USB Light Module Control API

DEVELOPER'S GUIDE

WERMA SIGNALTECHNIK

Revision 1.2
Dec, 04. 2006

Table Of Content

1. Introduction

The USB control element provides connectivity for a signal tower to the PC via USB.
The user controls the USB module by using a specified dynamic link library (DLL) file found on the CD provided with the USB element. The DLL contains several functions to set and retrieve the light module status.
The control over the light module is generally done using custom user application, the control over the module from the user application is done using a specific DLL which share all commonly needed functions.
The DLL library is a standard windows functions library which can be easily used with the most common application development tools such as Visual Basic, Visual C++ etc.

2. ULMC Interface Reference

2.1 Related files and executables

2.1.1 Runtime files

The DLL Library should reside in a file located in a the same folder with user project .EXE file. This file communicates with the USB module.

SLMA.DLL	Contains methods which provide control over the USB module

2.2 ULMC low level API

2.2.1 SetUsbDevice

STDMETHOD(SetUsbDevice)();

SetUsbDevice initialize all the available devices currently connected to the USB, This function also retrieves the devices number currently found in the system.

Parameters

None

Return Value

The amount of devices currently found in the system. When no devices found, zero will be returned.

Comments

This function is essential to start activating the devices

Compatibility

Drivers Release 1.0 and higher.

2.2.2 SetIOStatus

STDMETHOD(SetIOStatus)([in] int DevIdx, [in] char IOMask);

SetIOStatus sets the status for all IO's currently install in the configuration

Parameters

DevIdx

Index to the desire device currently available in the system. The first index is 1.

IOMask

Specifies the lights status mask of all the IO's. The value 1 at a given position in the mask indicates that the IO is set to ON state, the value 0 indicates the OFF state. This parameter can be a combination of the following IO port indexes.

Macro	Value	Description
NOIO	0	No port enable
IO1	1	IO #1 enable
IO2	2	IO #2 enable
IO3	4	IO #3 enable
IO4	8	IO #4 enable
IO5	16	IO #5 enable

Return Value

Completion code. 0 if successful, nonzero if not.

Comments

none

Compatibility

Drivers Release 1.0 and higher.

2.2.3 SetSingleIO

STDMETHOD(SetSingleIO)([in] int DevIdx, [in] char IOIdx, [in] bool Status);

SetSinlgeIO sets the status of a single IO

Parameters

DevIdx
Index to the desire device currently available in the system. The first index is 1.

IOIdx
Specifies the index of current desire IO. This value range between 1 to 5.

Status
Specifies the desire status for the given IO.
true – IO ON
false – IO OFF

Return Value

Completion code. 0 if successful, nonzero if not.

Comments

none

Compatibility

Drivers Release 1.0 and higher.

2.2.4 GetDeviceSerial

STDMETHOD(GetDeviceSerial)([in] int DevIdx, [out] long * TypeCode, [out] long * BatchCode, [out] long * SerialCode);

GetDeviceSerial returns the specified device serial code. This code is unique and can be use to distinguish between various device currently install in the system.

Parameters

DevIdx
: Index to the desire device currently available in the system. The first index is 1.

TypeCode
: Pointer to a variable that will contain the device type code

BatchCode
: Pointer to a variable that will contain the device batch code

SerialCode
: Pointer to a variable that will contain the device serial code

Return Value

Completion code. 0 if successful, nonzero if not.

Comments

None.

Compatibility

Drivers Release 1.0 and higher.

2.2.5 GetFirmwareVersion

STDMETHOD(GetFirmwareVersion)([in] int DevIdx, [out] long * Major, [out] long * Minor, [out] long * Build);

GetFirmwareVersion returns the specified device firmware version.

Parameters

DevIdx
: Index to the desire device currently available in the system. The first index is 1.

Major
: Pointer to a variable that will contain the device major firmware version value

minor
: Pointer to a variable that will contain the device minor firmware version value

build
: Pointer to a variable that will contain the device build firmware version value

Return Value

Completion code. 0 if successful, nonzero if not.

Comments

None.

Compatibility

Drivers Release 1.0 and higher.

2.2.6 GetLibVersion

STDMETHOD(GetLibVersion)([out] long * Major, [out] long * Minor);

GetLibVersion returns the currently used library version

Parameters

Major
Pointer to a variable that will contain the library major version value

minor
Pointer to a variable that will contain the library minor version value

Return Value

Completion code. 0 if successful, nonzero if not.

Comments

None.

Compatibility

Drivers Release 1.0 and higher.

3. Frequently Asked Questions

Q: Does the devices ID refer in DevIdx parameter in the above functions remains fixed in every activation of the software?
A: Not necessary. In every new binding of the DLL there is new enumeration over the USB devices, Therefore the ID might change according to the system state. If you need to refer to a fixed device independent from its current device index you should check the devices serial number. The devices serial number is returned by GetDeviceSerial function.

Q: Does the call to SetUsbDevice function is obligatory in order to access the device?
A: Yes, The user must call the SetUsbDevice in the beginning of his application in order to use the above functions.

Q: Can I dynamically add USB devices while using the above control functions?
A: When a new USB Light Module device is connected the SetUsbDevice must be called in order to enumerate and control the new device.

Q: The USB module is recognized by the computer but fails if I turn on the first light!
A: Some laptop computers do not provide enough power on the USB port for the USB element to work properly. This problem may occur also on some standard PCs. In this case you should use a USB hub with an external power supply to connect the USB devices to the computer.

Q: Can I use multiple USB signal towers on one PC?
A: Basically there is no problem in doing so. But as mentioned above there may be some problems with the power supply because some of the USB ports of the PC may share the same power supply on the mainboard. In this case use an external USB hub. Please make sure that the power supply is able to provide 500mA supply current for each connected module.

Q: How important is the cable?
A: Because of the current consumption of the USB device of up to 450mA it is important to use a cable of good quality. Cables of USB 2.0 standard should be used although the USB device operates on USB 1.1 standard. If you need to use a longer cable than the one shipped with USB element make sure that it has a wire diameter of the power lines of at least 24AWG or better.

Q: The USB module is recognized by the computer but fails if I turn on the first light!
A: Some laptop computers do not provide enough power on the USB port for the USB element to work properly. This problem may occur also on some standard PCs. In this case you should use a USB hub with an external power supply to connect the USB devices to the computer.

Q: The USB module is still not working correctly!
A: Maybe the problem is with your motherboard chipset driver for USB.
Make sure you have the latest driver for your motherboard installed.