

Software Developer Kit (SDK)
für das
SOUNDLIGHT
USBDMX-ONE Interface

VisualBasic.Net
DELPHI

(C) SOUNDLIGHT 2010 * ALLE RECHTE VORBEHALTEN * KEIN TEIL DIESER ANLEITUNG DARF OHNE SCHRIFTLICHE ZUSTIMMUNG DES HERAUSGEBERS IN IRGEND EINER FORM REPRODUZIERT, VERVIELFÄLTIGT ODER KOMMERZIELL GENUTZT WERDEN. * WIR HALTEN ALLE ANGABEN DIESER ANLEITUNG FÜR VOLLSTÄNDIG UND ZUVERLÄSSIG. FÜR IRRTÜMER UND DRUCKFEHLER KÖNNEN WIR JEDOCH KEINE GEWÄHR ÜBERNEHMEN. ÄNDERUNGEN, DIE DER WEITERENTWICKLUNG ODER DEM TECHNISCHEN FORTSCHRITT DIENEN, BLEIBEN VORBEHALTEN. VOR INBETRIEBNAHME HAT DER ANWENDER DIE ZWECKMÄSSIGKEIT DES GERÄTES FÜR SEINEN GEPLANTEN EINSATZ ZU PRÜFEN. SOUNDLIGHT SCHLIESST INSBESONDERE JEDE HAFTUNG FÜR SCHÄDEN -SOWOHL AM GERÄT ALS AUCH FOLGESCHÄDEN- AUS, DIE DURCH DIE VERWENDUNG DER HIER BESCHRIEBENEN FUNKTIONEN ENTSTEHEN.

SOUNDLIGHT *The DMX Company* Bennigser Str. 1 D-30974 Wennigsen Tel. 05045-912 93-11

susbdmx_version

Gibt die Versionsnummer der DLL zurück

VB.NET:

`Private Declare Function susbdmx_version Lib "susbdmx.dll" () As Short`

DELPHI:

`function susbdmx_version(): SmallInt; stdcall; external 'susbdmx.dll';`

Aufruf Beispiel:

`dllversion = susdmx_version()`

Übergabeparameter:

keine

Rückgabeparameter:

Short dllversion Versionsnummer der **susbdmx.dll** - Datei

susbdmx_open

Funktion zum Öffnen des Interfaceports

VB.NET:

`Private Declare Function susbdmx_open Lib "susbdmx.dll" (ByVal InterfaceNumber As Short, ByVal handle As Integer) As Boolean`

DELPHI:

`function susbdmx_open(InterfaceNumber: SmallInt; var handle: Integer): Boolean stdcall; external 'susbdmx.dll';`

Aufruf Beispiel VB.NET:

`result = susbdmx_open(InterfaceNumber , handle)`

Übergabeparameter:

Short InterfaceNumber Laufende Interfacenummer.
(Wird laufend durchnummeriert Interface 1 = Nummer 0, Interface 2=Nummer 1, u.s.w.)

Rückgabeparameter:

Integer handle Handle des Interface über das im weiteren Verlauf, Daten gesendet oder empfangen werden.
Boolean result gibt True zurück wenn das Interface erfolgreich geöffnet wurde.

susbdmx_close

Funktion zum Schließen des Interfaceports

VB.NET:

`Private Declare Function susbdmx_close Lib "susbdmx.dll" (ByVal handle As Integer) As Boolean`

DELPHI:

`function susbdmx_close(handle: Integer): Boolean; stdcall; external 'susbdmx.dll';`

Aufruf Beispiel VB.NET:

```
result = susbdmx_close(handle)
```

Übergabeparameter:

Integer handle Das Interface-Handle das beim Öffnen des Interfaces ermittelt wurde.

Rückgabeparameter:

Boolean result gibt True zurück wenn das Interface erfolgreich geschlossen wurde.

susbdmx_device_id

Funktion zum Abfragen der Geräte-ID

VB.NET:

```
Private Declare Function susbdmx_device_id Lib "susbdmx.dll" (ByVal handle As Integer, ByRef typid As Short) As Boolean
```

DELPHI:

```
function susbdmx_device_id(handle: Integer; var typid: SmallInt): Boolean; stdcall; external 'susbdmx.dll';
```

Aufruf Beispiel VB.NET:

```
result = susbdmx_device_id(handle , typid)
```

Übergabeparameter:

Integer handle Interface-Handle das beim Öffnen des Interfaces ermittelt wurde.

Rückgabeparameter:

Boolean result gibt True zurück wenn die Device-ID gelesen werden konnte.
Short typid Gibt die Typenkennung des Interfaces zurück.

susbdmx_device_version

Funktion zum Abfragen der Geräte Versionsnummer

VB.NET:

```
Private Declare Function susbdmx_device_version Lib "susbdmx.dll" (ByVal handle As Integer, ByRef pversion As Short) As Boolean
```

DELPHI:

```
function susbdmx_device_version(handle: Integer; var pversion: SmallInt): Boolean; stdcall; external 'susbdmx.dll';
```

Aufruf Beispiel VB.NET:

```
result = susbdmx_device_version(handle , pversion)
```

Übergabeparameter:

Integer handle Interface-Handle das beim Öffnen des Interfaces ermittelt wurde.

Rückgabeparameter:

Boolean result gibt True zurück wenn die Geräte Versionsnummer gelesen werden konnte.
Short pversion Geräte Versionsnummer

susbdmx_serial_get

Funktion zum Abfragen der Geräte-Seriennummer

VB.NET

`Private Declare Function susbdmx_serial_get Lib "susbdmx.dll" (ByVal handle As Integer, ByVal serialnumber As String, ByVal size As short) As Boolean`

DELPHI:

`function susbdmx_serial_get(handle: Integer; serialnumber: String; size: SmallInt): Boolean; stdcall; external 'susbdmx.dll';`

Aufruf Beispiel VB.NET:

`result = susbdmx_serial_get(handle , serialnumber , size)`

Übergabeparameter:

<code>Integer</code>	handle	Interface-Handle das beim Öffnen des Interfaces ermittelt wurde.
<code>Short</code>	size	Gibt die Länge des Seriennummertextes an Standard = 18

Rückgabeparameter:

<code>Boolean</code>	result	gibt True zurück wenn die Seriennummer gelesen werden konnte.
<code>String</code>	serialnumber	Gibt die Seriennummer als Text (Unicode) zurück

susbdmx_tx

Funktion zum Senden eines DMX-Universes

VB.NET:

`Private Declare Function susbdmx_tx Lib "susbdmx.dll" (ByVal handle As Integer, ByVal universe As Byte, ByVal numslots As Short, ByVal buffer As Byte, ByVal config As Byte, ByVal time As Single, ByVal time_break As Single, ByVal time_mab As Single, ByVal ptimestamp As Short, ByVal pstatus As Byte) As Boolean`

DELPHI:

`function susbdmx_tx(handle: Integer; universe: Byte; numslots: SmallInt; var buffer: Byte; config: Byte; time, time_break, time_mab: Single; var ptimestamp: SmallInt; var pstatus: Byte): Boolean; stdcall; external 'susbdmx.dll';`

Aufruf Beispiel VB.NET:

`result = susbdmx_tx(handle , universe , numslots , buffer(0) , config , time , time_break , time_mab , ptimestamp , pstatus)`

Übergabeparameter:

<code>Integer</code>	handle	Interface-Handle das beim Öffnen des Interfaces ermittelt wurde.
<code>Byte</code>	universe	Universennummer 0 oder 1* (*für zukünftige Produkterweiterungen)
<code>Short</code>	numslots	Anzahl der zu sendenden DMX-Kanäle.
<code>Byte</code>	buffer	Erste Element eines Byte-Arrays mit der Größe (numslots)
<code>Byte</code>	config	Konfigurations ID Standard = 2.0
<code>Single</code>	time	Aktualisierungsgeschwindigkeit Standard = 0.1
<code>Single</code>	time_break	Zeit für break Standard = 0.0002
<code>Single</code>	time_mab	Zeit für mab Standard = 0.002

Rückgabeparameter:

<code>Boolean</code>	result	Rückgabe ob das Universe erfolgreich gesendet werden konnte.
<code>Short</code>	ptimestamp	Rückgabe Zeitstempel

susbdmx_rx

Funktion zum Empfangen einen DMX-Univers

VB.NET:

`Private Declare Function susbdmx_rx Lib "susbdmx.dll" (ByVal handle As Integer, ByVal universe As Byte, ByVal slots_set As Short, ByRef buffer As Byte, ByVal timeout As Single, ByVal timeout_rx As Single, ByRef pslots_get As Byte, ByRef ptimestamp As Short, ByRef pstatus As Byte) As Boolean`

DELPHI:

`function susbdmx_rx(handle: Integer; universe: Byte; slots_set: SmallInt; var buffer: Byte; timeout: Single; timeout_rx: Single; var pslots_get: Byte; var ptimestamp: SmallInt; var pstatus: Byte): Boolean; stdcall; external 'susbdmx.dll';`

Aufruf Beispiel VB.NET:

`result = susbdmx_rx(handle, universe, slots_set, buffer(0), timeout, timeout_rx, pslots_get, ptimestamp, pstatus)`

Übergabeparameter:

Integer	handle	Interface-Handle das beim Öffnen des Interfaces ermittelt wurde.
Byte	universe	Universennummer 0 oder 1* (*für zukünftige Produkterweiterungen)
Short	slots_set	Anzahl der zu empfangenen DMX-Kanäle.
Byte	buffer	Erste Element eines Byte-Arrays mit der Größe (slots_set)
Single	timeout	Wartezeit für den Empfang eines kompletten Frame in Sekunden. Standard = 1.1
Single	timeout_rx	Wartezeit für den Empfang zwischen zwei DMX-Kanälen. Standard = 0.0

Rückgabeparameter:

Boolean	result	Rückgabe ob das Universe erfolgreich gesendet werden konnte.
Byte	pslots_get	Anzahl der aktuell zurückgegebenen DMX-Kanäle
Short	ptimestamp	Rückgabe Zeitstempel
Byte	pstatus	Rückgabe Status

susbdmx_id_led_set

Funktion zum Setzen des Zustandes der Interface LED.

VB.NET:

`Private Declare Function susbdmx_id_led_set Lib "susbdmx.dll" (ByVal handle As Integer, ByVal id As Byte) As Boolean`

DELPHI:

`function susbdmx_id_led_set(handle: Integer; id: Byte): Boolean; stdcall; external 'susbdmx.dll';`

Aufruf Beispiel VB.NET:

`result = susbdmx_id_led_set(handle, id)`

Übergabeparameter:

Integer	handle	Interface-Handle das beim Öffnen des Interfaces ermittelt wurde.
Byte	id	Zustandsnummer für die LED.

Rückgabeparameter:

Boolean	result	Rückgabe ob das Kommando erfolgreich ausgeführt werden konnte.
---------	--------	--

susbdmx_id_led_get

Funktion zum Lesen des Zustandes der Interface LED

VB.NET:

```
Private Declare Function susbdmx_id_led_get Lib "susbdmx.dll" (ByVal handle As Integer, ByRef id As Byte) As Boolean
```

DELPHI:

```
function susbdmx_id_led_get(handle: Integer; var id: Byte): Boolean; stdcall; external 'susbdmx.dll';
```

Aufruf Beispiel VB.NET:

```
result = susbdmx_id_led_get( handle , id)
```

Übergabeparameter:

Integer	handle	Interface-Handle das beim Öffnen des Interfaces ermittelt wurde.
----------------	--------	--

Rückgabeparameter:

Boolean	result	Rückgabe ob das Kommando erfolgreich ausgeführt werden konnte.
Byte	id	Aktuelle Zustandsnummer der LED.

PROGRAMMIERBEISPIELE in VB.NET

So wird ein DMX-Universer versendet

```
Private Declare Function susbdmx_open Lib "susbdmx.dll" (ByVal InterfaceNumber As Short, ByVal handle As Integer)
As Boolean
Private Declare Function susbdmx_close Lib "susbdmx.dll" (ByVal handle As Integer) As Boolean
Private Declare Function susbdmx_tx Lib "susbdmx.dll" (ByVal handle As Integer, ByVal universe As Byte, ByVal
numslots As Short, ByVal buffer As Byte, ByVal config As Byte, ByVal time As Single, ByVal time_break As Single, ByVal
time_mab As Single, ByVal ptimestamp As Short, ByVal pstatus As Byte) As Boolean

Private Sub OpenInterfaceAndSendOneFrame()
    '// Interface öffnen
    Dim handle as integer = 0
    Dim result as boolean = susbdmx_open(0,hwnd)

    If result then
        '// Wenn öffnen erfolgreich dann einen DMX-Datenpuffer mit 513 Bytes anlegen
        Dim dmxdata(513) as byte

        '// DMX Startbyte auf 0 setzen. (Norm)
        dmxdata(0)=0

        '// Zur Demonstration die DMX-Kanäle 1-3 mit Werten füllen.
        dmxdata(1)=255 : dmxdata(2)=255 : dmxdata(3)=128

        '// Rückgabewariablen definieren
        Dim ptimestamp as short=0
        Dim pstatus as byte=0

        '// DMX-Daten einmalig versenden.
        Call susbdmx_tx( handle , 0 , 513 , dmxdata(0) , 2 , 0.1! , 0.0002! , 0.002! , ptimestamp , pstatus)

        '// Interface wieder schließen
        result = susbdmx_close(handle)
    End If
End Sub
```

So wird ein DMX-Universer gelesen

```
Private Declare Function susbdmx_open Lib "susbdmx.dll" (ByVal InterfaceNumber As Short, ByVal handle As Integer)
As Boolean
Private Declare Function susbdmx_close Lib "susbdmx.dll" (ByVal handle As Integer) As Boolean
Private Declare Function susbdmx_rx Lib "susbdmx.dll" (ByVal handle As Integer, ByVal universe As Byte, ByVal
slots_set As Short, ByVal buf fer As Byte, ByVal timeout As Single, ByVal timeout_rx As Single, ByVal pslots_get As
Byte, ByVal ptimestamp As Short, ByVal pstatus As Byte) As Boolean

Private Sub OpenInterfaceAndReadOneFrame()
    '// Interface öffnen
    Dim handle as integer = 0
    Dim result as boolean = susbdmx_open(0,handle)

    If result then
        '// Wenn öffnen erfolgreich dann einen DMX-Datenpuffer mit 513 Bytes anlegen
        Dim dmxdata(513) as byte

        '// Rückgabewariablen definieren
```

```
Dim slots_get as byte
Dim ptimestamp as short=0
Dim pstatus as byte=0
```

```
'// DMX-Daten einmalig lesen incl. DMX-Startbyte
```

```
result = susbdmx_rx( handle , 0 , 513 , dmxdata(0) , 1.1! , 0.0! , slots_get , ptimestamp , pstatus)
```

```
'// Interface wieder schließen
```

```
result = susbdmx_close(handle)
```

```
End If
```

```
End Sub
```

DLL-AUFRUFE in C++

```
/*
 * SUSBDMX.H -- include file for SOUNDLIGHT  SUSBDMX.DLL
 * to communicate with Soundlight USBDMX-ONE usb to dmx512 interface.
 *
 */

#ifndef SUSBDMX_H
#define SUSBDMX_H

// define the dll space this file is used in
#ifdef SUSBDMX_DLL_EXPORT
#define SUSBDMX_TYPE __declspec(dllexport) __stdcall
#else
#define SUSBDMX_TYPE __declspec(dllimport) __stdcall
#endif

/*
 * DLL version
 */
static const USHORT SUSBDMX_DLL_VERSION = 0x0404;

/*
 * MACRO to verify dll version
 */
#define SUSBDMX_DLL_VERSION_CHECK() (susbdmx_version() >= SUSBDMX_DLL_VERSION)

/* *****
 * functions defined in the susbdmx.dll *
***** */

/*
 * susbdmx_version(): returns the version number (16bit, 4 digits BCD)
 * Current version is SUSBDMX_DLL_VERSION. Use the Macro
 * SUSBDMX_DLL_VERSION_CHECK() compare dll's and header files version.
 */
USHORT SUSBDMX_TYPE susbdmx_version();

/*
 * susbdmx_open(): open device number <device>, where 0 is the first
 * and unlimited devices are supported. The function returns FALSE
 * if <device> is not supported. Use returned handle h to access the
 * device later on. One device can be opened an unlimited number of times.
 */
BOOL SUSBDMX_TYPE susbdmx_open(USHORT device, PHANDLE h);

/*
 * susbdmx_close(): close the device identified by the given handle.
 */
BOOL SUSBDMX_TYPE susbdmx_close(HANDLE h);

/*
 * susbdmx_device_id(): read the device id of the device
 */
BOOL SUSBDMX_TYPE susbdmx_device_id(HANDLE h, PUSHORT pid);

/*
 * susbdmx_is_XXX(): identify the device identified by the given handle.
 * Each function returns TRUE if the device matches.
 */
BOOL SUSBDMX_TYPE susbdmx_is_usbdmxone(HANDLE h);

/*
 * susbdmx_product_get(): read the product string from the device.
 */
```

```

* size specifies the maximum size of the buffer pointed to by <string>
* (unit bytes).
*/
BOOL          SUSBDMX_TYPE    susbdmx_product_get(HANDLE h, PWCHAR string, USHORT size);

/*
* susbdmx_serial_get(): read the serial number from the device.
* size specifies the maximum size of the buffer pointed to by <string>
* (unit bytes).
*/
BOOL          SUSBDMX_TYPE    susbdmx_serial_get(HANDLE h, PWCHAR string, USHORT size);

/*
* susbdmx_device_version(): Read the the device version of a device.
* the device version is one of the values within the USBs configuration
* descriptor (BcdDevice). pversion is only valid if the function returns
* TRUE.
*/
BOOL          SUSBDMX_TYPE    susbdmx_device_version(HANDLE h, PUSHORT pversion);

/*
* susbdmx_tx(): transmitt a frame using the new protocol on bulk endpoints
*
* INPUTs:      h          - handle to the device, returned by usbdmx_open()
*              universe   - addressed universe
*              slots      - number of bytes to be transmitted, as well as sizeof(buffer)
*                          for DMX512: buffer[0] == startcode, slots <= 513
*              buffer     - data to be transmitted, !!! sizeof(buffer) >= slots !!!
*              config     - configuration of the transmitter, see below for possible values
*              time       - time value in s, depending on config, either timeout or delay
*              time_break - break time in s (can be zero, to not transmitt a break)
*              time_mab   - Mark-after-Break time (can be zero)
* OUTPUTs:     ptimestamp - timestamp of this frame in ms, does overrun
*              pstatus    - status of this transmission, see below for possible values
*/
BOOL          SUSBDMX_TYPE    susbdmx_tx(    IN HANDLE h, IN UCHAR universe, IN USHORT slots,
                                             IN PUCCHAR buffer, IN UCHAR config, IN FLOAT time,
                                             IN FLOAT time_break, IN FLOAT time_mab,
                                             OUT PUSHORT ptimestamp, OUT PUCCHAR pstatus);

/*
* values of config (to be ored together)
*/
#define SUSBDMX_BULK_CONFIG_DELAY      (0x01) // delay frame by time
#define SUSBDMX_BULK_CONFIG_BLOCK     (0x02) // block while frame is not transmitting
                                         (timeout given by time)
#define SUSBDMX_BULK_CONFIG_RX        (0x04) // switch to RX after having transmitted this frame
#define SUSBDMX_BULK_CONFIG_NORETX    (0x08) // do not retransmit this frame
#define SUSBDMX_BULK_CONFIG_TXIRQ     (0x40) // send data using transmitter IRQ instead of timer

/*
* susbdmx_rx(): receive a frame using the new protocol on bulk endpoints
*
* INPUTs:      h          - handle to the device, returned by usbdmx_open()
*              universe   - addressed universe
*              slots_set  - number of bytes to receive, as well as sizeof(buffer)
*                          for DMX512: buffer[0] == startcode, slots_set <= 513
*              buffer     - data to be transmitted, !!! sizeof(buffer) >= slots !!!
*              timeout    - timeout for receiving the total frame in s,
*              timeout_rx - timeout between two slots used to detect premature end of frames
* OUTPUTs:     pslots_get - number of slots actually received, *pslots_get <= slots_set
*              ptimestamp - timestamp of this frame in ms, does overrun
*              pstatus    - status of the reception, see below for possible values
*/
BOOL          SUSBDMX_TYPE    susbdmx_rx(IN HANDLE h, IN UCHAR universe, IN USHORT slots_set,
                                         IN PUCCHAR buffer, IN FLOAT timeout, IN FLOAT timeout_rx,

```

```
OUT PUSHORT pslots_get, OUT PUSHORT ptimestamp, OUT PUCHAR pstatus);
```

```
/*
 * values of *pstatus
 */
#define SUSBDMX_BULK_STATUS_OK                (0x00)
#define SUSBDMX_BULK_STATUS_TIMEOUT          (0x01) // request timed out
#define SUSBDMX_BULK_STATUS_TX_START_FAILED (0x02) // delayed start failed
#define SUSBDMX_BULK_STATUS_UNIVERSE_WRONG  (0x04) // wrong universe addressed\tabularnewline
#define SUSBDMX_BULK_STATUS_RX_OLD_FRAME    (0x10) // old frame not read
#define SUSBDMX_BULK_STATUS_RX_TIMEOUT      (0x20) // receiver finished with timeout (ored with others)
#define SUSBDMX_BULK_STATUS_RX_NO_BREAK    (0x40) // frame without break received (ored with others)
#define SUSBDMX_BULK_STATUS_RX_FRAMEERROR  (0x80) // frame finished with frame error (ored with
others)

/*
 * macro to check, if the return status is ok
 */
#define SUSBDMX_BULK_STATUS_IS_OK(s) (s == SUSBDMX_BULK_STATUS_OK)

/*
 * susbdmx_id_led_XXX(): get/set the "id-led", the way the TX-led is handled:
 * special value: see below
 * other:      the blue led blinks the given number of times and then pauses
 */
BOOL      SUSBDMX_TYPE  susbdmx_id_led_set(HANDLE h, UCHAR id);
BOOL      SUSBDMX_TYPE  susbdmx_id_led_get(HANDLE h, PUCHAR id);

/*
 * special values of id
 */
#define SUSBDMX_ID_LED_USB          (0xff) // display the USB status: blink with 2Hz on USB transactions
#define SUSBDMX_ID_LED_USB_RX      (0xfe) // display USB and receiver status. the LED blinks red if not valid
dmx signal in received

#endif // SUSBDMX_H
```