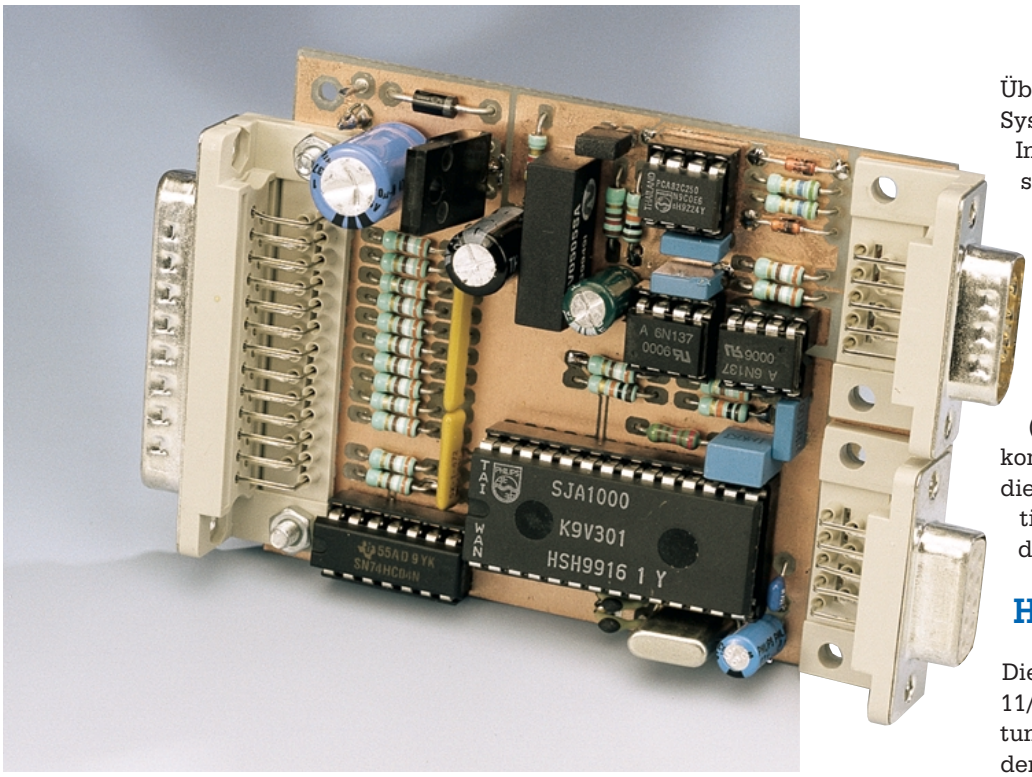
CAN-Bus-Interface für PC

Mit neuer Software

Entwurf von Reiner Lock

Die Möglichkeiten eines CAN-Bus-Systems steigen mit der Zahl unterschiedlicher Geräte, die am Bus betrieben werden können. Mit dem hier vorgestellten Interface kann jeder PC an einen CAN-Bus angeschlossen werden.



Um einen PC für den Anschluß eines CAN-Busses vorzubereiten, soll eine der vorhandenen Schnittstellen genutzt werden. Dadurch erübrigt sich eine spezielle I/O-Einsteckkarte, so dass die Schaltung auch an Notebooks arbeiten kann. Obwohl durchaus praktikabel, haben wir uns gegen die serielle Schnittstelle entschieden, denn nur die Parallelschnittstelle erlaubt es, die hohe Übertragungsrate eines CAN-Bus-Systems auszunutzen. Da über das Interface nicht nur Daten gesendet, sondern auch empfangen werden sollen, muß die Parallelschnittstelle für bidirektionalen Datenverkehr eingestellt sein. Dies geschieht bei Einsteckkarten über Jumper auf der Platine oder bei modernen Pentium-Boards über das BIOS-Setup (EPP-Modus). Auch Uralt-PCs kommen zu neuen Ehren, wenn diese mit kostengünstigen bidirektionalen Schnittstellenkarten aus dem Handel nachgerüstet werden.

Hardware

Die Schaltung ähnelt der in Elektor 11/99 vorgestellten Interface-Schaltung für Mikrocontroller wie ein Ei dem anderen, so dass die Beschreibung der Hardware und ihres Aufbaus entsprechend knapp ausfallen kann. Der Hauptunterschied zwischen beiden Schaltungen liegt natürlich in der Software.
Als CAN-Bus-Controller arbeitet wiederum ein SJA1000 (wer noch einen

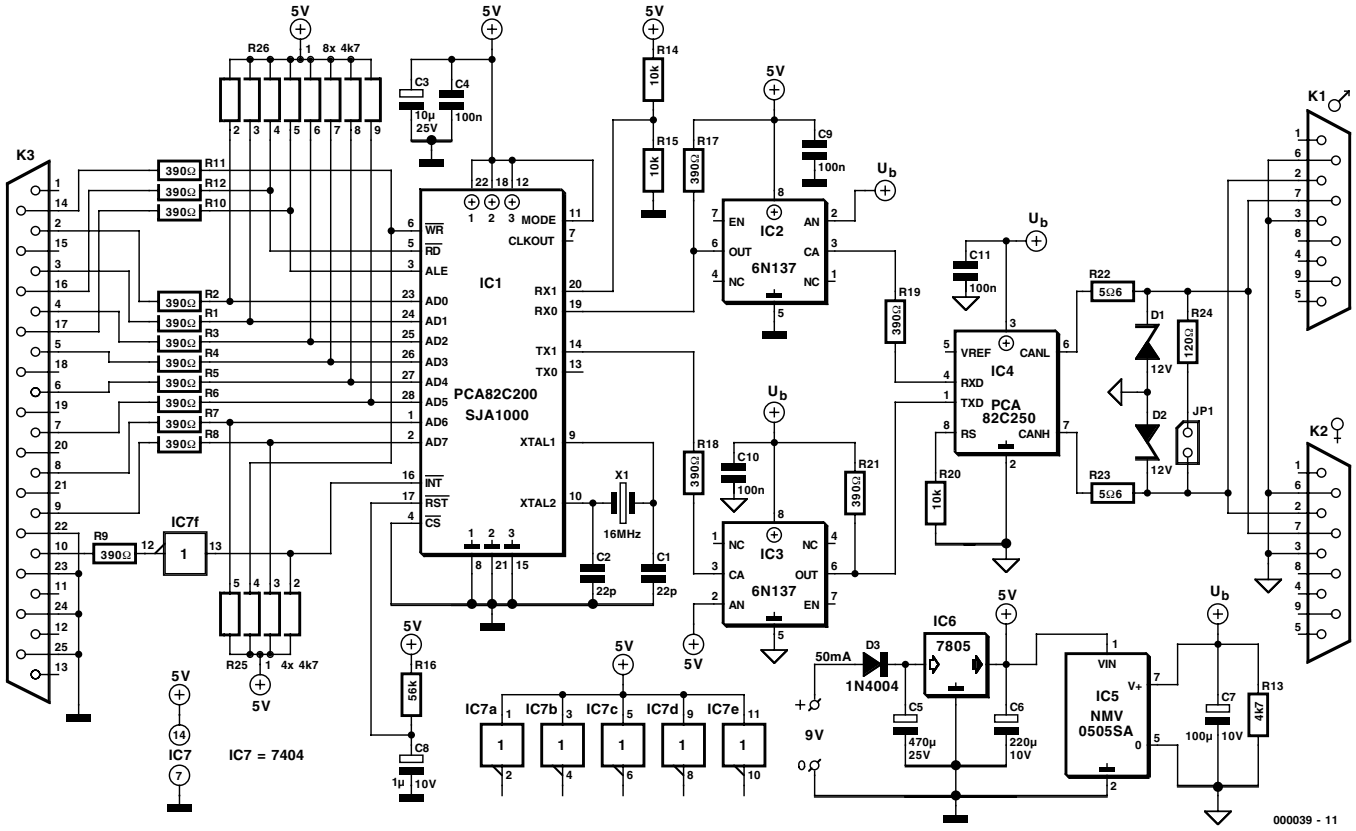


Bild 1. Die Schaltung des CAN-Bus-Interface zur parallelen PC-Schnittstelle.

PCA82C200 in Reserve hat, kann auch diesen Controller verwenden), der im Novemberheft ausführlich beschrieben wurde. Das vollständige Datenblatt kann man von *www.semiconductors.philips.com* herunterladen. Der Spannungsteiler R14/R15 ist eigentlich nur beim PCA82C200 nötig, schadet dem SJA1000 aber nicht. Besser ist es aber, R14 wegzulassen und R15 durch eine Drahtbrücke zu ersetzen, so dass der RX1-Anschluß fest auf Masse liegt. Die Software muss den internen Komparator CBP aus- und damit den internen Schmitt-Trigger des SJA1000 einschalten. Dazu schreibt man in das Register OCR den Wert $1A_{HEX}$ und setzt Bit 6 im Register CDR.

Der SJA1000 wurde mit seinen Schreib- und Leseleitungen für den direkten Anschluß an Mikrocontrollersysteme entwickelt. Die Schreib- und Leseleitungen müssen deshalb entsprechend dem Chip-Timing per Software bedient werden. Das Datenregister der parallelen Schnittstelle (Pin 2...9 von K3) wird direkt mit dem gemultiplexten Daten/Adreßbus des CAN-Controllers verbunden. Der Lese/Schreibbefehl und ALE gelangen über drei Centronics-Steuerleitungen (Autofeed, Init Printer, Select Input) zum Controller, während von den Statusregistern nur Acknowledge (Pin 10) als Interruptquelle für den SJA1000 genutzt wird.

Durch die galvanische Trennung des CAN-Busses wird die Einkopplung von Störspannungen in beide Richtungen erschwert, außerdem ergibt sich ein beruhigender Schutz des PCs vor Überspannungen auf dem CAN-Bus. Wer eine kostengünstige Lösung sucht und auf die galvanische Trennung verzichten möchte, kann einfach den DC/DC-Wandler weglassen und +5 V und Masse zwischen Primär- und Sekundärseite überbrücken.

Mit Jumper JP1 kann der Abschlußwiderstand von 120 Ω zugeschaltet werden. Grundsätzlich sollen die Abschlußwiderstände nur an den Enden der CAN-Bus-Linie angeschlossen werden. Wenn in einem System einzelne Busteilnehmer häufig an- und wieder abgekoppelt werden, ist es besser, wenn die Abschlußwiderstände auf der Platine nicht benutzt und statt dessen Sub-D-Stecker mit eingelötetem 120-Ω-Widerstand als Busabschluß verwendet werden.

Die Verdrahtung der 9-poligen Sub-D-Verbinder ist in der CAN-Norm nicht festgelegt. So wurde gegenüber früheren Elektor-Veröffentlichungen zum Thema CAN-Bus nicht nur die Anschlüsse 7 und 2 vertauscht, sondern auch mit dem Grundsatz gebrochen, daß der CAN-

Technische Daten

- Interface für parallele Schnittstelle im bidirektionalen EPP—Modus
- CAN-Bus-Anschluß über gebrückte 9-polige Sub-D-Verbinder
- Abschlußwiderstand 120 Ω zuschaltbar
- CAN-Controller SJA1000 oder PCA82C200
- Busankopplung über Transceiver PCA82C250
- Galvanische Trennung durch Optokoppler und Gleichspannungswandler
- Stromversorgung 9...12 V über Steckernetzteil

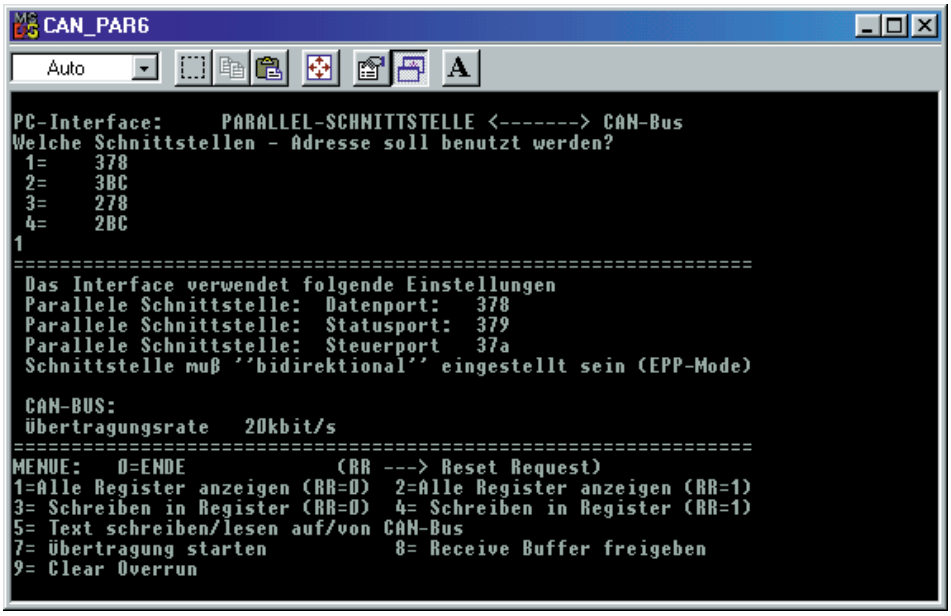


Bild 2. Programm-Menü des CAN-Controllers bei Reset Request=0.

Bus in beide Richtungen gleich ist. Statt dessen wird eine männliche und eine weibliche Buchse eingesetzt, so dass die Interface-Platine aus dem Bus entfernt werden und die beiden Buskabel einfach ineinander gesteckt werden können. Bedenken Sie, das die Numerierung eines Male- und eines Female-Verbinders gespiegelt sind.

Software

Zum PC-Interface gehört ein DOS-Programm, mit dem das Interface in Betrieb genommen werden kann. Das Programm und sein Quellkode ist auf Diskette 006004-1 erhältlich. Bevor das Programm gestartet wird, muß das Interface mit der Druckerschnittstelle und mit einer Versorgungsspannung von 9...12 V verbunden werden. Eine Verbindung zum CAN-Bus ist zu diesem Zeitpunkt noch nicht erforderlich.
Nach dem Programmstart unter DOS oder im DOS-Fenster gibt man die Adresse der Druckerschnittstelle an, anschließend taucht das Programm-Menü auf. Bei der ersten Menünummer erscheinen drei Spalten wie in

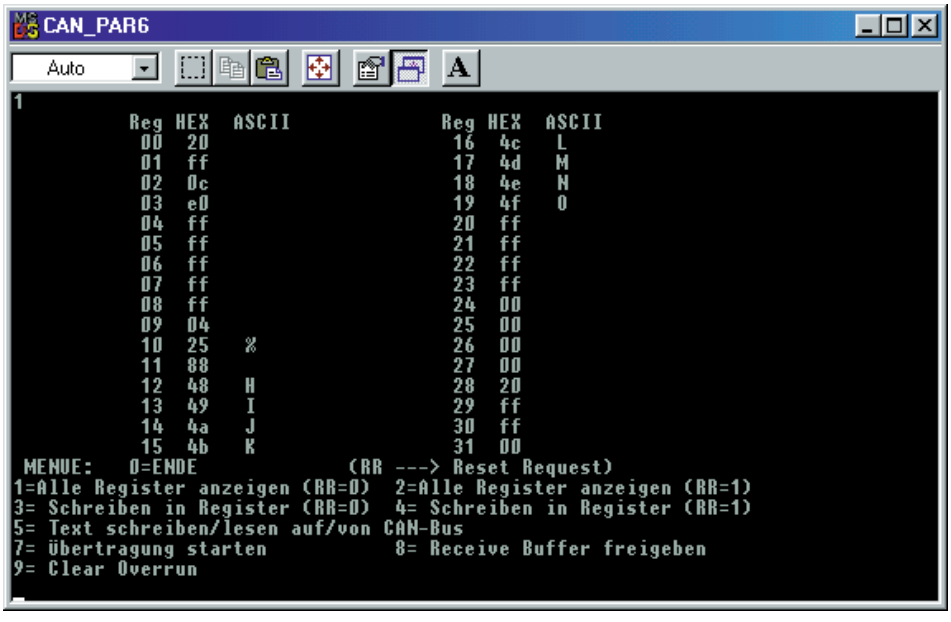


Bild 3. Register des CAN-Controllers. Die Register 4...8 sind gesperrt.

Bild 3. In der ersten Spalte (Reg) wird die Registeradresse, in der zweiten und dritten Spalte der Registerinhalt hexadezimal beziehungsweise als ASCII-Zeichen angezeigt. Sollte sich in allen Registern der Wert FF oder die Daten 00, 01, 02, 03 und so weiter befinden, dann ist entweder die Schnittstelle falsch adressiert oder sie arbeitet nicht bidirektional. Ein Hardware-Reset (Netzstecker raus, Netzstecker rein) kann diesen Makel beheben.
Funktioniert das Interface korrekt, so lassen sich alle Register mit Ausnahme der Register 4..8 lesen. Diese Register werden erst im Initialisierungsmodus zugänglich, wenn das Reset-Request-Bit (RR) des Command Registers auf 1 gesetzt ist. Der unterschiedliche Zugriff wird durch zwei verschiedene Schreib- beziehungsweise Lesebefehle im Menü (RR = 0 oder RR = 1) einfach möglich (siehe Kasten).

Ändern des Identifiers

Der Identifier ist nach Programmstart auf 300 (00100101100_B) eingestellt. Soll der ID beispielsweise 512 (01000000000_B) lauten, so müssen die Bits ID3...ID10 in das Register 10 und die Bits ID0...ID2 in das Register 11 an Bitposition 5...7 geschrieben werden. Bit 4 repräsentiert das RTR-Bit und wird bei der Übertragung eines Datensatzes (Data Frame) auf 0 gesetzt. Die Bits 0...3 legen die Anzahl der zu übertragenden Bytes nach der Formel

```
Anzahl= 8*DLC.3 + 4*DLC.2 +
2*DLC.1 + DLC.0
```

fest. Sollen acht Byte übertragen werden, so muß in das Register 11 der Wert 8_D (00001000_B) geschrieben werden. Mit dem Menübefehl 3 wird also 64_D in das Register 10 und anschließend 8_D in Register 11 geschrieben. Nach DIN ISO 11898 sind die Identifier 2032..2047 reserviert und dürfen nicht benutzt werden. Die sieben höchstwertigen Bits dürfen also nicht gleichzeitig 1 sein.

Ändern der Eingangsfilter

Durch die Eingangsfilterung wird erreicht, dass nur Datensätze mit vorher festgelegten Identifiern vom

CAN-Bus-Controller übernommen werden. Berücksichtigt werden nur die obersten 8 Bit des Identifiers. Eine Nachricht wird übernommen, wenn bei jeder Bitposition, bei der im *Acceptance Mask Register* eine 0 steht, eine Übereinstimmung zwischen der zugehörigen Bitposition im Identifier und im *Acceptance Code Register* festgestellt wird. Der Zugriff auf diese Register, erfolgt, wenn das RR-Bit vorher auf 1 gesetzt wurde. Daher muß zum Anzeigen der Register der Menübefehl 2 benutzt werden. In der Voreinstellung steht im *Acceptance Mask Register* (Nummer 5) 255_D (11111111_B), so dass alle einkommenden Nachrichten akzeptiert und übernommen werden. Das Register kann mit dem Menübefehl 4 (RR = 1) verändert werden.

Stückliste

Widerstände:
R1...R12,R17...R19,R21 = 390 Ω
R13 = 4k7
R14,R15,R20 = 10 k
R16 = 56 k
R22,R23 = 5Ω6
R24 = 120 Ω
R25 = 4-fach-SIL-Widerstandsarray 4k7
R26 = 8-fach-SIL-Widerstandsarray 4k7

Kondensatoren:
C1,C2 = 22 p
C3 = 10 µ/25 V stehend
C4,C9...C11 = 100 n, RM5
C5 = 470 µ/25 V stehend
C6 = 220 µ/10 V stehend
C7 = 100 µ/10 V stehend
C8 = 1 µ/10 V oder MKT R;5

Halbleiter:
D1,D2 = Z-Diode 12 V/400 mW
D3 = 1N4004
IC1 = PCA82C200 oder SJA1000 * (Philips)
IC2,IC3 = 6N137 (u.a. Toshiba)
IC4 = PCA82C250 (Philips)
IC5 = NMV0505SA (Newport, bei Farnell 589 810)
IC6 = 7805

Außerdem:
JP1 = Jumper
K1 = 9-polige Sub-D-Verbinder für Platinenmontage, gewinkelt, male
K2 = 9-polige Sub-D-Verbinder für Platinenmontage, gewinkelt, female
K3 = 25-polige Sub-D-Verbinder für Platinenmontage, gewinkelt, male
2 Lötnägel
X1 = Quarz 16 MHz
Software EPS 006004-1 (DOS-Interface mit C-Quellkode)
Platine EPS 000039-1

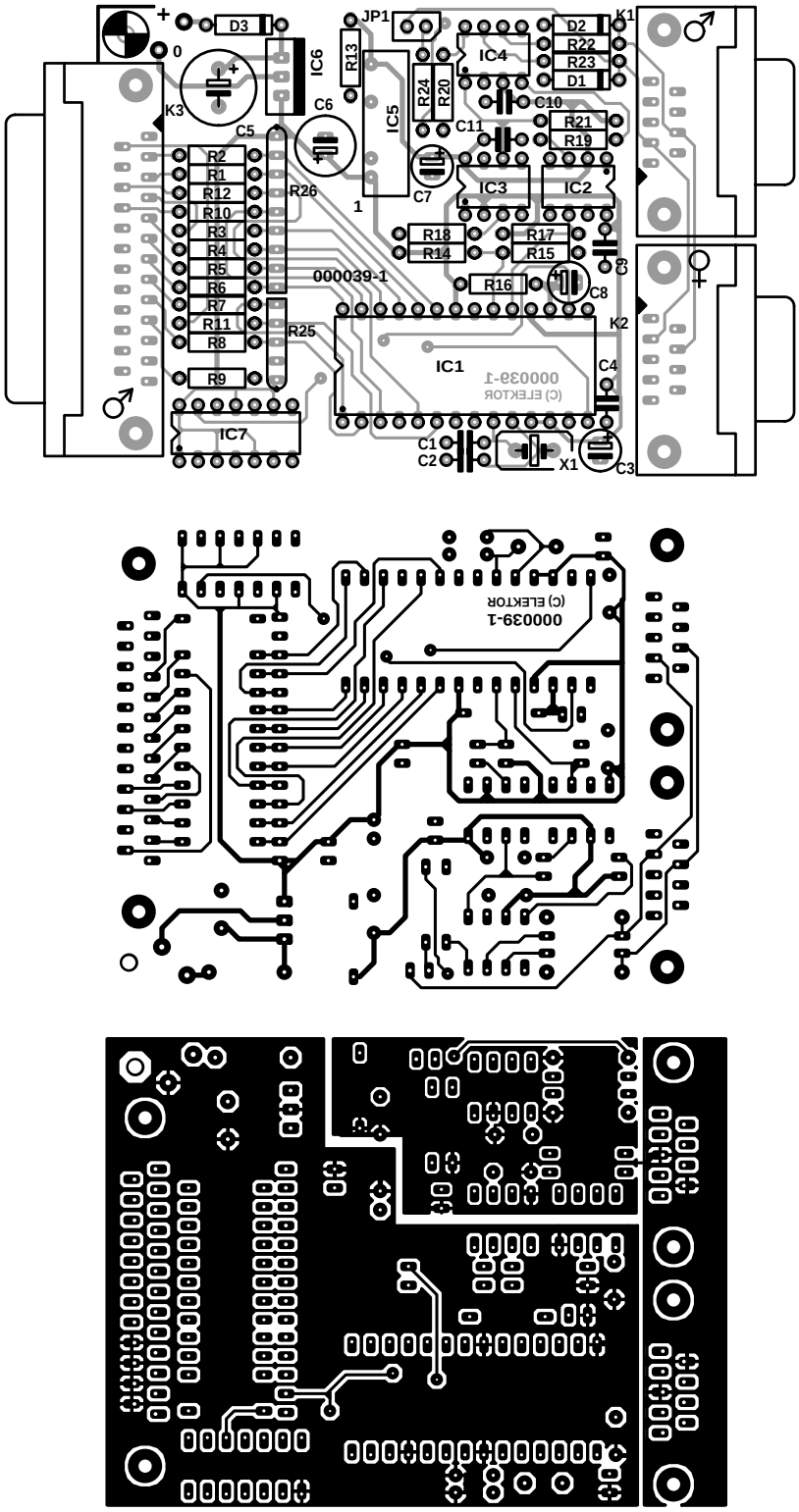


Bild 4. Layout und Bestückungsplan der Interface-Platine.

Software in C

Mit der Treibersoftware kann man auf alle 32 Steuerregister zugreifen. Über ein kleines Menü können die einzelnen Register geschrieben, gelesen und somit die gesamte Kommunikation abgewickelt werden. Außerdem besteht unter den Menüpunkten 11...13 die Möglichkeit, für Testzwecke direkt auf die PC-Register der Schnittstelle zuzugreifen. Die PC-Schnittstelle muß per Jumper oder im BIOS auf bidirektionalen Betrieb im EPP-Modus eingestellt werden. Die Übertragungsrate wird während der Initialisierung auf 1 MBit/s festgelegt.

Die Treibersoftware besteht im wesentlichen aus drei Funktionen:

1. void init82C200()
Die Funktion initialisiert den CAN-Bus-Controller entsprechend der Hardwarekonfiguration. Die Übertragungsrate ist auf 1 MBit/s eingestellt.

2. void wr_can(uchar adr, uchar wert)
Mit dieser Funktion kann jedes der 32 Controller-Register adressiert und beschrieben werden.

3. void rd_can(uchar adr)
Mit dieser Funktion kann jedes der 32 Controller-Register adressiert und gelesen werden.

Beispiele:

init82C200();	wird zu Beginn einmal aufgerufen, um die PC-Schnittstelle und den CAN-Bus-Controller zu initialisieren.
wr_can(12,50);	schreibt den Wert 50 in das Register 12 (1. Byte des Transmissionspuffers).
wr_can(1,10);	Transmission Request. Die Übertragung der Daten aus dem Transmissionspuffer wird gestartet.
wr_can(1,4);	Der Receive-Puffer wird wieder freigegeben
wr_can(0,1);	Reset-Request=1. Einige Register können nur in diesem Modus beschrieben und angezeigt werden.
wr_can(0,0);	Reset-Request=0. Einige Register können nur in diesem Modus beschrieben und angezeigt werden.
uinsigned char cw;	
cw=rd_can(22);	liest Register 22 (1. Register des Empfangspuffers)
cw=rd_can(2);	liest das Status-Register

CAN-Bus-Anschluss

Nach der erfolgreichen Inbetriebnahme sollte nun das Interface mit dem CAN-Bus verbunden werden. Für die weiteren Schritte muß sich mindestens ein weiterer Busteilnehmer am Bus befinden, zum Beispiel ein weiterer PC.
Auf den Menübefehl Nummer 7 hin wird eine Datenübertragung ausgelöst, der im Sendepuffer befindliche Datensatz entsprechend den Übertragungsparametern auf den Bus gegeben. Bei der Programminitialisierung wird bereits eine Buchstabenkombination in den Sendepuffer geschrieben, die nun bei der Empfangsstation im Empfangspuffer ankommt. Außerdem zeigt das Statusregister (Nummer 2) den Empfang einer Nachricht an, indem es Bit SR.0 setzt. Dieses Bit wird durch den Menübefehl 8 wieder zurückgesetzt. Der Controller besitzt einen zwei Datensätze umfassenden FIFO-Empfangspuffer. Läuft dieser Puffer über, so wird Bit SR.1 im Statusregister gesetzt. Menübefehl 9 setzt dieses Bit wieder zurück.
Eine Art "Fernschreibermodus" bietet der Menübefehl 5. Jedes über die Tastatur eingegebene Zeichen wird sofort auf den CAN-Bus übertragen und erscheint auf allen am PC angeschlossenen Empfangsstationen. Um eine eigene interessante Kommunikation mit diesem Interface aufzubauen, steht das Programm auch als Quellcode in C zur Verfügung. Wer das Programm entsprechend abändert, kann auch mit nur einem PC, zwei Schnittstellenkarten und zwei Interfaces Experimente am CAN-Bus durchführen.
Für die Kommunikation mit dem CAN-Bus-Controller stehen drei Funktionen zur Verfügung, die als Treiber sehr einfach in jedes C-Programm eingebunden werden können.

Mit der Funktion
`wr_can(uchar adr, uchar wert)`
wird das Byte *wert* in das mit *adr* bestimmte Register geschrieben.

Die Funktion
`rd_can(uchar adr)`
liefert den Inhalt des mit *adr* festgelegten Registers zurück.

Für die Initialisierung kann die Funktion
`initSJA1000()`
benutzt werden.

(000039)rg

Inhalt Diskette 006004-I

CAN_UK6	CPP	C-Quellkode in Englisch
CAN_PAR6	EXE	DOS-Programm in Deutsch
CAN_PAR6	CPP	C-Quellkode in Deutsch
CAN_UK6	EXE	DOS-Programm in Englisch
COPYRI~1	TXT	Copyrighthinweis
CONTENS	TXT	Dieser Text

Literatur

[1] W. Lawrentz (Hrsg.)
CAN System Engineering
1997 Springer Verlag New York

[2] Philips
SJA1000 Stand-alone
CAN controller DATA SHEET
www.semiconductors.philips.com

[3] Bosch
CAN Specification version 2.0
Robert Bosch Gmbh Stuttgart

[4] DIN ISO 11898
DIN, Beuth Verlag Berlin