Controlling Lighting Systems with DMX

Professional lighting is controlled with a protocol known as DMX. The object of DMX is to separate the controlling device from the circuitry that actually dims the instrument. This saves enormous expense, since heavy duty wiring no longer must be run from the lighting booth to the stage. In fact, the latest trend is to put the dimming circuitry in the lighting instruments themselves. This means lights can all be attached to a single feed which is always on. Lighting instruments are also developing new features, such as pan and tilt, color control, internal gobos[1], and even built-in lasers.

DMX control systems are now heavily computerized, providing a wide range of traditional lighting design functions such as crossfades between preset scenes and some modern tricks like chases and sweeps. These usually include MDI control of some sort, although it's not what we might hope. Scenes usually must be set up in the board, and all MIDI can do is trigger timed fades. Boards often have serial or even Ethernet control, but the actual control provided by that route is still limited. If we want to truly control the lights, we need to hack directly into the DMX data.

The hardware part of DMX is relatively simple. It is a serial data system based on a standard called RS 485. There is a transmitter at the control end, and all DMX dimmer packs and lighting instruments have receivers built-in. Each device has both an input and output connector for daisy-chain routing. The spec calls for 5 pin XLR connectors, although some devices use 3 pin microphone style connectors[2]. Since the system was designed by people used to working with power cables, the output connector is a female receptacle. The last device in the chain should be a "terminator". That is a plug with a 120 ohm resistor connected between pins 2 and 3. The cabling can be run as far as 300 meters.

The DMX transmitter can be part of a lighting board, or it may be a computer with interface box using a USB or network connector. Wireless interfaces are also beginning to appear. These receive data via UHF, Wi-Fi or IR and convert it to DMX cabled to the instruments. This greatly reduces the complexity of hoisted lighting installations. Wi-fi DMX can be controlled by practically anything, including iPhones.

The data format of DMX is even simpler-- after a break (data low) of 88 ms, a series of bytes are transmitted at 250 kbs. The first byte is a 0[3], called start code. This is followed by up to 512 bytes called "slots" or "channels". Dimmer packs and lighting instruments will have an address setting that tunes them to a particular channel. A device set to channel 10 will respond to the tenth byte sent[4]. Devices that need more data (such as

[1] A metal plate with holes cut in it to shape the light.
[2] The 2010 standard specifically forbids 3 pin connectors, but the inexpensive "DJ" models mostly use them.
[3] There are very rare exceptions for sending text or other specialized data.
[4] The first address is 1.

lights with tilt and pan) will watch a block of channels-- the address sets the first one. So if I have a set of four lights with 9 features, the starting addresses must be 1, 10, 19, and 28. Only enough bytes are sent to reach the highest address, so the fewer lights attached, the faster the system responds. Even so, the system can completely update 45 times a second.

Stand alone transmitters are available with USB or network input. Each seems to have its own input format, so you will have to do some manual reading and investigation to fix details.

USB to DMX

Enttec DMX USB Pro

The Enttec company seems to be one of the more popular interface suppliers, as their hardware costs in the hundreds rather than thousands of dollars. The USB pro box at $150 is the most economical I have seen. Their documentation is typical-- terse with obscure terminology. First it defines the control message format in a table:

Number of bytes	Description
1	Start of message, 0x7E
1	Message label
1	Data length LSB
1	Data length MSB
(Length)	Data bytes
1	End of message 0xE7

It then lists various types of message, to set transmission parameters[5], for instance. The one we are interested in is type 6: "Send DMX packet". Putting all of this together, we see the message to send 29 channels of data would be:

0x7E 6 30 0 [data] 0xE7

There are 30 bytes of data, including the 0 for the start code.

Figure 1 shows how to communicate with the Enttec box. This appears as a serial over USB device, so the serial object does the trick. Baud rate is not really used in USB communications, so the arguments to serial are not required. Note how the port list is updated. The print message to serial results in a port message with a list of names. These are split up and appended to the umenu. Serial defaults to the first item on the list. You can set this with arguments and forget the menu, but since the list is likely to change from time to time, it is best to see all of the options.

[5] The defaults are fine.

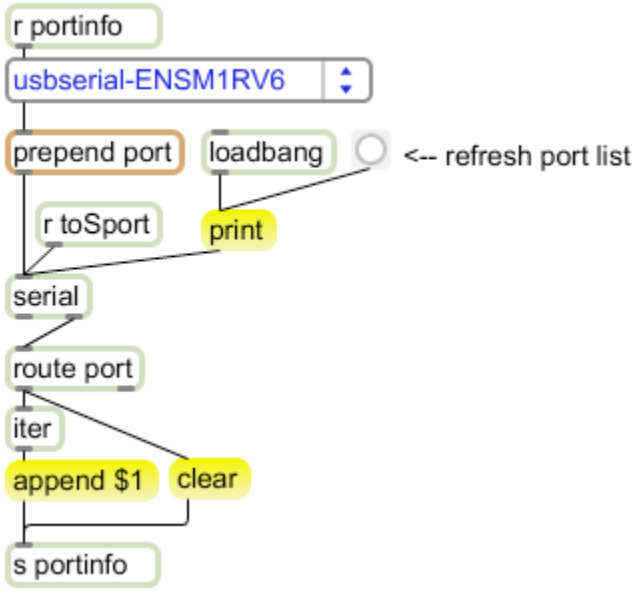

Figure 1.

Figure 2 shows the meat of the patch. Messages to update parameters are received by the trigger object. The messages will be formatted as lists of channel and value. The complete set of channel values is maintained in a coll. The trigger passes the list to the coll, then sends three distinct items to the serial object. The message 126 6 19 0 provides Enttec's header, the dump message to coll will provide the data, and the 231 is the end of message marker,

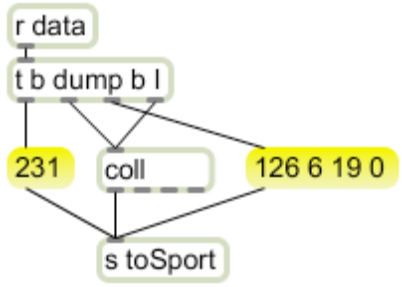

Figure 2.

Figure 3 shows a user interface that will provide the data. This patch controls two instruments, one on channel 1 and one on channel 10. Each has nine parameters.

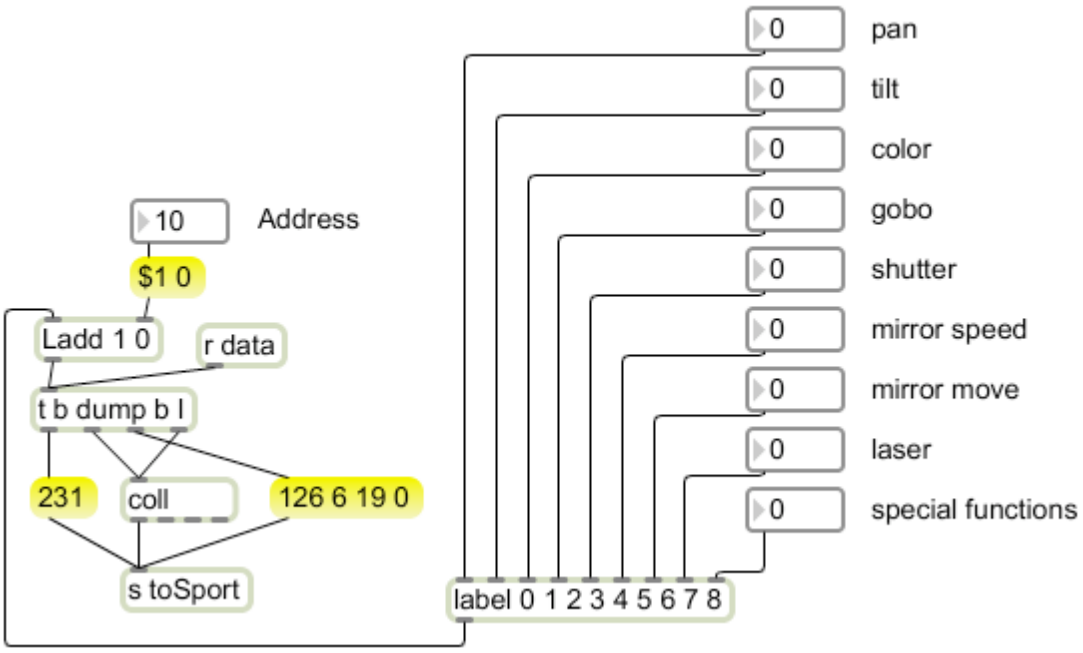

Figure 3.
The number boxes routed through Label generate the messages for each parameter of a lighting instrument. The Ladd will offset these to the starting address for each instrument.

Figure 4 shows typical contents of the coll in figure 2. I start with default settings, like pan and tilt centered and shutter open. Note that the start code is at address 0. Nothing in the patch should disturb that. (The range of the address number box is limited to 1 - 10)

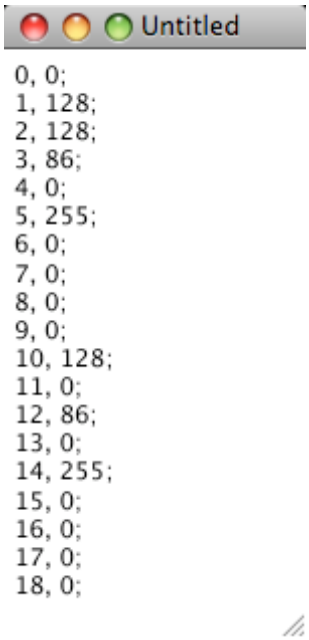

Figure 4.

Expanding this patch to control more instruments requires nothing more that adding items to the coll and changing the data length in the header[6].

It is easy to add mechanism to control light parameters. For instance, figure 5 will sweep the tilt of the second instrument:

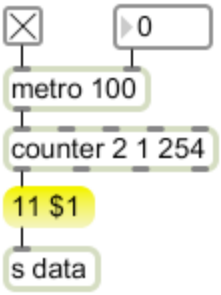

Figure 5.

ArtNet [notes only at this stage]

Artnet is a network protocol developed by a company called Artistic Licence (it's British). Unlike most theater equipment companies, they have published their protocol, and others are using it under the name openDMX. The Enttec ODE is a typical ArtNet converter box-- Ethernet in and DMX out. ArtNet uses the UDP protocol, so a control patch can be built around udpsend.

ArtNet expands the DMX vocabulary to include "nodes" and "universes". A node is a particular converter box with its own IP. A universe is a block of 512 DMX channels. ArtNet defines 255 universes. That's good, because with high tech lights requiring up to 30 DMX channels each, 512 doesn't seem like a big number any more. The universes are actually broken up into blocks of 16 subnets, so a universe designation might be 0,0 or 2,12.

ArtNet boxes can use either a closed network or the internet.. On a closed network the controller broadcasts all messages to either 2.255.255.255 or 10.255.255.255. On an internet connection the data is sent to a DHCP assigned IP for each node. The IP of an ArtNet box can be discovered by software that comes with the unit or via the Ethernet connection. The port address is 6454.

ArtNet is a full network protocol, with enough messages to manage hundreds of nodes in detail. For our purposes we can leave the network alone and use simple broadcast mode, or set up the network using the manufacturer's network utility software.

[6] The fact that the header wants length as LSB, MSB doesn't matter until the length reaches 256. That is coded as 1 1, followed by 2 1 and so on.

Message format

All messages start with the header Art-Net, which is 65 114 116 45 78 101 116 0 in ascii. The poll message is broadcast by the controller. Any nodes will respond with an information message:

Poll message	Data
Header	65 114 116 45 78 101 116 00
opcode	00 32 (poll)
Protocol version	00 14
Response	0 (reply to poll only)
Diagnostics	255 (all messages)

Reply message	Data
Header	65 114 116 45 78 101 116 00
opcode	00 33 (reply)
IP	aa bb cc dd
Further info	About 250 bytes

Data message	Data
Header	65 114 116 45 78 101 116 00
opcode	00 80 (DMX)
Protocol version	00 14
Sequence	0 (no sequencing)
Input	0
Universe	ssssuuuu*
Length MSB	
Length LSB	
DATA	Standard DMX data

*ssssuuuu -- subnet and universe packed into one byte, so universe 1,2 is 18.
Data must be an even number of bytes.

ArtNet Patch

All of these messages can be sent with udpsend, but there is a complication. Udpsend uses a simplified OSC protocol which ArtNet devices don't understand. Max doesn't let us compose plain UDP messages unless we work in Java. A third party external that manages this is aka.datagram by Masayuki Akamatsu. (Mac only I'm afraid.) Figure 6 shows a patch [untested] that uses this object to transmit the DMX data.

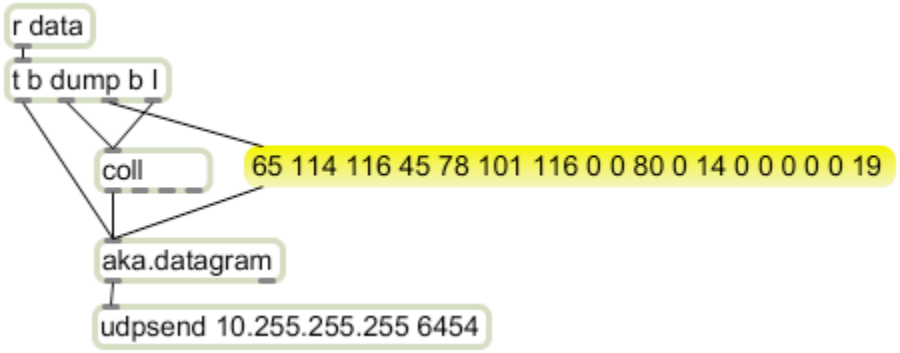
r data
t b dump b l
coll
65 114 116 45 78 101 116 0 0 80 0 14 0 0 0 0 0 19
aka.datagram
udpsend 10.255.255.255 6454

Figure 6.