the first message in buffer to CAN id = 0, len = 3
   * and data0 - data2 = 0,1,2
   */
```

```
  cmsg[0].id=0x00;
  cmsg[0].len=0x03;
  cmsg[0].len |= cmsg[0].len + (rtr<<4);
  for (i=0;i<3;i++)
    cmsg[0].data[i] = i;

  len=1;                        /* Number of valid messages in cmsg buffer*/

  retvalue =  canWrite( txhandle,
                    &cmsg[0],
                    &len,
                    NULL);

  if (retvalue != NTCAN_SUCCESS)
    printf("canWrite failed() with error %d!\n", retvalue);
  else
    printf("function canWrite() returned OK !\n");

  /* ################################################################ */

  retvalue = canClose (txhandle);

  if (retvalue != NTCAN_SUCCESS)
    printf("canClose failed with error %d!\n", retvalue);
  else
    printf("function canClose() returned OK !\n");

  /* ################################################################ */

  return(0);
}
```

7. Testprogramm cantest

7.1 `cantest` für die Kommandozeileneingabe als Beispielprogramm

Das Testprogramm `cantest` ist im Software-Paket als Quell-Code enthalten. Es kann daher als Beispielprogramm für die Funktionen des CAN-API genutzt werden.

`cantest` ist auf allen unterstützen Betriebssystemen lauffähig.

Im folgenden wird die Funktionsweise des Programms beschrieben. Das Listing des Quell-Codes ist in diesem Handbuch nicht extra abgedruckt. Es kann dem mitgelieferten Datenträger entnommen werden.

7.1.1 Funktionsbeschreibung

Wird `cantest` ohne Parameter aufgerufen, sucht es nach den bisher im System installierten esd-CAN-Boards und zeigt Board-Parameter wie Software-Revision und Board-Status an. Außerdem wird eine Liste der verfügbaren Funktionen und der Eingabe-Syntax angezeigt.

Die folgende Abbildung zeigt ein Beispiel für die Monitor-Ausgabe nach dem Aufruf von `cantest`:

```
CAN Test Rev 2.4.4  -- (c) 1997-99 electronic system design gmbh

Available CAN-Devices:
Net   0: ID=CAN_PCI331 Dll=1.5.01 Driver=1.6.05
         Firmware=0.C.08 Hardware=1.1.00 Baudrate=7fffffff Status=00000000
Net   1: ID=CAN_PCI331 Dll=1.5.01 Driver=1.6.05
         Firmware=0.C.08 Hardware=1.1.00 Baudrate=7fffffff Status=00000000

Syntax: cantest test-Nr [net id-1st id-last count
        txbuf rxbuf txtout rxtout baud testcount data0 data1 ...]
Test  0:  canSend()
Test  1:  canWrite()
Test  2:  canTake()
Test 12:  canTake() with time-measurement for 10000 can-frames
Test  3:  canRead()
Test 13:  canRead() with time-measurement for 10000 can-frames
Test  4:  canReadEvent()
Test  5:  canSendEvent()
Test  6:  Overlapped-canRead()
Test  7:  Overlapped-canWrite()
Test  9:  Wait for RTR reply
```

Abb. 7.1.1: Bildschirmausgabe nach dem Aufruf von `cantest`

Im oben gezeigten Beispiel ist zu erkennen, daß in dem System die beiden CAN-Netze mit den logischen Netznummern 0 und 1 auf der Hardware CAN-PCI/331 verfügbar sind. Daran anschließend ist die Eingabe-Syntax dargestellt.

Die Eingabeparameter müssen immer in der Reihenfolge, in der sie dargestellt sind, eingegeben werden. Soll z.B. nur der Parameter '`testcount`' verändert werden, so müssen alle vorangehenden Parameter ebenfalls eingegeben werden. Werden die folgenden Parameter nicht eingegeben, so wird der Default-Wert der Parameter angenommen.

7.1.2 Besonderheiten der VxWorks-Implementierung

Für die VxWorks Implementierung von `cantest` sind folgende Abweichungen von der allgemeinen Beschreibung zu beachten:

- Für den Programmaufruf ist die Schreibweise von cantest wie folgt: `canTest`

- Die Eingabe der Parameter muß unter VxWorks in Hochkommas ("...") erfolgen.

7.1.3 Parameter-Beschreibung

Alle Parameter, mit Ausnahme der Identifier müssen als Dezimalzahlen eingegeben werden. Für die Identifier sind sowohl Dezimalzahlen als auch Hexadezimalzahlen zulässig.

`test-Nr` Die verfügbaren Test-Nummern sind in der vorangegangenen Abbildung aufgelistet. Test Nummer 5 (canSendEvent()) ist zur Zeit ohne Funktion. Die Tests Nummer 6 und 7 (Overlapped-Funktionen) können nur bei Windows-Betriebssystemen ausgeführt werden.

`net` Logische Nummer des CAN-Netzes

`id-1st,id-last`

Lesen: Über diese beiden Grenzwerte kann ein Identifier-Bereich definiert werden, innerhalb dessen Nachrichten empfangen werden sollen. Sollen nur Nachrichten eines Identifiers ausgewertet werden, so ist für beide Parameter der gleiche Wert einzutragen.

Schreiben: `id-1st` wird als Tx-Identifier angenommen. `id-last` wird ignoriert.

Default: `id-1st` = 0
`id-last` = 0

Zahlenformat: Zahlenwerte ohne weitere Angaben werden als Dezimalzahlen ausgewertet. Die Angabe '0x' vor einer Zahl kennzeichnet eine Hexadezimalzahl.

Wertebereich für 11-Bit-Identifier: 0...2047 (dez)
0x0 ... 0x7FF (hex)
Beispiel: 0x3F = 63 (dez)

Wertebereich für 29-Bit-Identifier: 0x20000000 ... 0x3FFFFFFF (hex)
Beispiel: 0x2000003F = 63 (dez)

`count` Die Anzahl der CAN-Tx-Messages, die mit einem Funktionsaufruf übergeben werden sollen, wird hier eingetragen.
Default-Wert: `count` = 1

`txbuf` Größe der Tx-Queue.
Default-Wert: `tx-buf` = 10

`rxbuf` Größe der Rx-Queue.
Default-Wert: `rx-buf` = 100

`txout` Tx-Timeout in [ms].
Default-Wert: `txout` = 1000 ms

`rxout` Rx-Timeout in [ms].
Default-Wert: `rxout` = 5000 ms

`baud` Baudraten-Index (siehe Baudratentabelle bei der Funktion ‘canSetBaudrate’ auf Seite 18.)
Default-Wert: `baud` = 2 (500 kBit/s)

`testcount` Anzahl der Test-Schleifen, die durchlaufen werden sollen.
Default-Wert: bei write/send-Funktionen `testcount` = 10
bei read/take-Funktionen `testcount` = ∞
Um den Wert ‘∞’ einzustellen, ist für `testcount` ‘-1’ einzutragen.

`data0...data7` Zu sendende Daten-Bytes.
Wird für diese Parameter kein Eintrag vorgenommen, so werden bei Sendekommandos zwei 32-Bit-Zählerwerte gesendet. Werden ein bis acht Bytes angegeben, so werden die angegebenen Bytes gesendet. Die Eingabe der Daten-Bytes muß als Dezimalzahl erfolgen !

7.2 Testprogramm WinCANTest mit Windows-Oberfläche

Das Programm WinCANTest ist unter Windows 95/98 lauffähig. Es bietet ähnliche Funktionen wie das Programm cantest für die Kommandozeileneingabe. Folgende Einschränkungen sind bei WinCANTest gegenüber `cantest` zu beachten:

- Die Funktionen ‘`canTake() with time-measurement for 10000 can-frames`’, ‘`canRead() with time-measurement for 10000 can-frames`’ und ‘`Wait for RTR reply`’ werden noch nicht unterstützt.

- 29-Bit-Identifier nach CAN 2.0B werden zur Zeit noch nicht unterstützt.

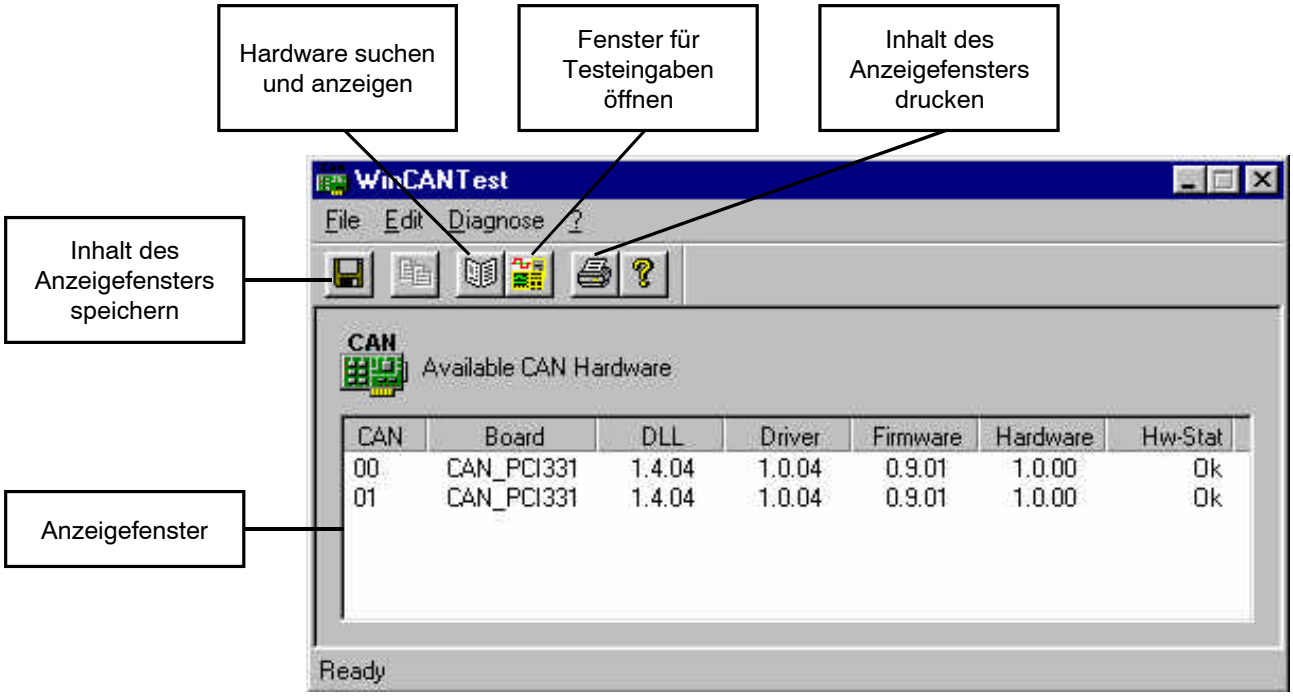


Abb. 7.2.1: WinCANTest-Fenster

Die Schaltflächenleiste bietet Funktionen, die auch über die Menüs aufgerufen werden können. Im Menü *File* finden sich weitergehende Funktionen zum Drucken.

Im Fenster *Test* werden das auszuführende Kommando, die Parameter und die Daten für die Test-Zugriffe eingetragen.

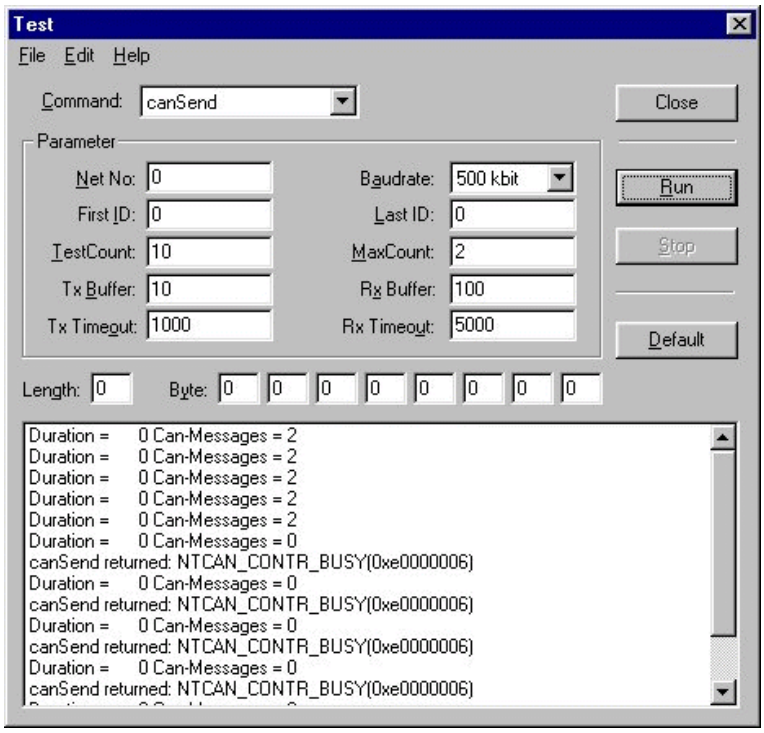


Abb. 7.2.2: Ein-Ausgabefenster für die CAN-Bus-Zugriffe

Die Beschreibung der Kommandos kann den Funktionsbeschreibungen in den ersten Kapiteln dieses Handbuches entnommen werden. Die Beschreibung der Parameter erfolgte bereits in dem vorangegangenen Kapitel ‘cantest für die Kommandozeileneingabe als Beispielprogramm’.

8. Monitor-Programm CANscope

8.1 Übersicht

Das Monitorprogramm CANscope kann unter allen unterstützten Windows-Betriebssystemen (Windows NT/2000/XP, Windows 95/98/ME) betrieben werden ! Eine ähnliche Bedienoberfläche mit ähnlichem Funktionsumfang wie für die Windows-Betriebssysteme ist für die unterstützten UNIX-Betriebssysteme (Linux, Linux-RTAI, LynxOS, PowerMAX OS, SGI-IRIX6.5 und Solaris) verfügbar.

CANscope ist ein Menü-gesteuertes Programm, das zur Überwachung und zum Test von CAN-Netzen dient. Seine selbsterklärende Bedienoberfläche bietet einen schnellen Einstieg in die Funktionsweise des CAN-Netzes.

Dieses Kapitel gibt eine ausführliche Beschreibung der einzelnen Menüpunkte von CANscope. Im Anschluß daran ist ein Anwendungsbeispiel aufgeführt.

8.2 Programmaufruf

Das Programm wird bei der Installation des Device-Drivers automatisch mit installiert. Vor Aufruf des Programms muß der Treiber gestartet werden.

Hinweis: CANscope kann parallel zu anderen Applikationen laufen und z.B. die dort verwendeten Identifier anzeigen oder auf beliebigen Identifiern senden. Es lassen sich auch mehrere CANscope-Fenster gleichzeitig öffnen.

Beim Programmaufruf in der Kommandozeile lassen sich Parameter angeben, die bereits beim Programmstart die persönlichen Voreinstellungen des Programms mit laden:

Aufruf	Funktion
`canscope --start`	CANscope wird aufgerufen und initialisiert, so daß sofort die eintreffenden CAN-Frames angezeigt werden.
`canscope` *`profilname`*	CANscope wird aufgerufen und die in der Profile-Datei *`profilname`* gespeicherten Parametereinstellungen werden übernommen.
`canscope` *`profilname`* `--start`	Kombination aus den beiden oben angegebenen Aufrufen.

Tabelle 8.2.1: Aufruf von CANscope mit Parametern

Das Anlegen und die Bedeutung von Profile-Dateien werden auf Seite 61 beschrieben.

8.3 Erläuterung der Funktionen der Oberflächenelemente

8.3.1 Darstellung des CANscope-Fensters

Das CANscope-Fenster hat folgenden Aufbau:

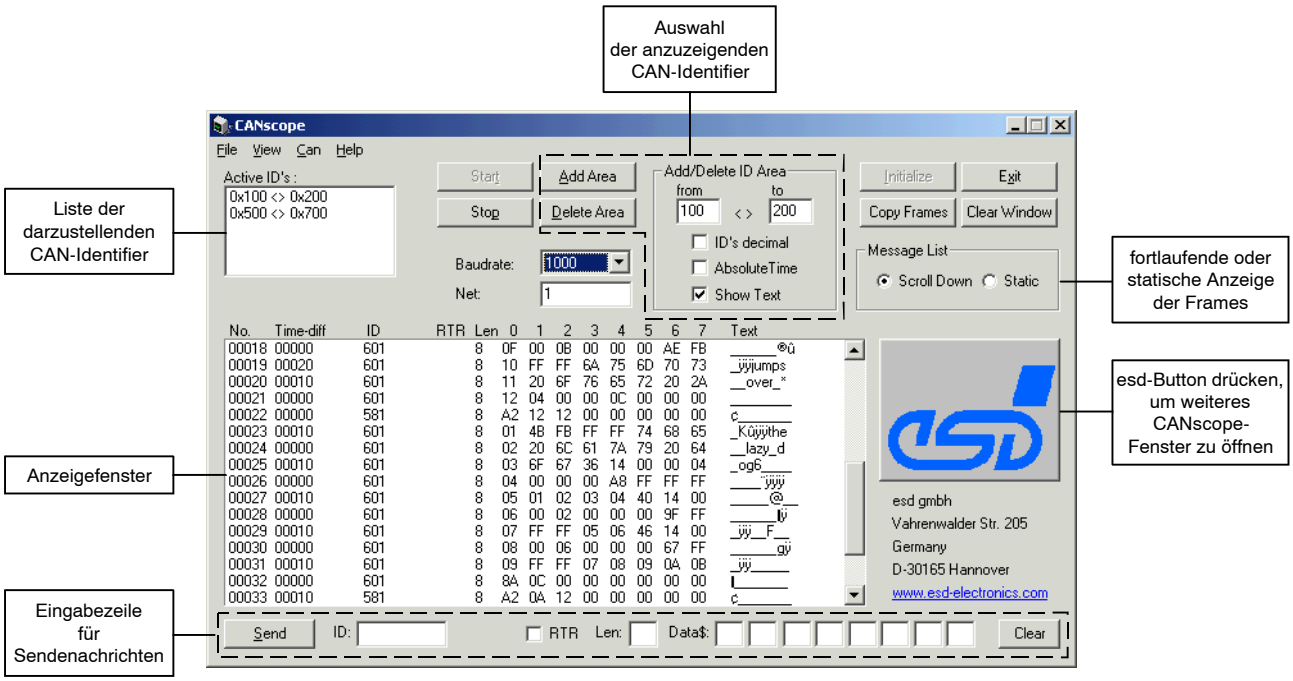


Abb. 8.3.1: CANscope-Fenster

8.3.2 Anzeigefenster und Schaltflächen

Anzeigefenster

Im Anzeigefenster erfolgt die Darstellung der gewünschten CAN-Frames.

Spalte	Format	Bedeutung
Frm.-No	dezimal	Nachrichtennummer Jeder der darzustellenden Frames erhält eine fortlaufende Nummer.
Time-diff	dezimal	Zeitstempel In dieser Spalte wird der Empfangszeitpunkt des dargestellten Frames angezeigt. Über den Auswahlpunkt *Absolute Time* läßt sich festlegen, ob die Zeit relativ zum ersten empfangenen Frame oder zum vorhergehenden Frame gezählt werden soll.
ID	dezimal oder hexadezimal	CAN-Identifier
RTR	keine Anzeige oder 'RTR'	Hier wird angezeigt, on der Frame mit aktivem RTR-Bit empfangen worden ist.
Len	0...8	Anzahl der gültigen Daten-Bytes des Frames.
0 1 ... 7	hexadezimal	Daten-Bytes als Hexadezimalzahlen.
Text	ASCII	Falls über *Show Text* gewählt, werden hier die empfangenen Daten als ASCII-Text angezeigt.

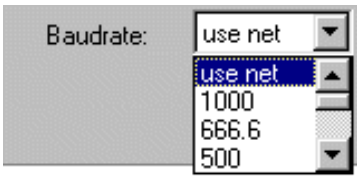
Tabelle 8.3.1: Bedeutung der Einträge im Anzeigefenster

Die anzuzeigenden Frames werden über die *Add Area*-Schaltfläche und die zugehörigen Eingabefelder ausgewählt.
Für die Darstellung der Frames kann außerdem gewählt werden, ob jeder eintreffende Frame in einer neuen Zeile angezeigt werden soll (Auswahlpunkt *Scroll Down*), oder ob die neuen Daten in einer bereits vorhandenen Zeile aktualisiert werden sollen (Auswahlpunkt *Static*).

Achtung!	Die Zeitdifferenzen werden bisher noch vom Monitor zu dem Zeitpunkt ermittelt, in dem eine oder mehrere Nachrichten aus dem Buffer des Treibers gelesen werden und sind daher stark von der Betriebssystemauslastung abhängig.

8.3.3 Beschreibung der Schaltflächen

Baudrate



In dieser Auswahlbox wird die Baudrate eingestellt. Wird der Eintrag *use net* gewählt, so arbeitet CANscope mit der Baudrate, die über den CAN-Treiber eingestellt wurde. Alterativ kann auch einer der angebotenen Zahlenwerte eingegeben werden.

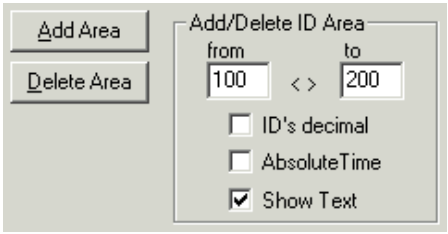
Net

In dieser Eingabebox ist die verwendete Netznummer der Karte einzugeben. Bei der ersten Karte im PC sind dieses normalerweise die Netznummern 0 und 1 (bei einer Ein-Kanal-Karte nur die 0), bei jeder weiteren Karte sind die Netznummern entsprechend aufsteigend (2 und 3, 4 und 5 ... bei Ein-Kanal-Karten 2, 4, ...).

Initialize

Diese Schaltfläche aktiviert die Initialisierung des CAN-PC-Moduls mit den eingestellten Werten für Netznummer und Baudrate. Ist die Schaltfläche inaktiv dargestellt, so hat das Modul die in den Eingabefeldern für Netznummer und Baudrate eingestellten Werte bereits übernommen.

Add/Delete ID Area



Hier werden die Unter- und Obergrenze des zu aktivierenden oder deaktivierenden ID-Bereichs eingegeben. Es ist erlaubt, ID-Bereiche zu aktivieren oder zu deaktivieren, die bereits aktiv bzw. inaktiv sind. Die ID's sind, falls nicht anders gewählt, hexadezimal einzutragen (siehe Schaltfläche *ID's decimal*).

ID's decimal

Nach dem Start des Monitors werden alle ID's hexadezimal dargestellt. Durch betätigen dieser Schaltfläche werden alle ID's im Anzeigefenster dezimal dargestellt und in allen ID-Eingabeboxen werden die hexadezimalen Werte durch entsprechende dezimale Werte ersetzt.

Absolute Time

Über den Auswahlpunkt *Absolute Time* läßt sich festlegen, ob die Zeit relativ zum ersten empfangenen Frame oder zum vorhergehenden Frame gezählt werden soll. Die absolute Zeitdarstellung erfolgt im Format `HH:MM:SS.msec` (jeweils 2 Ziffern für Stunden, Minuten und Sekunden und 3 für Millisekunden). Die relative Zeitdarstellung erfolgt in Millisekunden als fünfstellige Dezimalzahl.

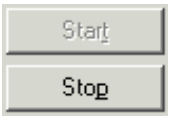
Show Text

Wird *Show Text* gewählt, so werden die empfangenen CAN-Daten in der Spalte 'Text' im Anzeigefenster als ASCII-Text dargestellt.

Add Area
Delete Area

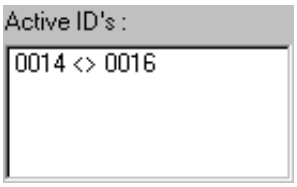
Mit diesen Schaltflächen wird der ID-Bereich, der in den Eingabeboxen angezeigt wird, aktiviert bzw. deaktiviert.

Start
Stop



Durch Drücken dieser Schaltflächen wird die Darstellung der empfangenen Nachrichten gestartet bzw. angehalten.

Active ID's



In diesem Fenster werden die aktiven ID-Bereiche angezeigt. Das einfache Anklicken mit der linken Maustaste übernimmt den Wertebereich in die Eingabefelder *from* und *to*. Ein Doppelklick löscht den angeklickten ID-Bereich.

Clear Window

Durch Betätigen dieser Schaltfläche werden alle CAN-Nachrichtenobjekte im Anzeigefenster gelöscht und der Nachrichten-Zähler wird auf Null zurückgesetzt.

Copy Frames

Bei Betätigung dieser Schaltfläche wird der Inhalt des Anzeigefensters als Textdatei in die Zwischenablage kopiert.

Message List (Scroll Down/Static)

Für die Darstellung der Frames kann ausgewählt werden, ob jeder eintreffende Frame in einer neuen Zeile angezeigt werden soll (*Scroll Down*), oder ob für jeden Identifier nur eine Zeile angezeigt wird, deren Inhalt beim Eintreffen neuer Daten aktualisiert wird (*Static*). Bei der 'statischen' Anzeige verdecken eintreffende RTR-Antwort-Frames sofort den gesendeten RTR-Frame.

Eingabezeile für Sendenachrichten

Die Daten der zu sendenden CAN-Nachrichten müssen in der Eingabezeile am unteren Fensterrand eingegeben werden. Es gelten die gleichen Datenformate wie für das Anzeigefenster. Werte außerhalb des gültigen Wertebereiches werden nicht übernommen.

Send

Durch Betätigen dieser Schaltfläche werden die in der Eingabezeile enthaltenen Daten gesendet.

Clear

Durch einmaliges Betätigen dieser Schaltfläche wird der Eintrag der Sendedaten in der Eingabezeile für Sendedaten gelöscht.
Durch zweimaliges Betätigen dieser Schaltfläche werden außerdem die Einträge für den CAN-Identifier und die Länge gelöscht.

8.3.4 Menüs

Das Hauptmenü enthält vier Menüpunkte:

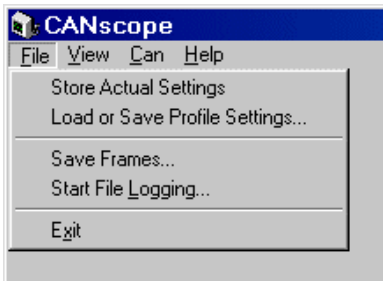


Abb. 8.3.2: Menüpunkte des CANscope und Unterpunkte des Menüs *File*

File

Store Actual Settings

Der Aufruf dieses Menüeintrags führt zum Speichern der aktuellen Einstellungen als Default-Einstellungen. Wird CANscope das nächste mal aufgerufen, entsprechen alle Einstellungen denen, die zum Zeitpunkt des Aufrufs von *Store Actual Settings* eingetragen waren.

Load or Save CANscope Profile

Die vorgenommenen Einstellungen der CANscope-Parameter können in Profile-Dateien gespeichert werden, um sie zu einem späteren Zeitpunkt wieder zu laden.
Nach dem Laden kann ein automatischer Start erfolgen.

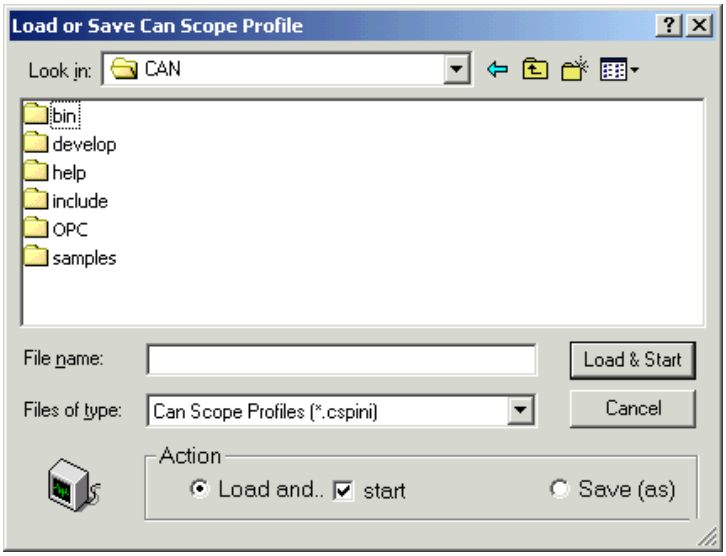


Abb. 8.3.3: Menü *Load or Save CANscope Profile*

Save Frames

Save Frames speichert alle im Anzeigefenster eingetragenen Frames (max. 2048). Sollen darüber hinaus auch die Frames aufgezeichnet werden, die normalerweise beim Überlauf des Anzeigefensters gelöscht werden, so muß die Aufzeichnung über *Start File Logging* erfolgen.

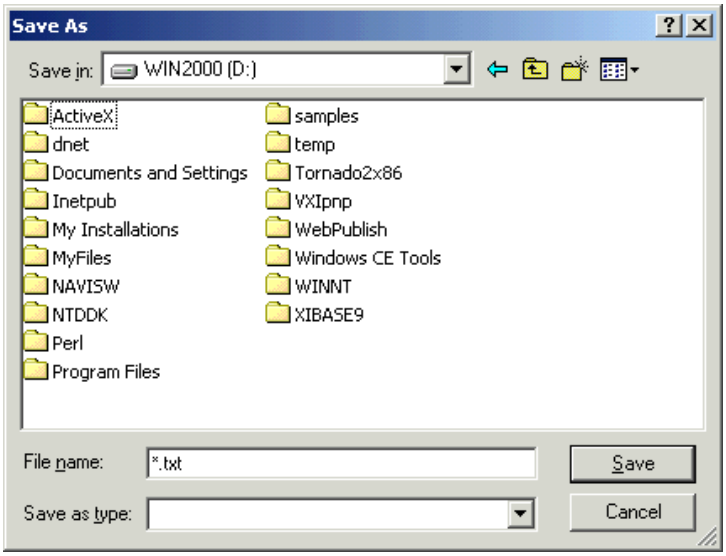


Abb. 8.3.4: Menü *Save As*

Start File Logging

Diese Funktion zeichnet alle empfangenen CAN-Frames in Text-Dateien auf.
Wird die Funktion *Limit File Size* aktiviert, so erfolgt die Aufzeichnung nicht in einer einzelnen Datei, sondern in mehreren Dateien. Die Dateinamen setzen sich hier aus einem frei wählbaren Namen und einer fortlaufenden Nummer zusammen:
filename`00001.txt` ...
filename`nnnnn.txt`.
Die einzugebende Dateigröße wird dabei mit einer Genauigkeit von ±20% erreicht. Wird *Disable Message List* aktiviert, so erfolgt nur die Aufzeichnung in dem angegebenen Log-file(s) und die Ausgabe der Frames im Anzeigefenster entfällt. Die Ausgabe der Frames im Anzeigefenster benötigt viel Rechenleistung. Falls festgestellt wird, daß bei hohen Datenraten nicht mehr alle Frames aufgezeichnet werden, kann auf diesem Wege weitere Rechenkapazität zur Aufzeichnung verfügbar gemacht werden. Das Programm meldet verlorengegangene Frames über einen Message-Lost-Zähler unterhalb des Fensters *Active ID's*.

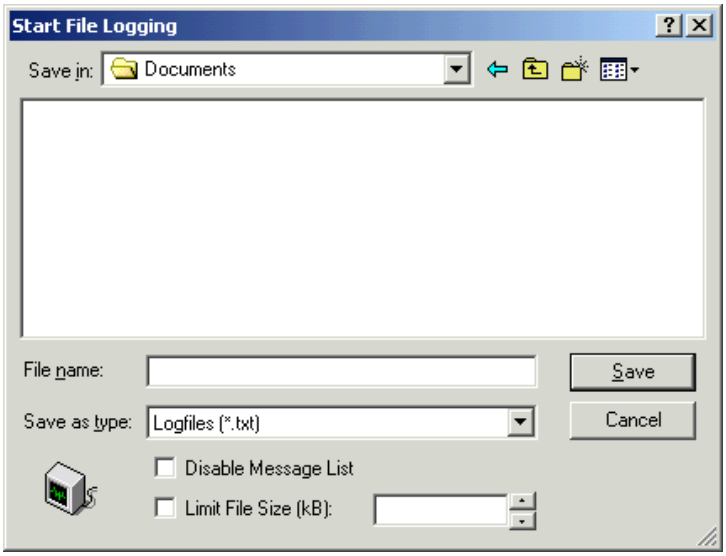


Abb. 8.3.5: Menü *Start File Logging*

Anmerkung: Das File-Logging benötigt viel Rechenkapazität. Bei großen CAN-Datenraten und langsamen PCs können sich daher der Bildaufbau und das Ansprechverhalten des PCs verschlechtern.

Beispielhafter Auszug eines Logfiles:

```
; Date : 03 21 2000
; Time : 10 21 47
; Netnumber : 0
; Baudrate : 1
;Frm.-No. Time    ID       RTR    Len    Can Message          (text)

00001    94303   0000            0

00002    00351   0000            0

00003    10905   1000            3      11      22      33    _"3
00004    00331   1000            3      11      22      33    _"3
00005    00190   1000            3      11      22      33    _"3
00006    00170   1000            3      11      22      33    _"3
00007    00190   1000            3      11      22      33    _"3
00008    00421   1000            3      11      22      33    _"3

00009    45415   1000      R     0
00010    00421   1000      R     0

00008    10:32:13.129    0000           4      AA      BB      CC      DD
```

View

Page Scroll

Wird diese Funktion aktiviert, so wird die Wiederholrate des Bildaufbaus verringert.

CAN

Enable 29-Bit-IDs

Wird diese Funktion aktiviert, so können mit CANscope auch Frames gesendet und empfangen werden, die 29-Bit lange CAN-Identifier besitzen. Voraussetzung hierfür ist selbstverständlich, daß die verwendete Hardware und der CAN-Treiber in der Implementierung für das eingesetztes Betriebssystem dafür ausgelegt sind! Welche Hardware-Module 29-Bit-Identifier unterstützen, können Sie auf Seite 13 nachlesen. Sollte die Implementierung des Treibers für ein bestimmtes Betriebssystem die 29-Bit-Identifier nicht unterstützen, so ist dies im Anhang bei den Installationshinweisen zu dem Betriebssystem abgedruckt.

Ist die Funktion aktiviert, so erscheint in der Eingabezeile für Sendenachrichten ein weiterer Auswahlpunkt *29-Bit*. Zum Senden von 29-Bit Frames muß er aktiviert sein.

8.4 Beispiel

Ziel dieses Beispiels ist es, auf den Identifiern 0x0E (14 dez) und 0x0F (15 dez) Daten zu empfangen. Die Daten sollen durch Senden einer RTR-Nachricht angefordert werden.

Das Beispiel ist nur funktionsfähig, wenn ein CAN-Teilnehmer mit der passenden Baudrate angeschlossen ist und der verwendete Identifier in der Lage ist auf mit Remote-Requests zu antworten!

Vorgehensweise:

1. Nach dem Programmaufruf die Schaltfläche *Start* drücken (initialisiert das Modul)

2. Falls notwendig, die Werte für Baudrate und Netznummer ändern.

3. Nicht (!) *ID's decimal* auswählen ! Nicht *Static View* auswählen !

4. In die Felder *Low Limit* und *High Limit* 'E' und 'F' (ohne führendes 0x) eintragen.

5. Starten.
   Empfangene Nachrichten auf 0x0E oder 0x0F werden jetzt dargestellt.

6. In der Eingabezeile für Sendenachrichten in das *ID*-Feld E (ohne führendes 0x) eintragen, in das *RTR*-Feld '1' und in das *Len*-Feld einen beliebigen Wert eintragen.

7. Nach Drücken der *Send*-Schaltfläche sind die CAN-Nachrichten im Nachrichten-Fenster zu sehen.

8. Wird jetzt die dezimale Darstellung aktiviert, werden sämtliche relevanten Werte in den Eingabeboxen und Darstellungsfenstern erneuert und die ID's der ab diesem Zeitpunkt empfangenen Nachrichten dezimal dargestellt.

9. Initialisierungs-Tool CANbatch

9.1 Übersicht

Das Programm CANbatch kann nur unter den Windows-Betriebssystemen (Windows NT/2000/XP, Windows 95/98/ME) betrieben werden !
29-Bit CAN-Identifier werden von dem Programm CANbatch zur Zeit noch nicht unterstütz.

CANbatch wurde entwickelt, um während der Test- und Entwicklungsphase von Projekten die Initialisierung der *anderen* CAN-Teilnehmer zu erleichtern. Typische Anwendungen sind die Initialisierung von CANopen-Modulen oder CAN-I/O-Modulen, die mit dem sogenannten ‘esd-Protokoll’ arbeiten.

Anmerkung: CANbatch dient *nicht* der Initialisierung des CAN-PC-Moduls!

Das Programm kann eine Liste von CAN-Nachrichten sequentiell senden und empfangene Antwort-Frames anzeigen.

9.2 Konfigurationsdateien

Die zu sendenden CAN-Nachrichten werden in Konfigurationsdateien im ASCII-Format abgelegt. Die Konfigurationsdateien sind frei konfigurierbar. Sie können beliebig viele verschiedene CAN-Nachrichten enthalten. CAN-Identifier, Länge, RTR-Bit und Daten können frei gewählt werden.

CANbatch bietet außerdem die Möglichkeit, das Senden der CAN-Nachrichten über einfache Programmschleifen zyklisch zu wiederholen und Wartezeiten zwischen den Nachrichten einzufügen.

9.2.1 Syntax

- Alle Zahlenwerte sind als Dezimalzahlen anzugeben.
- Kommentare werden durch ein Semikolon eingeleitet. Alles, was zwischen einem Semikolon und dem Zeilenende steht, wird als Kommentar gewertet.
- Die Gliederung der Datei erfolgt durch die Schlüsselwörter `[COMMON]` und `[PARAMETER]`, die in Eckigen Klammern stehen und in Großbuchstaben geschrieben sein müssen.
- Bei allen anderen Einträgen sind Groß- und Kleinbuchstaben gleichberechtigt und lassen sich beliebig kombinieren.

9.2.2 Aufbau der Konfigurationsdatei

Eine Konfigurationsdatei besteht aus zwei Teilen: den allgemeinen Definitionen `[COMMON]` und dem Parameterteil `[PARAMETER]`.

COMMON

In `[COMMON]` werden Parameter abgelegt, die für alle CAN-Nachrichten dieser Konfigurationsdatei gültig sind. Jedem der folgenden Parameter muß über ein Gleichheitszeichen ein Wert zugewiesen werden.

`Modulnummer` Hier kann bei Modulen, die mit dem esd-Protokoll arbeiten die Module-No. des Moduls eingetragen werden.
Für CANopen-Module ist dieser Eintrag ohne Bedeutung.

`Baudrate` Dem Parameter Baudrate können über 14 verschiedene Zahlenwerte ebensoviele verschiedene Baudraten zugewiesen werden. Die Zuordnung der Zahlenwerte zu den Baudraten ist in der Tabelle auf Seite 18 abgedruckt.

`Netznummer` Über diesen Parameter wird die logische Netz-Nummer des CAN-PC-Moduls (0, 1, 2 ...) zugewiesen.

`CTXID` Hier wird der Tx-Identifier eingetragen, auf dem der angesprochene CAN-Teilnehmer seine Antworten senden soll. Für das CAN-Modul, für das CANbatch aufgerufen wurde, ist dies ein Rx-Identifier, weil es die Antwort auf diesem Identifier empfängt.

`Typ` Mit diesem Parameter wird ausgewählt, ob die Nachrichten für ein esd-CAN-I/O-Modul mit ‘esd-Protocol’ oder für ein CANopen-Modul ausgelegt werden sollen. CANbatch formatiert die Nachrichten für das selektierte Übertragungsprotokoll und führt automatisch die erforderlichen Handshake-Sequenzen durch.
Zulässige Einträge: `CANopen` oder `esd`

In den folgenden Kapiteln sind Beispiele von Konfigurationsdateien aufgeführt, in denen der `[COMMON]`-Teil dargestellt ist.

PARAMETER

In dem Abschnitt `[PARAMETER]` werden die zu sendenden Nachrichten eingetragen. Der grundsätzliche Aufbau der Einträge ist wie folgt:

CANbatch-Command	ID	Length	data 1	data 2	data 3	data 4	data 5	data 6	data 7	data 8
`SEND` oder `RTR`	0...2047	0...8	zu sendende CAN-Daten							

Tabelle 9.2.1: Aufbau einer Parameterzeile für Sendeaufträge

weitere gültige CANbatch-Command-Einträge:

CANbatch-Command	Parameter TIME [ms]
`WAIT` `PAUSE`	0...65535

Tabelle 9.2.2: Parameterzeileneintrag für zeitweise Programmunterbrechungen

Der Eintrag '`WAIT, xx`' oder '`PAUSE, xx`' hält die Abarbeitung der Batch-Datei für die angegebene Zeit *xx* in ms an. Wird für *xx* der Wert '0' eingetragen, so wartet die Funktion, bis die Schaltfläche 'Weiter' gedrückt wird. `Wait` und `Pause` können alternativ verwendet werden.

CANbatch-Command	Parameter
`REPEAT` `END`	keine

Tabelle 9.2.3: Parameterzeileneintrag für Programmschleifen

Mit `REPEAT` läßt sich eine einfache Programmschleife aufbauen. Die Schleife wird über den Eintrag `END` geschlossen. Sie wird endlos durchlaufen und kann nur durch Drücken der Schaltfläche 'Stop' angehalten werden.

Die Art und Anzahl der zu übergebenden Parameter hinter dem Kommando `SEND` und `RTR` hängt davon ab, ob unter `Typ` ein Modul mit 'esd-Protocol' oder ein CANopen-Modul ausgewählt wurde.

Parameter für esd-CAN-Protocol-Module:

CANbatch-Command	ID	Length	data 1	data 2	data 3	data 4	data 5	data 6	data 7	data 8
`SEND` oder `RTR`	700 hex	6	Command	Sub-Command 1	Sub-Command 2	Module No.	Para-meter 1	Para-meter 2	-	-

Tabelle 9.2.4: Aufbau einer Parameterzeile für esd-CAN-Protokoll-Module

ID — Hier wird der Initialisierungs-Identifier (INIT-ID) des esd-CAN-Protocols eingetragen. Er hat immer den Wert `$700`.

Length — Die Datenlänge muß bei der Kommandoübergabe des esd-Protocols immer 6 Bytes betragen!

data 1 — Hier wird das `Command`-Byte gemäß esd-CAN-Protocol eingetragen.

data 2, data 3 — Hier werden die `Sub-Commands` gemäß esd-CAN-Protocol eingetragen.

data4 — Die `Module No.` selektiert während der Initialisierung das angesprochene esd-CAN-I/O-Modul auf dem CAN-Bus. Werden hier statt einer Nummer die Buchstaben '`MN`' eingetragen, so wird die Modul-Nummer verwendet, die im Menüfenster unter *Modul-Nr.* eingetragen ist.

data 5, data 6 — Hier werden die Inhalte der Initialisierungsparameter eingetragen.

Für das esd-CAN-Protocol ist ein eigenes Handbuch erhältlich. Dort werden die Kommandos und Parameter ausführlich beschrieben. Das esd-CAN-Protocol ist nur mit esd-CAN-I/O-Modulen lauffähig!

Beispiel einer Konfigurationsdatei für CAN-I/O-Module mit esd-CAN-Protocol:

```
; ---------------------------------------------
; DATEI: config.txt                           ;
; Konfigurationsdatei fuer CANBatch.EXE       ;
; ---------------------------------------------
; Projekt:                                    ;
; Version:                                    ;
;   Datum:                                    ;
;   Autor:                                    ;
;   Modul:   CPIO                             ;
;   EPROM:   >pio81fdg                        ;
;    Mode:   Mode 4                           ;
; ---------------------------------------------
;
[COMMON]
Modulnummer=1     ; vorgewählte Modulnummer
Baudrate=6 ; CAN-Baudrate, hier z.B. 125 kBit/s gewählt
Netznummer=0      ; Netznummer der PC-CAN-Karte
CTXID=300         ; Rückmelde-ID der CAN-Module = Variable MN
Typ=esd           ; gewählter Modultyp [esd, CANopen]
;
;
[PARAMETER]
;Request Configuration
;
;
; MUSTEREINTRAEGE FUER ESD-MODUS
; WAIT
; Beispiel 100 msec Pause einlegen
;Wait, 100

; CONFIG
; Byte-Aufbau / Werte in Hex-2-Darstellung angeben
; Variable Modulnummer Variable "MN" einsetzbar
; SEND cid l cc s1 s2 mm p1 <p2> <r1 r2> ;
; |    |   |  |  |  |  |  |    |   reserviert (optional)
; |    |   |  |  |  |  |  |   parameter2 (optional)
; |    |   |  |  |  |  |  parameter1
; |    |   |  |  |  |  Modulnummer
; |    |   |  |  |  Subkommando2
; |    |   |  |  Subkommando1
; |    |   |  Kommando
; |    |   Kommando-Laenge (= 6 fuer cc s1 s2 mm p1 p2)
; |    cid = config_id = 700 fuer esd-can-Module
; SEND oder RTR(remote request)
;
;
; für esd-CAN-Module:
; Kommando: 8x = Wert Parameter setzen
;           0x = Wert Parameter abfragen/anzeigen
;
; Beispiel:
;
; default_can
SEND 700 6 FF 03 00 MN AA AA   ; default_can Modul MN
; sende 6 Datenbyte mit ID 700, Daten: FF 03 00 MN AA AA
;
Wait, 15000       ; 15 sec Pause fuer Modul
;
; Ende der Beispieldatei "config.txt"
```

Parameter für CANopen-Module:

CANbatch-Command	ID	Length	data 1	data 2	data 3	data 4	data 5	data 6	data 7	data 8
SEND oder RTR	CAN-Identifier	Anzahl der Daten-Bytes	Command	Index-Low	Index-High	Sub-Index	Parameter-Byte LL	Parameter-Byte LH	Parameter-Byte HL	Parameter-Byte HH

Tabelle 9.2.5: Aufbau einer Parameterzeile für CANopen-Module

ID Hier wird der CAN-Identifier des CANopen-Zugriffs eingetragen.

Length Die Anzahl der Daten-Bytes muß immer auf den Wert ‘8’eingestellt werden!

data 1 Hier wird das CANopen-Command-Byte eingetragen (z.B.: 40 = SDO lesen, 23 = SDO schreiben).

data 2...4 Diese Bytes enthalten den Index und Sub-Index der CANopen-Nachricht.

data 5...8 Diese Bytes enthalten die Parameter-Daten. Bitte beachten sich die Byte-Orientierung!

Beispiel einer CANopen-Konfigurationsdatei:

```
; CANopen ModulNummer 2
; Ping auf ein Modul mit SDO 1000, bzw 1001
[COMMON]
Modulnummer=2       ; vorgewählte Modulnummer
Baudrate=6        ; 125 kBit
Netznummer=0      ; Netznummer der PC-CAN-Karte
CTXID=582         ; Rückmelde-ID der CAN-Module = Variable MN
Typ=CANopen  ; gewählter Modultyp [esd, CANopen]
;
[PARAMETER]
; SDO schreiben/lesen:
; SEND cid len cs I.L I.H S p.LL p.LH p.HL p.HH ;
; |      |   |  |   |   |  |  |    |    |    |
; |      |   |  |   |   |  |  |  Parameter (LL-HH)
; |      |   |  |   |   |  |  Subindex
; |      |   |  |   |   |  Index.H
; |      |   |  |   |  Index.L
; |      |   |  |  Kommando (40 = SDO lesen, 23 = SDO schreiben)
; |      |   |  Kommando-Laenge = 8 (+10 um ein Response anzuzeigen)
; |      |  cid = 602 für CANopen Module Nr. 2
; SEND oder RTR(remote request)
;
SEND 00 2 01 00                       ; preoperational->operational
Wait, 1000
SEND 602 18 40 00 10 00 00 00 00 00  ; "device type" lesen
Wait, 1000
SEND 602 18 40 01 10 00 00 00 00 00  ; "error register" lesen
Wait, 1000
SEND 602 18 40 08 10 00 00 00 00 00  ; manufacturer device name
Wait, 1000
SEND 602 18 40 09 10 00 00 00 00 00  ; hw version
Wait, 1000
SEND 602 18 40 0A 10 00 00 00 00 00  ; sw version
Wait, 1000
; SEND 00 2 80 00   ; operational->preoperational
```

9.3 CANbatch-Bedienoberfläche

9.3.1 Darstellung des CANbatch-Fensters

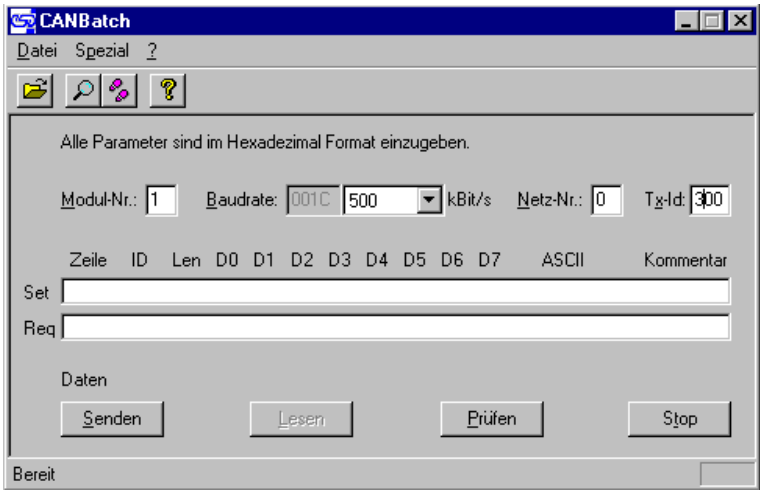


Abb. 9.3.1: CANbatch-Fenster

9.3.2 Bedienelemente

Set- und Request-Zeile

In der Set-Zeile werden die zu sendenden CAN-Nachrichten angezeigt. Hier ist nach dem Laden einer Konfigurationsdatei und vor dem Senden die erste Zeile der Konfigurationsdatei zu sehen.
Nach dem Drücken der Schaltfläche ‘Senden’ erscheinen nacheinander alle Sendenachrichten in dieser Zeile. Ist nicht der Einzelschrittmodus aktiviert oder sind Wartezeiten zwischen den CANbatch-Kommandos eingetragen, ist die Abarbeitung in der Regel so schnell, daß dies kaum wahrgenommen werden kann.

In der Request (Req)-Zeile werden die Antwort-Frames, die ggf. auf eine Sendenachricht folgen, angezeigt.
Um zu jeder Sendenachricht die entsprechende Antwort in der Request-Zeile lesen zu können, müssen Sie den Einzelschritt-Modus auswählen oder Wartezeiten zwischen den Sendekommandos einfügen.

Button Bar

In der Schaltflächenleiste finden Sie zur Zeit vier Symbole:

Karteikarte: Datei öffnen
Durch Drücken der linken Schaltfläche öffnet sich ein Menüfenster. Sie können eine Konfigurationsdatei laden. Diese Schaltfläche hat die selbe Funktion wie der Menüpunkt 'Datei'/'Öffnen'.

Module suchen mit der Lupe
Nach dem Drücken dieser Schaltfläche öffnet sich ein Menüfenster. Hier kann die Suche nach den verfügbaren CANopen-Modulen oder CAN-I/O-Modulen mit esd-CAN-Protocol gestartet werden. Alle von CANbatch gefundenen Module werden angezeigt.

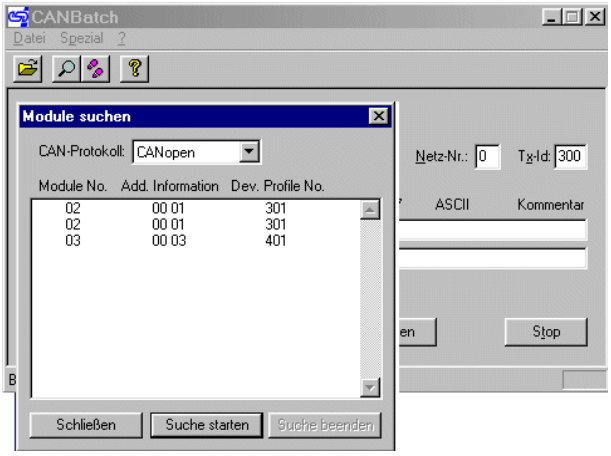


Abb. 9.3.2: Suche nach CAN-Modulen

Einzelschrittmodus
Der Einzelschrittmodus wird durch Drücken der Schaltfläche mit den beiden Fußabdrücken aktiviert. Der nächste Schritt (das nächste Sendekommando) wird jeweils durch Drücken der Schaltfläche 'Einzelschritt', die anstelle der Schaltfläche 'Senden' erscheint, aktiviert. Während des Betriebs kann vom Einzelschrittmodus in den normalen Programmablauf gewechselt werden. Alle folgenden Kommandos werden dann entsprechend der Konfigurationsdatei gesendet.

Modul-Nr., Baudrate, Netz-Nr., TxId

Modul-Nr. Hier wird die Modul-Nr. des anzusprechenden esd-CAN-I/O-Moduls eingetragen. Welche Modul-Nr. das CAN-Modul hat und wie die Modul-Nr. am CAN-Modul verändert wird, entnehmen Sie bitte dem Handbuch des esd-CAN-I/O-Moduls.

Baudrate In diesem Auswahlfeld wird die Baudrate gewählt. Es sind Baudraten von 10 kBit/s bis 1 MBit/s möglich.

Netz-Nr. Stehen mehrere CAN-Netze zur Verfügung, kann hier die Netz-Nummer eingestellt werden.
Bei der ersten Karte im PC sind dieses normalerweise die Netznummern 0 und 1 (bei einer Ein-Kanal-Karte nur die 0), bei jeder weiteren Karte sind die Netznummern entsprechend aufsteigend (2 und 3, 4 und 5 ... bei Ein-Kanal-Karten werden nur die geradzahligen Nummern gezählt: 2, 4, ...).

TxId Hier wird der CAN-Identifier eingetragen, auf dem die Antwort-Frames empfangen werden sollen. Die Antwort-Frames werden in der Zeile ‘Req.’ angezeigt.

9.3.3 Menüs

Datei In diesem Menü kann das Programm beendet, oder eine Konfigurationsdatei geladen werden.

Spezial Hier kann unter dem Eintrag ‘Module suchen’ die Suche nach den verfügbaren CANopen-Modulen oder CAN-I/O-Modulen mit esd-CAN-Protocol gestartet werden. Alle von CANbatch gefundenen Module werden angezeigt.

Über den Eintrag ‘Einzelschritt-Modus’ wird die manuelle Abarbeitung der Konfigurationsdatei in Schritten aktiviert. Der jeweils nächste Schritt (das nächste Sendekommando) wird durch Drücken der Schaltfläche ‘Einzelschritt’, die anstelle der Schaltfläche ‘Senden’ erscheint, aktiviert.

Diese Seite ist bewußt unbedruckt.

Anhang - Installation und Implementation

Inhalt

Seite

A 1. Wegweiser

Dieses Dokument ist für eine Vielzahl von esd-CAN-Boards gültig. Da die Boards teilweise als Platinen, teilweise aber auch in abgeschlossenen Gehäusen (z.B. CAN-USB-Mini) geliefert werden, wird im folgenden bei allgemeingültigen Aussagen von *Modulen* gesprochen.

Die in diesem Anhang beschriebenen Software-Installationshinweise sind abhängig vom jeweils eingesetzten Modul und dem Betriebssystem.

Die Installation der Hardware ist der Modul-spezifischen Dokumentation 'Hardware-Installation und technische Daten' zu entnehmen.

Die CAN-Software wird auf CD-ROM oder auf Disketten ausgeliefert. Erfolgt die Lieferung auf CD-ROM, so sind die Dateien in entsprechenden Unterverzeichnissen abgelegt. Bei Lieferung auf Disketten, befinden sich der Modul-spezifische CAN-Treiber, die DLL und die Konfigurationsprogramme auf einer Diskette. Auf einer zweiten Diskette befindet sich das SDK (Software Development Kit) mit den Dateien die zur Entwicklung und zum Test eigener Applikationen notwendig sind. Auf dieser Diskette finden Sie auch CANscope, CANbatch und CANtest.

Bitte gehen Sie bei der Installation der esd-CAN-Module (Ausnahme CAN-Bluetooth) wie folgt vor:

1. **Installation des Treibers**
   Suchen Sie in der Tabelle auf der folgenden Seite die Spalte mit 'Ihrem' Betriebssystem. In den Zeilen der Tabelle finden Sie Ihr CAN-Modul. Diese beiden Informationen führen Sie zu dem Feld, in dem die Kapitel aufgelistet sind, die Sie für Ihre Installation benötigen.

   Die ersten Kapitel beschreiben die Installation des Treibers.

   Besitzen Sie ein Windows-Betriebssystem, so müssen Sie noch den zweiten Schritt, die Installation des SDKs, durchführen. Für alle anderen Betriebssysteme ist dies nicht notwendig.

2. **Installation des SDK (Software Development Kit - nur für Windows-Betriebssysteme)**
   Nach der Treiber-Installation fahren Sie mit der Installation des SDK fort. Die SDK-Installation ist in dem gleichnamigen Kapitel ab Seite A-36 beschrieben. Sie ist nur für Windows-Betriebssysteme notwendig.

Achtung: Die Installation des CAN-Bluetooth-Treibers bildet hier eine Ausnahme! Der CAN-Bluetooth–Treiber ist generell zuletzt, also nach der Installation etwaiger anderer CAN-Treiber und nach der Installation des CAN-SDK zu installieren !

Die folgende Tabelle soll Ihnen das Auffinden der für Sie zutreffenden Installationsbeschreibung erleichtern:

CAN-Modul		Zutreffende Kapitel des Anhangs für **Windows-Betriebssysteme**		
		Windows NT	Windows 2000 Windows XP	Windows 95 Windows 98 Windows ME
CAN-ISA/200	im folgenden als **'ISA-Bus-Boards'** bezeichnet	A2.1, A2.5, A2.7	A2.2.2, A2.5, A2.7	A2.3.2, A2.5, A2.7
CAN-PC104/200 (nur SJA1000)				
CAN-ISA/331		A2.1, A2.5- A2.7	A2.2.2, A2.5- A2.7	A2.3.2, A2.5- A2.7
CAN-PC104/331				
CAN-PCI/200	im folgenden als **'PCI-Bus-Boards'** bezeichnet	A2.1, A2.5, A2.7	A2.2, A2.5, A2.7	A2.3.1, A2.5, A2.7
CAN-PCI/331		A2.1, A2.5- A2.7	A2.2, A2.5- A2.7	A2.3.1, A2.5- A2.7
CPCI-CAN/331				
PMC-CAN/331				
CPCI-CAN/360				
CAN-PCI/360				
CAN-USB-Mini		-	A2.2.1, A2.5- A2.7	A2.3.2, A2.5- A2.7
CAN-Bluetooth		-	A2.4, A2.5, A2.7	A2.4, A2.5, A2.7
CAN-PCC		A2.1, A2.5- A2.7	A2.2.2, A2.5- A2.7	A2.3.2, A2.5- A2.7

- ... zur Zeit noch keine Implementierung vorhanden

Tabelle A1.1.1: Matrix der für die Windows-Installation relevanten Kapitel

<table>
<tr><th colspan="2" rowspan="2">CAN-Modul</th><th colspan="9">Zutreffende Kapitel des Anhangs für alle nicht-Windows-Betriebssysteme</th></tr>
<tr><th>Linux</th><th>Linux-RTAI/RT</th><th>LynxOS</th><th>Power-MAX OS</th><th>Solaris</th><th>SGI-IRIX6.5</th><th>AIX</th><th>VxWorks</th><th>QNX4</th></tr>
<tr><td>CAN-ISA/200</td><td rowspan="4">im folgenden als 'ISA-Bus-Boards' bezeichnet</td><td rowspan="2">A3.1</td><td rowspan="2">A3.2</td><td rowspan="2">-</td><td rowspan="2">-</td><td rowspan="2">-</td><td rowspan="2">-</td><td rowspan="2">-</td><td rowspan="2">A3.8</td><td rowspan="2">A3.9</td></tr>
<tr><td>CAN-PC104/200 (nur SJA1000)</td></tr>
<tr><td>CAN-ISA/331</td><td rowspan="2">A3.1, A3.10</td><td rowspan="2">A3.2, A3.10</td><td rowspan="2">A3.3, A3.10</td><td rowspan="2">-</td><td rowspan="2">A3.5, A3.10</td><td rowspan="2">-</td><td rowspan="2">-</td><td rowspan="2">A3.8, A3.10</td><td rowspan="2">A3.9, A3.10</td></tr>
<tr><td>CAN-PC104/331</td></tr>
<tr><td>CAN-PCI/200</td><td rowspan="6">im folgenden als 'PCI-Bus-Boards' bezeichnet</td><td>A3.1</td><td>A3.2</td><td>A3.3</td><td>-</td><td>-</td><td>-</td><td>-</td><td>A3.8</td><td>A3.9</td></tr>
<tr><td>CAN-PCI/331</td><td rowspan="4">A3.1, A3.10</td><td rowspan="4">A3.2, A3.10</td><td rowspan="3">A3.3, A3.10</td><td rowspan="5">-</td><td rowspan="3">A3.5, A3.10</td><td rowspan="3">A3.6, A3.10</td><td rowspan="3">A3.7</td><td rowspan="3">A3.8, A3.10</td><td rowspan="3">A3.9, A3.10</td></tr>
<tr><td>CPCI-CAN/331</td></tr>
<tr><td>PMC-CAN/331</td></tr>
<tr><td>CPCI-CAN/360</td><td rowspan="2">-</td><td rowspan="2">-</td><td rowspan="2">-</td><td rowspan="2">-</td><td rowspan="2">-</td></tr>
<tr><td>CAN-PCI/360</td></tr>
<tr><td colspan="2">CAN-USB-Mini</td><td>A3.1</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td colspan="2">CAN-Bluetooth</td><td>A3.1</td><td>A3.2</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td colspan="2">CAN-PCC</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td colspan="2">VME-CAN2</td><td>-</td><td>-</td><td>A3.3</td><td>A3.4</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
<tr><td colspan="2">VME-CAN4</td><td>-</td><td>-</td><td>A3.3</td><td>A3.4</td><td>-</td><td>-</td><td>-</td><td>-</td><td>-</td></tr>
</table>

Tabelle A1.1.2: Matrix der für die jeweilige Installation relevanten Kapitel

Hinweis: Für das Betriebssystem **RTOS-UH** sind Treiber für die CAN-Module CAN-ISA/331, CAN-PC104/331, CAN-PCI/331, CPCI-CAN/331, PMC-CAN/331 und VME-CAN2 verfügbar. Installationshinweise sind zur Zeit nicht verfügbar.

A 2. Windows-Betriebssysteme

A 2.1 Installation unter Windows NT

A 2.1.1 Installation der Software-Treiber für die Host-CPU

A 2.1.1.1 Starten der Installation

Bei einem Update muß die alte Version des Treibers, wie in Abschnitt A 2.1.1.5 beschrieben, zuvor deinstalliert werden.

Achtung: Die vollständige Installation, Konfiguration und der Start des Treibers kann nur mit Administrator-Rechten auf der Windows NT Maschine erfolgen !

Wird die Software auf Disketten ausgeliefert, so finden Sie den Gerätetreiber auf der Diskette mit der Bezeichnung 'CAN-Driver'. Bei Auslieferung auf CD-ROM finden Sie den Treiber in dem entsprechenden Unterverzeichnis.

Zur Installation muß der Datenträger (Diskette oder CD-ROM) mit dem Gerätetreiber in das Laufwerk eingelegt und das Installationsprogramm ausgeführt werden. Dies kann entweder durch direkten Aufruf des Programms `setup.exe` geschehen oder indem in der Dialogbox *Eigenschaften von Software* im Ordner *Systemsteuerung* der Knopf *Installieren* gedrückt wird.

Der sich daran anschließende Installationsprozeß ist dialoggesteuert. Im Laufe der Installation können Sie zwischen einer von drei Konfigurationen auswählen:

Konfiguration	Beschreibung
Typical	Treiber, DLL und Programm zum Update der lokalen Firmware
Compact	Treiber, DLL und Konfigurationsprogramme
Custom	Benutzerdefinierte Konfiguration

Tabelle A 2.1.1: Konfigurationsauswahl

Nach Abschluß der Treiber-Installation und vor Inbetriebnahme des Treibers muß der Rechner neu gestartet werden. Dies kann direkt aus dem Setup-Programm nach Abschluß der Installation erfolgen.

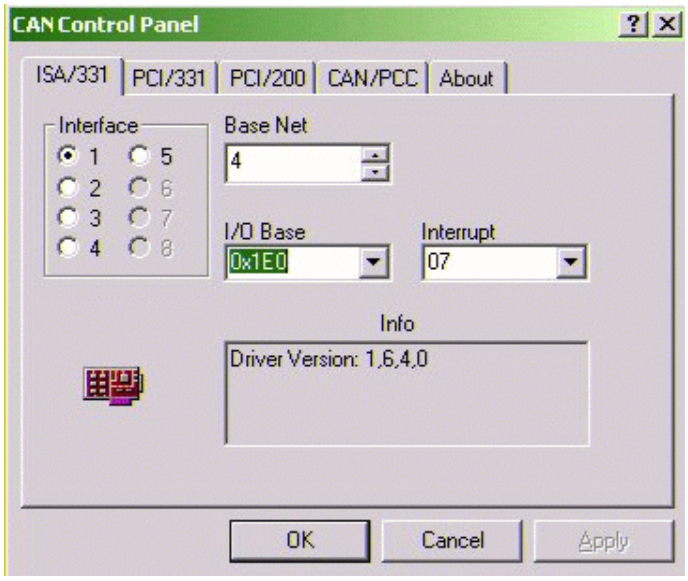
Installieren Sie anschließend das SDK, wie auf Seite A-36 beschrieben.

A 2.1.1.2 Aufruf des Konfigurationsprogramms CAN-Control

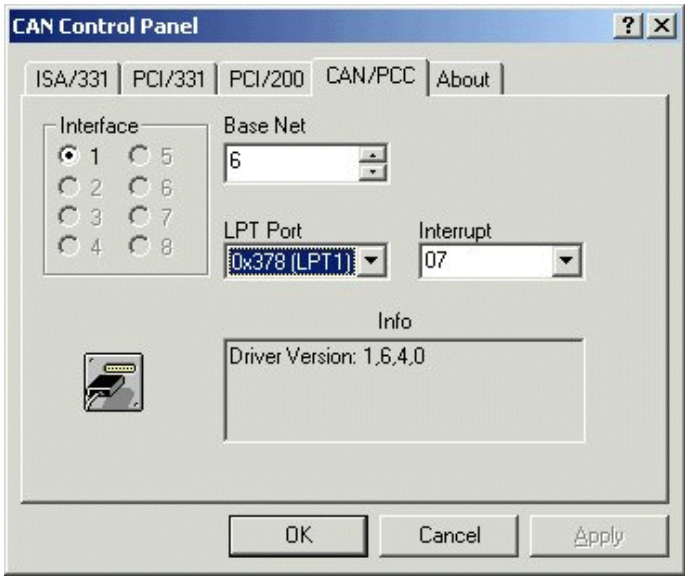
Zur Konfiguration des Treibers wird das Programm CAN-Control mitgeliefert. Es wird aus der Systemsteuerung heraus aufgerufen.

Die folgenden Abbildungen zeigen beispielhaft ein System mit vier esd-CAN-Modulen (ISA/331, PCI/331, PCI/200 und CAN-PCC). CAN-Control zeigt nur die esd-CAN-Module, die in dem System installiert sind. Haben Sie nur ein CAN-Modul installiert, so wird nur eine Karteikarte dargestellt.

Die folgende Abbildung zeigt die Einstellungsoptionen für CAN-**ISA**-Module:



Die folgende Abbildung zeigt die Einstellungsoptionen für das CAN-**PCC**-Modul:

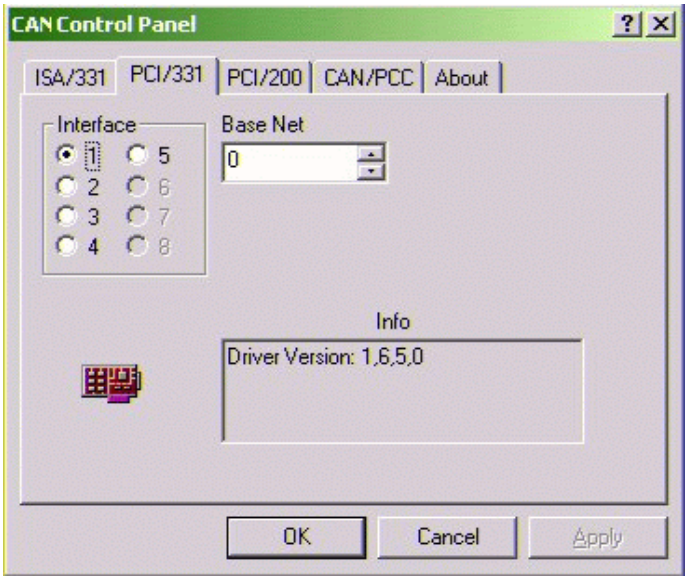


Im Feld *Interface* wird die Nummer des CAN-Moduls eingetragen, für das die Einstellungen gelten sollen. In *Base Net* wird dem CAN-Modul die Basis-Netz-Nummer zugeordnet. Hat ein Modul mehr als eine physikalisches CAN-Schnittstelle, so erhält die erste die in *Base Net* eingetragene logische Netz-Nummer und die folgenden werden davon ausgehend hochgezählt.

Achtung:	Der Anwender muß bei der Vergabe der logischen Netznummer dafür sorgen, daß keine Nummer doppelt vergeben wird!

Geben Sie für jedes CAN-Modul die von Ihnen vorgenommene Hardware-Konfiguration bezüglich der I/O-Port-Adresse ein, setzen Sie den Interrupt-Level und drücken Sie OK. Der Treiber ist nun installiert und konfiguriert.

Die folgende Abbildung zeigt die Einstellungsoptionen für CAN-**PCI**-Module. Hier erfolgt nur eine Zuordnung der logischen CAN-Netze:

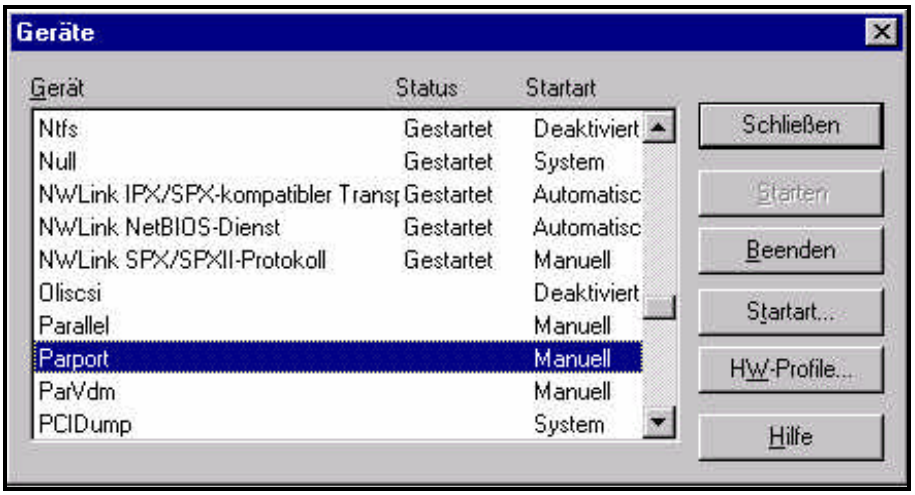


A 2.1.1.3 Nur CAN-PCC: Stoppen des Windows-NT-Treibers für die parallele Schnittstelle

Das CAN-PCC-Modul wird über die parallele Schnittstelle des Rechners betrieben. Standardmäßig greift der Windows-NT-Treiber auf diese Schnittstelle zu. Damit der CAN-Treiber ungehinderten Zugriff auf das parallele Port erhält, muß **vor dem Starten des CAN-Treibers** der Windows-NT-Treiber gestoppt werden. Hierzu ist wie folgt vorzugehen:

1. Unter *Einstellungen* wählen Sie *Systemsteuerung*.

2. Hier einen Doppelklick auf das Piktogramm *Geräte* ausführen. Das folgende Fenster erscheint:



3. In diesem Fenster muß nun das ‘Gerät’ *Parport* beendet werden. Daraufhin teilt das System mit, daß damit auch die Geräte *Parallel* und *ParVdm* beendet werden.

4. Damit diese Prozedur nicht mit jedem Systemhochlauf wiederholt werden muß, ist für alle drei Geräte *Parallel*, *Parport* und *ParVdm* die Startart auf *Manuell* zu setzen. Mit dieser Einstellung werden die Geräte nicht mehr automatisch mit dem Systemhochlauf gestartet.

A 2.1.1.4 Starten des Treibers

Zum Starten des Treibers sollte zunächst auf der Kommandozeile der Befehl `net start xxxx` eingegeben werden.

Für die verschiedenen CAN-Module werden die Treiber über folgende Aufrufe gestartet:

CAN-Modul	Kommandos zum Starten der Treiber
CAN-ISA/200	`net start c200i`
CAN-PC104/200	
CAN-ISA/331	`net start c331i`
CAN-PC104/331	
CAN-PCI/200	`net start c200`
CAN-PCI/331	`net start c331`
CPCI-CAN/331	
PMC-CAN/331	
CAN-PCI/360	`net start c360`
CPCI-CAN/360	
CAN-PCC	`net start cpcc`

Tabelle A 2.1.2: Kommandos zum Starten der Treiber

Erfolgt der Start des Treibers ohne Komplikationen, kann der Starttyp des CAN-Treibers im Dialog *Systemsteuerung/Geräte* von **Manuell** auf **Automatisch** gesetzt werden, so daß der Treiber nach jedem Kaltstart aktiviert wird und auch von anderen Benutzern ohne Administrator-Rechte genutzt werden kann.

Im Falle eines Problems beim Starten des Treibers kann die Fehlerursache der Ereignisanzeige von Windows NT 4.0 entnommen werden.

A 2.1.1.5 De-Installation der Software-Treiber

Die De-Installation des Treibers erfolgt in drei Schritten:

Achtung: Das Anhalten des Treibers und die vollständige De-Installation kann nur mit Administrator-Rechten auf der Windows NT Maschine erfolgen !

1. Beenden aller Applikationen, die die Dienste des Treibers nutzen

2. Der Treiber muß im Dialog *Systemsteuerung/Geräte* oder durch Eingabe von `net stop xxxx` auf der Kommandozeile angehalten werden

CAN-Modul	Kommandos zum Stoppen der Treiber
CAN-ISA/200	`net stop c200i`
CAN-PC104/200	
CAN-ISA/331	`net stop c331i`
CAN-PC104/331	
CAN-PCI/200	`net stop c200`
CAN-PCI/331	`net stop c331`
CPCI-CAN/331	
PMC-CAN/331	
CAN-PCI/360	`net stop c360`
CPCI-CAN/360	
CAN-PCC	`net stop cpcc`

Tabelle A 2.1.3: Kommandos zum Stoppen der Treiber

3. In der Dialogbox *Eigenschaften von Software* im Ordner *Systemsteuerung* den Eintrag für den CAN-Treiber auswählen und den Knopf *Hinzufügen/Entfernen* anwählen, damit alle Dateien und Registry-Einträge des Treibers gelöscht werden.

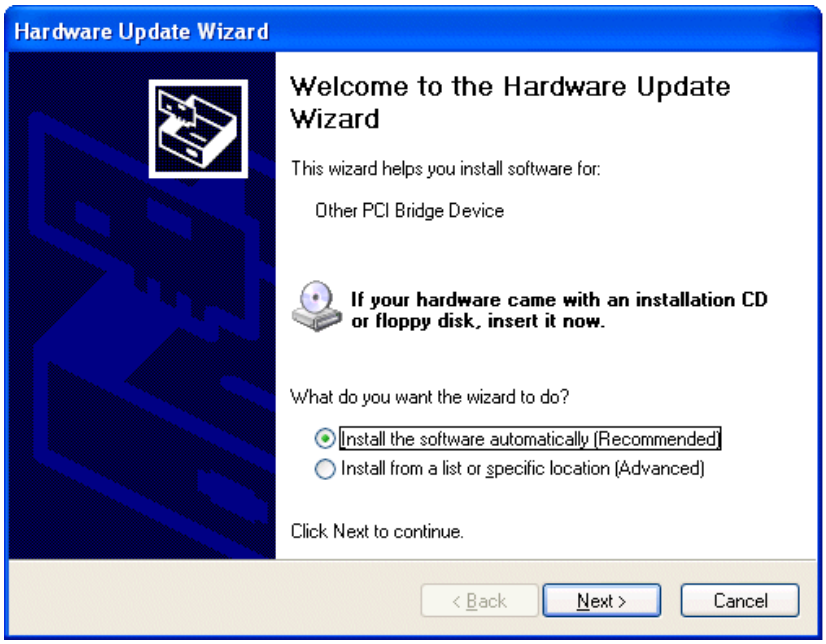
A 2.1.2 Ereignisverwaltung

Alle Fehlersituationen beim Start oder während des Betriebes werden in der Ereignisverwaltung von Windows NT 4.0 registriert.

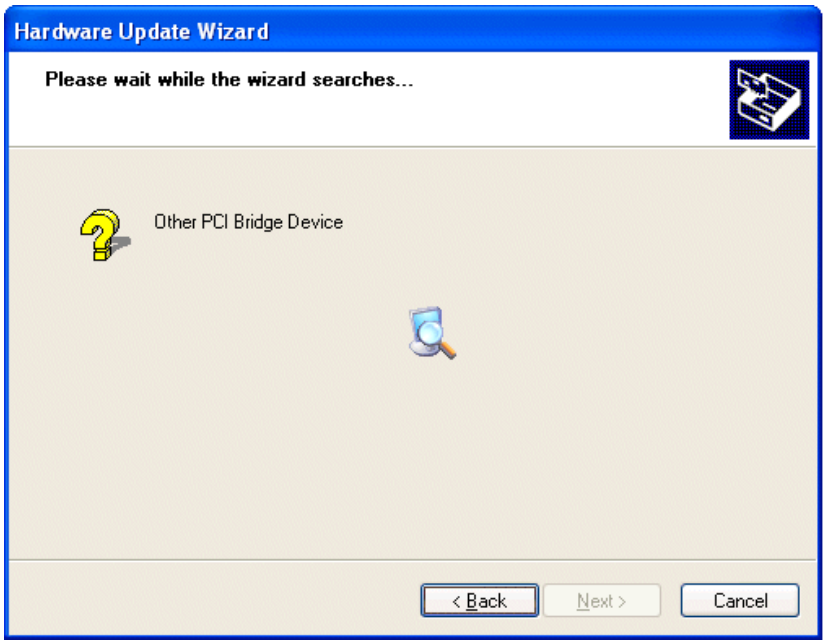
A 2.2 Installation unter Windows 2000/XP

A 2.2.1 Installation der PCI-Bus-Boards und des CAN-USB-Mini

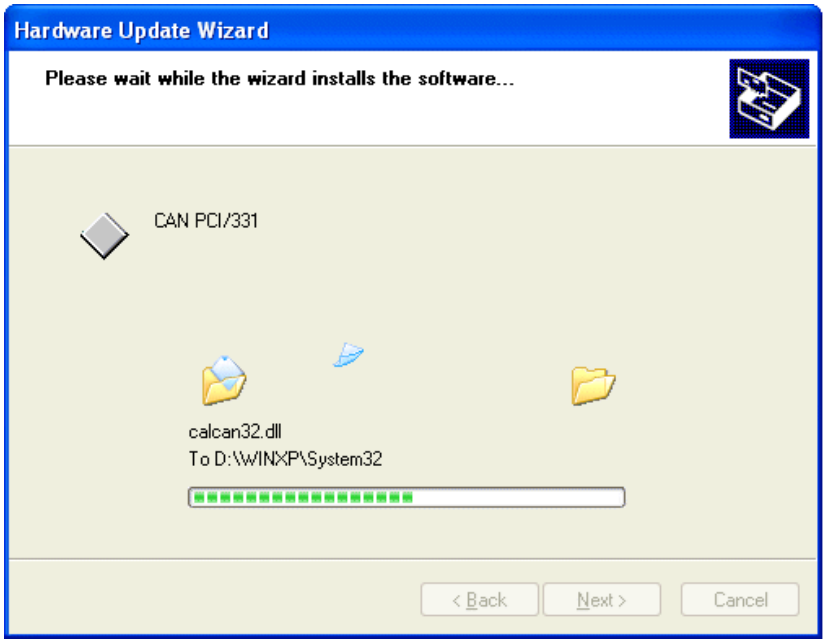
Installieren Sie zunächst das esd-CAN-Modul, wie im Hardware-Handbuch des Moduls beschrieben. Starten Sie dann Windows 2000/XP. Das System erkennt das neue CAN-Modul als 'Other PCI Bridge Device'.



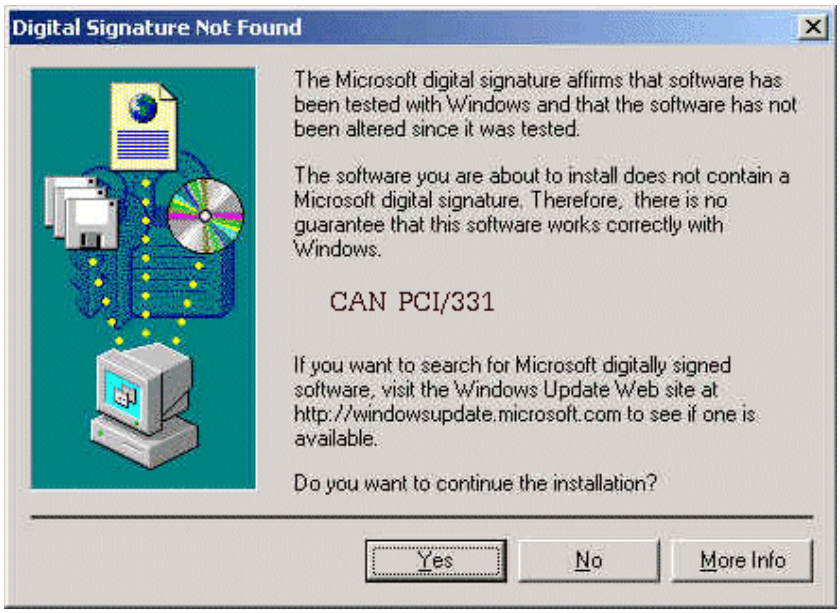
Legen Sie jetzt die CD-ROM oder Floppy-Disk mit dem CAN-Treiber in das entsprechende Laufwerk ein und drücken Sie auf *Next* (Weiter), um den Wizard (Assistenten) nach dem Treiber suchen zu lassen.



Der Wizard findet und installiert die Software.

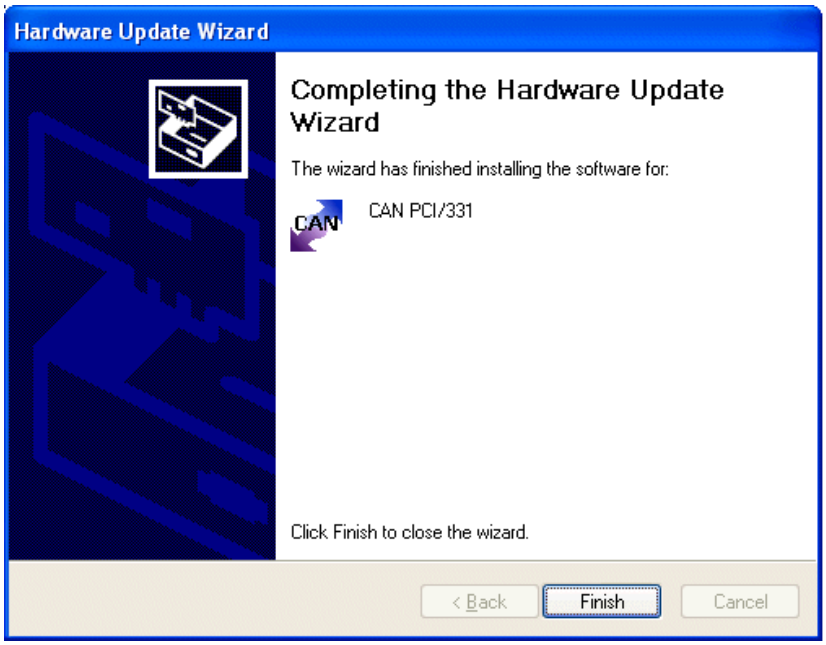


Nachdem Windows die Gültigkeit der *inf*-Datei akzeptiert hat, könnte sich das folgende besorgniserregende Menü öffnen...

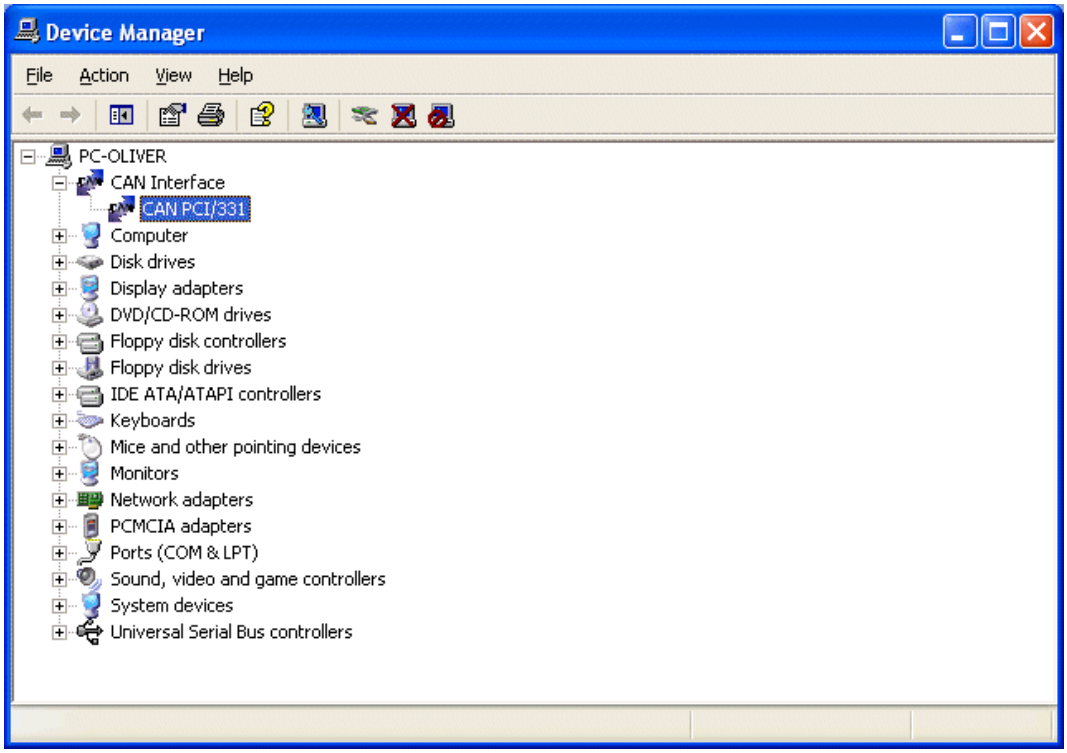


Microsoft hat den Treiber nicht auf die volle Kompatibilität zu Windows 2000 oder Windows XP getestet. Für den Fall, daß Sie den technischen Support von Microsoft kontaktieren müssen, kann es sein, daß der Kundendienstmitarbeiter Sie auffordert, alle 'unregistrierten', 'ungetesteten' Treiber von Ihrem Rechner zu entfernen.

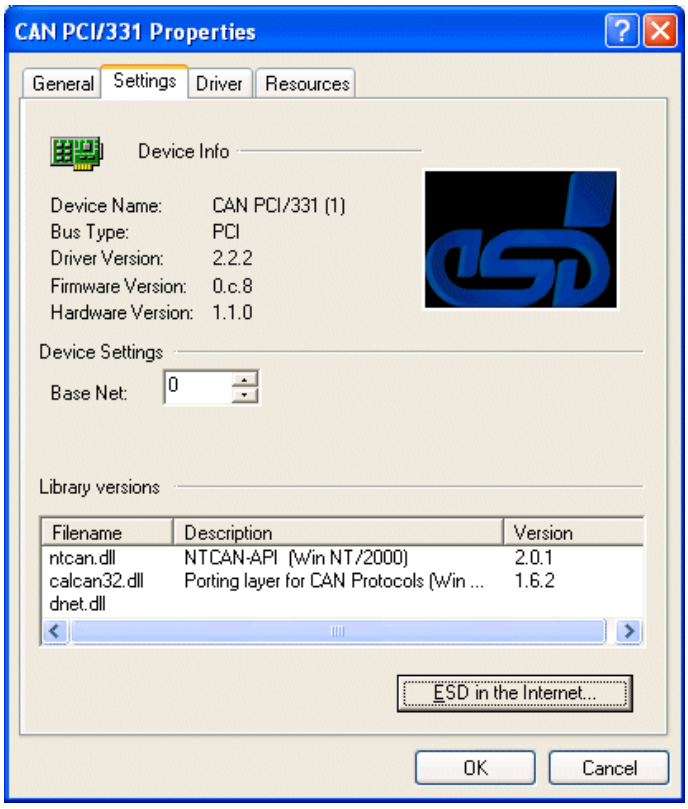
Wir wissen, daß der Treiber funktioniert. Sie können daher auf *Yes* drücken und mit der Installation fortfahren.



Nachdem die Installation abgeschlossen ist, starten Sie bitte Ihren Rechner neu und öffnen Sie den Device Manager (Geräte-Manager)



Das esd-CAN-Modul ist korrekt installiert (Beispiel hier: CAN-PCI/331). Ein Doppelklick auf den Eintrag des Moduls öffnet das Properties-Menü (Eigenschaften-Menü) des Treibers.



Im Eingabefeld *Base Net* wird dem CAN-Modul die Basis-Netz-Nummer zugeordnet. Hat ein Modul mehr als eine physikalisches CAN-Schnittstelle, so erhält die erste CAN-Schnittstelle die in *Base Net* eingetragene logische Netz-Nummer und die folgenden werden davon ausgehend hochgezählt.

Das Feld ist mit dem Wert ‘0’ vorbesetzt, so daß bei der ersten Karte im System die Netz-Nummern von ‘0’ aufsteigend vergeben werden.

Achtung:	Bei mehreren CAN-Modulen im System muß der Anwender bei der Vergabe der logischen Netznummer dafür sorgen, daß keine Nummer doppelt vergeben wird!

Die Treiber-Installation ist jetzt abgeschlossen. Installieren Sie anschließend das SDK, wie auf Seite A-36 beschrieben.

Hinweis:	Der CAN-Treiber der PCI-Boards und des CAN-USB-Moduls startet unter Windows 2000 automatisch mit dem Systemhochlauf.

Hinweis:	Das Deinstallieren des CAN-Treibers erfolgt über den Gerätemanager. Windows 2000 entfernt dabei nicht alle Dateien. Bleibt das CAN-Hardware-Modul angeschlossen, so würde Windows 2000 bei einem neuen Hochlauf das Gerät erkennen und versuchen, den Treiber wieder zu installieren.

A 2.2.2 Installation der ISA-Bus-Boards und der CAN-PCC unter Windows 2000

A 2.2.2.1 Installation der Software-Treiber für die Host-CPU

Hinweis:	Für den Betrieb der ISA-Bus-Boards und der CAN-PCC unter Windows 2000 werden die Windows NT-Treiber eingesetzt. Da es bei der Installation unter Windows 2000 einige Abweichungen gegenüber Windows NT gibt, ist die Installation der Treiber in diesem Kapitel explizit für Windows 2000 beschrieben.

A 2.2.2.1.1 Starten der Installation

Bei einem Update muß die alte Version des Treibers, wie auf Seite A-21 beschrieben, zuvor de-installiert werden.

Achtung:	Die vollständige Installation, Konfiguration und der Start des Treibers kann nur mit Administrator-Rechten auf der Windows 2000/XP Maschine erfolgen !

Wird die Software auf Disketten ausgeliefert, so finden Sie den Gerätetreiber auf der Diskette mit der Bezeichnung ‘CAN-Driver’. Bei Auslieferung auf CD-ROM finden Sie den Treiber in dem entsprechenden Unterverzeichnis.

Zur Installation muß der Datenträger (Diskette oder CD-ROM) mit dem Gerätetreiber in das Laufwerk eingelegt und das Installationsprogramm ausgeführt werden. Dies kann entweder durch direkten Aufruf des Programms `setup.exe` geschehen oder indem in der Dialogbox *Eigenschaften von Software* im Ordner *Systemsteuerung* der Knopf *Installieren* gedrückt wird.

Der sich daran anschließende Installationsprozeß ist dialoggesteuert. Im Laufe der Installation können Sie zwischen einer von drei Konfigurationen auswählen:

Konfiguration	Beschreibung
Typical	Treiber, DLL und Programm zum Update der lokalen Firmware
Compact	Treiber, DLL und Konfigurationsprogramme
Custom	Benutzerdefinierte Konfiguration

Tabelle A2.2.1: Konfigurationsauswahl

Nach Abschluß der Treiber-Installation vor Inbetriebnahme des Treibers muß der Rechner neu gestartet werden. Dies kann direkt aus dem Setup-Programm nach Abschluß der Installation erfolgen.

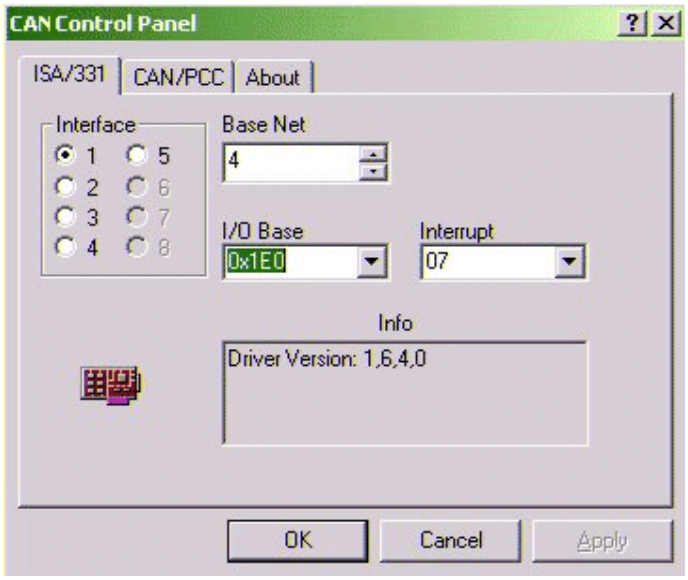
Installieren Sie anschließend das SDK, wie auf Seite A-36 beschrieben.

A 2.2.2.1.2 Aufruf des Konfigurationsprogramms CAN-Control

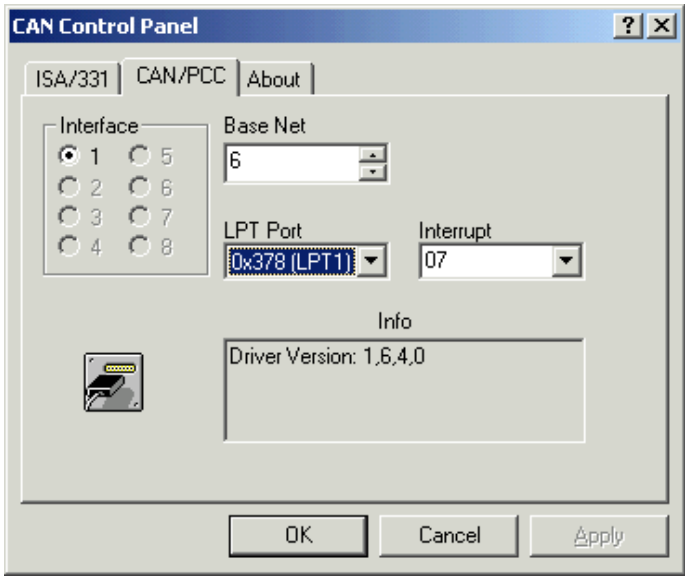
Zur Konfiguration des Treibers wird das Programm CAN-Control mitgeliefert. Es wird aus der Systemsteuerung heraus aufgerufen.

Die folgende Abbildungen zeigen beispielhaft ein System mit zwei esd-CAN-Modulen (CAN-ISA/331 und CAN-PCC). CAN-Control zeigt nur die esd-CAN-Module, die in dem System installiert sind. Haben Sie nur ein CAN-Modul installiert, so wird nur eine Karteikarte dargestellt.

Die folgende Abbildung zeigt die Einstellungsoptionen für CAN-**ISA**-Module:



Die folgende Abbildung zeigt die Einstellungsoptionen für das CAN-**PCC**-Modul.



Im Feld *Interface* wird die Nummer des CAN-Moduls eingetragen, für das die Einstellungen gelten sollen. In *Base Net* wird dem CAN-Modul die Basis-Netz-Nummer zugeordnet. Hat ein Modul mehr als eine physikalisches CAN-Schnittstelle, so erhält die erste die in *Base Net* eingetragene logische Netz-Nummer und die folgenden werden davon ausgehend hochgezählt.

Achtung: Der Anwender muß bei der Vergabe der logischen Netznummer dafür sorgen, daß keine Nummer doppelt vergeben wird!

Geben Sie für jedes CAN-Modul die von Ihnen vorgenommene Hardware-Konfiguration bezüglich der I/O-Port-Adresse ein, setzen Sie den Interrupt-Level und drücken Sie OK. Der Treiber ist nun installiert und konfiguriert.

A 2.2.2.1.3 Nur CAN-PCC: Stoppen des Systemtreibers für die parallele Schnittstelle

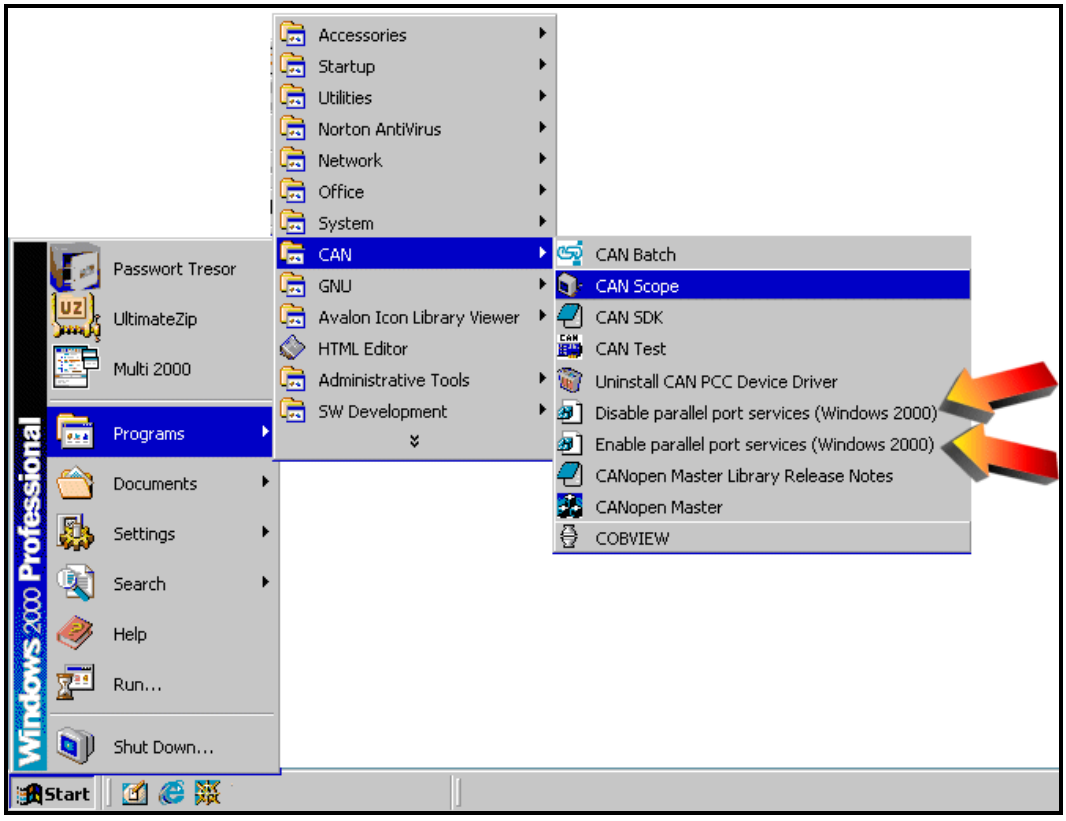
Das CAN-PCC-Modul wird über die parallele Schnittstelle des Rechners betrieben. Standardmäßig greift der Windows-2000-Treiber auf diese Schnittstelle zu. Damit der CAN-Treiber ungehinderten Zugriff auf das parallele Port erhält, muß **vor dem Starten des CAN-Treibers** der Windows-2000-Treiber gestoppt werden.

Hierfür stehen nach der Installation zwei Aufrufe zur Verfügung, die über *Programme/CAN* zu erreichen sind:

- **vor** dem Aufruf des CAN-Treibers muß der Aufruf von *Disable parallel port services (Windows 2000)* erfolgen *und* der Rechner neu gebootet werden

- wenn das Parallele Port wieder anderweitig weitergenutz werden soll (z.B. als Druckerport), muß zunächst der CAN-Treiber angehalten werden *und* der Aufruf *Enable parallel port services (Windows 2000)* erfolgen *und* dann der Rechner neu gebootet werden

Hinweis:	Falls das Starten des CAN-Treibers im Gerätemanager von 'Manuell' auf 'Automatisch' ungestellt worden ist, so ist dies wieder Rückgängig zu machen, wenn das parallele Port wieder für andere Anwendungen und nicht mehr für den CAN-Treiber verwendet werden soll.



A 2.2.2.1.4 Starten des Treibers

Das erstmalige manuelle Starten des Treiber ist nur bei CAN-ISA-Modulen und der CAN-PCC erforderlich.

Zum Starten des Treibers sollte zunächst auf der Kommandozeile der Befehl `net start` *`xxxx`* eingegeben werden.

Für die verschiedenen CAN-Module werden die Treiber über folgende Aufrufe gestartet:

CAN-Modul	Kommandos zum Starten der Treiber
CAN-ISA/200	`net start c200i`
CAN-PC104/200	
CAN-ISA/331	`net start c331i`
CAN-PC104/331	
CAN-PCC	`net start cpcc`

Tabelle A2.2.2: Kommandos zum Starten der Treiber

Erfolgt der Start des Treibers ohne Komplikationen, kann der Starttyp des CAN-Treibers im Dialog *Systemsteuerung/Geräte* von **Manuell** auf **Automatisch** gesetzt werden, so daß der Treiber nach jedem Kaltstart aktiviert wird und auch von anderen Benutzern ohne Administrator-Rechte genutzt werden kann.

Im Falle eines Problems beim Starten des Treibers kann die Fehlerursache der Ereignisanzeige von Windows 2000 entnommen werden.

A 2.2.2.2 De-Installation der Software-Treiber

Die De-Installation des Treibers der CAN-ISA-Boards und der CAN-PCC erfolgt in drei Schritten:

Achtung: Das Anhalten des Treibers und die vollständige De-Installation kann nur mit Administrator-Rechten auf der Windows 2000 Maschine erfolgen !

1. Beenden aller Applikationen, die die Dienste des Treibers nutzen

2. Der Treiber muß im Dialog *Systemsteuerung/Geräte* oder durch Eingabe von `net stop xxxx` auf der Kommandozeile angehalten werden

CAN-Modul	Kommandos zum Stoppen der Treiber
CAN-ISA/200	`net stop c200i`
CAN-PC104/200	
CAN-ISA/331	`net stop c331i`
CAN-PC104/331	
CAN-PCC	`net stop cpcc`

Tabelle A2.2.3: Kommandos zum Stoppen der Treiber

3. In der Dialogbox *Eigenschaften von Software* im Ordner *Systemsteuerung* den Eintrag für den CAN-Treiber auswählen und den Knopf *Hinzufügen/Entfernen* anwählen, damit alle Dateien und Registry-Einträge des Treibers gelöscht werden.

A 2.2.2.3 Besonderheiten der Windows 2000/XP-Implementierung

Das CAN-USB-Mini-Modul wird mit einem WDM-Treiber (C.2464.10) geliefert. Bei den aktuellen Treibern aller anderen Module handelt es sich nicht um WDM-Treiber, d.h.

- Die Resource-Vergabe erfolgt durch Absuchen des PCI-Busses und nicht durch Auswertung der von Windows 2000 zur Verfügung gestellten Daten.

- Das Power-Management wird zur Zeit nicht unterstützt.

- Der Treiber unterstützt Plug'n'Play. Das Anhalten des Treibers im Gerätemanager ohne Neustart ist jedoch nicht möglich.

A 2.2.2.4 Ereignisverwaltung

Alle Fehlersituationen beim Start oder während des Betriebes werden in der Ereignisverwaltung von Windows 2000/XP registriert.

A 2.3 Installation unter Windows 95/98/ME

Die Windows 95/98/ME-Software unterstützt maximal 5 CAN-Module des selben Typs in einem System. Eine Einschränkung für die Anzahl verschiedener CAN-Module in einem System gibt es nicht.

Hinweis: Das CAN-USB-Mini-Modul ist nicht unter Windows **95** lauffähig!

A 2.3.1 Installation von PCI-Bus-Boards und CAN-USB-Mini

1. Installieren Sie das Board gemäß der Hardware-Installationsbeschreibung.

2. Nach dem Hochlauf findet Windows 9x/ME das Board und teilt Ihnen dies mit. Wenn Windows keinen Treiber findet (z.B. bei der ersten Installation) werden Sie aufgefordert, den Datenträger (Diskette oder CD-ROM) mit dem Treiber einzulegen.

3. Zum Abschluß der Software-Installation müssen Sie den Computer herunterfahren. Beenden Sie vorher alle anderen Anwendungen, die noch geöffnet sind! Nach dem Neustart wird der Treiber automatisch gestartet und im Geräte-Manager existiert jetzt die neue Geräte-Klasse *CAN Controller* unter der alle esd-CAN-Karten angezeigt und konfiguriert werden können.

4. Installieren Sie jetzt das SDK, wie auf Seite A-36 beschrieben. Das SDK beinhaltet die Dateien, die zur Entwicklung von eigenen Applikationen notwendig sind und das Monitorprogramm CANscope.

Besonderheit bei der Installation der CAN-PCI/360:

CAN-PCI/360-Module, die vor dem 01.01.2000 gefertigt wurden, melden während der Installation ihren maximalen Speicherausbau von 128 MByte beim System an, auch wenn sie mit weniger Speicher bestückt sind. Bei diesen Boards kann es zu Konflikten mit einigen Grafik-Karten kommen, die hier ähnlich verfahren. Für diese Boards kann über den Gerätemanager der tatsächliche Speicherbedarf eingestellt werden. Näheres zum Gerätemanager finden Sie im Kapitel ‘Änderungen der Ressourcen-Einstellungen mit Hilfe des Geräte-Managers’ auf Seite A-30.

Bei CAN-PCI/360-Modulen, die ab dem 01.01.2000 gefertigt wurden tritt dieses Problem nicht mehr auf.

A 2.3.2 Installation von ISA-Bus-Boards und CAN-PCC

A 2.3.2.1 Hinweis zur Hardware-Installation bei ISA-Boards

Installieren Sie das Board erst im Computer, nachdem Sie über die Windows 9x/ME-Systemsteuerung die passende Konfiguration (Adreßbereich) ermittelt haben.

Nach der im folgenden beschriebenen Installation muß außerdem das SDK (siehe Seite A-36) installiert werden.

CAN-2.0B-Unterstützung der CAN-ISA/200 und CAN-PC104/200:

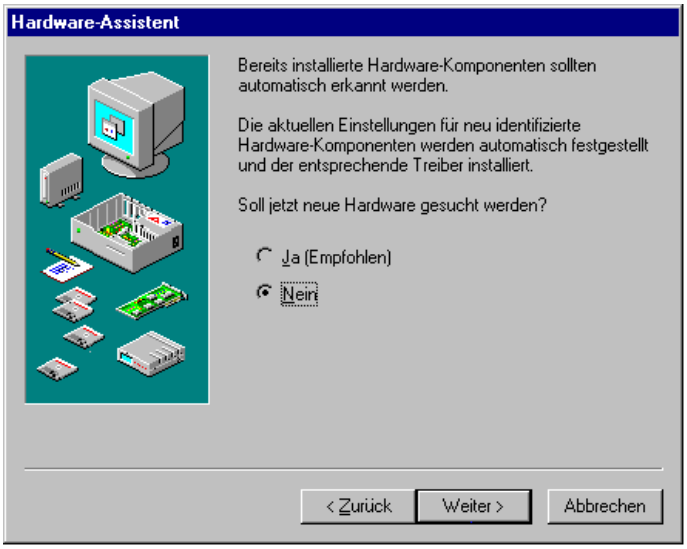
Ab der Treiber-Revision 1.2.0 können Sie für die CAN-ISA/200 und CAN-PC104/200 während der Installation zwischen einem Treiber für 11-Bit-IDs (CAN2.0A) und einem Treiber für 29-Bit-IDs (CAN 2.0B) wählen. Der CAN2.0B-Treiber ist etwas langsamer als der CAN2.0A-Treiber, auch wenn Sie ihn nur zum Senden und Empfangen von 11-Bit-IDs verwenden, weil hier mehr Zugriffe auf den CAN-Controller notwendig sind.
Voraussetzung für den CAN2.0B-Betrieb ist, daß Ihr Modul nicht mehr mit einem CAN-Controller des Typs 82C200 (bis 12/1999 ausgeliefert) bestückt ist, sondern z.B. mit einem SJA1000.

A 2.3.2.2 Installation der Geräte-Treiber

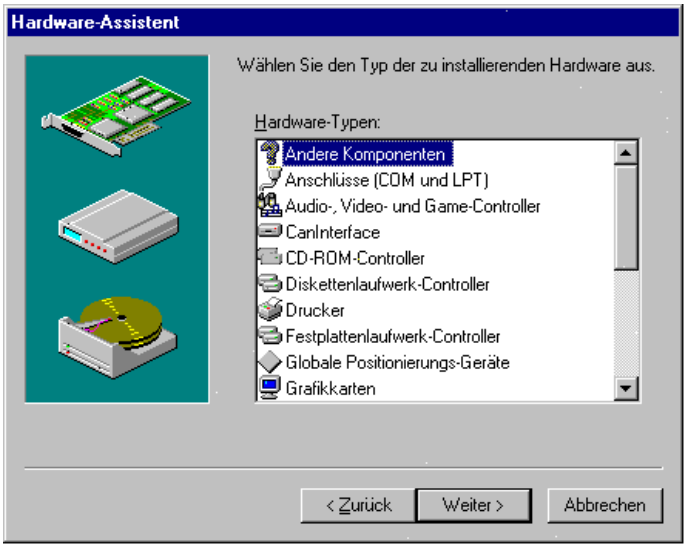
1. Aufruf
   Die Installation des Windows 9x/ME-Treibers des ISA-Boards erfolgt mit Hilfe des *Hardware-Assistenten*. Er wird in dem Fenster *Systemsteuerung* gestartet, indem das Icon *Hardware* aufgerufen wird:

   Danach muß sich das Startfenster des Hardware-Assistenten öffnen:

2. Im nächsten Fenster wird abgefragt, ob die Hardware gesucht werden soll. Da das Board dem System nicht bekannt ist (kein Plug&Play), muß hier *Nein* gewählt werden. Falls *Ja* ausgewählt wird, wird der Rechner relativ lange nach dem Board suchen, um sich dann mit der Meldung, er habe ein unbekanntes Board entdeckt, zurückmelden.



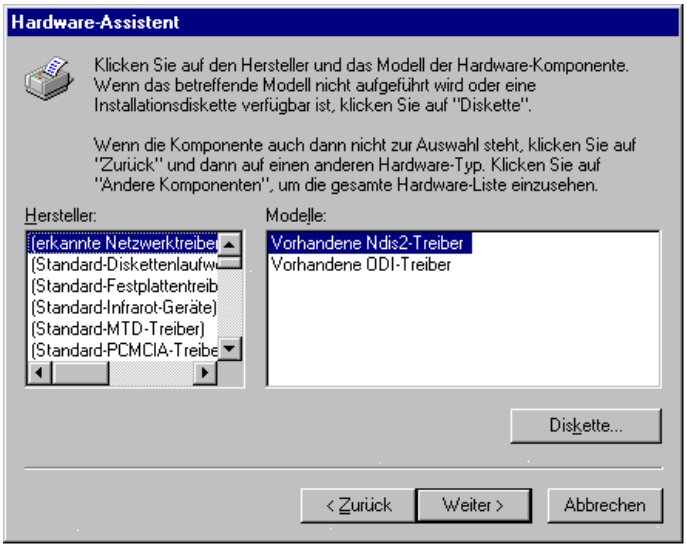
3. Danach erscheint das Fenster zur Auswahl des Typs der zu installierenden Hardware.



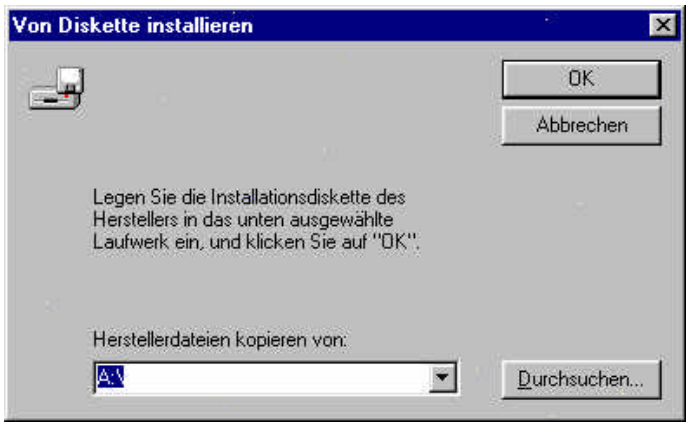
Hier muß *Andere Komponenten* ausgewählt und danach die Schaltfläche *Weiter* gedrückt werden. (Nur, wenn bereits ein esd-CAN-Treiber installiert ist, würde wie in dem oben abgebildeten Fenster der Auswahlpunkt *CAN Interface* mit erscheinen. Bei der ersten Installation wird er nicht angezeigt.)

Falls nicht bereits erfolgt, sollten Sie jetzt den mitgelieferten Datenträger (Diskette oder CD-ROM) in Ihr Laufwerk einlegen.

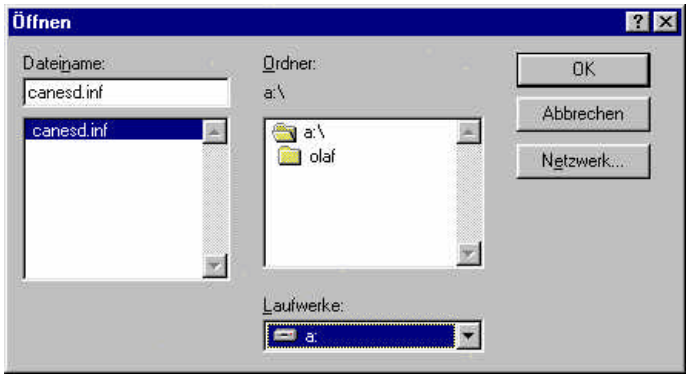
4. In dem Fenster, das sich nach der Geräte-Typ-Auswahl öffnet, muß erst die Schaltfläche *Diskette* und daraufhin die Schaltfläche *Weiter* gedrückt werden.



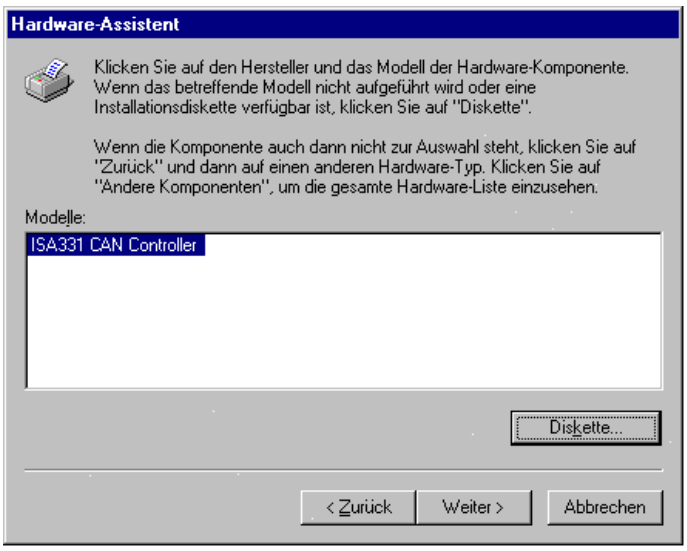
5. Das folgende Fenster erscheint. Drücken Sie hier die Schaltfläche *Durchsuchen*.



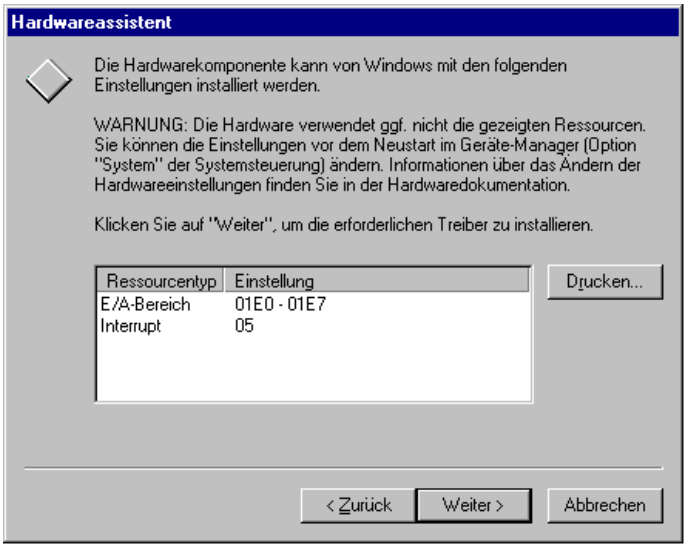
6. Wählen Sie nun die Datei `canesd.inf` aus und drücken Sie OK.



7. Das folgende Fenster, in dem alle esd-CAN-Treiber angezeigt werden, muß sich öffnen (in diesem Beispiel wird nur der ISA/331-Treiber angezeigt). Wählen Sie den Treiber Ihrer Karte aus und drücken Sie *Weiter*.



8. (Nur für CAN-ISA-Boards, sonst weiter mit Punkt 9) Windows 9x/ME installiert den Treiber, überprüft die System-Ressourcen, vergleicht sie mit den für das Board möglichen Konfigurationen und bietet in dem folgenden Fenster eine mögliche Konfiguration für das Board an:



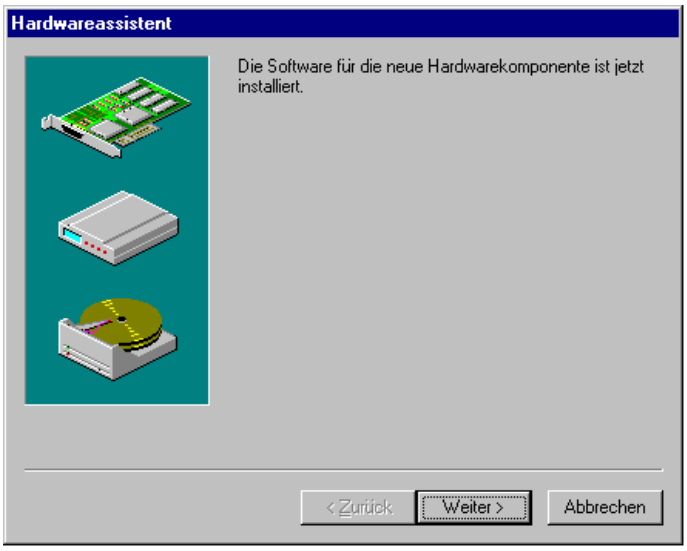
Achtung:
Der von der Systemsteuerung vorgeschlagene *E/A-Bereich* muß nicht mit dem Default-E/A-Bereich, der auf der Karte über Steckbrücken eingestellt ist, übereinstimmen.

Daher muß in jedem Fall die Steckbrücken-, oder Kodierschaltereinstellung auf dem CAN-ISA-Board mit dem hier vom System gewählten E/A-Adreßbereich verglichen, und ggf. angepaßt werden!

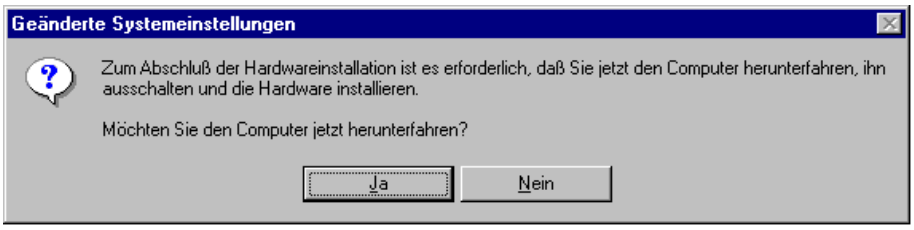
Drücken Sie die Schaltfläche *Weiter*.

Nur falls das System keine Vorschläge für Einstellungen bieten sollte:
Zeigt das System an, daß es keine Ressourcen für das CAN-Board besitzt, so ist die Installation trotzdem zu Ende zu führen. Nach der Installation müssen in diesem Fall die Ressourcen im System manuell über den Geräte-Manager neu verteilt werden.

9. Die erfolgreiche Installation des Software-Treibers wird über das folgende Fenster angezeigt:



10. Zum Abschluß der Software-Installation müssen Sie den Computer herunterfahren. Beenden Sie vorher alle anderen Anwendungen, die noch geöffnet sind!



Schalten Sie den Computer danach aus und **installieren Sie *jetzt* die Hardware**, wie im Hardware-Handbuch beschrieben! Vergleichen Sie die Standard-Steckbrücken, bzw. Kodierschaltereinstellung mit der unter Punkt 8 gewählten Einstellung und ändern Sie die Steckbrücken gegebenenfalls.

Installieren Sie jetzt das SDK, wie auf Seite A-36 beschrieben. Das SDK beinhaltet die Dateien, die zur Entwicklung von eigenen Applikationen notwendig sind und das Monitorprogramm CANscope.

A 2.3.2.3 Starten der Geräte-Treiber

A 2.3.2.3.1 ISA-Bus Boards

Nach der Installation schalten Sie den Rechner wieder ein und starten Windows 9x/ME neu. Der Treiber wird von Windows 9x/ME automatisch geladen und gestartet. Im Geräte-Manager existiert jetzt die neue Geräte-Klasse *CAN Controller,* unter der alle esd-CAN-Karten angezeigt und konfiguriert werden können.

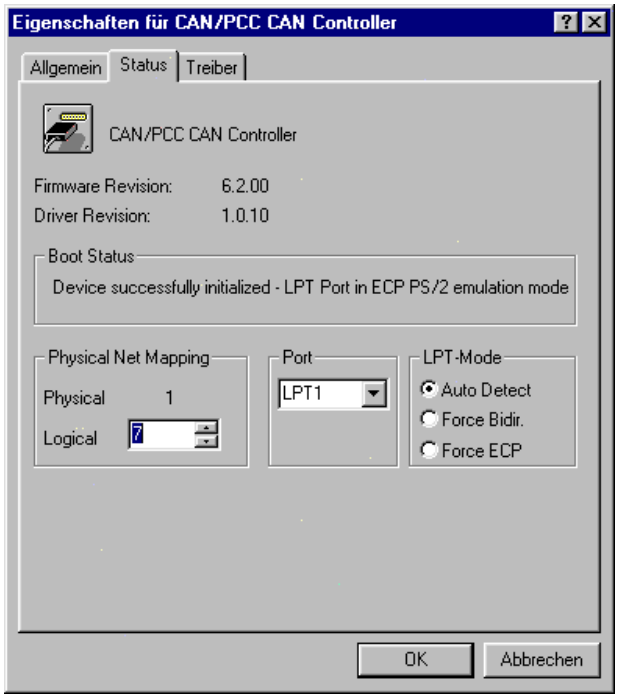
A 2.3.2.3.2 CAN-PCC

Nach der Installation schalten Sie den Rechner wieder ein und starten Windows 9x/ME neu. Der Treiber wird von Windows 9x/ME automatisch geladen jedoch nicht gestartet. Im Geräte-Manager existiert jetzt jedoch die neue Geräte-Klasse *CAN Controller,* unter der alle esd-CAN-Karten angezeigt und konfiguriert werden können.

Zum Starten des Treibers muß auf der Kommandozeile explizit der Befehl `cpcc start` eingegeben werden. Dies ist notwendig, damit die parallele Schnittstelle mit anderen Gerätetreibern geteilt werden kann. Nach dem erfolgreichen Start des Treibers besitzt dieser exklusiven Zugriff auf die parallele Schnittstelle bis dieser mit `cpcc stop` wieder beendet wird.

Ein weiterer Grund für die Ausführung eines expliziten Startkommandos ist, daß dem Treiber innerhalb von Windows 9x/ME keine eigenen Ressourcen zugeordnet sind. Er nutzt die bereits vorhandene Ressource 'Parallele Schnittstelle'. Daher existiert innerhalb des Gerätemanagers für diesen Treiber keine Schaltfläche *Ressourcen* (siehe folgendes Kapitel). Statt dessen muß in der Schaltfläche *Status* die zu nutzende parallele Schnittstelle (LPT1, LPT2, LPT3) unterhalb von Port angegeben werden und der Treiber verwendet automatisch die aktuelle Windows-9x/ME-Konfiguration (I/O-Adresse, Interrupt) für diese Schnittstelle .

Achtung:
Für den Betrieb der CAN-PCC unter Windows9x/ME muß die parallele Schnittstelle im Eigenschaften-Fenster auf die Betriebsart *Force ECP* eingestellt werden. Sollte diese Einstellung von der Hardware nicht unterstützt werden, so muß der bidirektionale Mode (*Force Bidir.*) gewählt werden.



A 2.3.2.4 Änderungen der Ressourcen-Einstellungen mit Hilfe des Geräte-Managers

A 2.3.2.4.1 Übersicht

Achtung:	Änderungen der Grundeinstellungen über den Geräte-Manager oder den Registry-Editor können zu Konflikten führen, so daß ein oder mehrere Geräte vom System nicht mehr erkannt werden! Der Geräte-Manager und der Registry-Editor sind Konfigurationswerkzeuge für fortgeschrittene Anwender, die mit der Parameter-Konfiguration vertraut sind und wissen, welche vielfältigen Auswirkungen Änderungen haben können.

Das CAN-ISA-Board besitzt feste Ressourcen-Einstellungen, die entweder während des Windows-Setups vom Hardware-Assistenten vergeben werden oder später, über den Geräte-Manager, konfiguriert werden können.

Es kann vorkommen, daß Windows 9x/ME das CAN-ISA-Board nicht konfigurieren kann, weil Konflikte mit anderen Geräten auftreten. In diesem Fall ist es notwendig, die Karte neu zu konfigurieren.

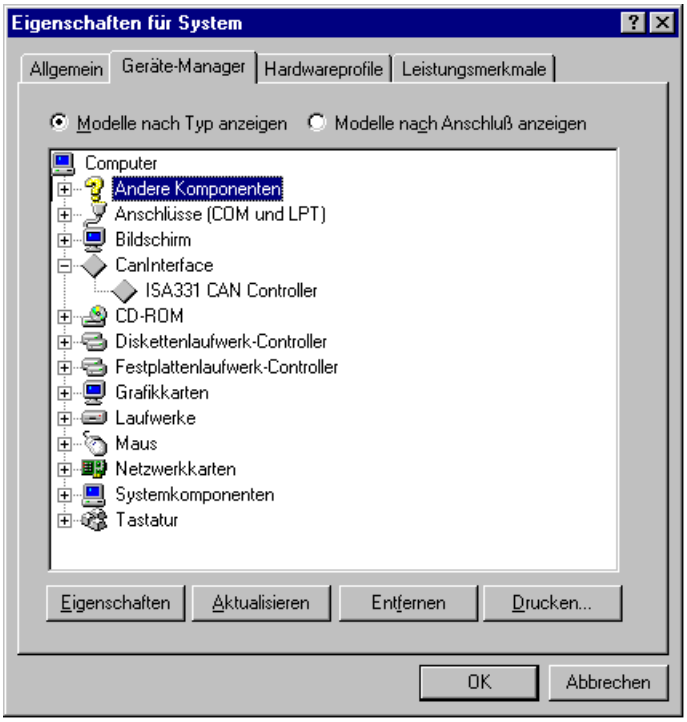
Um die Geräteeinstellung manuell zu ändern, kann der *Geräte-Manager*, der über *System-Optionen* unter *Systemsteuerung* aufgerufen wird, verwendet werden. Die Verwendung des Geräte-Managers schützt vor den Fehlern, die beim direkten Editieren der Registry-Einträge auftreten können.

Wenn Sie die Gerätekonflikte mit Hilfe des Geräte-Managers manuell lösen wollen, können Sie z.B. die folgenden Strategien anwenden:

- Sollte das Konfliktgerät ein Plug&Play-Gerät sein, so deaktivieren Sie es, damit es seine Ressourcen freigibt.
- Sollte das Konfliktgerät ein herkömmliches Gerät sein, so deaktivieren Sie es, indem Sie die Karte aus dem System entfernen und die Treiber entladen.
- Verteilen Sie die Ressourcen, die von anderen Geräten genutzt werden neu, um Ressourcen für das Konfliktgerät freizugeben.
- Ändern Sie die Steckbrücken, bzw. Kodierschalter des CAN-Boards, um die Karte an die neuen Einstellungen anzupassen.

A 2.3.2.4.2 Aktivieren des Geräte-Managers

1. Aufruf
   Klicken Sie unter *Systemsteuerung* in *System-Option* auf Schaltfläche des *Geräte-Managers*.
   oder
   Rechte Maustaste *Mein Computer*, drücken Sie *Einstellungen* im *Context*-Menü und klicken Sie dann auf *Geräte-Manager*.



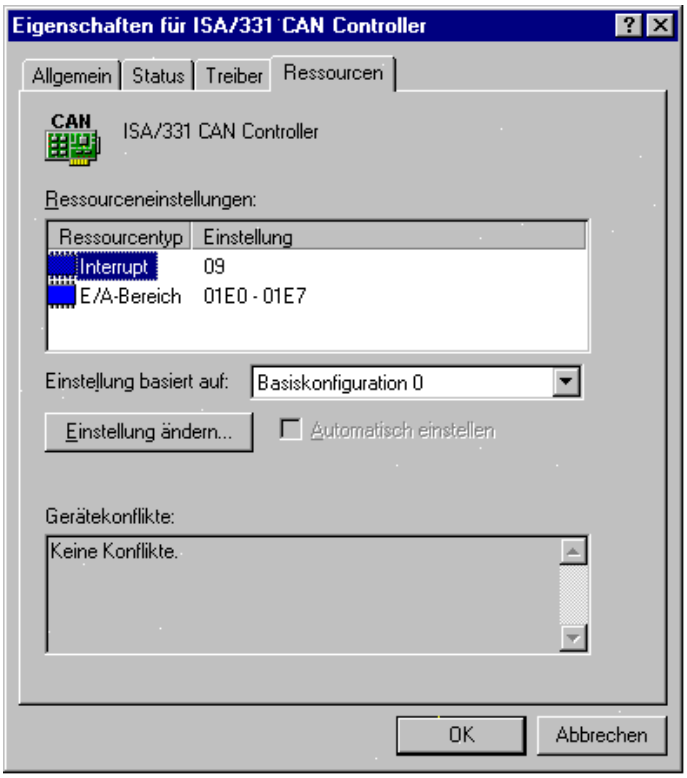
2. Ein Doppelklick mit der linken Maustaste auf den gewünschten Geräte-Typ in der Liste führt zur Anzeige aller Geräte dieses Typs im Computer.

3. Selektieren Sie das zu konfigurierende Gerät über einen Doppelklick. Oder markieren Sie es und drücken Sie die Schaltfläche *Eigenschaften*.

A 2.3.2.4.3 Ändern der Ressource-Einstellungen mit dem Geräte-Manager

1. Wählen Sie im Geräte-Manager per Doppelklick die Geräte-Klasse *CAN Controller*.
   Der Verzeichnisbaum verzweigt sich und die Geräte dieser Klasse, die in Ihrem Computer verfügbar sind, werden angezeigt.

2. Ein Doppelklick auf ein Gerät öffnet dessen Eigenschaftsfenster.
   Drücken Sie bei den Geräteeigenschaften die Schaltfläche *Ressourcen*.



   Die Liste *Gerätekonflikte* zeigt die Einstellungen anderer Geräte, die im Konflikt mit der aktuellen Einstellung der CAN-ISA-Karte stehen.

3. In der Liste *Ressourcentyp* kann die Einstellung gewählt werden, deren Änderung gewünscht ist, z.B. der Interrupt-Level. Um die geänderten Werte zu Übernehmen, muß die Schaltfläche *Einstellung ändern* betätigt werden.
   Änderungen sind nur möglich, wenn die Option *Automatisch einstellen* deaktiviert ist.
   Die Einstellungen *Interrupt* und *E/A-Bereich* (Adreßbereich) können unabhängig voneinander verändert werden.

   Die Dialogbox *E/A-Bereich* zeigt die verschiedenen Einstellungen, die unterstützt werden. Ein Interrupt, der mit einem Stern (*) gekennzeichnet ist, zeigt an, daß dieser Interrupt bereits von einem anderen Gerät belegt ist.

   Nach dem Drücken der *Einstellung ändern*-Schaltfläche könnte eine Fehlermeldung erscheinen, die besagt, daß diese Ressource-Einstellung nicht verändert werden können. In diesem Fall ist eine andere Einstellung zu wählen, bis eine Einstellung gefunden ist, die akzeptiert wird.

4. Wählen Sie Einstellungen, die nicht mit anderen Geräten in Konflikt stehen, und drücken Sie ‘OK’.

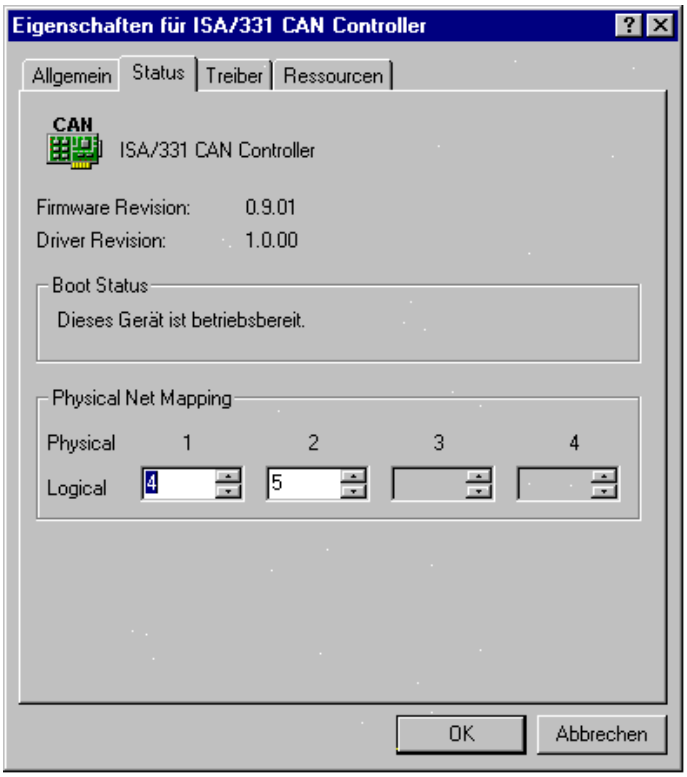
Beachten Sie, daß die Einstellung des Adreßbereiches über die Steckbrücke, bzw. den Kodierschalter mit der gewählten Einstellung übereinstimmen muß!

5. Beenden Sie Windows 9x/ME, ändern Sie ggf. die Hardware-Einstellungen von neu konfigurierten Geräten und starten Sie Windows 9x/ME neu.

A 2.3.2.4.4 Ändern der logischen Netz-Nummer

1. Wählen Sie im Geräte-Manager per Doppelklick die Geräte-Klasse *CAN Controller*.
   Der Verzeichnisbaum verzweigt sich und die Geräte dieser Klasse, die in Ihrem Computer verfügbar sind, werden angezeigt.

2. Ein Doppelklick auf ein Gerät öffnet dessen Eigenschaftsfenster.
   Drücken Sie bei den Geräteeigenschaften die Schaltfläche *Status*.



3. In *Physical Net Mapping* können den physikalischen Schnittstellen der CAN-Karte logische Netz-Nummern zugeordnet werden, über die sie von der Software aus angesprochen werden können. Beim ersten Start des Treibers werden der ersten vom Treiber unterstützten Karte logische Netznummern beginnend mit 0 zugeordnet. Werden mehrere esd-CAN-Karten unterschiedlichen Typs (z.B. CAN-PCI/331 und CAN-ISA/331) in einem Rechner betrieben, kommt es bei diesem Verfahren zwangsläufig zu Überschneidungen, so daß die Netznummern von Hand angepaßt werden müssen.

4. Eine Änderung der logischen Netznummern wird erst nach einem Neustart des Rechners gültig.

A 2.4 Installation und Konfiguration des CAN-Bluetooth-Moduls

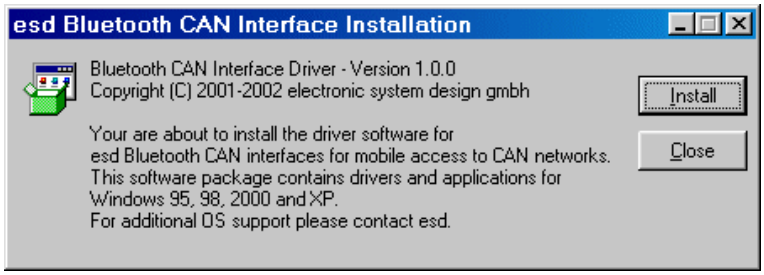
Das CAN-Bluetooth-Modul ist unter den Betriebssystemen Windows 98/ME und Windows 2000/XP lauffähig. In diesem Kapitel wird die Installation und die grundlegende Konfiguration des Moduls beschrieben. Die weitere Konfiguration über den integrierten WEB-Server ist im CAN-Bluetooth-Handbuch beschrieben.

Achtung: Der CAN-Bluetooth-Treiber ist generell erst nach der Installation des CAN-SDKs zu installieren! Sollten in einem System verschiedene esd-CAN-Module betrieben werden, so ist der CAN-Bluetooth-Treiber immer als letzter zu installieren!

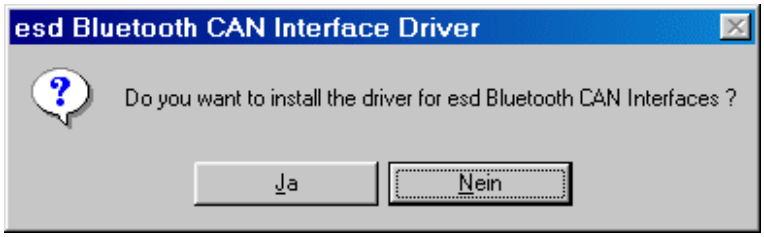
A 2.4.1 Installation

1. Installationsprogramm `canbtdrv.exe` aufrufen.

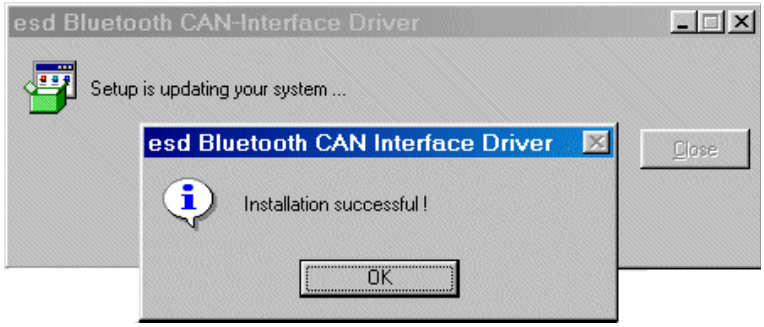
2. Nachdem sich das Fenster *esd Bluetooth CAN-Interface Driver* geöffnet hat, ist die Schaltfläche *Install* zu drücken.



3. Installation Starten

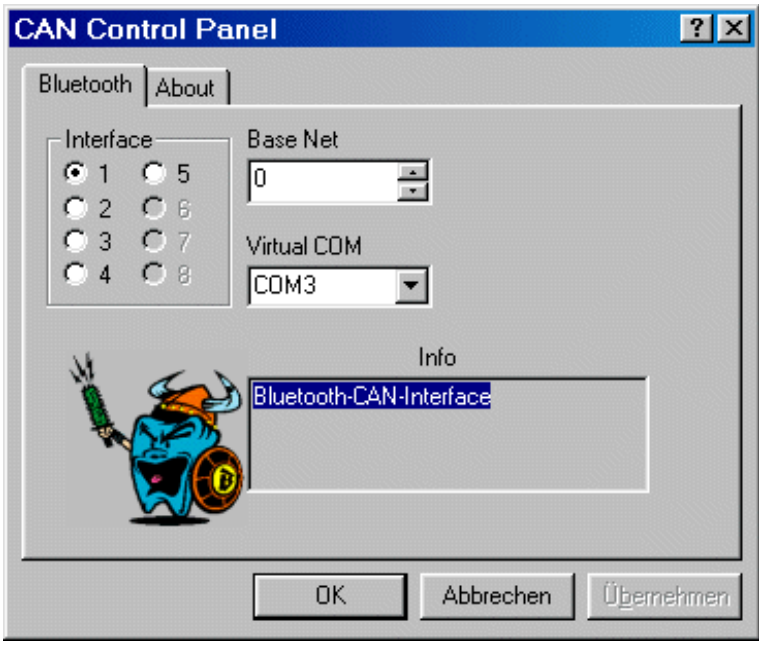


4. Das erfolgreiche Beenden oder das Fehlschlagen der Installation wird angezeigt.



A 2.4.2 Konfiguration

Nach der Installation finden Sie im Menü *Start* unter *Programme* das Konfigurationsprogramm *esd Bluetooth CAN-Interface Driver*. Hier wählen Sie *CANBT Config* und das Fenster *CAN Control Panel* öffnet sich.



Hier können folgende Einstellungen vorgenommen werden:

Interface Die Software unterstützt maximal 5 verschiedene esd-CAN-Bluetooth-Interfaces. Hier wählen Sie aus, welches Interface Sie konfigurieren möchten.

Base Net Hier wird die logische CAN-Netz-Nummer eingetragen. Es sind Werte zwischen 0 und 255 zulässig.

Virtual COM Das Bluetooth-Interface wird als virtuelle COM-Schnittstelle verwaltet. Es sind Werte zwischen 1 und 11 zulässig.

> **Hinweis:** Bluetooth-Geräte verwenden in der Regel virtuelle serielle Schnittstellen für die Kommunikation. Die Zuweisung zwischen COMx und Bluetooth-Gerät wird über die Software konfiguriert, die Ihre Bluetooth-Hardware steuert (z.B. PC-Card).

A 2.5 Installation des SDK (Software Development Kit)

Nach der Installation des Treibers kann das SDK installiert werden. Es beinhaltet die Dateien zur Entwicklung und zum Test von eigenen Applikationen.

Sie finden auf der CD-ROM ein eigenes Verzeichnis 'CAN_SDK'. Ist die Software auf Disketten ausgeliefert worden, so finden Sie das SDK auf einer eigenen Diskette.

Zur Installation ist lediglich das Programm `CANSDK.EXE` im SDK-Verzeichnis der CD-ROM, bzw. auf der SDK-Diskette aufzurufen. Es richtet auf Ihrem PC die notwendigen Verzeichnisse ein und kopiert die Dateien dort hinein.

`CANSDK.EXE` würde z.B. für das Modul CAN-PCI/331 einen Verzeichnisbaum ähnlich dem folgenden erstellen:

```
CAN----bin--------  canscope.exe
       |            cantest
       |            updatec331.exe
       |            :
       |
       ---include---  ntcan.h
       |            :
       |
       ---samples---  cantest.c
       |            :
       |
       ---develop---  vc
       |            bc
       |            :
       ---help
```

A 2.6 Update der Firmware und Umschaltung zwischen CAN2.0A und CAN2.0B

A 2.6.1 Firmware-Update

A 2.6.1.1 Übersicht

Die Update-Funktion der lokalen Firmware ist für die Boards CAN-ISA/200, CAN-PCI/200 und CAN-PC104/200 ohne Bedeutung, da diese Boards keinen lokalen CPU-Baustein besitzen. Bei dem Modul CAN-PCC wird die Update-Funktion ebenfalls nicht unterstützt.

Die Firmware für den lokalen Prozessor der intelligenten CAN-Boards ist in Flash-EPROM-Speicherbausteinen abgelegt. Bei Auslieferung der Karte enthalten sie die Firmware-Version, die zum Zeitpunkt der *Hardware-Herstellung* aktuell war.

Auf den mitgelieferten Disketten oder auf der CD-ROM befindet sich ein Update-Programm, mit dessen Hilfe die zum *Auslieferungstermin* gültige Software in das Flash-EPROM gebrannt werden kann.

In Abhängigkeit von dem Modul und dem Betriebssystem finden Sie entweder das Update-Programm '`canupd`' oder das Update-Programm '`upd`*xxx*' auf Ihrem Datenträger. Die Programme werden während der Treiberinstallation auf Ihrer Festplatte installiert.
Beide Programme sind von der Funktionalität her gleichwertig. Während `canupd` ein universelles Update-Programm für verschiedene Module ist, bietet `upd`*xxx* gezielte Updates für einzelne Module.

Die Kennung der Module erfolgt dabei über die Zeichenkombination beim Aufruf:

CAN-Modul	Eingabe-Syntax für `upd`*xxx*
CAN-ISA/331 CAN-PC104/331	`updc331i`
CAN-PCI/331 CPCI-CAN/331 PMC-CAN/331	`updc331`
CAN-PCI/360 CPCI-CAN/360	-

Tabelle A2.6.1: Eingabe-Syntax beim Update-Programm `upd`*xxx*

A 2.6.1.2 Prüfung der Firmware-Versionsnummer

Mit den Update-Programmen können Sie prüfen, welche Firmware-Version in Ihrem Flash-EPROM gespeichert ist und bei Bedarf selbständig Updates der lokalen Firmware durchführen.

Wir empfehlen, nach der Installation der Software-Treiber für die Host-CPU zu prüfen, ob die mitgelieferte Firmware aktueller ist als die im Flash-EPROM befindliche Firmware.

Funktionsweise:

1. Vor dem Starten von `canupd`, bzw. `upd`*`xxx`* muß der Software-Treiber der Host-CPU gestartet sein (siehe vorangegangene Kapitel).

2. Öffnen Sie eine Window-Konsole und wechseln Sie in das Installationsverzeichnis.

3. Starten Sie `canupd`, bzw. `upd`*`xxx`* über die Windows-Command-Shell.
   Der Aufruf des Programms ohne Parameterangabe führt zur Ausgabe der zur Verfügung gestellen Firmware mit Versionsnummern. Da `canupd` ein universelles Programm ist, enthält es Firmware für verschiedene Hardware-Anwendungen.

   Der Aufruf des Programms mit anschließender Eingabe der logischen Netz-Nummer der zu bearbeitenden Karte führt zur Ausgabe der Versionsnummer der im Flash-EPROM dieser Karte gespeicherten Firmware.

   Beispiel-Eingabe: `canupd 0`

(Die Firmware-Version und die Treiber-Version der Host-CPU können nach dem Start des Treibers alternativ auch über den Windows-Event-Log ermittelt werden. Hier wird zusätzlich angezeigt, welche Firmware-Versionen mit der angezeigten Treiber-Software betrieben werden können.)

A 2.6.1.3 Ausführen des Updates

1. Aufruf von `canupd` (bzw. `upd`*`xxx`*) mit gewünschter logischer Netz-Nummer (siehe oben).

2. Nach der Versionsüberprüfung erfolgt die Abfrage, ob das Programm mit dem Update fortfahren soll oder nicht.

   - Wird hier ein ‘y’ für JA eingegeben, so erfolgt anschließend die Neuprogrammierung des Flash-EPROMs **unabhängig davon, welche Firmware-Version aktueller ist!**

   - ‘n’ für NEIN bricht den Vorgang ab. Die Firmware im Flash-EPROM bleibt unverändert.

     Das Programm vergleicht dabei automatisch die Firmware-Versionen und zeigt auf dem Monitor in Klartext an, ob die gespeicherte Version noch aktuell ist oder ob ein Update erforderlich ist.

3. Um die neue Firmware zu aktivieren, muß der lokale Prozessor zurückgesetzt werden. Dies erfolgt durch Stoppen und Neustarten des Software-Treibers der Host-CPU, z.B. über die Kommandozeile:

```
net stop xxxx
net start xxxx
```

CAN-Board	Eintrag für *xxxx* in der Kommandozeile
CAN-ISA/331	`c331i`
CAN-PC104/331	
CAN-PCI/331	`c331`
CPCI-CAN/331	
PMC-CAN/331	
CAN-PCI/360	`c360`
CPCI-CAN/360	

Tabelle A2.6.2: Treiberbezeichnung bei Einträgen in der Kommandozeile

A 2.6.2 Umschaltung zwischen CAN2.0A und CAN2.0B-Betrieb

Alle CAN-Module, die die Update-Funktion unter Windows unterstützen, bieten außerdem die Möglichkeit, zwischen dem Betrieb des Treibers mit 11-Bit-CAN-Identifiern (CAN 2.0A) oder 29-Bit-CAN-Identifiern (CAN 2.0B) zu wählen.

Der Eintrag in die Kommandozeile muß hier wie folgt vorgenommen werden:

CAN-Modul	Eingabe-Syntax für CAN2.0A (11-Bit-ID)	Eingabe-Syntax für CAN2.0B (29-Bit-ID)
CAN-ISA/331 CAN-PC104/331	`updc331i -ta`	`updc331i -tb`
CAN-PCI/331 CPCI-CAN/331 PMC-CAN/331	`updc331 -ta`	`updc331 -tb`
CAN-PCI/360 CPCI-CAN/360	-	-

Tabelle A2.6.3: Wechsel auf CAN2.0A, bzw CAN2.0B

Um die neue Einstellung zu aktivieren, muß der lokale Prozessor zurückgesetzt werden. Das Rücksetzen und Neustarten erfolgt wie oben unter Punkt 3. beschrieben.

A 2.7 Einbindung in eigene C- /C++ - Projekte unter Windows

Um auf den CAN-Bus aus eigenen Projekten über die in Kapitel 3 beschriebenen API-Funktionen zuzugreifen, muß lediglich die Header-Datei *ntcan.h* (Standard API, Bestell.-Nr. C.20xx.10 und C.20xx.11) in die Quelltexte eingebunden und dem Linker die Datei *ntcan.lib* bekanntgemacht werden.

Für Borland C ab 5.0 ist die Datei *ntcanbc.lib* zu verwenden.

A 3. Unix-Betriebssysteme und AIX, VxWorks, QNX4

A 3.1 Installation unter Linux

Hinweise:
- Die Installation kann nur mit Supervisor-Rechten erfolgen.

- Für alle CAN-Module, die unter Linux betrieben werden ist zu beachten, daß der Linux-Kernel zur Zeit immer mit Multiprozessor-Support übersetzt werden muß!

A 3.1.1 Dateien des Linux-Paketes

Die Software-Treiber für Linux werden auf einer 3,5”-Diskette geliefert. Die Diskette enthält folgende Dateien:

<table>
<tr><th>Datei</th><th colspan="2">Beschreibung</th></tr>
<tr><td>README.c331</td><td>aktuelle Anmerkungen und Hinweise</td><td rowspan="3">Die Buchstabenkombination ‘c331’ zeigt die Dateien des Moduls CAN-PCI/331 an. Bei anderen Modulen werden diese Buchstaben wie folgt ersetzt:
<table>
<tr><th>CAN-Modul</th><th>Dateiname</th></tr>
<tr><td>VME-CAN2</td><td>-</td></tr>
<tr><td>VME-CAN4</td><td>-</td></tr>
<tr><td>CAN-ISA/200</td><td rowspan="2">c200i</td></tr>
<tr><td>CAN-PC104/200 (nur SJA1000)</td></tr>
<tr><td>CAN-ISA/331
CAN-PC104/331</td><td>c331i</td></tr>
<tr><td>CAN-PCI/200</td><td>c200</td></tr>
<tr><td>CAN-PCI/331
CPCI-CAN/331
PMC-CAN/331</td><td>c331</td></tr>
<tr><td>CAN-PCI/360
CPCI-CAN/360</td><td>c360</td></tr>
<tr><td>CAN-USB-Mini
CAN-USB/331</td><td>usb331</td></tr>
</table></td></tr>
<tr><td>z.B.
c331</td><td>dynamisch ladbarer Treiber</td></tr>
<tr><td>updc331</td><td>Programm zum Firmware-Update</td></tr>
<tr><td>libntcan.o</td><td colspan="2">ntcan-API</td></tr>
<tr><td>ntcan.h</td><td colspan="2">Header für die ntcan-API</td></tr>
<tr><td>cantest.c</td><td colspan="2">Quell-Code des Beispielprogramms ‘cantest’</td></tr>
<tr><td>cantest</td><td colspan="2">Binär-File des Beispielprogramms ‘cantest’</td></tr>
</table>

Tabelle A3.1.1: Dateien des Linux-Paketes

A 3.1.2 Ablauf der Installation unter Linux

1. Entpacken des Tar-Files oder der Tar-Disk

Zum Entpacken des Tarfiles muß folgendes Kommando aufgerufen werden:

```
tar -xvf c331-vx.y.z.tar
oder
tar -xvf / dev/fd0
```

Anmerkung: Die Eingabe '*c331*' muß für das Modul CAN-PCI/331 erfolgen. Bei anderen Modulen ist die in der folgenden Tabelle angegebene Buchstabenkombination einzusetzen. Dies gilt auch für die folgenden Kommandos.

CAN-Modul	Eingabe-Syntax
VME-CAN2	-
VME-CAN4	-
CAN-ISA/200	c200i
CAN-PC104/200 (nur SJA1000)	
CAN-ISA/331 CAN-PC104/331	c331i
CAN-PCI/200	c200
CAN-PCI/331 CPCI-CAN/331 PMC-CAN/331	c331
CAN-PCI/360 CPCI-CAN/360	c360
CAN-USB-Mini CAN-USB/331	usb331

Tabelle A3.1.2: Eingabe-Syntax beim Entpacken der Tar-Files

2. Erstellen der 'inodes' (als Supervisor)

Für CAN-PCI/331, CPCI-CAN/331 und PMC-CAN/331:

```
cd /dev
mknod --mode=a+rw can0 c 50 0
mknod --mode=a+rw can1 c 50 1
```

Für CAN-PCI/360, CPCI-CAN/360

```
cd /dev
mknod --mode=a+rw can0 c 51 0
mknod --mode=a+rw can1 c 51 1
mknod --mode=a+rw can2 c 51 2
mknod --mode=a+rw can3 c 51 3
```

Für CAN-ISA/331, CAN-PC104/331:

```
cd /dev
mknod --mode=a+rw can0 c 52 0
mknod --mode=a+rw can1 c 52 1
```

Für CAN-ISA/200, CAN-PC104/200 (hier nur für SJA1000):

```
cd /dev
mknod --mode=a+rw can0 c 53 0
```

3. Laden des CAN-Treibers (als Supervisor)

Der Treiber ist für die Linux-Kernels 2.029 bis 2.033 getestet (Stand 25.06.98). Falls ein anderer Kernel verwendet wird, kann der Treiber mit der Option '`-f` ' auch hier geladen werden.

Für CAN-PCI/331, CPCI-CAN/331 und PMC-CAN/331:

```
insmod c331
```

oder, falls ein anderer Linux-Kernel Verwendung findet:

```
insmod -f c331
```

Für CAN-PCI/360, CPCI-CAN/360:

```
insmod c360
```

oder, falls ein anderer Linux-Kernel Verwendung findet:

```
insmod -f c360
```

Für CAN-ISA/331 und CAN-PC104/331:

```
insmod c331i
```

oder, falls ein anderer Linux-Kernel Verwendung findet:

```
insmod -f c331i
```

(erkennt ein Module mit der I/O-Adresse `0x1E0` (hex) und ordnet ihm den IRQ **5** zu und erkennt außerdem ein Modul mit der Adresse `0x1E8` (hex) und ordnet ihm den IRQ **7** zu)

oder, falls andere Adressen oder Interrupts gewünscht sind:

```
insmod c331i io=0x port irq=0x irq
```

mit *`port`* = eingestellte Port-Adresse
`IRQ` = gewünschter Interrupt

Für CAN-ISA/200, CAN-PC104/200:

`insmod c200i`

oder, falls ein anderer Linux-Kernel Verwendung findet:

`insmod -f c200i` (erkennt Mod. mit der I/O-Adresse `0x1E8` (hex) und wählt den Interrupt **5**)

oder, falls andere Adressen oder Interrupts gewünscht sind:

`insmod c200i io=0x` *`port`* `irq=0x` *`irq`*

mit *`port`* = eingestellte Port-Adresse

`IRQ` = gewünschter Interrupt

(3A. Entladen des CAN-Treibers)

Das Entladen des CAN-Treibers ist z.B. nach einem Update der lokalen Firmware zum Rücksetzen des Prozessors notwendig. Dies kann z.B. beim CAN-PCI/331-Modul über das folgende Kommando erfolgen:

`rmmod c331`

4. Kontrolle der Installation

Ob die Installation erfolgreich war, kann in dem folgenden Verzeichnis kontrolliert werden:

`/var/log/messages`

Nach der erfolgreichen Installation kann über die ntcan-API auf den CAN-Bus zugegriffen werden (Einbindung von '*libntcan.o*' in die Applikation).

A 3.2 Installation unter Linux-RTAI und Linux-RT

Hinweise:

- Die Installation kann nur mit Supervisor-Rechten erfolgen.

- Die **Syntax** für die Dateinamen, die Installation, den Start und das Update ist für **Linux-RTAI und Linux-RT** nahezu identisch.
  Im folgenden ist die Syntax für Linux-RTAI beschrieben.

  Für Linux-RT sind lediglich die Zeichen 'rtai' gegen 'rtlinux' zu ersetzen!

  Beispiel:
  Laden des CAN-Treibers unter Linux-RTAI mit dem Aufruf:
  `insmod c331_rtai`

  Laden des CAN-Treibers unter Linux-RT mit dem Aufruf:
  `insmod c331_rtlinux`

A 3.2.1 Dateien des Linux-RTAI-, bzw. des Linux-RT-Paketes

Die Software-Treiber für Linux-RTAI werden auf einer 3,5”-Diskette geliefert. Die Diskette enthält folgende Dateien:

<table>
<tr><th>Datei</th><th colspan="2">Beschreibung</th></tr>
<tr><td>readme.txt</td><td>aktuelle Anmerkungen und Hinweise</td><td rowspan="2">Die Buchstabenkombination ‘c331’ zeigt die Dateien des Moduls CAN-PCI/331 an. Bei anderen Modulen werden diese Buchstaben wie folgt ersetzt:
<table>
<tr><th>CAN-Modul</th><th>Dateiname</th></tr>
<tr><td>CAN-ISA/200</td><td rowspan="2">c200i_rtai</td></tr>
<tr><td>CAN-PC104/200
(nur SJA1000)</td></tr>
<tr><td>CAN-ISA/331
CAN-PC104/331</td><td>c331i_rtai</td></tr>
<tr><td>CAN-PCI/200</td><td>c200_rtai</td></tr>
<tr><td>CAN-PCI/331
CPCI-CAN/331
PMC-CAN/331</td><td>c331_rtai</td></tr>
<tr><td>CAN-PCI/360
CPCI-CAN/360</td><td>c360_rtai</td></tr>
</table></td></tr>
<tr><td>z.B.
c331_rtai</td><td>dynamisch ladbarer Treiber</td></tr>
<tr><td>ntcan.h</td><td colspan="2">Header für die ntcan-API</td></tr>
<tr><td>cantest.c</td><td colspan="2">Quell-Code des Beispielprogramms ‘cantest’</td></tr>
<tr><td>cantest_rtai</td><td colspan="2">Binär-File des Beispielprogramms ‘cantest’</td></tr>
</table>

Tabelle A3.2.1: Dateien des Linux-RTAI-Paketes

A 3.2.2 Ablauf der Installation unter Linux-RTAI/RT

1. Entpacken des Tar-Files oder der Tar-Disk

Zum Entpacken des Tarfiles muß folgendes Kommando aufgerufen werden:

```
tar -xvfz c331-vx.y.z-linux-rtai.tgz
```

Anmerkung: Die Eingabe '*c331*' muß für das Modul CAN-PCI/331 erfolgen. Bei anderen Modulen ist die in der folgenden Tabelle angegebene Buchstabenkombination einzusetzen. Dies gilt auch für die folgenden Kommandos.

CAN-Modul	Eingabe-Syntax
CAN-ISA/200	c200i
CAN-PC104/200 (nur SJA1000)	
CAN-ISA/331 CAN-PC104/331	c331i
CAN-PCI/200	c200
CAN-PCI/331 CPCI-CAN/331 PMC-CAN/331	c331
CAN-PCI/360 CPCI-CAN/360	c360

Tabelle A3.2.2: Eingabe-Syntax beim Entpacken der Tar-Files

2. Laden des CAN-Treibers (als Supervisor)

Der Treiber ist für die Linux-Kernels 2.2.14 und Linux-RTAI 1.2 (01.07.2000) getestet (Stand 25.06.98). Falls ein anderer Kernel verwendet wird, kann der Treiber mit der Option '`-f` ' auch hier geladen werden.

Für CAN-PCI/331, CPCI-CAN/331 und PMC-CAN/331:

```
insmod c331_rtai
```

oder, falls ein anderer Linux-Kernel Verwendung findet:

```
insmod -f c331_rtai
```

Für CAN-PCI/360, CPCI-CAN/360:

```
insmod c360_rtai
```

oder, falls ein anderer Linux-Kernel Verwendung findet:

```
insmod -f c360_rtai
```

Für CAN-ISA/331 und CAN-PC104/331:

```
insmod c331i_rtai
```

oder, falls ein anderer Linux-RTAI-Kernel Verwendung findet:

```
insmod -f c331i_rtai
```

(erkennt ein Module mit der I/O-Adresse `0x1E0` (hex) und ordnet ihm den IRQ **5** zu und erkennt außerdem ein Modul mit der Adresse `0x1E8` (hex) und ordnet ihm den IRQ **7** zu)

oder, falls andere Adressen oder Interrupts gewünscht sind:

```
insmod c331i_rtai io=0x port irq=0x irq
```

mit *port* = eingestellte Port-Adresse
IRQ = gewünschter Interrupt

Für CAN-ISA/200, CAN-PC104/200:

```
insmod c200i_rtai
```

oder, falls ein anderer Linux-RTAI-Kernel Verwendung findet:

```
insmod -f c200i_rtai
```

(erkennt Module mit der I/O-Adresse `0x1E8` (hex) und wählt den Interrupt **5**)

oder, falls andere Adressen oder Interrupts gewünscht sind:

```
insmod c200i io=0x port irq=0x irq
```

mit *port* = eingestellte Port-Adresse
IRQ = gewünschter Interrupt

(2A. Entladen des CAN-Treibers)

Das Entladen des CAN-Treibers ist z.B. nach einem Update der lokalen Firmware zum Rücksetzen des Prozessors notwendig. Dies kann z.B. beim CAN-PCI/331-Modul über das folgende Kommando erfolgen:

```
rmmod c331_rtai
```

3. Kontrolle der Installation

Ob die Installation erfolgreich war, kann in dem folgenden Verzeichnis kontrolliert werden:

```
/var/log/messages
```

Nach der erfolgreichen Installation kann über die ntcan-API auf den CAN-Bus zugegriffen werden.

A 3.3 Installation unter LynxOS

A 3.3.1 Dateien des LynxOS-Paketes

Die Software-Treiber für LynxOS werden auf einer 3,5”-Diskette geliefert. Die Diskette enthält folgende Dateien:

<table>
<tr><th>Datei</th><th colspan="2">Beschreibung</th></tr>
<tr><td>instcan</td><td colspan="2">Script zum Laden und Entladen des Treibers</td></tr>
<tr><td>README.c331</td><td>aktuelle Anmerkungen und Hinweise</td><td rowspan="3">Die Buchstabenkombination ‘c331’ zeigt die Dateien des Moduls CAN-PCI/331 an. Bei anderen Modulen werden diese Buchstaben wie folgt ersetzt:
<table><tr><th>CAN-Modul</th><th>Dateiname</th></tr><tr><td>VME-CAN2</td><td>ican2</td></tr><tr><td>VME-CAN4</td><td>ican4</td></tr><tr><td>CAN-ISA/331
CAN-PC104/331</td><td>c331i</td></tr><tr><td>CAN-PCI/331
CPCI-CAN/331
PMC-CAN/331</td><td>c331</td></tr><tr><td>CAN-PCI/360
CPCI-CAN/360</td><td>-</td></tr></table>-... zur Zeit noch nicht implementiert</td></tr>
<tr><td>c331</td><td>dynamisch ladbarer Treiber</td></tr>
<tr><td>c331.info</td><td>Parameter-Datei für den Treiber (wird bei Installation und Deinstallation gelesen)</td></tr>
<tr><td>ntcan.o</td><td colspan="2">ntcan-API</td></tr>
<tr><td>ntcan.h</td><td colspan="2">Header für die ntcan-API</td></tr>
<tr><td>canupd</td><td colspan="2">Programm zum Firmware-Update</td></tr>
<tr><td>cantest.c</td><td colspan="2">Quell-Code des Beispielprogramms ‘cantest’</td></tr>
<tr><td>cantest</td><td colspan="2">Binär-File des Beispielprogramms ‘cantest’</td></tr>
</table>

Tabelle A3.3.1: Dateien des LynxOS-Paketes

A 3.3.2 Ablauf der Installation unter LynxOS

1. Entpacken des Tarfiles

Zum Entpacken des Tarfiles muß folgendes Kommando aufgerufen werden:

```
tar -xvf c331-lynx-v1.0.0.tar
```

Anmerkung: Die Eingabe '*c331*' muß für das Modul CAN-PCI/331 erfolgen. Bei anderen Modulen ist die in der folgenden Tabelle angegebene Buchstabenkombination einzusetzen:

CAN-Modul	Eingabe
VME-CAN2	ican2
VME-CAN4	ican4
CAN-ISA/331 CAN-PC104/331	c331i
CAN-PCI/331 CPCI-CAN/331 PMC-CAN/331	c331
CAN-PCI/360 CPCI-CAN/360	-

-... zur Zeit noch nicht implementiert

Tabelle A3.3.2: Eingabe-Syntax beim Entpacken der Tar-Files

2. Laden des CAN-Treibers:

```
instcan c331
```

(2.B Entladen des CAN-Treibers:)

Das Entladen des CAN-Treibers ist z.B. nach einem Update der lokalen Firmware zum Rücksetzen des Prozessors notwendig. Dies kann z.B. beim CAN-PCI/331-Modul über das folgende Kommando erfolgen:

```
instcan -u c331
```

A 3.4 Installation unter PowerMAX OS

A 3.4.1 Dateien des PowerMAX OS-Paketes

Die Software-Treiber für PowerMAX OS werden auf einer 3,5”-Diskette geliefert. Die Diskette enthält folgende Dateien:

<table>
<tr><th>Datei</th><th colspan="2">Beschreibung</th></tr>
<tr><td>README.ican4</td><td>aktuelle Anmerkungen und Hinweise</td><td rowspan="2">Die Buchstabenkombination ‘ican4’ zeigt die Dateien des Moduls VME-CAN4 an. Bei anderen Modulen werden diese Buchstaben wie folgt ersetzt:
<table><tr><th>CAN-Modul</th><th>Dateiname</th></tr><tr><td>VME-CAN2</td><td>ican2</td></tr><tr><td>VME-CAN4</td><td>ican4</td></tr></table></td></tr>
<tr><td>z.B.
ican4</td><td>dynamisch ladbarer Treiber</td></tr>
<tr><td>libntcan.o</td><td colspan="2">ntcan-API</td></tr>
<tr><td>ntcan.h</td><td colspan="2">Header für die ntcan-API</td></tr>
<tr><td>cantest.c</td><td colspan="2">Quell-Code des Beispielprogramms ‘cantest’</td></tr>
<tr><td>cantest</td><td colspan="2">Binär-File des Beispielprogramms ‘cantest’</td></tr>
</table>

Tabelle A3.4.1: Dateien des PowerMAX OS-Paketes

A 3.4.2 Ablauf der Installation unter PowerMAX OS

1. Kopieren des Tar-Files `vmecan4_v1.2.tar` **in das Home-Verzeichnis**

2. Entpacken des Tar-Files:
Zum Entpacken des Tarfiles muß folgendes Kommando aufgerufen werden:

```
tar xvf vmecan4_v1.2/ican4/
```

Anmerkung: Die Eingabe '*ican4*' muß für das Modul VME-CAN4 erfolgen. Bei anderen Modulen ist die in der folgenden Tabelle angegebene Buchstabenkombination einzusetzen. Dies gilt auch für die folgenden Kommandos.

CAN-Modul	Eingabe-Syntax
VME-CAN2	`ican2`
VME-CAN4	`ican4`

Tabelle A3.4.2: Eingabe-Syntax beim Entpacken der Tar-Files

3. Erstellen der 'inodes' (als Supervisor)

```
cd 4_2/vmecan4_v1.2/ican4/
```
oder
```
cd 4_3/vmecan4_v1.2/ican4/
```

4. Nur bei der ersten Installation: Adapterdefinition hinzufügen
Bei der ersten Installation der CAN-Module müssen die folgenden Zeilen zu der Datei `/usr/include/sys/adapt3er_vme.h/` hinzugefügt werden:

```
#define ADAPTER_ICAN2   (AVB+0x12)   /* 0x212 - esd ican2 */
#define ADAPTER_ICAN4   (AVB+0x13)   /* 0x213 - esd ican4 */
```

5. Nur bei Update eines vorhandenen Treibers:
Soll ein vorhandener Treiber aktualisiert werden, ist folgende Eingabe notwendig:
```
modadmin -U ican4
```

6. Installation
- bei der ersten Installation:
```
./install -f
```

mit anschließendem Reboot

- bei einem Update:
```
./install -u
```

7. Verzeichnis wechseln:
Wechseln zu
`~/4_2/vmecan4_v1.2` für PowerMAX OS 4.2
oder
`~/4_3/vmecan4_v1.2` für PowerMAX OS 4.3

8. Starten des Treibers:
Nach dem Aufruf von
`modadmin -l ican4`
zeigt der Treiber seine Start-Message an.

9. Aufruf von `cantest`:
Nach dem Aufruf von `cantest` mit
`./cantest`
zeigt das Programm vier verfügbare CAN-Netze an (net 0...3)

Nach der erfolgreichen Installation kann über die ntcan-API auf den CAN-Bus zugegriffen werden (Einbindung von '*libntcan.o*' in die Applikation).

A 3.5 Installation unter Solaris

A 3.5.1 Dateien des Solaris-Paketes

Die Software-Treiber für Solaris werden auf einer 3,5”-Diskette geliefert. Die Diskette enthält folgende Dateien:

<table>
<tr><th>Datei</th><th colspan="2">Beschreibung</th></tr>
<tr><td>install</td><td colspan="2">Script zur Installation des Treibers</td></tr>
<tr><td>README.c331</td><td>aktuelle Anmerkungen und Hinweise</td><td rowspan="3">Die Buchstabenkombination ‘c331’ zeigt die Dateien des Moduls CAN-PCI/331 an. Bei anderen Modulen werden diese Buchstaben wie folgt ersetzt:

<table>
<tr><th>CAN-Modul</th><th>Kennung</th></tr>
<tr><td>CAN-ISA/200</td><td rowspan="2">-</td></tr>
<tr><td>CAN-PC104/200 (nur SJA1000)</td></tr>
<tr><td>CAN-ISA/331
CAN-PC104/331</td><td>c331i</td></tr>
<tr><td>CAN-PCI/200</td><td>-</td></tr>
<tr><td>CAN-PCI/331
CPCI-CAN/331
PMC-CAN/331</td><td>c331</td></tr>
<tr><td>CAN-PCI/360
CPCI-CAN/360</td><td>-</td></tr>
</table>
-... zur Zeit noch nicht implementiert</td></tr>
<tr><td>c331</td><td>dynamisch ladbarer Treiber</td></tr>
<tr><td>c331.conf</td><td>Parameter-Datei für den Treiber (wird bei Installation und Deinstallation gelesen)</td></tr>
<tr><td>ntcan.o</td><td colspan="2">ntcan-API</td></tr>
<tr><td>ntcan.h</td><td colspan="2">Header für die ntcan-API</td></tr>
<tr><td>canupd</td><td colspan="2">Programm zum Firmware-Update</td></tr>
<tr><td>cantest.c</td><td colspan="2">Quell-Code des Beispielprogramms ‘cantest’</td></tr>
<tr><td>cantest</td><td colspan="2">Binär-File des Beispielprogramms ‘cantest’</td></tr>
</table>

Tabelle A3.5.1: Dateien des Solaris-Paketes

A 3.5.2 Ablauf der Installation unter Solaris

1. Entpacken des Tarfiles

Zum Entpacken des Tarfiles muß folgendes Kommando aufgerufen werden:

```
tar -xvf c331-solaris-vx.x.x.tar
```

(x.x.x = Treiber-Versionsnummer)

Anmerkung: Die Eingabe '*c331*' muß für das Modul CAN-PCI/331 erfolgen. Bei anderen Modulen ist die in der folgenden Tabelle angegebene Buchstabenkombination einzusetzen:

CAN-Modul	Eingabe-Syntax
CAN-ISA/200	-
CAN-PC104/200 (nur SJA1000)	
CAN-ISA/331 CAN-PC104/331	c331i
CAN-PCI/200	-
CAN-PCI/331 CPCI-CAN/331 PMC-CAN/331	c331
CAN-PCI/360 CPCI-CAN/360	-

-... zur Zeit noch nicht implementiert

Tabelle A3.5.2: Eingabe-Syntax beim Entpacken der Tar-Files

2. Vorbereitung des Systems für die dynamische Erzeugung von 'inodes':

Damit bei der Installation des Treibers automatisch 'inodes' unterhalb des `/dev`-Verzeichnisses erzeugt werden, muß die Datei `/etc/devlink.tab` angepaßt werden. Dies muß mit Supervisor-Rechten geschehen. Vor der Änderung sollte ein Backup dieser Datei erzeugt werden. Folgende Zeile muß in der Datei ergänzt werden:

```
type=can;    \M0
```

Bei der Erstinstallation des Treibers sollten nun automatisch die Dateien `/dev/can`*x* angelegt werden, wobei *x* die (hexadezimale) Netznummer angibt.

3. Anpassen der Konfigurationsdatei

3.1 Anpassen der Konfigurationsdatei bei CAN-ISA-Modulen:

Vor der Erstinstallation des Treibers muß dessen Treiber-Konfigurationsdatei an die Konfiguration der Hardware angepaßt werden. Folgende Einträge (Properties) müssen in *driver*.conf vorhanden sein, wobei *driver* durch den entsprechenden Treibernamen ersetzt werden muß. Lediglich die kursivgeschriebenen Properties müssen angepaßt werden. Für eine zusammenfassende Übersicht der (busspezifischen) Properties sei auf die entsprechenden Manual-Pages (driver.conf, sysbus, pci) verwiesen.

Name	Parameter	Bedeutung
name	String	Treibername
class	String	Bustyp
interrupts	Numeric	Interrupt Vektor
interrupt-priorities	Numeric	Interrupt Prioritäts Level (IPL)
reg	Numeric	Drei durch Kommas getrennte Werte, wobei der Zweite der CAN Interface-Basisadresse entspricht und der dritte die größe des I/O-Bereichs in Byte beschreibt.

Tabelle A3.5.3: Properties der Driver-Konfigurationsdatei

Der in *reg* angegebene Wert muß mit dem auf der Hardware durch Steckbrücken oder Kodierschalter konfigurierten Basisadresse übereinstimmen. Der IPL-Level muß auf einen High-Level Interrupt gelegt werden und der Interrupt-Vektor darf noch nicht durch eine andere Hardware-Komponente belegt sein. Nachfolgend ist eine Beispiel-Konfigurationsdatei für eine CAN-ISA/331 dargestellt, deren Basisadresse auf 0x1E0 (hex) konfiguriert wurde und die den Interrupt-Vektor 7 mit einem IPL von 11 benutzen soll:

```
# Copyright (c) 1998, by esd gmbh.
#
name="isa331" class="sysbus" interrupts=7 interrupt-priorities=11 reg=1,0x1e0,8;
```

3.2 Anpassung der Konfigurationsdatei bei CAN-PCI-Modulen:

Da sich die PCI-Geräte selbständig anmelden, erfolgt die Konfiguration und Zuordnung der Resource automatisch beim Hochlauf des Systems (Plug&Play). Der Gerätetreiber übernimmt die zugeordnete Resource automatisch, so daß keine manuelle Zuordnung in der Treiber-Konfigurationsdatei notwendig ist. Der einzige Parameter, der eingestellt werden kann ist die Interrupt-Priorität, die der IPL-Treiber verwenden soll (siehe auch Beispiel c331.conf).

```
# Copyright (c) 1999-2000 electronic system design gmbh
#

# do not remove the next line
interrupt-priorities=9;
```

4. Installation des CAN-Treibers

Zur Installation des Treibers muß das Skript `install` als Superuser ausgeführt werden. Es kopiert die Treiberdateien in das Zielverzeichnis, installiert und startet den Treiber. Von nun an wird der Treiber mit jedem Systemstart automatisch geladen und gestartet.

Nach der erfolgreichen Installation kann über die ntcan-API auf den CAN-Bus zugegriffen werden (Einbindung von '*ntcan.o*' in die Applikation).

5. Prüfen der Installation

5.1 Nur bei CAN-PCI-Modulen:

Wenn das Installations-Script fehlerfrei ausgeführt wurde, wird der Treiber gestartet und automatisch mit jedem folgenden System-Hochlauf geladen.
Durch Eingabe von `"modinfo | grep d3x"` im Root können Sie prüfen, ob der Treiber geladen worden ist. Es müßte eine Ausgabe ähnlich der folgenden erscheinen:

```
205 f5d11000   34d9 132   1  c331 (CAN PCI/331 driver v1.3.3)
```

5.2 Für alle Module:

Sie können prüfen, ob die Installation erfolgreich war, indem Sie in der folgenden Datei die Driver-Boot-Message lesen:

```
/var/adm/messages
```

6. Entladen des CAN-Treibers

Das Entladen des CAN-Treibers ist z.B. nach einem Update der lokalen Firmware zum Rücksetzen des Prozessors notwendig. Dies kann z.B. beim CAN-ISA/331-Modul über das folgende Kommando erfolgen, das als Superuser ausgeführt werden muß:

```
rem_drv c331i
```

A 3.6 Installation unter SGI-IRIX6.5

A 3.6.1 Dateien des SGI-IRIX6.5-Work-Paketes

Die Software-Treiber für SGI-IRIX6.5 werden auf einer 3,5”-Diskette geliefert. Die Diskette enthält folgende Dateien:

<table>
<tr><td>Datei</td><td colspan="2">Beschreibung</td></tr>
<tr><td>c331</td><td>CAN-Treiber (Object-Code)</td><td rowspan="3">Die Buchstabenkombination ‘c331’ zeigt die Dateien des Moduls CAN-PCI/331 an. Bei anderen Modulen werden diese Buchstaben wie folgt ersetzt:
<table><tr><td>CAN-Modul</td><td>Kennung</td></tr><tr><td>CAN-ISA/200
CAN-PC104/200</td><td>-</td></tr><tr><td>CAN-ISA/331
CAN-PC104/331</td><td>-</td></tr><tr><td>CAN-PCI/331
CPCI-CAN/331
PMC-CAN/331</td><td>c331</td></tr><tr><td>CAN-PCI/360
CPCI-CAN/360</td><td>-</td></tr></table>-... zur Zeit noch nicht implementiert</td></tr>
<tr><td>c331.master</td><td>Treiber-Konfigurationsdatei</td></tr>
<tr><td>c331.sm</td><td>Treiber-Konfigurationsdatei</td></tr>
<tr><td>makefile</td><td colspan="2">makefile zur Installation und zum Laden des Treibers</td></tr>
<tr><td>ntcan.o</td><td colspan="2">ntcan-API</td></tr>
<tr><td>ntcan64.o</td><td colspan="2">ntcan-API / 64-Bit-Version</td></tr>
<tr><td>ntcan.h</td><td colspan="2">Header für die ntcan-API</td></tr>
<tr><td>cantest.c</td><td colspan="2">Quell-Code des Beispielprogramms ‘cantest’</td></tr>
<tr><td>cantest</td><td colspan="2">Binär-File des Beispielprogramms ‘cantest’</td></tr>
</table>

Tabelle A3.6.1: Dateien des SGI-IRIX6.5-Paketes

A 3.6.2 Ablauf der Installation unter SGI-IRIX6.5

1. Login als Root

2. Entpacken des Tarfiles
Zum Entpacken des Tarfiles muß folgendes Kommando aufgerufen werden:

```
tar -xvf c331-IPXX-v2.1.0.tar
```

mit *XX* = Prozessorkennung (z.B. '32' für SGI-O2)

Anmerkung: Die Eingabe '*c331*' muß für das Modul CAN-PCI/331 erfolgen. Bei anderen Modulen ist die in der folgenden Tabelle angegebene Buchstabenkombination einzusetzen:

CAN-Modul	Eingabe
CAN-ISA/200 CAN-PC104/200	-
CAN-ISA/331 CAN-PC104/331	-
CAN-PCI/331 CPCI-CAN/331 PMC-CAN/331	c331
CAN-PCI/360 CPCI-CAN/360	-

-... zur Zeit noch nicht implementiert

Tabelle A3.6.2: Eingabe-Syntax beim Entpacken der Tar-Files

3. Verzeichnis wechseln

```
cd c331-IPXX-v2.1.0
```

4. Treiberdateien installieren

```
smake install
```

5. Treiber laden

```
smake load
```

Ist der Treiber korrekt installiert und geladen, so muß auf dem Monitor jetzt eine Meldung des Treibers erscheinen.

A 3.7 Installation unter AIX

A 3.7.1 Besonderheiten der AIX-Implementierung

- der CAN-Treiber ist für die Betriebssystem-Version AIX 4.2.1 ausgelegt
- Hardware-Plattform: PowerPC
- implementierte esd-CAN-Module: CAN-PCI/331, CPCI-CAN/331
- 29-Bit-Identifier werden zur Zeit noch nicht unterstützt

A 3.7.2 Dateien des AIX-Paketes

Die Software-Treiber für AIX wird auf einer 3,5”-Diskette geliefert. Die Diskette enthält eine Tar-Datei.

<table>
<tr><th>Datei</th><th colspan="2">Beschreibung</th></tr>
<tr><td>README.c331</td><td>aktuelle Anmerkungen und Hinweise</td><td rowspan="2">Die Buchstabenkombination ‘c331’ zeigt die Dateien des Moduls CAN-PCI/331 an. Bei anderen Modulen wird diese Zeichenkette u. U. wie folgt ersetzt:
<table>
<tr><th>CAN-Modul</th><th>Kennung</th></tr>
<tr><td>VME-CAN2</td><td>-</td></tr>
<tr><td>CAN-ISA/331
CAN-PC104/331</td><td>-</td></tr>
<tr><td>CAN-PCI/331
CPCI-CAN/331
PMC-CAN/331</td><td>c331</td></tr>
<tr><td>CAN-PCI/360
CPCI-CAN/360</td><td>-</td></tr>
</table>
-... zur Zeit noch nicht implementiert</td></tr>
<tr><td>c331</td><td>dynamisch ladbarer Treiber</td></tr>
<tr><td>ntcan.o</td><td colspan="2">ntcan-API</td></tr>
<tr><td>ntcan.h</td><td colspan="2">Header für die ntcan-API</td></tr>
<tr><td>nttest.c</td><td colspan="2">Quell-Code des Beispielprogramms ‘cantest’</td></tr>
</table>

Tabelle A3.7.1: Dateien des AIX-Paketes

A 3.7.3 Ablauf der Installation unter AIX

1. Login als Root

2. Entpacken des Tarfiles
Zum Entpacken des Tarfiles muß folgendes Kommando aufgerufen werden:

```
tar -xvf c331-ppc-vx.y.z.tar
```

(x.y.z = Treiber-Versionsnummer)

Anmerkung: Die Eingabe '*c331*' muß für das Modul CAN-PCI/331 erfolgen. Bei anderen Modulen ist u. U. eine andere Zeichenkombination notwendig:

CAN-Modul	Eingabe
VME-CAN2	-
CAN-ISA/331 CAN-PC104/331	-
CAN-PCI/331 CPCI-CAN/331 PMC-CAN/331	c331
CAN-PCI/360 CPCI-CAN/360	-

-... zur Zeit noch nicht implementiert

Tabelle A3.7.2: Eingabe-Syntax beim Entpacken der Tar-Files

3. Kopieren der Dateien
Beim Entpacken wird das Verzeichnis `/c331-ppc-vx.y.z` erstellt.
- Kopieren Sie aus diesem Verzeichnis die Dateien `cfg_pci331` und `ucfg_pci331` in das Verzeichnis `/usr/lib/methods/`.
- Kopieren Sie aus diesem Verzeichnis die Datei `c331` in das Verzeichnis `/usr/lib/drivers/pci/`.

4. Einträge in die Datenbank erstellen

```
run odmadd pci331
```

5. Aufrufen des Konfigurations-Managers

```
cfgmgr -v
```

6. Kontrolle der Dateien und Erstellen der symbolischen Verknüpfungen

Kontrollieren Sie, ob die Geräte `c33100` und `c33101` im Verzeichnis `/dev/` eingetragen sind. Erstellen Sie eine symbolische Verknüpfung:

```
ln -s /dev/c33100 /dev/can0
ln -s /dev/c33101 /dev/can1
```

Die Installation des Treibers ist jetzt abgeschlossen.

7. Funktionsfähigkeit des Treibers mit CANscope prüfen

- Vergewissern Sie sich, daß die Verdrahtung und Abschlüsse korrekt installiert sind und stellen Sie sicher, daß mindestens ein anderer funktionsfähiger CAN-Teilnehmer angeschlossen ist!

- Starten Sie das Monitorprogramm CANscope:
  `./canscope &`

- Stellen Sie die Baudrate Ihres CAN-Netzwerkes in CANscope für Netz 0 ein und drücken Sie die Schaltfläche *Init*.
  Im Anzeigefenster erscheint die Meldung ‘Init done’.

- Drücken Sie die Schaltfläche *Add Id*, um den CAN-Identifier-Bereich von 0 bis 2047 auszuwählen.

- Drücken Sie die Schaltfläche *Start*, um die Nachrichten des CAN-Bus zur Anzeige zu bringen.

8. Testprogramm `cantest`

Neben CANscope, das mit einer UNIX-typischen Oberfläche ausgestattet ist, kann unter AIX auch das Testprogramm `cantest`, das über Kommandozeileneingabe bedient wird, verwendet werden (Beschreibung siehe Kapitel ‘Testprogramm cantest’ ab Seite 50).

- cantest kompilieren:
  `gcc -o cantest nttest.c ntcan.o`

- canstest aufrufen:
  `./cantest`

A 3.8 Installation unter VxWorks

A 3.8.1 Dateien des VxWorks-Paketes

Die Software-Treiber für VxWorks werden auf einer 3,5”-Diskette geliefert. Die Diskette enthält folgende Dateien:

<table>
<tr><th>Datei</th><th colspan="2">Beschreibung</th></tr>
<tr><td>ldc331i</td><td>Script zum Laden des Treibers</td><td rowspan="5">Die Buchstabenkombination ‘c331’ zeigt die Dateien des Moduls CAN-ISA/331 an. Bei anderen Modulen werden diese Buchstaben wie folgt ersetzt:
<table>
<tr><th>CAN-Modul</th><th>Kennung</th></tr>
<tr><td>CAN-ISA/200
CAN-PC104/200</td><td>c200i</td></tr>
<tr><td>CAN-ISA/331
CAN-PC104/331</td><td>c331i</td></tr>
<tr><td>CAN-PCI/200</td><td>-</td></tr>
<tr><td>CAN-PCI/331
CPCI-CAN/331
PMC-CAN/331</td><td>c331</td></tr>
<tr><td>CAN-PCI/360
CPCI-CAN/360</td><td>-</td></tr>
</table>
-... zur Zeit noch nicht implementiert</td></tr>
<tr><td>c331i.sys</td><td>CAN-Treiber (Binär-Code)</td></tr>
<tr><td>c331iini</td><td>Beispielprogramm zum Starten des Treibers (Binär-Code)</td></tr>
<tr><td>c331iini.c</td><td>Beispielprogramm zum Starten des Treibers (Quell-Code)</td></tr>
<tr><td colspan="2"></td></tr>
<tr><td>ntcan.o</td><td colspan="2">ntcan-API</td></tr>
<tr><td>ntcan.h</td><td colspan="2">Header für die ntcan-API</td></tr>
<tr><td>cantest.c</td><td colspan="2">Quell-Code des Beispielprogramms ‘cantest’</td></tr>
<tr><td>cantest</td><td colspan="2">Binär-File des Beispielprogramms ‘cantest’</td></tr>
</table>

Tabelle A3.8.1: Dateien des VxWorks-Paketes

A 3.8.2 Hinweise zu VxWorks

A 3.8.2.1 Konfiguration des VxWorks-Treibers

Die Konfiguration muß für alle esd-CAN-Module unter VxWorks durchgeführt werden. Die folgende Beschreibung der Konfigurationsparameter in der Struktur CAN_INFO bezieht sich auf CAN-ISA-Module. Die Parameter `base` und `irq` haben für CAN-PCI-Module eine andere Funktion als für CAN-ISA-Module. Im anschließenden Kapitel 'Besonderheiten von PCI-CAN-Modulen unter VxWorks' werden diese beschrieben.

Zur Konfiguration des CAN-Treibers, muß die entsprechende Installationsfunktion mit einem Pointer auf die Startadresse der Struktur CAN_INFO aufgerufen werden. Das Modul `caninit.c` ist ein Beispiel für den Start des Treibers. Die Struktur CAN_INFO ist wie folgt aufgebaut:

```
struct CAN_INFO
{
    unsigned long base;
    unsigned char net[4];
    unsigned char prio;
    unsigned char irq;
    unsigned char reserved[6];
};
```

`base` CAN-ISA-Module: I/O-Adresse des CAN-ISA-Moduls
`base` muß auf den Wert eingestellt werden, der auf der Hardware per Steckbrücken oder Kodierschalter eingestellt ist. Wird `base` auf '0' gesetzt, so bricht der Treiber die Suche nach weiteren CAN-Schnittstellen ab.
CAN-PCI-Module: siehe folgendes Kapitel.

`net[0]` Logische Netznummer, die dem ersten physikalischen CAN-Port auf diesem Modul zugewiesen werden soll
Besitzt ein Modul mehr als ein CAN-Port, so werden die Ports beginnend mit dieser Nummer fortlaufend durchnumeriert.
Die Werte von `net[1]` bis `net[3]` werden vom Treiber ignoriert.
Der Anwender muß sicherstellen, daß in einem System alle Netznummern nur einmalig vergeben werden, andernfalls kann die Initialisierung des Treibers fehlschlagen.

`prio` Priorität der Backend-Task, die für jedes physikalische CAN-Interface gestartet wird

`irq` CAN-ISA-Module: Interrupt-Vektor für dieses CAN-ISA-Modul
CAN-PCI-Module: siehe folgendes Kapitel.

A 3.8.2.2 Besonderheiten von PCI-CAN-Modulen unter VxWorks

x86-Targets

Bei x86-Targets werden die PCI-Boards in der Regel durch das BIOS konfiguriert (Plug& Play). Die Gerätetreiber sind in der Lage, diese Konfiguration zu verwenden, wenn das BSP genug ungenutzte Einträge in der MMU-Memory-Descriptor-Tabelle besitzt, um den PCI-Adreß-Raum des CAN-Moduls dort einzutragen (was in der Regel der Fall ist).

Um dem Treiber mitzuteilen, daß er die Konfiguration aus dem BIOS-Setup verwenden soll, müssen Sie die Basisadresse (`base`) und den Interrupt (`irq`) in der Struktur CAN-INFO im Modul `caninit` auf '-1' setzen.

Power PC-Targets

- ohne Plug&Play-Unterstützung
  Einige PowerPC basierte Targets unterstützen die PCI-Konfiguration vor dem Hochlauf von VxWorks nicht. Hier müssen Sie selbst 3 MByte *freien* physikalischen Adreßraum und den Interrupt, den Ihr Target für den PCI-Slot, in dem das CAN-Modul steckt verwendet, finden. Tragen Sie die Basisadresse (`base`) und den Interrupt (`irq`) in die Struktur CAN_INFO der Funktion *caninit* ein, um dem Treiber die korrekten Werte in der Hochlaufphase mitzuteilen. Für die Basisadresse muß hierbei eine gerader Zahlenwert eingetragen werden. Der Interrupt-Vektor muß mit dem BSP-spezifischen Offset versehen werden.

- mit Plug&Play-Unterstützung
  Unterstützt das Target Plug&Play, so muß für die Basisadresse (`base`) der Wert '-1' eingetragen werden. In diesem Fall wird der PCI-Adreßraum bei einem Offset von 0xC0000000 (hex) angesprochen. Der Wert des Offsets des Adreßraums ist im Treiber festgelegt.
  Auch in diesem Fall muß der Interrupt-Vektor mit dem BSP-spezifischen Offset versehen werden.

A 3.8.2.3 Starten des Treibers

Zum Starten des Treibers ist

```
c331iStart()
```

einzugeben

Anmerkung: Die Eingabe '*c331*' muß für das Modul CAN-PCI/331 erfolgen. Bei anderen Modulen ist die in der folgenden Tabelle angegebene Buchstabenkombination einzusetzen. Dies gilt auch für die folgenden Kommandos.

CAN-Modul	Eingabe-Syntax
CAN-ISA/200	`c200i`
CAN-PC104/200 (nur SJA1000)	
CAN-ISA/331 CAN-PC104/331	`c331i`
CAN-PCI/200	-
CAN-PCI/331 CPCI-CAN/331 PMC-CAN/331	`c331`
CAN-PCI/360 CPCI-CAN/360	-

Tabelle A3.8.2: Eingabe-Syntax zum Starten des Treibers

A 3.8.2.3 Testprogramm `canTest` unter VxWorks

Mit dem Testprogramm `canTest` läßt sich die Funktion der CAN-Schnittstelle prüfen. `canTest` ist im Kapitel 'Testprogramm cantest' ab Seite 50 beschrieben. Da VxWorks standardmäßig weniger Parameter in der Kommandozeile zuläßt, als `canTest` benötigt, müssen die Parameter bei VxWorks als String, also in Hochkommas angegeben werden. Der String wird vom Testprogramm ausgewertet. Für alle Parameter, die nicht im String enthalten sind, werden die Default-Werte angenommen.

A 3.8.2.4 CAN2.0B-Unterstützung (29-Bit Identifier) unter VxWorks

Unter VxWorks wird CAN2.0B zur Zeit bei den Modulen CAN-PCI/331, CAN-CPCI331, CAN-PMC/331, sowie CAN-ISA/200, CAN-PC104/200, CAN-ISA/331 und CAN-PC104/331 unterstützt.

Ab Firmware-Version 0.C.09 bietet das CAN-Modul zwei Betriebsarten:
1. CAN 2.0A mit 11-Bit Identifiers und passiver CAN 2.0B-Unterstützung
2. CAN 2.0B mit aktiver 29-Bit Identifier-Unterstützung

Während der Treiber gestartet wird, wird der aktuelle Firmware-Mode an *stdout* ausgegeben.

Die Umschaltung zwischen den beiden Betriebsarten erfolgt während der Treiber läuft durch den Aufruf von

```
c331_switch (op, net)
```

`op` Betriebsart (Operation Mode)
Wird für `op` der Wert '0' eingetragen, so wird die aktuelle Betriebsart angezeigt. Wird der Wert '1' eingetragen, so nimmt die Firmware die neue Betriebsart zur Kenntnis, schaltet aber erst nach dem nächsten Hochlauf auf die neue Betriebsart um.

`net` logische Netz-Nummer

A 3.8.2.5 Entladen des Treibers

Ein komplettes Entladen des Gerätetreibers ist aufgrund der asymmetrischen APIs für allocation/deallocation der treiberspezifischen Ressourcen unter VxWorks nicht möglich.

Weitere Hinweise zu VxWorks:

Einsatz verschiedener CAN-Interfaces auf dem selben Target

Es ist nicht möglich, unter VxWorks verschiedene CAN-Interfaces, wie z.B. CAN-PCI/331 und CAN/ISA/331 gleichzeitig auf einem Target zu betreiben.

A 3.9 Installation unter QNX4

Die Installation kann nur mit Supervisor-Rechten erfolgen.

A 3.9.1 Dateien des QNX4-Paketes

Die Software-Treiber für QNX4 werden auf einer 3,5”-Diskette geliefert. Die Diskette enthält folgende Dateien:

<table>
<tr><th>Datei</th><th colspan="2">Beschreibung</th></tr>
<tr><td>readme.c331</td><td>aktuelle Anmerkungen und Installationshinweise</td><td rowspan="3">Die Buchstabenkombination ‘c331’ zeigt die Dateien des Moduls CAN-PCI/331 an. Bei anderen Modulen werden diese Buchstaben wie folgt ersetzt:

<table>
<tr><th>CAN-Modul</th><th>Kennung</th></tr>
<tr><td>CAN-ISA/200</td><td rowspan="2">c200i</td></tr>
<tr><td>CAN-PC104/200 (nur SJA1000)</td></tr>
<tr><td>CAN-ISA/331
CAN-PC104/331</td><td>c331i</td></tr>
<tr><td>CAN-PCI/331
CPCI-CAN/331
PMC-CAN/331</td><td>c331</td></tr>
<tr><td>CAN-PCI/360
CPCI-CAN/360</td><td>-</td></tr>
</table>
-... zur Zeit noch nicht implementiert</td></tr>
<tr><td>c331</td><td>CAN-I/O-Manager für CAN-PCI/331</td></tr>
<tr><td>updc331</td><td>Programm zum Firmware-Update</td></tr>
<tr><td>libntcan.a</td><td colspan="2">CAN-Access-API (Library)</td></tr>
<tr><td>ntcan.h</td><td colspan="2">Header für die CAN-Access-API</td></tr>
<tr><td>cantest.c</td><td colspan="2">Quell-Code des Beispielprogramms ‘cantest’</td></tr>
<tr><td>cantest</td><td colspan="2">Binär-File des Beispielprogramms ‘cantest’</td></tr>
</table>

Tabelle A3.9.1: Dateien des QNX4-Paketes

A 3.9.2 Ablauf der Installation unter QNX4

A 3.9.2.1 Installation von ISA-Boards

Führen Sie zuerst die Hardware-Installation des Moduls aus. Achten Sie bei ISA-Boards auf die korrekt eingestellte Board-Adresse. Bei Auslieferung sollte bereits die für die Standardkonfiguration zutreffende Adresse (z.B. `0x1E8` (hex) für CAN-ISA/200) eingestellt sein.

Aufruf des I/O-Managers
Loggen Sie sich als 'root' ein.
Starten Sie den I/O-Manager: '`c200i &`'

Nach dem Aufruf muß auf dem Bildschirm eine Ausgabe ähnlich der folgenden erscheinen:

```
C200i[0x1e8]: Using I/O-Base 0x1E8
C200i[0x1e8]: Using Interrupt 5
C200i[0x1e8]: "CAN_ISA200" with 1 Nets identified
C200i[0x1e8]: Hardware-Version=1.0.00
C200i[0x1e8]: Firmware-Version=0.0.00
C200i[0x1e8]: Driver-Version  =1.0.00
C200i[0x1e8]: Net 0 successfully created
```

Anmerkung: Der I/O-Manager wird in Abhängigkeit vom Modul über verschiedene Kommandos aufgerufen. Im oben aufgeführten Beispiel ist die Zeichenkombination '*`c200i`*' bei anderen Modulen wie folgt zu ersetzen:

CAN-Modul	Eingabe-Syntax
CAN-ISA/200	`c200i`
CAN-PC104/200 (nur SJA1000)	
CAN-ISA/331 CAN-PC104/331	`c331i`

Tabelle A3.9.2: Eingabe-Syntax beim Aufruf des I/O-Managers bei ISA-Boards

Kommando-Zeile des I/O-Managers für ISA-Boards

Der I/O-Manager kann durch die im folgenden beschriebenen Parameter konfiguriert werden.

Parameter	Funktion
-h	Anzeige eines Hilfe-Textes.
-n *net*	Zuweisung der logischen Netznummer. *net* = 0...255 default: *net* = 0
-p *port*	Selektion der ISA-Karte im System über die per Hardware eingestellte Port-Adresse (siehe Hardware-Handbuch des Moduls).
-i *irq*	Setzen des Interrupts, den das Board verwenden soll. Es ist eine freier Interrupt im System zu ermitteln, der hier eingestellt werden muß. Wird dieser Parameter nicht übergeben, so wird der Interrupt 5 verwendet.

Tabelle A3.9.3: Parameter des I/O-Managers bei ISA-Boards

Beispiel für die Installation zweier CAN-ISA/200-Boards in einem System:

Aufruf	Funktion
`c200i -p0x1e8 -i5 -n0 &`	Dem CAN-ISA/200-Modul mit der Adresse 0x1E8 (hex) werden der Interrupt 5 und die CAN-Netz-Nummer 0 zugewiesen
`c200i -p0x1e0 -i7 -n1 &`	Dem CAN-ISA/200-Modul mit der Adresse 0x1E0 (hex) werden der Interrupt 7 und die CAN-Netz-Nummer 1 zugewiesen

Tabelle A3.9.4: Beispiel für die Installation zweier CAN-ISA/200-Boards

Sie können jetzt die Kommunikation auf dem CAN-Bus mit dem Beispiel-Programm `cantest` testen.

Achtung:	Achten Sie vorher auf eine funktionsfähige Verdrahtung! Damit das Modul korrekt arbeiten kann, ist ein zweiter CAN-Teilnehmer notwendig, da sonst der CAN-Controller Fehlermeldungen erzeugt. Achten Sie auf korrekte Bus-Terminierung.

A 3.9.2.2 Installation von PCI-Boards

Führen Sie zuerst die Hardware-Installation des Moduls aus.

Loggen Sie sich als ‘root’ ein.
Starten Sie den I/O-Manager: ‘`c331 &`’

Jetzt muß auf dem Bildschirm eine Ausgabe ähnlich der folgenden erscheinen:

```
C331[0]: Using Interrupt 12
C331[0]: "CAN_PCI331" with 2 Nets identified
C331[0]: Hardware-Version=1.1.00
C331[0]: Firmware-Version=0.c.00
C331[0]: Driver-Version  =1.0.00
C331[0]: Net 0: Successfully created
C331[0]: Net 1: Successfully created
```

Anmerkung: Der I/O-Manager wird u. U. in Abhängigkeit vom Modul über verschiedene Kommandos aufgerufen. Im oben aufgeführten Beispiel ist die Zeichenkombination ‘*`c331`*’ bei anderen Modulen wie folgt zu setzen:

CAN-Modul	Eingabe-Syntax
CAN-PCI/331 CPCI-CAN/331 PMC-CAN/331	`c331`
CAN-PCI/360 CPCI-CAN/360	-

-... zur Zeit noch nicht implementiert

Tabelle A3.9.5: Eingabe-Syntax beim Aufruf des I/O-Managers bei PCI-Boards

Kommando-Zeile des I/O-Managers für PCI-Boards

Der I/O-Manager kann durch die im folgenden beschriebenen Parameter konfiguriert werden.

Parameter	Funktion
-h	Anzeige eines Hilfe-Textes.
-n *net*	Zuweisung der logischen Netznummer. *net* = 0...255 default: *net* = 0
-p *port*	Selektion des esd-CAN-PCI-Boards im System. Die Boards werden mit '0' beginnend durchnumeriert. Die Zuordnung zwischen den Nummern und den Boards legt der Plug&Play-Controller fest. Im Zweifelsfall sollte die Zuordnung durch einen Test überprüft werden.

Tabelle A3.9.6: Parameter des I/O-Managers bei PCI-Boards

Beispiel für die Installation zweier CAN-PCI/331-Boards in einem System:

Aufruf	Funktion
`c331 -p0 -n0 &`	Dem CAN-PCI/331-Modul mit der PCI-Nummer 0 werden die CAN-Netz-Nummern 0 und 1 zugewiesen.
`c331 -p1 -n2 &`	Dem CAN-PCI/331-Modul mit der PCI-Nummer 1 werden die CAN-Netz-Nummern 2 und 3 zugewiesen.

Tabelle A3.9.7: Beispiel für die Installation zweier CAN-PCI/331-Boards

Sie können jetzt die Kommunikation auf dem CAN-Bus mit dem Beispiel-Programm `cantest` testen.

Achtung:	Achten Sie vorher auf eine funktionsfähige Verdrahtung! Damit das Modul korrekt arbeiten kann, ist ein zweiter CAN-Teilnehmer notwendig, da sonst der CAN-Controller Fehlermeldungen erzeugt. Achten Sie auf korrekte Bus-Terminierung.

A 3.10 Update der lokalen Firmware

A 3.10.1 Übersicht

Update-Programme für die lokale Firmware sind zur Zeit **nur für Linux, Linux-RTAI, LynxOS, Solaris, VxWorks und QNX4-Betriebssysteme** implementiert.
Die Update-Funktion ist für die Module CAN-ISA/200 und CAN-PC104/200 ohne Bedeutung, da sie keinen lokalen CPU-Baustein besitzen. Bei dem Modul CAN-PCC wird die Update-Funktion ebenfalls nicht unterstützt.

Die Firmware für den lokalen Prozessor der intelligenten CAN-Boards ist in Flash-EPROM-Speicherbausteinen abgelegt. Bei Auslieferung der Karte enthalten sie die Firmware-Version, die zum Zeitpunkt der *Hardware-Herstellung* aktuell war.

Auf den mitgelieferten Disketten oder auf der CD-ROM befindet sich ein Update-Programm, mit dessen Hilfe die zum *Auslieferungstermin* gültige Software in das Flash-EPROM gebrannt werden kann.

In Abhängigkeit von dem Modul und dem Betriebssystem finden Sie entweder das Update-Programm '`canupd`' oder das Update-Programm '`upd`*`xxx`*' auf Ihrem Datenträger. Die Programme werden während der Treiberinstallation auf Ihrer Festplatte installiert.
Beide Programme sind von der Funktionalität her gleichwertig. Während `canupd` ein universelles Update-Programm für verschiedene Module ist, bietet `upd`*`xxx`* gezielte Updates für einzelne Module.

Die Kennung der Module erfolgt dabei über die Zeichenkombination beim Aufruf:

CAN-Modul	Eingabe-Syntax für `upd`*`xxx`*
CAN-ISA/331 CAN-PC104/331	`updc331i`
CAN-PCI/331 CPCI-CAN/331 PMC-CAN/331	`updc331`
CAN-PCI/360 CPCI-CAN/360	`updc360`

Tabelle 3.10.1: Eingabe-Syntax beim Update-Programm `upd`*`xxx`*

A 3.10.2 Prüfung der Firmware-Versionsnummer

Mit den Update-Programmen können Sie prüfen, welche Firmware-Version in Ihrem Flash-EPROM gespeichert ist und bei Bedarf selbständig Updates der lokalen Firmware durchführen.

Wir empfehlen, nach der Installation der Software-Treiber für die Host-CPU zu prüfen, ob die mitgelieferte Firmware aktueller ist als die im Flash-EPROM befindliche Firmware.

Der Aufruf von `canupd` (bzw. `upd`*`xxx`*) ohne Parameterangabe führt zur Ausgabe der zur Verfügung gestellen Firmware mit Versionsnummern. Da `canupd` ein universelles Programm ist, enthält es Firmware für verschiedene Hardware-Anwendungen.

Nach dem Laden des Treibers könnte bei Linux-Betriebssystemen die Firmware-Version auch in dem Verzeichnis `/var/log/messages` kontrolliert werden.

Der Aufruf von `canupd` (bzw. `upd`*`xxx`*) mit anschließender Eingabe der logischen Netz-Nummer der zu bearbeitenden Karte führt zur Ausgabe der Versionsnummer der im Flash-EPROM dieser Karte gespeicherten Firmware.

Beispiel-Eingabe: `canupd 0`

Das Programm vergleicht dabei automatisch die Firmware-Versionen und zeigt auf dem Monitor in Klartext an, ob die gespeicherte Version noch aktuell ist oder ob ein Update erforderlich ist.

A 3.10.3 Ausführen des Updates

1. Aufruf von `canupd` (bzw. `upd`*`xxx`*) mit gewünschter logischer Netz-Nummer (siehe oben).

2. Nach der Versionsüberprüfung erfolgt die Abfrage, ob das Programm mit dem Update fortfahren soll oder nicht.

   - Wird hier ein ‘y’ für JA eingegeben, so erfolgt anschließend die Neuprogrammierung des Flash-EPROMs **unabhängig davon, welche Firmware-Version aktueller ist!**

   - ‘n’ für NEIN bricht den Vorgang ab. Die Firmware im Flash-EPROM bleibt unverändert.

3. Um die neue Firmware zu aktivieren, muß der lokale Prozessor zurückgesetzt werden. Dies erfolgt durch Stoppen und Neustarten des Software-Treibers der Host-CPU.

Die folgende Tabelle zeigt die hierzu notwendigen Kommandoeingaben:

CAN-Modul	Stoppen und Neustarten des CAN-Treibers beim Betriebssystem			
	Linux	LynxOS	VxWorks	QNX4

CAN-ISA/331
CAN-PC104/331
	Stoppen:
	`rmmod c331i`
	
	Neustart:
	`insmod c331i` *)
		Stoppen:
		`instcan -u c331i`
		
		Neustart:
		`instcan c331i`
			Stoppen:
			System herunterfahren
			
			Neustart:
			`c331iStart()`
				Stoppen:
				Shell beenden mit `Ctrl-C` oder Prozeß beenden mit `kill`
				
				Neustart: `c331i &`

CAN-PCI/331
CPCI-CAN/331
PMC-CAN/331
	Stoppen:
	`rmmod c331`
	
	Neustart:
	`insmod c331` *)
		Stoppen:
		`instcan -u c331`
		
		Neustart:
		`instcan c331`
			Stoppen:
			System herunterfahren
			
			Neustart:
			`c331Start()`
				Stoppen:
				Shell beenden mit `Ctrl-C` oder Prozeß beenden mit `kill`
				
				Neustart: `c331 &`

CAN-PCI/360 CPCI-CAN/360	Stoppen: `rmmod c360` Neustart: `insmod c360` *)	-	-	-

*) Unterstützter Linux-Kernel und Standard-Konfiguration vorausgesetzt.

Tabelle A3.10.2: Kommandos zum Stoppen und Neustarten der CAN-Treiber

Stichwortverzeichnis