DEDITEC GMBH

USB-TTL-32 / USB-TTL-64

Hardware-Beschreibung

2011 November

INDEX

INDEX

INDEX

Einleitung

1. Einleitung

1.1. Vorwort

Wir beglückwünschen Sie zum Kauf eines hochwertigen DEDITEC Produktes!

Unsere Produkte werden von unseren Ingenieuren nach den heutigen geforderten Qualitätsanforderungen entwickelt. Wir achten bereits bei der Entwicklung auf flexible Erweiterbarkeit und lange Verfügbarkeit.

Wir entwickeln modular!
Durch eine modulare Entwicklung verkürzt sich bei uns die Entwicklungszeit und - was natürlich dem Kunden zu Gute kommt - ein fairer Preis!

Wir sorgen für eine lange Lieferverfügbarkeit!
Sollten verwendete Halbleiter nicht mehr verfügbar sein, so können wir schneller reagieren. Bei uns müssen meistens nur Module redesigned werden und nicht das gesamte Produkt. Dies erhöht die Lieferverfügbarkeit.

1.2. Kundenzufriedenheit

Ein zufriedener Kunde steht bei uns an erster Stelle!
Sollte mal etwas nicht zu Ihrer Zufriedenheit sein, wenden Sie sich einfach per Telefon oder mail an uns.
Wir kümmern uns darum!

1.3. Kundenresonanz

Die besten Produkte wachsen mit unseren Kunden. Für Anregungen oder Vorschläge sind wir jederzeit dankbar.

Hardware Beschreibung

2. Hardware Beschreibung

2.1. Einführung

Das USB-TTL-32 und das USB-TTL-64 finden dort Einsatz, wo über den USB-Bus direkt auf TTL-Ein- bzw. Ausgänge zugegriffen werden soll. Durch die mitgelieferte Treiberbibliothek ist ein einfaches Ansprechen der Module möglich.

Die TTL-Ein/Ausgänge können in 8-er Blöcken als Ein- oder Ausgang per Software programmiert werden.

2.2. Kurzanleitung Installation

2.2.1. Schritt 1 - Installation der Software und Treiber

Installieren Sie nun die DELIB-Treiberbibliothek mit der Datei "delib_install.exe" von der im Lieferumfang enthaltenen DEDITEC-Treiber CD.

Diese finden Sie im Verzeichnis "\zip\delib\delib_install.exe" der DEDITEC-Treiber CD.

Anmerkung: Auf unserer Website http://www.deditec.de/delib finden Sie immer die aktuellste DELIB Treiber Version.

2.2.2. Schritt 2 - Anschluss des Moduls

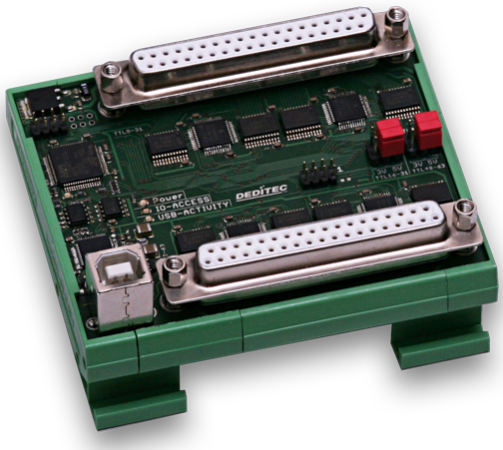
Verbinden Sie ihren PC per USB-Kabel mit dem USB-Anschluss des Moduls.

Nach etwa 20 Sekunden wird das Modul vom Treiber erkannt und kann nun getestet und betrieben werden.

2.2.3. Schritt 3 - Test der Verbindung und des Moduls

Im Startmenü finden Sie unter "Start -> Alle Programme -> DEDITEC -> DELIB -> Sample Programs" Beispiel-Programme um Ihr Modul zu testen.

2.3. Technische Daten



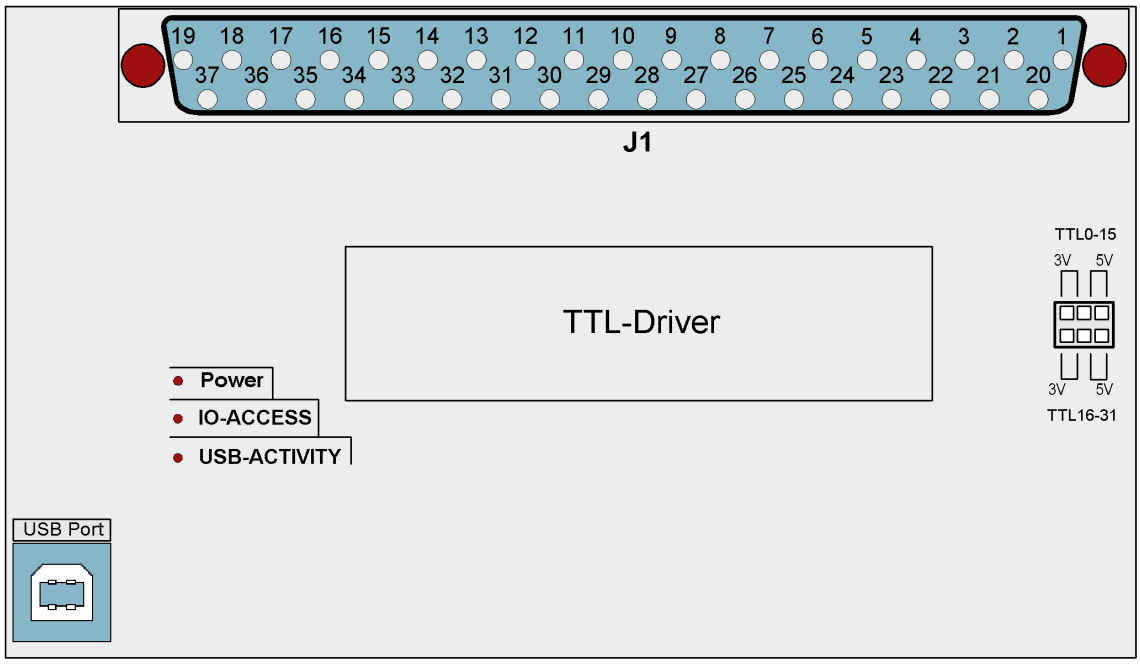
- USB-Interface (USB 1.1 / USB 2.0)
- Spannungsversorgung: +5V (wird über USB-Bus versorgt)
- TTL Pegel 5V bis 1,5V
- TTL-I/O's (in 8-er Blöcken als Ein- oder Ausgang einstellbar)
- Aktivitäts-LED Power (Signalisiert, dass sich das Modul in Betrieb befindet)
- IO-Access-LED (Signalisiert den Zugriff auf die TTL-I/O)
- USB-Activity-LED (Signalisiert, dass eine Signalverarbeitung über den USB-Bus stattfindet)
- Betriebstemperatur 10°C..+50°C
- Abmessungen 90 mm x 77 mm x 42 mm (L x B x H)

Produktspezifische Daten:

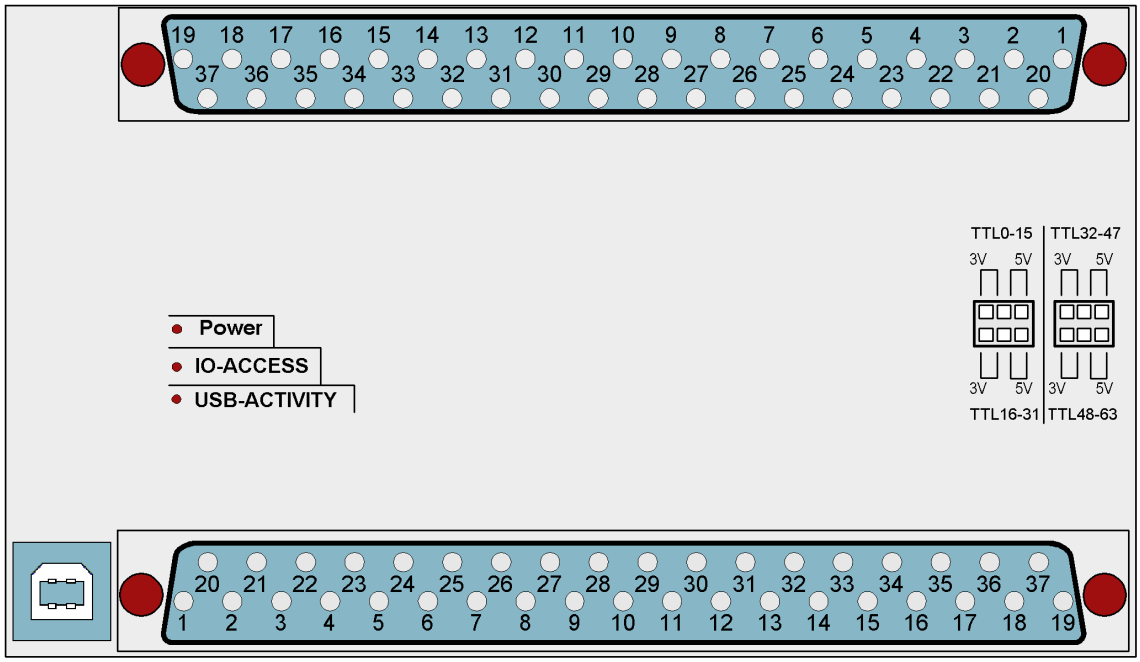
Produkt	TTL-I/O's	Steckverbinder
USB-TTL-32	32	1*37 polige D-Sub Buchse
USB-TTL-64	64	2*37 polige D-Sub Buchse

2.4. Übersichtsbilder

2.4.1. Übersichtsbild USB-TTL-32

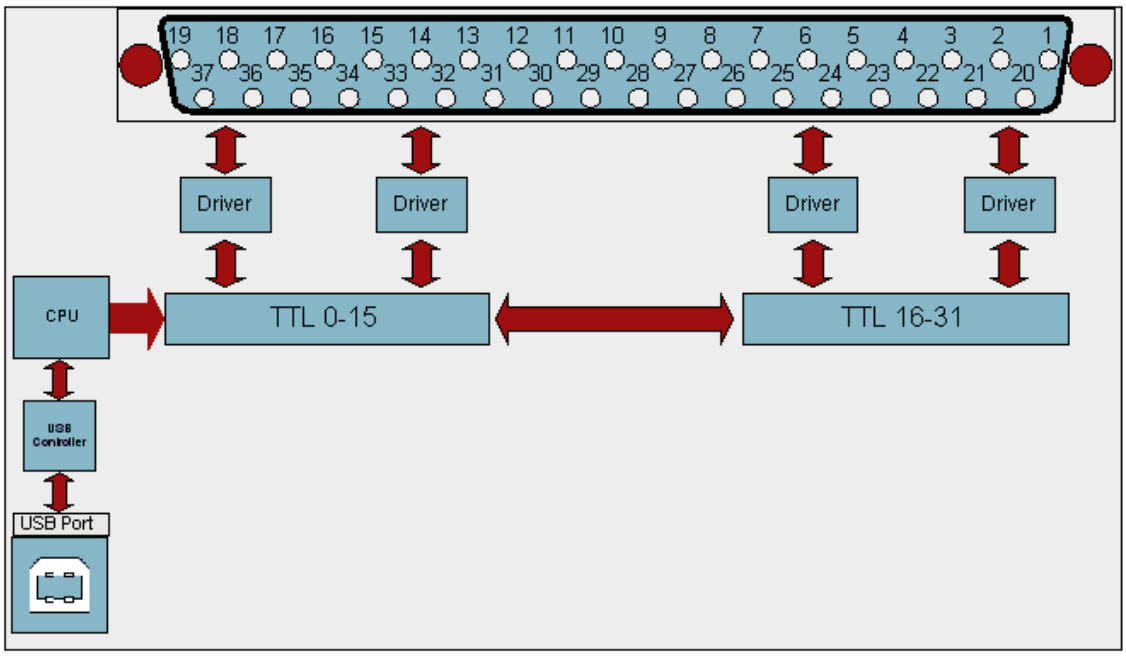


2.4.2. Übersichtsbild USB-TTL-64

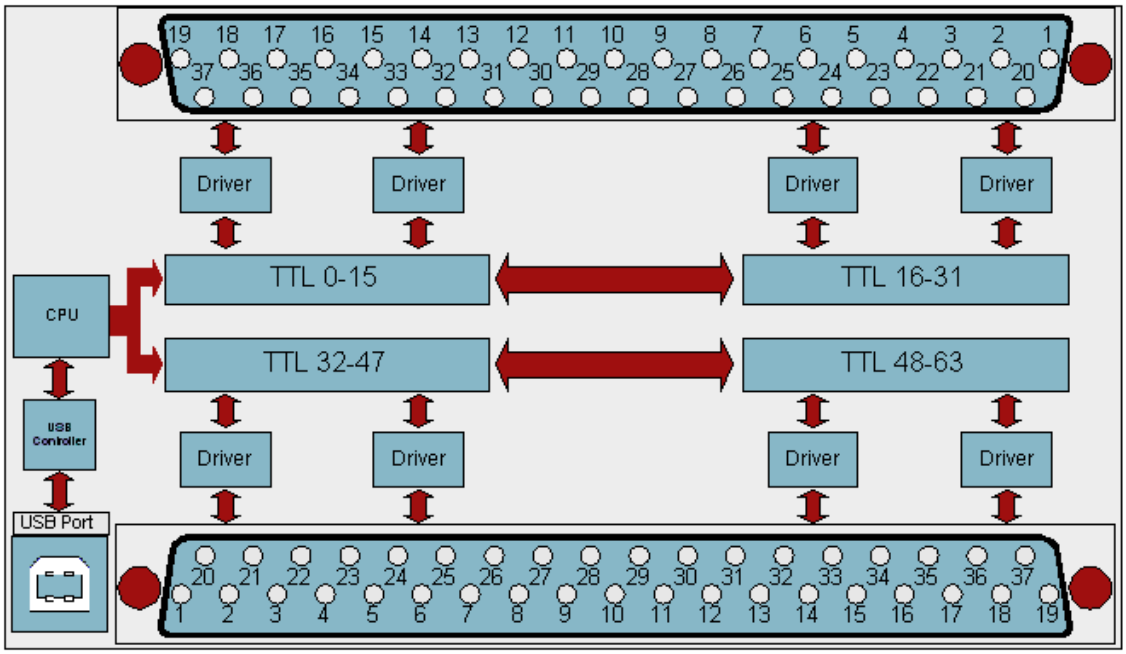


2.5. Blockschaltbilder

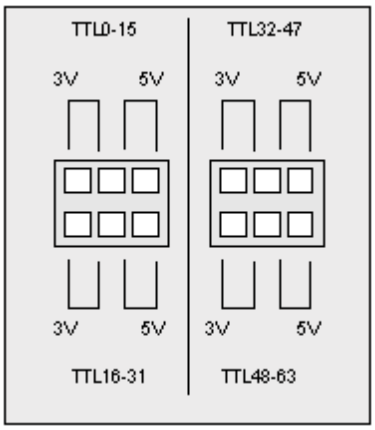
2.5.1. Blockschaltbild USB-TTL-32



2.5.2. Blockschaltbild USB-TTL-64



2.6. Spannungspegel der TTL-I/O's konfigurieren



TTL Pegel von 1,8V bis 5V:

Standardmäßig können Sie TTL Pegel wahlweise von 3,3V oder 5V über eine jeweilige Jumperbelegung einstellen.

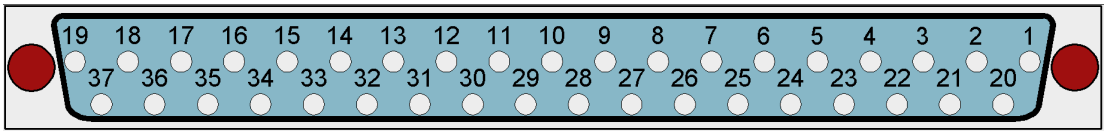
Entfernen Sie die Jumper auf dem Modul, so können Sie eine eigene Spannung von 1,8V bis 5V an die TTL-I/O's des Moduls anlegen, wodurch die Einsatzmöglichkeiten bei Ihnen erheblich gesteigert werden.

Wenn Sie eine eigene Spannung anlegen möchten, geschieht dies über den VIN-Pin siehe Kapitel **Pinbelegung**.

DieTTL-I/O's des Moduls werden dabei in 16er Blöcken konfiguriert.

2.7. Pinbelegung

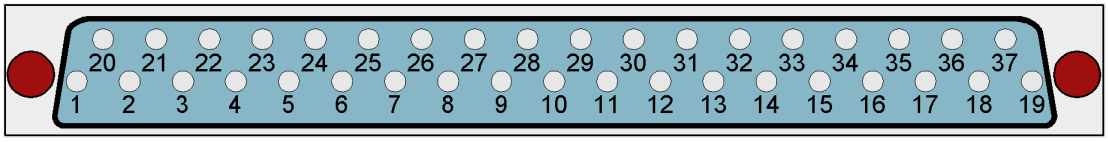
2.7.1. J1 - Pinbelegung USB-TTL-I/O 0-31



Port	Pin	Port	Pin
1	I/O 16	20	I/O 17
2	I/O 18	21	I/O 19
3	I/O 20	22	I/O 21
4	I/O 22	23	I/O 23
5	I/O 24	24	I/O 25
6	I/O 26	25	I/O 27
7	I/O 28	26	I/O 29
8	I/O 30	27	I/O 31
9	I/O 0	28	I/O 1
10	I/O 2	29	I/O 3
11	I/O 4	30	I/O 5
12	I/O 6	31	I/O 7
13	I/O 8	32	I/O 9
14	I/O 10	33	I/O 11
15	I/O 12	34	I/O 13
16	I/O 14	35	I/O 15
17	VIN 0-15	36	VIN 16-31
18	GND	37	GND
19	GND		

Anmerkung: Der VIN-Pin dient dazu, eine eigene Spannung an die TTL-I/O's des Moduls zu legen. Diese Spannung kann zwischen 1,8V und 5V liegen.

2.7.2. J2 - Pinbelegung USB-TTL-I/O 32-63



Port	Pin	Port	Pin
1	I/O 48	20	I/O 49
2	I/O 50	21	I/O 51
3	I/O 52	22	I/O 53
4	I/O 54	23	I/O 55
5	I/O 56	24	I/O 57
6	I/O 58	25	I/O 59
7	I/O 60	26	I/O 61
8	I/O 62	27	I/O 63
9	I/O 32	28	I/O 33
10	I/O 34	29	I/O 35
11	I/O 36	30	I/O 37
12	I/O 38	31	I/O 39
13	I/O 40	32	I/O 41
14	I/O 42	33	I/O 43
15	I/O 44	34	I/O 45
16	I/O 46	35	I/O 47
17	VIN 32-47	36	VIN 48-63
18	GND	37	GND
19	GND		

Anmerkung: Der VIN-Pin dient dazu, eine eigene Spannung an dieTTL-I/O's des Moduls zu legen. Diese Spannung kann zwischen 1,8V und 5V liegen.

Software

3. Software

3.1. Benutzung unserer Produkte

3.1.1. Ansteuerung über grafische Anwendungen

Wir stellen Treiberinterfaces z.B. für LabVIEW und ProfiLab zur Verfügung. Als Basis dient die DELIB Treiberbibliothek, die von ProfiLab direkt angesteuert werden kann.
Für LabVIEW bieten wir eine einfache Treiberanbindung mit Beispielen an!

3.1.2. Ansteuerung über unsere DELIB Treiberbibliothek

Im Anhang befindet sich die komplette Funktionsreferenz für das Integrieren unserer API-Funktionen in Ihre Software. Des Weiteren bieten wir passende Beispiele für folgende Programmiersprachen:

- C
- C++
- C#
- Delphi
- VisualBasic
- VB.NET
- MS-Office

3.1.3. Ansteuerung auf Protokollebene

Das Protokoll für die Ansteuerung unserer Produkte legen wir komplett offen. So können Sie auch auf Systemen ohne Windows oder Linux unsere Produkte einsetzen!

3.1.4. Ansteuerung über mitgelieferte Testprogramme

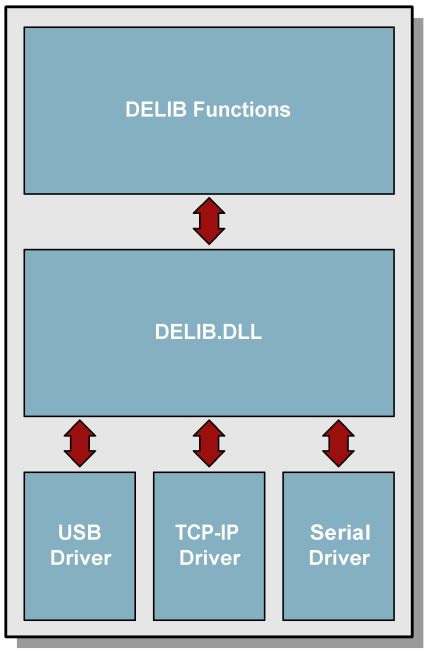
Für die wichtigsten Funktionen unserer Produkte stellen wir einfach zu bedienende Testprogramme zur Verfügung,. Diese werden bei der Installation der DELIB Treiberbibliothek direkt mit installiert.

So können z.B. Relais direkt getestet werden oder Spannungen am A/D Wandler direkt überprüft werden.

3.2. DELIB Treiberbibliothek

3.2.1. Übersicht

Die folgende Abbildung erläutert den Aufbau der DELIB Treiberbibliothek



Die DELIB Treiberbibliothek ermöglicht ein einheitliches Ansprechen von DEDITEC Hardware, mit der besonderen Berücksichtigung folgender Gesichtspunkte:

- Betriebssystem unabhängig
- Programmiersprachen unabhängig
- Produkt unabhängig

3.2.1.1. Programmieren unter diversen Betriebssystemen

Die DELIB Treiberbibliothek ermöglicht ein einheitliches Ansprechen unserer Produkte auf diversen Betriebssystemen. Wir haben dafür gesorgt, dass mit wenigen Befehlen alle unsere Produkte angesprochen werden können. Dabei spielt es keine Rolle, welches Betriebssystem Sie verwenden. - Dafür sorgt die DELIB !

3.2.1.2. Programmieren mit diversen Programmiersprachen

Für das Erstellen eigener Anwendungen stellen wir Ihnen einheitliche Befehle zur Verfügung. Dies wird über die DELIB Treiberbibliothek gelöst.
Sie wählen die Programmiersprache !

So können leicht Anwendung unter C++, C, Visual Basic, Delphi oder LabVIEW® entwickelt werden.

3.2.1.3. Schnittstellenunabhängiges programmieren

Schreiben Sie Ihre Anwendung schnittstellenunabhängig !
Programmieren Sie eine Anwendung für ein USB-Produkt von uns. - Es wird auch mit einem Ethernet oder RS-232 Produkt von uns laufen !

3.2.1.4. SDK-Kit für Programmierer

Integrieren Sie die DELIB in Ihre Anwendung. Auf Anfrage erhalten Sie von uns kostenlos Installationsskripte, die es ermöglichen, die DELIB Installation in Ihre Anwendung mit einzubinden.

3.2.2. Unterstützte Betriebssysteme

Unsere Produkte unterstützen folgende Betriebssysteme:

- Windows 7
- Windows Vista
- Windows XP
- Windows 2000
- Linux

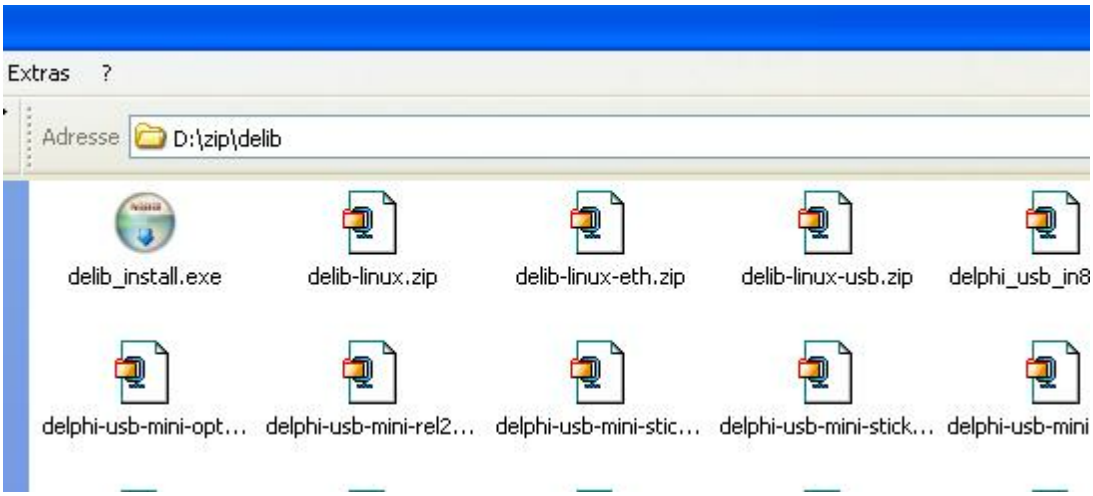
3.2.3. Unterstützte Programmiersprachen

Unsere Produkte sind über folgende Programmiersprachen ansprechbar:

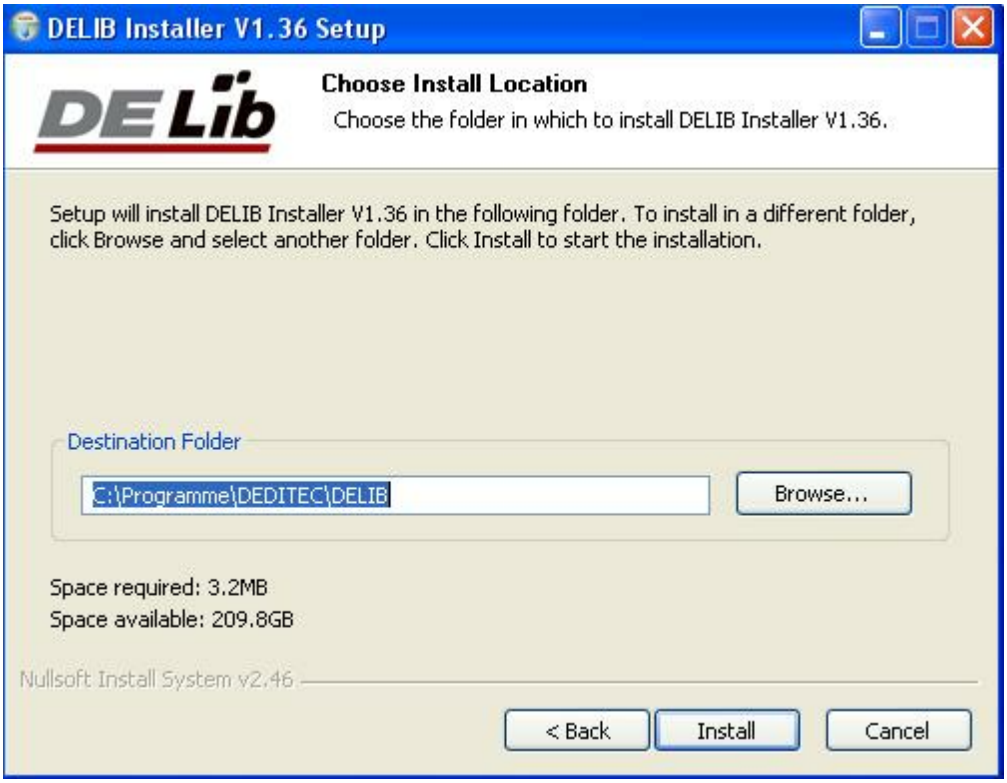
- C
- C++
- C#
- Delphi
- VisualBasic
- VB.NET
- MS-Office

3.2.4. Installation DELIB-Treiberbibliothek

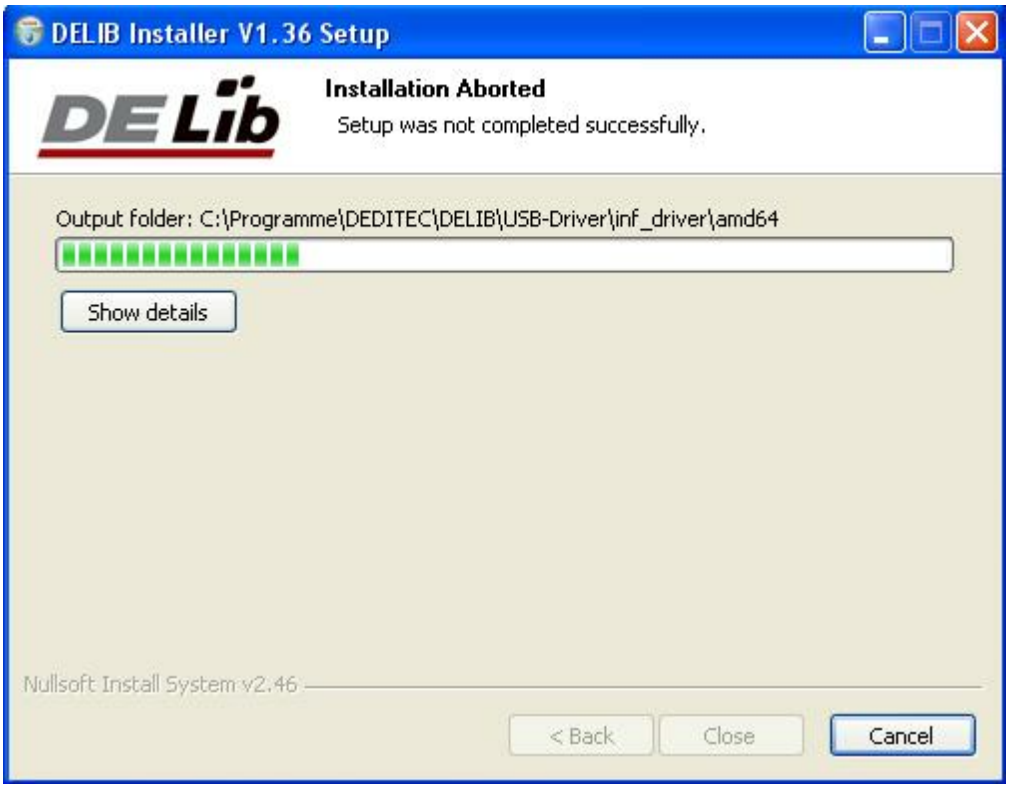
Startbild unseres DELIB Installers.



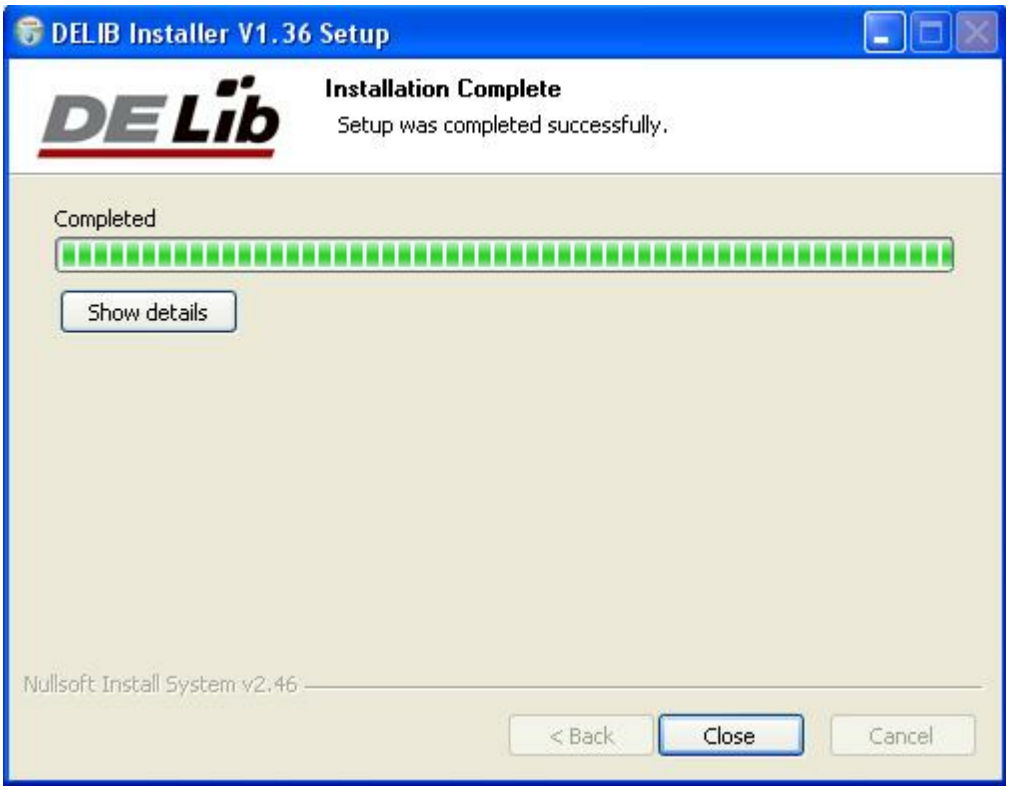
Legen Sie die DEDITEC driver CD in das Laufwerk und starten Sie **“delib_install.exe”**. Die DELIB-Treiberbibliothek ist auch unter http://www.deditec.de/delib erhältlich.



Drücken Sie auf **"Install"**.



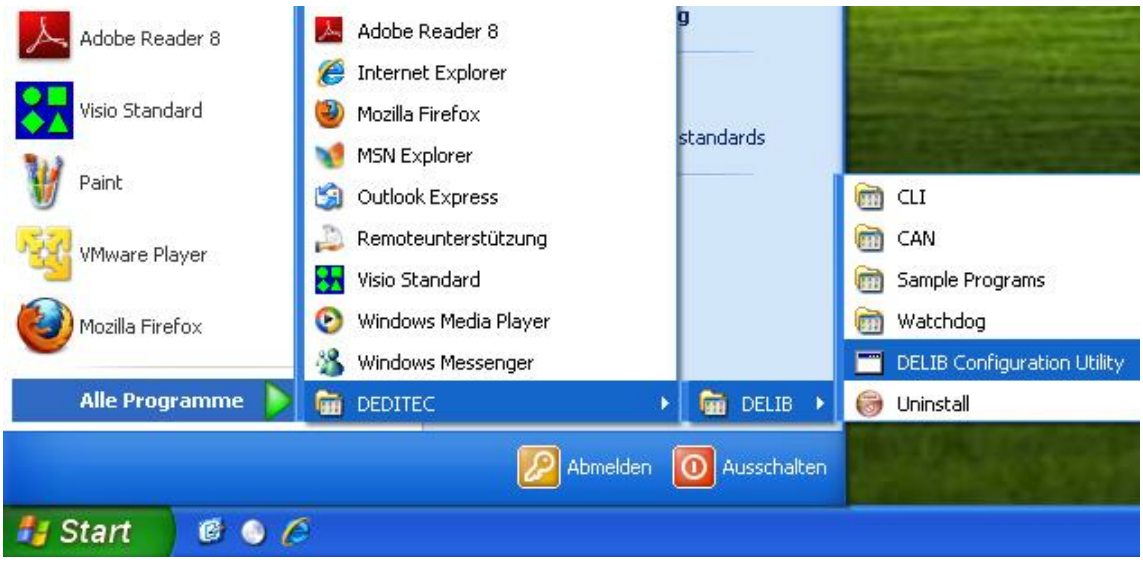
Die Treiber werden nun installiert.



Die DELIB Treiberbibliothek wurde nun Installiert. Drücken sie auf **"Close"** um die Installation zu beenden.

Mit dem **"DELIB Configuration Utility"** (nächstes Kapitel) können Sie Ihr Modul konfigurieren (dies ist nur nötig, wenn Sie mehr als ein Modul ansprechen möchten).

3.2.5. DELIB Configuration Utility



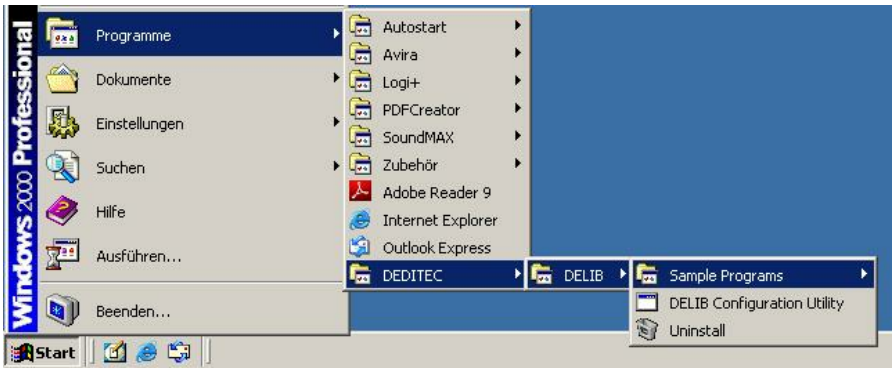
"DELIB Configuration Utility" wird auf dem folgendem Weg gestartet:
Start → Programme → DEDITEC → DELIB → DELIB Configuration Utility.

Das **"DELIB Configuration Utility"** ist ein Programm zur Konfiguration und Unterteilung Identischer USB-Module im System. Dies ist aber nicht nötig falls nur ein Modul vorhanden ist.

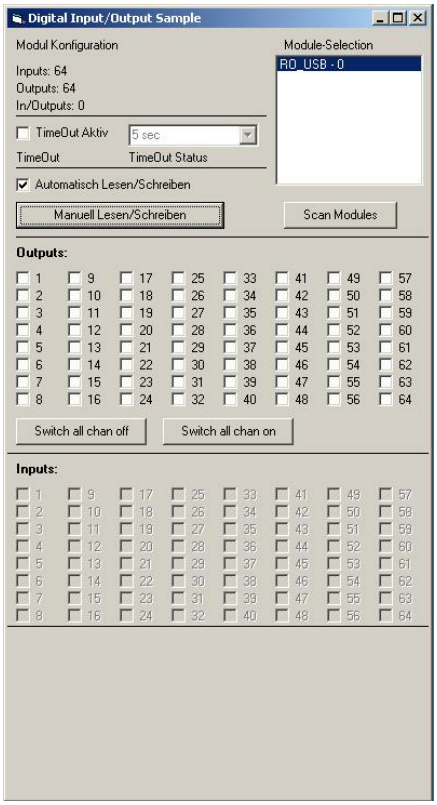
Weiteres zum Inhalt der **"DELIB Installation"**, siehe **"Manual für DELIB Treiberbibliothek"**

3.3. Testprogramme

3.3.1. Digital Input-Output Demo



"Digital Input-Output Demo" wird auf dem folgendem Weg gestartet:
Start → Programme → DEDITEC → DELIB → Digital Input-Output Demo.



Diese Grafik zeigt einen Test des RO-USB-O64-R64. Oben links kann man die Konfiguration des Moduls ablesen (64 Eingänge und 64 Ausgänge).

DELIB API Referenz

4. DELIB API Referenz

4.1. Verwaltungsfunktionen

4.1.1. DapiOpenModule

Beschreibung
Diese Funktion öffnet ein bestimmtes Modul.

Definition
ULONG DapiOpenModule(ULONG moduleID, ULONG nr);

Parameter
moduleID=Gibt das Modul an, welches geöffnet werden soll (siehe delib.h)
nr=Gibt an, welches (bei mehreren Modulen) geöffnet werden soll.
nr=0 -> 1. Modul
nr=1 -> 2. Modul

Return-Wert
handle=Entsprechender Handle für das Modul
handle=0 -> Modul wurde nicht gefunden

Bemerkung
Der von dieser Funktion zurückgegebene Handle wird zur Identifikation des Moduls für alle anderen Funktionen benötigt.

Programmierbeispiel

```
// USB-Modul öffnen
handle = DapiOpenModule(RO_USB1, 0);
printf("handle = %x\n", handle);
if (handle==0)
{
// USB Modul wurde nicht gefunden
printf("Modul konnte nicht geöffnet werden\n");
return;
}
```

4.1.2. DapiCloseModule

Beschreibung
Dieser Befehl schliesst ein geöffnetes Modul.

Definition
ULONG DapiCloseModule(ULONG handle);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls

Return-Wert
Keiner

Programmierbeispiel

```
// Modul schliessen
DapiCloseModule(handle);
```

4.1.3. DapiGetDELIBVersion

Beschreibung
Diese Funktion gibt die installierte DELIB-Version zurück.

Definition
ULONG DapiGetDELIBVersion(ULONG mode, ULONG par);

Parameter
mode=Modus, mit dem die Version ausgelesen wird (muss immer 0 sein).
par=Dieser Parameter ist nicht definiert (muss immer 0 sein).

Return-Wert
version=Versionsnummer der installierten DELIB-Version [hex]

Programmierbeispiel

```
version = DapiGetDELIBVersion(0, 0);
//Bei installierter Version 1.32 ist version = 132(hex)
```

4.1.4. DapiSpecialCMDGetModuleConfig

Beschreibung

Diese Funktion gibt die Hardwareaustattung (Anzahl der Ein- bzw. Ausgangskanäle) des Moduls zurück.

Definition

ULONG DapiSpecialCommand(ULONG handle, DAPI_SPECIAL_CMD_GET_MODULE_CONFIG, par, 0, 0);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

Anzahl der digitalen Eingangskanäle abfragen

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DI

Anzahl der digitalen Ausgangskanäle abfragen

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DO

Anzahl der digitalen Ein-/Ausgangskanäle abfragen

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DX

Anzahl der analogen Eingangskanäle abfragen

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_AD

Anzahl der analogen Ausgangskanäle abfragen

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DA

Anzahl der Stepperkanäle abfragen

par=DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_STEPPER

Return-Wert

Anzahl der digitalen Eingangskanäle abfragen

return=Anzahl der digitalen Eingangskanäle

Anzahl der digitalen Ausgangskanäle abfragen

return=Anzahl der digitalen Ausgangskanäle

Anzahl der digitalen Ein-/Ausgangskanäle abfragen

return=Anzahl der digitalen Ein-/Ausgangskanäle

Anzahl der analogen Eingangskanäle abfragen

return=Anzahl der analogen Eingangskanäle

Anzahl der analogen Ausgangskanäle abfragen

return=Anzahl der analogen Ausgangskanäle

Anzahl der Stepperkanäle abfragen

return=Anzahl der Stepperkanäle

Programmierbeispiel

```
ret=DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_GET_MODULE_CONFIG,
DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DI, 0, 0);
//Gibt die Anzahl der digitalen Eingangskanäle zurück
ret=DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_GET_MODULE_CONFIG,
DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DO, 0, 0);
//Gibt die Anzahl der digitalen Ausgangskanäle zurück
ret=DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_GET_MODULE_CONFIG,
DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DX, 0, 0);
//Gibt die Anzahl der digitalen Ein-/Ausgangskanäle zurück
ret=DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_GET_MODULE_CONFIG,
DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_AD, 0, 0);
//Gibt die Anzahl der analogen Eingangskanäle zurück
ret=DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_GET_MODULE_CONFIG,
DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_DA, 0, 0);
//Gibt die Anzahl der analogen Ausgangskanäle zurück
ret=DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_GET_MODULE_CONFIG,
DAPI_SPECIAL_GET_MODULE_CONFIG_PAR_STEPPER, 0, 0);
//Gibt die Anzahl der Stepperkanäle zurück
```

4.2. Fehlerbehandlung

4.2.1. DapiGetLastError

Beschreibung
Diese Funktion liefert den letzten erfassten Fehler.

Definition
ULONG DapiGetLastError();

Parameter
Keine

Return-Wert
Fehler Code
0=kein Fehler. (siehe delib.h)

Programmierbeispiel

```
ULONG error;
error=DapiGetLastError();
if(error==0) return FALSE;
printf("ERROR = %d", error);
```

4.2.2. DapiGetLastErrorText

Beschreibung
Diese Funktion liest den Text des letzten erfassten Fehlers.

Definition
extern ULONG __stdcall DapiGetLastErrorText(unsigned char * msg, unsigned long msg_length);

Parameter
msg = Buffer für den zu empfangenden Text
msg_length = Länge des Text Buffers

Programmierbeispiel

```
BOOL IsError ()
{
  if (DapiGetLastError () != DAPI_ERR_NONE)
       {
         unsigned char msg[500];

         DapiGetLastErrorText((unsigned char*) msg, sizeof(msg));
         printf ("Error Code = %x * Message = %s\n", 0, msg);
         return TRUE;
       }
  return FALSE;
}
```

4.3. TTL-Ein-/Ausgangs Richtungen setzen

4.3.1. DAPI_SPECIAL_CMD_SET_DIR_DX_8

Beschreibung

Dieser Befehl setzt die Richtung von TTL-Ein/Ausgängen (8-Bit weise).

Definition

void DapiSpecialCommand(ULONG handle, DAPI_SPECIAL_CMD_SET_DIR_DX_8, ULONG ch, ULONG dir, 0);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

ch=Gibt die Nummer des Ausgangs an, ab dem die Richtung gesetzt werden soll (0, 8, 16, 24 ..). Zwischenwerte sind ungültig

dir=(8-Bit) gibt die Richtung für 8 hintereinanderliegende Ein/Ausgänge an. (1=output / 0=input)

Programmierbeispiel

```
DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SET_DIR_DX_8, 0, 1, 0);
// Set Dir of TTL-I/O CH0 to out
```

4.4. Digitale Eingänge lesen

4.4.1. DapiDIGet1

Beschreibung
Dieser Befehl liest einen einzelnen digitalen Eingang.

Definition
ULONG DapiDIGet1(ULONG handle, ULONG ch);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt die Nummer des Eingangs an, der gelesen werden soll (0, 1, 2, 3, ..)

Return-Wert
Zustand des Eingangs (0/1)

4.4.2. DapiDlGet8

Beschreibung
Dieser Befehl liest gleichzeitig 8 digitale Eingänge.

Definition
ULONG DapiDlGet8(ULONG handle, ULONG ch);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt die Nummer des Eingangs an, ab dem gelesen werden soll (0, 8, 16, 24, ..)

Return-Wert
Zustand der gelesen Eingänge

4.4.3. DapiDlGet16

Beschreibung
Dieser Befehl liest gleichzeitig 16 digitale Eingänge.

Definition
ULONG DapiDlGet16(ULONG handle, ULONG ch);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt die Nummer des Eingangs an, ab dem gelesen werden soll (0, 16, 32, ...)

Return-Wert
Zustand der gelesen Eingänge

4.4.4. DapiDIGet32

Beschreibung
Dieser Befehl liest gleichzeitig 32 digitale Eingänge.

Definition
ULONG DapiDIGet32(ULONG handle, ULONG ch);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt die Nummer des Eingangs an, ab dem gelesen werden soll (0, 32, 64, ..)

Return-Wert
Zustand der gelesen Eingänge

Programmierbeispiel

```
unsigned long data;
// -----------------------------------------------------
// Einen Wert von den Eingängen lesen (Eingang 1-31)
data = (unsigned long) DapiDIGet32(handle, 0);
// Chan Start = 0
printf("Eingang 0-31 : 0x%x\n", data);
printf("Taste für weiter\n");
getch();
// -----------------------------------------------------
// Einen Wert von den Eingängen lesen (Eingang 32-64)
data = (unsigned long) DapiDIGet32(handle, 32);
// Chan Start = 32
printf("Eingang 32-64 : 0x%x\n", data);
printf("Taste für weiter\n");
getch();
```

4.4.5. DapiDIGet64

Beschreibung
Dieser Befehl liest gleichzeitig 64 digitale Eingänge.

Definition
ULONGLONG DapiDIGet64(ULONG handle, ULONG ch);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt die Nummer des Eingangs an, ab dem gelesen werden soll (0, 64, ..)

Return-Wert
Zustand der gelesen Eingänge

4.5. Digitale Ausgänge verwalten

4.5.1. DapiDOSet1

Beschreibung
Dieser Befehl setzt einen einzelnen Ausgang.

Definition
void DapiDOSet1(ULONG handle, ULONG ch, ULONG data);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt die Nummer des zu setzenden Ausgangs an (0 ..)
data=Gibt den Datenwert an, der geschrieben wird (0 / 1)

Return-Wert
Keiner

4.5.2. DapiDOSet8

Beschreibung
Dieser Befehl setzt gleichzeitig 8 digitale Ausgänge.

Definition
void DapiDOSet8(ULONG handle, ULONG ch, ULONG data);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt die Nummer des Ausgangs an, ab dem geschrieben werden soll (0, 8, 16, 24, 32, ..)
data=Gibt die Datenwerte an, die geschrieben werden

Return-Wert
Keiner

4.5.3. DapiDOSet16

Beschreibung
Dieser Befehl setzt gleichzeitig 16 digitale Ausgänge.

Definition
void DapiDOSet16(ULONG handle, ULONG ch, ULONG data);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt die Nummer des Ausgangs an, ab dem geschrieben werden soll (0, 16, 32, ..)
data=Gibt die Datenwerte an, die geschrieben werden

Return-Wert
Keiner

4.5.4. DapiDOSet32

Beschreibung
Dieser Befehl setzt gleichzeitig 32 digitale Ausgänge.

Definition
void DapiDOSet32(ULONG handle, ULONG ch, ULONG data);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt die Nummer des Ausgangs an, ab dem geschrieben werden soll (0, 32, 64, ..)
data=Gibt die Datenwerte an, die geschrieben werden

Return-Wert
Keiner

Programmierbeispiel

```
// Einen Wert auf die Ausgänge schreiben
data = 0x0000ff00; // Ausgänge 9-16 werden auf 1 gesetzt
DapiDOSet32(handle, 0, data); // Chan Start = 0
printf("Schreibe auf Ausgänge Daten=0x%x\n", data);
printf("Taste für weiter\n");
getch();
// ------------------------------------------------
// Einen Wert auf die Ausgänge schreiben
data = 0x80000000; // Ausgang 32 wird auf 1 gesetzt
DapiDOSet32(handle, 0, data); // Chan Start = 0
printf("Schreibe auf Ausgänge Daten=0x%x\n", data);
printf("Taste für weiter\n");
getch();
// ------------------------------------------------
// Einen Wert auf die Ausgänge schreiben
data = 0x80000000; // Ausgang 64 wird auf 1 gesetzt
DapiDOSet32(handle, 32, data); // Chan Start = 32
printf("Schreibe auf Ausgänge Daten=0x%x\n", data);
printf("Taste für weiter\n");
getch();
```

4.5.5. DapiDOSet64

Beschreibung
Dieser Befehl setzt gleichzeitig 64 digitale Ausgänge.

Definition
void DapiDOSet64(ULONG handle, ULONG ch, ULONG data);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt die Nummer des Ausgangs an, ab dem geschrieben werden soll (0, 64, ..)
data=Gibt die Datenwerte an, die geschrieben werden

Return-Wert
Keiner

4.5.6. DapiDOReadback32

Beschreibung
Dieser Befehl liest die 32 digitalen Ausgänge zurück.

Definition
ULONG DapiDOReadback32(ULONG handle, ULONG ch);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt die Nummer des Ausgangs an, ab dem zurückgelesen werden soll (0, 32, 64, ..)

Return-Wert
Zustand von 32 Ausgängen.

4.5.7. DapiDOReadback64

Beschreibung
Dieser Befehl liest die 64 digitalen Ausgänge zurück.

Definition
ULONGLONG DapiDOReadback64(ULONG handle, ULONG ch);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls
ch=Gibt die Nummer des Ausgangs an, ab dem zurückgelesen werden soll (0, 64, ..)

Return-Wert
Zustand von 64 Ausgängen.

4.6. Programmier-Beispiel

```
//***************************************************************************
//***************************************************************************
//***************************************************************************
//***************************************************************************
//***************************************************************************
//
//
//  product: usb-ttl-32 (ModuleID = USB_TTL_32)
//  configuration: ttl-io
//  programming language: vc
//
//
//  (c) DEDITEC GmbH, 2011
//  web: http://www.deditec.de/
//  mail: vertrieb@deditec.de
//
//
//***************************************************************************
//***************************************************************************
//***************************************************************************
//***************************************************************************
//***************************************************************************
//
//
// Please include the following library on linking: delib.lib
//
// This can be done at the project settings (Project/Settings/Link ->
// Object/library modules) .. extend the existing line with the ending
// "$(DELIB_LIB)\delib.lib"        (with quotation marks)
//
// Including the header file delib.h (Project/Settings/C/C++ -> select
category
// "Preprocessor" -> Additional inlude directories) .. enter the line
// "$(DELIB_INCLUDE)"       (with quotation marks)

#include <windows.h>
#include <stdio.h>
#include "conio.h"

#include "delib.h"

// ----------------------------------------------------------------------------
// GetLastError function

BOOL IsError()
{
       unsigned char msg[500];

       if (DapiGetLastError() != DAPI_ERR_NONE)
       {

              DapiGetLastErrorText((unsigned char*) msg, sizeof(msg));
              printf("Error Code = %x * Message = %s\n", 0, msg);
```

```
                DapiClearLastError();

                return TRUE;
        }

        return FALSE;
}

//***************************************************************************
//***************************************************************************
//***************************************************************************
//***************************************************************************
//***************************************************************************

void main(void)
{
        unsigned long handle;
        unsigned long data;

        // -------------------------------------------------------------------
        // Open Module

        handle = DapiOpenModule(USB_TTL_32,0);

        printf("Module handle = %x\n", handle);

        // -------------------------------------------------------------------
        // Module not found!

        if (handle==0)
        {
                printf("Could not open module!\n");
                printf("Press any key to exit\n");
                getch();
                return;
        }

        // -------------------------------------------------------------------
        // Module found!

        printf("Module has been opened\n");

        // -------------------------------------------------------------------
        // Switch i/o to inputs

        DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SET_DIR_DX_8, 0, 0, 0);
        IsError();
        printf("Channel 0-7 has been set to inputs\n");
        printf("Press any key to continue\n");
        getch();

        // -------------------------------------------------------------------
        // Read value of inputs 0-7

        data = DapiDIGet8(handle, 0);
        IsError();
```

```
        printf("Value of inputs 0-7 = %d\n", data);
        printf("Press any key to continue\n");
        getch();

        // ------------------------------------------------------------------------
        // Switch i/o to outputs

        DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_SET_DIR_DX_8, 0, 255, 0);
        IsError();
        printf("Channel 0-7 has been set to outputs\n");
        printf("Press any key to continue\n");
        getch();

        // ------------------------------------------------------------------------
        // Write values to outputs 0-7

        DapiDOSet8(handle, 0, 0xf0);
        IsError();
        printf("Write 0xf0 to outputs 0-7\n");
        printf("Press any key to continue\n");
        getch();

        // ------------------------------------------------------------------------
        // Readback a value of inputs 0-7

        data = DapiDIGet8(handle, 0);
        IsError();
        printf("Readback input 0-7 (from output 0-7)\n");
        printf("value = %d\n", data);
        printf("Press any key to continue\n");
        getch();

        // ------------------------------------------------------------------------
        // Close Module

        DapiCloseModule(handle);
        printf("Module closed\n");
        printf("End of program!\n");
        printf("Press any key to exit\n");
        getch();

        return ;
}
```

Anhang

5. Anhang

5.1. Revisionen

Rev 2.00	Erste DEDITEC Anleitung

5.2. Urheberrechte und Marken

Linux ist eine registrierte Marke von Linus Torvalds.

Windows CE ist eine registrierte Marke von Microsoft Corporation.

USB ist eine registrierte Marke von USB Implementers Forum Inc.

LabVIEW ist eine registrierte Marke von National Instruments.

Intel ist eine registrierte Marke von Intel Corporation

AMD ist eine registrierte Marke von Advanced Micro Devices, Inc.