Produktbeschreibung

USB 3132
basicCON 3132

Relaisschaltmodule
Nutzerhandbuch Version 1.1

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: 31.08.2011

Stand: August 2011

1 Installation

1.1 Hardwareinstallation

Wir empfehlen, die Gerätetreiber-Software vor dem Anschließen der Baugruppen an den PC/ Laptop zu installieren (siehe Abschnitt Treiberinstallation).

USB 3132:

Das USB 3132-Board kann nur in einem der GÖPEL electronic USB-Racks USB 1004, USB 1008 oder USB 1016 betrieben werden.

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

Wählen Sie einen freien Steckplatz in Ihrem USB-Rack aus. Falls vorhanden, muss zuerst das Slotblech entfernt werden, das den Steckplatz abdeckt. Dazu sind die beiden Schrauben zu lösen. Führen Sie das Board über die Führungsschienen vorsichtig in den vorbereiteten Steckplatz ein und drücken Sie es das letzte Stück, mit etwas Kraft, bis zum Anschlag in den Steckplatz hinein. Schrauben Sie die beiden äußeren, an der Frontplatte befindlichen Schrauben fest, damit das Board einen sicheren Sitz hat.

Fassen Sie das Board bei der Montage nur an den Rändern oder an der Frontplatte an. Berühren Sie niemals die Oberfläche, da sonst die Gefahr der Zerstörung von Bauteilen durch elektrostatische Entladung besteht.

Zur Entfernung des Boards aus dem Rack (falls notwendig), sind die beiden äußeren Schrauben wieder zu lösen. Mit dem an der Frontplatte befindlichen Hebel kann das Board aus dem Steckplatz herausgelöst und anschließend herausgezogen werden.

Bitte vergleichen Sie das Kapitel Adressierung zur Installation mehrerer USB 3132-Boards.

basicCON 3132:

Das basicCON 3132 kann direkt an den PC oder Laptop angeschlossen werden.

Verbinden Sie die auf der Rückseite befindliche USB-Buchse über das beigelegte USB-Kabel mit Ihrem Rechner.

Schließen Sie das beigelegte Netzteil oder eine eigene Spannungsquelle an die entsprechenden Anschlüsse ext. Power Supply auf der Rückseite des Gehäuses an (siehe Kapitel Stromversorgung).

Die Anschlüsse auf der Frontseite der USB 3132-Boards werden im Kapitel Belegung Frontsteckverbinder näher erläutert.

1.2 Treiberinstallation

Um die GÖPEL electronic USB-Treiber auf Ihrem System einzurichten, empfehlen wir das GUSB Treiber Setup.
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 Gerätetreiber unterstützt gegenwärtig ausschließlich Windows® 2000/ XP-Systeme!

Wenn Sie eigene Software für die Boards erstellen wollen, benötigen Sie die Dateien für die anwenderspezifische Programmierung (**.DLL*, **.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).

Durch die Plug-und-Play Fähigkeit von Windows® 2000/ XP wird das Gerät automatisch vom Betriebssystem erkannt.

Am Ende des Installationsprozesses werden Sie von Windows® aufgefordert, Ihren Rechner neu zu starten. Für einen sicheren und zuverlässigen Betrieb wird ein Neustart des Systems unbedingt empfohlen.

Nach der Treiberinstallation/ Hardwareinstallation können Sie überprüfen, ob die Boards einwandfrei vom System eingebunden wurden.
Die folgende Abbildung zeigt u.a. die erfolgreiche Einbindung eines USB 3132-Boards:

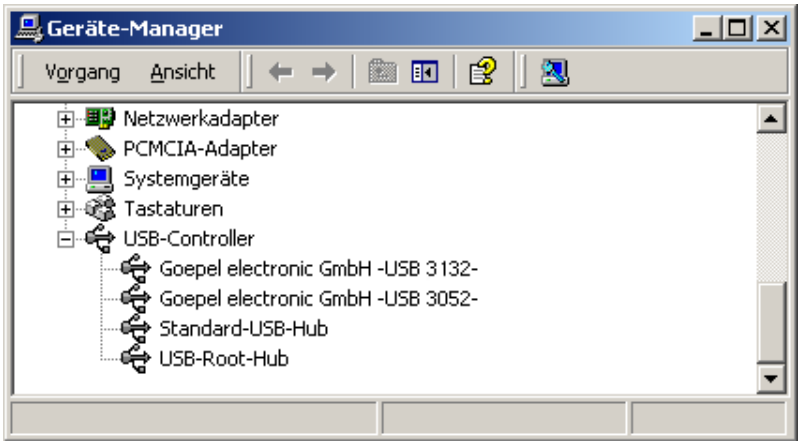

Abbildung 1-1: Anzeige Geräte-Manager

Beachten Sie bitte, dass der Geräte-Manager ALLE USB-Controller anzeigt, die von diesem Treiber unterstützt werden.

2 Hardware

2.1 Bestimmung

USB 3132 ist ein Relaisboard mit USB 2.0-Interface der GÖPEL electronic GmbH.

Dieses Board wird in der allgemeinen Steuerungs- und Prüftechnik, u.a. in der Automobiltechnik, zum Schalten großer Lasten verwendet.

Das Board hat 32 voneinander unabhängige Relais mit Normally Open Charakteristik.

Abbildung 2-1: Relaisboard USB 3132

Optional kann das USB 3132 Board auch in der Ausführung mit vier 8 zu 1 Relaismultiplexern geliefert werden.

Beachten Sie bitte, dass ein Download des Xilinx FPGAs für die Funktion der Boards unabdingbar ist (siehe Xilinx Download unter Programmieren über DLL-Funktionen)!

Zum Betrieb von USB 3132 Boards ist ein GÖPEL electronic USB-Rack USB 1004, USB 1008 oder USB 1016 erforderlich, das bis zu 16 GÖPEL electronic USB-Boards aufnehmen kann. Die Stromversorgung erfolgt in diesem Fall über das im Rack eingebaute Netzteil.

basicCON 3132 ist ein GÖPEL electronic GmbH stand-alone Gerät auf der Grundlage eines USB 3132 Relaisboards zum Anschluss an einen PC oder Laptop, das für den eigenständigen Einsatz außerhalb komplexer Testsysteme entwickelt wurde.
Die externe Spannungszufuhr von 7-25 VDC erlaubt die Nutzung dieses Gerätes zum Schalten elektrischer Signale bei beliebigen Anwendungen.

Abbildung 2-2: basicCON 3132

An der Geräterückseite befinden sich die folgenden Anschlüsse:

Abbildung 2-3: Geräterückseite

- USB-B-Buchse für das USB 2.0 Interface mit USB-Standardbelegung
- DC-Buchse für das mitgelieferte Steckernetzteil
- Bananenbuchsen zur Stromversorgung

Bitte nutzen Sie für die externe Stromversorgung entweder die beiden Bananenbuchsen ODER die DC-Buchse.

Zur Stromversorgung vergleichen Sie bitte die Hinweise im entsprechenden Kapitel.

2.2 Technische Daten

2.2.1 Abmessungen

(Breite x Höhe x Tiefe):

- USB 3132: 4 TE x 130 mm x 185 mm
- basicCON 3132: 120 mm x 50 mm x 180 mm

Die Angaben für USB 3132 beziehen sich auf ein Board im GÖPEL electronic USB-Rack.

2.2.2 Kennwerte

Ein Relaisboard USB 3132 hat folgende Kennwerte:

Symbol	Kennwert	Min.	Typ.	Max.	Einheit	Bedingung
I_S	Schaltstrom DC (Nom.)	$10*10^{-6}$		1	A	Ohmsche Last; 30V
U_S	Schaltspannung DC	10mV		30	V	Ohmsche Last; 1A
P_S	Schaltleistung			30	W	Ohmsche Last
$R_{contact}$	Kontaktwiderstand		50	100	mΩ	Pro Relaiskontakt
N	Schaltspiele (elektrisch)	10^5				
t_{on}	Anzugszeit		4	10	ms	
t_{off}	Abfallzeit		4	10	ms	
U_{EXT}	Externe Betriebsspannung	7	12	25	V	

Warnung

Bitte verwenden Sie zum Anschluss der Baugruppe an die USB-Schnittstelle des PCs/ Laptops das im Lieferumfang enthaltene USB-Kabel. Andere Kabel sind u. U. nicht geeignet!

2.2.3 Adressierung

Die Adressierung von USB 3132-Boards im GÖPEL electronic USB-Rack oder basicCON 3132 Geräten erfolgt ausschließlich über deren Seriennummern (siehe Ansteuersoftware): Das Board/ Gerät mit der KLEINSTEN Seriennummer hat immer die DeviceNumber 1.

Tip

Zur Erhöhung der Übersichtlichkeit empfehlen wir, im Falle mehrerer USB 3132-Boards diese in aufsteigender Reihenfolge ihrer Seriennummern im USB-Rack anzuordnen.

2.2.4 Stromversorgung

Das Board USB 3132 wird über das USB-Rack versorgt, in dem es installiert worden ist.

basicCON 3132 wird extern über die beiden Buchsen für ext. Power Supply (rot = plus/ blau = minus) mit 7-25VDC versorgt (ca. 500mA bei 12V).

Stattdessen kann auch die Buchse für das beigelegte Steckernetzteil mit dem Hohlstecker (2,1 x 5,5mm/ Polarität + innen) genutzt werden, siehe Abbildung 2-3.

2.3 Aufbau

2.3.1 Allgemeines

USB 3132 ist zum potenzialfreien Schalten von Lasten bzw. Steuer- und Messsignalen einsetzbar. Das Board verfügt über 32 Relais, die als Schließer benutzt werden. Die Anschlüsse der jeweiligen Relais sind auf den Steckverbinder XS1 geführt.
Optional ist durch Bestücken der Widerstände R7..R38 auch eine Realisierung von vier 8 zu 1 Relaismultiplexern möglich (siehe Abbildung 2-5).

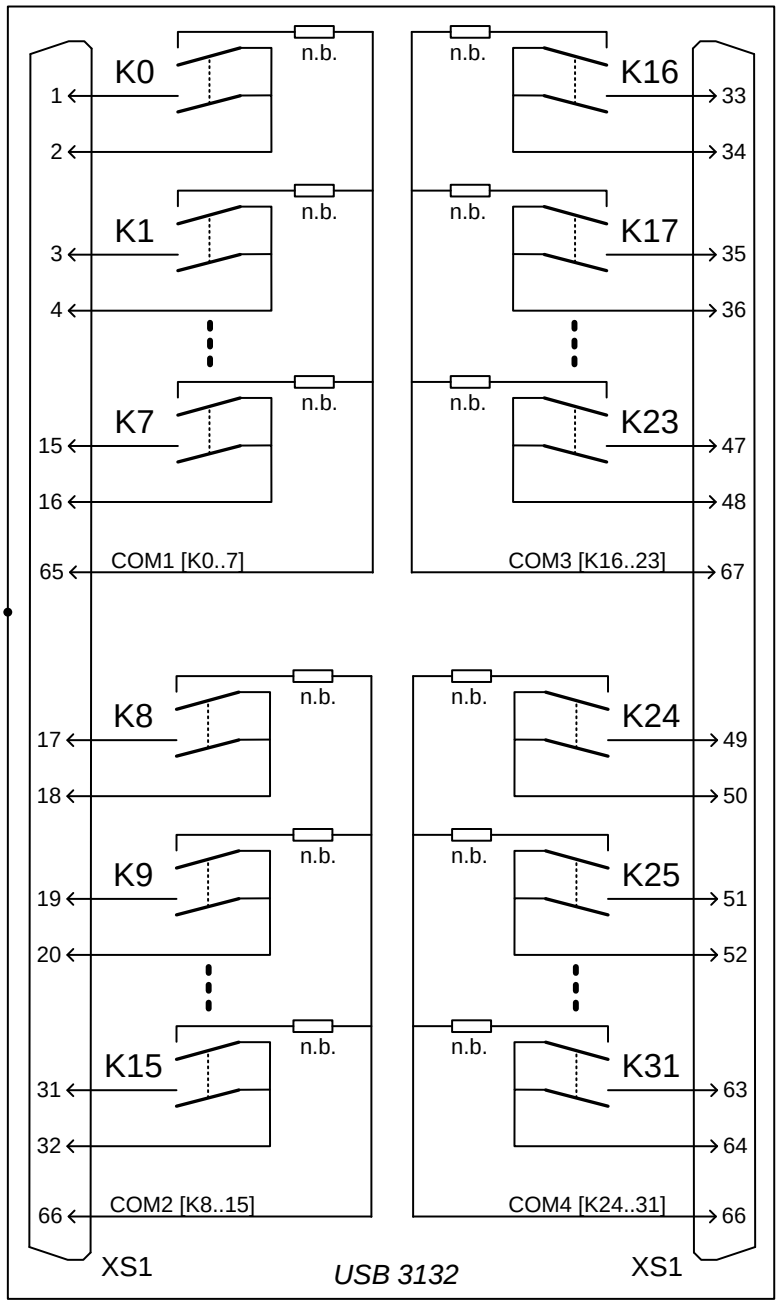

Abbildung 2-4: USB 3132

Abbildung 2-4 zeigt eine schematische Darstellung von USB 3132.

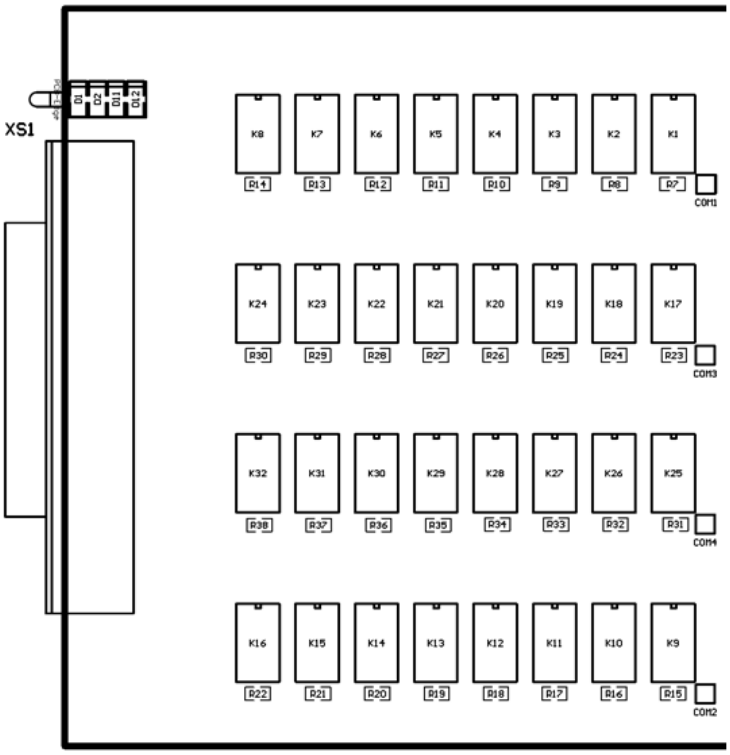

Abbildung 2-5: Positionen Relais/ Widerst.

In Abbildung 2-5 ist die Lage der Relais und Widerstände zur Multiplexerverschaltung dargestellt.

2.3.2 Front-LEDs

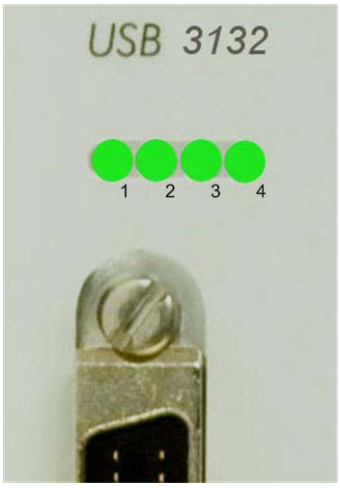

Abbildung 2-6: Frontplatte USB 3132

LED 1	LED 2	LED 3	LED 4	Bemerkung
leuchtet	aus			FPGA wird gerade konfiguriert (Xilinx Download)
aus	leuchtet	aus		FPGA ist konfiguriert (Xilinx Download abgeschlossen)
aus				Normaler Betriebszustand

2.3.3 Belegung Front-steckverbinder

Typ am Gerät: SCSI-Steckverbinder Stiftleiste 68-polig

Typ am Kabel: SCSI -Steckverbinder Buchsenleiste 68-polig

Die Belegung des Frontsteckverbinders ist in der folgenden Tabelle dargestellt:

Anschluss XS1	COM Widerstand	Signalname	Bemerkung
1	R7	Ch0_H	Relais K1 - Kontakt „Common“
2		Ch0_L	Relais K1 - Kontakt „Normally Open“
3	R8	Ch1_H	Relais K2 - Kontakt „Common“
4		Ch1_L	Relais K2 - Kontakt „Normally Open“
5	R9	Ch2_H	Relais K3 - Kontakt „Common“
6		Ch2_L	Relais K3 - Kontakt „Normally Open“
7	R10	Ch3_H	Relais K4 - Kontakt „Common“
8		Ch3_L	Relais K4 - Kontakt „Normally Open“
9	R11	Ch4_H	Relais K5 - Kontakt „Common“
10		Ch4_L	Relais K5 - Kontakt „Normally Open“
11	R12	Ch5_H	Relais K6 - Kontakt „Common“
12		Ch5_L	Relais K6 - Kontakt „Normally Open“
13	R13	Ch6_H	Relais K7 - Kontakt „Common“
14		Ch6_L	Relais K7 - Kontakt „Normally Open“
15	R14	Ch7_H	Relais K8 - Kontakt „Common“
16		Ch7_L	Relais K8 - Kontakt „Normally Open“
17	R15	Ch8_H	Relais K9 - Kontakt „Common“
18		Ch8_L	Relais K9 - Kontakt „Normally Open“
19	R16	Ch9_H	Relais K10 - Kontakt „Common“
20		Ch9_L	Relais K10 - Kontakt „Normally Open“
21	R17	Ch10_H	Relais K11 - Kontakt „Common“
22		Ch10_L	Relais K11 - Kontakt „Normally Open“
23	R18	Ch11_H	Relais K12 - Kontakt „Common“
24		Ch11_L	Relais K12 - Kontakt „Normally Open“
25	R19	Ch12_H	Relais K13 - Kontakt „Common“
26		Ch12_L	Relais K13 - Kontakt „Normally Open“
27	R20	Ch13_H	Relais K14 - Kontakt „Common“
28		Ch13_L	Relais K14 - Kontakt „Normally Open“
29	R21	Ch14_H	Relais K15 - Kontakt „Common“
30		Ch14_L	Relais K15 - Kontakt „Normally Open“
31	R22	Ch15_H	Relais K16 - Kontakt „Common“
32		Ch15_L	Relais K16 - Kontakt „Normally Open“
33	R23	Ch16_H	Relais K17 - Kontakt „Common“
34		Ch16_L	Relais K17 - Kontakt „Normally Open“
35	R24	Ch17_H	Relais K18 - Kontakt „Common“
36		Ch17_L	Relais K18 - Kontakt „Normally Open“
37	R25	Ch18_H	Relais K19 - Kontakt „Common“
38		Ch18_L	Relais K19 - Kontakt „Normally Open“
39	R26	Ch19_H	Relais K20 - Kontakt „Common“
40		Ch19_L	Relais K20 - Kontakt „Normally Open“

41	R27	Ch20_H	Relais K21 - Kontakt „Common“
42		Ch20_L	Relais K21 - Kontakt „Normally Open“
43	R28	Ch21_H	Relais K22 - Kontakt „Common“
44		Ch21_L	Relais K22 - Kontakt „Normally Open“
45	R29	Ch22_H	Relais K23 - Kontakt „Common“
46		Ch22_L	Relais K23 - Kontakt „Normally Open“
47	R30	Ch23_H	Relais K24 - Kontakt „Common“
48		Ch23_L	Relais K24 - Kontakt „Normally Open“
49	R31	Ch24_H	Relais K25 - Kontakt „Common“
50		Ch24_L	Relais K25 - Kontakt „Normally Open“
51	R32	Ch25_H	Relais K26 - Kontakt „Common“
52		Ch25_L	Relais K26 - Kontakt „Normally Open“
53	R33	Ch26_H	Relais K27 - Kontakt „Common“
54		Ch26_L	Relais K27 - Kontakt „Normally Open“
55	R34	Ch27_H	Relais K28 - Kontakt „Common“
56		Ch27_L	Relais K28 - Kontakt „Normally Open“
57	R35	Ch28_H	Relais K29 - Kontakt „Common“
58		Ch28_L	Relais K29 - Kontakt „Normally Open“
59	R36	Ch29_H	Relais K30 - Kontakt „Common“
60		Ch29_L	Relais K30 - Kontakt „Normally Open“
61	R37	Ch30_H	Relais K31 - Kontakt „Common“
62		Ch30_L	Relais K31 - Kontakt „Normally Open“
63	R38	Ch31_H	Relais K32 - Kontakt „Common“
64		Ch31_L	Relais K32 - Kontakt „Normally Open“
65		COM1	Relais K1..K8
66		COM2	Relais K9..K16
67		COM3	Relais K17..K24
68		COM4	Relais K25..K32

2.4 Lieferhinweise

USB 3132 (basicCON 3132 entsprechend) werden in folgenden Grundkonfigurationen geliefert:

- USB 3132.10 Board mit 32 Einzelrelais
- USB 3132.20 Board mit 4x 8 zu 1 Multiplexern

Falls Sie andere Konfigurationen benötigen
(z.B. 16 Einzelrelais/ 2x 8 zu 1 Multiplexer),
setzen Sie sich bitte mit unserem Vertrieb in Verbindung

Alle USB 3132 Board Varianten sind auch als stand-alone Variante basicCON 3132 erhältlich.

3 Ansteuersoftware

3.1 Programmieren über DLL-Funktionen

Mit den nachfolgend beschriebenen Funktionsaufrufen können USB 3132-Boards oder basicCON 3132-Geräte direkt aus diversen Hochsprachen angesprochen werden (VisualC++, CVI).

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

Informationen zu den Strukturen, Funktionen und Error-Codes enthält das C-Header File *GUSB_Platform.h* auf der mitgelieferten CD.

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
- Xilinx_Download
- Xilinx_Version
- Write_COMMAND
- Read_COMMAND
- SetRelay
- SetRelayMask
- GetRelay
- UpdateRelay

3.1.1 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 (Speicherbereich)
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 eines Zeigers pDriverInfo übergeben werden. Mit Hilfe des Parameters LengthInByte prüft die Funktion intern den korrekt initialisierten Anwenderspeicher.

Die Funktion füllt die 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 sind für die weitere Nutzung der USB-Hardware zwingend erforderlich.

3.1.2 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.

3.1.3 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 3132 = 15)

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.

3.1.4 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 3132 = 15)

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).

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.

3.1.5 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 3132 = 15)

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.

3.1.6 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 3132 = 15)

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)

DataLength

Vor Funktionsaufruf: Anzugebende Größe des Lesepuffers in Bytes
Nach Funktionsausführung: Anzahl der tatsächlich gelesenen Bytes
Siehe auch Steuerbefehle USB Controller
(z. Zt. min. 64 Byte pro Antwort)

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!

3.1.7 SetRelay

Die Funktion GUSB_Platform_3132_SetRelay konfiguriert die Sollkonfiguration der Relais des mit DeviceNumber indizierten USB 3132 Boards.

Format:

```
int GUSB_Platform_3132_SetRelay
   (unsigned int DeviceNumber,
    unsigned int RelVal_1_32,
    unsigned int HandlingControl);
```

Parameter:

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).

RelVal_1_32

Relaybits 1..32 (32 Bit Wert, Bit-orientiert)

HandlingControl (siehe auch unter Beschreibung weiter unten)

32 Bit Wert, 0 = Init, 1 = Normal

Beschreibung:

Die Funktion führt bei bei Übergabe des Wertes 0 für HandlingControl eine Initialisierung des Board-internen Relaishandlings aus, das nach jedem Powerup des Boards vor dem ersten Schalten erfolgen muss. Der Parameter RelVal_1_32 wird hierbei NICHT beachtet.

Mit HandlingControl = 1 konfiguriert die Funktion die Sollstellung ALLER 32 Relais mittels eines 32 Bit-Wertes.
Jedes Relaybit steht für die Sollstellung eines Relais. Zum Setzen des jeweiligen Relais muss das entsprechende Bit mit 1 und zum Rücksetzen des Relais mit 0 übergeben werden.

Das physikalische Schalten der Relais auf dem USB 3132 Board gemäß dieser Sollkonfiguration erfolgt mit dem Befehl UpdateRelay.

3.1.8 SetRelay-Mask

Die Funktion GUSB_Platform_3132_SetRelayMask konfiguriert die Sollkonfiguration der maskierten Relaisbits des mit DeviceNumber indizierten USB 3132-Boards.

Damit ist das Schalten einzelner Relais möglich.

Die Ausführung dieser Funktion ist erst nach Ausführung der Funktion SetRelay mit HandlingControl = 0 möglich.

Format:

```
int GUSB_Platform_3132_SetRelayMask
   (unsigned int DeviceNumber,
    unsigned int RelVal_1_32,
    unsigned int RelValMask_1_32);
```

Parameter:

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).

RelVal_1_32

Relaybits 1..32, 32 Bit Wert, Bit-orientiert

RelValMask_1_32

Maskenbits 1..32, 32 Bit Wert, Bit-orientiert

Beschreibung:

Die Funktion trägt den 32 Bit Wert in die Sollkonfiguration der Relais 1..32 mit der Einschränkung ein, dass nur die Bits der Soll-konfiguration geändert werden, deren zugehörige Maskenbits auf 1 stehen. D.h., die entsprechenden Registerwerte werden gesetzt.

Das physikalische Schalten der Relais auf dem USB 3132 Board gemäß dieser Sollkonfiguration erfolgt mit dem Befehl UpdateRelay.

3.1.9 GetRelay

Die Funktion GUSB_Platform_3132_GetRelay gibt die im Board programmierte Sollkonfiguration der Relais des mit DeviceNumber indizierten USB 3132 Boards zurück.

Format:

```
int GUSB_Platform_3132_GetRelay
   (unsigned int DeviceNumber,
    unsigned int *RelVal);
```

Parameter:

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).

RelVal

Zeiger auf eine Variable

Zur Struktur siehe das File *GUSB_Platform.h* auf der mitgelieferten CD

Beschreibung:

Der Bitwert ergibt den aktuellen Inhalt des Sollkonfigurationsegisters.

Beachten Sie bitte, dass mit dieser Funktion NICHT die tatsächlichen physikalischen Zustände der Relais ermittelt werden.

3.1.10 Update-Relay

Die Funktion GUSB_Platform_3132_UpdateRelay schaltet die Relais 1..32 des mit DeviceNumber indizierten USB 3132 Boards wie in der Sollkonfiguration angegeben.

Format:

```
int GUSB_Platform_3132_UpdateRelay
   (unsigned int DeviceNumber);
```

Parameter:

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).

Beschreibung:

Nach Ausführung dieser Funktion entspricht der tatsächliche Schaltzustand der Relais den Sollkonfigurationswerten.

Ein in der Sollkonfiguration auf den Wert 1 gesetztes Relaybit bedeutet Setzen des entsprechenden Relais, während der Wert 0 Rücksetzen bedeutet.

3.2 Programmieren mit LabVIEW

LLB unter Verwendung des Windows Device Treibers

In den Dateien *GUSB_Platform.llb* und *GUSB_Platform 3132.llb* der mitgelieferten CD befinden sich VIs, mit deren Hilfe USB 3132-Boards oder basicCON 3132-Geräte direkt unter LabVIEW angesprochen werden können.

Dabei werden die Funktionen genutzt, die im Abschnitt Programmieren über DLL-Funktionen beschrieben worden sind.

3.3 Steuerbefehle USB Controller

Der USB Controller ist für die Anbindung der USB 3132 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.

3.3.1 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

3.3.2 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

3.3.3 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