Produktbeschreibung

USB 3080
basicCAR 3080

CAN/ LIN/ K-LINE/ J1850 Interfaces
Nutzerhandbuch Version 1.3

GÖPEL electronic

Get the total Coverage!

GÖPEL electronic GmbH
Göschwitzer Str. 58/60
D-07745 Jena
Tel.: +49-3641-6896-597
Fax: +49-3641-6896-944
E-Mail: ats_support@goepel.com
http://www.goepel.com

Gedruckt: 09.06.2010

Stand: Juni 2010

1 Installation

1.1 Hardwareinstallation

Die Hardware-Installation beschränkt sich bei USB 3080/ basicCAR 3080 i. Allg. auf den Austausch von Transceivermodulen.

Stellen Sie bitte unbedingt sicher, dass alle Installationsarbeiten im **ausgeschalteten** Zustand Ihres Systems erfolgen!

Wenn es notwendig ist, Transceivermodule zu tauschen, wird das entsprechende Gerät gemäß seinen Gegebenheiten geöffnet.
Dabei sind die allgemeinen Regeln zur Vermeidung von elektrostatischen Entladungen zu beachten.
Transceivermodule dürfen nie unter Spannung gezogen oder gesteckt werden!
Außerdem ist unbedingt ein lagerichtiges Stecken der Module zu realisieren (siehe Bestückung).

1.2 Treiberinstallation

Um die GÖPEL electronic USB-Treiber auf Ihrem System einzurichten, muss das GUSB Treiber Setup ausgeführt werden.
Starten Sie dazu das auf der mitgelieferten CD enthaltene Setup Programm *GUSB-Setup-*.exe* (der Stern steht für die Versionsnummer) und folgen Sie den Anweisungen.

Der zur Verfügung stehende Devicetreiber unterstützt gegenwärtig ausschließlich Windows® 2000/ XP-Systeme!

Wenn Sie eigene Software für USB 3080 bzw. basicCAR 3080-Baugruppen erstellen wollen, benötigen Sie ggf. zusätzliche Dateien für die anwenderspezifische Programmierung (*.LLB, *.H).
Diese werden nicht automatisch übernommen und müssen deshalb manuell von der mitgelieferten CD in Ihr Entwicklungsverzeichnis kopiert werden.

Die USB-Schnittstelle nutzt, falls möglich, die high-speed Datenrate entsprechend USB2.0 Spezifikation (ansonsten full-speed).

Nach der Treiberinstallation können Sie überprüfen, ob die Baugruppen einwandfrei vom System eingebunden worden sind.

Die folgende Abbildung zeigt die erfolgreiche Einbindung von vier USB 3080 (bzw. basicCAR 3080):

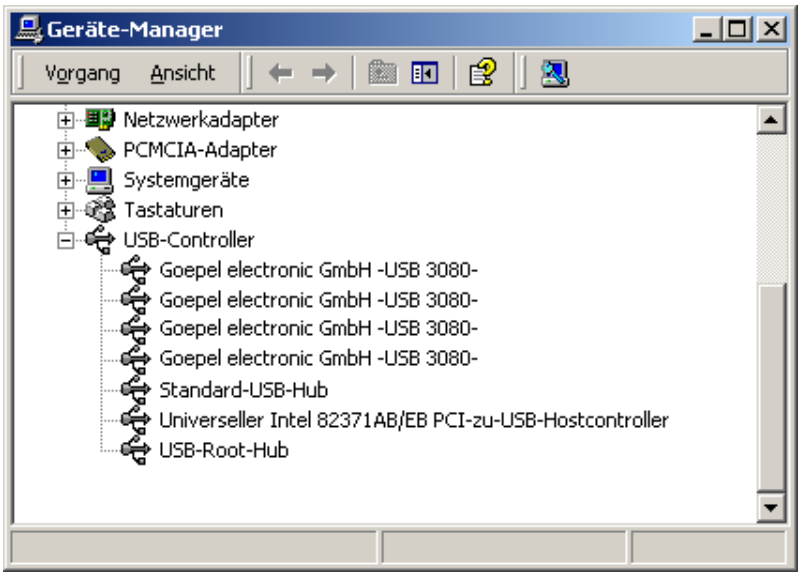

Abbildung 1-1: Anzeige Geräte-Manager

Beachten Sie bitte, dass der Geräte-Manager ALLE USB-Controller anzeigt.

2 Hardware

2.1 Bestimmung

Die Multi-Interface-Boards USB 3080 sind Kommunikationsboards mit USB 2.0-Interface der GÖPEL electronic GmbH.

Diese Boards werden in der allgemeinen Steuerungstechnik verwendet, u.a. in der Automobiltechnik.

Abbildung 2-1: USB 3080

Beachten Sie bitte, dass ein Download des Xilinx FPGAs für die Funktion eines USB 3080 Boards unabdingbar ist (siehe Xilinx_Download unter Windows Device Treiber)!

Zum Betrieb von USB 3080 Boards ist das GÖPEL electronic USB-Rack erforderlich, das bis zu 16 GÖPEL electronic USB-Boards aufnehmen kann.

Die Stromversorgung erfolgt in diesem Fall über das eingebaute Netzteil.

basicCAR 3080 ist ein GÖPEL electronic GmbH Stand-alone-Gerät auf der Grundlage eines USB 3080 Kommunikationsboards zum Anschluss an einen PC oder Laptop, das für den eigenständigen Einsatz außerhalb komplexer Testsysteme entwickelt wurde.

Die externe Spannungszufuhr erlaubt die Nutzung dieses Gerätes zur Datenaufnahme und Signalkontrolle in Kraftfahrzeugen.

Abbildung 2-2: basicCAR 3080

Die Stromversorgung mit 8-25 VDC (und ca. 350 mA bei 12 V) erfolgt über die beiden Buchsen für ext. Power Supply (rot = plus/ blau= minus) an der dem Steckverbinder für die Kommunikationsschnittstellen gegenüber liegenden Geräteseite. Diese Buchsen werden zur Versorgung der internen Logik genutzt. Der Stromversorgungsanschluss an der blauen Buchse ist mit den GND-Anschlüssen der USB-Schnittstelle verbunden.

Alle Anschlüsse der Kommunikationschnittstellen sind galvanisch vom USB-Interface und von der internen Logik getrennt.

USB 3080/ basicCAR 3080-Baugruppen bieten in der maximalen Ausbaustufe folgende Ressourcen:

- 2 x CAN
- 2 x LIN oder K-Line
- 1 x J1850 VPW
- 1 x J1850 PWM
  (in diesem Fall ist nur EIN LIN oder K-Line Interface möglich)
- 4 x digital Input
- 4 x digital Output
- 2 x analog Input
- 1 x Weckleitung

Die Kommunikation für J1850 PWM erfolgt über die Schnittstelle für K-Line/ LIN2!!!

Sollten die Ressourcen eines basicCAR 3080 für Ihre Anwendungen nicht ausreichen, steht das GÖPEL electronic USB-Rack zur Verfügung, das bis zu 16 GÖPEL electronic USB-Boards aufnehmen kann.
In diesem Fall erfolgt die Stromversorgung über ein eingebautes Netzteil mit 230V oder 115V Kaltgeräte-Anschluss an der Rückseite des Racks.

2.2 Technische Daten

2.2.1 Abmessungen

(Breite x Höhe x Tiefe):

- USB 3080: 4 TE x 130 mm x 185 mm
- basicCAR 3080: 145 mm x 70 mm x 220 mm

Die Angaben für USB 3080 beziehen sich auf ein Board im GÖPEL electronic USB-Rack.

2.2.2 Kennwerte

Symbol	Kennwert	Min.	Typ.	Max.	Einheit	Bemerkung
V_{BAT}	Batteriespannung		12	27	V	Abh. v. Transceivertyp
	Übertragungsrate			1	MBaud	CAN
	Übertragungsrate			22	kBaud	LIN
R_{bus}	Abschlusswiderstand 1		120		Ohm	CAN Jumper gesteckt
R_{bus}	Abschlusswiderstände 2		10		kOhm	CAN Jumper gesteckt
R_{Pullup}	Pullup-Widerstand		680		Ohm	K-Line Jumper gesteckt
V_{in}	Eingangsspannung	3,3		50	V	Digital Input
V_{out}	Ausgangsspannung			V_{BAT}	V	Digital Output, OC
V_{in}	Eingangsspannung		20	26	V	Analog Input
V_{iso}	galvanische Trennung	560			V	USB In-/Output

Die analogen Eingangs-Kanäle (analog inputs) sind mit dem Schaltkreis LTC 1400 (AD-Wandler) von Linear Technology realisiert. Dieses Bauelement weist eine Auflösung von 12 Bit sowie einen Eingangsspannungsbereich von 0..4,095V auf.

Durch den Eingangs-Spannungsteiler (122K/22K) ergibt sich für die gemessene Spannung:
V_{mess} = AD-Wandlerwert * 1mV * (122K/22K).

Beachten Sie bitte, dass die Höhe der Versorgungsspannung für die Transceiver (Batteriespannung V_{Bat}) von dem Transceiver bestimmt wird, der die NIEDRIGSTE maximale Versorgungsspannung hat!

2.3 Aufbau

2.3.1 Allgemeines

Bei den einzelnen Baugruppen von USB 3080/ basicCAR 3080 dient ein ASIC (TC1775) als Interface zum USB-Bus. Dieser beinhaltet alle notwendigen Funktionsblöcke, die für eine Kommunikation mit dem Rechner-Bus notwendig sind.

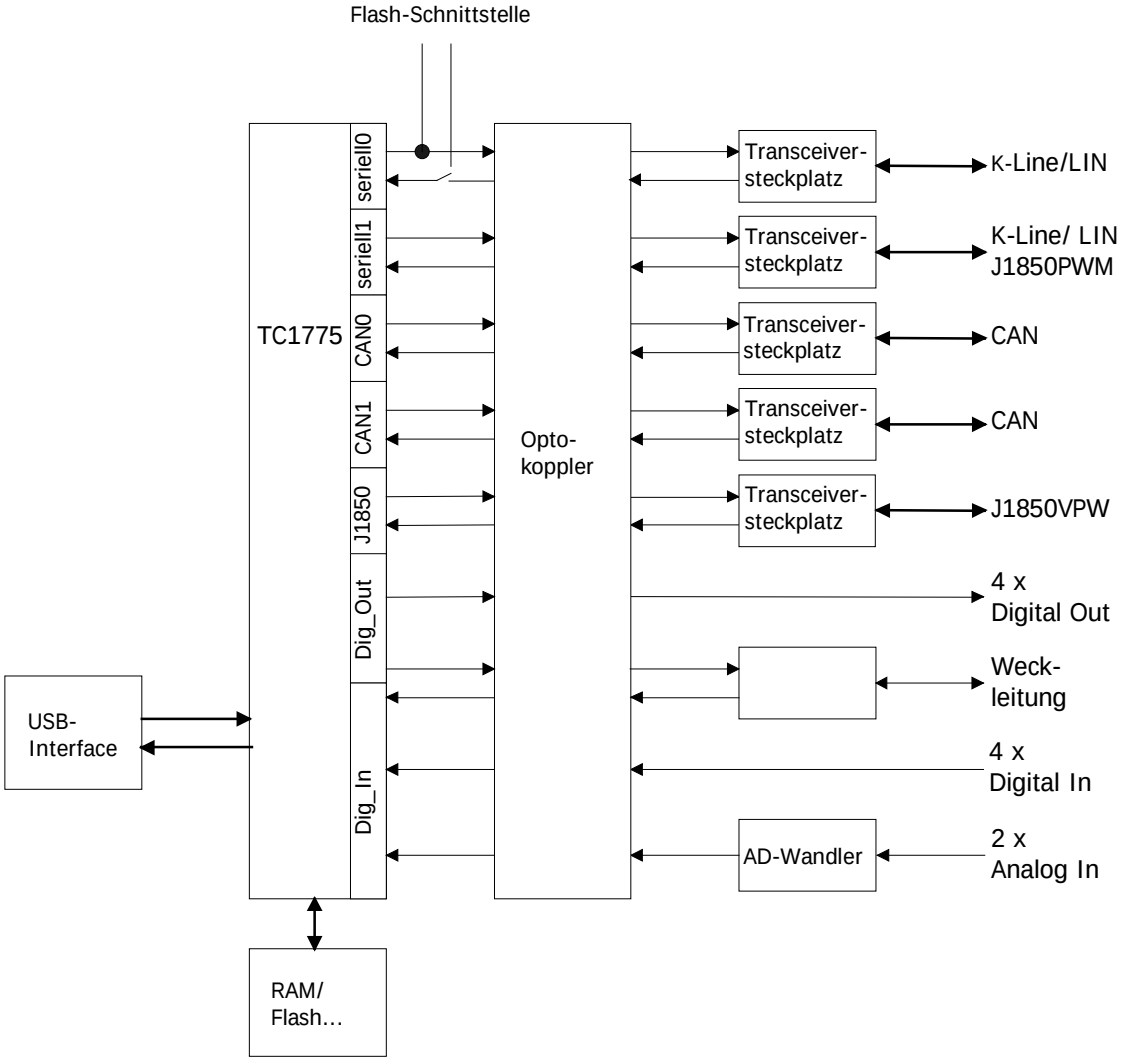

Abbildung 2-3: Blockdiagramm eines Kommunikationsboards USB 3080

Bitte verwenden Sie zum Anschluss der Baugruppen an die USB-Schnittstelle des PCs die im Lieferumfang enthaltenen USB-Kabel. Andere Kabel sind u. U. nicht geeignet!

2.3.2 Kommunikationsschnittstellen

2 x CAN-Interface Version 2.0b:

Für die uneingeschränkte Funktion eines CAN-Interfaces an einem Netzwerk ist der verwendete Transceiver entscheidend. Häufig funktionieren CAN-Netzwerke nur, wenn alle Teilnehmer kompatible Transceiver im Netz haben.

Damit die Nutzer eines USB 3080-Boards keinen Einschränkungen unterliegen, sind die Transceiver als steckbare Module ausgeführt. Dabei stehen verschiedene Varianten (Highspeed, Lowspeed, Single-Wire u.a.) zur Auswahl, die einfach auszutauschen sind.

Neben dem Transceiver ist der Busabschlusswiderstand für die einwandfreie Funktion des CAN-Netzwerkes wichtig.
Werden Highspeed CAN-Transceiver verwendet, ist i. Allg. ein 120 Ohm Widerstand für jede CAN-Schnittstelle aktiv. Diese Widerstände können durch Ziehen der Jumper J1401 bzw. J1501 deaktiviert werden.

Bei Verwendung von Lowspeed CAN-Transceivern sind i. Allg. zwei Abschlusswiderstände von 10 kOhm für RTH und RTL für jede CAN-Schnittstelle aktiv (durch Bestücken der Jumper J1402/ J1403 bzw. J1502/ J1503). In diesem Fall müssen die Jumper J1401 bzw. J1501 geöffnet sein.

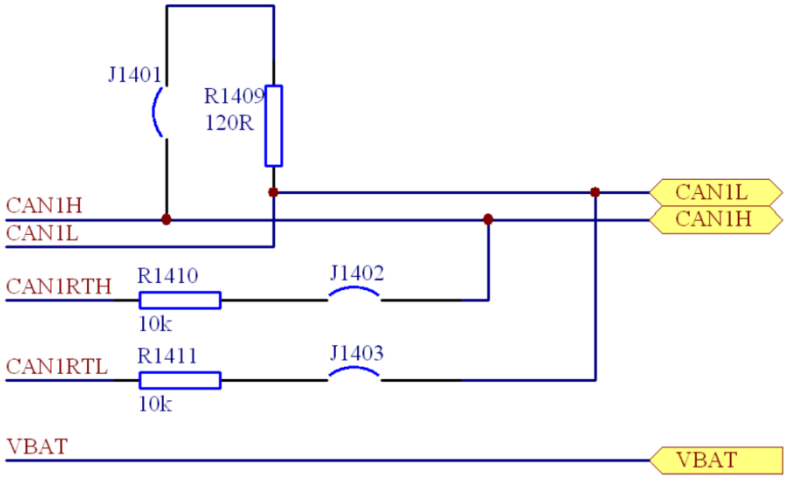

Abbildung 2-4: CAN Schnittstelle

2 x LIN-Interface Version 2.0 oder
2 x K-Line Interface (ISO 9141)

LIN:

Die Transceiver sind als steckbare Module ausgeführt. I. Allg. wird der TJA1020 für diese Transceiver verwendet.

In der Standardausführung der Transceivermodule kann per Software über die Relais Rel1 für LIN1 bzw. Rel2 für LIN2 zwischen Master- und Slave-Konfiguration umgeschaltet werden. Die Pullup-Widerstände für LIN befinden sich auf dem Transceivermodul, sodass die Jumper J1601 bzw. J1701 NICHT bestückt werden dürfen!

Über die Anschlüsse V_{BAT} wird die Versorgungsspannung der Transceivermodule angeschlossen. Gemäß der LIN-Spezifikation soll diese Versorgung über eine Verpolschutzdiode erfolgen, sodass die Jumper J1602 bzw. J1703 NICHT bestückt werden dürfen.

K-Line:

Die Transceiver sind als steckbare Module ausgeführt. I. Allg. wird der L9637 für diese Transceiver verwendet.

Über die Anschlüsse V_{BAT} wird die Versorgungsspannung der Transceivermodule angeschlossen. Zum Überbrücken der Verpolschutzdiode für V_{BAT} für LIN müssen die Jumper J1602 bzw. J1703 bestückt sein.

Falls der Pullup-Widerstand gegen V_{BAT} aktiviert werden soll, müssen die Jumper J1601 bzw. J1701 bestückt werden.

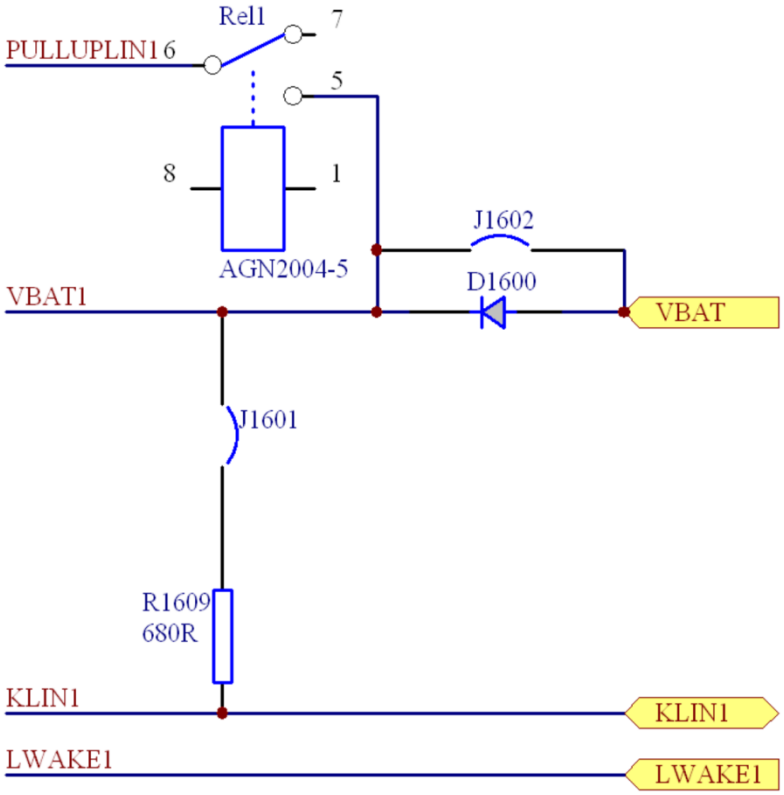

Abbildung 2-5:
LIN/ K-Line Schnittstelle

J1850 Interfaces:

Die Transceiver sind als steckbare Module ausgeführt.

I. Allg. wird der AU5780 für J1850 VPW Transceiver verwendet.

Die Ausgangsschaltung für den J1850 PWM Transceiver ist mit diskreten Bauelementen realisiert.

Falls ein J1850 VPW Interface realisiert werden soll, muss der Transceiver auf dem Steckplatz für den J1850 Transceiver bestückt werden.
Um ein J1850 PWM Interface zu realisieren, wird der Transceiver auf dem Steckplatz für K-Line/ LIN 2 Transceiver bestückt (siehe Abbildung 2-6).

J1701 darf bei einem J1850 PWM Interface NICHT bestückt sein!

2.3.3 Adressierung

Die Adressierung von USB 3080-Baugruppen z.B. im GÖPEL electronic USB-Rack erfolgt ausschließlich über deren Seriennummern (siehe Ansteuersoftware): Die Baugruppe mit der KLEINSTEN Seriennummer ist immer das Gerät Nummer 1.

Zur Erhöhung der Übersichtlichkeit empfehlen wir, USB 3080-Baugruppen in aufsteigender Reihenfolge der Seriennummern im USB-Rack anzuordnen.

2.3.4 Bestückung

Abbildung 2-6 zeigt schematisch die Bestückungsseite einer USB 3080-Baugruppe. In dieser Abbildung ist die Lage der Transceiver-module, Steckverbinder, DIP-Schalter und Jumper zu erkennen.

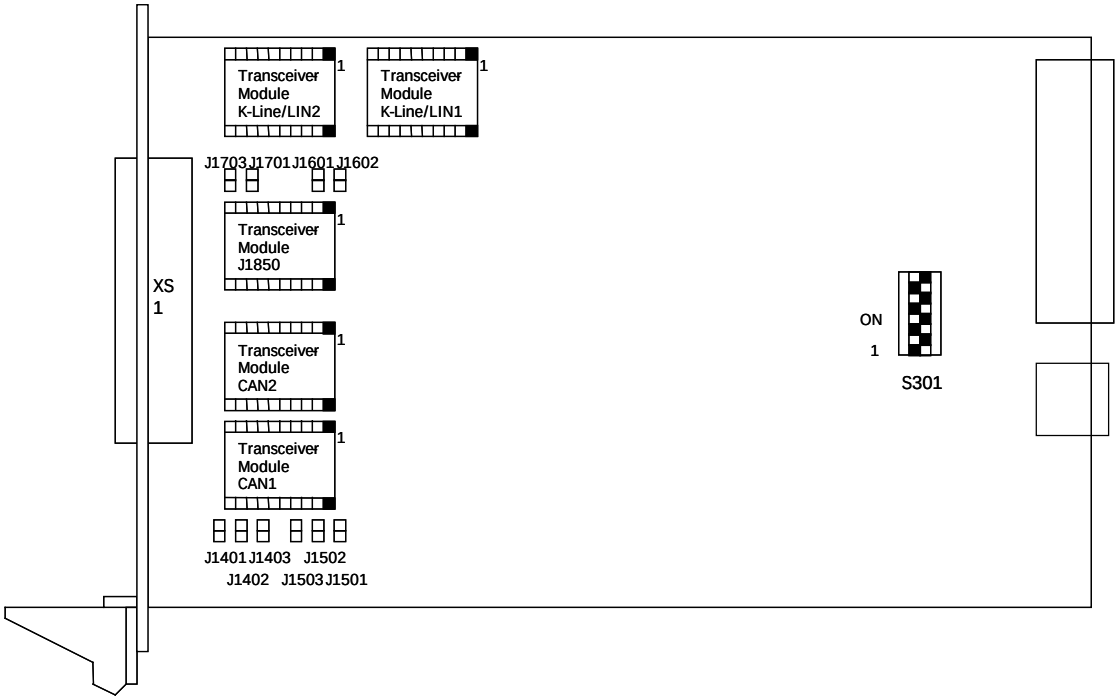

Abbildung 2-6: Schematischer Bestückungsplan einer USB 3080-Baugruppe

Die Konfigurationselemente aus Abbildung 2-6 sowie die Bezeichnungen der Steckverbinder für die Transceiverbestückung werden in der folgenden Tabelle erläutert:

XS1401	Transceivermodul für CAN1
J1401	Jumper zum Aktivieren des **120Ω** Abschlusswiderstandes für CAN1
J1402	Jumper zum Aktivieren des **10kΩ** Abschlusswiderstandes RTH für CAN1
J1403	Jumper zum Aktivieren des **10kΩ** Abschlusswiderstandes RTL für CAN1
XS1501	Transceivermodul für CAN2
J1501	Jumper zum Aktivieren des **120Ω** Abschlusswiderstandes für CAN2
J1502	Jumper zum Aktivieren des **10kΩ** Abschlusswiderstandes RTH für CAN2
J1503	Jumper zum Aktivieren des **10kΩ** Abschlusswiderstandes RTL für CAN2
XS1601	Transceivermodul für LIN1/ K-Line1
J1601	Jumper zum Aktivieren des **680Ω** Pullupwiderstandes gegen V_{BAT} für K-Line1
J1602	Jumper zum Überbrücken der Verpolschutzdiode für V_{BAT} für LIN1
XS1701	Transceivermodul für LIN2/ K-Line2/ J1850 PWM
J1701	Jumper zum Aktivieren des **680Ω** Pullupwiderstandes gegen V_{BAT} für K-Line2
J1703	Jumper zum Überbrücken der Verpolschutzdiode für V_{BAT} für LIN2
XS1801	Transceivermodul für J1850 VPW
S301	DIP-Schalter der USB 3080-Boards zur Konfiguration des Microcontrollers. **Die Einstellung darf nicht verändert werden!**

2.3.5 Belegung Front-steckverbinder

Typ: DSub 25-polig Buchse

Die Schnittstellen stehen über diesen Steckverbinder an der Frontseite der Kommunikationsboards USB 3080 zur Verfügung.

Die Belegung ist in der folgenden Tabelle dargestellt:

lfd. Nr.	Anschluss XS1	Signalname	Bemerkung
1	1	CAN1_High	CAN-Bus-Leitung High
2	14	CAN1_Low	CAN-Bus-Leitung Low
3	2	CAN2_High	CAN-Bus-Leitung High
4	15	CAN2_Low	CAN-Bus-Leitung Low
5	3	VBAT	Versorgungsspannungseingang für Transceiver (siehe Kennwerte)
6	16	GND	Massepotenzial Kommunikationsschnittstellen
7	4	LIN1/ K-Line1	abhängig von Transceiver-Bestückung
8	17	L-Line1/ WAKE1	abhängig von Transceiver-Bestückung
9	5	LIN2/ K-Line2/ J1850 PWM+	abhängig von Transceiver-Bestückung
10	18	L-Line2/ WAKE2/ J1850 PWM-	abhängig von Transceiver-Bestückung
11	6	J1850 VPW	
12	19	-	Bitte nicht belegen!
13	7	VBAT	Versorgungsspannungseingang für Transceiver (siehe Kennwerte)
14	20	GND	Massepotenzial Kommunikationsschnittstellen
15	8	Analog Input1	
16	21	Analog Input2	
17	9	Digital Input1	
18	22	Digital Output1	
19	10	Digital Input2	
20	23	Digital Output2	
21	11	Digital Input 3	
22	24	Digital Output3	
23	12	Digital Input 4	
24	25	Digital Output4	
25	13	Weckleitung	

Bei K-Line sind ggf. auf PIN 17/ 18 die Anschlüsse für die L-Line verdrahtet (in Abhängigkeit von der Ausgangsbeschaltung).

Bei LIN sind ggf. auf PIN 17/ 18 die WAKE-Anschlüsse verdrahtet (in Abhängigkeit von der Transceiver-Wahl).

Pin 3 und 7 (V_{BAT}) sowie Pin 16 und 20 (GND) sind auf jeder USB 3080/ basicCAR 3080-Einzelbaugruppe gebrückt!

USB-Schnittstelle

Die USB-B-Buchse (USB-Standardbelegung) für das USB 2.0 Interface befindet sich an der dem Steckverbinder für die Kommunikations-schnittstellen gegenüber liegenden Seite eines USB 3080-Boards.

2.3.6 LED Anzeige

Die LEDs zeigen folgende Zustände an:

Rote LED D100: /HDRST Hardware Reset Indication-Ausgang des Microcontrollers

Grüne LED D700: Zustandsanzeige Spannung 5V (intern)
Grüne LED D701: Zustandsanzeige Spannung 3,3V (intern)
Grüne LED D702: Zustandsanzeige Spannung 2,5V (intern)

Gelbe LED D801: Zustandsanzeige CAN 1
Gelbe LED D802: Zustandsanzeige CAN 2
Gelbe LED D803: Zustandsanzeige K-Line/LIN 1
Gelbe LED D804: Zustandsanzeige K-Line/LIN 2

Die LEDs sind folgendermaßen auf der Frontplatte angeordnet:

D702 D701 D700 D100

D801 D802 D803 D804

2.4 Lieferhinweise

USB 3080/ basicCAR 3080-Baugruppen werden in folgenden Basisvarianten geliefert:

- 1x CAN Schnittstelle und 1x LIN Schnittstelle oder
- 1x CAN Schnittstelle und 1x K-Line Schnittstelle

Diese Basisvarianten können durch folgende Optionen erweitert werden:

- 1x Zusätzliche CAN Schnittstelle
- 1x Zusätzliche LIN Schnittstelle oder K-Line Schnittstelle
- 1x Zusätzliche J1850 VPW Schnittstelle
- 1x Zusätzliche J1850 PWM Schnittstelle

Wenn Sie die Option 1x Zusätzliche J1850 PWM Schnittstelle wählen, ist die Option 1x Zusätzliche LIN Schnittstelle oder K-Line Schnittstelle NICHT möglich.

Außer der Auswahl der Schnittstelle selbst muss auch der Typ des zugehörigen CAN/ LIN/ K-Line/ J1850 Transceivers für jede Schnittstelle festgelegt werden.

Für jede CAN/ K-Line/ J1850-Schnittstelle sind außerdem die erforderlichen Funktionalitäten anzugeben.

Sollten die Ressourcen eines basicCAR 3080 für Ihre Anwendungen nicht ausreichen, steht ein GÖPEL electronic USB-Rack zur Verfügung, das bis zu 16 GÖPEL electronic USB-Boards aufnehmen kann.
In diesem Fall erfolgt die Stromversorgung über ein eingebautes Netzteil mit 230 V oder 115 V Kaltgeräte-Anschluss an der Rückseite des Racks.

3 Ansteuersoftware

Zur Einbindung der USB 3080/ basicCAR 3080-Hardware in eigene Applikationen existieren drei Möglichkeiten:

- Programmieren über G-API
- Programmieren über DLL-Funktionen
- Programmieren mit LabVIEW

3.1 Programmieren über G-API

Das bevorzugte User Interface für diese GÖPEL Hardware ist die G-API (GÖPEL-API).

Sie finden alle benötigten Informationen im Ordner *G-API* der mitgelieferten CD.

3.2 Programmieren über DLL-Funktionen

Die Programmierung über DLL-Funktionen ist weiterhin für bestehende Projekte möglich, bei denen noch nicht mit der GÖPEL G-API gearbeitet werden kann.

Die Dokumentation GÖPEL Firmware senden wir Ihnen auf Anforderung gern zu. Bitte setzten Sie sich bei Bedarf mit unserem Vertrieb in Verbindung.

Der in der folgenden Funktionsbeschreibung verwendete Begriff GUSB_Platform ist der Name eines USB Treibers der GÖPEL electronic GmbH.

Informationen zu den Strukturen, Datentypen und Error-Codes enthalten die Header – die entsprechenden Dateien finden Sie auf der mitgelieferten CD.

In diesem Nutzerhandbuch ist unter Controller IMMER der den CAN/ LIN/ K-Line/ J1850-Schnittstellen zugeordnete Microcontroller einer USB 3080/ basicCAR 3080-Baugruppe zu verstehen.

Unter USB Controller wird in diesem Nutzerhandbuch der Controller verstanden, der das USB 2.0-Interface auf der USB 3080/ basicCAR 3080-Baugruppe bereitstellt.

3.2.1 Windows Device Treiber

Die für die Programmierung unter Verwendung des Windows Device Treibers nutzbaren DLL-Funktionen sind in den folgenden Abschnitten beschrieben:

- Driver_Info
- DLL_Info
- Write_FIFO
- Read_FIFO
- Read_FIFO_Timeout
- Write_COMMAND
- Read_COMMAND
- Xilinx_Download
- Xilinx_Version

3.2.1.1 *Driver_Info*

Die Funktion GUSB_Platform_Driver_Info dient zur Status-Abfrage des Hardware-Treibers und zur internen Initalisierung der erforderlichen Handles.

Diese Funktion MUSS einmalig vor dem Aufruf aller anderen Funktionen des GUSB_Platform Treibers ausgeführt werden.

Format:

```
int GUSB_Platform_Driver_Info(GUSB_Platform_DriverInfo *pDriverInfo,
                              unsigned int LengthInByte)
```

Parameter:

Zeiger, z.B. pDriverInfo
auf eine Datenstruktur
Zur Struktur siehe das File *GUSB_Platform.h* auf der mitgelieferten CD

LengthInByte

Größe des Speicherbereiches, auf den pDriverInfo zeigt, in Bytes

Beschreibung:

Die Funktion GUSB_Platform_Driver_Info gibt Informationen über den Status des Hardware-Treibers zurück.

Dazu muss der Funktion die Adresse des Zeigers pDriverInfo übergeben werden. Mit Hilfe des Parameters LengthInByte prüft die Funktion intern den korrekt initialisierten Anwenderspeicher.

Die Funktion füllt die Struktur, auf die pDriverInfo zeigt, mit Angaben zur Treiberversion, der Anzahl aller sich im System befindenden USB Controller (die von diesem Treiber unterstützt werden), und Informationen darüber, wie z.B. die Seriennummer(n).

Die Bereitstellung der Hardwareinformationen und die Initialisierung der zugehörigen Handles ist für die weitere Nutzung der USB-Hardware zwingend erforderlich.

3.2.1.2 DLL_Info

Die Funktion GUSB_Platform_DLL_Info dient zur Abfrage von Informationen über die DLL.

Format:

```
int GUSB_Platform_DLL_Info(GUSB_Platform_DLLinfo *DLLinformation)
```

Parameter

Zeiger, z.B. DLLinformation
auf eine Datenstruktur
Zur Struktur siehe das File *GUSB_Platform.h* auf der mitgelieferten CD

Beschreibung:

Die Funktion GUSB_Platform_DLL_Info gibt die Struktur DLLinfo zurück. Der erste Integerwert enthält die Versionsnummer der *GUSB_Platform.dll*.

Beispiel:

Die Versionsnummer **1.23** wird als Wert **123** zurückgegeben, Version **1.60** als Wert **160**.

3.2.1.3 *Write_FIFO*

Die Funktion GUSB_Platform_Write_FIFO dient zum Senden eines Befehls zum Controller.

Format:

```
int GUSB_Platform_Write_FIFO(unsigned int DeviceName,
                             unsigned int DeviceNumber,
                             t_USB_FIFO_Interface_Buffer *pWrite,
                             unsigned int DataLength)
```

Parameter:

DeviceName

Typ des adressierten Gerätes (Nummer, die in *GUSB_Platform_def.h* deklariert ist, für USB 3080/ basicCAR 3080 = 6)

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

Zeiger, z.B. pWrite
auf den Bereich für Schreibdaten

DataLength

Größe des Speicherbereiches, auf den pWrite zeigt, in Bytes
Die Daten bestehen aus Befehlskopf und Befehlsbytes
(z. Zt. max. 1024 Byte pro Befehl)

Beschreibung:

Die Funktion GUSB_Platform_Write_FIFO sendet einen Befehl zum Controller.

Die allgemeine Befehlsstruktur ist im Abschnitt Allgemeines zur Firmware der Dokumentation GÖPEL Firmware beschrieben.

3.2.1.4 Read_FIFO

Die Funktion GUSB_Platform_Read_FIFO dient zum Lesen einer Antwort vom Controller.

Format:

```
int GUSB_Platform_Read_FIFO(unsigned int DeviceName,
                            unsigned int DeviceNumber,
                            t_USB_FIFO_Interface_Buffer *pRead,
                            unsigned int *DataLength)
```

Parameter:

DeviceName

Typ des adressierten Gerätes (Nummer, die in *GUSB_Platform_def.h* deklariert ist, für USB 3080/ basicCAR 3080 = 6)

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

Zeiger, z.B. pRead
auf den Lesepuffer
Nach erfolgreicher Funktionsausführung befinden sich die Daten im Lesepuffer, bestehend aus Antwortkopf und Antwortbytes
(z. Zt. max. 1024 Byte pro Antwort)

DataLength

Vor Funktionsaufruf: Anzugebende Größe des Lesepuffers in Bytes

Nach Funktionsausführung: Anzahl der tatsächlich gelesenen Bytes

Beschreibung:

Die Funktion GUSB_Platform_Read_FIFO liest die älteste vom Controller geschriebene Antwort. Ist während einer Timeout-Zeit von 100 ms (nicht einstellbar) keine Antwort empfangen worden, liefert die Funktion jedoch KEINEN Fehler zurück: In diesem Fall ist der Wert für die Anzahl der tatsächlich gelesenen Bytes = 0 !!!

3.2.1.5 Read_FIFO_Timeout

Die Funktion GUSB_Platform_Read_FIFO_Timeout dient zum Lesen einer Antwort vom Controller, wobei ein Timeout vorzugeben ist.

Format:

```
int GUSB_Platform_Read_FIFO_Timeout(unsigned int DeviceName,
                                    unsigned int DeviceNumber,
                                    t_USB_FIFO_Interface_Buffer *pRead,
                                    unsigned int *DataLength,
                                    unsigned int Timeout)
```

Parameter:

DeviceName

Typ des adressierten Gerätes (Nummer, die in *GUSB_Platform_def.h* deklariert ist, für USB 3080/ basicCAR 3080 = 6).

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

Zeiger, z.B. pRead
auf den Lesepuffer
Nach erfolgreicher Funktionsausführung befinden sich die Daten im Lesepuffer, bestehend aus Antwortkopf und Antwortbytes
(z. Zt. max. 1024 Byte pro Antwort)

DataLength

Vor Funktionsaufruf: Anzugebende Größe des Lesepuffers in Bytes

Nach Funktionsausführung: Anzahl der tatsächlich gelesenen Bytes

Timeout

Angabe in Millisekunden (Standardwert: 500)

Beschreibung:

Die Funktion GUSB_Platform_Read_FIFO_Timeout liest die älteste vom Controller geschriebene Antwort.
Ist während der einstellbaren Timeout-Zeit keine Antwort empfangen worden, liefert die Funktion jedoch KEINEN Fehler zurück: In diesem Fall ist der Wert für die Anzahl der tatsächlich gelesenen Bytes = 0 !!!

3.2.1.6 Write_COMMAND

Die Funktion GUSB_Platform_Write_COMMAND dient zum Senden eines Configuration-Befehls zum USB Controller.

Format:

```
int GUSB_Platform_Write_COMMAND(unsigned int DeviceName,
                                unsigned int DeviceNumber,
                                t_USB_COMMAND_Interface_Buffer *pWrite,
                                unsigned int DataLength)
```

Parameter:

DeviceName

Typ des adressierten Gerätes (Nummer, die in *GUSB_Platform_def.h* deklariert ist, für USB 3080/ basicCAR 3080 = 6).

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

Zeiger, z.B. pWrite
auf den Bereich für Schreibdaten

DataLength

Größe des Speicherbereiches, auf den pWrite zeigt, in Bytes
Siehe auch Steuerbefehle USB Controller
(z. Zt. max. 64 Byte pro Befehl)

Beschreibung:

Die Funktion GUSB_Platform_Write_COMMAND sendet einen Befehl zum USB Controller.

Die allgemeine Struktur ist im Abschnitt Steuerbefehle USB Controller beschrieben.

3.2.1.7 Read_COMMAND

Die Funktion GUSB_Platform_Read_COMMAND dient zum Lesen einer Antwort vom USB Controller.

Format:

```
int GUSB_Platform_Read_COMMAND(unsigned int DeviceName,
                               unsigned int DeviceNumber,
                               t_USB_COMMAND_Interface_Buffer *pRead,
                               unsigned int *DataLength)
```

Parameter:

DeviceName

Typ des adressierten Gerätes (Nummer, die in *GUSB_Platform_def.h* deklariert ist, für USB 3080/ basicCAR 3080 = 6).

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

Zeiger, z.B. pRead
auf den Lesepuffer
Nach erfolgreicher Funktionsausführung befinden sich die Daten im Lesepuffer, bestehend aus Antwortkopf und Antwortbytes
Siehe auch Steuerbefehle USB Controller
(z. Zt. min. 64 Byte pro Antwort)

DataLength

Vor Funktionsaufruf: Anzugebende Größe des Lesepuffers in Bytes

Nach Funktionsausführung: Anzahl der tatsächlich gelesenen Bytes

Beschreibung:

Die Funktion GUSB_Platform_Read_COMMAND liest die älteste vom USB Controller geschriebene Antwort zurück.

Werden mehrere Antworten vom USB Controller bereitgestellt, werden maximal zwei dieser Antworten in den Puffer des USB Controllers geschrieben.

Weitere ggf. bereitgestellte Antworten gehen verloren!

3.2.1.8 Xilinx_Download

Die Funktion GUSB_Platform_Xilinx_Download dient zum Laden eines FPGA-Files in den XILINX.

Format:

```
int GUSB_Platform_Xilinx_Download(unsigned int DeviceName,
                                  unsigned int DeviceNumber,
                                  char *pFileName,
                                  unsigned char *pFirmwareErrorCode)
```

Parameter:

DeviceName

Typ des adressierten Gerätes (Nummer, die in *GUSB_Platform_def.h* deklariert ist, für USB 3080/ basicCAR 3080 = 6).

DeviceNumber

Nummer des adressierten Gerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

pFileName

Pfad des zu ladenden FPGA-Files

pFirmwareErrorCode

Fehlercode, der während der Abarbeitung dieser DLL-Funktion auftritt (bei Fehlercode 0 ist kein Fehler aufgetreten)

(error codes -> card firmware siehe *GUSB_Platform_def.h*)

Beschreibung:

Die Funktion GUSB_Platform_Xilinx_Download dient zum Laden eines FPGA-Files in den XILINX (Extension **.cfd*).

Die geladenen Daten sind flüchtig. Deshalb muss die Funktion nach Power Off erneut ausgeführt werden.

Nach Xilinx_Download ist eine Wartezeit von ca. 500 ms erforderlich, da der Controller ein Power-On-Reset durchläuft.

Anschließend ist der Firmware-Befehl 0x10 Software Reset auszuführen, um vom Bootloader-Modus in den Normal-Modus zu gelangen.

3.2.1.9 *Xilinx_Version*

Die Funktion GUSB_Platform_Xilinx_Version ermöglicht das Auslesen der geladenen XILINX-Firmwareversion.

Format:

```
int GUSB_Platform_Xilinx_Version(unsigned int DeviceName,
                                 unsigned int DeviceNumber,
                                 unsigned int *Version)
```

Parameter:

DeviceName

Typ des adressierten Gerätes (Nummer, die in *GUSB_Platform_def.h* deklariert ist, für USB 3080/ basicCAR 3080 = 6)

DeviceNumber

Nummer des adressiertenGerätes. Wenn mehrere Geräte gleichen Typs angeschlossen sind, erfolgt die Nummerierung in aufsteigender Reihenfolge der Seriennummern (das Gerät mit der NIEDRIGSTEN Seriennummer hat immer DeviceNumber 1).

Version

XILINX Softwareversion

Beschreibung:

Mit der Funktion GUSB_Platform_Xilinx_Version kann die Versionsnummer der im FPGA geladenen Software ausgelesen werden.

Beispiel:

Die Versionsnummer **2.34** wird als Wert **234** zurückgegeben, Version **2.60** als Wert **260**.

3.3 Programmieren mit LabVIEW

3.3.1 LabVIEW über G-API

Auf der mitgelieferten CD befindet sich eine VI-Sammlung, mit deren Hilfe USB 3080/ basicCAR 3080-Baugruppen unter LabVIEW angesprochen werden können.

Dabei nutzen die LabVIEW VIs die Funktionen der GÖPEL G-API.

3.3.2 LLB unter Verwendung des Windows Device Treibers

Auf der mitgelieferten CD befindet sich eine VI-Sammlung, mit deren Hilfe USB 3080/ basicCAR 3080-Baugruppen unter LabVIEW angesprochen werden können.

Dabei werden die Funktionen genutzt, die im Abschnitt Windows Device Treiber beschrieben worden sind.

3.4 Weitere GÖPEL Software

PROGRESS, Programm Generator und myCAR der GÖPEL electronic GmbH sind komfortable Programme zur Prüfung mit GÖPEL-Hardware.

Weitere Informationen zur Nutzung dieser Programme finden Sie in den entsprechenden Softwarebeschreibungen.

3.5 Steuerbefehle USB Controller

Der USB Controller ist für die Anbindung der USB 3080/ basicCAR 3080-Baugruppe an den PC über USB 2.0 zuständig.

An diesen USB Controller können Nachrichten (i. Allg. USB Befehle) gesendet werden, die für Konfigurationszwecke benötigt werden.

3.5.1 USB Befehlsaufbau

Ein USB Befehl besteht aus vier Bytes Header und den Daten (nicht alle USB Befehle benötigen Daten!).

Der Header eines USB Befehls ist folgendermaßen aufgebaut:

Bytenummer	Bedeutung	Inhalt
0	StartByte	0x23 (ASCII-Zeichen „#“)
1	Command	(0x..) Verwendete Codes entsprechend USB Befehle
2	reserviert	0x00
3	reserviert	0x00

3.5.2 USB Antwortaufbau

Genau wie der USB Befehl, ist auch die USB Antwort in vier Bytes Header und die Daten unterteilt (nicht alle USB Befehle senden Daten zurück!).

Der Header einer USB Antwort ist folgendermaßen aufgebaut:

Bytenummer	Bedeutung	Inhalt
0	StartByte	0x24
1	Command	(0x..) Verwendete Codes entsprechend USB Befehle
2	Length	Vom Befehl abhängige Länge
3	ErrorCode	Gibt den Fehlercode des Befehls zurück

3.5.3 USB Befehle

Gegenwärtig steht nur der USB Befehl READ_SW_VERSION zur Verfügung.

Command	Bezeichnung	Bedeutung
0x04	READ_SW_VERSION	Liefert die Version des USB Controllers Antwort: Byte 4: low Byte generic Softwareversion Byte 5: high Byte generic Softwareversion Byte 6: low Byte Softwareversion des funktionellen Teiles Byte 7: high Byte Softwareversion des funktionellen Teiles

C

F

G

L

S

U

W