Achtung!

Unbedingt Installationshinweise beachten!
(Siehe Kap. 2 des Hardware-Manuals.)

Die in den Installationshinweisen vorgegebene Reihenfolge für die einzelnen Schritte der Hardware-Installation sind unbedingt zu befolgen, da sonst schwerwiegende Schäden am PC, bzw. Laptop oder am CANPCC-Modul entstehen können !

Insbesondere ist *zuerst* der Niedervoltstecker in das CAN-PCC-Modul zu stecken und erst *danach* das Modul mit dem PC/Laptop zu verbinden! Andernfalls besteht die Gefahr, daß mit den spannungsführenden Teilen des Niedervoltsteckers (+5V am mittleren Kontakt) geerdete Metallteile der Gehäuse berührt werden und so ein Kurzschluß erfolgt !

esd gmbh

CAN - PCC

CAN-Interface für eine parallele Schnittstelle

Hardware-Handbuch

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. 205
D-30165 Hannover

Telefon: 0511/37298-0
FAX: 0511/37298-198
Email: info@esd-electronics.com
Internet: http://www.esd-electronics.com

Dokument-Datei:	I:\TEXTE\DOKU\MANUALS\CAN\CENTRONI\CANPP12H.MA6
Datum des Ausdrucks:	09.07.97

Versionsbezeichnung der Platine:	CANPP-2
Versionsbezeichnung der Schaltpläne:	CANPP02, Rev. 1.1

Änderungen in den Kapiteln

Die hier aufgeführten Änderungen im Dokument betreffen sowohl Änderungen in der *Hardware* als auch reine Änderungen in der *Beschreibung* der Sachverhalte.

Kapitel	Änderungen gegenüber Handbuch-Version 1.1
-	Neues Platinenlayout.
4.2	Neue Stromlaufpläne.

Weitere technische Änderungen vorbehalten.

Achtung!

Unbedingt Installationshinweise beachten!
(Siehe Kap. 2 dieses Hardware-Manuals.)

Die in den Installationshinweisen vorgegebene Reihenfolge für die einzelnen Schritte der Hardware-Installation sind unbedingt zu befolgen, da sonst schwerwiegende Schäden am PC, bzw. Laptop oder am CANPCC-Modul entstehen können !

Insbesondere ist *zuerst* der Niedervoltstecker in das CAN-PCC-Modul zu stecken und erst *danach* das Modul mit dem PC/Laptop zu verbinden! Andernfalls besteht die Gefahr, daß mit den spannungsführenden Teilen des Niedervoltsteckers (+5V am mittleren Kontakt) geerdete Metallteile der Gehäuse berührt werden und so ein Kurzschluß erfolgt !

Inhalt

Seite

1. Übersicht

1.1 Beschreibung des Moduls

Das Modul stellt eine bidirektionale Schnittstelle zwischen CENTRONICS und CAN-Bus zur Verfügung. Es bietet daher eine unkomplizierte und preiswerte Ankopplung eines PC's, Laptops o. ä. an den CAN-Bus.

Mit Hilfe des CAN-PCC-Moduls können z.B. vom PC aus andere CAN-Module gesteuert oder überwacht werden.

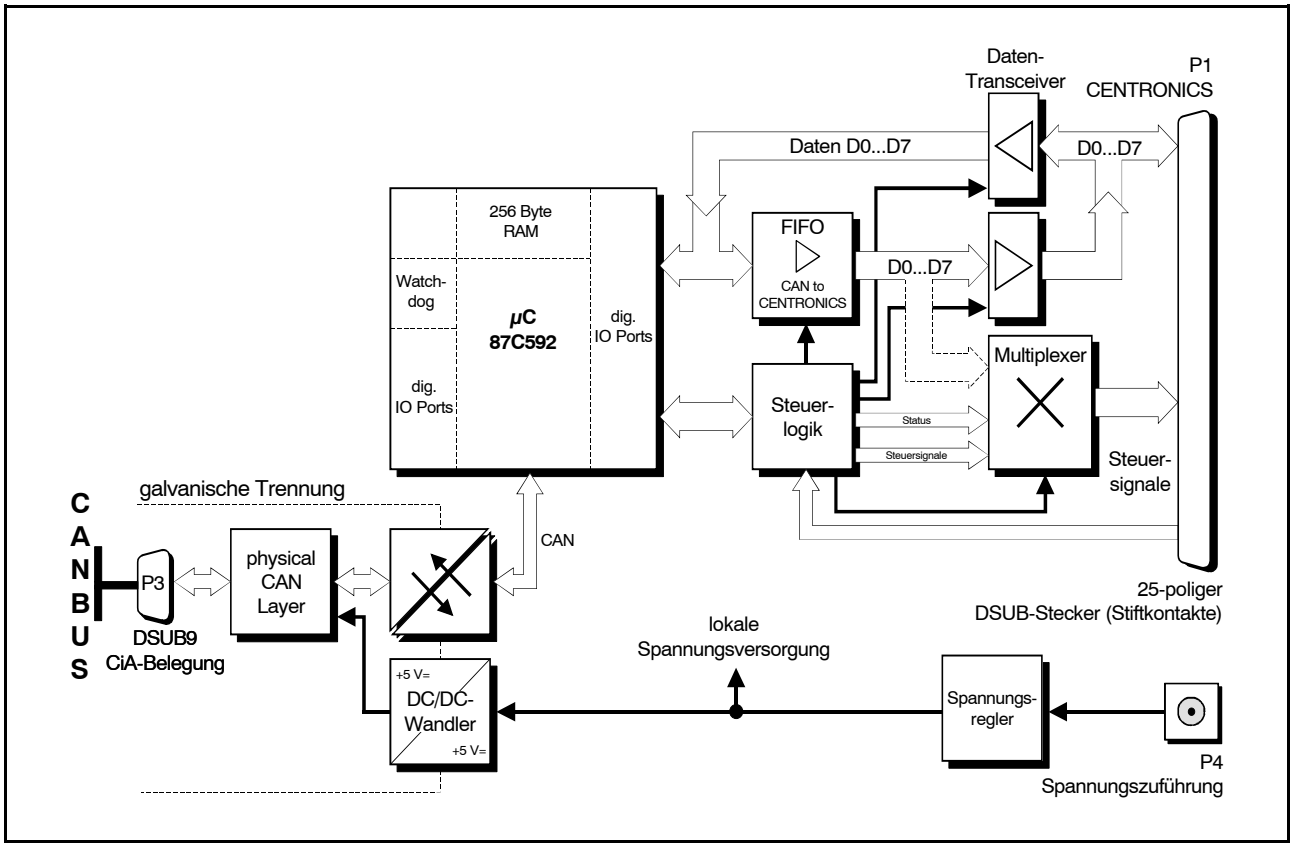


Abb. 1.1.1: Blockschaltbild des Moduls

Als CAN-Controller kommt ein 87C592 zum Einsatz, der 2048 CAN-Identifier unterstützt. Das CAN-Interface ist über Optokoppler und DC/DC-Wandler von der CENTRONICS-Schnittstelle galvanisch getrennt. Die Versorgungsspannung des Moduls muß extern zugeführt werden. Dies kann z.B. über ein Tastatur-Adapterkabel vom PC/Laptop aus oder über ein Miniatur-Steckernetzteil erfolgen. Der zulässige Eingangsspannungsbereich kann den technischen Daten auf den folgenden Seiten entnommen werden.

Übersicht

Der Datentransfer von der CENTRONICS-Schnittstelle zum CAN-Bus erfolgt Byte-weise. Die Daten, die vom CAN-Bus zur CENTRONICS-Schnittstelle übertragen werden, können ebenfalls Byte-weise übertragen werden.

Bei älteren PC-CENTRONICS-Schnittstellen ist der Datenpfad jedoch nicht immer bidirektional ausgeführt. Die mitgelieferte Software erkennt dies, und nutzt in diesem Fall die Steuerleitungen für die Datenübertragung. Hier werden jeweils vier Bits parallel übertragen.

Die vom CAN-Bus eintreffenden Daten werden in einem (mindestens) 512 Bytes umfassenden FIFO zwischengespeichert, um zu gewährleisten, daß bei größeren Datenraten keine Daten verlorengehen.

Der CENTRONICS-Anschluß erfolgt über eine 25-polige DSUB-Buchse. Zum Anschluß des CAN-Bus ist ein 9-poliger DSUB-Stecker vorgesehen.

Die komplette Kommunikations-Firmware des Moduls ist im Lieferumfang enthalten. Die Anwenderschnittstelle zwischen PC und CAN-Bus bilden Libraries, die Kommandos zur Parametrierung des Moduls und zur Steuerung des Datentransfers enthalten. Es sind Libraries für Windows (.dll), OS/2 und DOS lieferbar. Ein Installationsprogramm sorgt für die einfache Inbetriebnahme und die Anpassung an den PC.

Über verschiedene Software-Pakete (Optionen) für den PC werden die Schicht-7-Protokolle für CAL, CANopen und SDS unterstützt.

1.2 Gehäuseansicht

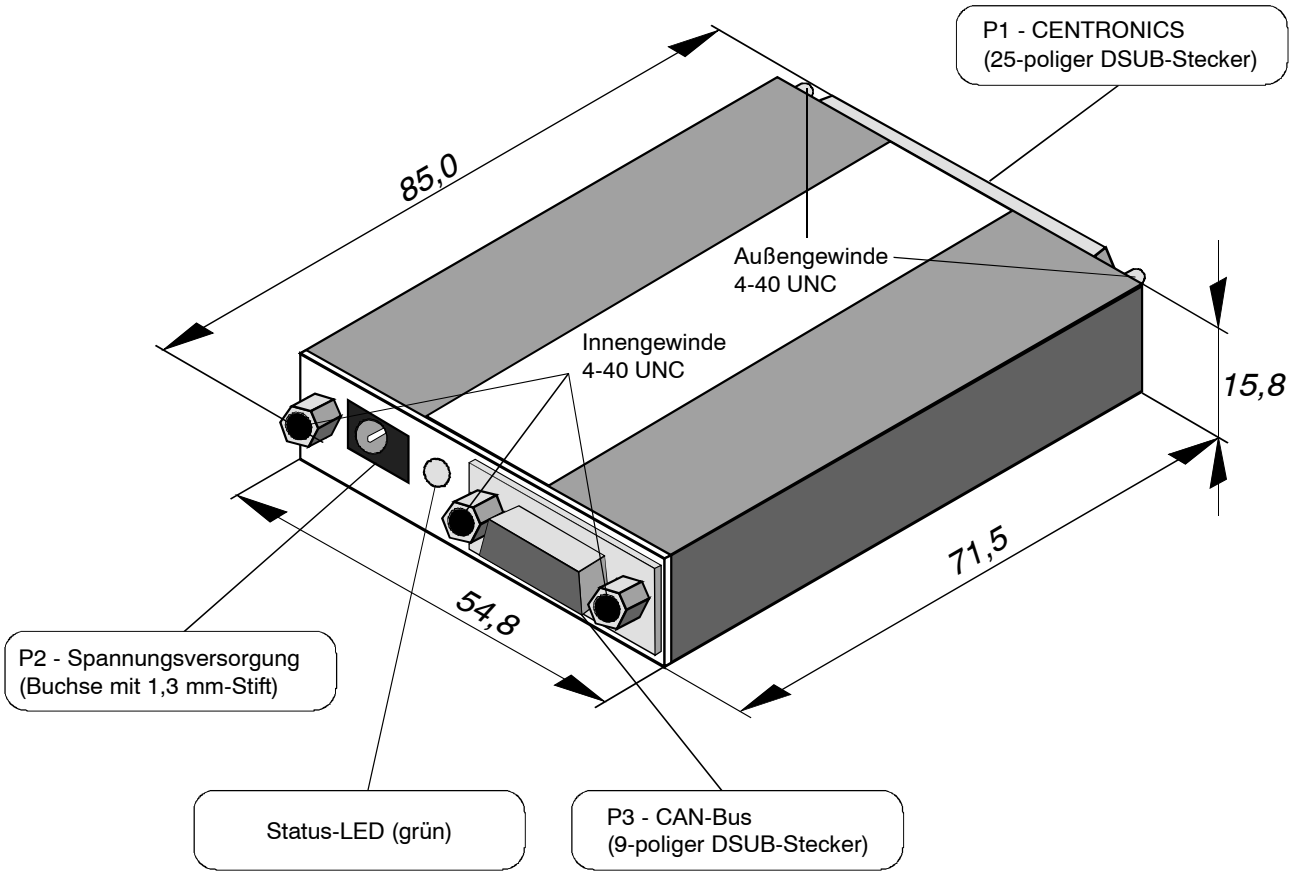


Abb. 1.2.1: Abmessungen und Steckerbezeichnung des CAN-PCC-Moduls

1.3 Allgemeine technische Daten des Koppler-Moduls

Temperaturbereich	zulässige Umgebungstemperatur: 0...50 °C
Luftfeuchtigkeit	max. 90%, nicht kondensierend
Versorgungsspannung	Eingangsspannungsbereich: 5,0 V (DC) ... 9 V (DC) zulässige Restwelligkeit: $\triangle U_{\sim} < 100$ mV (f < 100 kHz) Stromverbrauch $I_{VCC} \approx 150$ mA (typisch für $U_{VCC} = 9$ V, 20 °C)
Steckverbinder	P1 (DSUB25/Stifte) - CENTRONICS P2 (DC-Leistungsbuchse, 1.3 mm Stift) - Spannungsversorgung P3 (DSUB9/Stifte) - CAN-Bus-Anschluß
Abmessungen	85,0 mm x 54,8 mm x 15,8 mm
Gewicht	120 g

Tabelle 1.3.1: Allgemeine Daten des Moduls

1.4 Technische Daten des Miniatur-Steckernetzteils

Das Miniatur-Steckernetzteil ist nicht im Lieferumfang des Koppler-Moduls enthalten.

Temperaturbereich	zulässige Umgebungstemperatur: 0...50 °C
Luftfeuchtigkeit	max. 90%, nicht kondensierend
Netzspannung	U_{IN} = 230 V ±10% f = 45...60 Hz
Ausgangsspannung	U_{OUT} = 6 V ±10%
Steckverbinder	230 V (AC): Netzteilgehäuse 6 V (DC): Niedervoltsteckverbinder 3,5 x 1,3 mm mit 1,5 m Kabel
Abmessungen	68,5 mm x 45,0 mm 23,0 mm

Tabelle 1.4.1: Technische Daten des Miniatur-Steckernetzteils

1.5 CAN- und µController-Baugruppen

CAN-Interface	Physical Layer gemäß ISO 11898,
Übertragungsrate	programmierbar von 10 kBit/s bis 1 MBit/s
CAN-Identifier	das Modul unterstützt 2048 CAN-Identifier
µController	87C592, 16 MHz
EEPROM	zur Speicherung der CAN-Parameter vorgesehen (wird zur Zeit nicht unterstützt)
LED-Anzeige	grüne LED zur Statusanzeige des µControllers
Galvanische Trennung des CAN-Interface gegenüber µController-Baugruppen	nach VDE 0110b§8, Isolationsgruppe C und Einbau in Schaltschrank: 300 VDC / 250 VAC

Tabelle 1.5.1: Technische Daten der CAN- und µController-Baugruppen

1.6 Software-Unterstützung

Im Lieferumfang ist die komplette Firmware enthalten. Sie beinhaltet die lokale Treiber-Software, die im EPROM des CAN-Controllers abgelegt ist und eine Anwenderschnittstelle, die in Form einer Microsoft C-Library auf 3,5"-Disketten mitgeliefert wird. Die Library ist mit WINDOWS™-Betriebssystemen ab Version 3.0 nutzbar.

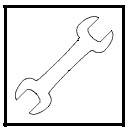
Die Software wird in einem separaten Handbuch beschrieben.

1.7 Bestellhinweise

Typ	Eigenschaften	Bestell-Nr.
CAN-PCC	CAN-CENTRONICS-Koppler incl. PC-Software	C.2422.01
CAN-PCC-ADPT-MDIN	Adapterkabel für Spannungsabgriff von Tastaturkabel mit MINI-DIN-Steckern (wird für Spannungszuführung über Laptop oder Notebook benötigt)	C.2422.12
CAN-PCC-ADPT-DIN	Adapterkabel für Spannungsabgriff von Tastaturkabel mit DIN-Steckern (wird für Spannungszuführung über PC benötigt)	C.2422.13
CAN-PCC-NT	Miniatur-Steckernetzteil mit Spannungsver-sorgungsleitung Leitung für CAN-PCC, Netzspannung 230 V (AC)/50 Hz (Alternative zu C.2422.12 oder C.2422.13)	C.2422.14
CAN-PCC-MD	Anwenderhandbuch in deutsch 1*)	C.2422.20

1*) Wird das Handbuch gemeinsam mit dem Modul bestellt, so wird es kostenlos mitgeliefert.

Tabelle 1.7.1: Bestellhinweise



2. Installationshinweise

Die in den Installationshinweisen vorgegebene Reihenfolge für die einzelnen Schritte der Hardware-Installation sind unbedingt zu befolgen, da sonst schwerwiegende Schäden am PC, bzw. Laptop oder am CANPCC-Modul entstehen können!!

Bitte verwenden Sie für die Verdrahtung nur die vorgesehenen Adapter und Leitungen.

Die Installation muß in folgenden Schritten erfolgen:

1. PC, bzw. Laptop ausschalten.

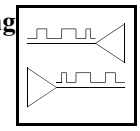
2. Spannungsversorgung an das CAN-PCC-Modul anschließen.
   Der Anschluß erfolgt entweder mit einem Adapterkabel, das an die Tastaturbuchse des PC's (9-polige DIN-Buchse) oder an die Tastaturbuchse des Laptops (5-polige-Mini-DIN-Buchse) anzuschließen ist, oder über eine Steckernetzteil.
   In jedem Fall ist zuerst der Niedervoltstecker in das CAN-PCC-Modul zu stecken und danach das Modul mit dem PC/Laptop zu verbinden! Andernfalls besteht die Gefahr, das mit den spannungsführenden Teilen des Niedervoltsteckers (+5V am mittleren Kontakt) geerdete Metallteile der Gehäuse berührt werden und so ein Kurzschluß erfolgt!

3. CAN-Netz anschließen

4. CAN-PCC an das gewünschte CENTRONICs-Port des PC's/Laptops anschließen.
   Um die CENTRONICs-Stecker sicher zu verbinden, sind die Befestigungsschrauben anzuziehen.

5. PC/Laptop einschalten
   Nach dem Einschalten muß die grüne Status-LED des CAN-PCC-Moduls blinken. Die verschiedenen Blinkzustände sind auf Seite 17 erläutert.

6. Jetzt kann mit der Software-Installation fortgefahren werden.
   Die Software-Installation wird im Software-Handbuch beschrieben.



3. Beschreibung der Baugruppen

3.1 CAN-Bus-Interface

3.1.1 Bitrate

Die Bitrate läßt sich im Bereich von 10 kBit/s bis 1,0 MBit/s programmieren. Nach einem RESET muß der Controller initialisiert werden, bevor er auf dem CAN-Bus Daten empfangen oder senden kann. Weitere Informationen hierzu finden sich im Software-Handbuch des Moduls.

3.1.2 Sende-und Empfangsschaltung des CAN-Interface (Physical Layer)

Das physikalische Interface des esd-CAN-Bus entspricht der Norm ISO 11898. Der Anschluß an die Busleitung erfolgt über einen 9-poligen DSUB-Stecker, der als Stiftkontakt (P3) ausgeführt ist.

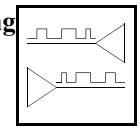
Als CAN-Bus-Transceiver werden auf dem Modul z.B. der Si9200 oder der 82C250 eingesetzt.

Die galvanische Trennung der Signale zum CAN-Bus erfolgt über Optokoppler.

3.2 CENTRONICS-Interface

Das CENTRONICS-Interface ist auf einen 25-poligen DSUB-Stecker mit Stiftkontakten geführt. Die Daten- und Steuersignale werden von CMOS-Logikbausteinen (z.B. 74HC244) getrieben und empfangen. Die Datensignale und die Steuersignale 'AUTOFEED', 'RESET' und 'SELECT_IN' sind mit 4,7 kΩ-Pull-Up-Widerständen an +5 V versehen. Das Steuersignal 'STROBE' ist mit einem 1 kΩ-Pull-Up-Widerstand an +5 V versehen.

Daten-Bytes, die vom CAN-Bus kommend an die CENTRONICS-Schnittstelle übertragen werden sollen, können in einem (mindestens) 512 Bytes umfassenden FIFO zwischengespeichert werden. Daten-Bytes, die von der CENTRONICS-Schnittstelle kommend auf dem CAN-Bus gesendet werden sollen, werden direkt vom CAN-Controller 87C592 empfangen und verarbeitet.

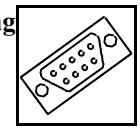


3.3 Status-LED

Die Status-LED nimmt in Abhängigkeit vom aktuellen Modul-Status verschiedene Blinkzustände an. Die unten aufgeführte Tabelle zeigt die Bedeutung der verschiedenen Leuchtzustände der LED.

Zustand der Status-LED	Status des CAN-Moduls
LED blinkt: 250 ms grün, 50 ms aus	Zustand des Moduls Ok, Initialisierung des Controllers mit Bitrate ist erfolgt.
LED blinkt: 40 ms grün, 40 ms aus	Aktueller CAN-Bus-Fehler.
LED blinkt: 50 ms grün, 250 ms aus	Zustand nach dem Einschalten oder RESET (Bitrate noch nicht gesetzt) oder Modul hat einen vorübergehenden CAN-Bus-Fehler festgestellt, der jetzt jedoch nicht mehr aktiv ist.
LED konstant aus	Spannungsversorgung unterbrochen oder Spannungsversorgung OK, aber alle anderen Funktionen des Moduls außer Betrieb -‣ CAN-Controller arbeitet nicht oder ein interner Fehler ist aufgetreten. (Der Anwender sollte sich zur Behebung dieses Fehlers mit dem Lieferanten in Verbindung setzen.)

Tabelle 3.3.1: Anzeige der Status-LED



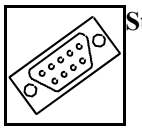
4. Anhang

4.1 Steckerbelegungen

4.1.1 CENTRONICS-Stecker P1

Signal	Pin		Signal
STROBE*	1	14	AUTOFEED
D0	2	15	ERROR
D1	3	16	RESET*
D2	4	17	SELEC_IN*
D3	5	18	GND
D4	6	19	GND
D5	7	20	GND
D6	8	21	GND
D7	9	22	GND
ACK_IRQ*	10	23	GND
BUSY	11	24	GND
PE	12	25	GND
SLCT	13		

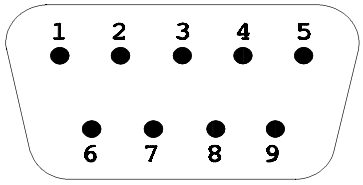
25-poliger DSUB-Stecker mit Stiftkontakten



Steckerbelegung

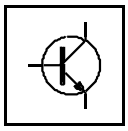
4.1.2 CAN-Bus-Stecker P3 (9-pol DSUB Stiftkontakte)

Pin-Zuordnung



Pin-Belegung:

Signal	Pin		Signal
		1	reserved
CAN_GND	6		
		2	CAN_L
CAN_H	7		
		3	CAN_GND
reserved	8		
		4	reserved
reserved	9		
		5	reserved



4.2 Stromlaufpläne

CAN-PCC

Windows 3.11 API

Software-Handbuch

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. 205
D-30165 Hannover

Telefon: 0511/37298-0
FAX: 0511/37298-198
Email: info@esd-electronics.com
Internet: http://www.esd-electronics.com

Handbuch-Datei:	I:\TEXTE\DOKU\MANUALS\CAN\PCC\CANPP15S.MA6
Datum des Ausdrucks:	13.11.97

Beschriebene Software:	CAN-PCC Software
Revision/Datum:	Rev. 1.0 / 05.97

Änderungen in der Software und/oder der Dokumentation

Änderung in diesem Handbuch gegenüber der Version 1.3	Änderung in der Software	Änderung in der Dokumentation
Hinweise zur Parametrierung der parallelen Schnittstelle des PC's eingefügt.	-	x
-	-	-

Inhalt

Seite

1. Übersicht

1.1 Zur Software-Dokumentation

Dieses Handbuch beschreibt die **Windows 3.11**-Software der CAN-PCC. Nach dieser Übersicht und den anschließenden Installationshinweisen werden die Funktionen erläutert, die der Anwender einsetzen kann. Die Software benötigt weitere Funktionen, die jedoch nicht für den Aufruf durch den Anwender bestimmt sind. Sie werden zur Information trotzdem kurz aufgeführt, da sie auch in der Header-Datei erscheinen.

Vereinbarungen
Die Abkürzung **PC** steht in diesem Handbuch nicht nur für Personal Computer. Hier sind, sofern nicht ausdrücklich unterschieden wird, auch andere kompatible Geräte, an die das Koppelmodul angeschlossen werden kann, gemeint (also z.B. Laptops oder Notebooks).

1.2 Lieferumfang

Die komplette Kommunikations-Firmware des Moduls ist im Lieferumfang enthalten. Die Anwenderschnittstelle zwischen PC und CAN-Bus bilden Libraries, die Kommandos zur Parametrierung des Moduls und zur Steuerung des Datentransfers enthalten. Ein Installationsprogramm sorgt für die einfache Inbetriebnahme und die Anpassung an den PC.

Die Software wird auf einer 3,5"-Diskette geliefert. Auf der Diskette sind zwei übergeordnete Verzeichnisse angelegt:

- Das Verzeichnis `CPCCFINST` enthält das Installationsprogramme für die CENTRONICS-Schnittstelle.
- Das Verzeichnis `CANCPCC.DLL` enthält Dateien für den Einsatz des Moduls mit einer DLL für Windows 3.1.

Das Unterverzeichnis `CANCPCC.DLL\SAMPLES` enthält Beispiele im Quell-Code für die schnelle Einarbeitung in die DLL.

CPCCINST

`CPCCINST.EXE`	CENTRONICS-Installationsprogramm, erzeugt die Datei **`CPCC.INI`**
`CPCC .INI`	Konfigurationsdatei für Loader.

CPCC.DLL

`CAN_AC2 .H`	Include-File mit funktionellen Prototypen und Struktur-Definitionen
`CAN_AC2 .DLL`	Dynamische Link-Bibliothek für Windows 3.1
`BAC2_TST.EXE`	Ausführbares Sample-Programm
`CPCC .H`	Wie `CAN_AC2.DLL`, aber geordnet
`CPCC .INI`	Zur Information
`INFO .TXT`	Installationshinweise
`NOTE .ESD`	Hinweise zum Setzen der Baudrate
`README .TXT`	Hinweise zur Diskette

\SAMPLES

`AC2 .H`	Include-Datei mit funktionellen Prototypen und Struktur-Definitionen
`BAC2_TST.C`	Quell-Datei des Haupt-Moduls des Testprogramms
`BINI_DLG.C`	Quell-Datei der Dialogbox-Routinen für die Initialisierung
`BAC2_DLG.C`	Quell-Datei der Dialogbox-Routinen für das Senden und Empfangen
`BAC2_TST.RES`	Dialog-resource-Datei
`BAC2_DLG.H`	Dialog-Definitionen
`BAC2_TST.H`	Interne Definitionen
`AC2_DEFS.H`	Schlüsselwort-Definitionen
`RESOURCE.H`	Dialog-resource-include-Datei
`BAC2_TST.DEF`	Windows-Definitionsdatei
`CAN2_TST.ICO`	Icon
`BAC2_TST.MAK`	Microsoft Visual C++ makefile

2. Konfiguration der parallelen Schnittstelle

Die Einstellung der parallelen Schnittstelle des PC's muß überprüft und gegebenenfalls angepaßt werden. Die Vorgehensweise zur Konfiguration der Schnittstelle ist abhängig vom PC und dem eingesetzten BIOS. Die folgende Beschreibung hat daher nur beispielhaften Charakter:

1. Beim Hochlauf des PC's den BIOS-Setup aufrufen.

2. Unter dem Auswahlpunkt 'Chip Set Features' kontrollieren, daß der 'Parallel Port Mode' wie folgt eingestellt ist, und Einstellung ggf. anpassen:

   Zulässige Einstellungen:
   1. 'Normal' oder 'SPP' (Standard Parallel Port)
   2. **'EPP' (Enhanced Parallel Port)**
      Die Einstellung 'EPP' ist der Einstellung 'Normal' oder 'SPP' vorzuziehen.

   Unzulässige Einstellungen:
   1. 'ECP' (Extended Capability Port)
   2. 'ECP+EPP'

3. Setup mit Speicherung der Parameter beenden

Bei parallelen Schnittstellen, die sich nicht auf dem Motherboard, sondern auf einer zusätzliche I/O-Platine befinden, kann die Konfiguration in der Regel über Steckbrücken erfolgen. Bitte nehmen Sie hierzu das Handbuch der Schnittstellenkarte zur Hilfe.

3. Software-Installation

In diesem Kapitel wird die Installation der Software auf dem PC beschrieben. Die Installation der Hardware des Moduls ist im Hardware-Handbuch beschrieben.

Vorgehensweise:

1. Entfernen Sie alle Programme und Treiber, die auf die von Ihnen gewählte CENTRONICs-Schnittstelle zugreifen können.

2. Erstellen Sie (manuell) ein Verzeichnis auf Ihrer Festplatte für die CAN-PCC-Software und kopieren Sie die gewünschten Dateien (DLL) von der Diskette in dieses Verzeichnis.

3. Kopieren Sie die Bibliotheken in das entsprechende Verzeichnis Ihres Compilers oder stellen Sie das Bibliotheks-Verzeichnis des Compilers auf das von Ihnen erstellte Verzeichnis ein.

4. Kopieren Sie das Verzeichnis CPCCINST ebenfalls in das Compiler-Verzeichnis (bereits vorhandene Dateien werden überschrieben).

5. Der Aufruf des Programms CPCCINST.EXE führt zur Überprüfung der Hardware. Das Programm ist selbsterklärend, folgen Sie den angezeigten Anweisungen.

4. Betriebsarten der CAN-PCC-Bibliothek

Der eingesetzte CAN-Controller unterstützt nur 11-Bit-Identifier. Parameter, die die Eingabe von 29-Bit-Identifiern erfordern, sind für zukünftige Anwendungen vorgesehen.

Für das Zwischenspeichern von CAN-Nachrichten sind prinzipiell zwei Modes möglich: Der FIFO-Mode und der CAN-Object-Mode. **In dieser Software-Version ist nur der FIFO-Mode implementiert.**

4.1 FIFO-Mode

Nachrichten, die empfangen oder gesendet werden sollen, werden sequentiell, nach dem Prinzip eines FIFOs (First-in-first-out) zwischen dem CAN-Bus und dem PC ausgetauscht. Die Nachricht, die zuerst eintrifft, wird auch zuerst weitergeleitet. Dies gilt sowohl für Status-Nachrichten als auch für Kommandos.

4.2 Object-Mode

Der Object-Mode ist in dieser Software-Version noch nicht implementiert !
Sollten Sie diesen Mode benötigen, fragen Sie die Verfügbarkeit bitte an.

5. Funktionen der CAN-PCC-Bibliothek

5.1 Ausführbare Funktionen

5.1.1 CANPC_reset_chip

Dieses Kommando beendet eine evtl. laufende CAN-Übertragung, setzt das CENTRONICS-Port zurück und liest die Datei CPCC.INI aus dem Programmverzeichnis.

`CANPC_reset_chip(void)`

Eingabeparameter: -

Rückgabeparameter:

0...	RESET war erfolgreich
-4...	Timeout-Fehler auf dem CAN-PCC-Modul beim Zugriff auf den CAN-Controller

5.1.2 CANPC_initialize_chip

Diese Funktion definiert das Bit-Timing des CAN-Controllers. Die Parameter *presc*, *sjw*, *tseg1* und *tseg2* werden in das Bit-Timing-Register des CAN-Controllers übertragen. Sie geben die Werte vor, die das Bit-Timing des CAN-Protokolls beschreiben.

Die Baudrate wird nach der folgenden Formel errechnet, wobei jedoch noch weitere, hier nicht mit aufgeführte Randbedingungen beachtet werden müssen (siehe Datenblatt des CAN-Controllers 87C592)

$$\text{Baudrate} = \frac{f_{crystal}}{2 * presc * (1 + tseg1 + tseg2)}$$

Taktfrequenz $f_{crystal}$ =16 MHz

Beispiele für Baudrateneinstellung:

Bit-Rate [kBit/s]	Index	Controller-Register		Prescaler	Sync Jump With	Time Segment 1	Time Segment 2	Sample Mode
		BTR0 [HEX]	BTR1 [HEX]	*presc*	*sjw*	*tseg1*	*tseg2*	*SAM*
1000	0	00	14	1	1	5	2	0
666.6	1	00	18	1	1	9	2	0
500	2	00	1C	1	1	13	2	0
333.3	3	01	18	2	1	9	2	0
250	4	01	1C	2	1	13	2	0
166	5	02	1C	3	1	13	2	0
125	6	03	1C	4	1	13	2	0
100	7	43	1C	4	2	16	3	0
66.6	8	45	2F	6	2	16	3	0
50	9	47	1C	8	2	16	3	0
33.3	10	4B	2F	12	2	16	3	0
20	11	53	1C	20	2	16	3	0
12.5	12	5F	2F	32	2	16	3	0
10	13	67	1C	40	2	16	3	0

Tabelle 5.1.1: Parameter für die Einstellung der Baudrate

```
int CANPC_initialize_chip(  int presc,
                            int sjw,
                            int tseg1,
                            int tseg2,
                            int sam)
```

Eingabeparameter:	*presc*...	CAN-Prescaler	[1..32]	
	sjw...	CAN-Synchronisation-Jump-Width	[1..4]	
	tseg1...	CAN-Time-Segment-1	[1..16]	
	tseg2...	CAN-Time-Segment-2	[1..8]	
	sam...	CAN-Sample-Mode	0...	1 Sample
			1...	3 Samples

Rückgabeparameter:

0...	Initialisierung war erfolgreich
-1...	Parameter-Fehler (falsche Wertangabe)
-4...	Timeout-Fehler auf dem CAN-PCC-Modul beim Zugriff auf den CAN-Controller

5.1.3 CANPCC_inibit

Diese Funktion definiert wie `CANPC_initialize_chip` das Bit-Timing des CAN-Controllers. Hier wird die Baudrate jedoch durch die Übergabe eines Index-Wertes von 0...15 oder durch die Übergabe der Parameter ***btr0*** und ***brt1*** eingestellt.

Erst über diese Funktion wird auf dem Modul eine Baudrate eingestellt. Vorher arbeitet der Controller nicht auf dem CAN-Bus!

Die angegebenen typischen Leitungslängen basieren auf Erfahrungswerten aus der Praxis. Die minimal erreichbaren Leitungslängen ergeben sich aus den 'worst case'-Verzögerungszeiten der eingesetzten Bauteile.

index	87C592-Register		Bit-Rate [kBit/s]	typische Werte der erreichbaren Leitungslänge l_{max} [m]	minimal erreichbare Leitungslänge l_{min} [m]
	btr0 [HEX]	***btr1*** [HEX]			
0	00	14	1000	37	20
1	00	18	666.6	80	65
2	00	1C	500	130	110
3	01	18	333.3	180	160
4	01	1C	250	270	250
5	02	1C	166	420	400
6	03	1C	125	570	550
7	04	1C	100	710	700
8	45	2F	66.6	1000	980
9	09	1C	50	1400	1400
10	4B	2F	33.3	2000	2000
11	18	1C	20	3600	3600
12	5F	2F	12.5	5400	5400
13	31	1C	10	7300	7300
14	00	16	800	59	42

Die Angaben in der Tabelle basieren auf den Grenzwerten des Bit-Timings des CAN-Protokolls, den Laufzeiten des lokalen CAN-Interface und den Laufzeiten des Kabels. Die Laufzeit des Kabels ist mit ca. 5,5 ns/m angenommen. In den Angaben sind weitere Einflüsse z.B. durch fehlende Abschlußwiderstände, den spezifischen Widerstand, die Geometrie des Kabels oder äußere Störeinflüsse bei der Übertragung nicht miteinbezogen!

Tabelle 5.1.2: Parameter ***index***, ***btr0*** und ***btr1*** zur die Einstellung der Baudrate

int CANESDinibit(*rate*)

Eingabeparameter: ***rate***... Der Parameter ***rate*** wird je nach eingegebenem Wertebereich verschieden ausgewertet. Er steht entweder für den Parameter ***index*** oder für die beiden Parameter ***btr0*** und ***btr1***. Die Bedeutung dieser Parameter ist in der oben aufgeführten Tabelle gezeigt.

[$0000...$000E]...	***index*** (siehe Tabelle oben)
[$0010...$FFFF]...	Auswertung in der Form [xxyy] mit xx = ***btr0*** (siehe Tabelle oben) yy = ***btr1*** (siehe Tabelle oben)

Rückgabeparameter:

0...	Initialisierung war erfolgreich
-1...	Parameter-Fehler (falsche Wertangabe)
-4...	Timeout-Fehler auf dem CAN-PCC-Modul beim Zugriff auf den CAN-Controller

5.1.4 CANPC_set_acceptance

Mit Hilfe dieses Kommando kann ein Empfangsfilter für CAN-Daten-Frames und Remote-Request-Frames konstruiert werden. Das Filter läßt sich für Standard- oder Extended-CAN-Identifier erzeugen. Es werden nur die Frames empfangen, deren Identifier-Bits dem Acceptance-Filter entsprechen.

Die Werte der Bits, die im Parameter *AccMask*... (Acceptance-Mask) mit einer '1' initialisiert wurden, müssen mit den Werten der zugehörigen Bits im Parameter *AccCode*... (Acceptance-Code) übereinstimmen. Eine '0' im Parameter Acceptance-Mask bedeutet, daß dieses Bit nicht überprüft wird.

Da das Modul keine 29-Bit-Identifier unterstützt, erfolgt auch keine Überprüfung der Identifier. Die Parameter *AccCodeXtd* und *AccMaskXtd* müssen aber trotzdem übergeben werden.

Das Funktion `CANPC_set_acceptance` kann parallel zu der Funktion `CANESD_enable_id` eingesetzt werden. Beide Funktionen sind gleichwertig und können auch alternativ verwendet werden.

```
int CANPC_set_acceptance(  unsigned int  AccCodeStd,
                           unsigned int  AccMaskStd,
                           unsigned long AccCodeXtd,
                           unsigned long AccMaskXtd )
```

Eingabeparameter:

AccCodeStd...	Acceptance-Code Standard [$0000...$07FF]
AccMaskStd...	Acceptance-Mask Standard [$0000...$07FF]
AccCodeXtd...	Acceptance-Code Extended [$00000000...$1FFFFFFF]
AccMaskXtd...	Acceptance-Mask Extended [$00000000...$1FFFFFFF]

Rückgabeparameter:

0...	Kommando erfolgreich ausgeführt
-4...	Timeout-Fehler auf dem CAN-PCC-Modul beim Zugriff auf den CAN-Controller

Beispiel: Der Funktionsaufruf mit den folgenden Parametern würde alle Identifier zulassen:

```
CANPC_set_acceptance( 0, 0, 0, 0)
```

5.1.5 CANPCC_enable_id und CANPCC_disable_id

Mit Hilfe dieses Kommandos kann ein Empfangs-Filter für einzelne oder mehrere Daten- oder Remote-Frames aufgebaut, bzw. zurückgenommen werden. Durch mehrmaligen Aufruf können außerdem verschiedene Identifier-Gruppen selektiert werden.
Die Auswahl erfolgt jeweils durch die Eingabe des ersten und des letzten Identifiers, der empfangen werden soll. Für die Selektion einzelner Identifier muß hier zweimal der selbe Wert eingetragen werden.
Das Funktion `CANPCC_enable_id` kann parallel zu der Funktion `CANPC_set_acceptance` eingesetzt werden.

```
int CANPCC_enable_id(   int idfirst,
                        int idlast)

int CANPCC_disable_id(  int idfirst,
                        int idlast)
```

Eingabeparameter:
- *idfirst*... erster zugelassener Rx-identifier [$0000...$07FF]
- *idlast*... letzter zugelassener Rx-identifier [$0000...$07FF]

Rückgabeparameter:
- 0... Kommando erfolgreich ausgeführt
- -4... Timeout-Fehler auf dem CAN-PCC-Modul beim Zugriff auf den CAN-Controller

5.1.6 CANPC_enable_fifo_transmit_ack

Diese Funktion erteilt die Freigabe für Bestätigungsmeldungen an die Applikation über Frames, die im FIFO-Mode gesendet wurden. Die Bestätigung kann mit dem Kommando `CANPC_ read_ac` gelesen werden. Wurde diese Funktion nicht aufgerufen, so wird auch keine Meldungen erzeugt.

```
int CANPC_enable_fifo_transmit_ack(void)
```

Eingabeparameter: -

Rückgabeparameter:
- 0... Kommando erfolgreich ausgeführt
- -4... Timeout-Fehler auf dem CAN-PCC-Modul beim Zugriff auf den CAN-Controller

5.1.7 CANPC_send_data

Dieses Kommando sendet ein Daten-Frame mit den angegebenen Parametern auf den CAN-Bus.

```
int CANPC_send_data(   unsigned long  Ident,
                       int            Xtd,
                       int            DataLength,
                       byte           *pData)
```

Eingabeparameter:

Ident... Identifier [$0000...$07FF, $00000000...$1FFFFFFF]

Xtd... Identifier-Länge
- 0: Standard-Identifier
- 1: Extended-Identifier (wird nicht unterstützt)

DataLength... Anzahl der Daten-Bytes, die gesendet werden sollen.[0...8]

pData... Zeiger auf das Adreßfeld, in dem die zu sendenden Daten abgelegt sind.

Rückgabeparameter:

- 0... Kommando erfolgreich ausgeführt
- -1... Transmit-Busy bei CAN-Error und Ack-Modus, d.h. der Sendevorgang war noch nicht beendet, als im Acknowledge-Modus ein CAN-Bus-Fehler auftrat
- -4... Timeout-Fehler auf dem CAN-PCC-Modul beim Zugriff auf den CAN-Controller

5.1.8 CANPC_send_remote

Dieses Kommando sendet ein Remote-Frame mit den angegebenen Parametern auf den CAN-Bus. Das gesendete Remote-Frame hat immer die Länge 0. Trotzdem wird im Parameter ***DataLength*** die Anzahl der gesendeten Daten-Bytes (hier eben 0) mit übertragen.

```
int CANPC_send_remote(  unsigned long  Ident,
                        int            Xtd,
                        int            DataLength)
```

Eingabeparameter:

Ident...	Identifier [$0000...$07FF, $00000000...$1FFFFFFF]		
Xtd...	Identifier-Länge	0:	Standard-Identifier
		1:	Extended-Identifier (wird nicht unterstützt)
DataLength...	Anzahl der Daten-Bytes, die angefragt werden. [0...8]		

Rückgabeparameter:

0...	Kommando erfolgreich ausgeführt.
-1...	Transmit-Busy bei CAN-Error und Ack-Modus, d.h. der Sendevorgang war noch nicht beendet, als im Acknowledge-Modus ein CAN-Bus-Fehler auftrat
-4...	Timeout-Fehler auf dem CAN-PCC-Modul beim Zugriff auf den CAN-Controller

5.1.9 CANPC_read_ac

Über diese Funktion wird die Applikation über die Sendung und den Empfang von Nachrichten, sowie verschiedene Fehlerbedingungen informiert. Die Funktion muß per Polling abgefragt werden.

Die Rückgabe-Codes 1...12 (RC1 bis RC12) definieren, welche Elemente der zurückgegebenen Struktur relevant sind.

`int CANPC_read_ac( param_struct*write_ac_param )`

Elemente der Struktur *`param_struct`*:

`unsigned long` ***`Ident`***...	Identifier eines empfangenen oder gesendeten Frames (RC1, RC2, RC3, RC8, RC9, RC10, RC11, RC12).
`int` ***`DataLength`***...	Anzahl der empfangenen (RC1, RC9) oder gesendeten (RC3, RC10) Daten-Bytes.
`int` ***`RecOverrun_flag`***...	Die letzten empfangenen Daten wurden nicht vom PC gelesen und wurden von den neuen Daten überschrieben (RC1, RC2, RC9, RC12). Nur in dem Mode Object-Buffer.
`int` ***`RCV_fifo_lost_msg`***...	Wird beim Einsatz des PCC-Kopplermoduls nicht verwendet.
`byte` ***`RCV_data[8]`***...	Empfangene Daten-Bytes (RC1, RC9).
`int` ***`AckOverrun_Flag`***...	Die Bestätigung des letzten gesendeten Frames ist von der Applikation bisher nicht gelesen worden (RC3, RC10). Nur im Object-Buffer-Mode.
`int` ***`XMT_ack_fifo_lost_acks`***...	Wird beim Einsatz des PCC-Kopplermoduls nicht verwendet.
`int` ***`XMT_rmt_fifo_lost_remotes`***...	Wird beim Einsatz des PCC-Kopplermoduls nicht verwendet.
`int` ***`Bus_state`***...	CAN-Bus-Status (RC5): `0`... Fehler liegt vor `1`... kein Fehler `2`... Bus nicht angeschlossen

`int` ***`Error_state`***... Weitere Fehlerursachen (RC7):
- `0`... kein Fehler
- `8`... Hardware-FIFO inkonsistent (-> Überlauf oder Verbindung gestört)

`int` ***`CAN`***... wird nicht gesetzt

`unsigned long` ***`Time`***... Zeitpunkt der angezeigten Events mit einer Auflösung von 1 µs. Der Zähler wird mit CANPC_start_chip zurückgesetzt. (RC1, RC2, RC9, RC12 und RC3, RC5, RC8, RC10, RC11 im FIFO-Mode)

Rückgabeparameter (Rückgabe-Codes - RC's) des Kommandos:

`0`... Keine neuer Event auf dem PCC-Kopplermodul.

`1`... Standard-Daten-Frame empfangen

`2`... Standard-Remote-Frame empfangen

`3`... Sendung eines Standard-Daten-Frames ist bestätigt.

`4`... Nicht implementiert.

`5`... Änderung des Bus-Status.

`6`... Nicht implementiert.

`7`... Eine weitere Fehlerbedingung liegt an.

`8`... Sendung eines Standard-Remote-Frames ist bestätigt.

Die Rückgabewerte 9...12 werden zur Zeit (05/97) nicht unterstützt.
`9`... Extended-Daten-Frame empfangen.
`10`... Sendung eines Extended-Daten-Frames ist bestätigt.
`11`... Sendung eines Extended-Remote-Frames ist bestätigt.
`12`... Extended-Remote-Frame empfangen.

`-4`... Timeout-Fehler auf dem CAN-PCC-Modul beim Zugriff auf den CAN-Controller.

5.2 Weitere Funktionen

Die folgenden Funktionen sind aus Gründen der Kompatibilität zu früheren Software-Versionen implementiert. Obwohl sie in der Header-Datei aufgeführt sind, hat der Anwender keinen Anspruch auf diese Funktionen und sollte sie nicht einsetzen. Der Aufruf führt zu keiner Fehlermeldung:

INIPC_initialize_board
CANPC_reset_board
CANPC_set_mode
CANPC_set_output
CANPC_enable_fifo
CANPC_enable_timestamps
CANPC_optimize_rcv_speed
CANPC_start_chip

Außerdem werden einige Funktionen mit der Endung ...2 aufgeführt. Für diese gilt das gleiche wie oben erwänt.

Die folgenden Funktionen sind ebenfalls in der Header-Datei aufgeführt. Sie sind zur Zeit jedoch nicht implementiert und antworten beim Aufruf stets mit einem Fehlercode:

CANPC_reinitialize
CANPC_enable_dyn_obj_buf
CANPC_initialize_interface
CANPC_define_object
CANPC_send_remote_object
CANPC_supply_object_data
CANPC_supply_rcv_object_data
CANPC_send_object
CANPC_write_object
CANPC_read_rcv_data
CANPC_read_xmt_data

5.3 Die Kommandosequenz

5.3.1 Zeitlicher Ablauf der Kommandos

Die Kommandos der CAN-PCC-Library müssen in einem bestimmten zeitlichen Ablauf aufgerufen werden (Kommando-Sequenz):

1. Nach dem Programmstart muß der CAN-Controller-Baustein zurückgesetzt werden, indem `CANPC_reset_chip` aufgerufen wird.

2. Über den Aufruf des Kommandos `CANPC_inibit` oder `CANPC-initialize_chip` wird der Controller-Baustein initialisiert (Baudrate).

3. Die Auswahl der oder des Identifiers kann alternativ über `CANPC_set_acceptance` oder `CANPC_enable_id` erfolgen.

4. Als Option kann über `CANPC_enable_fifo_transmit_ack` die Bestätigung für das erfolgte Senden von Nachrichten gewählt werden.

5. Jetzt kann das Senden und Empfangen der CAN-Messages erfolgen.

5.3.2 Grafische Darstellung der Kommandosequenz