dmxusbpro External for Max/MSP

Version 1.2

Written by Olaf Matthes.

Content

Overview

The goal of this document is to describe the features and possibilities that the dmxusbpro external adds to the Max/MSP environment. This document assumes some familiarity with Max from a user's standpoint.

The basic functionality of dmxusbpro is to allow to access the ENTTEC DMX USB Pro interface (http://www.enttec.com/dmxusb_pro.php) and to send or receive DMX data. It thus allows to control any kind of lighting equipment that has DMX512 capabilities.
Due to it's capability to receive DMX data, it can also be used to control Max/MSP with a light board or any other equipment that is capable of sending DMX data.

Thanks

The DMX USB Pro interface that was used while developing this software was donated by Jeremy Hochman from Element Labs, http://www.elementlabs.com/.

Development was also supported through some valuable input from the *Opendmxusb-users* mailing list and from ENTTEC.

The dmxusbpro External

Creating the Object

Technically speaking, the DMX USB Pro interface gets recognised by the operating system as an additional serial port. On Windows these ports can be accessed by specifying the COM port number (usually 3 and upwards), on Mac you have to use the full device name, something like `/dev/cu.usbserial-001`. An object created without any arguments looks like this:

Optinal creation arguments specify the device to open, the number of DMX channels to send and the refresh rate.
When no creation arguments are specified, no device will be opened on object creation. The left outlet will report received DMX data and the right outlet signals whether a device is opened or not.

Inlets and Outlets

As the above image shows, dmxusbpro has one inlet and two outlets. The inlet receives control messages discussed in the next section.

The outlets – from left to right – have the following functions:

- (list) Received DMX data: <startcode> <channel> <value>
- (int) State of connection (1 = connected, 0 = not connected).

The DMX data always contains the startcode (usually 0), followed by a channel number (1-512) and the corresponding value (0-255). Only channels that have changed their value get reported.

The right outlet indicates whether the external is connected with a widget or not. On some lethal communication errors the object closes the connection to the widget, so monitoring this outlet allows to detect some errors.

Supported Methods

The following methods are supported by the dmxusbpro external:

`open <device>`

Opens a device specified by name (i.e. COM3 on windows or /dev/cu.usbserial-001 on Mac) or by number (i.e. 3 for COM 3, works on Windows only).

To figure out the name of the device on Mac OS X, open the Network panel in the System preferences. The DMX USB Pro interface will be listed there as modem device. Another way is to use the terminal: type `cd /dev` followed by pressing enter and then `ls` to list the content of the /dev directory. Sroll a bit to find a device that starts with cu.usbserial-XXX.

WARNING: the device name on OS X depends on the USB port you are using! So if you plug the device from one USB port to another the device name will change and you have to edit your Max patch. Also please note that some USB ports on Mac computers don't seem to provide enough power to reliably power the DMX USB Pro interface. If you get mysterious kernel panics or crashes try another port and/or a higher quality USB cable.

On Windows your DMX USB Pro interface is one of the serial ports, usually COM3.

In version 1.1and above you can use the `getdevs` method described below to find connected interfaces!

On systems that have more than one DMX USB Pro interface there is no realiable way to find out which is which. So you have to open one and see which one of your interface is reacting. On Windows the interface that is plugged in first has the lowest serial port number. On Mac the serial port names depend on the USB plug you plug the interface in.
Newer ineterfaces have a serial number, so you can query the serial number to find out which is which.

`getdevs`

Available in version 1.1 only: Get a list of all available DMX USB Pro interfaces on your system printed to the Max window. In fact, this function finds all devices that look like DMX USB Pro interface (i.e. all devices that use the same FTDI USB – Serial converter chipset and the corresponding drivers).

`close`

Closes the device. A device has to be closed before a new one can be opened.

`send`

Set the object to send mode. The widget will start to transmit DMX frames. Please note that due to the electrical specifications of the USB DMX Pro Interface (i.e. the DMX input and output plugs are electrically connected and just provide mechanically different access to the same data lines) any incoming DMX data will be 'mixed' with the DMX data from the computer. This results in a corrupted DMX stream! ***Make sure no DMX data is connected to the input when using the interface to* send *data!***

`receive`

Set the object to receive mode. Received DMX data will be output through the object's left outlet. The format of the data is a list of three integers containing *startcode*, *channel number* and *channel value*. Only when the DMX value changed a channel will be output.
Since this output generates a lot of data on complex scene changes you should avoid connecting a `print` object directly to it since any graphics updates slows down processing significantly.

`getparams`

Get and display the widget's internal parameters like *refresh rate*, *break time* and *mark-after-break time*. This function should be used before setting any of these values.

`getserial`

Get and display the widget's serial number. Please note that early widgets didn't have a serial number and will be reported with -1.

`refresh <value>`

Set the refresh rate. The value (int) determines how many DMX frames the widget will send per second when in send mode. Possible values are 1 – 40, default is determined by the value that is currently set in the widget's firmware.

`breaktime <value>`

Sets the break time (int) for the DMX signal. Default is 192 us. The values actually used by the interface will be rounded to multiples of 10.67 us.

`mabtime <value>`

Sets the mark-after-break time (int) for the DMX signal. Default is 21 us. The values actually used by the interface will be rounded to multiples of 10.67 us.

(list) `<channel> <value> [<channel> <value> [<channel> <value>]]`

A list of integer pairs sets the values of the DMX channels. The first value determines the channel (1 – 512), the second one it's DMX value (0 – 255). A channel value of 0 can be used to set all channels to the same value.
To set more than one channel at the same time, up to 256 channel-value pairs can be combined.

`blackout <state>`

Sets all DMX channels temporarily to zero (i.e. "blackout") if state is nonzero. If state is zero the original DMX values will be sent again.

`channels <value>`

Determines the number of DMX channels the widget will send (1 – 512). Reducing this to only as many channels as are used reduces any unnecessary CPU load and data traffic and thus increases overall performance of your patch.

`startcode <value>`

Sets the DMX startcode (0 – 255, usually 0). Some values have special functions, like 23 for DMX text frames, 85 for ASC test packets and 207 for System Information Packets (SIP). These values should not be used since they are not intended to send the usual channel values.

`firmware [<filename>]`

Available in version 1.1 and above only: Updates the internal firmware of the widget with the file being specified. If no filename is given the user will be prompted with a file-open dialog. **Please note that this feature should only be used when no DMX signals are connected to the widget! Use at your own risk!**

Installation

System Requirements

This software is for use with Max/MSP 4.5 (and probably later). Separate versions are available for the Windows version and the Mac OS X version of Max/MSP. Max/MSP prior to version 4.5 or Max/MSP on Mac OS 9 is not supported.

The latest dmxusbpro release can always be found at http://www.nullmedium.de/dev/dmxusbpro/.

Installing dmxusbpro

The installation of dmxusbpro follows the standard Max/MSP ways. After extracting the archive, the object can be found in either the `build-mac` or `build-win` folders, depending on the operating system you're using.

To permanently install dmxusbpro, copy the object (dmxusbpro or dmxusbpro.mxe) from the corresponding build folder into the externals folder of your Max distribution (under 'Application Support' on Mac or 'Common Files' on Windows). Copy the help patch (dmxusbpro.help) into the max-help folder of your Max distribution.

Please make sure to install the latest driver from the FTDI website:

http://www.ftdichip.com/Drivers/VCP.htm

License

License Agreement

The Software is provided by Olaf Matthes (the Author) to you, a single user, for a license fee. This license gives you the royalty-free right to use components (external objects and patchers) of the Software in your own ORIGINAL WORKS (including but not limited to Max or Pd patchers, collectives, and stand-alone applications). You have the non-exclusive right to use, distribute, and license such ORIGINAL WORKS to third parties without payment of any further license fees, so long as a your terms of use for your ORIGINAL WORKS are sufficient to protect the copyright and license of the Software.
YOU MAY NOT SELL OR LEASE THE SOFTWARE, NOR MAY YOU TAKE A FEE OR COMMISSION FOR PROVIDING THE SOFTWARE TO ANOTHER PERSON, NOR MAY YOU INCLUDE THE SOFTWARE WITH OTHER SOFTWARE THAT IS SOLD FOR A FEE WITHOUT PRIOR WRITTEN PERMISSION FROM THE AUTHOR.

No Warranty

THE SOFTWARE IS PROVIDED “AS IS” AND WITHOUT WARRANTY OF ANY KIND.
The Author expressly disclaims all warranties, express or implied. No oral or written information or advice given by the Author shall create a warranty or in any way increase the scope of this warranty.
UNDER NO CIRCUMSTANCES, INCLUDING NEGLIGENCE, SHALL THE AUTHOR BE LIABLE FOR ANY INCIDENTAL, SPECIAL, OR CONSEQUENTIAL DAMAGES THAT RESULT FROM THE USE OR INABILITY TO USE THE SOFTWARE, EVEN IF THE AUTHOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.