DEDITEC GMBH

USB-WATCHDOG-STICK

Hardware-Beschreibung

2011 Februar

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

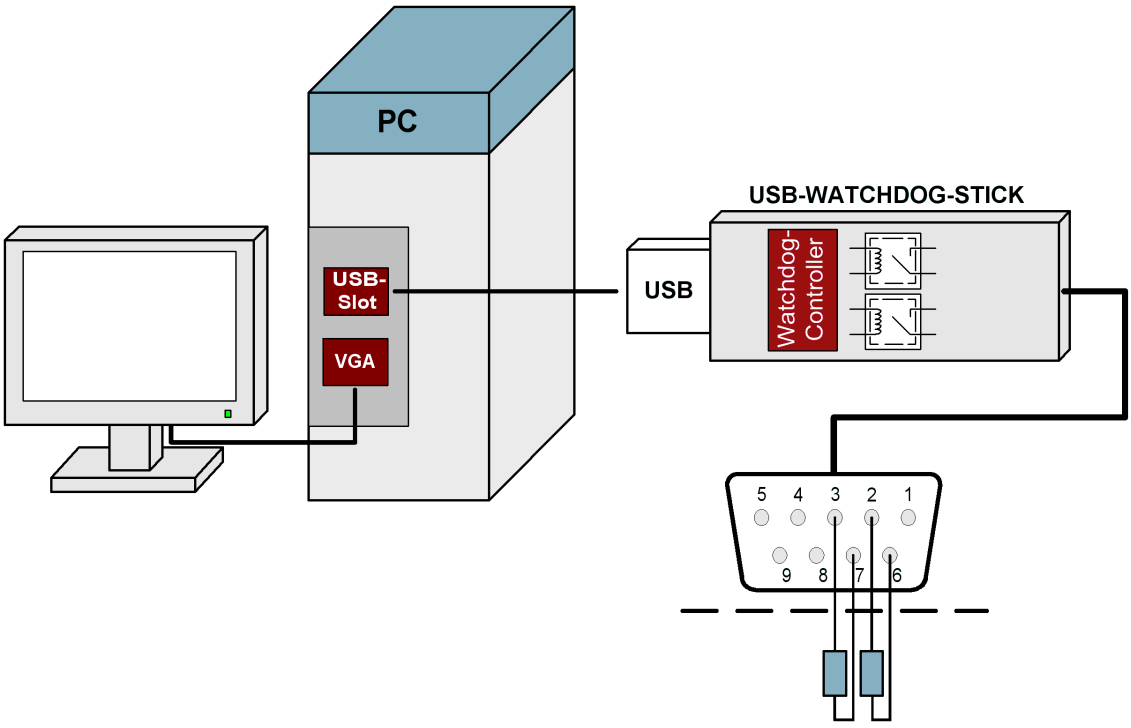
2. Hardware Beschreibung

Die in einem USB-Stick Gehäuse untergebrachte Elektronik sorgt für eine Überwachung mit Watchdogfunktion. Der eingesetzte Microcontroller wird in regelmäßigen Intervallen vom PC über den USB-BUS abgefragt.

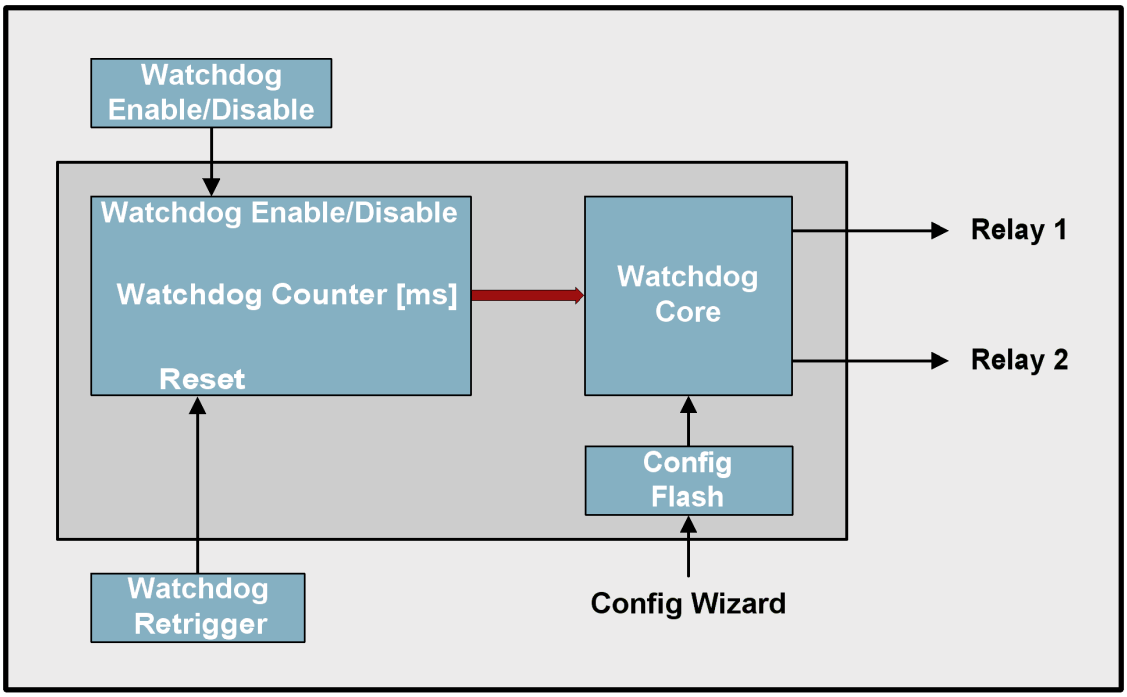
Bei einer Zeitüberschreitung schaltet der Watchdog-Stick dann selbständig die beiden Relais mit einem vom Benutzer vorzugebenem Ablauf.

So können durch die Relais, mit entsprechender Verkabelung, bei einem Timeout z.B. der PC-Reset betätigt werden, ein externes SMS-Modem kann Warnungen versenden oder eine angeschlossene Hupe signalisiert einen Alarm.

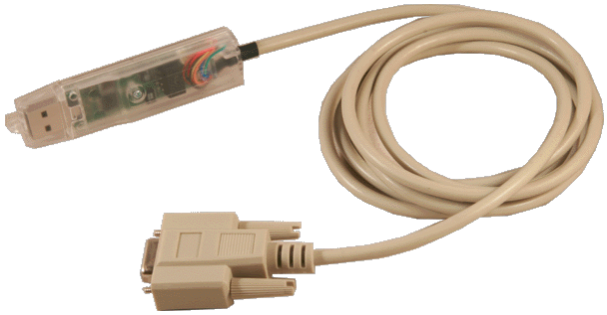
Ein mitgeliefertes Testprogramm und ein Konfigurationstool erleichtern den schnellen Einstieg!



2.1. Übersichtsbild

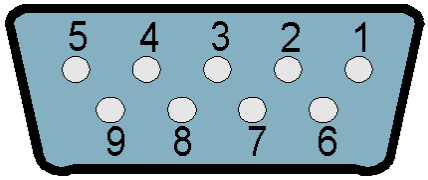


2.2. Technische Daten



- USB-Stick-Ausgabemodul mit USB 2.0 / USB 1.1 Interface
- Microcontroller-Überwachung
- 10ms bis 10h Timeoutzeiten einstellbar
- Windows Watchdog API
- 2 Relais für Schaltvorgänge
- Anschlußkabel (ca 1,8 m) mit DSUB-9 Buchse
- Abmessungen: 84,5 * 21 * 12,5 / 9,5 mm (ohne Kabel)

2.3. Pinbelegung DSUB-9 Buchse



Pin	Beschreibung
3 & 7	Relais 1
2 & 6	Relais 2
1, 4, 5, 8, 9	NC

2.4. Kontroll LED

Die LED auf dem Watchdog-Stick zeigt folgenden Status an:

Beschreibung	LED Blink Sequenz
Watchdog Disabled	LED dauerhaft an (geht bei PC-Zugriffen kurz aus)
Watchdog Enabled	1*an + lange Pause
Watchdog Retrigger	1*an + lange Pause
Watchdog Timeout	2*kurz an + lange Pause

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

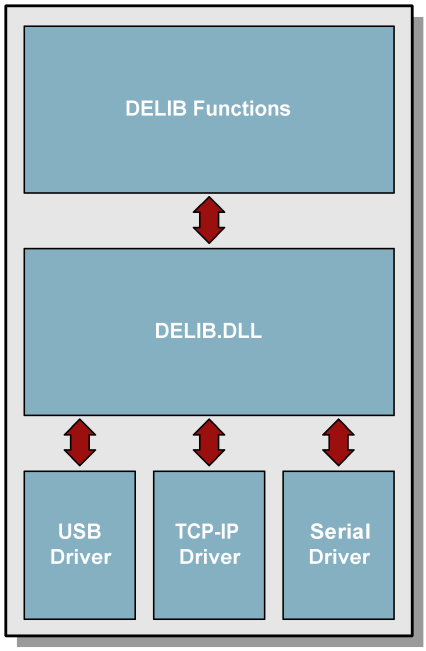
Für die wichtigsten Funktionen unserer Produkte stellen wir einfach zu bedienende Testprogramme zur Verfügung,. Diese werden bei der Installation der DELIB Treiberbibliothek direkt mit installiert.

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:

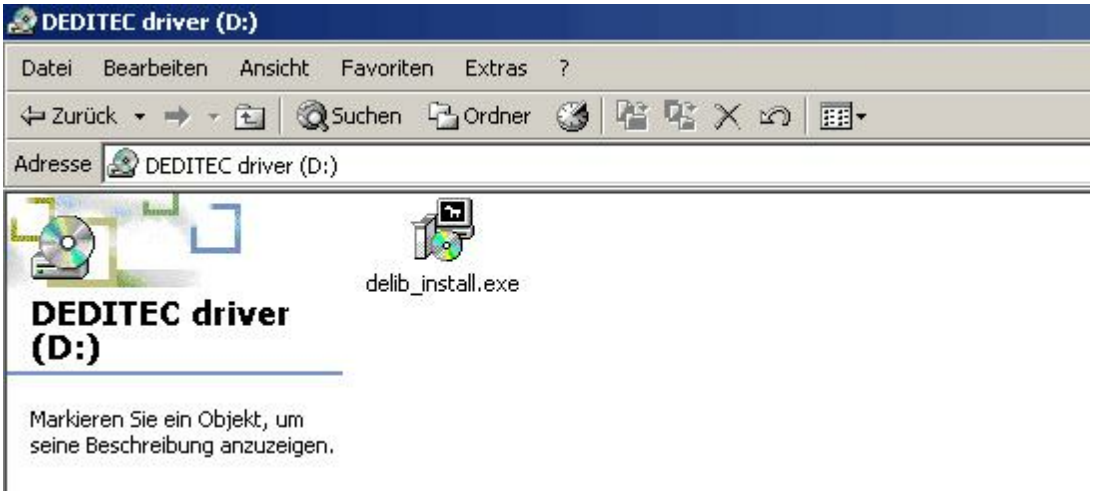
- Windows 7
- Windows Vista
- Windows XP
- Windows 2000
- Linux

3.2.3. Unterstützte Programmiersprachen

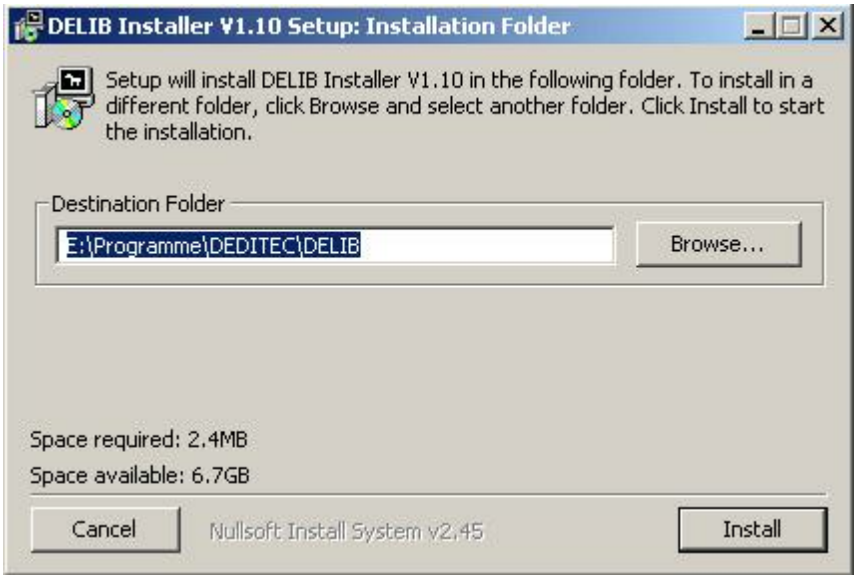
Unsere Produkte sind über folgende Programmiersprachen ansprechbar:

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

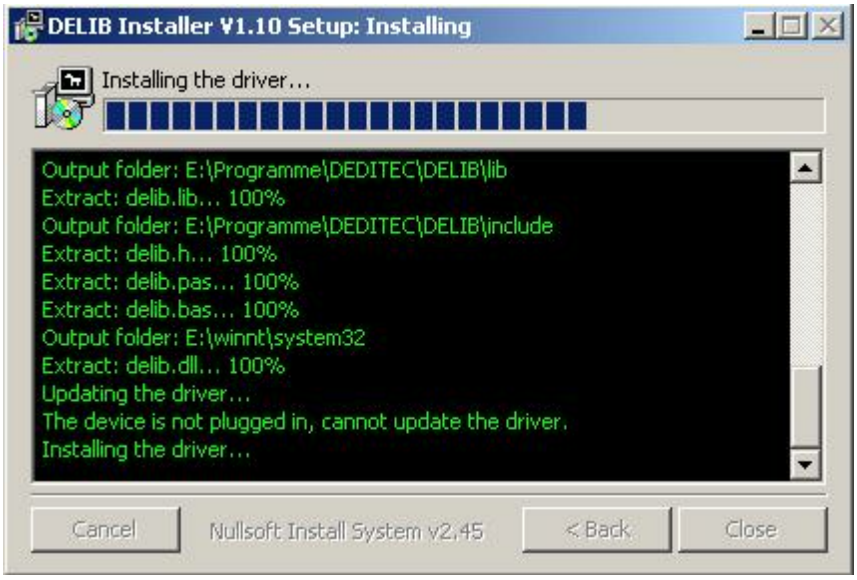
3.2.4. Installation DELIB-Treiberbibliothek



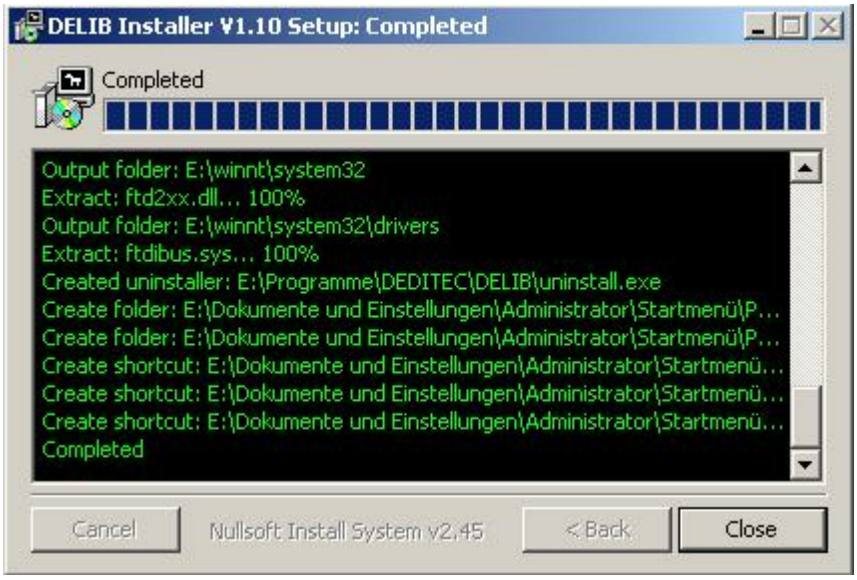
Legen Sie die DEDITEC driver CD in das Laufwerk und starten Sie **“delib_install.exe”**. Die DELIB-Treiberbibliothek ist auch unter http://www.deditec.de/delib erhältlich.



Drücken Sie auf **“Install”**.



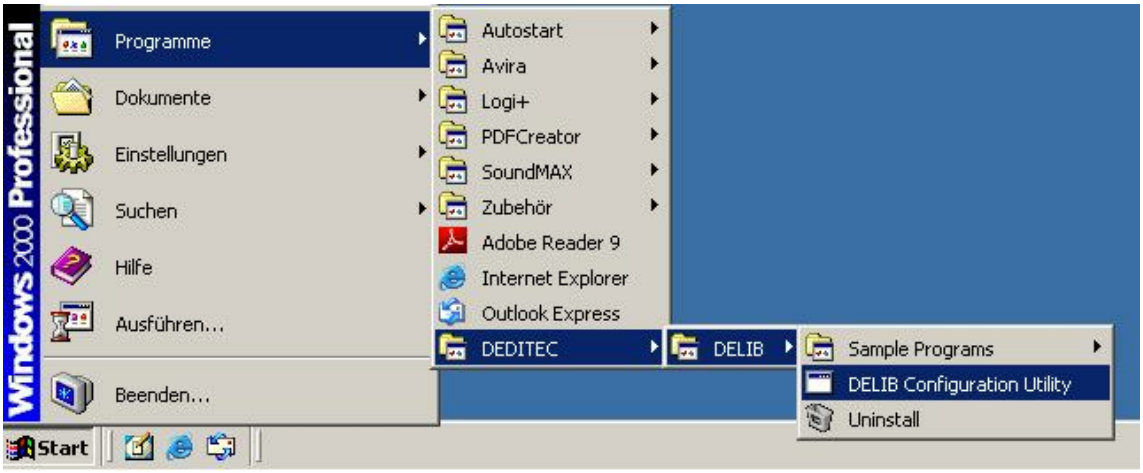
Die Treiber werden nun installiert.



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



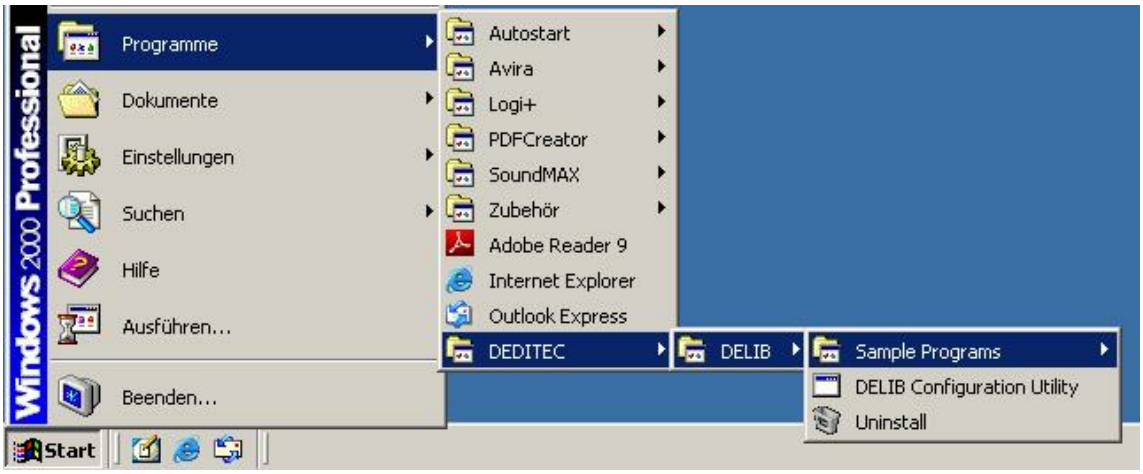
"DELIB Configuration Utility" wird auf dem folgendem Weg gestartet:
Start → Programme → DEDITEC → DELIB → DELIB Configuration Utility.

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. Watchdog Demo



"Watchdog Demo" wird auf folgendem Weg gestartet:
Start -> Programme -> DEDITEC -> DELIB -> Watchdog Demo.

Diese Grafik zeigt einen Test des Watchdog-Sticks. Oben links kann man die Konfiguration des Moduls ablesen.

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);
```

4.2. 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. Watchdog Funktionen

4.3.1. DapiWatchdogEnable

Beschreibung
Diese Funktion aktiviert den Watchdog.

Definition
void DapiWatchdogEnable(ULONG handle);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls

Return-Wert
Keiner

Programmierbeispiel

```
DapiWatchdogEnable(handle);
//Aktiviert den Watchdog
```

4.3.2. DapiWatchdogDisable

Beschreibung
Diese Funktion deaktiviert den Watchdog.

Definition
void DapiWatchdogDisable(ULONG handle);

Parameter
handle=Dies ist das Handle eines geöffneten Moduls

Return-Wert
Keiner

Programmierbeispiel

```
DapiWatchdogDisable(handle);
//Deaktiviert den Watchdog
```

4.3.3. DapiWatchdogRetrigger

Beschreibung

Diese Funktion retriggert den Watchdog-Timer.

Definition

void DapiWatchdogRetrigger(ULONG handle);

Parameter

handle=Dies ist das Handle eines geöffneten Moduls

Return-Wert

Keiner

Programmierbeispiel

```
DapiWatchdogRetrigger(handle);
//Retriggert den Watchdog-Timer
```

4.4. Programmier-Beispiel

```
// ******************************************************************************
// ******************************************************************************
// ******************************************************************************
// ******************************************************************************
// ******************************************************************************
//
// (c) DEDITEC GmbH, 2011
//
// Samples for USB-WATCHDOG
//
// vc-usb-watchdog-sample.cpp
//
//
// ******************************************************************************
// ******************************************************************************
// ******************************************************************************
// ******************************************************************************
// ******************************************************************************
//
//
// Folgende Bibliotheken beim Linken mit einbinden: delib.lib
// Dies bitte in den Projekteinstellungen (Projekt/Einstellungen/Linker(Objekt-
Bibliothek-Module)  .. letzter Eintrag konfigurieren


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


// ------------------------------------------------------------------------
// ------------------------------------------------------------------------
// ------------------------------------------------------------------------
// ------------------------------------------------------------------------
// ------------------------------------------------------------------------


void main(void)
{
       unsigned long handle;
       unsigned long wd_status, wd_timeout, wd_counter;


       // ----------------------------------------------------
       // USB-Modul öffnen
       handle = DapiOpenModule(USB_WATCHDOG,0);

       printf("handle = %x\n", handle);
       if (handle==0)
       {
              // USB Modul wurde nicht gefunden
              printf("Modul konnte nicht geöffnet werden\n");
              printf("TASTE für weiter\n");
              getch();
              return;
```

```
	}


	// -----------------------------------------------------
	// activate watchdog
	DapiWatchdogEnable(handle);
	printf("watchdog is activated\n");

	printf("Taste für weiter\n");
	getch();

	// -----------------------------------------------------
	// activate watchdog
	DapiWatchdogDisable(handle);
	printf("watchdog is deactivated\n");

	printf("Taste für weiter\n");
	getch();

	// -----------------------------------------------------
	// activate watchdog + retrigger
	DapiWatchdogEnable(handle);
	DapiWatchdogRetrigger(handle);
	printf("watchdog is enabled+retriggered\n");

	wd_status=0;

	while(!kbhit() && wd_status!=2)
	{
		wd_status = DapiSpecialCommand(handle, DAPI_SPECIAL_CMD_WATCHDOG,
DAPI_SPECIAL_WATCHDOG_GET_STATUS, 0, 0);
		wd_counter = DapiSpecialCommand(handle,
DAPI_SPECIAL_CMD_WATCHDOG, DAPI_SPECIAL_WATCHDOG_GET_WD_COUNTER_MSEC, 0, 0);
		wd_timeout = DapiSpecialCommand(handle,
DAPI_SPECIAL_CMD_WATCHDOG, DAPI_SPECIAL_WATCHDOG_GET_TIMEOUT_MSEC, 0, 0);
		printf("WD-Status=%d * WD-Timeout=%dms * WD-Counter = %dms\n",
wd_status, wd_timeout, wd_counter);
	}

	if(wd_status==2)
	{
		printf("status=2 -> Watchdog TIMEOUT !!!!\n");
	}
	printf("Taste für weiter\n");
	getch();


	// -----------------------------------------------------
	// Modul wieder schliessen
	DapiCloseModule(handle);

	printf("TASTE für weiter\n");
	getch();

	return ;
}
```

Anhang

5. Anhang

5.1. Revisionen

Rev 2.00	Erste Version

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.