Attention!

Strictly follow the installation notes!
(See chapt. 2 of the hardware manual).

The order of the individual steps of the hardware installation as specified in the installation notes must be followed at any rate, because otherwise serious damage can be caused to PC or laptop, or to the CAN-PCC module!

Especially the low-voltage connector has to be connected to the CAN-PCC module *first* and only *after that* the module is to be connected to the PC/laptop! Otherwise live parts of the low-voltage connector (+5V between the middle contact) could come into contact with earthed metal parts of the cases and cause a short circuit!

esd gmbh

CAN - PCC

CAN Interface For One Parallel Interface

Hardware Manual

N O T E

The information in this document has been carefully checked and is believed to be entirely reliable. **esd** makes no warranty of any kind with regard to the material in this document, and assumes no responsibility for any errors that may appear in this document. **esd** reserves the right to make changes without notice to this, or any of its products, to improve reliability, performance or design.

esd assumes no responsibility for the use of any circuitry other than circuitry which is part of a product of **esd** gmbh.

esd does not convey to the purchaser of the product described herein any license under the patent rights of **esd** gmbh nor the rights of others.

esd electronic system design gmbh
Vahrenwalder Str. 205
D-30165 Hannover
Germany

Phone: +49-511-37298-0
Fax: +49-511-37298-198
Email: info@esd-electronics.com
Internet: http://www.esd-electronics.com

Manual File:	I:\TEXTE\DOKU\MANUALS\CAN\CENTRONI\CANPP12H.EN6
Date of Print:	12.03.1998

Version Designation of Board:	CANPP-2
Version Designation of Circuit Diagrams:	CANPP02, Rev. 1.1

Changes in the chapters

The changes in the manual listed below affect changes in the *hardware*, as well as changes in the *description* of the facts only.

Chapter	Alterations versus manual version 1.1
-	First English issue (manual).

Technical details are subject to change without notice.

Attention!

Strictly follow the installation notes!
(See chapt. 2 of the hardware manual).

The order of the individual steps of the hardware installation as specified in the installation notes must be followed at any rate, because otherwise serious damage can be caused to PC or laptop, or to the CAN-PCC module!

Especially the low-voltage connector has to be connected to the CAN-PCC module *first* and only *after that* the module is to be connected to the PC/laptop! Otherwise live parts of the low-voltage connector (+5V between the middle contact) could come into contact with earthed metal parts of the cases and cause a short circuit!

Contents

Page

1. Overview

1.1 Module Description

The module offers a bidirectional interface between CENTRONICS and CAN, and therefore an easy and inexpensive coupling of a PC, laptop, or something similar, to the CAN.

By means of the CAN-PCC module other CAN modules can be controlled or monitored for instance by a PC.

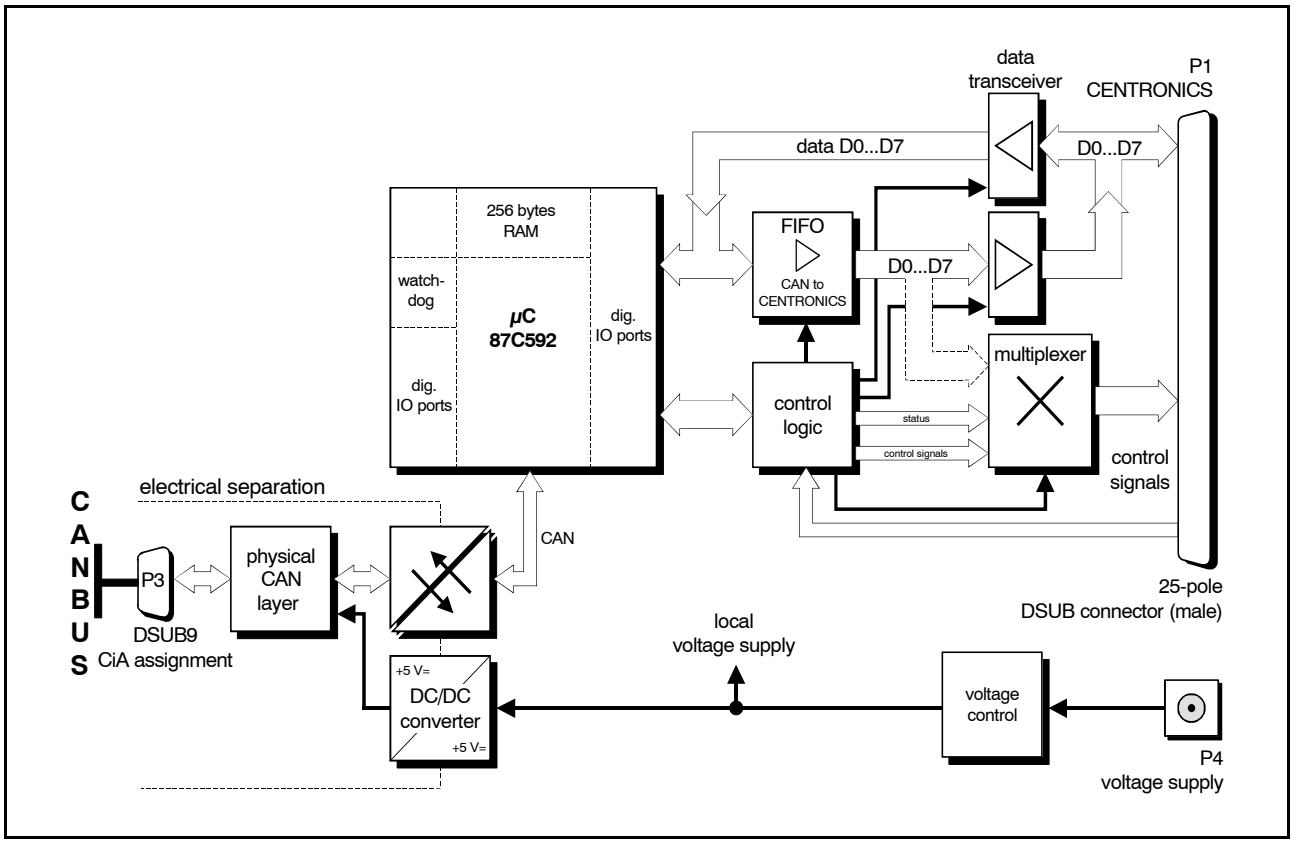


Fig. 1.1.1: Block circuit of the module

The CAN controller is a 87C592, which supports 2048 CAN identifiers. The CAN interface is electrically separated from the CENTRONICS interface by optocouplers and DC/DC converters. The voltage supply of the module has to be feeded externally. This can for instance be made via a keyboard adaptor cable from the PC/laptop or via a miniature connector-power pack. The permissible input-voltage range can be taken from the technical data on the following pages.

Overview

Data is transferred from the CENTRONICS interface to the CAN byte by byte. Data which is transferred vice versa (CAN to CENTRONICS) can also be transferred byte by byte.

Older PC-CENTRONICS interfaces, however, do not always have a bidirectional data path. The software contained in the product package recognizes this and then uses the control lines for the data transfer. In this case four bits each are transferred parallel.

The data transmitted by the CAN is buffered in a (minimum) 512 bytes-sized FIFO so that no data is lost at larger data rates.

The CENTRONICS is connected by means of a 25-pole male DSUB connector. The CAN is connected via a 9-pole male DSUB connector.

The complete communication firmware of the module is contained in the product package. The user interface between PC and CAN is made of libraries which contain commands to parameterize the module and control the data transfer. Libraries for Windows (.dll), OS/2 and DOS are available. An installation program makes it easy to take the module into operation and adapt it to the PC.

The layer-7 protocols for CAL, CANopen and SDS are supported by means of various software packages (optional) for the PC.

1.2 Case View

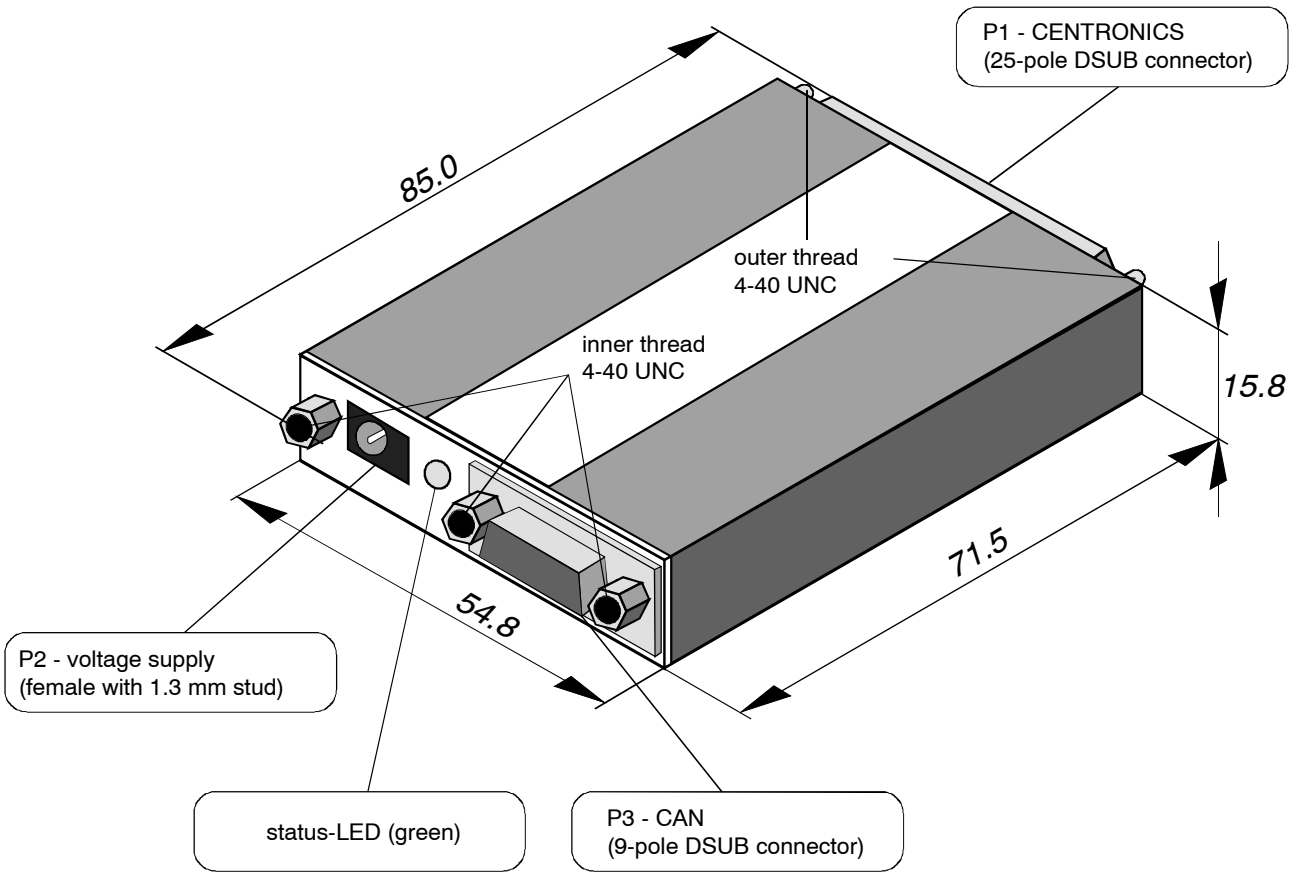


Fig. 1.2.1: Dimensions and connector designations of the CAN-PCC module

1.3 General Technical Data of the Interface Module

Temperature range	permissible ambient temperature: 0...50 °C
Humidity	max. 90%, non-condensing
Supply voltage	input-voltage range: 5.0 V (DC) ... 9 V (DC) permissible residual ripple: $\triangle U_{\sim} < 100$ mV (f < 100 kHz) power consumption $I_{VCC} \approx 150$ mA (typical for $U_{VCC} = 9$ V, 20 °C)
Connectors	P1 (DSUB25/male) - CENTRONICS P2 (DC-power socket, 1.3 mm stud) - voltage supply P3 (DSUB9/male) - CAN connection
Dimensions	85.0 mm x 54.8 mm x 15.8 mm
Weight	120 g

Table 1.3.1: General data of the module

1.4 Technical Data of the Miniature Connector-Power Pack

The miniature connector-power pack is not contained in the product package.

Temperature range	permissible ambient temperature: 0...50 °C
Humidity	max. 90%, non-condensing
Mains voltage	U_{IN} = 230 V ±10% f = 45...60 Hz
Output voltage	U_{OUT} = 6 V ±10%
Connectors	230 V (AC): power-pack case 6 V (DC): low-voltage connector 3.5 x 1.3 mm with 1.5 m cable
Dimensions	68.5 mm x 45.0 mm x 23.0 mm

Table 1.4.1: Technical data of the miniature connector-power pack

1.5 CAN- and µController Units

CAN interface	physical layer in accordance with ISO 11898,
Transmission rate	programmable from 10 kbit/s to 1 Mbit/s
CAN identifiers	the module supports 2048 CAN identifiers
µcontroller	87C592, 16 Mhz
EEPROM	memory for CAN parameters (not yet supported)
LED display	green LED for status display of the µcontroller
Electrical separation of the CAN interface from the µcontroller units	under German VDE 0110b§8 regulation, isolation group C and installation into cubicle: 300 VDC / 250 VAC

Table 1.5.1: Technical data of the CAN- and µcontroller units

1.6 Software Support

The complete firmware is contained in the product package. The firmware contains the local driver software which is stored in the EPROM of the CAN controller and a user interface, which can be found on 3.5" disks as a Microsoft C-Library. This library can be used with WINDOWS™-operating sytems from version 3.0.

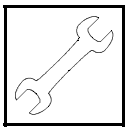
The software will be described in a separate manual.

1.7 Order Information

Type	Properties	Order no.
CAN-PCC	CAN-CENTRONICS interface, incl. PC software	C.2422.01
CAN-PCC-ADPT-MDIN	Adaptor cable for voltage tapping from keyboard cable via MINI-DIN connectors (is needed for voltage supply via laptop or notebook)	C.2422.12
CAN-PCC-ADPT-DIN	Adaptor cable for voltage tapping from keyboard cable via DIN connectors (is needed for voltage supply via PC)	C.2422.13
CAN-PCC-NT	Miniature connector-power pack with voltage supply line for CAN-PCC, mains voltage 230 V (AC)/50 Hz (alternatively for C.2422.12 or C.2422.13)	C.2422.14
CAN-PCC-ME	English manual 1*)	C.2422.21

1*) If manual and module are ordered together, the manual is included in the module price.

Table 1.7.1: Order Information



2. Installation Notes

The order for the individual steps of the hardware installation must strictly be followed, because otherwise serious damage can be caused at PC or laptop or at the CAN-PCC module!!

For wiring please use only the adaptors and lines provided.

Keep to the following steps to install the module:

1. Switch off the PC or laptop.

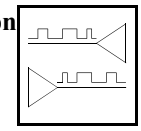
2. Connect voltage supply to the CAN-PCC module.
   The voltage supply is either connected via an adaptor cable which has to be connected to the keyboard socket of the PC (9-pole DIN socket) or the keyboard socket of the laptop (5-pole mini-DIN socket), or via a connector-power pack.
   The low-voltage connector must always be plugged into the CAN-PCC module first and then the module has to be connected to the PC/laptop! Otherwise voltage carrying parts of the low-voltage connector (+5V in the middle contact) could come into contact with earthed metal parts of the cases, which would lead to a short circuit!

3. Connect CAN net.

4. Connect CAN-PCC to the desired CENTRONICS port of the PC/laptop.
   In order to connect the CENTRONICS connectors securely, the fastening screws have to be tightened.

5. Switch on PC/laptop.
   After the PC/laptop has been switched on, the green status LED of the CAN-PCC module has to flash. The various flash status are described on page 17.

6. Now the software installation can be started.
   The software installation will be described in the software manual.



3. Unit Description

3.1 CAN-Interface

3.1.1 Bitrate

The bitrate can be programmed in ranges from 10 kbit/s to 1.0 Mbit/s. After a RESET the controller has to be initialized before it can receive or transmit data on the CAN. Further information on this can be found in the software manual of the module.

3.1.2 Transmit- and Receive Circuit of the CAN Interface (Physical Layer)

The physical interface of the esd-CAN is in accordance with the ISO 11898 norm. The interface is connected to the busline via a male 9-pole DSUB connector (P3).

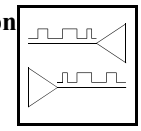
The Si9200 or the 82C250 are used as CAN transceivers on the module.

The signals are electrically separated from the CAN by optocouplers.

3.2 CENTRONICS Interface

The CENTRONICS interface can be connected via a male 25-pole DSUB connector. The data and control signals are transmitted and received by CMOS logic components (e.g. HC244). The data signals and the control signals 'AUTOFEED', 'RESET' and 'SELECT_IN' have 4.7 kΩ-pull-up resistors at +5 V. The control signal 'STROBE' has a 1 kΩ-pull-up resistor at +5 V.

Data bytes coming from the CAN to be transmitted to the CENTRONICS interface can be buffered in a (minimum) 512 bytes-sized FIFO.
Data bytes coming from the CENTRONICS interface to be transmitted to the CAN are directly received and handled by the CAN controller 87C592.

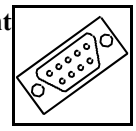


3.3 Status LED

Depending on the current module status the status LED changes into various flash status. These are listed in the table below.

Status of status LED	Status of CAN module
LED flashes: 250 ms green,	Module status OK, initialisation of the controller with bitrate has occurred.
LED flashes: 40 ms green,	Current CAN error.
LED flashes: 50 ms green,	Status after switching-on or RESET (bitrate has not yet been set) or module has detected a temporary CAN error which is not active anymore,
LED constantly off	Voltage supply is interrupted or voltage supply is OK, but all other functions of the module are out of order -▸ CAN controller does not work or an internal error has occurred. (For the removal of this error the user should contact the dealer.)

Table 3.3.1: Display of the status LED



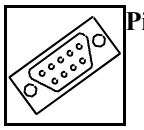
4. Appendix

4.1 Connector Assignments

4.1.1 CENTRONICS Connector 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		

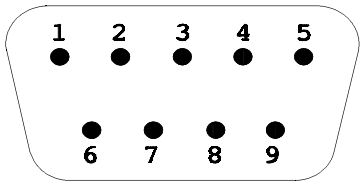
25-pole male DSUB connector



Pin Assignment

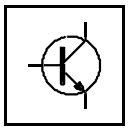
4.1.2 CAN Connector P3 (9-pole male DSUB connector)

Pin Orientation:



Pin Assignment:

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



4.2 Circuit Diagrams

CAN-PCC

Windows 3.11 API

Software Manual

N O T E

The information in this document has been carefully checked and is believed to be entirely reliable. **esd** makes no warranty of any kind with regard to the material in this document, and assumes no responsibility for any errors that may appear in this document. **esd** reserves the right to make changes without notice to this, or any of its products, to improve reliability, performance or design.

esd assumes no responsibility for the use of any circuitry other than circuitry which is part of a product of **esd** gmbh.

esd does not convey to the purchaser of the product described herein any license under the patent rights of **esd** gmbh nor the rights of others.

esd electronic system design gmbh
Vahrenwalder Str. 205
D-30165 Hannover
Germany

Tel: +49-511-372-980
Fax: +49-511-633-650
Email: pm@esd.h.eunet.de

Manual File:	I:\TEXTE\DOKU\MANUALS\CAN\CENTRONI\CANPP14S.EN6
Date of Print:	12.03.98

Described Software:	CAN-PCC software
Revision/Date:	Rev. 1.0 / 05.97

Changes in the software and/or documentation

Alterations in this manual versus version 1.3	Alterations in the software	Alterations in the documentation
First English issue (manual).	-	-
-	-	-

Contents

Page

1. Overview

1.1 About the Software Documentaion

This manual describes the **Windows 3.11** software of the CAN-PCC. Following this overview and the installation notes, the functions will be explained which can be used by the user. The software requires further functions which are not to be used by the user, however. Still, they will be explained shortly, because they also appear in the header file.

Agreements
The abbrevation **PC** does not only mean personal computer in this manual. If not clearly differentiated, other compatible devices which can be connected to the interface module are also called PC (for instance laptops or notebooks).

1.2 Product Package

The complete communication firmware of the module is contained in the product package. The user interface between PC and CAN is made by libraries which contain commands to parameterize the module and control the data transfer. An installation program makes it easy to put the module into operation and to adapt it to the PC.

The software is contained on a 3.5" disk with two root directories:

- The directory `CPCCFINST` contains the installation program fro the CENTRONICS interface.
- The directory `CANCPCC.DLL` contains data for the use of the module with a DLL for Windows 3.1.

The subdirectory `CANCPCC.DLL\SAMPLES` contains examples in source code for quickly getting used to DLL.

CPCCINST

CPCCINST.EXE	CENTRONICS installation program, generates CPCC.INI file
CPCC .INI	Configuration file for loader.

CPCC.DLL

CAN_AC2 .H	Include file with functional prototypes and structure definitions
CAN_AC2 .DLL	Dynamical link libraries for Windows 3.1
BAC2_TST.EXE	Executable sample program
CPCC .H	Like CAN_AC2.DLL, but ordered
CPCC .INI	For information
INFO .TXT	Installation notes
NOTE .ESD	Notes on setting the bitrate
README .TXT	Notes on the disk

\SAMPLES

AC2 .H	Include file with functional prototypes and structure definitions
BAC2_TST.C	Source file of the main module of the test program
BINI_DLG.C	Source file of the dialogue-box routines for the initialisation
BAC2_DLG.C	Source file of the dialogue-box routines for transmitting and receiving
BAC2_TST.RES	Dialogue-resource file
BAC2_DLG.H	Dialogue definitions
BAC2_TST.H	Internal definitions
AC2_DEFS.H	Keyword definitions
RESOURCE.H	Dialogue-resource-include file
BAC2_TST.DEF	Windows-definition file
CAN2_TST.ICO	Icon
BAC2_TST.MAK	Microsoft Visual C++ makefile

2. Configuration of the Parallel Interface

It is important to check and possibly adapt the setting of the parallel interface of the PC. The configuration of the interface depends on the PC and the BIOS used. The following description is therefore only exemplatory:

1. Call the BIOS setup when booting the PC.

2. Check under 'Chip Set Features' that the 'Parallel Port Mode' is set as follows and adjust the setting if necessary:

   Permissible settings:
   1. ‘Normal’ or ‘SPP’ (Standard Parallel Port)
   2. **‘EPP’ (Enhanced Parallel Port)**
      The setting ‘EPP’ is better than ‘Normal’ or ‘SPP’.

   Non-permissible settings:
   1. ‘ECP’ (Extended Capability Port)
   2. ‘ECP+EPP’

3. Finish setup by storing parameters

Parallel interfaces which are not on the motherboard, but on an additional I/O board can normally be configurated by jumpers. In this case, please refer to the manual of the interface board.

3. Software Installation

This chapter describes the software installation on the PC. For hardware installation of the module refer to the hardware manual.

Installation:

1. Remove all programs and drivers which can access the CENTRONICS interface you have selected.

2. Generate (manually) a directory on your harddisk for the CAN-PCC software and copy the desired files (DLL) from the disk into your new directory.

3. Copy the libraries into the according directory of your compiler or adjust the library directory of your compiler to the directory you have generated.

4. Copy the directory CPCCINST also into the compiler directory (files already existing are overwritten).

5. By calling the program CPCCINST.EXE the hardware is checked. The program is self-explaining, just follow the instructions shown on your screen.

4. Operating Modes of the CAN-PCC Library

The CAN controller used supports only 11-bit identifiers. Parameters which require 29-bit identifiers are for future applications.

Principally two modes are possible for buffering CAN messages: The FIFO mode and the CAN-object mode. **In this software version, however, only the FIFO mode is implemented.**

4.1 FIFO Mode

Messages which have been received or which are to be transmitted are sequentially exchanged between the CAN and the PC, like a FIFO (first-in-first-out). The message received first is also passed on first. This applies for status messages as well as for commands.

4.2 Object Mode

The object mode has not been implemented in this software version!
If you need this mode, please enquire about its availability.

5. Functions of the CAN-PCC Library

5.1 Executable Functions

5.1.1 CANPC_reset_chip

By means of this command a possibly running CAN transmission is terminated, the CENTRONICS port is reset and the CPCC.INI is read out of the program.

`CANPC_reset_chip(void)`

Input parameter: -

Output parameter:

`0`...	RESET was successful
`-4`...	Timeout error on the CAN-PCC module when accessing the CAN controller

5.1.2 CANPC_initialize_chip

By means of this function the bit timing of the CAN controller is defined. The parameters ***presc***, ***sjw***, ***tseg1*** and ***tseg2*** are transferred into the bit-timing register of the CAN controller. The values of these parameters describe the bit timing of the CAN protocol.

The bitrate is calculated with the following formula, however, further marginal conditions have to be considered (for this refer to the data sheet of CAN controller 87C592)

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

Clock frequency $f_{crystal}$ =16 MHz

Examples for bitrate setting:

Bitrate [kbit/s]	Index	Controller register		Prescaler *presc*	Sync jump with *sjw*	Time segment 1 *tseg1*	Time segment 2 *tseg2*	Sample mode *SAM*
		BTR0 [HEX]	BTR1 [HEX]					
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

Table 5.1.1: Parameters for setting the bitrate

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

Input parameters:	*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

Output parameters:

- `0`... Initialisation has been successful
- `-1`... Parameter error (wrong value entry)
- `-4`... Timeout error on the CAN-PCC module when accessing the CAN controller

5.1.3 CANPCC_inibit

Like `CANPC_initialize_chip` this function defines the bit timing of the CAN controller. Here, however, the bitrate is set by transmitting an index value of 0...15 or by transmitting the parameters *`btr0`* and *`brt1`*.

Only this function sets a bitrate on the module. Before that the controller does not operate on the CAN!

The typical line lengths specified below are based on facts from experience. The minimum reachable line lengths result from the 'worst case' delay times of the components used.

index	87C592-register		Bitrate [kbit/s]	typical values of reachable line lengths l_{max} [m]	minimum reachable line lengthn 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

The specifications in the table are based on the limit values of the bit timing of the CAN protocol, the delay times of the local CAN interface and the delay times of the cable. The delay time of the cable is assumed at about 5.5 ns/m. Further influences, as for instance caused by missing terminators, the specific resistance, the cable geometry or external disturbances during transmission, have not been considered in the specifications!

Table 5.1.2: Parameters *`index`*, *`btr0`* and *`btr1`* for setting the bitrate

int CANESDinibit(*rate*)

Input parameters: *rate*... Depending on the value range specified, the parameter *rate* is evaluated in different ways. It stands either for the parameter *index* or for the two parameters *btr0* and *btr1*. The meaning of these parameters is shown in the table above.

[$0000...$000E]...	*index* (see previous table)
[$0010...$FFFF]...	evaluation in the form [xxyy] with xx = *btr0* (see previous table) yy = *btr1* (see previous table)

Output parameters:

0...	Initialisation has been successful
-1...	Parameter error (wrong value entry)
-4...	Timeout error on the CAN-PCC module when accessing the CAN controller

5.1.4 CANPC_set_acceptance

By means of this command a reception filter for CAN-data frames and remote-request frames can be made. This filter can be generated for standard or extended-CAN identifiers. Only those frames are received whose identifier bits are compliant to the acceptance filter.

The values of the bits which have been initialized with a '1' in the parameter *AccMask*... (acceptance mask), have to comply to the values of the according bits in the parameter *AccCode*... (acceptance code). A '0' in the parameter acceptance mask means that this bit is not checked.

Because the module does not support 29-bit identifiers, the identifiers are not checked. The parameters *AccCodeXtd* and *AccMaskXtd* have to be transmitted anyhow.

The function CANPC_set_acceptance can be used parallel to the function CANESD_enable_id. Both functions are equal and can also be used alternatively.

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

Input parameters:

AccCodeStd...	acceptance-code standard [$0000...$07FF]
AccMaskStd...	acceptance-mask standard [$0000...$07FF]
AccCodeXtd...	acceptance-code extended [$00000000...$1FFFFFFF]
AccMaskXtd...	acceptance-mask extended [$00000000...$1FFFFFFF]

Output parameters:

0...	Command successfully executed
-4...	Timeout error on the CAN-PCC module when accessing the CAN controller

Example: The function call with the following parameters would allow all identifiers:

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

5.1.5 CANPCC_enable_id and CANPCC_disable_id

By means of this command a receive filter for individual or various data or remote frames can be generated or taken back. By calling this function several times it is furthermore possible to select various identifier groups.
The selection is made by entering the first and the last identifier which is to be received. In order to select individual identifiers the same value has to be entered twice.
The function `CANPCC_enable_id` can be used parallel to the function `CANPC_set_acceptance.`

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

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

Input parameters:
- *idfirst*... first permitted Rx-identifier [$0000...$07FF]
- *idlast*... last permitted Rx-identifier [$0000...$07FF]

Output parameter:
- `0`... Command successfully executed
- `-4`... Timeout error on the CAN-PCC module when accessing the CAN controller

5.1.6 CANPC_enable_fifo_transmit_ack

By means of this function acknowledge messages to the application by frames which have been transmitted in FIFO mode are enabled. The acknowledgement can be read by the command `CANPC_read_ac`. If this function had not been called, no messages are generated.

```
int CANPC_enable_fifo_transmit_ack(void)
```

Input parameters: -

Output parameters:
- `0`... Command successfully executed
- `-4`... Timeout error on the CAN-PCC module when accessing the CAN controller

5.1.7 CANPC_send_data

By means of this command a data frame with the specified parameters is transmitted to the CAN.

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

Input parameters:

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

Xtd... Identifier length
- 0: Standard identifier
- 1: Extended identifier (is not being supported)

DataLength... Number of data bytes to be transmitted.[0...8]

pData... Pointer to the address field in which the data to be transmitted are stored.

Output parameters:

- 0... Command successfully executed
- -1... Transmit-busy at CAN-error and Ack-mode, that means that the transmission had not been finished when a CAN error occurred in the acknowledge mode.
- -4... Timeout error on the CAN-PCC module when accessing the CAN controller

5.1.8 CANPC_send_remote

By means of this command a remote frame with the specified parameters is transmitted to the CAN. The transmitted remote frame has always the length '0'. The number of transmitted data bytes (here 0) is transmitted nevertheless in the parameter *DataLength*.

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

Input parameters:

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

Xtd... Identifier length
- 0: Standard identifier
- 1: Extended identifier (is not being supported)

DataLength... Number of data bytes which are requested. [0...8]

Output parameters:

0... Command successfully executed

-1... Transmit-busy at CAN-error and Ack-mode, that means that the transmission had not been finished when a CAN error occurred in the acknowledge mode.

-4... Timeout error on the CAN-PCC module when accessing the CAN controller

5.1.9 CANPC_read_ac

By means of this function the application is informed about the transmission and the reception of messages and various error status. This function has be requested by polling.

The output codes 1...12 (RC1 to RC12) define the relevant elements of the returned structured.

`int CANPC_read_ac( param_struct*write_ac_param )`

Elements of the structure *param_struct*:

`unsigned long` *Ident*...	Identifier of a received or transmitted frame (RC1, RC2, RC3, RC8, RC9, RC10, RC11, RC12).
`int` *DataLength*...	Number of received (RC1, RC9) or transmitted (RC3, RC10) data bytes.
`int` *RecOverrun_flag*...	The data received last has not been read by the PC and has been overwritten by new data (RC1, RC2, RC9, RC12). Only in object-buffer mode.
`int` *RCV_fifo_lost_msg*...	Does not apply when using the PCC interface.
`byte` *RCV_data[8]*...	Received data bytes (RC1, RC9).
`int` *AckOverrun_Flag*...	The application has not yet read the acknowledgement of the frames transmitted last (RC3, RC10). Only in object-buffer mode.
`int` *XMT_ack_fifo_lost_acks*...	Does not apply when using the PCC interface.
`int` *XMT_rmt_fifo_lost_remotes*...	Does not apply when using the PCC interface.
`int` *Bus_state*...	CAN status (RC5): `0`... error `1`... no error `2`... bus is not connected

`int` ***`Error_state`***...	Further error causes (RC7): `0`... no error `8`... hardware FIFO is inconsistent (-> overflow or connection is disturbed)
`int` ***`CAN`***...	is not set
`unsigned long` ***`Time`***...	Time of the displayed events with a resolution of 1 µs. The counter is reset by CANPC_start_chip. (RC1, RC2, RC9, RC12 and RC3, RC5, RC8, RC10, RC11 in FIFO mode)

Output parameters (output codes - RCs) of the command:

`0`... No new event on the PCC interface.

`1`... Standard-data frame has been received.

`2`... Standard-remote frame has been received.

`3`... Transmission of a standard-data frame has been acknowledged.

`4`... Not implemented.

`5`... Bus status has changed.

`6`... Not implemented.

`7`... A further error status applies.

`8`... Transmission of a standard-remote frame has been acknowledged.

The output values 9...12 are not yet (05/97) being supported.

`9`... Extended-data frame has been received.
`10`... Transmission of an extended-data frame has been acknowledged.
`11`... Transmission of an extended-remote frame has been acknowledged.
`12`... Extended-remote frame has been received.

`-4`... Timeout error on the CAN-PCC module when accessing the CAN controller.

5.2 Further Functions

The following functions have been implemented for reasons of compatibility to earlier software versions. Even though the functions are listed in the header file, the user has no right to these functions and should not use them. Calling them does not trigger an error message:

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

Furthermore some functions are listed with the ending ...2. For these the same applies as mentioned above.

The following functions are also listed in the header file. They are not yet implemented, however, and always respond with an error code when called:

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 The Command Sequence

5.3.1 Chronologial Order of Commands

The commands of the CAN-PCC library have to be called in a fixed chronological order (command sequence):

1. After starting the program, the CAN-controller component has to be reset by calling `CANPC_reset_chip`.

2. By calling the command `CANPC_inibit` or `CANPC-initialize_chip` the controller component is initialized (bitrate).

3. The identifier(s) can be selected alternatively by means of `CANPC_set_acceptance` or `CANPC_enable_id`.

4. Optionally the acknowledgement for the transmission of messages can be selected by means of `CANPC_enable_fifo_transmit_ack.`

5. Now CAN messages can be transmitted or received.

5.3.2 Graphical Display of the Command Sequence