Cleware USB Geräte mit Linux

Version 3.1.3
21.05.2005

Cleware GmbH
Nedderend 3
24876 Hollingstedt
Deutschland
www.cleware.de

1. Allgemeines

Alle Cleware-Geräte lassen sich auch unter SuSE Linux 8.0 (Kernel 2.4.18), SuSE Linux 9.0 (2.4.21) und SuSE Linux 9.2 (2.6.8-24) betreiben. Frühere Linux-Versionen sind ohne Erweiterungen des Kernels nicht in der Lage, die Geräte anzusprechen. Die Versionen 8.1 und 8.2 haben ein paar Probleme mit den USB-Geräten. Wir empfehlen, auf die Version 9.2 umzusteigen.

Alle Funktionen der Geräte werden über eine Library „USBaccess.a“ und der dazuge-hörigen Headerdatei „USBaccess.h“ angesprochen. Die Funktionen sind weitgehend mit dem Interface der Windows-DLL identisch, sodaß die darauf basierenden Programm nahezu ohne Änderung auf beiden Betriebssystemen übersetzbar sind.

Alle Dateien für Linux sind in dem Archiv „SuSE9.2 (Kernel 2.6.8-24).zip“ zu finden. Unter Linux können die Dateien mit dem Programm „unzip“ entpackt werden. Entsprechend dem Geist von Linux ist auch der gesamte Sourcecode für die Library in dem Archiven zu finden.

Bevor die Geräte erstmals angesprochen werden können, müssen diese in /dev/usb als hiddev0, hiddev1, etc. angelegt sein. Leider ist dies nach der Installation von Linux nicht der Fall. Die Geräte können mit dem beiligenden Skript „make_hid“ automatisch angelegt werden. Hierzu muß als User root das Kommando „source make_hid“ gegeben werden. Danach sollten die Geräte unter /etc/usb zu sehen sein. Das Anlegen der Geräte ist ein einmaliger Vorgang und muß nur wiederholt werden, wenn das Linux neu installiert wird.

Neben dem Interface sind die Beispielprogramme „Example“, „USBswitch“ und USBwatch“ zur Verfügung. Um das Linux-System mit der graphischen Oberfläche „ClewareControl“ auf einem Windows-PC zu verbinden, kann man die Meßwerte über TCP/IP Sockets übertragen. Hierzu liegt das Programm „send2cc“ bei. Damit werden die Meßdaten angeschlossener Temperatur- und Feuchtigkeitssensoren an einen Windows-PC übertragen. Das Programm und seine Installation ist in Kapitel 5 beschrieben.

```
Hi,

I wrote a program for all versions of Linux which can handle all
functionality of the USB devices you sell. Altough it has only be
tested with the USB-Temp (only got 2 of those), I'm confident it
works with the other devices as well. It includes a man-pages
explaining how to use it. Of course it is GPL it uses the Cleware
usb-access library for maximum compatibility.
http://www.vanheusden.com/clewarecontrol/


Folkert van Heusden
```

2. USBaccess.h

Die Datei USBaccess.h beinhaltet die Schnittstelle zu dem USB-Geräten der Cleware GmbH. Nach dem Einbinden der Datei können die Geräte geöffnet und gelesen werden. Die einzelnen Definitionen und Methoden sind identisch mit den Windows-Definitionen. Die detailierte Beschreibung der Methoden ist in der Windows API-Beschreibung zu finden.

```
// DLL class definitions for access to USB HID devices
//
// (C) 2004 Copyright Cleware GmbH
// All rights reserved
//

……

typedef int HANDLE ;
const int USBaccessVersion = 109 ;

class USBACCESS_API CUSBaccess {

  public:
    enum USBactions {     LEDs=0, EEwrite=1, EEread=2, Reset=3 } ;

    enum LED_IDs {        LED_0=0, LED_1=1, LED_2=2, LED_3=3 } ;

    enum SWITCH_IDs {
         SWITCH_0=0x10, SWITCH_1=0x11, SWITCH_2=0x12, SWITCH_3=0x13,
         SWITCH_4=0x14, SWITCH_5=0x15, SWITCH_6=0x16, SWITCH_7=0x17,
         SWITCH_8=0x18, SWITCH_9=0x19, SWITCH_10=0x1a, SWITCH_11=0x1b,
         SWITCH_12=0x1c, SWITCH_13=0x1d, SWITCH_14=0x1e, SWITCH_15=0x1f
         } ;
    enum USBtype_enum {   ILLEGAL_DEVICE=0,
                          LED_DEVICE=0x01,
                          WATCHDOG_DEVICE=0x05,
                          AUTORESET_DEVICE=0x06,
                          SWITCH1_DEVICE=0x08, SWITCH2_DEVICE=0x09,
                          SWITCH3_DEVICE=0x0a, SWITCH4_DEVICE=0x0c,
                          TEMPERATURE_DEVICE=0x10,
                          TEMPERATURE2_DEVICE=0x11,
                          TEMPERATURE5_DEVICE=0x15,
                          HUMIDITY1_DEVICE=0x20,
                          CONTACT00_DEVICE=0x30, CONTACT01_DEVICE=0x31,
                          …, CONTACT15_DEVICE=0x3f
                          } ;
```

```
  public:
    CUSBaccess() ;
    virtual ~CUSBaccess() ;   // maybe used as base class

    int       OpenCleware() ;  // returns number of Cleware devices
    int       CloseCleware() ; // close all Cleware devices
    int       Recover(int devNum) ; // try to find disconnected devices,
                                    // returns true if succeeded
    HANDLE    GetHandle(int deviceNo) ;
    int[1]    GetValue(int deviceNo, unsigned char *buf, int bufsize) ;
    int[1]    SetValue(int deviceNo, unsigned char *buf, int bufsize) ;
    int[1]    SetLED(int deviceNo, enum LED_IDs Led, int value) ;
                                  // value: 0=off 7=medium 15=highlight
    int[1]    SetSwitch(int deviceNo, enum SWITCH_IDs Switch, int On) ;
                                                      // On: 0=off, 1=on
    int[2]    GetSwitch(int deviceNo, enum SWITCH_IDs Switch) ;
    int[2]    GetSwitchConfig(int deviceNo, int *switchCnt,
                              int *buttonAvailable) ;
    int[1]    SetTempOffset(int deviceNo, double Soll, double Ist) ;
    int[1]    GetTemperature(int deviceNo, double *Temperature, int *time) ;
    int[1]    GetHumidity(int deviceNo, double *Humidity, int *timeID) ;
    int[1]    ResetDevice(int deviceNo) ;
    int[1]    StartDevice(int deviceNo) ;
    int       GetVersion(int deviceNo) ;
    int       GetUSBType(int deviceNo) ;
    int       GetSerialNumber(int deviceNo) ;
    int       GetDLLVersion() { return USBaccessVersion ; }
    int[2]    GetManualOnCount(int deviceNo) ;
                // returns how often switch is manually turned on
    int[2]    GetManualOnTime(int deviceNo) ;
                // returns how long (seconds) switch is manually turned on
    int[2]    GetOnlineOnCount(int deviceNo) ;
                // returns how often switch is turned on by USB command
    int[2]    GetOnlineOnTime(int deviceNo) ;  // returns how long (seconds)
                                    // switch is turned on by USB command
  } ;
```

[1] = Returnwert TRUE falls ok, FALSE im Fehlerfall
[2] = Returnwert im Fehlerfall –1

3. API-Beispiel

Das folgende einfache Beispiel zeigt die Anwendung der API zum Auslesen des Temperatursensors und zum Schalten eines Schalters unter C++. Wird das Programm ohne Argument aufgerufen, wird nach Temperatursensoren gesucht und in einer Schleife 10 Meßwerte ausgegeben. Mit einem Argument aufgerufen wird ein Schalter geschaltet (0=aus, 1=ein). Eine erweiterte Version des Bespiels befindet sich auf der CD.

```
...
#include "USBaccess.h"

int
main(int argc, char* argv[]) {
  CUSBaccess CWusb ;
  printf("Start USB Access Beispiel!\n") ;
  int USBcount = CWusb.OpenCleware() ;
  printf("OpenCleware fand %d Geräte\n", USBcount) ;

  for (int devID=0 ; devID < USBcount ; devID++) {
    if (argc == 2) { // nur den Schalter schalten
      if (CWusb.GetUSBType(devID) == CUSBaccess::SWITCH1_DEVICE) {
        if (argv[1][0] == '0')
          CWusb.SetSwitch(devID, CUSBaccess::SWITCH_0, 0) ;
        else if (argv[1][0] == '1')
          CWusb.SetSwitch(devID, CUSBaccess::SWITCH_0, 1) ;
        else
          printf("Falsches Argument für den Schalter\n") ;
        break ;
      }
      else
        continue ;    // die anderen Interessieren uns nicht
    }


    if ( CWusb.GetUSBType(devID) != CUSBaccess::TEMPERATURE_DEVICE &&
         CWusb.GetUSBType(devID) != CUSBaccess::TEMPERATURE2_DEVICE)
      continue ;       // nur Temperatur lesen!

    CWusb.ResetDevice(devID) ;
    Sleep(500) ;     // etwas warten

    // nun 10 Messwerte abfragen
    for (int cnt=0 ; cnt < 10 ; cnt++) {
      double temperatur ;
      int    zeit ;
      if (!CWusb.GetTemperature(devID, &temperatur, &zeit)) {
        printf("GetTemperature(%d) fehlgeschlagen\n", devID) ;
        break ;
      }
      printf("Messwert %lf Grad Celsius, Zeit = %d\n",
                                        temperatur, zeit) ;
      Sleep(1200) ;
    }
  }

  CWusb.CloseCleware() ;
  return 0;
}
```

4. Programm „send2cc“

Das Programm „send2cc“ ist ein Programm zur Übertragung von Informationen der Cleware USB-Geräte eines Linux-PCs an die graphische Windowsoberfläche ClewareControl. Es werden alle Cleware Temperatursensoren und Feuchtigkeitssensoren unterstützt. Die Sensoren werden in einem festen Zeitinterval von 2 Sekunden abgefragt

Das Programm „send2cc“ hat verschiedene Optionen:

-s servername	spezifiziert den ClewareControl Server
-p	spezifiziert den Port (default 54741)
-r	im Fehlerfall automatisch wiederholen
-d	gebe debuginfos aus
-v	gebe Versionsnumber aus
-h	zeige Optionen

Die Angabe des Servernames ist als einzige Option zwingend erforderlich. Die Angabe kann als Name des Serves oder als IP-Adresse erfolgen.

Tritt bei der Verbindung ein Fehler auf, wird das Programm normalerweise beendet. Eine endlose Wiederholung der Verbindungsaufnahme kann mit der Option –r erreicht werden. Dies ist sinnvoll, wenn die Applikation als Service im Hintergrund läuft. Zwischen den einzelnen Versuchen wird jeweils 10 Sekunden gewartet.

Soll das Programm als Service im Hintergrund laufen, ist unter SuSE-Linux das folgende Vorgehen möglich (als User root):

- Kopieren von send2cc nach /usr/sbin
- chmod 755 /usr/sbin/send2cc
- Kopieren des Skriptes rcSend2cc nach /etc/rc.d
- chmod 755 /etc/rc.d/ rcSend2cc
- cd /etc/rc.d/rc5.d (ggfs. auch rc3.d ...)
- ln -s ../ rcSend2cc S20send2cc
- ln -s ../ rcSend2cc K20send2cc

In dem Skipt „rcSend2cc“ muß noch der Name des Servers eingetragen werden, der die Meßwerte erhalten soll. Die entsprechende Zeile lautet „CCSERVERNAME=XXXX“. Der Name XXXX ist auszutauschen. Zum Testen kann das Programm dann in /etc/rc.d mit „rcSend2cc start“ gestartet werden. Die Applikation ClewareControl sollte nun die angeschlossenen Sensoren mit den Meßwerten anzeigen. Wie ClewareControl als Server konfiguriert wird, ist im Handbuch zu ClewareControl ausführlich beschrieben.

5. Programm „readtemp“

Während in dem Beispiel „Example“ der erste gefundene Temperatursensor ausgelesen wird, ist das Programm „readtemp“ in der Lage, alle angeschlossenen Temperatur- und Feuchtigkeits-sensoren auszuwerten.

6. Programm „USBswitch“

Das kleine Programm USBswitch steuert einen USB-Switch und ist auch in der Lage, eine Schalter aus einer Gruppe von mehreren Schaltern auszuwählen. Folgende Optionen sind möglich:

```
USBswitch [-n device] [0 | 1] [-d]
  -n device  use device with this serial number
  0 | 1      turns switch off(0) or on(1)
  -d         print debug infos
  -s         secure switching - wait and ask if switch was turned
  -r         read the current setting
  -v         print version
  -h         print command usage
```

7. Programm „USBwatch“

Das Programm USBwatch sendet alle 2 Sekunden ein Lebenszeichen an einen angeschlossenen USB-AutoReset oder USB-Watchdog.