

**. Application Interface
for Windows®**



Version 4.1.0
06.10.2011

Cleware GmbH
Nedderend 3
D-24876 Hollingstedt
Germany
www.cleware.de

Cleware USB devices API

1. Contents

2.The application interface in general.....	2
3.USBaccess.h.....	3
4.API Functions.....	6
5.API C++ Example.....	11
6.API C Example.....	13

2. The application interface in general

The devices build by Cleware GmbH may be controlled by programs build by the user or by products from third party vendors. For this purpose three files are supplied for interfacing on Windows® systems. The Linux API is described in the separate Linux documentation. The API files are located in the installation directory of Cleware.

- | | |
|------------------|-------------------------|
| 1. USBaccess.h | – Definition of the API |
| 2. USBaccess.lib | – Link information |
| 3. USBaccess.dll | – Executables |

ActiveX note: This interface will not be supported any longer. If these interfaces were needed the previous version 3.6 or earlier must be used. Please note that a new registry based interface is available (see ClewareControl documentation) and a web server will come up in one of next releases.

Cleware USB devices API

3. USBAccess.h

The file USBAccess.h contains the interface to the Cleware USB devices. After including this header file the devices could be opened and accessed.

```
const int USBAccessVersion = 410 ;

class USBACCESS_API CUSBAccess {
public:
    enum USBactions { LEDs=0, EEwrite=1, EEread=2, Reset=3,
                    KeepCalm=4, GetInfo=5,
                    StartMeasuring=6 // USB-Humidity, USB-Contact
                    Configure=7, // USB-IO16-V10, USB-Counter-V05
                    Display=8, // USB/Display
                    RunPoint=10 // USB-Encoder } ;

    enum LED_IDS { LED_0=0, LED_1=1, LED_2=2, LED_3=3 } ;
    enum SWITCH_IDS { SWITCH_0=0x10, // use this
                    SWITCH_1=0x11, ..., 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, HUMIDITY2_DEVICE=0x21,
                      CONTACT00_DEVICE=0x30, CONTACT01_DEVICE=0x31,
                      ..., CONTACT15_DEVICE=0x3f
                    } ;

private:
    class CUSBAccessBasic * X ;

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

    virtual int OpenCleware() ; // returns number of Cleware devices
    virtual int CloseCleware() ; // close all Cleware devices
    virtual HANDLE GetHandle(int deviceNo) ;
    virtual int Recover(int devNum) ;
    // try to find disconnected devices, return true if succeeded
    virtual int1 GetValue(int deviceNo, unsigned char *buf,
                        int bufsize) ;
    virtual int1 SetValue(int deviceNo, unsigned char *buf,
                        int bufsize) ;
    virtual int1 SetLED(int deviceNo, enum LED_IDS Led, int value) ;
    // value: 0=off 7=medium 15=highlight
    virtual int1 SetSwitch(int deviceNo, enum SWITCH_IDS Switch, int On) ;
    // On: 0=off, 1=on
    virtual int2 GetSwitch(int deviceNo, enum SWITCH_IDS Switch) ;
    virtual int2 GetSeqSwitch(int deviceNo, enum SWITCH_IDS Switch,
                            int seqNum) ; // On: 0=off, 1=on, -1=error
    virtual int2 GetSwitchConfig(int deviceNo, int *switchCount,
                                int *buttonAvailable) ;
    virtual int1 GetTemperature(int deviceNo) ;
    virtual int1 GetHumidity(int deviceNo) ;

    virtual int1 GetTemperature(int deviceNo, double *Temperature,
                                int *timeID) ;
```

Cleware USB devices API

```
virtual int1 GetHumidity(int deviceNo, double *Humidity,
                        int *timeID) ;
virtual int1 ResetDevice(int deviceNo) ;
virtual int1 StartDevice(int deviceNo) ;
virtual int1 CalmWatchdog(int deviceNo, int minutes,
                        int minutes2restart) ;

virtual int  GetVersion(int deviceNo) ;
virtual int  GetUSBType(int deviceNo) ;
virtual int  GetSerialNumber(int deviceNo) ;
virtual int  GetDLLVersion() { return USBAccessVersion ; }
virtual int2 GetManualOnCount(int deviceNo) ;
    // returns how often switch is manually turned on
virtual int2 GetManualOnTime(int deviceNo) ;
    // returns how long (seconds) switch is manually turned on
virtual int2 GetOnlineOnCount(int deviceNo) ;
    // returns how often switch is turned on by USB command
virtual int2 GetOnlineOnTime(int deviceNo) ;
    // returns how long (seconds) switch is turned on by USB command
virtual int2 GetMultiSwitch(int deviceNo, unsigned long int *mask,
                            unsigned long int *value, int seqNumber) ;
virtual int2 SetMultiSwitch(int deviceNo, unsigned long int value);
virtual int2 SetMultiConfig(int deviceNo, unsigned long int dir) ;
virtual int2 GetCounter(int deviceNo, int counter) ;
virtual int2 SyncDevice(int deviceNo, unsigned long int mask) ;
virtual int2 GetMultiConfig(int deviceNo) ;
virtual int2 IsAmpel(int deviceNo) ;
virtual int2 IsAlarm(int deviceNo) ;
virtual int2 IsCutter(int deviceNo) ;
virtual int2 IsSolidState(int deviceNo) ;
} ;

extern "C" {
    USBACCESS_API CUSBAccess * _stdcall USBAccessInitObject() ;
    USBACCESS_API void _stdcall USBAccessUnInitObject(CUSBAccess *) ;
} ;

...

// functional C interface (FCW = Function CleWare)

USBACCESS_API CUSBAccess * _stdcall FCWInitObject() ;
USBACCESS_API void _stdcall FCWUnInitObject(CUSBAccess *obj) ;
USBACCESS_API int _stdcall FCWOpenCleware(CUSBAccess *obj) ;
USBACCESS_API int _stdcall FCWCloseCleware(CUSBAccess *obj) ;
USBACCESS_API int _stdcall FCWRecover(CUSBAccess *obj,
                                     int deviceNo) ;

...

1 = Returns TRUE if ok, FALSE in case of an error
2 = Return -1 in case of an error
```

Cleware USB devices API

The Methods of the class `CUSBaccess` got the attribute “virtual” to enable the usage with Delphi. To use the class `USBaccess` with Delphi, the function `USBaccessInitObject()` must be called first, because the C++ class must be created in the C++ context.

An alternative is the use of simple functions without a class. This allows the usage of the language C. All methods of the class `USBaccess` are also available as functions. The function names always starts with the letters `FCW`.

Before using the functionals interface, the access must be prepared by calling the function `FCWInitObject()`. The returned value is used as the first argument in all other `FCW` functions. Before leaving the program, the function `FCWUnInitObject(obj)` will do the cleanup of the the environment. A sample for using the functional interface is shown in chapter 5.

Cleware USB devices API

4. API Functions

CUSBaccess *FCWInitObject() ;

Initialize the functional interface.

void FCWUnInitObject(CUSBaccess *obj) ;

Closes the functional interface.

int OpenCleware() ;

int FCWOpenCleware(CUSBaccess *obj) ;

Looks for Cleware USB devices and opens them. The number of found devices is returned.

int CloseCleware() ;

int FCWCloseCleware(CUSBaccess *obj) ;

Closes all open Cleware USB devices.

int Recover (int deviceNo) ;

int FCWRecover (CUSBaccess *obj, int deviceNo) ;

When reading from and writing to the USB device failed several times, the device could be automatically searched and setup to the initial state.

HANDLE GetHandle(int deviceNo) ;

HANDLE FCWGetHandle (CUSBaccess *obj, int deviceNo) ;

For later use.

int GetValue(int deviceNo, unsigned char *buf, int bufsize) ;

**int FCWGetValue(CUSBaccess *obj, int deviceNo, unsigned char *buf,
int bufsize) ;**

For later use. The return value is 0 in case of an error, else > 0.

int SetValue(int deviceNo, unsigned char *buf, int bufsize) ;

int FCWSetValue(CUSBaccess *obj, int deviceNo, unsigned char *buf, int bufsize) ;

For later use. The return value is 0 in case of an error, else > 0..

int SetLED(int deviceNo, enum LED_IDS Led, int value) ;

int FCWSetLED(CUSBaccess *obj, int deviceNo, enum LED_IDS Led, int value) ;

For later use. The return value is 0 in case of an error, else > 0.

Cleware USB devices API

float GetTemperature(int deviceNo) ;

float FCWDGetTemperature (CUSBaccess *obj, int deviceNo) ;

Simple and recommended function to get the current temperature. The returned value is the temperature or the value -200. to indicate an error.

float GetHumidity(int deviceNo) ;

float FCWDGetHumidity(CUSBaccess *obj, int deviceNo) ;

Simple and recommended function to get the current humidity. The returned value is the humidity or the value -200. to indicate an error.

int GetTemperature(int deviceNo, double *Temperature, int *timeID) ;

int FCWGetTemperature (CUSBaccess *obj, int deviceNo, double *Temperature, int *timeID) ;

Former request for the current temperature. Please use the simplified function call.

This function will set the temperature and timeID to the actual values. The returned time is based on a device internal time. This time is used to assure that two requests deliver different times. At least one second must separate adjacent calls. If several requests returns the same time, the device may be in trouble and a reset is necessary. This must be initiated by calling the “ResetDevice” function. The return value is 0 in case of an error, else > 0.

int GetHumidity(int deviceNo, double *Humidity, int *timeID) ;

int FCWGetHumidity (CUSBaccess *obj, int deviceNo, double * Humidity, int *timeID) ;

Former request for the current humidity. Please use the simplified function call.

This function will set the humidity and timeID to the actual values. The returned time is based on a device internal time. This time is used to assure that two requests deliver different times. At least two seconds must separate adjacent calls. The return value is 0 in case of an error, else > 0.

int SetSwitch(int deviceNo, enum SWITCH_IDs Switch, int On) ;

int FCWSetSwitch(CUSBaccess *obj, int deviceNo, enum SWITCH_IDs Switch, int On) ;

Turns an USB-Switch on or off. The argument “Switch” defines which switch to set. If the argument “On” is 1, the switch is turned on. If “On” is 0, the switch is turned off. The return value is 0 in case of an error, else > 0.

int GetSwitch(int deviceNo, enum SWITCH_IDs Switch) ;

int FCWGetSwitch(CUSBaccess *obj, int deviceNo, enum SWITCH_IDs Switch) ;

Get the current switch or contact state. The argument “Switch” defines which switch to get the status from. If the returned value is 1 if the switch is on and it is 0 otherwise. In case of an error, the return value is set to -1.

Cleware USB devices API

int GetSeqSwitch(int deviceNo, enum SWITCH_IDs Switch, int seqNum) ;

**int FCWGetSeqSwitch(CUSBAccess *obj, int deviceNo,
enum SWITCH_IDs Switch, int seqNum) ;**

Get the current switch state. The argument “Switch” defines which switch to get the status from. If the returned value is 1 if the switch is on and it is 0 otherwise. The seqNum argument should be 0. In case of an error, the return value is set to -1.

This command solves the problem that the USB data is buffered and maybe outdated. Using GetSeqSwitch will repeatedly get data from the buffer until the state represent the situation at the time of the call.

int ResetDevice(int deviceNo) ;

int FCWResetDevice(CUSBAccess *obj, int deviceNo) ;

Reset a Cleware device. The return value is 0 in case of an error, else > 0. If the device is a temperature or humidity sensor, the command causes a hardware reset of the sensor. Other devices will do a cold start when disconnected for a short time.

int StartDevice(int deviceNo) ;

int FCWStartDevice(CUSBAccess *obj, int deviceNo) ;

The USB-Humidity must get a start command to enter the measuring loop. This command must be send after a reset or when the commands SetValue oder GetValue were used. When StartDevice is called for other devices it will be ignored. In case of an error, the return value ist 0, otherwise != 0.

int CalmWatchdog(int deviceNo, int time1 , int time2) ;

int FCWCalmWatchdog(CUSBAccess *obj, int deviceNo, int time1 , int time2) ;

Send a life signal to an USB-Watdog. The time until the next life signal must be detected is supplied in minutes. If “time1” is -1 the device is activated immediately (USB-AutoReset). A value of 0 will deactivate the watchdog. The return value is 0 in case of an error, else > 0.

If the device is an USB-AutoReset “time2” defines the time distance for the secondary reset for cases the first reset will not restart the PC up to point where life signals will be send again. The value is defined in minutes and could be in the range 0 – 255. If the value is 0, the secondary reset is turned off.

If the device is an USB-WatchLight, the green light is turned on when the CalmWatchdog command is executed. The argument “time1” defines the offset in seconds until the red light is turned on. The yellow light is turned on after “time2” seconds.

int GetVersion(int deviceNo) ;

int FCWGetVersion(CUSBAccess *obj, int deviceNo) ;

Version number of this device.

Cleware USB devices API

int GetUSBType(int deviceNo) ;

int FCWGetUSBType(CUSBaccess *obj, int deviceNo) ;

Device type. Possible values are defined in the USBtype_enum, eg. SWITCH1_DEVICE or TEMPERATURE2_DEVICE.

int GetSerialNumber(int deviceNo) ;

int FCWGetSerialNumber(CUSBaccess *obj, int deviceNo) ;

Serial number of this device.

int GetDLLVersion() ;

int FCWGetDLLVersion() ;

Get the version of the DLL.

int GetManualOnCount(int deviceNo) ;

int FCWGetManualOnCount(CUSBaccess *obj, int deviceNo) ;

If the device is configured as an USB-Watchdog or USB-AutoReset, this counter reflects how often a reset was done by an USB command.

int GetManualOnTime(int deviceNo) ;

int FCWGetManualOnTime(CUSBaccess *obj, int deviceNo) ;

int GetOnlineOnTime(int deviceNo) ;

int FCWGetOnlineOnTime(CUSBaccess *obj, int deviceNo) ;

These commands are used with an USB-Switch equipped with a button for manual switching. In case of an error -1 will be returned.

int GetOnlineOnCount(int deviceNo) ;

int FCWGetOnlineOnCount(CUSBaccess *obj, int deviceNo) ;

Get the “turned on by USB command” counter (USB-Switch version 15 and higher). In case of an error -1 will be returned. If the device is configured as an USB-Watchdog or USB-AutoReset, this counter reflects how often a reset was done by missing the USB life signal.

int GetMultiSwitch(int deviceNo, unsigned long int *mask,

unsigned long int *value, int seqNumber) ;

int FCWGetMultiSwitch(CUSBaccess *obj, int deviceNo,...);

All input channels of the device USB-IO16 could be scanned with the command “GetMultiSwitch”. The argument “mask” indicates the channels which changes the values since the last call of GetMultiSwitch. Channel 0 is the least significant bit (LSB). The argument shows the current status of all channels. The argument “seqNum” should be set to 0. Please note: The USB-Contact is polled using GetSwitch(..).

Cleware USB devices API

int SetMultiSwitch(int deviceNo, unsigned long int value) ;
int FCWSetMultiSwitch(CUSBaccess *obj, int deviceNo, unsigned long int value) ;
All output channels of the USB-IO16 will be set with the command "SetMultiSwitch".
The argument "value" holds the new state of the channels. Channel 0 is the least significant bit (LSB).

int SetMultiConfig(int deviceNo, unsigned long int directions) ;
int FCWSetMultiConfig(CUSBaccess *obj, int deviceNo, unsigned long int dirs) ;
The configuration of the USB-IO16 is defined with the command SetMultiConfig. Every channel is assigned a bit in the direction argument. Channel 0 is the least significant bit (LSB). If the bit is 1 the channel is configured as an input, otherwise as an output.

int GetMultiConfig(int deviceNo) ;
int FCWGetMultiConfig(CUSBaccess *obj, int deviceNo) ;
The configuration of the USB-IO16 is returned with the command SetMultiConfig. Every channel is assigned a bit in the return value. Channel 0 is the least significant bit (LSB). If the bit is 1 the channel is configured as an input, otherwise as an output. A return value of -1 indicates an error.

int IsAmpel(int deviceNo) ;
int FCWIsAmpel(CUSBaccess *obj, int deviceNo) ;
The USB traffic light devices are registered as switch-devices. To find out, if the switch will turn a traffic light on or off, the function IsAmpel may be called. It returns a value greater than 0, the device is a traffic light.

int IsAlarm(int deviceNo) ;
int FCWIsAlarm(CUSBaccess *obj, int deviceNo) ;
The USB-Alarm buzzer is registered as an USB-Switch. This function tells if this switch turn an integrated buzzer on and off,

int IsCutter(int deviceNo) ;
int FCWIsCutter(CUSBaccess *obj, int deviceNo) ;
The USB-Cutter is registered as an USB-Switch. This function tells if this switch is an USB-Cutter.

int IsSolidState(int deviceNo) ;
int FCWIsSolidState(CUSBaccess *obj, int deviceNo) ;
When the switch is an electronic low voltage switch, this function return a value greater than 0..

int GetCounter(int deviceNo, enum COUNTER_IDS counter) ;
int FCWGetCounter(CUSBaccess *obj, int devNo, enum COUNTER_IDS countr);
The counter value of the USB-Counter is requested.

int SyncDevice(int deviceNo, unsigned long int mask) ;
int FCWSyncDevice(CUSBaccess *obj, int deviceNo, unsigned long int mask) ;
For internal purposes.

Cleware USB devices API

5. API C++ Example

The following simple example shows the usage of the API to read from a temperature sensor and to turn a switch with C++. If the program will be called without an argument, a temperature device will be searched and if found, the temperature will be read and printed 10 times. If the program will be called with an argument, a switch device is expected and turned on (1) or off (0). Any other argument will cause the program to read the current state of the switch and print it on the screen.

An special behaviour is implemented in case the program was renamed. If the name contains the string "on" or "off", a switch is assumed and this one is turned on or off. This is useful to control the switch with cmdline commands.

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

int
main(int argc, char* argv[]) {
    CUSBaccess CWusb ;

    printf("Start USB Access Beispiel!\n") ;

    int USBcount = CWusb.OpenCleware() ;
    printf("OpenCleware found %d devices\n", USBcount) ;

    int readTemperature = 1 ;
    int switchState = -1 ;
    if (argc >= 2) {
        readTemperature = 0 ;
        if (argv[1][0] == '0')
            switchState = 0 ;
        else if (argv[1][0] == '1')
            switchState = 1 ;
        // else ask for state
    }
    else { // check if name contains "on" or "off"
        for (char *pt=argv[0] ; *pt ; pt++) {
            if (*pt == 'o' || *pt == 'O') {
                if (pt[1] > 0 && (pt[1] == 'n' || pt[1] == 'N')) {
                    switchState = 1 ;
                    break ;
                }
                if (pt[1] > 0 && pt[2] > 0 && (pt[1] == 'f' || pt[1] == 'F')
                    && (pt[2] == 'f' || pt[2] == 'F')){
                    switchState = 0 ;
                    break ;
                }
            }
        }
    }
    if (switchState >= 0) // "on" or "off" was found
        readTemperature = 0 ;
}

if (readTemperature) {
    for (int devID=0 ; devID < USBcount ; devID++) {
        int devType = CWusb.GetUSBType(devID) ;
        if ( devType != CUSBaccess::TEMPERATURE_DEVICE &&
            devType != CUSBaccess::TEMPERATURE2_DEVICE)
            continue ; // read only temperatur!
```

Cleware USB devices API

```
CWusb.ResetDevice(devID) ;
Sleep(500) ; // wait a bit to settle after reset

// get 10 values
for (int cnt=0 ; cnt < 10 ; cnt++) {
    double temperatur ;
    int zeit ;
    if (!CWusb.GetTemperature(devID, &temperatur, &zeit)) {
        printf("GetTemperature(%d) failed\n", devID) ;
        break ;
    }
    printf("Measured %lf degrees Celsius, time = %d\n",
        temperatur, zeit) ;
    Sleep(1200) ;
}
}
else {
    for (int devID=0 ; devID < USBcount ; devID++) {
        if (CWusb.GetUSBType(devID) == CUSBaccess::SWITCH1_DEVICE) {
            if (switchState >= 0)
                CWusb.SetSwitch(devID, CUSBaccess::SWITCH_0, switchState) ;
            else {
                int cnt = CWusb.GetOnlineOnCount(devID) ;
                int state = CWusb.GetSwitch(devID, CUSBaccess::SWITCH_0) ;
                printf("Switch %d: count=%d, state = %d\n",
                    devID, cnt, state) ;
            }
            break ;
        }
    }
}

CWusb.CloseCleware() ;

return 0;
}
```

Some command samples:

copy Example.exe SwitchOn.exe	first copy
copy Example.exe SwitchOff.exe	second copy
Example 1	turn switch on
SwitchOff	turn switch off
SwitchOn	turn switch on again
Example ?	print the current switch setting
Example	print current temperature

A typical usage of “SwitchOn” and “SwitchOff” is signaling incoming mail with the rules of “Outlook”.

Cleware USB devices API

6. API C Example

```
// WatchService.cpp : Send a signal to all watchdog devices every second
// Options: -b run a thread in background
```

```
#include "stdio.h"
#include "windows.h"
#include "USBaccess.h"

#define maxWatchCnt 4

DWORD WINAPI
WatchdogLoop(LPVOID lpParameter) {
    int devCnt = 0 ;
    int watchIDs[maxWatchCnt] ;
    int watchCnt = 0 ;
    CUSBaccess *cw = 0 ;
    int i ;

    cw = FCWInitObject() ;
    if (cw != 0) ;
        devCnt = FCWOpenCleware(cw) ;

    for (i=0 ; i < devCnt ; i++) {
        enum FCWUSBtype_enum type = FCWGetUSBType(cw, i) ;
        if (type == WATCHDOG_DEVICE || type == AUTORESET_DEVICE)
            watchIDs[watchCnt++] = i ;
    }

    if (watchCnt <= 0) {
        printf("no USB-Watchdog or USB-AutoReset devices found\n") ;
    }
    else {
        while (1) { // loop forever
            for (i=0 ; i < watchCnt ; i++)
                FCWCalmWatchdog(cw, watchIDs[i], 1, 0) ; // timeout 1 minute
            Sleep(1000) ; // 1000 ms
        }
    }

    return 0 ;
}

int
main(int argc, char* argv[]) {
    int DebugInfos = 0 ;
    int runInBackground = 0 ;
    char *progName = argv[0] ;
    int err = 0 ;

    for (argc--, argv++ ; argc > 0 ; argc--, argv++) {
        if (argv[0][0] == '-') {
            switch (argv[0][1]) {
                case 'd':
                case 'D':
                    DebugInfos = 1 ; // not used now
                    break ;
                case 'b':
                case 'B':
                    runInBackground = 1 ;
                    break ;
            }
        }
    }
}
```

Cleware USB devices API

```
if (runInBackground) {
    char *execStr = progName ;
    STARTUPINFO startupinfo ;
    PROCESS_INFORMATION processInfo ;
    ZeroMemory(&startupinfo, sizeof(startupinfo));
    startupinfo.cb = sizeof(startupinfo) ;
    startupinfo.dwFlags = 0 ;
    ZeroMemory(&processInfo, sizeof(processInfo));
    if (CreateProcess(0, execStr, 0, 0, FALSE,
        NORMAL_PRIORITY_CLASS,
        0, 0, &startupinfo, &processInfo) == 0) {
        err = GetLastError() ;
        printf("Datei %s: Fehler beim öffnen (%d)", execStr, err) ;
    }
}
else
    WatchdogLoop(0) ;

return err ;
}

}
```