Produktbeschreibung

USB 3052
basicCAN 3052

CAN Schnittstellen
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 3052/ basicCAN 3052 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 die 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 einer voll bestückten USB 3052 bzw. basicCAN 3052-Baugruppe (USB 3052) mit vier Controllern:

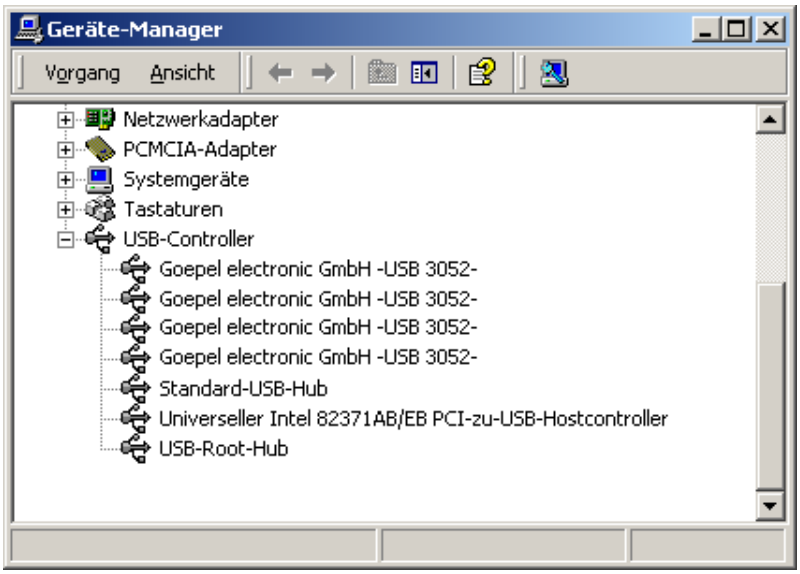

Abbildung 1-1: Anzeige Geräte-Manager

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

2 Hardware

2.1 Bestimmung

USB 3052 CAN Boards 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 3052 mit 4x CAN

Beachten Sie bitte, dass ein Download des Xilinx FPGAs für die Funktion eines USB 3052 Boards unabdingbar ist (siehe Xilinx_Download unter Windows Device Treiber)!

Zum Betrieb von USB 3052 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.

basicCAN 3052 ist ein GÖPEL electronic GmbH Stand-alone-Gerät auf der Grundlage eines USB 3052 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 z.B. in Kraftfahrzeugen.

Abbildung 2-2: basicCAN 3052

Die Stromversorgung mit 8-25 VDC (bei vier CAN Schnittstellen ca. 600 mA Ruhestrom bei 12 V) erfolgt über die beiden Buchsen für ext. Power Supply (rot = plus/ blau= minus) an der dem Steckverbinder für die CAN-Schnittstellen 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 CAN-Schnittstellen sind galvanisch vom USB-Interface und von der internen Logik getrennt.

Eigenschaften von USB 3052/ basicCAN 3052:

- 2 bis 4 CAN Schnittstellen Version 2.0b je nach Ausbaustufe
- Erweiterte Triggerfunktionen mit je einem Triggerinput und Triggeroutput zum Frontsteckverbinder bzw. zur Backplane
- Abschaltmöglichkeit der CAN Sendepfade ohne Verlust der Empfangsdaten, wenn kein CAN-Acknowledge empfangen wird (Firmwarebefehl 0x1C CAN Hardware-Sendepfad steuern)
- Galvanische Trennung der CAN Schnittstellen vom USB-Interface und von der internen Logik
- Jede CAN Schnittstelle verfügt über einen 32 Bit µController (TriCore TC1765, 40MHz)
- Visualisierung der Controllerzustände auf der Frontplatte mit je zwei LEDs (siehe LED Anzeige)
- Hohe Flexibilität durch steckbare Transceivermodule

In diesem Nutzerhandbuch ist unter Controller IMMER einer der jeder CAN Schnittstelle zugeordneten Microcontroller zu verstehen (unabhängig von der Bezeichnung „CAN Controller“ auf der Frontplatte eines USB 3052 Boards).

Sollten die Ressourcen eines basicCAN 3052 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 230V oder 115V Kaltgeräte-Anschluss an der Rückseite des Racks.

2.2 Technische Daten

2.2.1 Abmes-sungen

(Breite x Höhe x Tiefe):

- USB 3052: 4 TE x 130 mm x 185 mm
- basicCAN 3052: 126 mm x 51 mm x 183 mm

Die Angaben für USB 3052 beziehen sich auf Board im USB-Rack der GÖPEL electronic GmbH.

2.2.2 Kennwerte

Eine USB 3052/ basicCAN 3052-Baugruppe hat folgende Kennwerte:

Symbol	Kennwert	Min.	Typ.	Max.	Einheit	Bemerkung
Ext. Power Supply	Versorgungsspannung für interne Logik	8	12	25	V	
V_{BAT}	Batteriespannung		12	27/ 50	V	Abh. v. Transceivertyp
	Übertragungsrate			1	MBaud	CAN
R_{bus}	Abschlusswiderstand 1		120		Ohm	CAN Jumper gesteckt
R_{bus}	Abschlusswiderstände 2		5,1		kOhm	CAN Jumper gesteckt
	Externer Triggereingang	3,3		50	V	
	Externer Triggerausgang			V_{BAT}	V	Open Collector Ausgang über npn-Transistor: max. 50mA/ 200mW
V_{iso}	galvanische Trennung	560			V	CAN-Schnittstellen/ interne Logik und USB-Interface

Um am externen Triggerausgang (open Collector) der Schnittstelle einen Spannungshub zu erzeugen, muss ein externer Pullup-Widerstand über eine Spannungsquelle an diesen Ausgang geschaltet werden (z.B. 10kΩ über V_{Bat}).

Die zusätzlichen Triggereingänge können auch für die Einspeisung eines externen Taktsignals genutzt werden.

2.3 Aufbau

2.3.1 Allgemeines

USB 3052/ basicCAN 3052-Baugruppen verfügen in der Basisversion über zwei CAN Schnittstellen der Version 2.0b und können über Aufsatzboards und weitere Transceiver auf insgesamt vier CAN Schnittstellen erweitert werden.

Jede CAN Schnittstelle wird durch einen eigenen Microcontroller bedient.

Die USB-Informationen werden auf den Baugruppen an die Controller verteilt.

Bitte verwenden Sie zum Anschluss von USB 3052/ basicCAN 3052-Baugruppen an die USB-Schnittstelle des PCs die im Lieferumfang enthaltenen USB-Kabel.

Andere Kabel sind u. U. nicht geeignet!

2.3.2 Adressierung

Die Adressierung einzelner USB 3052/ basicCAN 3052-Baugruppen bei gleichzeitigem Betrieb mehrerer Baugruppen am gleichen Rechner erfolgt ausschließlich über die Seriennummern der CAN-Controller (siehe Ansteuersoftware): Der CAN-Controller mit der KLEINSTEN Seriennummer ist immer das Gerät Nummer 1.

Zur Erhöhung der Übersichtlichkeit empfehlen wir, USB 3052/ basicCAN 3052-Baugruppen in aufsteigender Reihenfolge der Seriennummern ihrer CAN-Controller im USB-Rack anzuordnen/ am Rechner anzuschließen.

2.3.3 Kommunikationsschnittstellen

Maximal 4 x CAN Schnittstellen Version 2.0b:

Für die uneingeschränkte Funktion einer CAN Schnittstelle in einem Netzwerk ist der verwendete Transceiver entscheidend. Häufig funktionieren CAN Netzwerke nur, wenn alle Teilnehmer kompatible Transceiver im Netz haben.

Damit die USB 3052/ basicCAN 3052-Nutzer 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 Transceiver verwendet, ist i. Allg. ein 120 Ohm Widerstand für jede CAN-Schnittstelle auf dem Board bestückt. Diese Widerstände können durch Ziehen der Jumper J1..J4 deaktiviert und durch bedrahtete Widerstände mit dem gewünschten Wert an den Positionen RP11, 12 bis RP41, 42 ersetzt werden (siehe Abbildung 2-3).

Bei Verwendung von Lowspeed Transceivern befinden sich zwei Abschlusswiderstände von 5,1 kOhm für RTH und RTL pro CAN Schnittstelle auf dem Transceiver-Modul. In diesem Fall darf KEIN bedrahteter Widerstand bestückt werden, und der Jumper muss geöffnet sein.

Für die folgenden Transceivertypen ist der Anschluss der Batteriespannung an die Pins 15, 18, 21 bzw. 24 des Steckverbinders XS1 (V_Bat1..V_Bat4, siehe Belegung Steckverbinder) für die jeweilige CAN-Schnittstelle notwendig:

- TJA1041A
- TJA1054
- PCA82C252
- B10011S

2.3.4 Bestückung

Abbildung 2-3 zeigt schematisch die Bestückungsseite eines USB 3052-Boards.
In dieser Abbildung ist die Lage der optionalen Aufsatzboards sowie der Transceivermodule für jede Schnittstelle zu erkennen. Die Positionen der Jumper J1..J4 für die Aktivierung/ Deaktivierung der Bus-Abschlusswiderstände bzw. der optional zu bestückenden bedrahteten Widerstände sind ebenfalls ersichtlich. Ein gesteckter Jumper bedeutet, dass der Bus-Abschlusswiderstand von 120 Ohm aktiv ist.

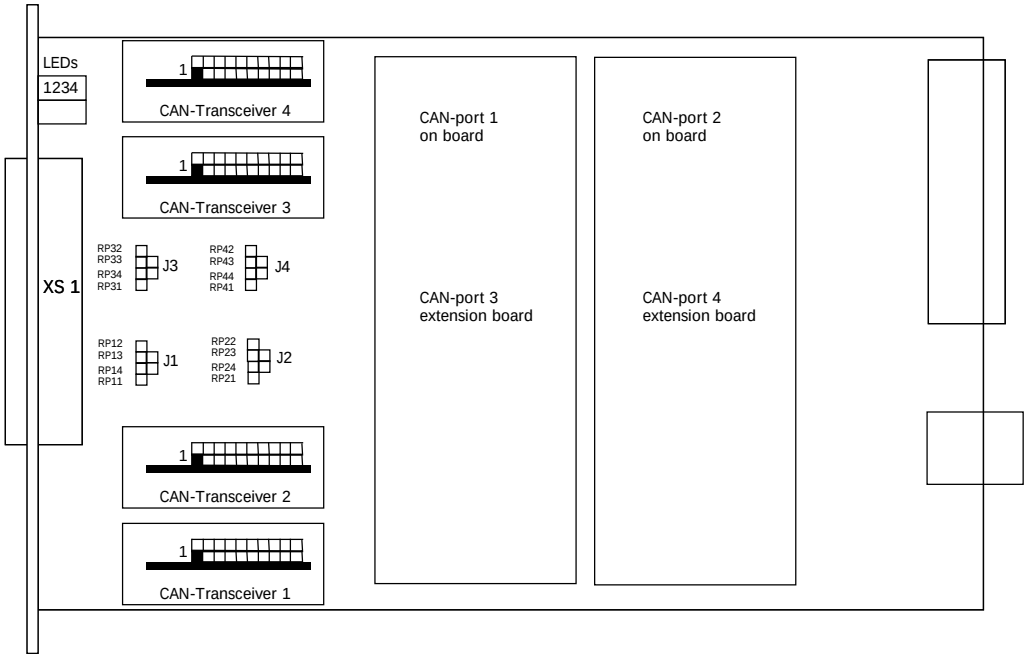

Abbildung 2-3: Schematischer Bestückungsplan eines USB 3052-Boards

Die Konfigurationselemente aus Abbildung 2-3 werden in der folgenden Tabelle erläutert:

CAN Transc.1	Transceivermodul für CAN1
CAN Transc.2	Transceivermodul für CAN2
CAN Transc.3	Transceivermodul für CAN3
CAN Transc.4	Transceivermodul für CAN4
J1	Jumper zum Aktivieren des **120Ω** Busabschlusswiderstandes (onboard) für CAN1
J2	Jumper zum Aktivieren des **120Ω** Busabschlusswiderstandes (onboard) für CAN2
J3	Jumper zum Aktivieren des **120Ω** Busabschlusswiderstandes (onboard) für CAN3
J4	Jumper zum Aktivieren des **120Ω** Busabschlusswiderstandes (onboard) für CAN4
RP 11, 12	Position für optionalen bedrahteten Abschlusswiderstand – CAN1
RP 21, 22	Position für optionalen bedrahteten Abschlusswiderstand – CAN2
RP 31, 32	Position für optionalen bedrahteten Abschlusswiderstand – CAN3
RP 41, 42	Position für optionalen bedrahteten Abschlusswiderstand – CAN4

2.3.5 Belegung Steckverbinder

Typ: DSub 25-polig Buchse

Die CAN Schnittstellen stehen über diesen Steckverbinder mit folgender Belegung zur Verfügung:

lfd. Nr.	Anschluss XS1	Signalname	Bemerkung
1	14	CAN1_High	CAN-Bus-Leitung High
2	2	CAN1_Low	CAN-Bus-Leitung Low
3	15	V_Bat1	Versorgungsspannungseingang für Transceiver 1
4	1	GND	Massepotenzial Transceiver/ Trigger
5	17	CAN2_High	CAN-Bus-Leitung High
6	5	CAN2_Low	CAN-Bus-Leitung Low
7	18	V_Bat2	Versorgungsspannungseingang für Transceiver 2
8	4	GND	Massepotenzial Transceiver/ Trigger
9	20	CAN3_High	CAN-Bus-Leitung High
10	8	CAN3_Low	CAN-Bus-Leitung Low
11	21	V_Bat3	Versorgungsspannungseingang für Transceiver 3
12	7	GND	Massepotenzial Transceiver/ Trigger
13	23	CAN4_High	CAN-Bus-Leitung High
14	11	CAN4_Low	CAN-Bus-Leitung Low
15	24	V_Bat4	Versorgungsspannungseingang für Transceiver 4
16	10	GND	Massepotenzial Transceiver/ Trigger
17	3	INPUT1	Triggereingang 1
18	16	OUTPUT1	Triggerausgang 1
19	6	INPUT2	Triggereingang 2
20	19	OUTPUT2	Triggerausgang 2
21	9	INPUT3	Triggereingang 3
22	22	OUTPUT3	Triggerausgang 3
23	12	INPUT4	Triggereingang 4
24	25	OUTPUT4	Triggerausgang 4
25	13	GND	Massepotenzial Transceiver/ Trigger

USB-Schnittstelle

Die USB-B-Buchse (USB-Standardbelegung) für das USB 2.0 Interface befindet sich an der dem Steckverbinder für die CAN Schnittstellen gegenüber liegenden Seite eines USB 3052-Boards.

2.3.6 LED Anzeige

Die auf der Frontplatte des USB 3052-Boards angeordneten Leuchtdioden geben Auskunft über den momentanen Betriebszustand des jeder CAN Schnittstelle (auch als „CAN Port“ bezeichnet) zugeordneten Controllers. Einer Schnittstelle sind je eine grüne und rote LED zugeteilt, die wie folgt angeordnet sind:

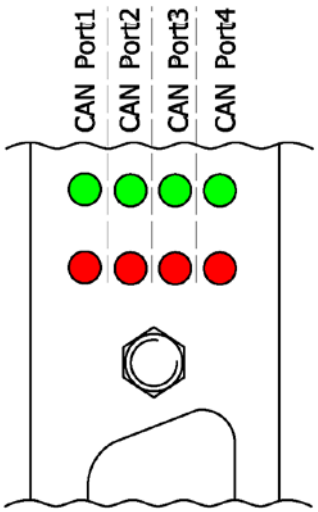

Abbildung 2-4: LED Anzeige

Anzeigezustände dieser LEDs:

LED grün	LED rot	Bemerkung
leuchten dauerhaft		Controller läuft nicht Wahrscheinliche Fehlerursache: Xilinx_download nicht erfolgt
blinken abwechselnd		Bootloadersoftware läuft Wahrscheinliche Fehlerursache: Softwarereset nicht erfolgt
leuchten nicht		Firmware läuft
leuchtet (kurz)	leuchtet nicht	Firmware läuft und arbeitet Befehl ab

2.4 Lieferhinweise

USB 3052/ basicCAN 3052-Baugruppen werden in folgenden Varianten geliefert:

- 2x CAN Schnittstelle (Basisvariante)
- 3x CAN Schnittstelle
- 4x CAN Schnittstelle

Zur jeweiligen Schnittstelle muss auch der Typ des zugehörigen CAN Transceivers festgelegt werden.

Für jede CAN Schnittstelle sind außerdem die erforderlichen Funktionalitäten anzugeben.

Zum Betrieb von USB 3052 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.

3 Ansteuersoftware

Zur Einbindung der USB 3052/ basicCAN 3052-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 der jeweiligen CAN-Schnittstelle zugeordnete Microcontroller einer USB 3052/ basicCAN 3052 Baugruppe zu verstehen.

Jedem dieser Controller ist ein eigener USB Controller zugeordnet, der das USB 2.0-Interface 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

Zuordnung der USB Controller zu einer USB 3052/ basicCAN 3052 Baugruppe

Eine USB 3052/ basicCAN 3052 Baugruppe erscheint mit zwei bis vier USB Geräten im Gerätemanager von Windows, da jeder Controller (und somit jeder CAN Knoten) einen eigenen USB Controller besitzt (siehe Abbildung 1-1 im Abschnitt Treiberinstallation).

Um diese USB Geräte der/ den USB 3052/ basicCAN 3052-Baugruppe(n) und deren Controllern zuordnen zu können, sind zunächst die Seriennummern über den Befehl Driver_Info zu ermitteln.

Die Zuordnung (u.a. die DeviceNumber) wird durch den Rest aus der ganzzahligen Division der Seriennummern durch die Zahl 4 bestimmt. D.h., die jeweilige Seriennummer ist durch 4 zu teilen, aber ohne Kommastellen (Modulo, mathematisches Formelzeichen mod).

Es gilt folgende Regel:

Seriennummer mod 4	Controller
0	1
1	2
2	3
3	4

Beispiel 1: eine USB 3052/ basicCAN 3052-Baugruppe mit 4 Controllern:

Seriennummer	Controller	Baugruppe	DeviceNumber
20070040	1	1	1
20070041	2	1	2
20070042	3	1	3
20070043	4	1	4

Beispiel 2: zwei USB 3052/ basicCAN 3052-Baugruppen mit je 2 Controllern:

Seriennummer	Controller	Baugruppe	DeviceNumber
20070080	1	1	1
20070081	2	1	2
20070084	1	2	3
20070085	2	2	4

Werden bei einer Anordnung nach Beispiel 2 z.B. bei Baugruppe 1 weitere Controller nachgerüstet, ändert sich die Nummerierung bei DeviceNumber wie folgt:

Beispiel 3: zwei USB 3052/ basicCAN 3052-Baugruppen mit vier bzw. zwei Controllern:

Seriennummer	Controller	Baugruppe	DeviceNumber
20070080	1	1	1
20070081	2	1	2
20070082	3	1	3
20070083	4	1	4
20070084	1	2	5
20070085	2	2	6

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 eineDatenstruktur
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 3052/ basicCAN 3052 = 4)

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).

Beachten Sie bitte die Hinweise unter Zuordnung der USB Controller ... im Abschnitt Windows Device Treiber: Die Anzahl der „Geräte“ und damit der DeviceNumbers entspricht der Anzahl der vorhandenen CAN Schnittstellen.

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 3052/ basicCAN 3052 = 4)

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).

Beachten Sie bitte die Hinweise unter Zuordnung der USB Controller ... im Abschnitt Windows Device Treiber: Die Anzahl der „Geräte“ und damit der DeviceNumbers entspricht der Anzahl der vorhandenen CAN Schnittstellen.

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 3052/ basicCAN 3052 = 4)

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).

Beachten Sie bitte die Hinweise unter Zuordnung der USB Controller ... im Abschnitt Windows Device Treiber: Die Anzahl der „Geräte“ und damit der DeviceNumbers entspricht der Anzahl der vorhandenen CAN Schnittstellen.

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 3052/ basicCAN 3052 = 4)

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).

Beachten Sie bitte die Hinweise unter Zuordnung der USB Controller ... im Abschnitt Windows Device Treiber: Die Anzahl der „Geräte“ und damit der DeviceNumbers entspricht der Anzahl der vorhandenen CAN Schnittstellen.

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 3052/ basicCAN 3052 = 4)

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).

Beachten Sie bitte die Hinweise unter Zuordnung der USB Controller ... im Abschnitt Windows Device Treiber: Die Anzahl der „Geräte“ und damit der DeviceNumbers entspricht der Anzahl der vorhandenen CAN Schnittstellen.

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.

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.

Diese Funktion kann nur auf dem USB Controller des ERSTEN Controllers einer USB 3052/ basicCAN 3052 Baugruppe ausgeführt werden.

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 3052/ basicCAN 3052 = 4)

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).

Beachten Sie bitte die Hinweise unter Zuordnung der USB Controller ... im Abschnitt Windows Device Treiber: Die Anzahl der „Geräte“ und damit der DeviceNumbers entspricht der Anzahl der vorhandenen CAN Schnittstellen.

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 die Controller ein Power-On-Reset durchlaufen. 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 3052/ basicCAN 3052 = 4).

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 3052/ basicCAN 3052-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 3052/ basicCAN 3052-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

Die USB Controller sind für die Anbindung der USB 3052/ basicCAN 3052 Baugruppe an den PC über USB 2.0 zuständig.

An diese 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

I

L

S

U

W