Programming the USB2DMX

by Jan Menzel[*]

3rd January 2002

Abstract

This document describes how the USB2DMX DMX512 interface for the USB-port by Lighting-Solutions[1] has to be programmed. It includes all information the software engineer has to supply to the operating systems USB-stack to talk to the USB2DMX.

1 Identifying

Lighting-Solutions has the Vendor ID 0x0ce1 (dec. 3297). The product ID of the USB2DMX is 0x0001. bDeviceClass [1, p. 197] is 0xff (vendor-specific device class), bDeviceSubClass is 0x00 and bDeviceProtocol 0x01.

The manufacturer string can be read as “Lighting-Solutions” and the product string as “USB2DMX”.

2 Configuring

The USB2DMX is bus powered. Therefor is has four different configurations to suite the needs and available power. If unconfigured, both, transmitter and receiver, are disabled. Configuration 1 has only the transmitter enabled, configuration 2 has transmitter and receiver enabled while configuration 3 has only the receiver enabled. Before firmware version 1.1 (see section 5 for details) configuration 3 is not available.

All configuration have bInferfaceClass [1, p. 202] = 0xff (vendor-specific class), bInferfaceSubClass = 0 and bInterfaceProtocol = 0xff (vendor-specific protocol).

In addition each configuration has its own string descriptor.

3 Programming

All information exchange with the USB2DMX is done using control transfer to EP0 (in and out) and Vendor specific requests [1]. As specified, the direction of data flow is determined from bit 7 in bmRequestType [1, p. 183]. Bits 5 and 6 has to be set to sign that this is a vendor specific request.

4 Requests

The requested function is determined from evaluating the bRequest [1, p. 183] field. 8 Requests are implemented:

[*]jan@lighting-solutions.de
[1]http://www.lighting-solutions.de

Name	bRequest
DMX_TX_MEM	0x04
DMX_TX_SLOTS	0x05
DMX_TX_STARTCODE	0x06
DMX_TX_FRAMES	0x07
DMX_RC_MEM	0x08
DMX_RC_SLOTS	0x09
DMX_RC_STARTCODE	0x0A
DMX_RC_FRAMES	0x0B

4.1 DMX_TX_MEM, DMX_RC_MEM

Access transmitter (DMX_TX_MEM) or receiver (DMX_RC_MEM) memory.

The USB2DMX sends back or expects wLength bytes[2]. The data is read from or written to memory with offset wIndex. wIndex = 0 is the first slot of the DMX512 frame. Boundaries are not checked by the USB2DMX. Writing slots behind 512 should be avoided. Reading slots larger then 512 will return unknown data.

wValue has to have the value 0x0000 by default. If wValue is set to 0x0001, the read or write is blocked until the current frame has been transmitted or received completely.

The blocking read/write feature is not available before firmware version 1.1 (see section 5 for details).

4.2 DMX_TX_SLOTS, DMX_RC_SLOTS

Get or set the number of slots to transmit (DMX_TX_SLOTS) or read the number of slots received in the last DMX512 frame (DMX_RC_SLOTS).

To set the number of slots to be transmitted send it as wValue with DMX_TX_SLOTS request without a data stage (wLength = 0). Setting the number of slots to be received will result in a error.

On reading the number of slots transmitted or last received send DMX_TX_SLOTS or DMX_RC_SLOTS request and expect the result in 2 bytes back. The lower byte is transmitted first.

The default is to transmit 512 slots per frame.

4.3 DMX_TX_STARTCODE, DMX_RC_STARTCODE

Get or set the transmitters or receivers start-code.

To set the transmitter or receiver start-code send is as wValue with DMX_TX_STARTCODE DMX_RC_STARTCODE request. Note the the receiver will only read frames with the set start-code.

To read the current start-code send DMX_TX_STARTCODE or DMX_RC_STARTCODE and expect the result as 1 byte back.

The default start-code is 0x00 for both transmitter and receiver.

4.4 DMX_TX_FRAMES, DMX_RC_FRAMES

Get the number of transmitted or received frames.

Both, transmitter and receiver, are having individual 32bit counters for counting the number of transmitted or received frames. The counters are cleared on power up and incremented on any successful transmission or reception of DMX512 frames. Note that the receiver does only count frames, with have the start-code equal to the set start-code (see section 4.3).

The frame counter can only be read read by sending DMX_TX_FRAMES or DMX_RC_FRAMES request and expecting the result as 4 bytes back. The lowest byte is again transmitted first.

Trying to set the frame counter will result in a error.

This counters are designed to give the programmer the possibility to check e.g. whether a new DMX512 frame has been successfully received. Sins the counter is 32bit the number is almost unique.

[2]one byte is treated as one DMX512 slot

5 Changes

With firmware version 1.1 (bcdDevice [1, p. 198] = 0x0101) configuration 3 (read only) and blocking read/write was added.

References

[1] Universal Serial Bus Specification, Revision 1.1, 23.09.1998, `http://www.usb.org`